Departamento de Engenharia de Computação e Automação Industrial
Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas

EA978

Sistemas de Informações Gráficas

Síntese de Imagens: Uma Introdução ao Mundo de Desenho e Pintura dos Sistemas Digitais.

Profa. Wu, Shin - Ting

Março de 2009

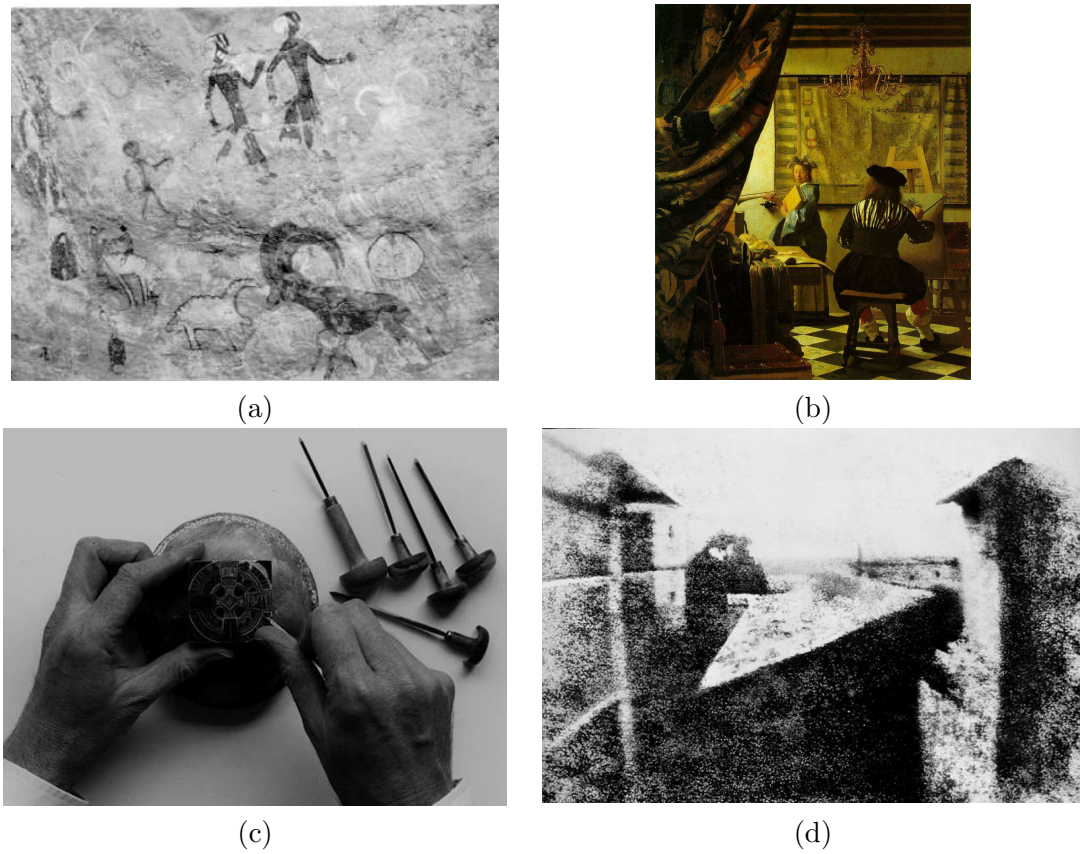
Conteúdo

Capítulo 1

Introdução

Como diz o provérbio chinês: "Uma imagem vale mais do que mil palavras", a comunicação visual através de imagens sempre foi uma forma eficaz para transmitir conceitos diretamente relacionados com a realidade que nos cerca. Desde o tempo das cavernas, os homens já faziam Embora ninguém saiba dizer ao certo o que significavam as pinturas rupestres, conjectura-se que eles as utilizavam como uma forma alternativa de comunicação. Através das pinturas retratavam cenas do cotidiano e expressavam seus sentimentos, suas idéias ou algumas mensagens (Figura 1.1.(a)).

Mesmo com o desenvolvimento da escrita alfabética, imagens nunca perderam sua importância na representação e expressão visual. A história de imagens evoluiu das pinturas rupestres para pinturas com uso de pigmentos e tintas na Idade Média (Figura 1.1.(b)). Pensando na impressão e reprodução de desenhos e pinturas "em massa", foram desenvolvidas várias técnicas de gravura durante a Revolução Industrial. O artista retira as partes de uma superfície plana, denominada matriz, de forma que ela represente uma pintura ou um desenho. Depois a superfície recebe uma camada de tinta e, com uma prensa, a tinta é transferida para um papel (Figura 1.1.(c)). A constante busca pelo maior realismo levou os grandes artistas renascentistas a desenvolverem conceitos com rigor científico para reproduzirem a realidade tal como a observavam. Um dos conceitos é o de câmara escura, em inglês *pin-hole camera*, que permite a criação de imagens por meio de exposição luminosa. A câmara escura foi utilizada por muitos artistas para esboçar pinturas e o seu princípio é utilizado até hoje. A primeira fotografia reconhecida, no entanto, remonta ao ano de 1826 e é atribuída ao francês Joseph Nicéphore Niépce, quando ele conseguiu produzir uma imagem numa placa de estanho coberta com um derivado de petróleo fotossensível chamado Be-



(a) (b)

(c) (d)

Figura 1.1: Evolução das técnicas: (a) pinturas rupestres; (b) pintura em cavalete (Fonte: http://www.ibiblio.org/wm/paint/auth/vermeer/art-painting/art-painting.jpg); (c) gravura (Fonte: http://www.woodengravers.co.uk/process.html) e (d) primeira fotografia (Fonte: http://pt.wikipedia.org/wiki/Fotografia).

tume da Judéia (Figura 1.1.(d)). A partir de então, avanço tecnológico tem sistematicamente melhorado a qualidade das imagens fotográficas, reduzindo o seu tempo de produção e o seu custo.

Com o aumento quase exponencial na capacidade de processamento, os computadores tem se tornado um instrumento de grande valia em todas as tarefas relacionadas com imagens. Imagens são geeradas, com auxílio de compuadores, em projetos de engenharia e arquitetônicos, em visualização de informação, em simulações para testes de hipóteses, em treinamentos, em análise do comportamento de um sistema a baixo custo, em manifestações artísticas, em propaganda e publicidade, e em diversos tipos de entretenimento. Mediado por computadores, imagens são utilizadas para diagnosticar doenças, para prever tempo, para monitorar um ambiente, para explorar lugares inacessíveis, para inspecionar a qualidade dos produtos, para preservar documentos e memória e para restaurar peças históricas. Nesta disciplina entendemos estas imagens como **informação gráfica**.

O objetivo desta disciplina é proporcionar uma visão introdutória de "computação de imagens". Nosso foco não é nos aplicativos existentes para produzir, manipular, interpretar ou analisar imagens, e sim nas técnicas engenhosas desenvolvidas para tornar uma realidade tais aplicativos rodando em máquinas que foram originalmente projetadas com "inteligência lógica e aritmética". Tentaremos responder nesta disciplina as seguintes perguntas (Figura 1.2):

1. Sem "inteligência visual", como as máquinas podem ver figuras geométricas 3D, cores, tons e sombreamento?
2. Sem "inteligência visual", como elas podem produzir, a partir dos "conceitos espaciais", imagens próximas às que a nossa visão consegue perceber?
3. Sem "inteligência visual", como elas podem interpretar e analisar as imagens?

Esperamos que com esta visão introdutória, você consiga aplicar as técnicas existentes para criar melhores soluções a um problema específico em benefício da sociedade.

1.1 Processamento Visual

Para inserir em máquinas digitais "faculdades visuais", é fundamental que entendamos um processo de percepção. Um dos sistemas de visão mais complexo é o sistema da visão humana. Uma grande variedade de experimentos

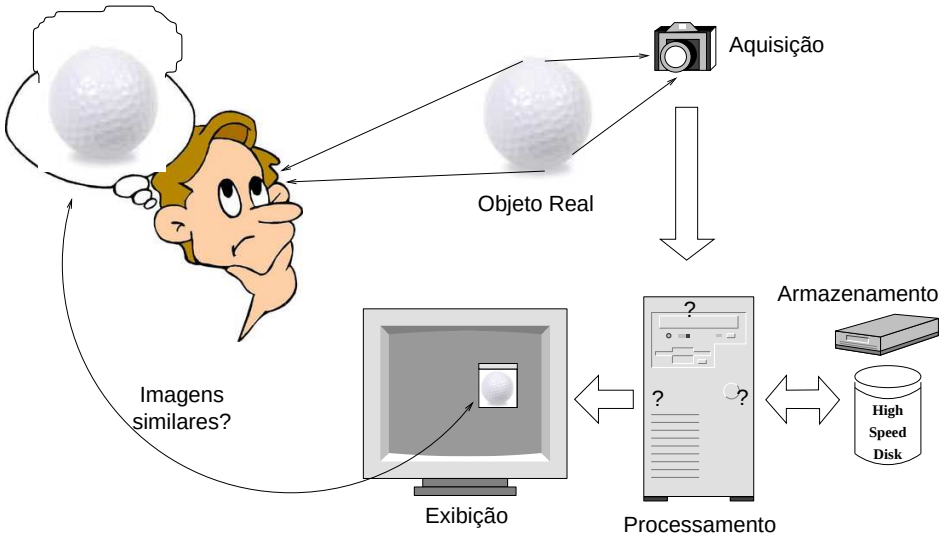


Figura 1.2: Processamento humano × processamento computacional.

e simulações sobre fenômenos ópticos, fisiológicos e psicológicos tem sido conduzida para conceber procedimentos programáveis.

A visão humana é um sistema complexo que começa com os dois olhos. As radiações luminosas atravessam o cristalino (lente biconvexa flexível) e convergem na membrana interna da parte posterior do olho, denominada retina. A retina é formada pela ramificação do nervo ótico que transmite as sensações luminosas ao cérebro (Figura 1.3). A imagem que se forma na retina é real, invertida e menor do que o objeto. Entretanto, "percebemos" os objetos em posição correta graças à forma como os sinais visuais são processados. De acordo com um modelo de percepção visual de três estágios, os sinais luminosos são, em primeiro lugar, transformados em características elementares, como forma, cor, textura e orientação. Em seguida, estas características simples são agrupadas em padrões e, conforme as atividades de atenção do observador, objetos armazenados na sua memória são "evocados" para criar percepção de um objeto como todo. Em 2007, os neurocientistas do MIT obtiveram resultados promissores com este modelo no reconhecimento de distintos objetos em diferentes ambientes.

A nossa visão é estérea. Quando olhamos para um objeto, são formadas duas imagens retinianas, uma em cada olho. Devido ao afastamento entre os dois olhos, as imagens não são idênticas. A observação simultânea dessas duas imagens **ligeiramente diferentes** força movimentos musculares dos

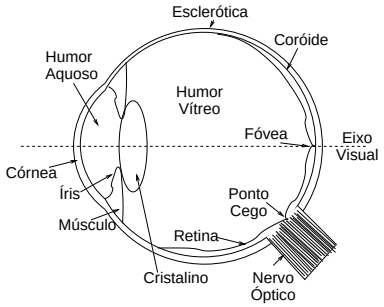


Figura 1.3: Esquema simplificado de um olho humano.

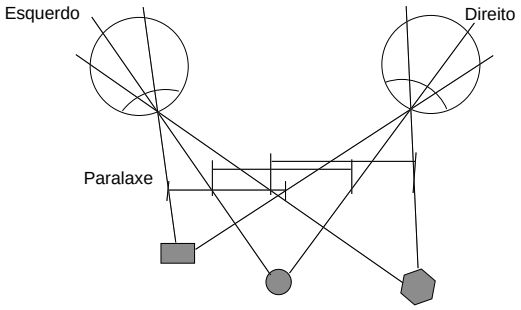


Figura 1.4: Paralaxe.

olhos para a paralaxe na distância entre dois pontos de diferentes alturas (Figura 1.4). Estes movimentos permitem que o cérebro diferencie a distância entre tais pontos, proporcionando percepção de relevo/profundidade. Os estudos revelam, no entanto, que 20% da população não possui visão estérea. Mesmo assim, não deixamos de ter a percepção de profundidade. Outros elementos, como oclusão, tonalidade e perspectiva, também propiciam percepção de profundidade.

A compreensão da habilidade visual tem sido muito importante no desenvolvimento de tecnologias dos dispositivos de exibição, como por exemplo,

acuidade: é a capacidade para distinguir dois elementos distintos. Ela nos dá uma idéia o limite de densidade de informação que a visão de um observador médio consegue perceber.

resposta Gaussiana: é a ação inibidora de cada fotoreceptor fora do seu centro de campo receptivo, de forma que nas regiões onde ocorrem variações abruptas de luz, elas aparentam ser mais claras ou mais escuras, criando ilusões como **Bandas de Mach** (Figura 1.5.(a)).

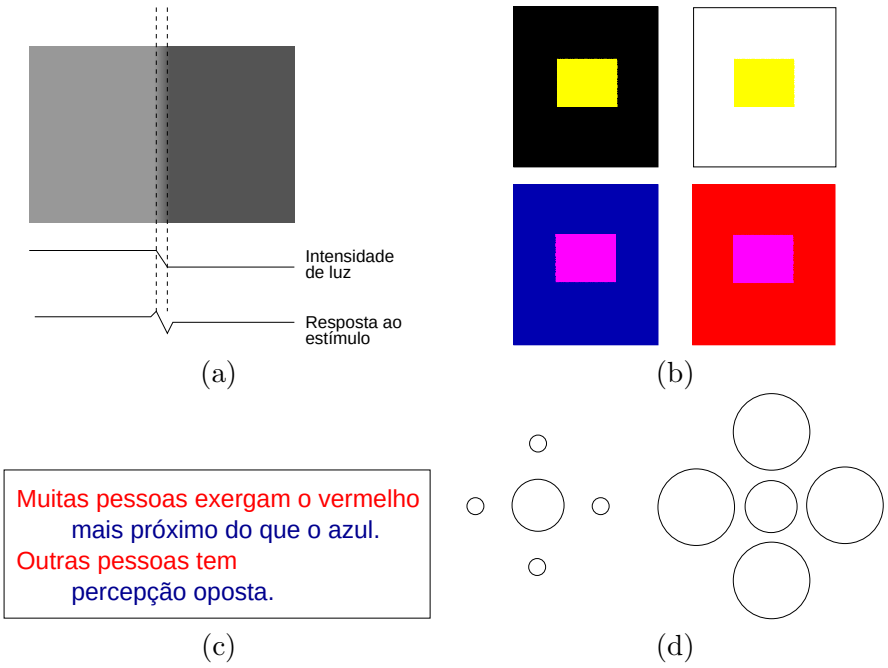


Figura 1.5: Habilidades visuais: (a) banda de Mach; (b) ilusão de contraste; (c) ilusão de profundidade; (d) ilusão de Ebbinghaus.

contraste de cor: é a ação inibidora de cores, de forma que uma mesma cor colocada sobre fundos distintos causa diferente percepção. Observe que na Figura 1.5.(b), o quadrado amarelo sobre o fundo preto parece mais "claro" que o quadrado de mesma cor sobre o fundo branco, enquanto o quadrado magenta sobre o fundo azul aparece mais "avermelhado" e o quadrado de mesma cor sobre o fundo vermelho aparece mais "azulado".

aberração cromática: é causada pela diferença dos pontos focais de radiações de comprimentos de onda distintos ao atravessarem o cristalino. Com isso, temos percepção de objetos de cores diferentes em distinta profundidade (Figura 1.5.(c)).

precisão na avaliação de tamanho: é a percepção diferenciada do tamanho de uma mesma forma quando circundada por outras figuras de tamanho diferente. A ilusão de Ebbinghaus é um exemplo desta percepção (Figura 1.5.(d)).

constância em percepção: é a capacidade de "perceber" as cores/intensidades pelo diferencial entre eles, e não pelos seus valores absolutos. É esta capacidade de balanceamento que nos faz perceber o "mesmo" branco sob luz solar e sob a luz de vela.

1.2 Sistemas de Aquisição

É o primeiro estágio de qualquer sistema de visão computacional. O princípio de funcionamento dos dispositivos deste estágio se baseia no modelo de visão humana que consiste em captura de radiações luminosas pelas células fotoreceptoras. Tipicamente, um sistema de aquisição de imagens digitais compreende três módulos: um dispositivo físico sensível a radiações eletromagnéticas para capturá-las e transformá-las em sinais elétricos proporcionais ao nível de energia recebida; um digitalizador para digitalizar os sinais elétricos; e um processador para melhorar a qualidade da imagem e sua codificação em um formato, de preferência, portátil.

Uns dispositivos são sensíveis a radiações de comprimentos de onda na faixa de 5pm a 1nm (raios X), outros são sensíveis a radiações na faixa de 700nm a 1mm (infravermelhas), e há equipamentos projetados para serem sensíveis a radiações do intervalo 400nm a 700nm (luz visível). As câmeras digitais, por exemplo, contém um arranjo de fotosensores a radiações luminosas. Se o fotosensor for um dispositivo de carga acoplada, em inglês *charge-coupled devices* CCD, ele "mede" a intensidade luminosa recebida por efeito

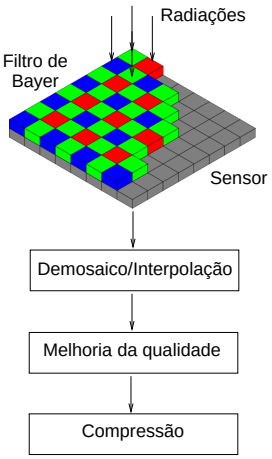


Figura 1.6: Aquisição de imagens.

fotoelétrico. Este arranjo de fotosensores pode ser coberto por um mosaico de filtros Bayer, sendo 50% dos fotosensores "recebem" a luz verde; 25%, luz vermelha e 25%, luz azul, a fim de emular o padrão de distribuição dos três tipos de cones na retina humana. Através de um algoritmo de interpolação de cor, a imagem "bruta" em mosaico de cores primárias é transformada em uma imagem colorida. Processamentos adicionais, como melhorar o contraste ou balancear cores para que elas pareçam mais próximas possíveis da nossa percepção, podem acontecer antes de codificar as imagens em um formato que possa ser lido por outros aplicativos. Há vários formatos de codificação. Eles podem ser sem compressão ou com compressão. Se for com compressão, distingue-se ainda compressão com perda ou sem perda. Entre os formatos mais pouplares temos JPEG, acrônimo em inglês para *Joint Photographic Experts Group*, GIF, acrônimo em inglês para *Graphics Interchange Format* e TIFF, acrônimo em inglês para *Tagged Image File Format*.

Há uma grande diversidade de sistemas de aquisição, de tecnologia bem distinta. De acordo com o valor (tipo de informação) contido em cada amostra (x, y) capturada por estes sistemas de aquisição, as imagens são classificadas em

imagens convencionais ou fotos: a informação em cada ponto da imagem é função da intensidade luminosa no ponto, como as obtidas por uma câmera digital. São também conhecidas como **imagens de intensidade**.

Imagens de trasmissão: a informação em cada ponto da imagem é função de algumas propriedades físicas dos objetos de interesse capazes de alterar de forma diferenciada algum fenômeno físico. Por exemplo, objetos que apresentam propriedades ópticas (difração, reflexão e refração) muito diferenciadas podem modificar de forma visualmente perceptível a direção de um feixe luminoso homogêneo que passa por eles, como raios X; ou objetos que apresentam comportamentos magnéticos bem distintos podem ter a sua polaridade alterada quando sujeitos a um intenso campo magnético, como as imagens de ressonância magnética – MRI. As diferenças detectadas são processadas e codificadas em imagens.

Imagens de profundidade: o valor em cada ponto é a profundidade do ponto em relação ao dispositivo de captura.

1.3 Sistemas de Exibição

Essencialmente, existem duas classes de dispositivos para exibição de imagens: os vetoriais e os *raster*.

Os dispositivos vetoriais reinaram da década 1960 a 1980 (Figura 1.7). Eles são fundamentados no modelo de plotador, em inglês *pen-plotter*. A principal caracterítistica destes dispositivos é que eles conseguem fazer traçados retos em qualquer direção. Portanto, as instruções de controle se resumem a um conjunto de vetores descritos pelos seus dois pontos extremos, o inicial e o final. A unidade processadora de *display*, em inglês *display processing unit* – DPU, lê a **lista de (instruções para) exibição** de figuras geométricas, também conhecida como *display list* em inglês, armazenada em memória principal, decodifica-a em sinais de controle analógicos, e envia tais sinais para o dispositivo, entre os qusis estão os sinais de posicionamento ou de movimento do "pincel eletrônico" do dispositivo. O "pincel eletrônico" varia da tecnologia do dispositivo de exibição: pode ser um feixe eletrônico em monitores de tecnologia CRT (canhões de raios catódicos) ou canetas em um plotador.

Nos dispositivos vetoriais o tempo de regeneração por quadro depende da quantidade de segmentos a serem desenhados. Quanto maior for a quantidade, maior será o tempo. Em monitores de tecnologia CRT, a frequencia de regeneração pode ser muito menor que 30Hz, gerando o efeito incômodo de cintilação. Além disso, o modo de desenho em dispositivos vetoriais não é muito apropriado para pintar uma área. Como uma solução alternativa, surgiu no início da década 1970 dispositivos de exibição do tipo *raster*

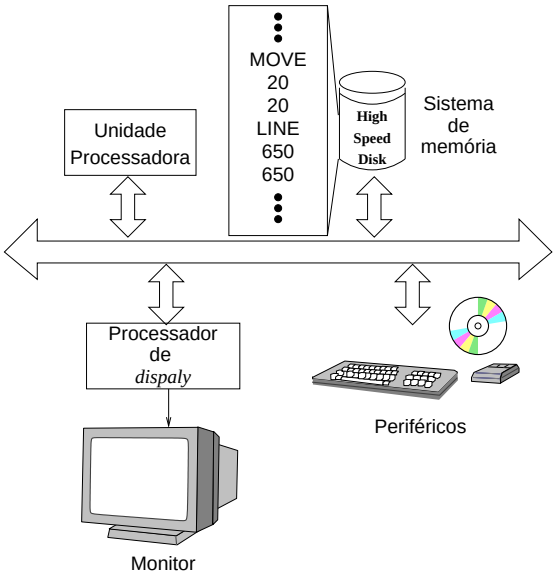


Figura 1.7: Arquitetura de um dispositivo de exibição vetorial.

(Figura 1.8). Estes dispositivos são fundamentados no modelo de preenchimento de *pixel*, em inglês *pixel-filling*, e caracterizam-se por exibir uma imagem através das suas amostras organizadas em um arranjo bi-dimensional denominado **memória de exibição**, em inglês *frame-buffer*. Cada amostra é representada por um conjunto de atributos, entre os quais a cor. O controlador de vídeo é responsável por varrer este arranjo periodicamente para regenerar o conteúdo da tela e o processador de *display*, pelas operações sobre os *pixels* (*picture element*) após a discretização. No caso de monitores de tecnologia de tubos catódicos, o movimento dos canhões é sempre o mesmo (varredura horizontal e retraço). Com isso, foi possível duplicar a frequencia de regeneração fazendo varreduras de forma intercalada, ou seja as linhas pares e ímpares são regeneradas alternadamente. Neste caso, dizemos que o monitor é um monitor **entrelaçado**. Do contrário, dizemos que o monitor é **entrelaçado**.

Os algoritmos que produzem imagens levam em consideração a tecnologia de exibição das imagens. Os dispositivos de saída são hoje predominantemente do tipo *raster*; portanto, nós veremos nesta disciplina os métodos de produção de imagens apropriadas para serem exibidas neles. Um fluxo de controle típico em placas gráficas de última geração é apresentado na Figura 1.9. No primeiro estágio, os vértices das figuras geométricas são pro-

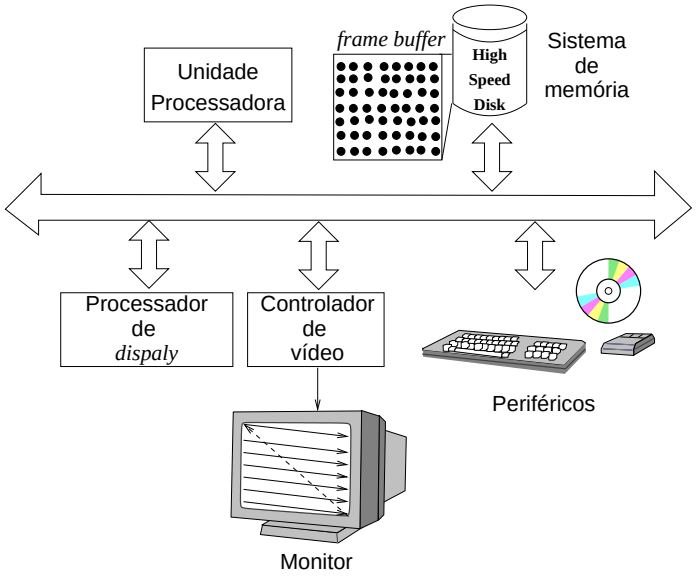


Figura 1.8: Arquitetura de um dispositivo de exibição raster.

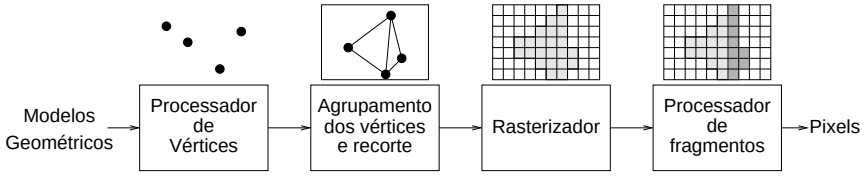


Figura 1.9: Fluxo de controle de renderização.

cessadas. Tanto a sua posição espacial quanto a sua cor são computadas. Em seguida, os vértices são agrupados para constituir figuras geométricas como todo, antes de serem recortadas no tamanho especificado pela aplicação. Depois, é aplicado um **algoritmo de rasterização** para discretizar as figuras geométricas. E, finalmente, é possível ainda aplicar algumas operações sobre as amostras discretas (fragmentos) para, por exemplo, criar efeitos visuais.

1.4 Modelos de Imagens

Para processar as imagens em sistemas computacionais utilizando unidades lógico-aritméticas, é necessário representar as imagens com um modelo matemático. Uma imagem bi-dimensional pode ser tratada como uma função f que depende de duas variáveis (coordenadas x e y de um plano) ou de

três variáveis. Quando ela varia com o tempo, uma terceira variável t é adicionada. O valor da função pode ser um valor escalar ou um vetor de valores reais, dependendo das informações contidas em cada imagem. O valor de uma **imagem monocromática** pode ser um valor (escalar) da luminância, isto é intensidade luminosa; enquanto numa imagem **colorida** ou **multiespectral**, o valor da função deveria ser, teoricamente, um vetor de n valores reais, cada qual corresponde a uma cor espectral para representar a distribuição espectral da cor.

Considerando uma imagem como uma função, podemos classificá-la em:

imagem contínua: se o domínio e o contra-domínio da função são contínuos,

imagem discreta: se o domínio da função é discreto, e

imagem digital: se o domínio e o contra-domínio da função são discretos.

A discretização do domínio de coordenadas (x, y) é conhecida como **amostragem** e a discretização do contra-domínio, ou seja dos valores ou da amplitude de f, é chamada **quantização**. Particularmente, quando a função f só assume dois valores, dizemos que a imagem é *binária* ou preto-e-branco.

De acordo com a quantidade de elementos, ou valores, no domínio e no contra-domínio, é possível caracterizar o "grau de detalhamento" de uma imagem digital em termos de **resoluções**. Distinguem-se quatro tipos de resolução:

resolução espacial (amostragem): define a proximidade entre as amostras de uma imagem discreta/digital (a disposição espacial das amostras em um plano forma um reticulado quadrado ou hexagonal).

resolução espectral (quantização): define a quantidade de cores espectrais existentes na imagem.

resolução radiométrica (quantização): define a quatidade de níveis de intensidade luminosa distinguíveis.

resolução temporal (amostragem): define o intervalo entre duas imagens subsequentes. É útil para caracterizar imagens dinâmicas.

1.5 Áreas de Pesquisa

Com advento de computadores de uso geral e surgimento de novos conceitos de engenharia de *software*, foi possível criar as primeiras imagens numa tela

de monitor de raios catódicos durante a Segunda Guerra Mundial. Cientistas e artistas se uniram para vencer mais um desafio: desenhar com um feixe de elétrons sobre uma tela de fósforos! As primeiras imagens foram geradas pelo Ben Laposky com uso de um computador analógico em 1950 (Figura 1.10.(a)). A possibilidade de utilizar uma máquina eletrônica para exibir imagens fez a Força Aérea Americana investir no desenvolvimento de um sistema computadorizado de controle de tráfico aéreo no seu programa de defesa aérea SAGE (*Semi-Automatic Ground Environment*). Este sistema, provido de soluções pioneiras de interação com uso de um revólver óptico, ficou operacional em 1958 (Figura 1.10.(b)). O termo **Computação Gráfica** foi introduzido pelo *designer* gráfico da Boeing, William Fetter, para referir o trabalho que ele desenvolvia em 1960: projetar um modelo humano para simulações ergonométricas em distintos ambientes (Figura 1.10.(c)).

O grande marco em computação gráfica interativa e interface homem-máquina foi o trabalho de doutorado de Ivan Sutherland, apresentado em 1963, quando ele implementou o sistema *Sketchpad* com o qual ele conseguiu criar, manipular, copiar e armazenar desenhos técnicos via uma caneta óptica (Figura 1.10.(d)). Para isso ele desenvolveu as primeiras **representações geométricas** e os primeiros algoritmos de transformação e recorte. Com esforço conjunto de pesquisadores de uma grande variedade de áreas de conhecimento, tanto o *hardware* quanto os algoritmos de computação gráfica tem evoluído muito rapidamente nestas últimas quatro décadas. Hoje em dia quase não se distingue mais o que é "virtual" e o que é real ... e é quase impossível predizer até onde chegaremos com o lançamento constante de novas séries de placas de vídeo programáveis.

A necessidade de comunicação entre dois mundos separados por um imenso oceano, o velho Mundo (Europa) e o novo Mundo (América), impulsionou no início dos anos 1900 o desenvolvimento de técnicas de transmissão de dados, incluindo as imagens. Em 1907, Belin apresentou uma solução pioneira de **processamento de imagens** para transmití-las por um sistema sem fio (Figura 1.11.(a)). Uma imagem era transformada em uma gravura, a profundidade de cujos sulcos variava de acordo com o nível de cinza em cada ponto, e a partir desta gravura são gerados sinais elétricos cuja intensidade dependia da profundidade dos sulcos. Em 1920 foi implantado o sistema Bartlane de transmissão submarina entre Londres e Nova Iorque. As imagens dos jornais eram digitalizadas. Para isso, era preciso "transformar" as imagens contínuas espacial- e colorimentricamente em um conjunto finito de amostras, cuja cor devia ser um dos 5 possíveis níveis de cinza. Em 1929, a resolução foi aumentada para 15 níveis de cinza, como ilustra a Figura 1.11.(b). A fase de maior crescimento no desenvolvimento

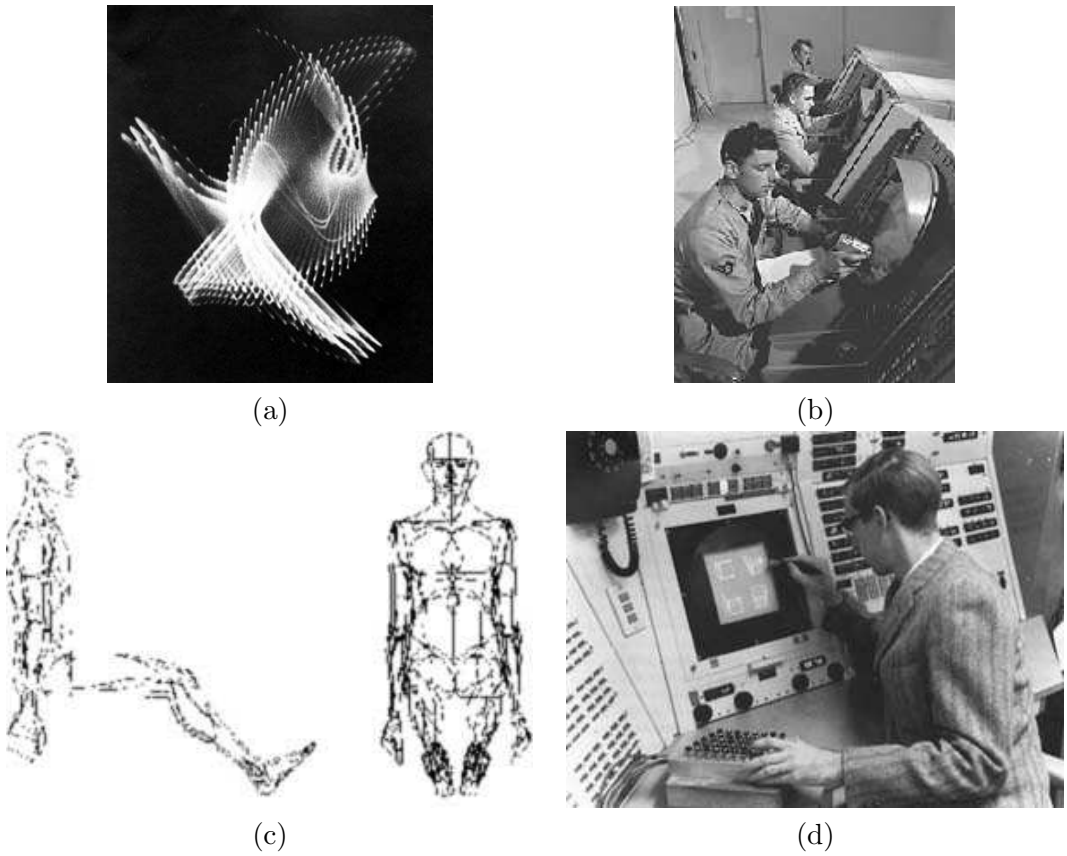
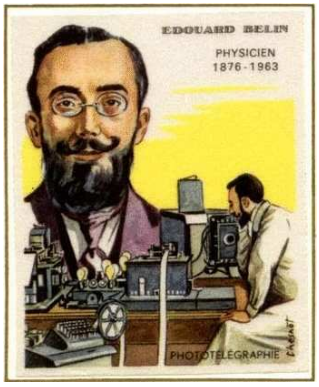


Figura 1.10: Imagens computadorizadas: (a) primeiras imagens; (b) primeiras interações com uso de um revólver óptico; (c) *Boeing Man*; (d) Sketchpad (Fonte: http://design.osu.edu/carlson/history/lessons.html).

das técnicas de processamento de imagens digitais se deu somente nos anos 1960, principalmente em *Jet Jet Propulsion Laboratory*, *Massachusetts Institute of Technology* – MIT, *Bell Labs* e *University of Maryland*. Entre as motivações destacam-se: corrigir as distorções inerentes à câmera de captura e melhorar a qualidade das imagens da Lua enviadas pela sonda Ranger e pelo seu sucessor Surveyor; realçar as imagens médicas, como de raios X, para facilitar análise e interpretação humana; e converter os sinais entre os padrões do sistema televisivo. Figura 1.11.(c) mostra uma imagem enviadda pela sonda Ranger. A proliferação das técnicas de processamento de imagens só aconteceu, no entanto, nos anos 1970 com o barateamento do custo e aumento da velocidade dos processadores. Hoje em dia, técnicas de processamento de imagens são adicionalmente utilizadas tanto na criação de efeitos especiais como na atenuação de artefatos digitais encontrados em imagens sintéticas ((Figura 1.11.(c))).

Em paralelo à proliferação de processamento de imagens e à melhoria da qualidade das imagens capturadas pelas câmeras, emergiu uma nova área de pesquisa denominada **Visão Computacional**. Ela consiste em desenvolver conceitos e teorias para construir algoritmos capazes de extrair informação a partir de imagens reais, de forma a permitir execução automática de algumas tarefas tediosas como detecção de objetos, reconhecimento de padrões, rastreamento e restauração de objetos. Como computadores não possuem inteligência visual, é necessário conceber uma forma de representar de forma unívoca os objetos de interesse. No desenvolvimento de um sistema de visão computacional, vários modelos propostos pelos fisiologistas, neurologistas e psicólogos para explicar o sistema de percepção humana foram levados em consideração. Figura 1.12 faz uma analogia entre a visão humana e a visão computacional: as câmeras e o *frame grabber* desempenham o papel similar de aquisição de imagens que o olho humano tem; enquanto o pré-processamento e a análise dos sinais, que ocorre no caminho da retina para córtex e no córtex, é feito pelos programas que são executados nos processadores. Hoje em dia, os métodos de visão computacional são ainda aplicados para "recuperar", a partir das imagens, **modelos geométricos 3D** de figuras complexas e utilizar estes modelos recuperados para sintetizar outras imagens. Isso reduz drasticamente o tempo que um *designer* despende durante o estágio de modelagem de uma cena sintética. Esta técnica é conhecida como **imageamento baseado em imagens**, em inglês *image-based rendering*.

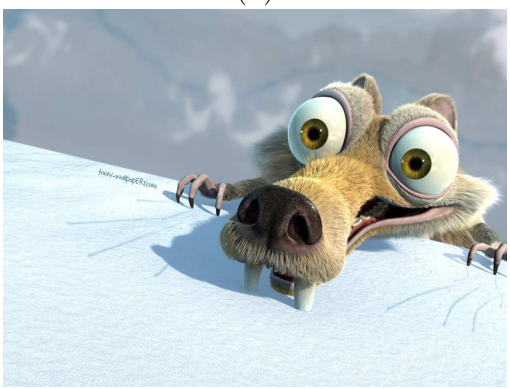
Sintetizando, em torno das imagens digitais atuam três grandes áreas de pesquisa de Computação: Computação Gráfica, Processamento de Imagens e Visão Computacional. Grosso modo, Computação Gráfica se ocupa



(a)

(b)

(c)



(d)

Figura 1.11: Processamento de imagens: (a) transmissão sem fio; (b) imagem quantizada em 15 níveis de cinza e enviada pelo sistema Bartlane (Fonte: http://www.digicamhistory.com/); (c) imagem da Lua enviada pela sonda Ranger em 31/07/1964; e (d) imagem sintética do filme "Idade do Gelo".

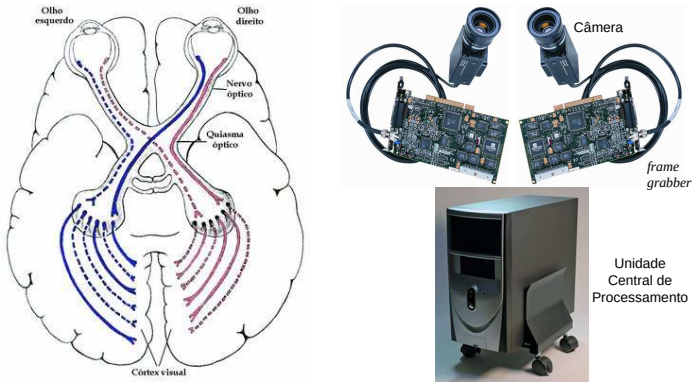


Figura 1.12: Visão Computacional: analogia entre (a) visão humana e (b) visão computacional.

da transformação de idéias, conceitos ou modelos em imagens reproduzíveis pelos dispositivos eletrônicos; Processamento de Imagens foca em transformação entre imagens; e Visão Computacional estuda interpretação e análise de imagens mediado por máquinas autônomas.

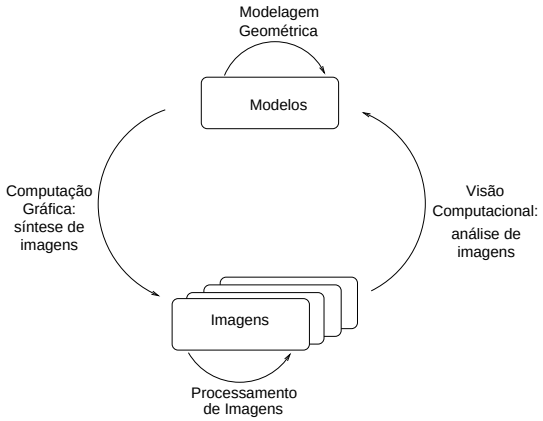


Figura 1.13: Áreas de pesquisa correlatas.

Capítulo 2

Programação Orientada a Eventos

O objetivo deste capítulo é dar uma noção sobre programação orientada a eventos. Este é o modelo de programação utilizado para implementação de aplicativos gráficos interativos. Após a leitura deste capítulo, você deve ser capaz de

- descrever o princípio de funcionamento dos principais dispositivos (físicos) gráficos.
- caracterizar os dispositivos lógicos e estabelecer a correspondência entre os dispositivos lógicos e os dispositivos físicos.
- explicar a organização de um sistema de janelas e o processamento de eventos.
- escrever um programa orientado a eventos utilizando interfaces de programação de aplicativos (API).

Um dos aspectos mais fascinantes dos atuais sistemas de informação gráfica é a sua interatividade. Diferente dos programas tradicionais, que seguem um fluxo de controle pré-estabelecido, o usuário pode interferir no controle de fluxo destes sistemas, compartilhando com o computador a responsabilidade na execução de uma tarefa, como ilustra o **ciclo de interação** na Figura 2.1. Através dos **dispositivos de saída**, o sistema computacional exibe para usuário o estado dos dados. O usuário vê/lê a informação apresentada, interpreta, avalia e intervém no fluxo de controle através dos

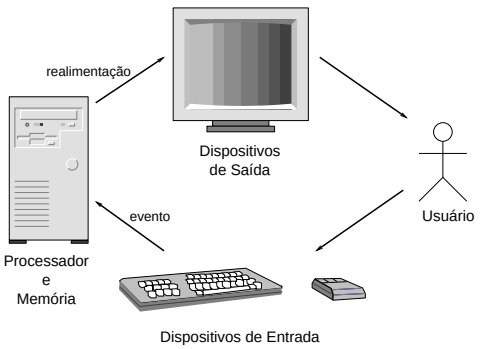


Figura 2.1: Ciclo de Interação.

dispositivos de entrada. Os sinais de dispositivos de entrada são processados como **eventos** pelo processador que dispara um tratador apropriado para modificar o estado dos dados armazenados em memória. Finalmente, o novo estado dos dados deve ser informado para usuário decidir a próxima ação. A interação é repetida até que o estado de dados almejado seja alcançado.

Há dois pontos críticos neste ciclo de interação. Ambos são relacionados com a possível discrepância entre a capacidade de processamento de uma máquina e a capacidade de processamento humano. De um lado, a máquina pode exibir os estados dos dados de forma ilegível ou inadequada para usuários interpretá-los e tomar ações corretas. De outro lado, o usuário pode gerar uma sequência de ações que não consiga fazer o processador alcançar o estado almejado. Um bom projeto de interface homem-máquina pode atenuar as barreiras, melhorando a usabilidade de um sistema gráfico. O foco deste capítulo é, no entanto, uma breve visão do ambiente em que tais programas são implementados. Na seção 2.1 daremos uma visão introdutória da tecnologia dos dispositivos de entrada e de saída. Em seguida, mostramos na seção 2.2 como tais dispositivos são abstraídos no nível de programação para facilitar o seu uso via um sistema de janelas. Na seção 2.3 detalharemos o processamento de eventos em sistemas de janelas, cuja compreensão facilita o entendimento do modelo de programação orientada a eventos. Finalmente, apresentamos um exemplo de programação com uso da biblioteca de componentes gráficos GLUT (`http://www.opengl.org/resources/libraries/glut/`) que consiste uma simples interface de programação orientada a eventos para a interface gráfica OpenGL (`http://www.opengl.org/`).

2.1 Dispositivos Físicos

Em sistemas de informação gráfica são fundamentais os periféricos, através dos quais o computador consegue interagir com o mundo externo. Em geral, um sistema de informação gráfica é provido de um dispositivo de saída (monitor) e dois dispositivos de entrada (um teclador e um apontador). Hoje em dia, é muito comum encontrar também escaneadores, *joystick*, *webcam*, mesas digitadoras, microfone, alto-falantes e impressoras.

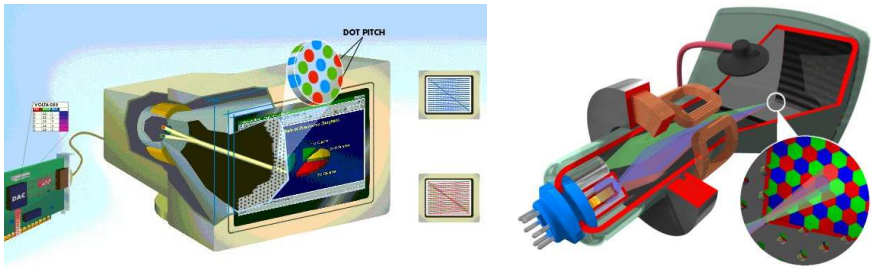
2.1.1 Dispositivos de Saída

Características básicas de monitores de vídeo e impressoras são apresentadas nesta seção.

Monitores de Tubo Catódico

O monitor de tubo catódico (CRT) é o dispositivo de saída mais popular. As informações são exibidas através de uma tela semelhante à TV. A tela é revestida por uma película de fósforos. Um feixe de elétrons, cuja direção é controlada pelas placas de deflexão, varre a tela linha por linha. Pontos de luz aparecem na tela quando a energia de elétrons é transferida para fósforos. Cada ponto é chamado *pixel* ou *picture element*. A quantidade de *bits* necessários para distinguir a intensidade em cada ponto é conhecida como **profundidade** da tela. Ao chegar no canto inferior direito, o feixe retorna para o canto superior esquerdo e reinicia a varredura horizontal (Figura 2.2.(a)). No caso ce monitores coloridos, temos em cada ponto três tipos de fósforos capazes de emitirem luzes azuis, verdes e vermelhas ao serem excitados. Três feixes de elétrons são utilizados para controlar individualmente cada tipo de fósforo produzindo uma grande variedade de cores.

O número máximo de pontos distinguíveis é conhecido como **resolução** do monitor. A resolução é dependente do tipo de fósforo, da intensidade da imagem, e do sistema de deflexão e foco. A razão entre a quantidade de pontos horizontais e a quantidade de pontos verticais necessários para produzir um segmento de mesmo comprimento é conhecida como **razão de aspecto** (*aspect ratio*) . A razão de aspecto típica de um monitor é 4:3, ou seja 4 pontos horizontais tem o mesmo comprimento de 3 pontos verticais. A luz emitida pelo fósforo tem curta **persistência**. Usualmente, 10 a 60 μs. A tela precisa, portanto, ser **retraçada** (*refresh*) periodicamente, ou seja, fósforos re-energizados periodicamente, a fim de manter a imagem "fixa" na



(a) (b)

Figura 2.2: Monitor de tubo catódico: (a) monocromático e (b) colorido (Fonte: http://www.pctechguide.com/42CRTMonitors_Anatomy.htm).

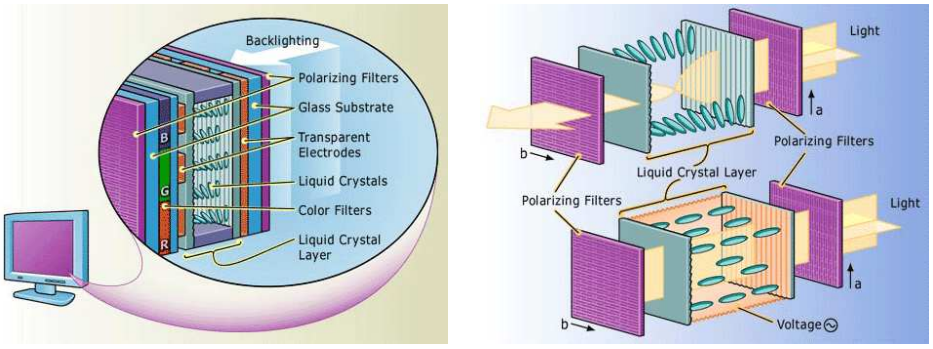
tela. Caso a frequência de retraço for baixa, ocorre o efeito de cintilação (*flicker*), onde uma sombra parece percorrer constantemente a tela, fazendo com que esta pareça estar piscando.

Com o aumento da resolução e da profundidade a quantidade de dados que trafega entre computador e monitor aumentou excessivamente, comprometendo o desempenho o sistema. Placas de vídeo com um processador próprio, GPU ou acelerador gráfico, conseguiram melhorar comparativamente o desempenho em dezenas de vezes. Com elas, o computador passou a transmitir a descrição da imagem com base em primitivas gráficas numa linguagem própria. A descrição é interpretada e executada na GPU, tendo como resultado uma imagem. O processo de retraço também passou a ser atribuição do processador de vídeo, não havendo necessidade do processador principal re-enviar uma imagem que não sofresse alterações. Além disso, o processo de envio das modificações de uma imagem passou a ser feito por diferença, ficando a cargo do processador de vídeo alterar consistentemente a imagem.

Monitores LCD

Cada vez mais populares, os monitores de cristal líquido LCD (*Liquid Crystal Display*) já são considerados por muitos indispensáveis ao uso do computador. Além de emitirem pouquíssima radiação nociva, consomem menos energia e ocupam menos espaço.

As telas de LCD são formadas por um material denominado cristal líquido. As moléculas desse material são distribuídas entre duas lâminas transparentes polarizadas. Essa polarização é orientada de maneira diferente nas duas lâminas, de forma que estas formem eixos polarizadores per-



(a) (b)

Figura 2.3: Monitor LCD.

pendiculares. Estas finas placas possuem pequenos sulcos, isolados entra si, cada um com um eletrodo transparente. Cada um destes sulcos representa um *pixel*. Atrás do conjunto há uma fonte luminosa (Figura 2.3.(a)). A primeira lâmina só permite passar a luz orientada verticalmente. Essa luz atravessa o cristal "retorcido" e fica orientada horizontalmente. Ela consegue, portanto, atravessar a lâmina frontal. Dizemos então que as moléculas assumem o estado transparente. Quando se aplica uma tensão diferente, as moléculas ficam alinhadas e assumem o estado opaco, pois impedem a passagem da luz pela lâmina frontal (Figura 2.3.(b)).

No caso de LCDs mono-cromáticos, cada ponto da tela corresponde a um *pixel* da imagem. Já no caso dos monitores coloridos, cada *pixel* é constituído por um grupo de 3 pontos, um verde, um vermelho e outro azul. Como nos monitores CRT, as cores são obtidas através de diferentes combinações de tonalidades dos três pontos.

Impressoras

Duas classes de impressoras são populares: a jato de tinta e a laser. Nas impressoras a jato de tinta as cabeças de impressão jogam pequenos jatos de tinta sobre o papel formando o desenho da página montada no computador. E nas impressoras a laser um feixe de luz laser "desenha" sobre o cilindro com cargas negativas. Como o toner tem carga positiva, ele adere às áreas negativas do cilindro. Com o toner fixado sobre seu corpo, o cilindro rola sobre a folha de papel, que se movimenta sobre uma cinta abaixo dele. Antes de entrar na cinta, o papel recebe uma carga negativa de eletricidade. Como ela é maior que a imagem eletrostática sobre o cilindro, o papel "atrai" o

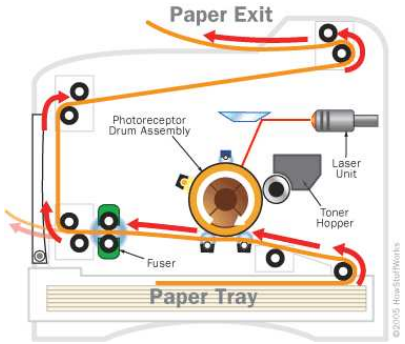


Figura 2.4: Impressora a laser.

toner enquanto o cilindro gira sobre ele. Para que o papel não fique colado no cilindro, o papel é "descarregado" imediatamente após a transferência do toner. Finalmente o papel passa por um fusor, dispositivo que emite calor para fundir o toner com as fibras do papel (Figura 2.4). Depois que o toner fique fixado no papel, uma lâmpada de descarga aplica uma luz muito intensa sobre o cilindro para apagar a imagem que estava gravada. Em seguida, ele recebe novamente uma carga elétrica positiva para a próxima impressão.

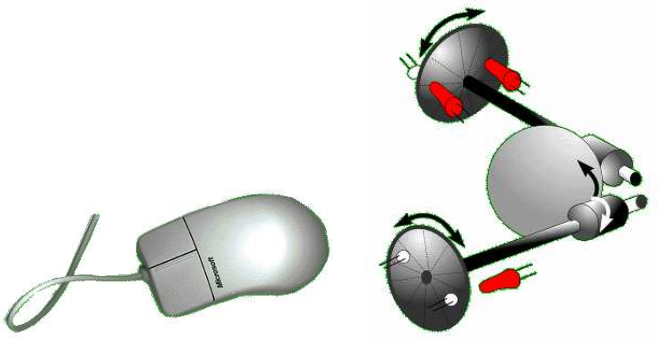
2.1.2 Dispositivos de Entrada

São através dos dispositivos de entrada que um computador consegue captar as ações dos usuários. Há dispositivos que captam variações nos estados do dispositivo e há dispositivos que detectam o estado corrente do dispositivo.

Mouse

O principal objetivo do *mouse* é traduzir **movimentos relativos** da mão de um usuário em sinais que o computador possa utilizar.

No interior de um **mouse mecânico** há uma esfera que toca a mesa sobre a qual o *mouse* move. Dois roletes dentro do *mouse* tocam a esfera. Um dos roletes é orientado para detectar o movimento na direção X e outro na direção Y (90^0 em relação ao primeiro). Quando a esfera gira, um ou ambos os roletes giram também. Cada rolete se conecta a uma haste, e esta haste gira um disco com furos na borda. Quando um rolete gira, a sua haste e o disco giram. No outro lado do disco há um LED (diodo emissor de luz) infravermelho e um sensor infravermelho. Os furos no disco bloqueiam o feixe de luz que vem do LED, permitindo que o sensor infravermelho capte



(a) (b)

Figura 2.5: Mouse.

pulsos de luz. A frequência de pulsação está diretamente relacionada com a velocidade do *mouse* e a distância que ele percorre. Um microprocessador lê os pulsos do sensor infravermelho, codifica-os em dados binários e envia estes dados binários para o computador através do fio do *mouse* (Figura 2.5).

Desenvolvido pela Agilent Technologies e lançado no final de 1999, o **mouse óptico** utiliza uma pequena luz vermelha emitida por um LED. Esta luz atinge a superfície onde o *mouse* fica apoiado e é refletida. A luz refletida é captada pelo sensor CMOS (*Complementary Metal Oxid Semiconductor*). Milhares de imagens captadas são enviadas a cada segundo para o processador digital de sinais (DSP) que procura extrair os padrões em cada imagem. Através da **variação relativa** dos padrões na sequência de imagens, o DSP consegue determinar se o *mouse* movimentou, qual foi a distância de movimento e em qual velocidade.

Além de proporcionarem movimentos mais suaves aos *cursors*, o *mouse* óptico é capaz de trabalhar em quase toda superfície sem um *mouse pad*.

Teclado

O teclado junto com o *mouse* formam o cojunto de periféricos indispensáveis em computadores pessoais. Enquanto o *mouse* foi projetado para capturar posições (relativas) espaciais, o teclado é utilizado predominantemente na escrita de textos e controle de funções do sistema operacional de um computador. Inspirado de antigas máquinas de escrever, ele consiste essencialmente de um arranjo de botões, denominados **teclas**, que são ligados a um *chip* dentro do teclado, onde identifica a tecla pressionada e manda para o processador o código correspondente (Figura 2.6.(a)). Aproximadamente 50%

(a) (b)

Figura 2.6: Teclado.

delas produzem códigos que representam letras, números e sinais (caracteres alfa-numéricos) e outras afetam o modo como o microcomputador opera ou agem sobre o próprio teclado, como as teclas "Caps Lock" e as de controle de cursor. Figura 2.6.(b) ilustra a disposição padrão de um teclado com 105 teclas.

Mesa Digitadora

Um *mouse* não reage à pressão nem a mudanças mais sutis na direção e inclinação da mão. Isso o coloca em posição desfavorável em relação a uma mesa digitadora com a qual consegue-se trabalhar de forma mais parecida com a que se trabalha com ferramentas artísticas tradicionais, como variar a espessura ou a intensidade de um traçado exercendo mais ou menos pressão sobre a caneta.

Uma mesa digitalizadora, ou *tablet* em inglês, é composta de uma prancheta e uma caneta (Figura 2.7). Há mesas digitadoras de várias tecnologias. Uma delas é baseada em indução eletromagnética, como as da Wacom, que elimina a necessidade de uso de baterias em canetas. A área retangular da prancheta contém fios horizontais e verticais que operam como bobinas. No modo de transmissão, a prancheta gera sinal eletromagnético que induz um sinal no circuito LC contido na caneta. Em seguida, o modo das bobinas é alternado para recepção para captar o sinal induzido na caneta. Com base na intensidade do sinal, a mesa digitalizadora consegue determinar a **posição absoluta** da caneta sobre a área retangular sem que esta toque na superfície. Quando a mesa digitalizadora é provida de sensor de pressão, um toque da caneta na superfície pode equivaler a um clique e arrastar a ponta da caneta é como clicar e arrastar. Há mesas digitalizadoras que conseguem determinar o ângulo da caneta em relação à superfície ou que possuem, no

Figura 2.7: Mesa digitalizadora.

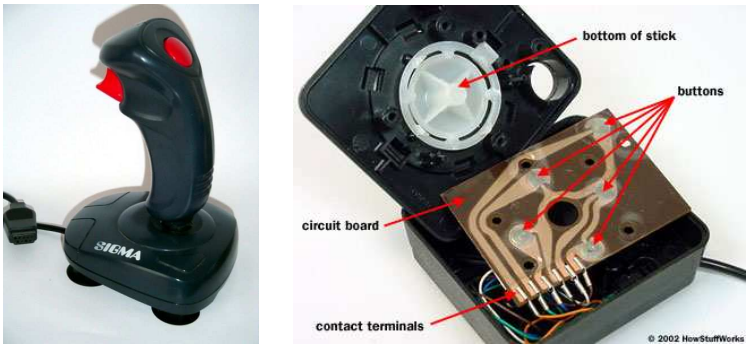
corpo de suas canetas, botões que servem como duplo clique, botão direito ou outra função programável.

Joystick

O **joystick** é um periférico usado freqüentemente para controlar os jogos de vídeo. Ele consegue traduzir, de forma natural, os deslocamentos de uma mão através de um bastão ou botões de disparo. Este bastão é fixado a uma base plástica com um revestimento de borracha flexível (Figura 2.8.(a)). Nesta base encontra-se uma placa de circuito constituída de várias trilhas que se conectam a diversos terminais de contato, a partir dos quais o *joystick* se comunica com um computador através de fios comuns. O circuito na placa é, de fato, constituído de diversos circuitos menores entre os terminais de contato (Figura 2.8.(b)). Quando o bastão é movido, um destes circuitos pode ser fechado e o computador consegue reconhecer a posição correta do bastão pelo circuito que se fechou. Por exemplo, empurrar o bastão para frente fecha a "chave para frente" e empurrá-lo para a esquerda fecha a "chave esquerda". Há *joysticks* mais sofisticados que conseguem captar adicionalmente movimentos sutis de posição.

Câmera Digital

Tanto câmeras digitais como câmeras convencionais possuem uma série de lentes para focalizar a luz e criar a imagem de uma cena. Mas em vez de focalizar essa luz sobre um pedaço de filme e disparar um processo químico de reações, a câmera digital o faz sobre um dispositivo semicondutor, conhecido como *charge coupled device* CCD em inglês, que grava a luz eletronicamente. Um processador então decompõe essas informações eletrônicas em



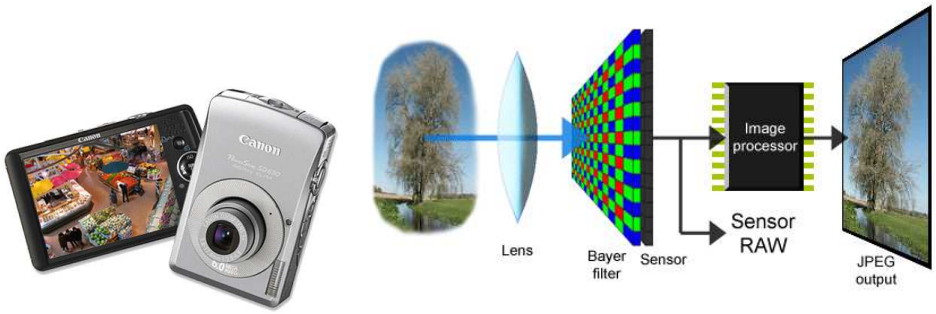
(a) (b)

Figura 2.8: Joystick.

dados digitais. Assim, sem aguardar a revelação, pode-se visualizar imediatamente o que foi capturado através de uma tela de cristal líquido (LCD) (Figura 2.9.(a)).

O sensor CCD só consegue perceber a intensidade da luz. Uma maneira mais econômica e prática para capturar uma imagem colorida é colocar permanentemente um filtro chamado **conjunto de filtro de cores** sobre cada sensor individual. Ao decompor o sensor em uma variedade de células vermelhas, azuis e verdes, é possível obter, através de interpolações, informações suficientes nos arredores de cada sensor para estimar de forma bastante precisa sobre a cor verdadeira naquele local. O padrão mais comum de filtros é o **padrão de filtro Bayer**, que alterna uma fileira de filtros vermelhos e verdes com uma fileira de filtro azuis e verdes. A quantidade de células não é dividida por igual: há tantas células verdes quanto azuis e vermelhas combinadas (Figura 2.9.(b)). Isso ocorre porque o olho humano não é igualmente sensível a todas as três cores. É necessário incluir mais informações provenientes das células verdes para criar uma imagem que o olho perceberá como uma "cor verdadeira".

Ao invés de filmes analógicos, as câmeras digitais usam diversos sistemas de armazenagem digitais para guardar as imagens capturadas. A mídia mais utilizada são as memórias *flash*. Os fabricantes de câmeras digitais freqüentemente desenvolvem suas próprias memórias *flash*, como cartões SmartMedia, cartões CompactFlash e *Memory Sticks*. Outros dispõem possibilidades de armazenar em dispositivos como discos rígidos. Não importa o tipo de armazenamento que usem, todas as câmeras digitais precisam de muito espaço para suas imagens. Para aproveitar ao máximo o espaço de armazenamento, quase todas as câmeras digitais usam algum tipo de compactação de dados



(a) (b)

Figura 2.9: Câmera digital.

para diminuir o tamanho dos arquivos. O formato de arquivo compactado mais utilizado é JPEG. No entanto, se quisermos preservar a qualidade original das imagens, recomenda-se utilizar formato sem compactação. Neste caso, o formato TIFF é o mais difundido entre as câmeras digitais.

Uma câmara digital provida de uma conexão USB e de um aplicativo que consegue capturar as suas imagens numa frequência pré-especificada e distribuí-lo através de uma conexão da internet é conhecida como **webcam**.

2.2 Dispositivos Lógicos

Sob o ponto de vista de programação, é interessante que o acesso aos periféricos ocorra de forma mais independente possível das características tecnológicas de cada um. O conceito de dispositivos lógicos foi introduzido para permitir que os programas "vejam" os dispositivos pelas suas funções e não pela sua especificação técnica.

2.2.1 Dispositivos de Saída

O primeiro passo para exibição de uma imagem num dispositivo de saída é a transmissão dos dados a partir do computador ou dispositivo de armazenamento de dados para o dispositivo de saída. O processador de imagem rasteriza os dados e converte a informação recebida em uma imagem gráfica; esta imagem é então enviada para o dispositivo de saída.

A arquitetura dominante para os dispositivos de saída é a de *frame-buffer*, conforme ilustra Figura 2.10. Nesta arquitetura as imagens discretas são geradas pelos processadores e transmitidas para um dispositivo de saída.

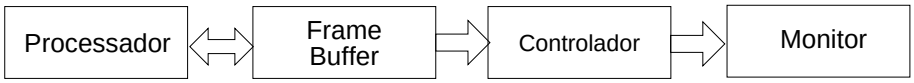


Figura 2.10: *Frame buffer*.

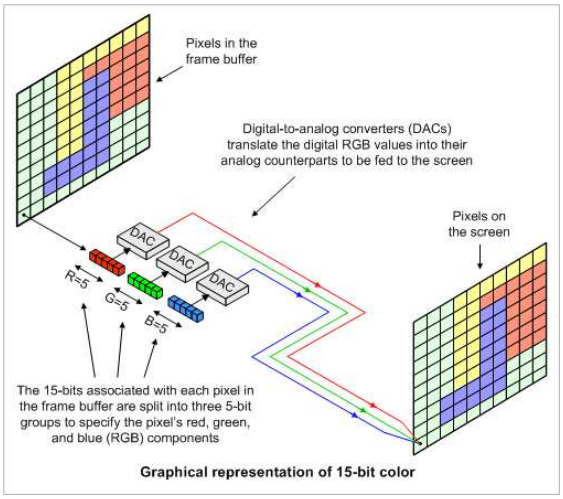


Figura 2.11: Conversão digital–analógica.

Os valores de cada *pixel* da imagem são armazenados em uma memória do dispositivo, que é conhecida como *frame-buffer*. O conteúdo desta memória é convertido para os sinais de imagem através de um controlador. A técnica de conversão varia de acordo com a tecnologia do dispositivo de exibição.

No caso de dispositivos de tecnologia baseada em raios catódicos, o controlador de vídeo varre o *frame buffer*, de linha em linha, e converte os valores das cores primárias (vermelho, verde e azul) armazenados em cada posição da memória em tensões que controlam a intensidade de feixe de elétrons que excitam as partículas de fósforo no correspondente *pixel* da tela de vidro do monitor (Figura 2.11). No caso de dispositivos de tecnologia baseada em cristal líquido, ao invés de tensões, o controlador converte os valores das cores primárias em opacidade do cristal. E nas impressoras, tais valores são transformados em carga eletrostática positiva sobre o tambor (impressoras a laser) ou em gotículas de tinta emitidas através de minúsculas aberturas na cabeça de impressão (impressoras a jato de tinta).

2.2.2 Dispositivos de Entrada

De acordo com o padrão gráfico GKS (*Graphical Kernel System*), distinguem-se seis dispositivos de entrada lógicos:

locator: retorna a posição de um ponto da imagem. Exemplos de dispositivos físicos que realizam esta tarefa básica são *mouse* e teclado. Outros dispositivos conhecidos são *joystick*, *trackball* e *tablet*.

stroke: retorna uma sequência de pontos. Tipicamente, esta tarefa pode ser realizada com uso de um *mouse*.

pick: retorna a referência de um objeto contido na imagem exibida pelo dispositivo de saída. O dispositivo físico que tipicamente realiza esta função é o *light-pen*.

valuator: retorna um valor numérico (escalar). Exemplo típico de um *valuator* são os *dials* (potenciômetros).

string: retorna uma sequência de caracteres alfa-numéricos. O dispositivo físico típico que realiza esta tarefa é o teclado.

choice: retorna uma opção dentre um conjunto de alternativas pré-definidas. Exemplo típico de um dispositivo físico que realiza esta tarefe é o *tablet* com funções pré-definidas. É comum, hoje em dia, mapear um conjunto de alternativas às teclas do teclado, tornando-o também um dispositivo lógico de *choice*. Podemos ainda utilizar o *mouse* como um dispositivo lógico de *choice*, se mapearmos cada um dos seus botões a uma das funções pré-definidas ou provermos na interface do aplicativo sub-janelas com as opções pré-definidas.

Um sistema gráfico interativo suporta, usualmente, mais de uma classe de dispositivos lógicos. No mínimo, *locator* (*mouse*) e *string* (teclado).

Considera-se ainda que os dispositivos lógicos podem operar em um dos três modos: **requisição**, **amostragem** e **evento**. No modo de requisição, o aplicativo habilita o dispositivo e fica aguardando a entrada de um dado. No modo de amostragem, o aplicativo amostra o estado do dispositivo periodicamente. E no modo de evento, o dispositivo dispara o processamento de eventos quando há algum dado disponível. Na seção 2.3 explicaremos o processamento de eventos.

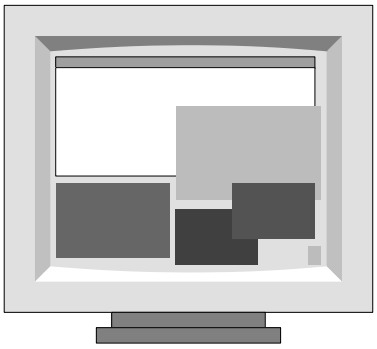


Figura 2.12: Uma área de trabalho com multijanelas.

2.3 Sistema de Janelas

Um dos desafios tem sido projeto e implementação de uma interface amigável entre um "processador digital" e um "controlador humano", aproveitando ao máximo a tecnologia multimodal disponível. Nesta seção apresentamos o conceito de **sistema de janelas** que facilita a implementação dessa interface. Um sistema de janelas é uma "camada de software" que gerencia o uso compartilhado dos dispositivos de entrada e de saída entre vários aplicativos, de forma similar a um sistema operacional que gerencia os recursos computacionais entre vários processos. Em um sistema de janelas, uma tela do monitor é considerada uma área de trabalho através da qual um conjunto de aplicativos pode interagir com o usuário e cada aplicativo tem uma sub-área própria, como se a tela do monitor estivesse dividida em várias subáreas totalmente independentes (Figura 2.12). Estas sub-áreas, usualmente retangulares, são conhecidas como **janelas** e são organizadas hierarquicamente em árvore, sendo a tela considerada a **janela-raíz**. Todas as janelas são referenciáveis por meio de identificadores (ID).

Existe, em adição ao sistema de janelas, um **gerenciador de janelas** que provê facilidades para usuários personalizar a aparência das janelas e a forma de interações através dos periféricos. Em Microsoft Windows este gerenciador é integrado ao sistema de janelas, enquanto em X Windows, o gerenciador é um componente em separado. Exemplos de gerenciador de janelas para X Windows: Motif Windows Manager, Tab Windows Manager e Window Maker.

As ações dos usuários, como movimento de *mouse*, aperto/soltura de um botão de *mouse*, pressionamento de uma tecla de teclado e focalização de uma janela, são traduzidas em **eventos** e despachados para a janela apropri-

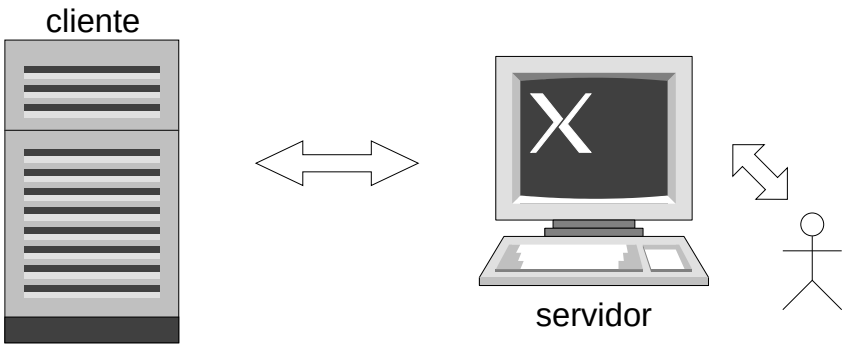


Figura 2.13: Modelo cliente-servidor.

ada para serem tratadas. Em Microsoft Windows tanto o programa aplicativo quanto a interface com usuários estão residentes numa mesma máquina. O X Windows, usado frequentemente em conjunto com o sistema operacional UNIX, opera segundo o **modelo cliente-servidor**: o **servidor** X "serve" vários programas aplicativos, conhecidos como **cliente** X, recebendo as suas requisições de saída gráfica como também enviando para eles os eventos gerados pelo usuário. O servidor e o cliente podem estar instalados em uma mesma máquina ou em máquinas distintas. Este modelo de comunicação permite o uso de janelas e o controle de aplicativos de modo transparente através da rede (Figura 2.13).

No restante desta seção vamos detalhar o processamento de eventos.

2.3.1 Fila de Eventos

Como se consegue "sincronizar" as ações não-programáveis de um usuário com uma sequência de instruções pré-estabelecidas em um programa aplicativo? Uma solução popular é por meio de uma **fila (principal) de eventos (de entrada)**, como em X Windows. Após traduzir as ações em eventos, estes são inseridos em uma estrutura. O **tratador de eventos** os remove da estrutura de acordo com o princípio FIFO (*first in, first out*) e os converte em **eventos de interação** antes de despachá-los para serem tratados por um fragmento de códigos. O tratador de eventos pode tanto concatenar uma sequência de eventos de entrada em um único evento de interação, como CTRL+ALT+Del e duplos cliques, quanto mascarar eventos irrelevantes. Algumas classes de eventos processáveis em todos os sistemas de janela: eventos de dispositivo de apontamento como *mouse*, eventos de teclado e eventos de janela, como criar, destruir e focalizar uma janela.

2.3.2 Despachador de Eventos

Sendo a tela do monitor fragmentada em diversas janelas, muitas delas compartilhando o mesmo conjunto de *pixels*, é necessário elaborar uma estratégia de associação entre eventos de interação com trechos de código de processamento para obter resposta apropriada. Quando a posição do *cursor* em relação a uma janela não é ambígua, esta associação é simples. Em alguns sistemas a informação posicional do apontador é utilizada para despachar também os eventos do teclado: eles são encaminhados para a janela que contém o *cursor*. O problema se complica quando existe mais de uma janela sob o *cursor*. Neste caso, qual deve ser o critério de escolha da janela? Uma solução é a introdução do conceito de **foco**: somente uma janela pode estar em foco em cada instante e todos os eventos gerados num dado momento são despachados para a janela em foco.

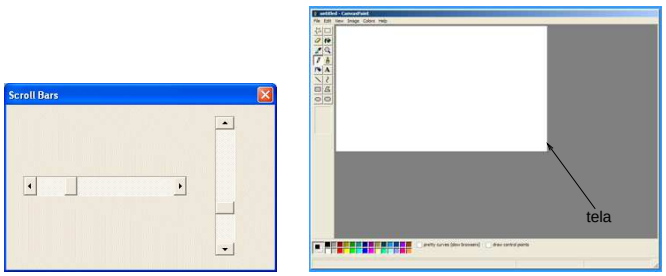
2.3.3 Tratador de Eventos

Um ciclo de interação envolve o disparo de um evento de interação, o despacho deste evento à janela correta e a execução de um trecho de código condizente com a ação esperada (Figura 2.1). A sequência de instruções a serem executadas deve ser, portanto, dependente do contexto de aplicação. Como se integram estas instruções ao restante dos códigos de processamento de eventos? Uma solução é pelo mecanismo callback que permite passar como argumento o endereço do tratador de eventos, provido pela aplicação, para o sistema de janelas. Assim, quando um evento é disparado, o sistema de janelas automaticamente chama o seu correspondente tratador para executar a tarefa pré-programada.

2.4 Programação

Entendendo a arquitetura de um sistema de janelas, fica mais fácil entender a forma como um programa orientado a eventos é estruturado. Ele possui sempre um programa inicial contendo instruções de inicialização e um **laço principal de eventos** cuja tarefa é simplesmente retirar os eventos da fila e despachá-los. As funções de inicialização são as instruções comuns a todos os aplicativos, como mascaramento de eventos irrelevantes, registro de função callback e criação de uma janela principal. O comportamento real de um aplicativo é, de fato, definido pelos tratadores de evento registrados com callback.

Para reduzir o tempo de desenvolvimento de uma interface gráfica com



(a) (b)

Figura 2.14: *Widgets*: (a) barra de rolagem e (b) tela.

usuário (GUI – *graphical user interface*), são disponíveis bibliotecas de componentes de interface gráfica, ou widgets em inglês, em cima de um sistema de janelas. Um widget é um componente capaz de realizar uma ou mais tarefas de interação com um tipo específico de dados. Tipicamente, um *widget* é desenhado com base no padrão de projeto de *software* **MVC**, constituído de dois componentes distintos, **M**odelo de dados e **V**isualização ou aparência gráfica, e um componente de **C**ontrole ou modo de interação. Uma barra de rolagem é, por exemplo, um widget que tem como modelo, um intervalo de valores numéricos dependents do aplicativo, como aparência um deslizador, e como controle, a possibilidade de um usuário "deslocar" o botão do deslizador que pode ser traduzido, por exemplo, como "rolar" uma área de dimensões maiores do que a da janela onde ela é exibida (Figura 2.14.(a)). Em especial, o widget de tela (Canvas widget) é um componente que tem aparência de uma tela e provê facilidades gráficas 2D e/ou 3D para criar e manipular livremente os modelos da dados do aplicativo (Figura 2.14.(a)). Entre as bibliotecas de widgets amplamente utilizadas temos GTK+, wxWidgets, Qt e Swing da linguagem de programação Java.

Nesta disciplina utilizaremos a biblioteca de widgets GLUT (*OpenGL Utility Toolkit*) para implementar simples interfaces de aplicativos gráficos 3D. Além de ser independente do sistema de janelas, GLUT provê um widget de tela capaz de se comunicar a interface de programação de aplicativos OpenGL e utiliza o modelo de callback para registrar os tratadores de eventos, como ilustra o seguinte código extraído do livro Redbook. O programa desenha um quadrado branco na janela com uso da interface OpenGL (funções gl*) quando ele recebe um evento de janela. O arquivo glut.h deve ser sempre incluído para definir os macros e constantes utilizados. Com uso das funções glutInitWindowSize(), glutInitWindowPosition() e glutCreateWindow(), foi definida uma janela "hello" de tamanho 250×250 em $(100, 100)$ que se co-

munica com API OpenGL (Figura 2.15) e registrado o evento de criar esta janela. Somente o evento de janela não é mascarado e o seu tratador display() é registrado através da função glutDisplayFunc(). O tratador simplesmente desenha um quadrado branco com os comandos de OpenGL quando a janela sofre alguma alteração. O laço principal de eventos glutMainLoop() fica "aguardando" tais eventos. Ressaltamos que a janela definida só aparece na tela quando entra no laço principal de eventos e o evento de "criar a janela" é processado.

```
#include <GL/glut.h>

void display(void)
{
/* clear all pixels  */
   glClear (GL_COLOR_BUFFER_BIT);

/* draw white polygon (rectangle) with corners at
 * (0.25, 0.25, 0.0) and (0.75, 0.75, 0.0)
 */
   glColor3f (1.0, 1.0, 1.0);          // cor branca
   glBegin(GL_POLYGON);
      glVertex3f (0.25, 0.25, 0.0);
      glVertex3f (0.75, 0.25, 0.0);
      glVertex3f (0.75, 0.75, 0.0);
      glVertex3f (0.25, 0.75, 0.0);
   glEnd();

/* don't wait!
 * start processing buffered OpenGL routines
 */
   glFlush ();
}

void init (void)
{
/* select clearing color  */
   glClearColor (0.0, 0.0, 0.0, 0.0);

/* initialize viewing values  */
   glMatrixMode(GL_PROJECTION);
```

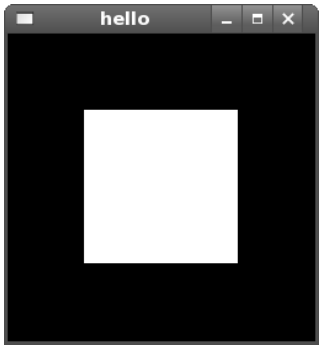


Figura 2.15: hello.c.

```
   glLoadIdentity();
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}


/*
 * Declare initial window size, position, and display mode
 * (single buffer and RGBA).  Open window with "hello"
 * in its title bar.  Call initialization routines.
 * Register callback function to display graphics.
 * Enter main loop and process events.
 */
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (250, 250);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("hello");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;   /* ANSI C requires main to return int. */
}
```

Capítulo 3

Modelagem Geométrica

O objetivo deste capítulo é apresentar para você distintas representações de uma figura geométrica por meio de valores numéricos, através dos quais consegue-se caracterizá-la e desenhá-la em sistemas (digital) de informação gráfica. Portanto, após a leitura deste capítulo, você deve ser capaz de

- distinguir o conceito de pontos e o conceito de vetores.
- distinguir as diferentes representações de um um ponto e de um vetor no espaço.
- distinguir diferentes formas para descrever uma figura geométrica e derivar através delas os vetores normais.
- aproximar uma figura geométrica por malhas poligonais.

Modelos geométricos são fundamentais tanto em síntese quanto em análise de imagens. Para síntese de imagens, que consiste em determinar a luminância/brilhância das superfícies presentes numa "cena sintética", no mínimo dois dados geométricos sobre cada ponto visível de superfícies devem ser conhecidos: a sua **posição** e o **vetor normal** da superfície. Um dos focos de pesquisa na área de Modelagem Geométrica é prover mecanismos para modelar, armazenar, e manipular os objetos sintéticos em sistemas computacionais. Quando se trata de análise de imagens, ou seja extrair informação sobre uma cena a partir das suas imagens "reais" capturadas pelos sensores, informação geométrica, como variação de vetores normais na vizinhança de um ponto, e fotométrica tem sido muito utilizada. Pesquisa

na área de Modelagem Geométrica tem contribuída com algoritmos de inferência de forma geométrica de um objeto presente numa cena a partir dos dados geométricos mensuráveis localmente, como posição e vetor normal.

Há uma grande variedade de modelos geométricos para representar formas geométricas processáveis pelos sistemas digitais. Uns modelos, como representações por funções, são considerados analiticamente precisos; enquanto outros, como aproximações por malhas poligonais, são mais apropriados para renderização eficiente. Há ainda modelos orientados a usuários menos experientes, como representação por CSG (*Constructive Solid Geometry*) que permite construção de figuras geométricas complexas a partir de um *kit* de formas básicas. Tais formas básicas podem ser descritas tanto por funções como por malhas poligonais. Uma característica comum destes modelos é a representação dos pontos por uma n-tupla de números. Isso nos permite associar uma figura geométrica a um sistema de equações/inequações e processá-la com uso de operações aritméticas tradicionais.

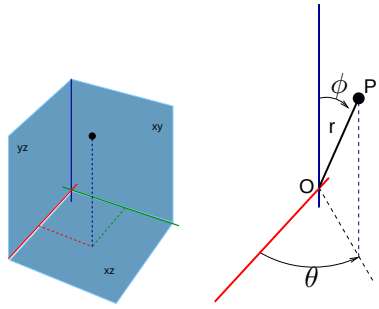
3.1 Pontos e Vetores

Em sistemas de informação gráfica uma **posição** no espaço é denominada um **ponto**, enquanto o deslocamento de um ponto P_B em relação a um outro ponto P_A é representado pelo **vetor** (de deslocamento), que é uma grandeza física provida de magnitude (módulo), direção e sentido. Ponto é, portanto, uma "identificação" de um local para a qual não tem sentido aplicar operações aritméticas. Veremos na seção 3.1.1 que se pode associar, de forma biunívoca, uma posição do espaço por uma n-tupla de valores numéricos (escalares). Quando esses valores numéricos correspondem às coordenadas cartesianas ou afins, podemos obter o **vetor** (de deslocamento) entre P_A e P_B por meio da subtração entre as suas coordenadas

$$\mathbf{v} = P_B - P_A. \tag{3.1}$$

Observe que existe uma infinidade de pares de pontos cuja diferença resulta em $\mathbf{v}$. A escolha do ponto-origem P_A e do ponto-extremidade P_B é irrelevante na especificação de um vetor (de deslocamento).

Diferentemente do espaço de pontos, são definidas no espaço de vetores operações de multiplicação por escalar, adição, subtração, produto escalar, produto vetorial. Algumas dessas operações serão detalhadas na seção 3.1.3.



(a) (b)

Figura 3.1: Coordenadas: (a) cartesianas e (b) polares.

3.1.1 Representação

Um ponto de uma figura geométrica em um espaço de dimensão n pode ser representado por uma lista de n valores numércios, denominados **coordenadas**, $n-1$, $(x_1, x_2, \cdots, x_n)$. Em sistemas de informação gráfica é conveniente descrever esta n-tupla de números por matriz-linha

$$\begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \end{bmatrix}$$

ou por matriz-coluna

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}.$$

A forma mais usual para identificar um ponto é utilizar valores correspondentes às suas distâncias (com sinal) em relação a um conjunto de hiperplanos de dimensão $n-1$. Tal sistema é conhecido como **sistema de coordenadas retangulares** ou **cartesianas** e as coordenadas são conhecidas por **coordenadas cartesianas**. Figura 3.1.(a) mostra as coordenadas cartesianas x, y e z de um ponto em relação aos planos yz, xz e xy, respectivamente.

Para problemas que envolvem direções variadas em relação a um ponto O fixo (**pólo**), é conveniente especificar um ponto P em $\Re^3$ com uso de **coordenadas polares** (r, θ, ϕ), onde r é a distância entre os pontos O e P, e θ e ϕ são ângulos entre a direção $\overrightarrow{OP}$ e dois eixos (de referência) que passam por O, como ilustra Figura 3.1.(b).

Considerando que os eixos de referência sejam, respectivamente, as intersecções $x = xy \cap xz$ e $z = xz \cap yz$ dos "planos de referência cartesianos", a projeção do raio r sobre o eixo z ($r\ cos\phi$) corresponde à coordenada z do ponto e a projeção do raio r sobre o plano xy é a hipotenusa do triângulo de catetos iguais aos valores das coordenadas x e y do ponto, isto é,

$$x = r\ cos\theta\ sen\phi \quad y = r\ sen\theta\ sen\phi \quad z = r\ cos\phi. \tag{3.2}$$

Das coordenadas cartesianas pode-se também derivar as coordenadas polares. Observe que a distância entre o ponto (x, y, z) e a origem (pólo) é o raio r, o ângulo entre r e o eixo z é ϕ, e o ângulo entre a projeção do raio r sobre o plano xy e o eixo x é θ

$$r = \sqrt{x^2 + y^2 + z^2} \quad \phi = arccos\frac{z}{r} \quad \theta = arctg\frac{y}{x} \tag{3.3}$$

Num sistema cartesiano o **vetor** de deslocamento entre dois pontos quaisquer $P_1 = (x_1, y_1, z_1)$ e $P_2 = (x_2, y_2, z_2)$ pode ser representado pela diferença entre suas coordenadas (distâncias)

$$\mathbf{v}_{12} = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} - \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{bmatrix}.$$

A **magnitude** de $\mathbf{v}_{12}$, ou distância entre P_1 e P_2, é

$$|\mathbf{v}_{12}| = d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}. \tag{3.4}$$

Observe que se o ponto-origem P_1 estiver na origem do sistema de referência, as coordenadas do vetor de deslocamento $\mathbf{v}_{12}$ coincidem com as coordenadas do ponto P_2. Embora P_2 e $\mathbf{v}_{12}$ sejam idênticos numericamente, eles são distintos sementicamente: um é simplesmente uma "identidade" do ponto no espaço e o outro, o seu deslocamento em relação à origem do sistema de referência. Portanto, somente sobre $\mathbf{v}_{12}$ são definidas as operações de adição e multiplicação por escalar e ele é conhecido como **vetor-posição** do ponto P_2.

Introduzindo o conceito de vetor-posição, podemos então modelar um conjunto de pontos como um espaço vetorial. Para gerar um espaço 3D, são necessários três vetores linearmente independentes. Agora, podemos apresentar uma terceira alternativa para representar um vértice P no espaço $\Re^3$: fixar uma origem O e uma base de 3 vetores quaisquer linearmente independentes, $\overrightarrow{v}_{(1)}$, $\overrightarrow{v}_{(2)}$ e $\overrightarrow{v}_{(3)}$, e considerá-lo como um vetor-posição resultante da combinação linear destes três vetores-base

$$\overrightarrow{OP} = P - O = \alpha_1 \overrightarrow{v}_{(1)} + \alpha_2 \overrightarrow{v}_{(2)} + \alpha_3 \overrightarrow{v}_{(3)}. \tag{3.5}$$

Os valores $(\alpha_1, \alpha_2, \alpha_3)$ são denominados **coordenadas afins**.

O deslocamento entre dois pontos P_1 e P_2 pode ser obtido como diferença entre dois vetores-posição $\overrightarrow{OP_1}$ e $\overrightarrow{OP_2}$

$$\mathbf{v}_{12} = \overrightarrow{OP_2} - \overrightarrow{OP_1} = P_2 - O - P_1 + O = P_2 - P_1$$

e as coordenadas afins do vetor correspondem ao resultado da subtração das coordenadas afins de P_1 e P_2

$$\begin{aligned} \mathbf{v}_{12} &= \overrightarrow{OP_2} - \overrightarrow{OP_1} \\ &= (\alpha_{12}\mathbf{v}_{(1)} + \alpha_{22}\mathbf{v}_{(2)} + \alpha_{32}\mathbf{v}_{(3)}) - (\alpha_{11}\mathbf{v}_{(1)} + \alpha_{21}\mathbf{v}_{(2)} + \alpha_{31}\mathbf{v}_{(3)}) \\ &= (\alpha_{12} - \alpha_{11})\mathbf{v}_{(1)} + (\alpha_{22} - \alpha_{21})\mathbf{v}_{(2)} + (\alpha_{32} - \alpha_{31})\mathbf{v}_{(3)}. \end{aligned}$$

Em particular, se escolhermos uma **base canônica**, $\mathbf{x} = (1, 0, 0)$, $\mathbf{y} = (0, 1, 0)$ e $\mathbf{z} = (0, 0, 1)$, cada vetor-posição (x, y, z) pode ser expresso como uma combinação linear desta base canônica

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = x \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + y \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + z \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Observe que os escalares (x, y, z) correspondem exatamente às coordenadas cartesianas. Assim, ao invés de pensar em planos de referência, podemos adotar a base canônica, constituída pelos eixos $\mathbf{x}$, $\mathbf{y}$ e $\mathbf{z}$, como uma referência Modelando o referencial com uso de vetores, podemos ainda distinguir duas orientações:

- orientação mão-direita: ao rodarmos os dedos da mão-direita partindo-se do vetor $\mathbf{x}$ para o vetor $\mathbf{y}$, o polegar aponta para a direção positiva do vetor $\mathbf{z}$ (Figura 3.2.(a)).
- orientação mão-esquerda: ao rodarmos os dedos da mão-esquerda partindo-se do vetor $\mathbf{x}$ para o eixo $\mathbf{y}$, o polegar aponta para a direção positiva do eixo $\mathbf{z}$ (Figura 3.2.(b)).

Exemplo 3.1 *Dado um cubo de lado igual a 2 unidades, conforme Figura 3.3. Quais são as coordenadas cartesianas dos vértices do cubo centrado na origem de um sistema de referência cartesiano? E as coordenadas afins, considerando que a origem e os vetores de base sejam coincidentes com os do sistema de referência cartesiano? E as coordenadas polares, considerando que o pólo esteja na origem do referencial cartesiano e os eixos coincidentes com os eixos x e z?*

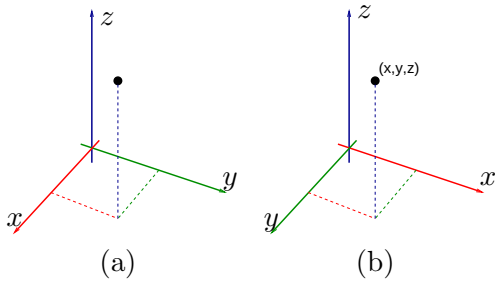


Figura 3.2: Sistema de coordenadas cartesianas: (a) mão-direita; (b) mão-esquerda.

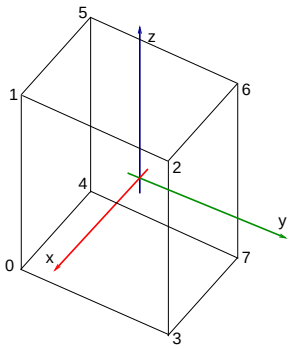


Figura 3.3: Cubo.

Índice de vértice	*Cartesianas*	*Afins*	*Polares*
0	*(1.0, -1.0, -1.0)*	*(1.0, -1.0, -1.0)*	*($\sqrt{3}$, 125.26, -45)*
1	*(1.0, -1.0, 1.0)*	*(1.0, -1.0, 1.0)*	*($\sqrt{3}$, 54.736, -45)*
2	*(1.0, 1.0, 1.0)*	*(1.0, 1.0, 1.0)*	*($\sqrt{3}$, 54.736, 45)*
3	*(1.0, 1.0, -1.0)*	*(1.0, 1.0, -1.0)*	*($\sqrt{3}$, 125.26 , 45)*
4	*(-1.0, -1.0, -1.0)*	*(-1.0, -1.0, -1.0)*	*($\sqrt{3}$, 125.26, 45)*
5	*(-1.0, -1.0, 1.0)*	*(-1.0, -1.0, 1.0)*	*($\sqrt{3}$, 54.736, 45)*
6	*(-1.0, 1.0, 1.0)*	*(-1.0, 1.0, 1.0)*	*($\sqrt{3}$, 54.736 , -45)*
7	*(-1.0, 1.0, -1.0)*	*(-1.0, 1.0, -1.0)*	*($\sqrt{3}$, 125.26 , -45)*

3.1.2 Combinação Convexa

Um conceito muito útil em sistemas de informação gráfica é **combinação convexa**. A combinação convexa de um conjunto de m vetores P_0P_i, em coordenadas afins, é uma soma ponderada destes vetores tal que os pesos α_i sejam não-negativos e a soma desses pesos seja igual a 1

$$\begin{aligned}\overrightarrow{P_0P} &= P - P_0 = \sum \alpha_i (P_i - P_0) = \sum \alpha_i P_i - \sum \alpha_i P_0 = \sum \alpha_i P_i - P_0, \\ &\alpha_i \geq 0 \;\; e \;\; \sum \alpha_i = 1,\end{aligned}$$

da qual derivamos uma expressão que, aparentemente, contradiz o que afirmamos na seção 3.1: soma de pontos!

$$P = \sum \alpha_i P_i \quad , \alpha_i \geq 0 \;\; e \;\; \sum \alpha_i = 1. \tag{3.6}$$

Isso significa que, dado um conjunto de pontos P_i, o ponto P pode ser determinado, de forma única e independente do referencial, pelos pesos $(\alpha_1, \alpha_2, \ldots, \alpha_m)$. Estes pesos são denominados **coordenadas baricêntricas** em relação aos pontos de referência P_i.

Uma propriedade importante de combinação convexa é que o ponto P está sempre contido no "interior" do **fecho convexo** do conjunto de pontos P_i e o lugar geométrico de todos os pontos gerados pela Eq.(3.6) é o próprio fecho. Um **segmento** $\overline{P_AP_B}$ pode ser, portanto, representado algebricamente como o lugar geométrico de um conjunto de pontos, cujas coordenadas baricêntricas (α_1, α_2) satisfazem duas propriedades: $0 \leq \alpha_1, \alpha_2 \leq 1$ e $\alpha_1 + \alpha_2 = 1$. Podemos "ver" um **triângulo** $P_AP_BP_C$ como o lugar geométrico de todas as possíveis combinações convexas dos seus três vértices P_A, P_B e P_C, isto é, pontos de coordenadas baricêntricas $(\alpha_1, \alpha_2, \alpha_3)$ em relação a P_A, P_B e P_C que satisfazem as propriedades $0 \leq \alpha_1, \alpha_2, \alpha_3 \leq 1$ e $\alpha_1 + \alpha_2 + \alpha_3 = 1$

Exemplo 3.2 *As coordenadas baricêntricas do* **ponto médio** *de um segmento são* $(\frac{1}{2}, \frac{1}{2})$ *e as coordenadas baricêntricas do* **baricentro** *de um triângulo são* $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$*. Se as coordenadas cartesianas dos pontos extremos do segmento forem* $P_1 = (0, 5)e\ P_2 = (3, 7)$*, as coordenadas cartesianas do ponto médio deste segmento são* $(1.5, 6.0)$*. Com mais um ponto não colinear* $P_3 = (3, 10)$*, define-se um triângulo cujo baricentro tem as coordenadas cartesianas* $P = \frac{1}{3}P_1 + \frac{1}{3}P_2 + \frac{1}{3}P_3 = (2, 11)$*. Se alterarmos as coordenadas dos pontos* P_1*,* P_2 *e* P_3 *teremos outras figuras geométricas, mas as coordenadas baricêntricas do ponto médio e do baricentro serão sempre as mesmas.*

3.1.3 Operações com Vetores

Dados dois vetores $\mathbf{v}_1 = (x_1, y_1, z_1)$ e $\mathbf{v}_2 = (x_2, y_2, z_2)$ num sistema cartesiano $\Re^3$, são comuns as seguintes operações em sistemas de informação gráfica:

Multiplicação por escalar: $\mathbf{v} = \alpha\mathbf{v}_1 = (\alpha x_1, \alpha y_1, \alpha z_1)$

Adição: $\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2 = (x_1 + x_2, y_1 + y_2, z_1 + z_2)$

Subtração: $\mathbf{v} = \mathbf{v}_1 - \mathbf{v}_2 = (x_1 - x_2, y_1 - y_2, z_1 - z_2)$

Produto Escalar: $\mathbf{v}_1 \cdot \mathbf{v}_2 = |\mathbf{v}_1||\mathbf{v}_1| cos\ \theta = x_1x_2 + y_1y_2 + z_1z_2$, onde θ é o ângulo entre os dois vetores. Uma interpretação geométrica do produto escalar é projeção do vetor $\mathbf{v}_1$ sobre $\mathbf{v}_2$, ou vice-versa. O produto escalar é também muito utilizado para obter o ângulo entre dois vetores. Se $\mathbf{v}_1$ e $\mathbf{v}_2$ forem normalizados, ou seja $|\mathbf{v}_1| = |\mathbf{v}_2| = 1$, então $\mathbf{v}_1 \cdot \mathbf{v}_2 = cos\ \theta$.

Produto Vetorial: $|\mathbf{v}_1 \times \mathbf{v}_2| = |\mathbf{v}_1||\mathbf{v}_1| sen\ \theta$ e $\mathbf{v}_1 \times \mathbf{v}_2 = (y_1z_2 - y_2z_1, x_2z_1 - x_1z_2, x_1y_2 - x_2y_1)$, onde θ é o ângulo entre os dois vetores. Observe que a área do triângulo formado pelos dois vetores $\mathbf{v}_1$ e $\mathbf{v}_2$ é $\frac{1}{2}|\mathbf{v}_1 \times \mathbf{v}_2|$.

3.2 Figuras Geométricas

Nesta altura você deve estar se perguntando para quê servem tantas formas distintas de representar pontos no espaço por n-tupla de valores numéricos, se o nosso interesse é na modelagem de formas geométricas em sistemas de informação gráfica. Isso é porque uma figura geométrica pode ser caracterizada como um conjunto de pontos do espaço que satisfazem um certo número de propriedades e muitas destas propriedades são descritíveis por equações

e inequações que relacionem as coordenadas que apresentamos. Isso certamente facilita o seu processamento digital. Por exemplo, um triângulo definido pelos pontos $P_A = (1, 0)$, $P_B = (0, 1)$ e $P_C = (0, 0)$ pode ser representado

- em coordenadas cartesianas como um sistema constituído pelas três equações de semi-planos:

$$\begin{cases} x \geq 0 \\ y \geq 0 \\ x + y - 1 \leq 0 \end{cases}$$

- em coordenadas polares: $(r,\, \theta)$, com $r \in [0,\, \frac{1}{(\cos\theta + \sin\theta)}]$, $\theta \in [0,\, \frac{\pi}{2}]$;
- em coordenadas afins: se for na base canônica, a representação é a mesma em coordenadas cartesianas; e
- em coordenadas baricêntricas: $P(t_1, t_2, t_3) = t_1 P_A + t_2 P_B + t_3 P_C$, como vimos na seção 3.1.2.

Na seção 3.1.2 vimos as restrições que devem ser impostas sobre as coordenadas bricêntricas para obter um segmento e um triângulo. Na seção 3.2.1 são apresentadas mais funções que descrevem a relação das coordenadas cartesianas dos pontos no espaço $\Re^3$ pertencentes a uma figura geométrica de interesse, na seção 3.2.2 é detalhada a representação de curvas e superfícies de Bézier e na seção 3.2.3 são apresentadas formas distintas para derivar vetores normais das figuras geométricas a partir das suas formulações analíticas. Estes vetores normais são essenciais na síntese de imagens foto-realísticas.

3.2.1 Representação de Propriedades Geométricas

Em Geometria Analítica é comum descrever as propriedades que um conjunto de pontos de uma figura geométrica deve satisfazer por meio da relação das suas coordenadas cartesianas. Um exemplo simples é a circunferência. Ela é o lugar geométrico de todos os pontos, em coordenadas cartesianas (x, y), que distam de um ponto (x_c, y_c) denominado centro, de um valor R, isto é,

$$(x - x_c)^2 + (y - y_c)^2 = R^2,$$

ou seja,

$$f(x, y) = (x - x_c)^2 + (y - y_c)^2 - R^2 = 0.$$

Dentre as representações com uso de n coordenadas cartesianas, distinguem-se:

impícitas : a relação entre as n coordenadas dos pontos do objeto é expressa por (in)equações de n variáveis.

explícitas : uma coordenada é expressa explicitamente em função de todas as outras $n-1$ coordenadas.

É possível ainda especificar uma figura geométrica através de um domínio de parâmetros definido arbitrariamente, desde que cada n-tupla de valores destes parâmetros corresponda a um ponto da figura. Neste caso, uma figura geométrica é "vista" como uma aplicação de um domínio de parâmetros em pontos no espaço $\Re^3$, representados por coordenadas cartesianas. Se a figura geométrica for uma curva, o domínio de parâmetros é univariável. Se a figura for uma superfície, o domínio é bi-dimensional contendo dois parâmetros. Esta forma de representação de figuras geométricas é conhecida como **representação paramétrica**.

Seguem-se a representação paramétrica de algumas superfícies quádricas conhecidas:

- Esfera: $P(\theta,\phi) = (rcos\theta, rsen\theta cos\phi, rsen\theta sen\phi)$, $0 \leq \theta \leq \pi$, $0 \leq \phi \leq 2\pi$,
- Elipsóide: $P(\theta,\phi) = (acos\theta, bsen\theta cos\phi, bsen\theta sen\phi)$, $0 \leq \theta \leq \pi$, $0 \leq \phi \leq 2\pi$,
- Toro: $P(\theta,\phi) = (h + acos\theta, (k + bsen\theta)cos\phi, (k + bsen\theta)sen\phi)$, $0 \leq \theta \leq 2\pi$, $0 \leq \phi \leq 2\pi$,
- Paraboloide: $P(\theta,\phi) = (au^2, 2aucos\phi, 2ausen\phi)$, $0 \leq u \leq u_{max}$, $0 \leq \phi \leq 2\pi$, e
- Cilindro: $P(\theta,\phi) = (rcos\theta, rsen\theta, u)$, $0 \leq u \leq u_{max}$, $0 \leq \theta \leq 2\pi$.

Exemplo 3.3 *Um segmento (curva) é o lugar geométrico do conjunto de pontos cujas coordenadas cartesianas satisfazem as seguintes propriedades*

- *em representação paramétrica:* $P(t) = (t, 2t, t)$ *e* $0 \leq t \leq 1.0$.
- *em representação implícita:* $f(x,y,z) = x + z - y = 0$, $0 \leq x \leq 1.0$.
- *em representação explícita:* $y = x + z$, $0 \leq x \leq 1.0$.

Uma esfera de raio r (superfície) é o lugar geométrico de pontos cujas coordenadas cartesianas satisfazem as seguintes propriedades

- *em representação paramétrica:* $P(\theta, \phi) = (r\ cos\theta\ sen\phi, r\ sen\theta\ sen\phi, r\ cos\phi)$, $0 \leq \theta \leq 2\pi\ \ e\ -\frac{\pi}{2} \leq \phi \leq \frac{\pi}{2}$.
- *em representação implícita:* $f(x, y, z) = x^2 + y^2 + z^2 - r^2$.
- *em representação explícita:* $z = \pm\sqrt{r^2 - x^2 - y^2}$.

A representação implícita f de uma figura geométrica é muito utilizada para **classificação de pertinência** de um ponto P em relação à figura, pois se substituirmos as coordenadas do ponto na função f, temos três possíveis situações:

1. $f < 0$: P está em um lado (interior) da figura.
2. $f = 0$: P está sobre a figura, ou seja, é um ponto da figura.
3. $f > 0$: P está em outro lado (exterior) da figura.

3.2.2 Funções de Bernstein

Para ilustrar a aplicação do conceito de combinação convexa em modelagem de figuras geométricas mais complexas, são apresentadas nesta seção as **curvas** e **superfícies de Bézier**.

Geometricamente, os pontos sobre as curvas de Bézier, de grau n, $P(t), t \in [0, 1]$ são obtidos através da combinação convexa de um conjunto de pontos $\{P_0, P_1, \cdots, P_n\}$, denominados **pontos de controle**, com uso de funções de Bernstein

$$P(t) = \sum_{i=0}^{n} B_{n,i}(t) P_i.$$

É comum denominar a seqüência dos pontos de controle de **polígono de controle**.

Os termos $B_{n,i}(t)$ das funções de Bernstein satisfazem a igualdade

$$((1-t)+t)^n = \sum_{i=0}^{n} B_{n,i}(t) = \sum_{i=0}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) t^i (1-t)^{n-i} = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i} = 1 \quad \forall t.$$

Uma curva de Bézier de grau n é, de fato, o lugar geométrico de um subconjunto dos pontos do fecho convexo de $n+1$ pontos de controle P_i. Os fatores de ponderação $B_{n,i}(t)$ correspondem às coordenadas baricêntricas do ponto $P(t)$ em relação a estes pontos de controle, como ilustra Figura 3.4.

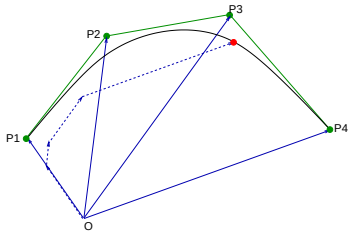


Figura 3.4: Curva de Bézier.

Quando $n = 2$, as funções

$$\begin{aligned} B_{2,0}(t) &= (1-t)^2 = 1 - 2t + t^2 \\ B_{2,1}(t) &= 2t(1-t) = 2t - 2t^2 \\ B_{2,2}(t) &= t^2 \end{aligned}$$

constituem uma base do espaço de polinômios de grau 2. Dados três pontos P_0, P_1 e P_2, obtém-se uma curva de Bézier quadrática

$$P(t) = (1 - 2t + t^2)P_0 + (2t - 2t^2)P_1 + (t^2)P_2.$$

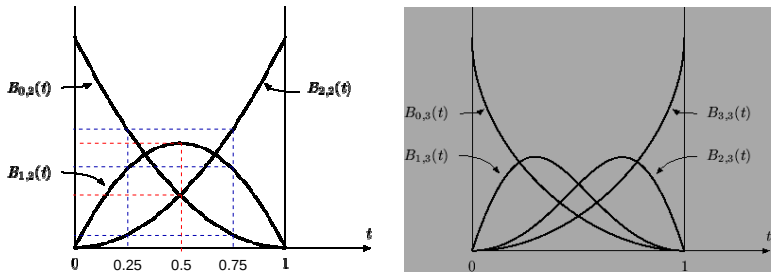
Utilizando a notação matricial,

$$P(t) = \begin{bmatrix} B_{2,0}(t) & B_{2,1}(t) & B_{2,2}(t) \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix}$$

A matriz quadrada 2×2 é conhecida como **matriz-base de Bézier quadrática** e o vetor de pontos de controle, **vetor de geometria de Bézier**. Figura 3.5.(a) mostra o gráfico das funções de (base) $B_{2,0}(t)$, $B_{2,1}(t)$ e $B_{2,2}(t)$ no intervalo $t \in [0, 1]$. Observe que neste intervalo as funções só assumem valores não nulos em $t = 0$ e $t = 1$. Quando $t = 0$ as coordenadas baricêntricas do ponto correspondente são $(1, 0, 0)$ e quando $t = 1$ as coordenadas baricêntricas passam a ser $(0, 0, 1)$. Para o restante dos valores de t, as coordenadas baricêntricas não se anulam. Por exemplo, para $t = 0.5$ e $t = 0.25$, as coordenadas baricêntricas dos pontos correspondentes são $(0.25, 0.5, 0.25)$ e $(0.5625, 0.375, 0.0625)$, respectivamente. Isso significa que se alterarmos a posição de um ponto de controle P_i, todos os pontos da curva se "mexerão".

As curvas de Bézier cúbicas são as mais utilizadas dentre as curvas de Bézier. As funções de Bernstein que as definem são

$$B_{3,0}(t) = (1-t)^3$$



(a) (b)

Figura 3.5: Funções de Bernsteins: (a) quadráticas e (b) cúbicas.

$$\begin{aligned} B_{3,1}(t) &= 3t(1-t)^2 \\ B_{3,2}(t) &= 3t^2(1-t) \\ B_{3,3}(t) &= t^3. \end{aligned}$$

Elas constituem uma base do espaço de polinômios de grau 3.

Utilizando a notação matricial,

$$\begin{aligned} P(t) &= \left[\begin{array}{cccc} B_{3,0}(t) & B_{3,1}(t) & B_{3,2}(t) & B_{3,3}(t) \end{array}\right] \left[\begin{array}{c} P_0 \\ P_1 \\ P_2 \\ P_3 \end{array}\right] \\ &= \left[\begin{array}{cccc} t^3 & t^2 & t & 1 \end{array}\right] \left[\begin{array}{cccc} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} P_0 \\ P_1 \\ P_2 \\ P_3 \end{array}\right]. \end{aligned}$$

A matriz quadrada 3×3 é conhecida como **matriz-base de Bézier cúbica**. Figura 3.5.(b) mostra o gráfico das funções de (base) Bernstein cúbicas no intervalo $t \in [0, 1]$.

É comum "emendar" as curvas de Bézier cúbicas para construir um *spline* de geometria complexa. Para assegurar uma "emenda suave" exige-se tipicamente que, além dos pontos-extremo serem os mesmos, a direção dos vetores-tangente nos pontos de "emenda" seja a mesma, como no ponto P da Figura 3.6. Como se determina os vetores-tangente a uma curva em um ponto $P(t)$? Basta calcular a derivada no ponto

$$P'(t) = \sum_{i=0}^{n} B'_{n,i}(t) P_i.$$

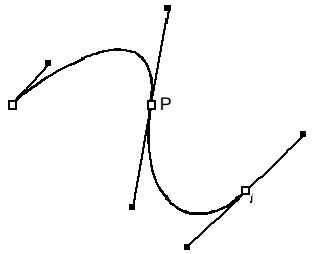


Figura 3.6: *Spline* constituído por 2 segmentos de curva cúbicos.

Particularmente, para uma curva cúbica, o vetor-tangente em cada ponto $P(t)$ é dado por

$$P'(t) = (-3 + 6t - 3t^2)P_0 + (3 - 3t^2)P_1 + (6t - 9t^2)P_2 + (3t^2)P_3$$

Em notação matricial,

$$P'(t) \;=\; \left[\begin{array}{cccc} t^3 & t^2 & t & 1 \end{array}\right] \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ -3 & -3 & -9 & 3 \\ 6 & 0 & 6 & 0 \\ -3 & 3 & 0 & 0 \end{array}\right] \left[\begin{array}{c} P_0 \\ P_1 \\ P_2 \\ P_3 \end{array}\right].$$

Podemos obter superfícies de Bézier $P(u, v)$ com uso de funções de Bernstein através da combinação convexa dos pontos resultantes da combinação convexa dos pontos de controle (Figura 3.7)

$$P(u, v) = \sum_{i=0}^{m} (\sum_{j=0}^{n} P_{ij} B_{n,j}(u)) B_{m,i}(v). \tag{3.7}$$

O reticulado de pontos de controle P_{ij} definem a **malha de controle** da superfície.

3.2.3 Vetores Normais

O vetor normal **n** de uma superfície em cada ponto é essencial na síntese foto-realística de figuras geométricas.

Quando se conhece a função da figura geométrica de interesse, pode-se derivar a partir de função o vetor normal exato em cada ponto:

Representação implícita f(x,y,z): $\mathbf{n} = \frac{\nabla f}{|\nabla f|}$, onde $\nabla f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})$

Representação paramétrica P(u,v): $\mathbf{n} = \frac{\frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v}}{|\frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v}|}$

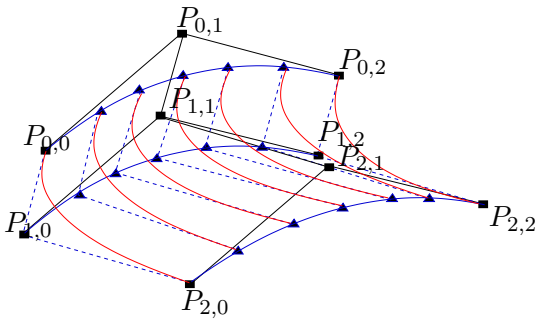


Figura 3.7: Superfície de Bézier.

3.3 Malhas Poligonais

A representação poligonal consiste em descrever um objeto por uma malha de **facetas poligonais**, cada qual é representada por uma sequência fechada e **orientada** de arestas. As **arestas**, por sua vez, podem ser exatamente definidas pelos seus extremos que são conhecidos por **vértices**, cujas posições são reprsentáveis pelas coordenadas. Informação sobre a localização de cada vértice é denominada **geométrica** enquanto informação sobre a conectividade entre eles é usualmente conhecida por **topológica**.

Esta representação é muito utilizada para aproximar figuras geométricas cujos pontos não são descritíveis por uma função conhecida. A estrutura mais simples para armazenar a conectividade entre os vértices é descrever cada faceta poligonal como uma lista de vértices, como ilustra Figura 3.8.

Exemplo 3.4 *As faces do cubo do Exemplo 3.1 podem ser representados como lista dos índices de vértices. Figura 3.8 mostra parcialmente uma possível estrutura para armazenar estes dados.*

Índice de face	*Lista de índice de vértices*
0	*0, 3, 2, 1*
1	*2, 3, 7, 6*
2	*0, 4, 7, 3*
3	*1, 2, 6, 5*
4	*4, 5, 6, 7*
5	*0, 1, 5, 4*

3.3.1 Aproximação Poligonal

Em síntese de imagens, o objetivo é "desenhar" as figuras geométricas de interesse numa tela de exibição. Por questões de eficiência, os tipos de

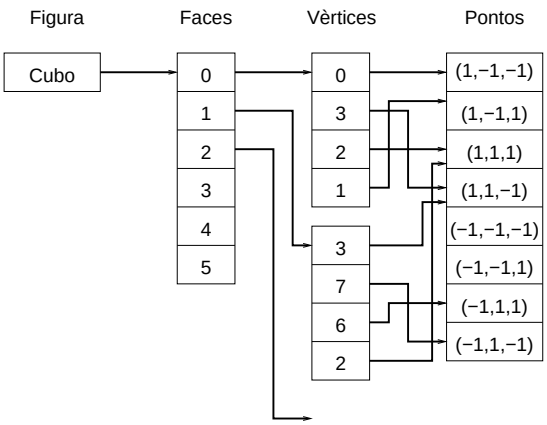


Figura 3.8: Estrutura dos vértices de um cubo.

superfícies suportadas pelas placas de vídeo são ainda limitados à classe de segmentos e polígonos/facetas planas. Portanto, se quisermos desenhar uma figura geométrica, representada por uma função com os recursos disponíveis em uma placa de vídeo, precisamos aproximá-la em uma polilinha ou uma malha poligonal antes do seu envio para o fluxo de renderização suportado por *hardware*.

A aproximação poligonal de curvas e superfícies representadas implicitamente é ainda um problema em aberto, enquanto a aproximação poligonal de figuras representadas parametricamente é relativamente simples. A forma mais trivial é dividir o domínio de parâmetros em intervalos uniformes: no caso de curvas definidas sobre o domínio $t \in [0, 1]$, elas serão aproximadas pela seqüência de pontos $P(0 * \Delta t), P(1 * \Delta t), P(2 * \Delta t), \ldots, P(1)$, e no caso de superfícies definidas sobre o domínio uv, elas serão aproximadas por facetas de tamanho $\Delta u \times \Delta v$ definidas pela seqüência de vértices $r(i\Delta u, j\Delta v)$, $r((i+1)\Delta u, j\Delta v)$, $r((i+1)\Delta u, (j+1)\Delta v)$ e $r(i\Delta u, (j+1)\Delta v)$, onde i e j representam número de passos nas duas ddireções do domínio, conforme ilustra Figura 3.9.

3.3.2 Vetores Normais

Se a figura geométrica é uma malha poligonal cuja faceta é definida pela seqüência de pontos orientados no sentido-anti-horário $v_1 v_2 v_3 \ldots v_n$, o vetor normal de cada faceta pode ser obtido a partir dos três primeiros pontos v_1,

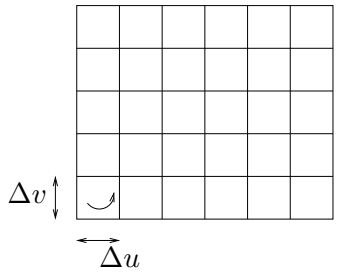


Figura 3.9: Subdivisão do domínio de parâmetros.

v_2 e v_3

$$\mathbf{n} = \frac{\overrightarrow{v_1v_2} \times \overrightarrow{v_2v_3}}{|\overrightarrow{v_1v_2} \times \overrightarrow{v_2v_3}|}.$$

Se a malha poligonal for uma aproximação de uma superfície suave, é comum estimar o vetor normal no vértice P como a normalização da média aritmética dos vetores normais de todas as faces adjacentes a P.

Capítulo 4

Transformações Geométricas

O objetivo deste capítulo é apresentar para você uma forma de manipular uma figura geométrica, seja para alterar suas dimensões seja para reposicioná-la no espaço, por meio de operações sobre valores numéricos. Portanto, após a leitura deste capítulo, você deve ser capaz de

- caracterizar e representar translação, rotação, mudança de escala, cisalhamento e reflexão de pontos.
- caracterizar e representar transformações afins de pontos.
- explicar o papel de cada bloco da matriz de transformação de ordem 4.
- representar pontos por coordenadas homogêneas e aplicá-lo.
- aplicar transformação geométrica sobre vetores.
- definir uma matriz de transformação para reposicionar os pontos de uma figura geométrica no espaço.

Observa atentamente Figura 4.1. O boneco foi construído com uma mesma figura geométrica básica: cubo, conforme a especificação dada no Exemplo 3.4. São 9 blocos. Cada bloco foi posicionado em relação ao bloco central, após as suas dimensões terem sido devidamente ajustadas. Há duas alternativas para obter computacionalmente a imagem apresentada na Figura 4.1: por analogia às técnicas de pintura, desenhando os pontos bi-dimensionais sobre a tela plana; e por analogia às técnicas de fotografia,

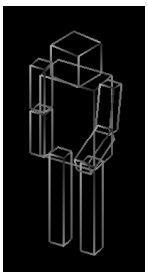


Figura 4.1: Boneco.

"imageando" a maquete construída com figuras geométricas no espaço tridimensional.

A segunda técnica tem sido preferida pelos usuários de sistemas de informação gráfica, por estar mais próxima da sua forma de percepção. Neste caso, a tarefa de produção de uma imagem 2D é de responsabilidade de duas partes: homens e máquinas. É o "processador humano" que define a forma e a disposição espacial das figuras geométricas, enquanto o "processador digital" se encarrega de produzir imagens digitais a partir das definições geradas pelo "processador humano". Para suportar este paradigma, é necessário traduzir a "visão humana" em uma "visão de processador predominantemente numérico". Nós vimos no capítulo 3 como uma máquina pode "ver" as formas geométricas e neste capítulo, veremos como ela pode "entender" os movimentos especificados pelos usuários. Adicionalmente. o sistema deve realizar 3 estágios de **transformação** para produzir uma imagem a partir delas:

1. transformação que **altera** as posições das figuras geométricas no espaço;

2. transformação que **projeta** as figuras geométricas num plano de projeção; e

3. transformação que **mapeia** os pontos do plano de projeção para o referencial da tela de saída do dispositivo de exibição.

Podemos entender aqui uma **transformação** como uma aplicação f que faz corresponder um ponto P do domínio $\mathcal{R}^n$ com um ponto do contradomínio $\mathcal{S}^n$:

$$f : P \in \mathcal{R}^n \mapsto \mathcal{S}^n. \tag{4.1}$$

Teoricamente, tais pontos podem estar representados em coordenadas cartesianas, polares, afins ou baricêntricas (Seção 3.1.1). No entanto, para tirar

proveito dos conceitos e estruturas providos pela Álgebra Linear, os algoritmos de transformação mais conhecidos são desenvolvidos em cima das coordenadas cartesianas.

Uma breve revisão sobre operações com matrizes é dada na seção 4.7. Em seguida, mostramos na seção 4.2 as cinco transformações consideradas básicas: translação (*translation*), rotação (*rotation*), mudança de escala (*scaling*), espelhamento (*reflection*) e deslocamento relativo linear ou cisalhamento (*shearing*). Na seção 4.3 apresentamos as transformações afins. Todas estas transformações podem ser representadas como produtos com matrizes de ordem 4, conforme detalha a seção 4.4. Para adequar a esta representação em matriz de ordem 4, foi necessário representar os pontos em coordenadas homogêneas. Na seção 4.5 faremos uma breve apresentação destas coordenadas. Veremos na seção 4.6 transformações de vetores. Dispondo deste fundamento teórico, abordaremos duas questões práticas. Na seção 4.7 apresentaremos dois paradigmas para definir uma transformação que realiza modificações almejadas e na seção 4.8 mostraremos como simplificar a transformação de todos os pontos de uma figura geométrica aplicando a propriedade de invariância sob transformações afins.

4.1 Operações com Matrizes

Veremos que as transformações sobre os pontos de uma figura geométrica podem ser "vistas" como operações com matrizes. Vamos revisar algumas propriedades destas operações nesta seção.

Dadas duas matrizes $A = (a_{ij})_{m\times p}$ e $B = (b_{ij})_{q\times n}$. A **soma** das duas matrizes A e B é possível, se $m = q$ e $p = n$. Neste caso, o resultado é uma matriz $C = (c_{ij})_{m\times p}$, onde $c_{ij} = a_{ij} + b_{ij}$.A operação satisfaz as seguintes propriedades:

Comutativa: $A + B = B + A$

Associativa: $(A + B) + C = A + (B + C)$

Existência de elemento neutro: $A + 0 = 0 + A = A$

Existência de elemento inverso: $A + (-A) = (-A) + A = 0$

Transposta da soma: $(A + B)^t = A^t + B^t$. A matriz **transposta** de A, indicada por A^t, tem as linhas trocadas ordenadamente pelas colunas de A.

O **produto** das duas matrizes A e B é definido, se $p = q$. O resultado é uma matriz $C = (c_{ij})_{m \times n}$, onde $c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}$. Considere mais uma matriz D, tal que os produtos $B.D$ e $A.D$ sejam definidos . As seguintes propriedades são satisfeitas:

Associativa: $A.(B.D) = (A.B).D$.

Distributiva à direita: $A.(B + D) = A.B + A.D$.

Distributiva à esquerda: $(A + B).D = A.D + B.D$.

Existência de elemento neutro: $A.I = I.A = A$. Uma matriz I de ordem n é uma matriz identidade, se $i_{jk} = 1$, quando $j = k$ e os outros elementos são nulos.

Invariante em relação à prioridade: $A.B.D = (A.B).D = A.(B.D)$.

Transposta do produto: $(A.B)^t = B^t.A^t$.

É importante lembrar que

1. a multiplicação de matrizes **não é comutativa**.
2. na multiplicação de matrizes, podemos ter $A.B = 0$ com $A \neq 0$ e $B \neq 0$.
3. nem todas as matrizes quadradas são inversíveis. Quando uma matriz A é inversível, a sua inversa é indicada por A^{-1} e $A.A^{-1} = I$. Uma técnica muito conhecida para computar uma matriz inversa é o método de eliminação de Gauss.
4. uma matriz **ortogonal** é uma matriz cuja inversa coincide com a sua transposta.

4.2 Transformações Geométricas Básicas

Há duas formas equivalentes de "realizar" uma transformação geométrica, tendo como resultado um mesmo conjunto de pontos: considerar que o referencial dos dois domínios $\mathcal{R}^n$ e $\mathcal{S}^n$ seja o mesmo (Figura 4.2.(a)), ou considerar que cada domínio tenha o seu próprio referencial e que as coordenadas dos pontos em relação ao referencial sejam invariantes (Figura 4.2.(b)).

Apresentamos nesta seção as cinco transformações geométricas básicas. Mostramos ainda que cada uma destas transformações pode ser processado

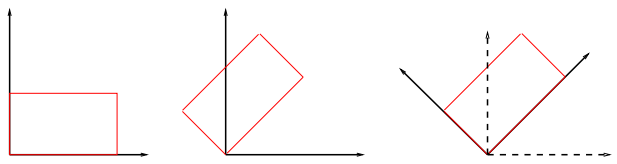


Figura 4.2: Transformações equivalentes: (a) das coordenadas, mantendo referencial fixo; (b) do referencial, mantendo coordenadas fixas.

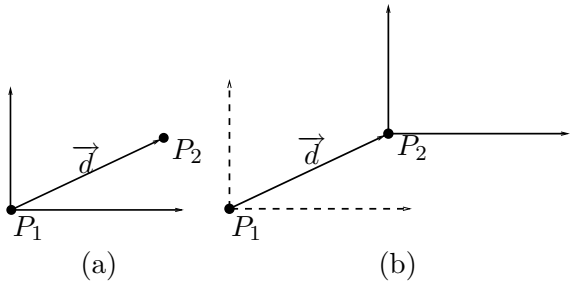


Figura 4.3: Translação ou deslocamento.

como um sistema de equações lineares e este sistema de equações é numericamente idêntico para as duas formas de "ver" uma transformação. Esta constatação pode simplificar a solução de muitos problemas e, em termos de programção, ela nos permite utilizar o mesmo código para processar dois contextos distintos.

4.2.1 Translação

A **translação** de um ponto $P_1 = (x_1, y_1, z_1)$ num espaço tri-dimensional é o **deslocamento** $\overrightarrow{d} = (d_x, d_y, d_z)$ deste ponto para $P_2 = (x_2, y_2, z_2)$. Este deslocamento corresponde à adição do montante d_x, d_y e d_z à coordenada cartesiana x_1, y_1 e z_1, respectivamente (Figura 4.3.(a))

$$\begin{aligned} x_2 &= x_1 + d_x \\ y_2 &= y_1 + d_y \\ z_2 &= z_1 + d_z \end{aligned} \tag{4.2}$$

Usando notação matricial temos

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}. \tag{4.3}$$

Vimos na seção 3.1.1 que as coordenadas cartesianas coincidem com as coordenadas afins quando o referencial é uma base canônica. Vamos utilizar as coordenadas afins para mostrar que, se deslocarmos o referencial pelo vetor de deslocamento $\overrightarrow{d} = (d_x, d_y, d_z)$ "carregando" o ponto P_1, a nova posição espacial do ponto é a mesma dada pela Eq.(4.2), como ilustra Figura 4.3.(b)

$$\begin{aligned}
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 + d_x \\ y_1 + d_y \\ z_1 + d_z \end{bmatrix}. \qquad (4.4)
\end{aligned}$$

4.2.2 Mudança de Escala

A **mudança de escala** de uma figura geométrica implica essencialmente em alteração das suas dimensões, ampliando-a ou reduzinado-a. Em termos de vetores, isso equivale a mudar a magnitude dos pares de vetores definidos pelos pontos da figura geométrica. Dados dois pontos A e B em $\Re^3$ de uma figura. A magnitude do vetor $\overrightarrow{AB}$ é

$$|\overrightarrow{AB}| = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2 + (A_z - B_z)^2}.$$

Mudar a escala do vetor por um fator corresponde a multiplicar $|\overrightarrow{AB}|$ por um escalar s, isto é,

$$s|\overrightarrow{AB}| = s\sqrt{(A_x - B_x)^2 + (A_y - B_y)^2 + (A_z - B_z)^2}.$$

Para simplificar as operações, é comum considerar que o ponto A seja origem, ou seja, $A = (0, 0, 0)$. Neste caso, teremos

$$s|\overrightarrow{AB}| = s\sqrt{(B_x)^2 + (B_y)^2 + (B_z)^2} = \sqrt{(sB_x)^2 + (sB_y)^2 + (sB_z)^2}.$$

Com isso, reduz-se a multiplicação de um vetor por um escalar em multiplicação de um ponto por um escalar! Observe que $(0, 0, 0)$ é um ponto invariante sob a mudança de escala. Figura 4.4.(a) ilustra a multiplicação de um vetor por $s = 2$ tendo como ponto A a origem do referencial.

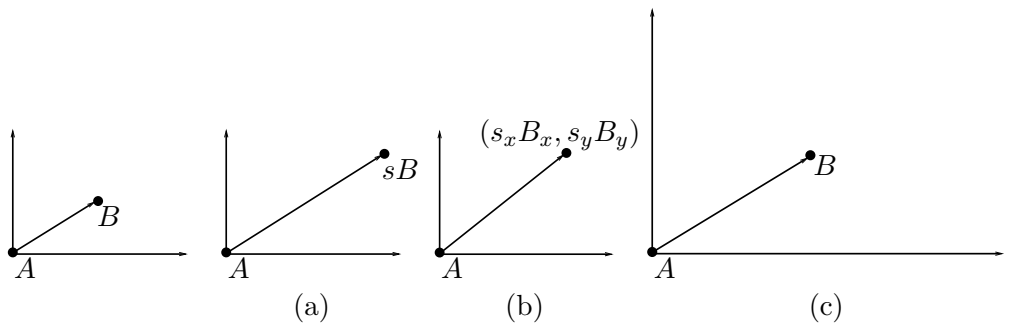


Figura 4.4: Mudança de Escala

Quando o fator de escala é igual para todas as coordenadas, dizemos que a transformação é **uniforme**. Se substituirmos o escalar s pelo vetor (s_x, s_y, s_z) para produzir efeitos diferenciados ao longo dos três eixos de referência, estaremos aplicando uma transformação **não-uniforme**

$$\begin{aligned} x_2 &= s_x x_1 \\ y_2 &= s_y y_1 \\ z_2 &= s_z z_1, \end{aligned} \tag{4.5}$$

como ilustra Figura 4.4.(b) que resultou da aplicação do fator $(1.5, 2)$ sobre o ponto B.

Em notação matricial, a transformação é equivalente a

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}. \tag{4.6}$$

Uma outra forma de pensar seria considerar que os pontos são representados pelas coordenadas afins em relação à base canônica. Se estes vetores forem multiplicados pelos fatores de escala (s_x, s_y, s_z), as novas coordenadas afins, coincidentes com as coordenadas cartesianas, assumem os seguintes valores

$$\begin{aligned} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} &= \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \end{aligned}$$

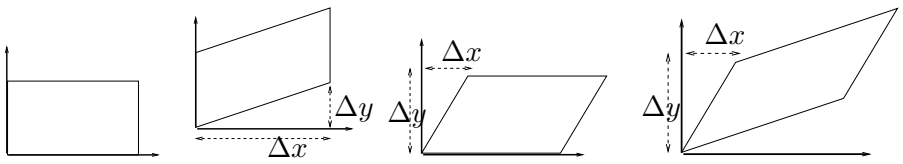


Figura 4.5: Deslocamento Relativo Linear

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x x_1 \\ s_y y_1 \\ s_z z_1 \end{bmatrix},$$

que, numericamente, são iguais aos obtidos pela Eq.(4.5).

4.2.3 Deslocamento Relativo Linear

Esta transformação, conhecida em inglês como *shearing* ou em português como **cisalhamento**, caracteriza-se pela variação linear de uma coordenada do ponto $(x_{1,v}, x_{2,v}, x_{3,v})$ em relação aos valores das outras, ou seja, a nova coordenada transformada $x_{i,r}$ em $\Re^3$ pode ser expressa como:

$$x_{i,r} = x_{i,v} + \sum_{j=1, j\neq i}^{3} (sh_{ij} x_{j,v}), \tag{4.7}$$

onde sh_{ij} é a taxa de variação da coordenada x_i em relação à coordenada x_j. Figura 4.5.(a) e 4.5.(b) ilustram deslocamento na direção y com $sh = \frac{\Delta y}{\Delta x}$ e na direção x com $sh = \frac{\Delta x}{\Delta y}$, respectivamente. Figura 4.5.(c) mostra o resultado conjunto dos dois deslocamentos.

Usando a notação matricial, podemos exprimir o cisalhamento das coordenadas (x_1, y_1, z_1) através do seguinte produto:

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & sh_{xy} & sh_{xz} \\ sh_{yx} & 1 & sh_{yz} \\ sh_{zx} & sh_{zy} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}. \tag{4.8}$$

Se os pontos são representados pelas coordenadas afins em relação à base canônica, podemos obter as mesmas coordenadas cartesianas dadas pela Eq.(4.8) aplicando a transformação sobre os vetores-base e mantendo constantes as coordenadas afins em relação aos novos vetores-base (linha

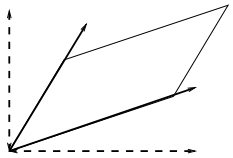


Figura 4.6: Cisalhamento do referencial.

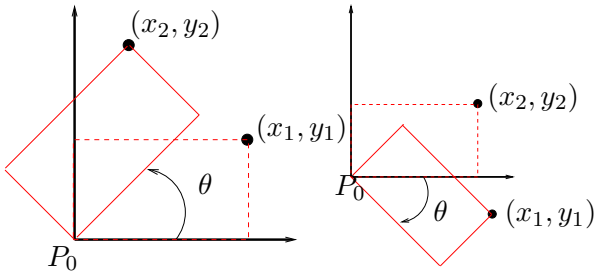


Figura 4.7: Rotação: (a) no sentido anti-horário, (b) no sentido horário.

cheia na Figura 4.6)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & sh_{xy} & sh_{xz} \\ sh_{yx} & 1 & sh_{yz} \\ sh_{zx} & sh_{zy} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & sh_{xy} & sh_{xz} \\ sh_{yx} & 1 & sh_{yz} \\ sh_{zx} & sh_{zy} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}.$$

4.2.4 Rotação

Rotações são transformações em que os pontos giram de um ângulo θ em torno de um dado ponto P_0. Por convenção, atribuímos valores positivos a θ quando o sentido de giro for anti-horário (Figura 4.7.(a)) e negativos quando for horário (Figura 4.7.(b)).

Representando os pontos (x_i, y_i) em coordenadas polares, com relação ao pólo P_0 e o eixo de referência x,

$$x_i = r_i cos\phi_i$$
$$y_i = r_i sen\phi_i,$$

onde r_i é a distância entre P_0 e o ponto (x_i, y_i) e ϕ_i, o ângulo entre a direção do vetor-posição (x_i, y_i) e o eixo de referência, a rotação de um ângulo θ do ponto P_1 faz com que o ângulo entre o novo vetor-posição (x_2, y_2) e o eixo de referência fique igual a $\theta + \phi_1$. Assim

$$\begin{aligned} x_2 &= r_1 cos(\theta + \phi_1) \\ &= r_1 cos\phi_1 cos\theta - r_1 sen\phi_1 sen\theta \\ &= x_1 cos\theta - y_1 sen\theta \\ y_2 &= r_1 sen(\theta + \phi_1) \\ &= r_1 sen\phi_1 cos\theta + r_1 cos\phi_1 sen\theta \\ &= y_1 cos\theta + x_1 sen\theta. \end{aligned}$$

Em forma matricial, temos a seguinte relação entre as coordenadas dos pontos antes e depois da rotação:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} cos\theta & -sen\theta \\ sen\theta & cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}.$$

Esta transformação equivale a girar um ponto em torno de um eixo z em $\Re^3$:

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}. \tag{4.9}$$

Analogamente, pode-se derivar a matriz de rotação em torno do eixo x (segundo a regra da mão-direita em relação ao eixo de rotação):

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & sen\theta \\ 0 & -sen\theta & cos\theta \end{bmatrix} \tag{4.10}$$

e em torno do eixo y,

$$R_y(\theta) = \begin{bmatrix} cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & cos\theta \end{bmatrix}. \tag{4.11}$$

Em alguns textos, os ângulos de rotação em torno dos eixos x, y e z são chamados, respectivamente, **ângulo de guinada** (*yaw*), **ângulo de declividade** (*pitch*) e **ângulo de rotação longitudinal** (*roll*).

Ao invés de manter o referencial e rodar os pontos em torno da origem, podemos rodar o referencial. Numericamente, os resultados são iguais. Por

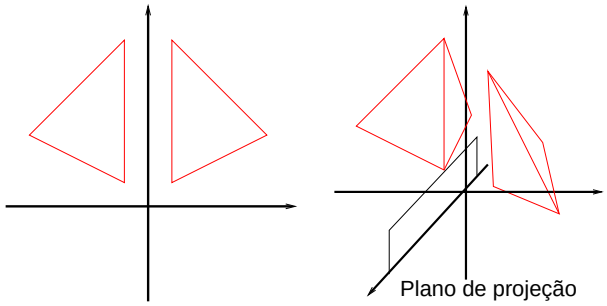


Figura 4.8: Reflexão.

exemplo, para rodar um ponto em torno do eixo z, como Figura 4.2.(b), temos

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \left[\begin{bmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right] \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{bmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \right].$$

4.2.5 Reflexão

Reflexão é uma rotação bem particular, em que um ponto "sai" do espaço em que ele está contido, dá um giro de 180^o no espaço de uma dimensão maior e volta em uma "posição espelhada" da original.

Num espaço bi-dimensional define-se a reflexão em relação a uma reta (Figura 4.8.(a)), enquanto num espaço tri-dimensional fala-se em reflexão em relação a um plano (Figura 4.8.(b)).

Por inspeção, pode-se concluir que, em termos de coordenadas, reflexão de um ponto (x_1, y_1, z_1) em relação ao plano xy é equivalente à inversão do sinal da coordenada z do ponto

$$\begin{aligned} x_2 &= x_1 \\ y_2 &= y_2 \\ z_2 &= -z_1. \end{aligned} \tag{4.12}$$

Em notação matricial, temos

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}. \tag{4.13}$$

De forma análoga, podemos obter a matriz de reflexão de um ponto em relação ao plano yz:

$$M_{yz} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

e em relação ao plano xz:

$$M_{xz} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.15}$$

Considerando que os pontos sejam representados por coordenadas afins em relação à base canônica, podemos aplicar essas matrizes de reflexão sobre os vetores-base do referencial e manter as coordenadas afins para chegar os mesmos valores numéricos de coordenadas cartesianas obtidas pelas equanções anteriores. Por exemplo, para reflexão em relação ao plano xy, as novas coordenadas são iguais às da Eq.(4.12)

$$\begin{aligned} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} &= \left[\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right] \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \right]. \end{aligned}$$

4.3 Transformações Afins, Lineares e Rígidas

Uma **transformação afim** é uma transformação que preserva paralelismo. Dados dois pontos (x_1, y_1, z_1) e (x_2, y_2, z_2), ela é definida pelo seguinte sistema de equações

$$\begin{aligned} x_2 &= a_{0,0}x_1 + a_{0,1}y_1 + a_{0,2}z_1 + a_{0,3} \\ y_2 &= a_{1,0}x_1 + a_{1,1}y_1 + a_{1,2}z_1 + a_{1,3} \\ z_2 &= a_{2,0}x_1 + a_{2,1}y_1 + a_{2,2}z_1 + a_{2,3}. \end{aligned} \tag{4.16}$$

Comparando Eq.(4.16) com Eqs.(4.3), (4.6), (4.8), (4.9) e (4.13), conclui-se que todas as cinco transformações apresentadas na seção 4.2 são casos particulares de transformação afim; portanto, elas preservam paralelismo.

Vimos também que as coordenadas cartesianas coincidem com as coordenadas afins de uma base canônica. Isso nos permite "ver" o mesmo resultado através de dois modos distintos de aplicação de uma mesma transformação $\mathcal{T}$: transformar as coordenadas com o referencial fixo (p.ex., girar uma caneta com a mão), ou, mantendo as coordenadas fixas, transformar o seu referencial(p.ex., girar em torno da uma caneta fixa sobre uma mesa). Traduzindo para notação matricial, temos a seguinte igualdade:

$$\begin{aligned}
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \right] + \begin{bmatrix} a_{0,3} \\ a_{1,3} \\ a_{2,3} \end{bmatrix} \\
&= \left[\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right] \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \begin{bmatrix} a_{0,3} \\ a_{1,3} \\ a_{2,3} \end{bmatrix} .
\end{aligned} \tag{4.17}$$

Com exceção das translações, as outras quatro transformações básicas são também conhecidas como **transformações lineares**, pois satisfazem as duas propriedades:

Aditividade: $f(x+y) = f(x) + f(y)$;

Homogeneidade: $f(\alpha x) = \alpha f(x)$, onde α é um escalar.

Observamos ainda que rotações, reflexões/espelhamentos (rotação de 180^0) e translações são as transformações geométricas básicas que preservam as medidas e os ângulos de uma figura geométrica. Portanto, elas são também conhecidas como **transformações rígidas**.

Embora as transformação afins sejam mais gerais, especializar as transformações em 5 classes básicas mostradas na seção 4.2 continua sendo a forma preferida na maioria de sistemas de informação gráfica, porque tais transformações tem uma interpretação física de fácil compreensão.

4.4 Concatenação de Matrizes

Na seção 4.2 vimos que as transformações básicas sobre os pontos podem ser representadas como operações com matrizes, isto é,

$$P_2 \;=\; P_1 + T \quad ,$$

$$
\begin{aligned}
P_2 &= SP_1 \ , \\
P_2 &= ShP_1 \ , \\
P_2 &= RP_1 \ , \\
P_2 &= MP_1 \ ,
\end{aligned}
$$

onde T, S, Sh, R e M denotam, respectivamente, matriz de translação, matriz de redimensionamento, matriz de cisalhamento, matriz de rotação e matriz de reflexão. Observe que, exceto a translação, todas transformações podem ser representadas como produto de matrizes. Dessa maneira, excluindo a translação, podemos representar uma sequência $\mathcal{M}$ de transformações básicas como um produto de matrizes, isto é,

$$M_n(M_{n-1}(\ldots(M_2(M_1(M_0P)))\ldots)) = (M_nM_{n-1}\ldots M_2M_1M_0)P = \mathcal{M}P.$$

O fato da translação não poder ser expressa como um produto de matrizes dificulta a elaboração de um procedimento comum para as transformações básicas. Este problema pode ser contornado com a inclusão de uma quarta coordenada w na representação de pontos e a extensão da matriz de transformação por mais uma coluna e uma linha. Com isso, a translação de um ponto passa a ser processada como um produto de matrizes tanto num espaço bidimensional

$$\begin{bmatrix} x+d_x \\ y+d_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

quanto em um espaço tridimensional

$$\begin{bmatrix} x+d_x \\ y+d_y \\ z+d_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Em notação matricial,

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}.$$

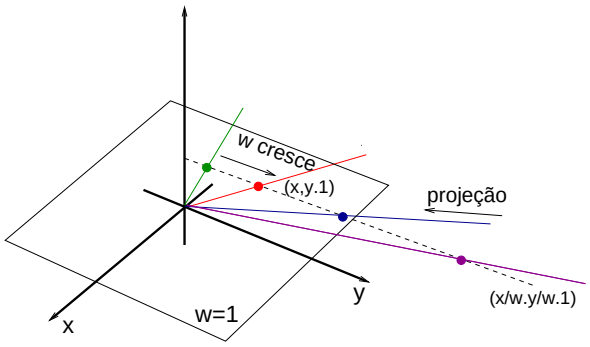


Figura 4.9: Coordenadas homogêneas.

Podemos ver a matriz $M_{4,4}$ como uma **concatenação** de 4 matrizes

$$\begin{bmatrix} U_{m,m} & \vdots & U_{m,1} \\ \cdots & \cdots & \cdots \\ U_{1,m} & \vdots & U_{1,1} \end{bmatrix}, \tag{4.18}$$

A matriz $U_{3,3}$ representa as transformações lineares e a matriz $U_{3,1}$ corresponde ao vetor de deslocamento. Veremos na seção 4.5 que a matriz $U_{1,3}$ está relacionada com projeções perspectivas, enquanto o escalar $U_{1,1}$ atua como um fator de escala uniforme

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & s \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ s \end{bmatrix} = \begin{bmatrix} \frac{1}{s}x_1 \\ \frac{1}{s}y_1 \\ \frac{1}{s}z_1 \\ 1 \end{bmatrix}.$$

4.5 Coordenadas Homogêneas

Depois das coordenadas cartesianas, polares, afins e baricêntricas, introduzimos na seção 4.4 uma quinta forma para representar um ponto no espaço $\Re^n$: por uma $n+1$-tupla de valores numéricos. Nesta representação os pontos em $\Re^n$ são projeções dos pontos do espaço $\Re^{n+1}$ sobre um hiperplano, usualmente $w = 1$, ao longo do raio de projeção que passa pela origem, ou seja, $(x, y, z, w) \sim (\frac{x}{w}, \frac{y}{w}, \frac{z}{w}, \frac{w}{w}) \sim (\frac{x}{w}, \frac{y}{w}, \frac{z}{w})$, como mostra Figura 4.9. As coordenadas de todos os pontos em $\Re^{n+1}$ que compartilham a mesma projeção em $\Re^n$ são conhecidas como **coordenadas homogêneas**.

Esta interpretação nos permite uniformizar a representação dos pontos localizados no "finito" e no "infinito" do espaço $\Re^3$. Mantendo as coordenadas x, y e z constantes, se diminuirmos o valor de w, as coordenadas $\frac{x}{w}$, $\frac{y}{w}$ e $\frac{z}{w}$ aumentam, como exemplifica a seguinte sequência

w	x	y	z	$\frac{x}{w}$	$\frac{y}{w}$	$\frac{z}{w}$
100	1	2	3	0.01	0.02	0.03
10	1	2	3	0.1	0.2	0.3
1	1	2	3	1	2	3
1/10	1	2	3	10	20	30
1/100	1	2	3	100	200	300
1/1000	1	2	3	1000	2000	3000

No limite de $w \to 0$, a 4-tupla $(x, y, z, 0)$ representa um ponto no infinito na direção (x, y, z). Os outros pontos com coordenadas $(x, y, z, 1)$ correspondem aos pontos ordinários do espaço $\Re^3$. Figura 4.9 ilustra esta idéia para um espaço bi-dimensional.

A representação de pontos por coordenadas homogêneas permite manipular as figuras geométricas e suas projeções de maneira uniforme. Vamos ilustrar esta idéia através de um exemplo numérico num sistema bi-dimensional. Seja um quadrado de vértices $P_0 = (1, 1, 1)$, $P_1 = (2, 1, 1)$, $P_2 = (2, 2, 1)$, $P_3 = (1, 2, 1)$ e uma matriz de transformação com o bloco $U_{1,2} \neq 0$ (Figura 4.10.(a))

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

O produto desta matriz com os vértices é (Figura 4.10.(b))

$$\begin{bmatrix} P_0' & P_1' & P_2' & P_3' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 \\ 2 & 3 & 3 & 2 \end{bmatrix}.$$

O efeito da transformação é um cisalhamento dos pontos na direção w, conforme vimos na seção 4.2.3. Em seguida, ao dividirmos as coordenadas x, y por w, o quadrado será projetado, de forma perspectiva, sobre o plano $w = 1$ (Figura 4.10.(c))

$$\begin{bmatrix} P_0' & P_1' & P_2' & P_3' \end{bmatrix} = \begin{bmatrix} 0.5 & 0.67 & 0.67 & 0.5 \\ 0.5 & 0.33 & 0.67 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Quando se trata de pontos "no infinito" ($w = 0$), o efeito de cisalhamento na direção w é o deslocamento destes pontos para finito ($w \neq 0$), distorcendo

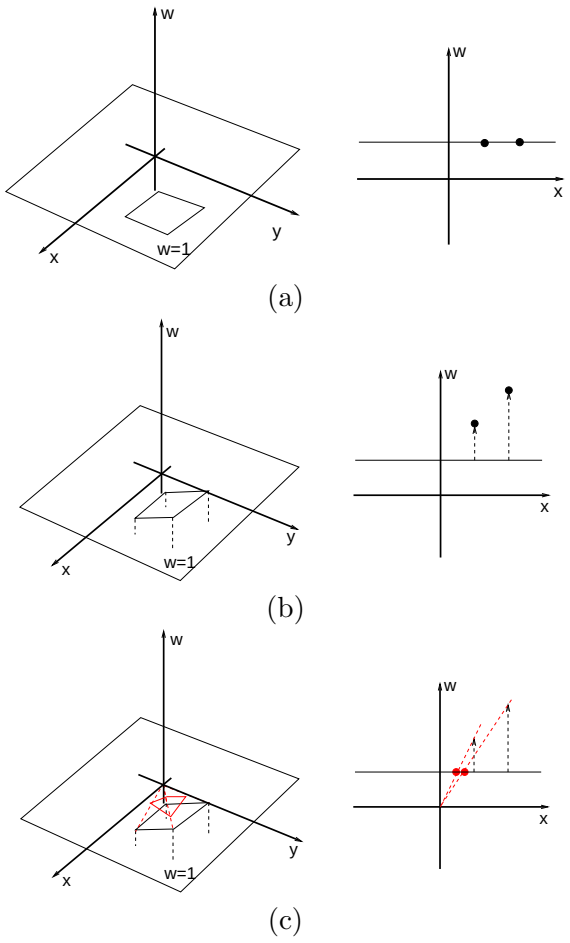


Figura 4.10: Transformação projetiva.

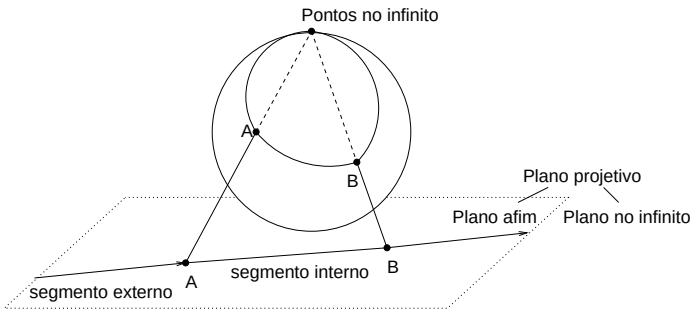


Figura 4.11: Segmento interno e externo.

perspectivamente a figura geométrica. Estes pontos, onde os raios paralelos convergem após este "deslocamento", são conhecidos como **pontos de fuga** (*vanishing points*).

Até agora só analisamos casos em que $w \geq 0$. O que significa $w < 0$? Vamos revisitar as coordenadas dos pontos com uma sequência crescente de w, mas com sinal trocado

w	x	y	z	$\frac{x}{w}$	$\frac{y}{w}$	$\frac{z}{w}$
-1/1000	1	2	3	-1000	-2000	-3000
-1/100	1	2	3	-100	-200	-300
-1/10	1	2	3	-10	-20	-30
-1	1	2	3	-1	-2	-3
-10	1	2	3	-0.1	-0.2	-0.3
-100	1	2	3	-0.01	-0.02	-0.03

Observe que agora se diminuirmos o valor de w, as coordenadas $\frac{x}{w}$, $\frac{y}{w}$ e $\frac{z}{w}$ aumentam, tendendo para 0. No limite de $w \to -\infty$, temos também um ponto no infinito na direção (x, y, z). Só que agora a aproximação é pelo semi-espaço $w < 0$, emendando dois sub-segmentos localizados em sub-espaços distintos. Observe que nesta representação por coordenadas homogêneas, dois pontos definem dois segmentos, ao invés de um só: um **segmento interno** (da visão clássica), constituído de "pontos finitos", e um **segmento externo**, que contém um "ponto infinito"(Figura 4.11).

4.6 Transformação de Vetores

Até agora só falamos sobre transformações de pontos. E os vetores? Será que seguem o mesmo molde de transformação? Para responder esta pergunta, basta lembrarmos que um vetor é, por definição, a subtração de dois pontos

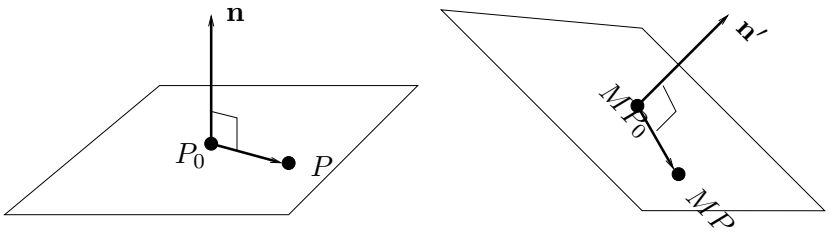


Figura 4.12: Vetor normal.

P_1 e P_2 representados em coordenadas cartesianas

$$\overrightarrow{P_1P_2} = P_2 - P_1.$$

Observe que, em coordenadas homogêneas, um vetor terá a coordenada $w = 0$, já que $w_2 = w_1 = 1$.

Se aplicarmos uma transformação M em ambos os pontos P_2 e P_1, o vetor sofrerá o mesmo efeito

$$MP_2 - MP_1 = M(P_2 - P_1) = M\overrightarrow{P_1P_2}.$$

Pode-se, portanto, aplicar diretamente a transformação sobre as coordenadas do vetor, no lugar de aplicar sobre os pontos e subtrair os resultados.

Um dos vetores mais utilizados em sistemas de informação gráfica é o vetor normal $\mathbf{n}$ de superfície. Se aplicarmos uma transformação M sobre os pontos da superfície, como P_0 e P da Figura 4.12, será que podemos aplicar M diretamente sobre as coordenadas do vetor normal para obter os novos vetores normais da superfície?

Vamos traduzir a nossa pergunta em relações algébricas entre as coordenadas

$$\mathbf{n} \cdot \overrightarrow{P_0P} = N^t\mathcal{P} = 0,$$

onde N e $\mathcal{P}$ são representações matriciais de $\mathbf{n}$ e $\overrightarrow{P_0P}$, respectivamente. Vimos que se aplicarmos a transformação M sobre os pontos do plano, o vetor $\overrightarrow{P_0P}$ será transformado em $M\mathcal{P}$. Sabemos ainda que o vetor normal N' da nova superfície deve satisfazer a relação

$$N'^t(M\mathcal{P}) = 0.$$

Agora, estamos prontos para responder a pergunta: qual é a transformação M' tal que $N' = M'N$ e

$$(M'N)^t(M\mathcal{P}) = N^tM'^t(M\mathcal{P}) = N^t(M'^tM)\mathcal{P} = 0?$$

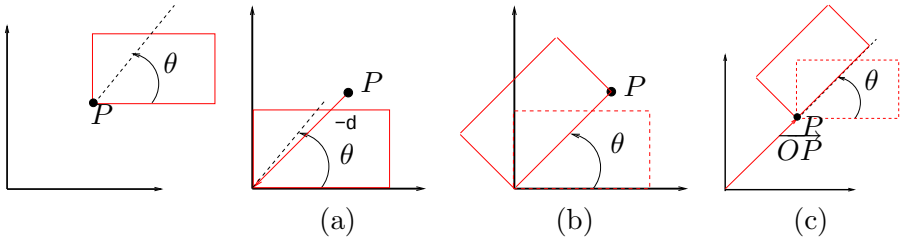


Figura 4.13: Rotação.

Comparando as equações do plano antes e após a transformação, chega-se na solução $M'^t M = I$, ou seja,

$$(M'^t)^t = M' = (M^{-1})^t.$$

4.7 Composição de Transformações

Uma das perguntas mais frequentes durante a criação de uma cena sintética é como representar em valores numéricos uma transformação que modifique a forma ou a posição de uma figura geométrica. Como seria, por exemplo, a matriz de transformação para ampliar ou rodar uma figura geométrica em torno de um ponto arbitrariamente escolhido? Esta é uma pergunta difícil de responder, já que a solução não é única. Uma alternativa é tentar "visualizar mentalmente" a figura geométrica no espaço 3D e imaginar a sequência de ações que poderia ser feita para obter o resultado desejado. É importante lembrar que o ponto invariante da figura deve estar na origem quando se deseja rodá-la ou redimensioná-la.

Figura 4.13 ilustra a rotação de um retângulo em torno de um ponto P por um ângulo θ. Em primeiro lugar, precisamos identificar o ponto invariante que deve ser deslocado até a origem antes da aplicação da Eq.(4.9). Neste caso, o ponto invariante é P, o inverso de cujo vetor-posição coincide com o vetor de deslocamento da figura geométrica para origem (Figura 4.13.(b)). Em seguida, aplica-se a rotação (Figura 4.13.(c)). Para finalizar, devemos retornar o ponto P à sua posição inicial (Figura 4.13.(d)). Sintetizando, a transformação desejada pode ser obtida pela composição de 3 transformações básicas: translação, rotação e translação.

Se forem conhecidas as coordenadas da nova posição de pelo menos quatro pontos, não coplanares e não colineares três a três, $P_0 = (x_0, y_0, z_0, 1)$, $P_1 = (x_1, y_1, z_1, 1)$, $P_2 = (x_2, y_2, z_2, 1)$, $P_3 = (x_3, y_3, z_3, 1)$, uma segunda alternativa para obter a solução é utilizar o fato de que a transformação

procurada deve satisfazer a relação

$$\begin{bmatrix} x'_0 & x'_1 & x'_2 & x'_3 \\ y'_0 & y'_1 & y'_2 & y'_3 \\ z'_0 & z'_1 & z'_2 & z'_3 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ z_0 & z_1 & z_2 & z_3 \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

onde P'_i são as coordenadas da nova posição. Como P_0, P_1, P_2 e P_3 não são coplanares nem colineares, três a três, a matriz

$$\begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ z_0 & z_1 & z_2 & z_3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

é inversível. Daí, a matriz de transformação pode ser obtida pelo produto de duas matrizes

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x'_0 & x'_1 & x'_2 & x'_3 \\ y'_0 & y'_1 & y'_2 & y'_3 \\ z'_0 & z'_1 & z'_2 & z'_3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ z_0 & z_1 & z_2 & z_3 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1}.$$

4.8 Invariância sob Transformações

Até agora só discutimos transformações sobre pontos e vetores. Na prática, o que nos interessa é a transformação de uma figura geométrica. Vimos na seção 3.2 que há diversas formas para descrever uma figura geométrica. Como se transforma uma figura geométrica? Será que precisamos aplicar a transformação em todos os pontos da figura geométrica? Nesta seção vamos mostrar que, para representações baseadas em combinação convexa, isso não é necessário.

Dada uma malha poligonal constituída pelos vértices interligados pelos segmentos. Se descrevermos os pontos do segmento definido pelos pontos P_1 e P_2 como uma **combinação convexa** destes pontos (Figura 4.14)

$$P = (1-t)P_1 + tP_2 \quad , t \in [0,1] \tag{4.19}$$

e aplicarmos uma transformação linear L sobre os pontos P equivale a aplicarmos a transformação nas coordenadas cartesianas de P_1 e P_2 e combiná-las com a mesma função convexa, ou seja,

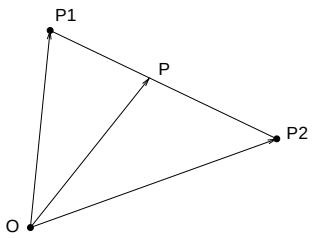


Figura 4.14: Combinação convexa de dois pontos.

$$\begin{aligned} LP &= L(1-t)P_1 + tP_2 \\ &= L(1-t)P_1 + LtP_2. \\ &= (1-t)LP_1 + tLP_2. \end{aligned}$$

Dizemos, então, que a representação de segmento é **invariante sob transformações lineares**.

Para translações, se adicionarmos o vetor de deslocamento $\mathbf{d}$ a cada vértice, os pontos do segmento resultante serão também deslocados deste montante, como mostra na seguinte derivação

$$\begin{aligned} [(1-t)(P_1+\mathbf{d}) + t(P_2+\mathbf{d})] &= [(1-t)(P_1+\mathbf{d}) + t(P_2+\mathbf{d})] \\ &= [(1-t)P_1 + tP_2] + ((1-t)\mathbf{d} + t\mathbf{d}) \\ &= (P+\mathbf{d}). \end{aligned}$$

Portanto, a representação dada pela Eq.(4.19) é também **invariante sob translações**.

Concluindo, se utilizarmos a Eq.(4.19) para representar os segmentos da malha poligonal, podemos reduzir a transformação em todos os pontos da malha pela transformação em seus vértices e aplicar a combinação convexa sobre eles. Isso reduz a complexidade do processo.

As curvas e superfícies de Bézier são também invariantes sob transformações afins, porque é uma representação baseada na combinação convexa dos pontos de controle.

Capítulo 5

Transformações Projetivas

O objetivo deste capítulo é mostrar como um sistema de informação gráfica consegue transformar as formas geométricas tri-dimensionais armazenadas na sua memória em uma imagem plana que estamos habituados a ver e desenhá-la na área de exibição (*viewport*) de uma janela sobre uma tela de monitor. Após a leitura deste capítulo, você deve ser capaz de

- distinguir e caracterizar, por meio de matrizes de transformaação, os tipos de projeção encontrados em desenhos técnicos, artísticos e em imagens como a visão humana percebe.
- definir e caracterizar os espaços utilizados ao longo do algoritmo de projeção: WC (sistema de universo – *World Coordinate System*), VRC (sistema da câmera – *View Reference System*), NDC (sistema normalizado – *Normalized Device Coordinate System*) e DC (sistema do dispositivo – *Device Coordinate System*).
- dizer o papel de cada espaço ao longo do processo de projeção.
- caracterizar estes espaços por diferentes modelos e saber a equivalência entre estes modelos.
- controlar os parâmetros dos modelos de câmera e de dispositivo para obter projeções desejadas.
- determinar as matrizes de transformação entre os espaços e as matrizes de transformação no espaço VRC para chegar a um volume de visão "padrão".

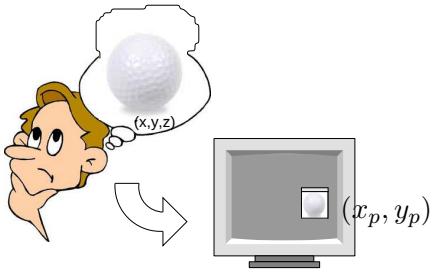
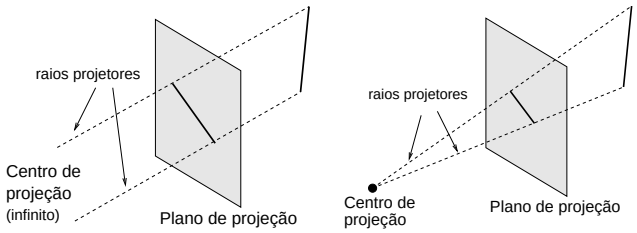


Figura 5.1: Transformação Projetiva.

Uma alternativa para obter imagens bi-dimensionais sintéticas é através do processo de imageamento das figuras geométrica concebidas no espaço tri-dimensional, mapeando os pontos (x, y, z) no espaço $\Re^3$ em *pixels* (x_p, y_p) do monitor, conforme ilustra Figura 5.1. Para isso, o sistema deve prover 3 estágios de transformação: ajuste da geometria e da posição das formas geométricas tri-dimensionais; projeção da cena construída em uma ou mais imagens planas; e mapear a imagem para a área de desenho de uma janela da tela do monitor a fim de exibí-la. No capítulo 4 foram apresentadas as transformações básicas e sua composição para realizar as tarefas do primeiro estágio. Neste capítulo vamos ver como podemos substituir a habilidade dos artistas e dos desenhistas de criar imagens de distintos tipos de projeção por manipulações algébricas e exibí-las na área de desenho de uma janela da tela de um computador.

Antes de apresentarmos o princípio dos algoritmos de transformação projetiva na seção 5.3, vamos mostrar na seção 5.1 uma taxonimia das projeções planas. Essencialmente, distinguem-se duas classes de projeções: **paralelas** e **perspectivas**. Embora as imagens obtidas com as projeções perspectivas se aproximem mais da percepção humana, as projeções paralelas são amplamente utilizadas em aplicações que requerem a preservação das proporções entre as medidas originais nas imagens, como em desenhos técnicos. Daremos, ainda, uma breve introdução ao sistema de visão humana na seção 5.2, cujo entendimento é fundamental para modelar o processo de geração de imagens próximas à nossa percepção. Só então, detalharemos um algoritmo de transformação projetiva na seção 5.4.



(a) (b)

Figura 5.2: Projeções: (a) paralelas e (b) perspectivas.

5.1 Taxonomia das Projeções

As projeções perspectivas planas foram muito utilizadas pelos artistas para produzir imagens próximas à percepção humana numa tela de pintura. No entanto, na área técnica predominam-se as projeções paralelas. As duas projeções diferem essencialmente na posição da câmera:

paralelas: a câmera fica no "infinito" $(x, y, z, 0)$ e os raios projetores que incidem sobre o plano de projeção são, consequentemente, paralelos (Figura 5.2.(a)).

perspectivas: a câmera fica num ponto do espaço $(x, y, z, 1)$ e os raios projetores que incidem sobre o plano de projeção convergem neste ponto (Figura 5.2.(b)).

Dentre cada uma destas duas classes de projeção, podemos ainda distinguir sub-classes a serem detalhadas nas seguintes subseções.

5.1.1 Projeções Paralelas

As projeções paralelas são caracterizadas por terem raios projetores paralelos, o que preserva o paralelismo dos segmentos projetados. Definimos como **fator de redução**, em inglês *foreshortening factor*, a razão entre o comprimento de um segmento projetado e o seu comprimento original. Denotamos por f_x, f_y e f_z, respectivamente, o fator de redução dos vetores-base $(1, 0, 0)$, $(0, 1, 0)$ e $(0, 0, 1)$ do referencial canônico sobre um plano de projeção, conforme ilustra a Figura 5.4. Observamos que a escolha do referencial no plano de projeção é arbitrária. Para simplificar os cálculos, assumimos que a sua origem coincida com a origem dos vetores-base para simplificar os cálculos e que haja uma transformação T que os projete sobre o plano

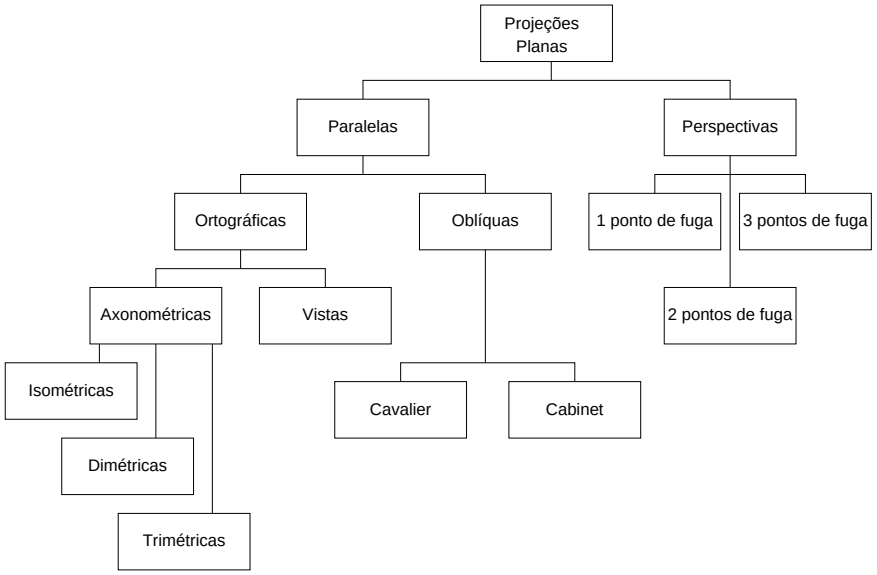


Figura 5.3: Taxonomia das projeções.

$$T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ 0 & 0 & 0 \end{bmatrix}, \tag{5.1}$$

Os fatores de redução dos vetores-base $(1,0,0)$, $(0,1,0)$ e $(0,0,1)$, são portanto, respectivamente,

$$f_x = \frac{\sqrt{x_x^2 + y_x^2}}{1} = \sqrt{x_x^2 + y_x^2} \tag{5.2}$$

$$f_y = \frac{\sqrt{x_y^2 + y_y^2}}{1} = \sqrt{x_y^2 + y_y^2} \tag{5.3}$$

$$f_z = \frac{\sqrt{x_z^2 + y_z^2}}{1} = \sqrt{x_z^2 + y_z^2}. \tag{5.4}$$

Das projeções paralelas, podemos distinguir dois casos, quanto à posição relativa dos raios projetores em relação ao plano de projeção. Quando os raios projetores são perpendiculares ao plano de projeção, dizemos que as projeções são **ortográficas**; caso contrário, elas são ditas **oblíquas**.

As projeções ortográficas podem ser classificadas, quanto aos fatores de redução dos vetores-base em

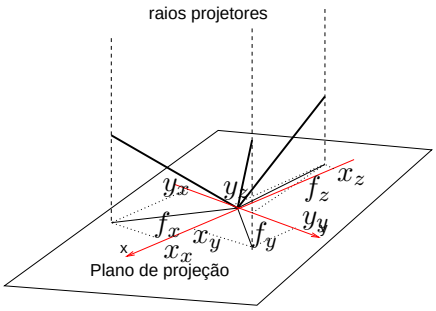


Figura 5.4: Fatores de redução.

vistas, quando os raios projetores são paralelos a um dos eixos do referencial; portanto, um dos fatores de redução é 0 e os outros dois fatores são iguais a 1. Na Figura 5.5 a imagem obtida com os raios projetores paralelos ao eixo z é considerada a vista **de topo** e tem f_x=f_y=1; a imagem obtida com raios paralelos ao eixo x é a vista **de frente** que tem f_y=f_z=1; e a obtida com raios projetores paralelos ao eixo y é a vista **de lado**, com f_x=f_z=1 (Figura 5.5). Estas projeções tem a vantagem de preservar as medidas das faces paralelas ao plano de projeção. Substituindo os valores dos fatores de redução na Eq. 5.1, obtém-se para cada vista uma matriz de transformação:

1. de topo:
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.5}$$

2. de frente:
$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

3. de lado:
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

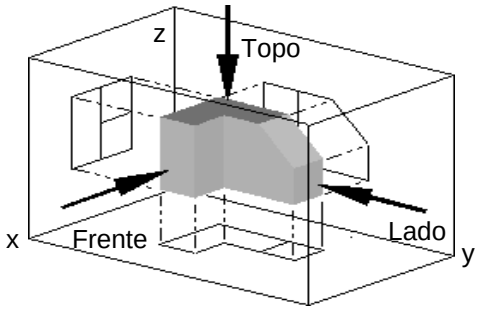
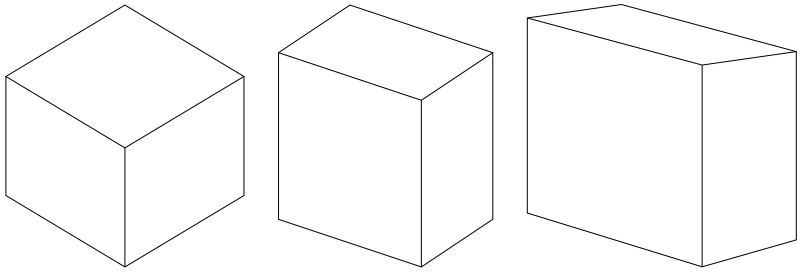


Figura 5.5: Vistas de uma figura geométrica.



(a) (b) (c)

Figura 5.6: Projeções axonométricas de um cubo: (a) isométrica, (b) dimétrica e (c) trimétrica.

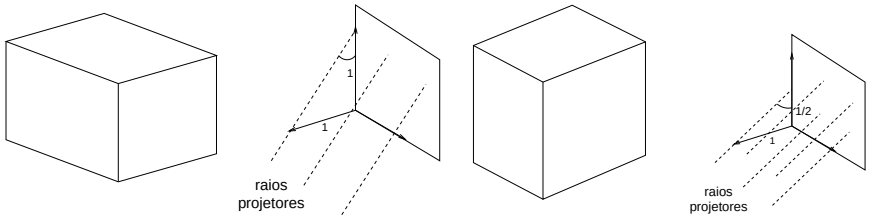
axonométricas, quando nenhum dos três fatores de redução assume valores nulos. Tipicamente, um dos eixos do referencial aparece "vertical" no desenho. Estas projeções proporcionam melhor percepção de profundidade, a custo da "perda das medidas originais". Dentre as projeções axonométricas, distinguem-se

isométricas: $f_x{=}f_y{=}f_z$ (Figura 5.6.(a)).

dimétricas: um par de fatores de redução é igual, ou seja, $f_x{=}f_y$, $f_x{=}f_z$ ou $f_y{=}f_z$ (Figura 5.6.(b)).

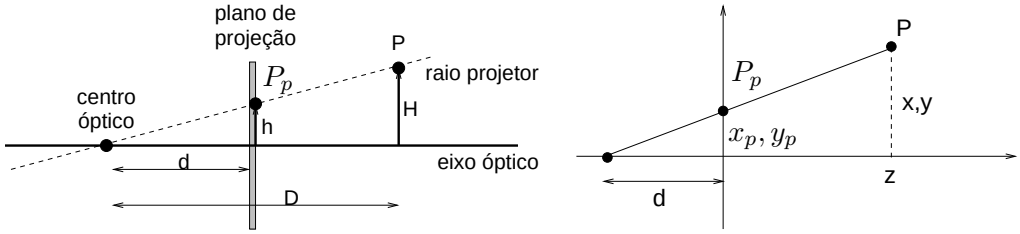
trimétricas: os fatores de redução são distintos (Figura 5.6.(c)).

Diferentemente das projeções axonométricas, as projeções oblíquas conseguem proporcionar percepção de profundidade preservando as medidas originais em alguns lados. As projeções oblíquas se distinguem pelo ângulo em que os raios projetores incidem sobre o plano de projeção. As projeções mais conhecidas são:



(a) (b)

Figura 5.7: Projeções oblíquas de um cubo: (a) cavalier e (b) cabinet.



(a) (b)

Figura 5.8: Projeção perspectiva: (a) visão descritiva e (b) visão analítica.

cavalier: o ângulo de incidência é 45^o; portanto, o fator de redução dos segmentos perpendiculares ao plano de projeção é igual a 1 (Figura 5.7.(a)).

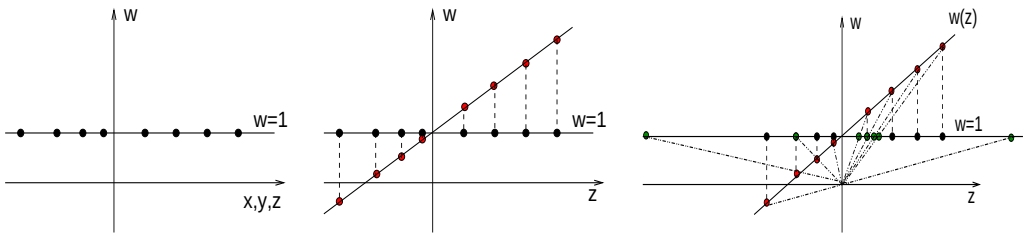
cabinet: o ângulo de incidência é 63.43^o. Com isso, o fator de redução dos segmentos perpendiculares ao plano de projeção é $\frac{1}{2}$ (Figura 5.7.(b)).

5.1.2 Projeções Perspectivas

As projeções perspectivas caracterizam-se por apresentar um **centro óptico**, no qual convergem todos os raios projetores. Somente o raio na direção denominada o **eixo óptico** não sofre distorção ao longo da sua trajetória, mantendo-se perpendicular ao plano de projeção (Figura 5.8). Com base nestes elementos, podemos descrever a geometria de uma projeção perspectiva utilizando o princípio de semelhança de triângulos, isto é,

$$\frac{d}{h} = \frac{D}{H}$$

Se **escolhermos apropriadamente um referencial**, tal como na Figura 5.8.(b), as coordenadas x e y, como x_p e y_p, corresponderão às distâncias dos pontos em relação ao eixo óptico (eixo z). Assim, um ponto $P = (x, y, z)$,



(a) (b) (c)

Figura 5.9: Coordenada w: (a) dos pontos originais; (b) após cisalhamento; (c) após homogeneização.

$z > 0$, e a sua projeção $P_p = (x_p, y_p, d)$ sobre o plano $z = 0$ na Figura 5.8.(b) guardam a relação

$$x_p = \frac{xd}{z+d} \quad y_p = \frac{yd}{z+d} \quad z_p = 0. \tag{5.6}$$

Em notação matricial,

$$\begin{bmatrix} x \\ y \\ z \\ \frac{z}{d} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{5.7}$$

Comparando com o que vimos na seção 4.5, esta transformação expressa um cisalhamento com a taxa de variação $\frac{1}{d}$ da coordenada w em relação à coordenada z (pontos em vermelho na Figura 5.9.(b)). E ao homogeneizarmos as coordenadas do espaço $\Re^4$ para o hiperplano $w = 1$, estaremos "projetando perspectivamente" os pontos sobre ele, obtendo figuras geométricas perspectivas em $\Re^3$ (pontos em verde na Figura 5.9.(c)).

As retas paralelas na direção $(0, 0, 1, 0)$ convergem no ponto $(0, 0, 0, \frac{1}{d}) = (0, 0, 0, 1)$ conforme ilustra Figura 5.10. Figura 5.11.(a) mostra o efeito da projeção sobre um cubo. Este tipo de projeção é conhecido como **projeção perspectiva com um ponto de fuga**.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{d} \end{bmatrix}$$

Além do cisalhamento da coordenada w em relação à coordenada z, podemos ainda cisalhá-la em relação à coordenada x. O efeito conjunto é

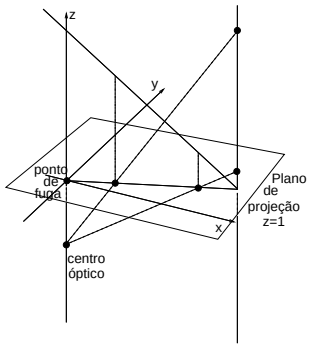


Figura 5.10: Ponto de fuga.

conhecido como **projeção perspectiva com dois pontos de fuga** (Figura 5.11.(b)). Utilizando notação matricial, isso equivale a ter dois elementos não nulos p e r na sub-matriz $U_{1,m}$ da Eq. 4.18. Se o aplicarmos sobre um feixe de vetores $(1, 0, 0, 0)$, este convergirá no ponto $(\frac{1}{p}, 0, 0, 1)$

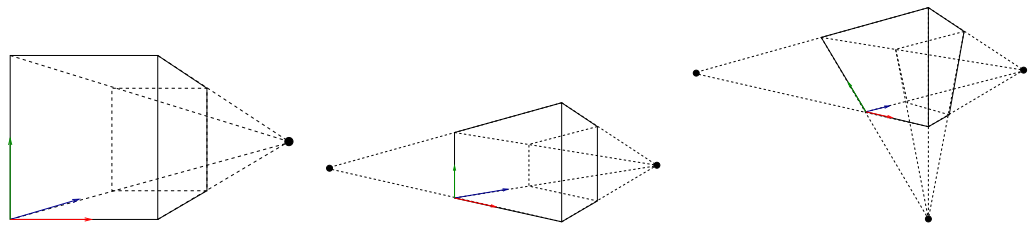
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ p & 0 & r & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ p \end{bmatrix}$$

E, finalmente, é possível adicionar cisalhamento em relação à coordenada y. Com isso, no "processo de homogeneização" retas paralelas aos três vetores-base convergirão em três pontos distintos no "finito" do espaço $\Re^3$, como mostra Figura 5.11.(c). Dizemos, então, que é uma **projeção perspectiva com três pontos de fuga**. Matematicamente, este conceito pode ser "modelado" com três elementos não nulos p, q e r na sub-matriz $U_{1,m}$ da Eq. 4.18. Se o aplicarmos sobre um feixe de vetores $(0, 1, 0, 0)$, este convergirá no ponto $(0, \frac{1}{q}, 0, 1)$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ p & q & r & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ q \end{bmatrix}$$

5.2 Noções do Sistema de Visão

Motivado pela criação de imagens mais próximas possíveis à forma como a visão humana percebe, procurou-se entender e modelar o sistema de visão



(a) (b) (c)

Figura 5.11: Projeção perspectiva de um cubo: (a) 1 ponto de fuga; (b) 2 pontos de fuga; (c) 3 pontos de fuga.

desde Renascimento. Os resultados foram fundamentais para o desenvolvimento e a evolução das pinturas em perspectiva que fornece uma "ilusão" de profundidade bem próxima da forma como percebemos um fenômeno tridimensional. Hoje aplicamos as técnicas desenvolvidas tanto na síntese de imagens foto-realistas como em visão computacional. Nesta seção mostraremos como o sistema de visão humana funciona sob o ponto de vista geométrico, concluindo com um modelo geométrico simplificado, porém suficiente, tanto para síntese quanto para análise de imagens.

A visão humana inicia o seu processamento no momento em que a luz entra na **pupila** do olho. A pupila é uma abertura no meio da íris e o seu diâmetro varia automaticamente com a intensidade da luz ambiente. O **cristalino** inverte e foca o sinal luminoso numa região do olho conhecida por **retina**. O cristalino divide o interior do olho em duas partes: a anterior, cheia de um líquido conhecido como o **humor aquoso**, e a posterior, cheia de uma substância gelatinosa denominada **humor vítreo**, como ilustra Figura 5.12. Esses humores possuem índices de refração sensivelmente iguais e muito próximos do da água.

O conjunto córnea-cristalino do olho humano comporta-se como uma lente convergente. O cristalino não é uma lente absolutamente rígida. Suas camadas periféricas são relativamente moles, de modo que sob ação de músculos ciliares que o envolvem, o cristalino se torna mais ou menos convergente. Se os objetos estão distantes, o cristalino fica mais fino, e para ver objetos próximos, ele se torna mais espesso. Esta faculdade do cristalino se adaptar chama-se **acomodação visual**. A acomodação, porém, não é ilimitada. A distância mínima, a partir da qual o olho não é capaz de focar nitidamente uma imagem sobre a retina, é conhecida por **distância mínima de visão distinta**. Nesse caso, a tensão dos músculos ciliares é máxima na acomodação. Para um olho normal, esta distância é aproxima-

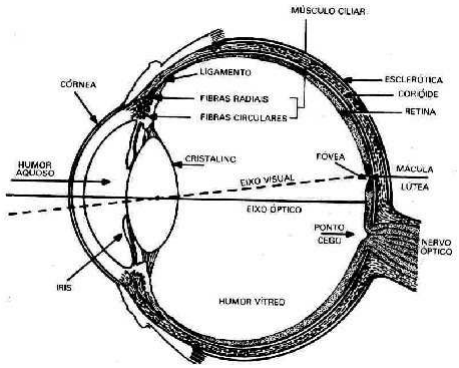


Figura 5.12: Olho (Fonte:http://www.geocities.com/ppoeys/globo_ocular2.jpg).

damente 25 cm. Ela aumenta com a idade. **Distância máxima de visão distinta** é a distância de um ponto (remoto) para a qual os músculos ciliares ficam relaxados.

É também limitada a capacidade do olho humano distinguir dois pontos. Esta capacidade é medida por **acuidade visual** que define o menor ângulo visual para o qual o olho consegue discriminar dois pontos, ou por **poder de resolução visual** que é a menor distância em que dois pontos fiquem distinguíveis para o olho. O valor médio da acuidade visual para um olho normal é em torno de 1 minuto de ângulo.

A retina contém dois tipos de células fotoreceptoras: **bastonetes** e **cones**. Os bastonetes, responsáveis pela visão acromática e estimuláveis a baixa intensidade luminosa, se encontram essencialmente na periferia da retina; enquanto os cones, capazes de distinguir cores num nível "normal" de luminosidade, são responsáveis pela visão cromática e se encontram na parte central da retina denominada **fóvea**. Usualmente, os bastonetes são 1000 vezes mais sensíveis à luz que os cones.

Os bastonetes diferem dos cones também na sensibilidade aos comprimentos de onda luminosa. Em média, os bastonetes são mais sensíveis às ondas de comprimento de 510 nm (azul esverdeado) enquanto os cones, às de comprimento 555nm (amarelo esverdeado). Isso explica por que distinguimos melhor objetos de cor azul do que os de cor vermelha num ambiente de baixa iluminação.

O **nervo óptico**, constituído pelos axônios das **células ganglionares**, transmite os sinais visuais para o córtex cerebral. Entre a retina e o nervo óptico, existem ainda células receptoras capazes de transformar sinais visuais em sinais neurais/elétricos. No córtex cerebral, esses sinais neurais

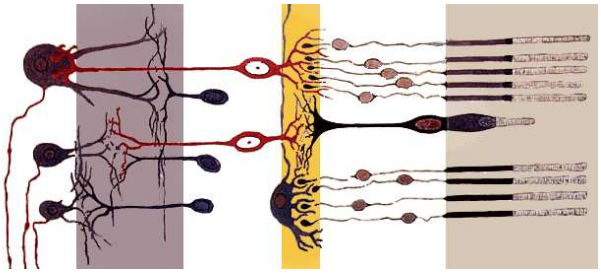
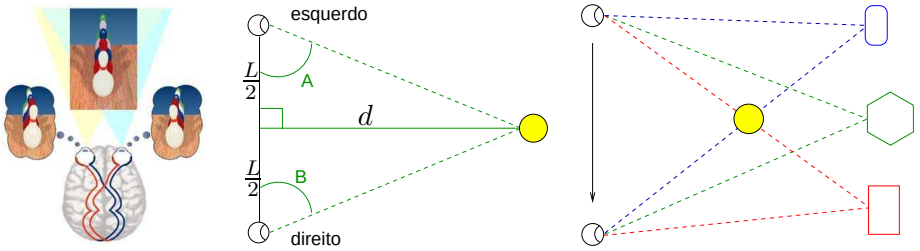


Figura 5.13: Retina (Fonte:http://pt.wikipedia.org/wiki/Ficheiro:Fig_retine.png).



(a) (b) (c)

Figura 5.14: Visão binocular: (a) campo de visão (Fonte:http://br.geocities.com/saladefisica5/leituras/binocular30.jpg), (b) paralaxe da visão binocular e (c) paralaxe do movimento.

são analisados e interpretados, produzindo diferentes "percepções visuais" (Figura 5.13). Na região da retina onde sai o nervo óptico não existem bastonetes nem cones, de forma que os sinais que chegam neste ponto não são enviados ao cérebro. Este ponto é conhecido como **ponto cego**.

Como já mencionado no capítulo 1, a nossa visão é binocular, constituída de dois olhos, proporcionando uma extensão angular, conhecido também como **campo de visão** ou *field of view* em inglês, de aproximadamente 150^o na vertical e 180^o na horizontal (Figura 5.14.(a)). Os impulsos elétricos, que cada olho recebe, são ligeiramente diferentes por causa da distância $L \sim 64mm$ entre os olhos. Isso resulta em **paralaxe**. Esses sinais convergem no **córtex occipital** do cérebro formando uma única imagem. O processo de fusão dos sinais de duas imagens para transformá-los numa única imagem é um dos responsáveis pela percepção de profundidade (Figura 5.14.(b)).

Além da visão binocular, o sistema visual possui outros recursos que nos ajudam na percepção de profundidade com um único olho, denominado por **visão monocular**:

paralaxe do movimento: consiste na percepção do movimento relativo de um objeto fixo em relação aos outros, também fixos, quando o observador desloca de um ponto ao outro (Figura 5.14.(c)).

sombreamento: ou variação da tonalidade das cores sob efeitos luminosos. Este recurso foi utilizado pelo Galileu para concluir que existem montanhas e crateras na Lua, muitos anos antes do primeiro homem pisar nela.

oclusão: através da oclusão dos objetos distantes pelos objetos mais próximos do observador aumenta a percepção de proximidade.

perspectiva: a percepção de linhas horizontais paralelas como linhas convergentes num ponto "infinito" reforça a noção de distância dos objetos.

acomodação automática da curvatura do cristalino: objetos de diferentes distâncias demandam distintos esforços musculares para focá-los.

atenuação atmosférica: objetos distantes parecem menos nítidos que os objetos próximos

No momento estamos interessados em formação de imagens na retina sob o ponto de vista geométrico. Para isso, utilizamos um esquema extremamente simplificado do olho denominado **olho reduzido**, no qual os meios transparentes são substituídos por uma única lente delgada convergente cujo centro óptico está 17mm da retina e todas as outras partes do olho não necessárias à construção geométrica da imagem são omitidas (Figura 5.15.(a)). Os sinais visuais captados podem ser representados como uma imagem real, invertida e menor que o objeto na retina (Figura 5.15.(b)). O fato de a imagem ser invertida não causa nenhum incômodo, pois os impulsos nervosos transmitidos pelo nervo óptico até o cérebro são interpretados de modo coerente com a nossa forma de percepção. Se rodarmos a imagem duas vezes, como na Figura 5.15.(c), pode-se perceber que, geometricamente, a "formação de imagens na retina" é equivalente a uma projeção perspectiva ilustrada na Figura 5.8. Portanto, poderemos aplicar Eq. 5.7 para obter uma imagem de geometria próxima à nossa percepção visual.

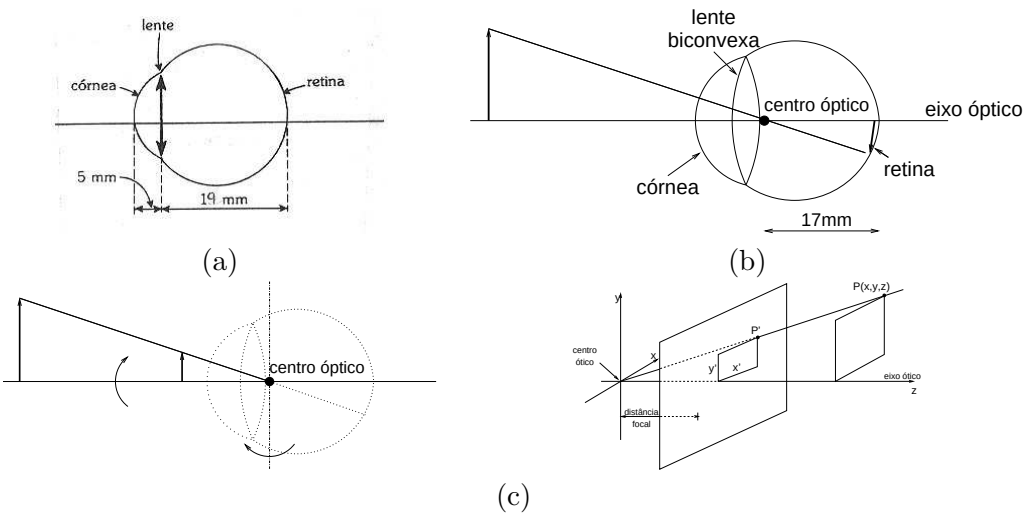


Figura 5.15: Formação de imagens: (a) olho reduzido (Fonte:http://www.geocities.com/ppoeys/olho_reduzido.jpg); (b) modelo simplificado; (c) projeção perspectiva; e (d) modelo matemático.

5.3 Espaços

Na seção 5.2 mostramos que o processo de formação de imagens na retina pode ser modelado como uma projeção perspectiva com um ponto de fuga e na seção 5.1 apresentamos uma variedade de projeções. Para cada tipo de projeção, demos ainda uma possível forma de manipulá-lo como produto de matrizes simples desde o plano de projeção, o centro óptico e a direção projetora satisfaçam as condições estabelecidas. Na prática, tais condições dificilmente são atendidas, como ilustra Figura 5.16. Naturalmente, pergunta-se se casos como estes também possam ser representados e manipulados como produto de matrizes. Adicionalmente, já que o objetivo é que o processo de projeção seja realizado pelo computador, é interessante que tenhamos uma interface simples para especificar os diversos tipos de projeção e um algoritmo que derive a partir dela uma imagem bi-dimensional dentro da expectativa.

A solução que apresentaremos no restante deste capítulo se baseia no fato de que a escolha de um referencial cartesiano (base canônica) está diretamente relacionado com o grau de simplicidade das matrizes de transformação. Ao invés de utilizarmos um único referencial, o processo é realizado com auxílio de cinco referenciais, a fim de que ao longo do processo os elementos da cena tenham sempre coordenadas com uma semântica próxima

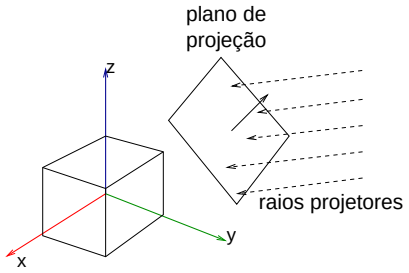


Figura 5.16: Uma projeção com arbitrário plano de projeção e arbitrária direção de projeção.

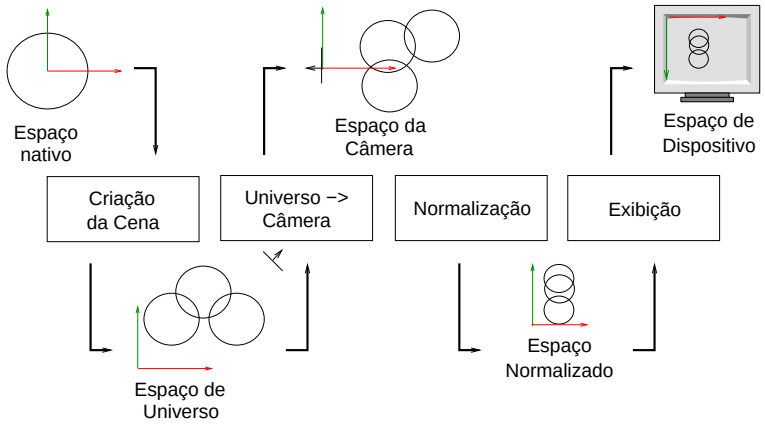


Figura 5.17: Fluxo de Projeção.

da nossa concepção (Figura 5.17):

Espaço do Objeto: é o espaço em que uma figura geométrica é definida. Procura-se, preferencialmente, utilizar um referencial para o qual a formulação algébrica que representa a figura em valores numéricos seja mais simples e expressiva possível, como vimos na seção 3.2.1. O sistema de coordenadas associado a este espaço é denominado de **sistema de coordenadas do objeto**, **de modelamento** ou **local**.

Espaço de Universo: é o espaço em que se encontram todos os elementos que compõem uma cena de interesse, como as formas geométricas, as fontes de luz e a câmera virtual. Uma finalidade deste espaço é posicionar relativamente os elementos da cena; portanto, deve-se transformar as coordenadas do espaço local de cada elemento para as deste espaço.

O referencial comum a todos os elementos da cena é denominado **sistema de coordenadas de universo WC** (ou **do mundo**, em inglês *world-coordinate system*).

Espaço da Câmera: é o espaço em que o centro óptico e o eixo óptico da câmera/do observador coincidam, respectivamente, com a origem e o eixo "z" do referencial. O uso deste espaço pode facilitar a interpretação das coordenadas dos pontos e simplificar a aritmética de projeção. O referencial utilizado para especificar este espaço é conhecido como **sistema de coordenadas de câmera** ou **do observador** (*viewing-reference coordinate system* – VRC). Usualmente é neste espaço em que são especificadas as dimensões do volume de visão de interesse, ou seja campo de visão em conjunto com a profundidadde de visão, porque os seus lados são paralelos aos eixos-base do referencial deste espaço (Figura 5.18).

Espaço Normalizado: é o espaço em que as coordenadas dos elementos de uma cena são normalizadas com respeito ao volume de visão de interesse. O sistema referencial deste espaço é conhecido como **sistema de coordenadas normalizado**, em inglês *normalized device coordinate system* – NDC. O uso deste espaço aumenta a reusabilidade, já que as coordenadas dos elementos da cena ficarão normalizadas num padrão pré-estabelecido, independentemente das dimensões do volume de visão. Adicionalmente, veremos no capítulo 6 que, por eficiência, os elementos da cena que estiverem fora do campo de visão devem ser removidos antes de enviá-los para o fluxo de imageamento e o algoritmo de remoção pode ser reduzido em problemas de intersecção entre os elementos e o volume de visão. Se a representação dos planos que delimitam o volume de visão for simples, a solução do problema será simples.

Espaço da Imagem: é o espaço onde uma projeção da cena de interesse é exibida. Em inglês, este espaço é conhecido por *viewport*. As coordenadas dos elementos da cena, em *pixels*, são dadas em relação a um sistema de referência denominado **sistema de coordenadas da imagem** ou **do dispositivo**, em inglês, *screen-coordinate system* – DC).

Nas seguintes subseções apresentamos os modelos mais conhecidos para estes espaços.

5.3.1 Modelos de Câmera

Da seção 5.1, vimos que um sistema de projeção é determinado, sob o ponto de vista geométrico, por cinco variáveis:

- centro óptico,
- eixo óptico,
- orientação da câmera,
- plano de projeção, e
- tipo de projeção (paralela ou perspectiva).

Dependendo da aplicação, estas variáveis são desdobradas em parâmetros de nomes distintos. Um modelo utilizado em Computação Gráfica possui, por exemplo, os seguintes parâmetros, dados no referencial de universo:

1. *view reference point* – VRP (posição de um ponto do plano de projeção),
2. *view plane normal* – VPN (vetor normal do plano de projeção),
3. *view up vector* – VUP (orientação da câmera),
4. *projection reference point* – PRP (posição da câmera),
5. tipo de projeção.

Os parâmetros VRP e VPN definem o plano de projeção, o parâmetro PRP corresponde ao centro óptico e o eixo óptico corresponde à reta que passa por PRP e o centro CW da face frontal do volume de visão de interesse (Figura 5.18.(a)). Mas como se especifica o centro CW? A forma mais simples para determiná-lo seria considerar que as coordenadas dos vértices do volume de visão sejam especificadas no referencial da câmera, ou seja, quando a direção do vetor normal ao plano de projeção coincide com o eixo "z". Neste caso particular, os valores dessas coordenadas estão diretamente relacionados com as dimensões do volume, conforme ilustra Figura 5.18.(b).

Uma outra alternativa para especificar as quatro variáveis de um sistema de projeção perspectiva é através dos seguintes parâmetros:

1. posição do observador – *eye*,
2. centro de interesse,
3. distância d,

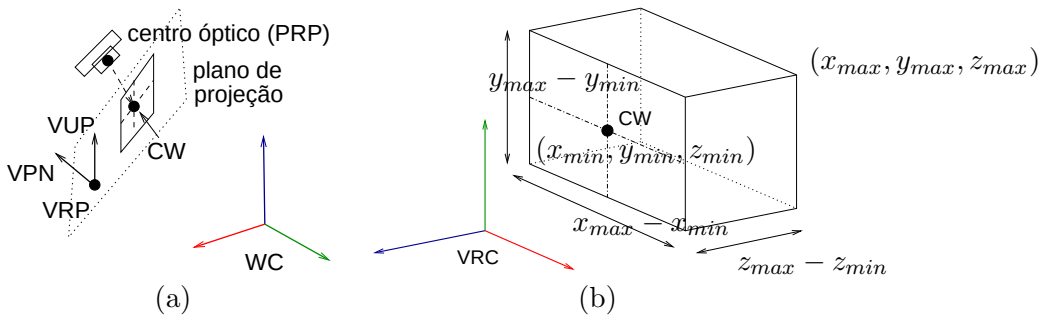


Figura 5.18: Modelo da câmera 1: (a) parâmetros e (b) volume de visão.

4. *view up vector* – VUP (orientação da câmera),

5. largura L, altura H e profundidae D da janela.

A posição do observador corresponde ao centro óptico, enquanto o eixo óptico passa por este ponto e o centro de interesse. Neste modelo, a direção do vetor normal do plano de projeção VPN é sempre coincidente com a direção do eixo óptico, o plano de projeção fica a distância d do observador, e a interseção do eixo óptico com o plano de projeção é o centro da janela CW. Com isso, pode-se derivar, a partir do comprimento L, da altura H e da profundidade D, os valores das coordenadas dos vértices do volume de visão, de acordo com a Figura 5.19.(a). No entanto, é comum encontrar, no lugar de L, H e D, os valores $(base, topo, direita, esquerda, frente, fundo)$, tal que $(topo-base)$, $(direita-esquerda)$ e $(fundo-frente)$ correspondam a H, L e D do volume de visão. Se utilizarmos o sistema de referência da câmera VRC, esses valores coincidirão com os valores das coordenadas dos vértices do volume de visão (Figura 5.19.(b)). Outro variante para especificação do volume de visão seria por 2 parâmetros: razão de aspecto L:H e o campo de visão *fovy* (Figura 5.19.(c)).

Em Visão Robótica, um sistema de projeção é comumente definido através de 5 parâmetros (Figura 5.20):

1. a distância focal da lente d,

2. a posição da câmera $\mathbf{w}_0$,

3. o ângulo θ (*pan*) do eixo óptico em relação ao vetor $(1, 0, 0, 0)$ do sistema de referência de universo (direção do eixo óptico),

4. o ângulo ϕ (*tilt*) em relação ao vetor $(0, 0, 1, 0)$ do sistema de referência de universo (orientação da câmera), e

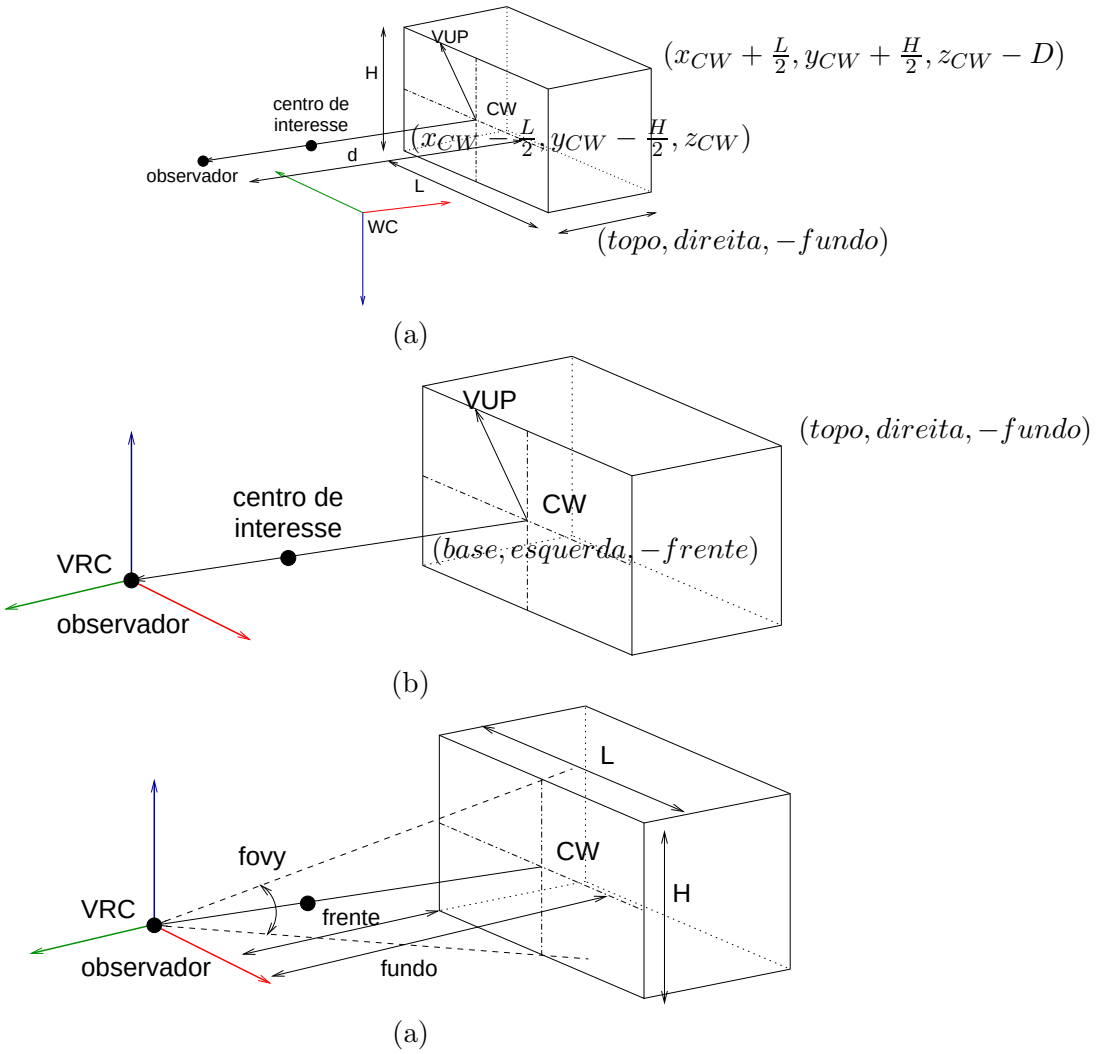


Figura 5.19: Modelo da câmera 2: (a) parâmetros, (b) volume de visão em VRC e (c) *fovy* no lugar da distância d ($d = \frac{H}{2tg\frac{fovy}{2}}$).

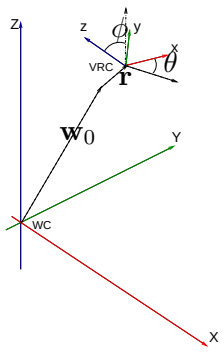


Figura 5.20: Modelo da Câmera 3.

5. o deslocamento $\mathbf{r}$ do plano de projeção em relação à base de sustentação da câmera (junto com $\mathbf{w}_0$ define a posição do centro óptico).

5.3.2 Modelos de Espaço Normalizado

A motivação primária do uso de um espaço normalizado é tornar o estágio de imageamento independente das dimensões do volume de visão, normalizando-as em intervalos de valores pré-estabelecidos. E quais são os intervalos de valores mais apropriados? Um dos critérios de escolha pode ser simplicidade no algoritmo de remoção dos elementos fora do volume de visão antes da transferência de uma cena para o fluxo de imageamento. No capítulo 6 veremos que um algoritmo de recorte consiste essencialmente em determinação de intersecção seguida de classificação e seleção dos resultados de interesse. Portanto, para o critério de simplicidade, podemos escolher os seguintes 6 planos para formar um **volume de visão canônico** de projeções paralelas (Figura 5.21.(a))

$$x = -1 \; ; \quad x = 1 \; ; \quad y = 1 \; ; \quad y = 1 \; ; \quad z = 0 \; ; \quad z = -1; \tag{5.8}$$

e para projeções perspectivas (Figura 5.21.(b)),

$$x = z \; ; \quad x = -z \; ; \quad y = z \; ; \quad y = -z \; ; \quad z = -z_{min} \; ; \quad z = -1 \tag{5.9}$$

com o eixo óptico coincidente com o eixo z do sistema de referência escolhida e perpendicular ao plano de projeção.

5.3.3 Modelos de Dispositivo

Vimos na seção 2.3 que, trabalhando em conjunto com um sistema de janelas, podemos desenhar as figuras geométricas numa área de desenho retangular,

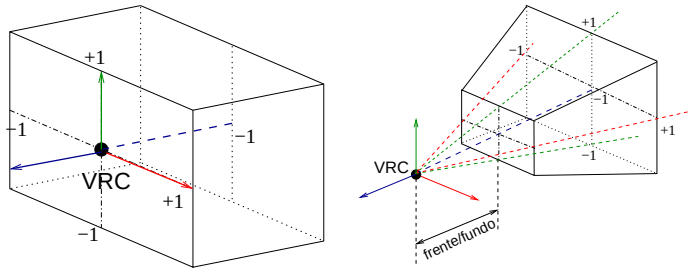


Figura 5.21: Volume de visão canônico: (a) paralelo e (b) perspectivo.

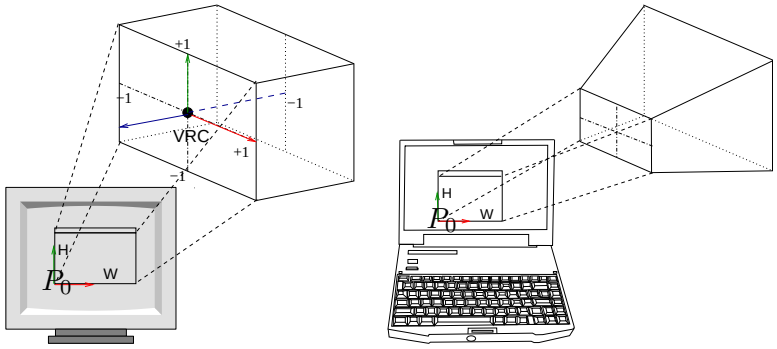
de dimensões $W \times H$ em *pixels* e posicionada em $P_0 = (U_o, V_o)$, especificamente alocada para elas. Como já comentamos, esta área é conhecida como *viewport*, em inglês. De que maneira o computador faria tal desenho? Basta "mapear" as imagens projetadas na área de desenho, como mostra Figura 5.22. Para isso, é necessário conhecer os seguintes parâmetros da área de exibição:

1. dimensões H e W do *viewport* e
2. posição (U_0, V_0) do *viewport*.

Um possível referencial, que representaria o *viewport* para o qual as suas dimensões aparecem explicitamente nos valores das coordenadas dos pontos, seria aquele apresentado na Figura 5.17: fixar a origem no seu canto esquerdo superior e considerar os eixos-base paralelos à janela. Isso, no entanto, implica em uma operação de espelhamento/reflexão em torno do eixo x, a mais, no processo de mapeamento de NDC para DC. Em decorrência disso, tem se preferido utilizar em sínteses de imagens o referencial que tem a origem em (U_0, V_0) e os dois vetores-base paralelos aos lados da janela (linhas em verde e em vermelho na Figura 5.22).

5.4 Matrizes de Transformação Projetiva

Uma vez apresentados os "espaços intermediários" que auxiliam uma transformação projetiva, vamos mostrar nesta seção, passo a passo, como se manipula algebricamente as formas geométricas entre estes espaços para chegar nas imagens desejadas. O objetivo é ilustrar como se consegue, de forma intuitiva, chegar a uma matriz de transformação complexa dividindo o problema em sub-problemas mais simples.



(a) (b)

Figura 5.22: Mapeamento em Viewport do volume canônico: (a) paralelo e (b) perspectivo.

Ao longo desta seção, para distinguir os referenciais dos quatro espaços, utilizaremos as seguintes convenções para designar suas respectivas coordenadas:

WC: (x, y, z);

VRC: (u, v, n), com u_{min}, u_{max}, v_{min}, v_{max}, $-F$ e $-B$, denotando *esquerda*, *direita*, *base*, *topo*, $frente$ e $fundo$, respectivamente (Figura 5.19.(b));

NDC: (u_n, v_n, n_n);

DC: (U, V).

Observamos aqui que utilizaremos como modelo de câmera o modelo ilustrado na Figura 5.18 e como o modelo de dispositivo o modelo mostrado na Figura 5.22.

5.4.1 WC para VRC

Estrategicamente, vamos considerar que o referencial do sistema de referência VRC seja definido na seguinte maneira: origem em VRP, eixo z, designado por $R3$, na direção do vetor VPN, o eixo y, representado por $R2$, na direção da projeção do vetor VUP sobre o plano de projeção, isto é,

$$R3 = \frac{VPN}{\|VPN\|};\ R1 = \frac{VUP \times R3}{\|VUP \times R3\|} \text{ e } R2 = R3 \times R1.$$

Para isso, mudar as coordenadas no referencial WC para o referencial VRC, precisaremos "colocar" a origem $(0, 0, 0, 1)$ do VRC no ponto VRP =

$(VRP_x, VRP_y, VRP_z, 1)$, cujas coordenadas são dadas em WC, através de uma translação

$$T(-VRP_x, -VRP_y, -VRP_z) \;=\; \begin{bmatrix} 1 & 0 & 0 & -VRP_x \\ 0 & 1 & 0 & -VRP_y \\ 0 & 0 & 1 & -VRP_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (5.10)$$

Em seguida, precisamos "alinhar" os vetores-base, $(1,0,0)$, $(0,1,0)$ e $(0,0,1)$, do VRC com os vetores $R1$, $R2$ e $R3$, respectivamente. Em outras palavras, precisaremos encontrar uma transformação $\mathcal{T}$, tal que,

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathcal{T} \cdot \begin{bmatrix} R1_x & R2_x & R3_x \\ R1_y & R2_y & R3_y \\ R1_z & R2_z & R3_z \end{bmatrix}.$$

Desse sistema é imediato concluir que a inversa da matriz formada pela concatenação dos três vetores R_1, R_1 e R_3 é a transformação procurada. Se olharmos com mais cuidado, perceberemos que a matriz concatenada é uma matriz ortogonal. Isso simplifica ainda mais a solução que consiste em simples transposição da matriz concatenada. "Estendendo" a matriz $\mathcal{T}$ para dimensão 4, temos

$$R \;=\; \begin{bmatrix} R1_x & R1_y & R1_z & 0 \\ R2_x & R2_y & R2_z & 0 \\ R3_x & R3_y & R3_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (5.11)$$

Aplicando essas transformações sobre os pontos P das formas geométricas obteremos as coordenadas dos pontos P' em VRC:

$$P' \;=\; R \cdot T(-VRP) \cdot P.$$

5.4.2 VRC

Agora, dependendo do tipo de projeção, aplicaremos distintas transformações sobre as formas geométricas em VRC de forma que o volume de visão fique orientado na forma como está mostrado na Figura 5.21, porém sem os valores normalizados.

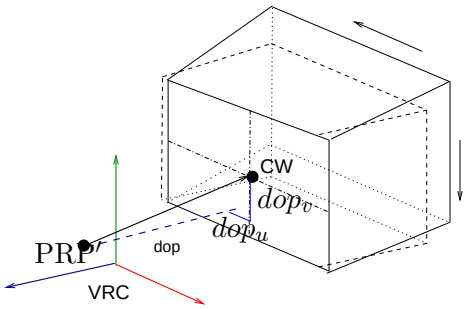


Figura 5.23: Cisalhamento do espaço VRC.

Paralela

Se o eixo óptico, cuja direção é definida pelos pontos PRP e CW, não for paralelo ao eixo n do referencial VRC, devemos fazer um cisalhamento das coordenadas u e v em relação à coordenada n, na seguinte proporção

$$\frac{\Delta u}{\Delta n} = \frac{dop_u}{-dop_n} \text{ e}$$
$$\frac{\Delta v}{\Delta n} = \frac{dop_v}{-dop_n},$$

onde $dop = CW - PRP'$ e

$$PRP' = R \cdot T(-VRP) \cdot PRP. \tag{5.12}$$

Em notação matricial, isso corresponde a aplicar em coordenadas $(u, v, n, 1)$ dos pontos a seguinte transformação.

$$SH = \begin{bmatrix} 1 & 0 & -\frac{dop_u}{dop_n} & 0 \\ 0 & 1 & -\frac{dop_v}{dop_n} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.13}$$

Após esta transformação garantimos que os lados do volume de visão fiquem paralelos ao eixo n. Agora só falta deslocar o centro da janela $(\frac{u_{max}+u_{min}}{2}, \frac{v_{max}+v_{min}}{2}, -F, 1)$ para que a frente do volume de visão fique centrado na origem do referencial VRC. Isso pode ser realizada pela matriz de transformação

$$T_{par} = \begin{bmatrix} 1 & 0 & 0 & -\frac{u_{max}+u_{min}}{2} \\ 0 & 1 & 0 & -\frac{v_{max}+v_{min}}{2} \\ 0 & 0 & 1 & F \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.14}$$

Perspectiva

Neste caso, PRP é o centro óptico e ele precisa estar na origem do referencial VRC (Figura 5.21.(b)). Para isso podemos aplicar em todas as figuras geométricas o deslocamento que leve o ponto PRP' (Eq. 5.12) para a origem, ou seja, $T(-PRP')$.

Após este deslocamento, o eixo óptico passará pela origem (Figura 5.24.(a)). Se o eixo não coincidir com o eixo n do referencial, poderemos aplicar Eq. 5.13 com as coordenadas do centro de janela devidamente transformadas, em toda cena para que o volume de visão fique orientado de tal forma que o eixo óptico, do sistema de projeção em consideração, esteja sobre o eixo n (Figura 5.24.(b)). Em seguida, vamos fazer uma mudança na escala do volume de visão de forma que o campo de visão *fovy* fique 90^o. Para isso, precisaremos aplicar os fatores de escala f_u e f_v, respectivamente, nos segmentos $(u_{max} - u_{min})$ e $(v_{max} - v_{min})$, de maneira que os seus comprimentos fiquem iguais à distância $2F$, isto é,

$$f_u \frac{u_{max}-u_{min}}{2} \;=\; F \text{ e } f_v \frac{v_{max}-v_{min}}{2} \;=\; F.$$

Esta mudança pelos fatores de escala pode ser representada pela matriz de transformação

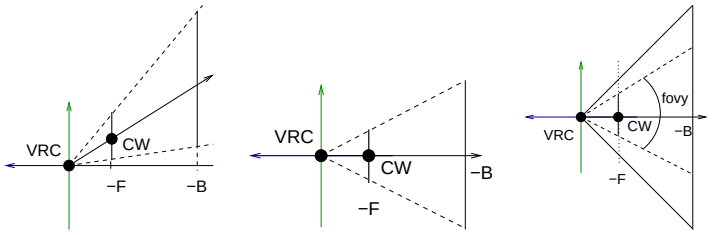
$$S_{fovy} \;=\; \begin{bmatrix} \frac{2F}{(u_{max}-u_{min})} & 0 & 0 & 0 \\ 0 & \frac{2F}{(v_{max}-v_{min})} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (5.15)$$

Conforme veremos na seção 5.4.3, este passo facilitará a normalização do volume de visão, pois ele transforma o volume de visão numa pirâmide de base quadrada e altura igual à metade dos lados da base.

5.4.3 VRC para NDC

Na seção 5.4.2 vimos como podemos "deformar" o volume de visão no espaço VRC para que ele fique simétrico em relação ao eixo n do referencial VRC. Nesta seção vamos ver a normalização das coordenadas com respeito às dimensões do volume de visão "deformado" $(u_{min}, u_{max}, v_{min}, v_{max}, -F, -B)$ para que ele se transforme em um dos volumes canônicos ilustrados na Figura 5.21.

No caso de um volume de visão paralelo, o fator de escala seria $\frac{2}{u_{max}-u_{min}}$, $\frac{2}{v_{max}-v_{min}}$ e $\frac{1}{B-F}$ para as direções u, v e n do referencial VRC, respectivamente, ou seja,



(a) (b) (c)

Figura 5.24: Transformação do volume de visão: (a) centro óptico na origem, (b) eixo óptico sobre eixo n e (b) $fovy = 90^o$.

$$S_{par} = \begin{bmatrix} \frac{2}{u_{max}-u_{min}} & 0 & 0 & 0 \\ 0 & \frac{2}{v_{max}-v_{min}} & 0 & 0 \\ 0 & 0 & \frac{1}{B-F} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.16}$$

Multiplicando as matrizes dadas pelas Eqs. 5.10, 5.11, 5.13, 5.14 e 5.16, obtemos uma matriz de normalização de um volume de visão para volume canônico paralelo representado pela Eq. 5.8

$$N_{par} \;=\; S_{par} \cdot T_{par} \cdot SH \cdot R \cdot T(-VRP_x, -VRP_y, -VRP_z) \tag{5.17}$$

No caso de um volume de visão perspectivo, chegamos em um volume de visão piramidal de base quadrada e altura igual à metade dos lados da base na seção 5.4.2. Podemos agora aplicar o mesmo fator de escala para todas as três direções neste volume para normalizá-lo. Este fator deve ser tal que "estique"/"comprima" o plano de fundo do volume de visão de $n = -B$ para $n = -1$. Em forma matricial, temos a seguinte transformação de escala

$$S_{per} = \begin{bmatrix} \frac{1}{B} & 0 & 0 & 0 \\ 0 & \frac{1}{B} & 0 & 0 \\ 0 & 0 & \frac{1}{B} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.18}$$

Já que conseguimos chegar em volumes canônicos, vamos tentar transformar o volume canônico perspectivo em volume canônico paralelo. Isso facilitará a tarefa de mapeamento do espaço NDC para DC. O algoritmo poderá tratar ambos os casos de forma uniforme, considerando como entrada somente o volume canônico paralelo. Como será esta transformação?

Vamos supor neste caso que não saibamos decompor este mapeamento em transformações básicas e utilizarems a segunda alternativa dada na seção 4.7, tentando determinar quatro correspondências entre os dois volumes e calcular a mtriz de transformação afim entre elas sob a restrição de que haja somente um ponto de fuga na direção n. Uma possível matriz de transformação seria resultado da concatenação de três sub-matrizes: mudança de escala ao longo do eixo n, deslocamento ao longo do eixo n e cisalhamento da coordenada w em relação à coordenada n. Em outras palavras, ela teria o seguinte aspecto:

$$P_{per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & 0 \end{bmatrix},$$

onde a, b e c são incógnitas. Vamos aplicar esta matriz sobre os 4 pares de correspondêndias mostrados na Figura 5.25 para determinar os valores destas incógnitas, isto é,

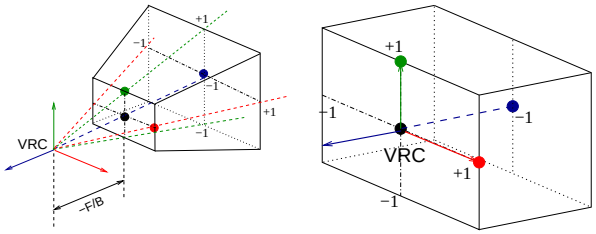
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & \frac{F}{B} \\ 0 & 0 & \frac{F}{B} & 0 \\ -\frac{F}{B} & -1 & -\frac{F}{B} & -\frac{F}{B} \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & \frac{F}{B} \\ 0 & 0 & \frac{F}{B} & 0 \\ -a\frac{F}{B}+b & -a+b & -a\frac{F}{B}+b & -a\frac{F}{B}+b \\ -c\frac{F}{B} & -c & -c\frac{F}{B} & -c\frac{F}{B} \end{bmatrix}$$

$$\sim \begin{bmatrix} 0 & 0 & 0 & -c \\ 0 & 0 & -c & 0 \\ \frac{-a\frac{F}{B}+b}{-c\frac{F}{B}} & \frac{-a+b}{-c} & \frac{-a\frac{F}{B}+b}{-c\frac{F}{B}} & \frac{-a\frac{F}{B}+b}{-c\frac{F}{B}} \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Igualando os elementos das duas últimas matrizes, chegamos à seguinte



(a) (b)

Figura 5.25: Correspondências entre volumes canônicos: (a) perspectivo e (b) paralelo.

matriz de transformação entre os volumes canônicos:

$$P_{per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{1-\frac{F}{B}} & \frac{\frac{F}{B}}{1-\frac{F}{B}} \\ 0 & 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{B}{B-F} & \frac{F}{B-F} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{5.19}$$

Resumindo, para transformar uma cena vista perspectivamente em espaço WC em um volume canônico paralelo podemos utilizar a sequência de transformações dadas pelas Eqs. 5.10, 5.11, 5.12, 5.13, 5.15, 5.18 e 5.19, ou seja,

$$N_{per} = P_{per} \cdot S_{per} \cdot S_{fovy} \cdot SH \cdot T(-PRP') \cdot R \cdot T(-VRP). \tag{5.20}$$

Vamos apresentar dois exemplos numéricos para mostrar os efeitos de cada passo da normalização de um cubo definido em WC:

$$\begin{bmatrix} -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 2 & 2 & 0 & 0 & 2 & 2 & 0 & 0 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Exemplo 5.1 *Dada a seguinte especificação:*

VRP (WC) = (-2,0,-2,1)
VUP (WC) = (0,1,0,0)
VPN (WC) = (1,0,1,0)
PRP (WC) = (2,0,2,1)
$(u_{min}, u_{max}, v_{min}, v_{max})$ *(VRC) = (-2,2,-4,2)*
B (VRC) = 10

F (VRC) = 0
tipo de projeção: PARALELA.

Ao invés de calcular a matriz N_{par} e aplicá-la sobre a figura geométrica, aplicaremos sucessivamente as matrizes $T(-PRP)$, R, SH, T_{par} e S_{par} para mostrar como a figura "se transforma" para chegar na forma canônica.

Passo 1 *$T(-VRP)$:*

$$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 2 & 2 & 0 & 0 & 2 & 2 & 0 & 0 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 \\ 2 & 2 & 0 & 0 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 & 3 & 3 & 3 & 3 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Passo 2 *R:*

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 \\ 2 & 2 & 0 & 0 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 & 3 & 3 & 3 & 3 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 1.41 & 0 & 1.41 & -1.41 & 0 & -1.41 & 0 \\ 2 & 2 & 0 & 0 & 2 & 2 & 0 & 0 \\ 1.41 & 2.83 & 1.41 & 2.83 & 2.83 & 4.24 & 2.83 & 4.24 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Passo 3 *SH: A direção dop pode ser obtida através de*

$$\begin{aligned} CW &= (\frac{u_{max}+u_{min}}{2}, \frac{v_{max}+v_{min}}{2}, -F, 1) \\ &= (\frac{2+(-2)}{2}, \frac{2+(-4)}{2}, 0, 1) = (0, -1, 0, 1). \\ dop &= CW - R \cdot T(-VRP) \cdot PRP \\ &= (0, -1, -4\sqrt{2}, 0). \end{aligned}$$

$$\begin{bmatrix} 1 & 0 & -\frac{0}{-4\sqrt{2}} & 0 \\ 0 & 1 & -\frac{-1}{-4\sqrt{2}} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1.41 & 0 & 1.41 & -1.41 & 0 & -1.41 & 0 \\ 2 & 2 & 0 & 0 & 2 & 2 & 0 & 0 \\ 1.41 & 2.83 & 1.41 & 2.83 & 2.83 & 4.24 & 2.83 & 4.24 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 1.41 & 0 & 1.41 & -1.41 & 0 & -1.41 & 0 \\ 1.75 & 1.5 & -0.25 & -0.5 & 1.5 & 1.25 & -0.5 & -0.75 \\ 1.41 & 2.83 & 1.41 & 2.83 & 2.83 & 4.24 & 2.83 & 4.24 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Passo 4 *T_{par}: O centro da sua janela frontal é* $(0, -1, 0, 1)$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$\begin{bmatrix} 0 & 1.41 & 0 & 1.41 & -1.41 & 0 & -1.41 & 0 \\ 1.75 & 1.5 & -0.25 & -0.5 & 1.5 & 1.25 & -0.5 & -0.75 \\ 1.41 & 2.83 & 1.41 & 2.83 & 2.83 & 4.24 & 2.83 & 4.24 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 1.41 & 0 & 1.41 & -1.41 & 0 & -1.41 & 0 \\ 2.75 & 2.5 & 0.75 & 0.5 & 2.5 & 2.25 & 0.5 & 0.25 \\ 1.41 & 2.83 & 1.41 & 2.83 & 2.83 & 4.24 & 2.83 & 4.24 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Passo 5 *S_{par}:*

$$\begin{bmatrix} \frac{2}{2-(-2)} & 0 & 0 & 0 \\ 0 & \frac{2}{2-(-4)} & 0 & 0 \\ 0 & 0 & \frac{1}{10-0} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$\begin{bmatrix} 0 & 1.41 & 0 & 1.41 & -1.41 & 0 & -1.41 & 0 \\ 2.75 & 2.5 & 0.75 & 0.5 & 2.5 & 2.25 & 0.5 & 0.25 \\ 1.41 & 2.83 & 1.41 & 2.83 & 2.83 & 4.24 & 2.83 & 4.24 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 0.705 & 0 & 0.705 & -0.705 & 0 & -0.705 & 0 \\ 0.917 & 0.833 & 0.250 & 0.167 & 0.833 & 0.750 & 0.167 & 0.083 \\ 0.141 & 0.283 & 0.141 & 0.283 & 0.283 & 0.424 & 0.283 & 0.424 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Exemplo 5.2 *Neste exemplo usaremos a mesma especificação do Exemplo 5.1, mudando apenas o tipo de projeção de paralela para perspectiva.*

Assim, as transformações no passo 1 e passo 2 são as mesmas. Mostramos somente os cálculos a partir do passo 3, aplicando as matrizes de transformação nos vértices do cubo em cada passo ao invés de computar N_{per}

Passo 3 *$T(-PRP')$: Como vimos no Exemplo 5.1*

$$PRP' \;=\; R \cdot T(-VRP)PRP \;=\; (0, 0, 4\sqrt{2}, 1).$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -4\sqrt{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

$$\begin{bmatrix} 0 & 1.41 & 0 & 1.41 & -1.41 & 0 & -1.41 & 0 \\ 2 & 2 & 0 & 0 & 2 & 2 & 0 & 0 \\ 1.41 & 2.83 & 1.41 & 2.83 & 2.83 & 4.24 & 2.83 & 4.24 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 1.410 & 0 & 1.410 & -1.410 & 0 & -1.410 & 0 \\ 2 & 2 & 0 & 0 & 2 & 2 & 0 & 0 \\ -4.247 & -2.827 & -4.247 & -2.827 & -2.827 & -1.417 & -2.827 & -1.417 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Passo 4 *SH: Diferentemente do Exemplo 5.1, precisamos atualizar as coordenadas do centro da janela*

$$\begin{aligned} dop &= T(PRP') \cdot CW - PRP' \\ &= (0, -1, -8\sqrt{2}, 0). \end{aligned}$$

$$\begin{bmatrix} 1 & 0 & -\frac{0}{-8\sqrt{2}} & 0 \\ 0 & 1 & -\frac{-1}{-8\sqrt{2}} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

$$\begin{bmatrix} 0 & 1.410 & 0 & 1.410 & -1.410 & 0 & -1.410 & 0 \\ 2 & 2 & 0 & 0 & 2 & 2 & 0 & 0 \\ -4.247 & -2.827 & -4.247 & -2.827 & -2.827 & -1.417 & -2.827 & -1.417 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 1.410 & 0 & 1.410 & -1.410 & 0 & -1.410 & 0 \\ 2.375 & 2.250 & 0.375 & 0.250 & 2.250 & 2.125 & 0.250 & 0.125 \\ -4.247 & -2.827 & -4.247 & -2.827 & -2.827 & -1.417 & -2.827 & -1.417 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Passo 5 $S_{per} \cdot S_{fovy}$*:*

$$\begin{bmatrix} \frac{2}{(2-(-2))(10)} & 0 & 0 & 0 \\ 0 & \frac{2}{(2-(-4))(10)} & 0 & 0 \\ 0 & 0 & \frac{1}{10} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$\begin{bmatrix} 0 & 1.410 & 0 & 1.410 & -1.410 & 0 & -1.410 & 0 \\ 2.375 & 2.250 & 0.375 & 0.250 & 2.250 & 2.125 & 0.250 & 0.125 \\ -4.247 & -2.827 & -4.247 & -2.827 & -2.827 & -1.417 & -2.827 & -1.417 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 0.071 & 0 & 0.071 & -0.071 & 0 & -0.071 & 0 \\ 0.079 & 0.075 & 0.012 & 0.008 & 0.075 & 0.071 & 0.008 & 0.004 \\ -0.425 & -0.283 & -0.425 & -0.283 & -0.283 & -0.142 & -0.283 & -0.142 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Note que até este passo o paralelismo das linhas é preservado!

Passo 6 P_{per}*: Como* $F = 0$*, temos*

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}.$$

$$\begin{bmatrix} 0 & 0.071 & 0 & 0.071 & -0.071 & 0 & -0.071 & 0 \\ 0.079 & 0.075 & 0.012 & 0.008 & 0.075 & 0.071 & 0.008 & 0.004 \\ -0.425 & -0.283 & -0.425 & -0.283 & -0.283 & -0.142 & -0.283 & -0.142 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 0.071 & 0 & 0.071 & -0.071 & 0 & -0.071 & 0 \\ 0.079 & 0.075 & 0.012 & 0.008 & 0.075 & 0.071 & 0.008 & 0.004 \\ -0.425 & -0.283 & -0.425 & -0.283 & -0.283 & -0.142 & -0.283 & -0.142 \\ 0.425 & 0.283 & 0.425 & 0.283 & 0.283 & 0.142 & 0.283 & 0.142 \end{bmatrix}.$$

Dividindo as coordenadas x*,* y *e* z *pela coordenada* w*, obteremos os pontos no espaço euclideano* $\Re^3$*.*

5.4.4 NDC para DC

Tendo sempre como resultado do processo de normalização um volume canônico paralelo, basta aplicarmos Eq. 5.5 para obtermos a imagem da cena e mapear esta imagem no *viewport* da tela alocado para esta finalidade, de acordo com o esquema da Figura 5.22.(a). Para isso, a seguinte sequência de transformações é suficiente: fixar a origem em $(-1,-1)$ para ampliar a imagem nas dimensãoes da janela, sem deslocar este ponto, e relocar a origem do referencial DC para o *pixel* (U_0, V_0). Em termos de matriz,

$$\mathcal{V} = \begin{bmatrix} 1 & 0 & 0 & U_0 \\ 0 & 1 & 0 & V_0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{W}{2} & 0 & 0 & 0 \\ 0 & \frac{H}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \qquad (5.21)$$

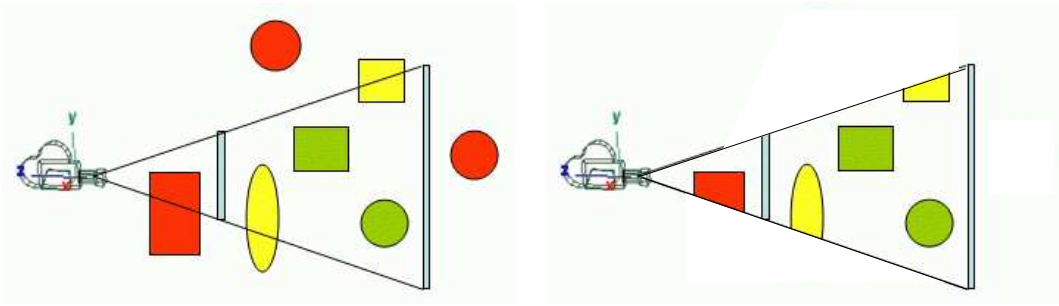
Capítulo 6

Recorte

O objetivo deste capítulo é apresentar uma técnica que melhore o desempenho do imageamento de uma cena tri-dimensional – o **recorte**. A técnica consiste essencialmente em remover antecipadamente as partes que não estejam dentro do volume de visão para reduzir a quantidade de operações que não contribuam efetivamente na qualidade da imagem exibida. Após a leitura deste capítulo, você deve ser capaz de

- dizer o papel dos algoritmos de recorte na síntese de imagens.
- citar as principais idéias em que se basearam os algoritmos de recorte e as diferenças entre eles.
- aplicar os algoritmos de recorte.
- decidir o momento apropriado para aplicar um algoritmo de recorte em um fluxo de projeção.
- explicar a influência da coordenada w em recorte.

De acordo com a Figura 5.22 somente as figuras geométricas contidas no volume de visão canônico aparecem na área de desenho de uma janela. O restante das figuras devem ser removidas para não "poluir" outras áreas da tela, como ilustra a Figura 6.1. O processo de remoção das partes que estejam fora da janela de exibição é denominado **recorte**, em inglês *clipping*. Esta tarefa, aparentemente simples (quem já não recortou uma figura?), é um dos problemas básicos em sistemas de informação gráfica. Como um computador, sem tesoura, estilete ou canivete, consegue recortar corretamente as imagens representadas por um monte de números?



(a) (b)

Figura 6.1: Cena: (a) sem recorte e (b) com recorte (Fonte: http://www.lighthouse3d.com/opengl/viewfrustum/.

Uma alternativa seria rasterizar as figuras geométricas, transformando os valores reais das suas coordenadas em *pixels* como veremos no capítulo 10, e classificar somente as amostras escolhidas. No entanto, com o aumento da complexidade das cenas este procedimento, mesmo realizado em *hardware*, passou a ser proibitivo para aplicativos interativos que requerem no mínimo 15 quadros por segundo (15 fps[1]). Muitos pontos seriam processados desnecessariamente ao longo do fluxo de projeção e de rasterização para serem descartados, após rasterização. Além disso, veremos na seção 6.4 que a divisão pela coordenada w em projeções perspectivas pode introduzir ambiguidade na representação. Surgiu-se, então, a proposta de remover as partes fora do volume de visão mais cedo possível ao longo do fluxo de projeção. Para diferenciar a primeira alternativa da segunda, é comum encontrar na literatura o termo *scissoring* referenciando recorte no **espaço de dispositivo** após rasterização e o termo **recorte** para remoção ocorrida em estágios anteriores a rasterização.

Neste capítulo, o nosso foco é o recorte ocorrido antes da rasterização. Remover os pontos que estão fora da janela/do volume de visão se reduz, numericamente, em um problema de intersecção entre os seus segmentos/planos limitantes e as figuras geométricas de uma cena e classificação do resultado. Descrevendo as figuras geométricas e os segmentos/planos limitantes como equações, esse problema é equivalente a computar a solução de um sistema de equações. No entanto, além do problema de representação algébrica não ser ainda um problema fechado, nem sempre as representações conhecidas são fáceis para serem processadas. Por exemplo, como você des-

[1] fps – *frames per second*.

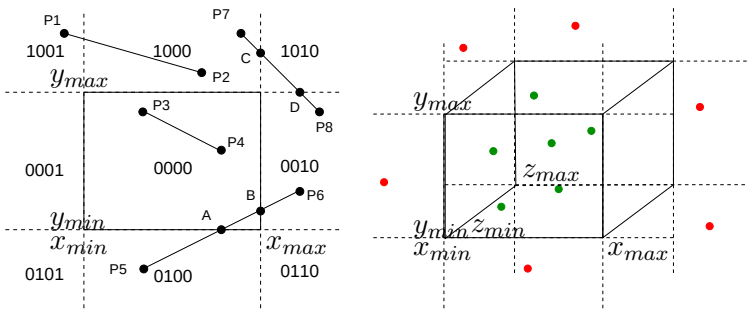


Figura 6.2: Recorte de pontos em relação a: (a) janela de exibição e (b) volume de visão.

creveria, com uso das funções que você conhece, um tetraedro? Um cubo? Um bule de chá? Para uniformizar o tratamento, os processadores gráficos limitaram o seu escopo de figuras geométricas para malhas poligonais cujas projeções são também malhas poligonais. Estas malhas são constituídas por três tipos de primitivas: pontos, segmentos e polígonos (seção 3.3).

Apresentaremos algoritmos de recorte destes três tipos de primitivas nas seções 6.1, seção 6.2 e seção 6.3, respectivamente. Depois mostraremos na seção 6.4 em quais estágios do fluxo de projeção poderemos introduzir estes algoritmos de recorte.

6.1 Recorte de Pontos

O recorte de um ponto (x, y) em relação a uma janela de exibição retangular $(x_{min}, y_{min}, x_{max}, y_{max})$ consiste essencialmente em verificar a sua pertinência à janela que corresponde ao seguinte sistema de inequações (Figura 6.2.(a)):

$$x_{min} \leq x \leq x_{max} \quad e \quad y_{min} \leq y \leq y_{max}$$

Se for um ponto (x, y, z) em relação a um volume de visão paralelo, acrescenta-se mais duas inequações ao sistema (Figura 6.2.(b)):

$$z_{min} \leq z \leq z_{max}$$

E para um volume de visão perspectivo? Podemos dividir a solução em dois passos: primeiro, transformar o volume de visão perspectivo para um volume paralelo (seção 5.4.3) e depois, fazer recorte em relação a este volume de visão paralelo.

6.2 Recorte de Segmentos

Em maioria dos casos, os segmentos ou estão totalmente dentro ou estão totalmente fora da região de interesse. Seria, portanto, interessante poder distinguir eficientemente estes casos antes de computar desnecessariamente as intersecções. Apresentaremos na seção 6.2.1 um algoritmo simples e eficiente para executar esta tarefa e veremos como podemos reduzir outros casos em "casos triviais". Em seguida, apresentaremos um algoritmo eficiente para tratar casos mais gerais na seção 6.2.2 e um variante na seção 6.2.3.

6.2.1 Algoritmo de Cohen-Sutherland

O algoritmo de Cohen-Sutherland se baseia em uma simples constatação: um segmento está totalmente contido em uma região, se e somente se, os seus pontos extremos estão contidos nela, e um segmento está totalmente fora dela, se os seus pontos extremos estiverem em um semi-espaço que não contenha a região. Cohen e Sutherland conceberam uma forma de dividir o espaço em sub-espaços aa partir da região de interesse e atribuíram um código para cada sub-espaço de tal forma que uma operação lógica *AND*, *bit* a *bit*, é suficiente para separar os segmentos que não interceptam com os lados da região. Os códigos são conhecidos como *outcodes*.

No caso de uma janela retangular, os seus quatro lados dividem o espaço em 9 sub-espaços, como mostra Figura 6.3.(a), e são utilizados 4 *bits* para representar cada sub-espaço seguindo a seguinte convenção, do bit menos significativo para o mais:

- primeiro *bit* (esquerdo): $x < x_{min}$.
- segundo *bit* (direito): $x > x_{max}$.
- terceiro *bit* (embaixo): $y < y_{min}$.
- quarto *bit* (acima): $y > y_{max}$.

A localização de cada ponto em relação aos 9 sub-espaços pode ser computada com uso das inequações dadas na seção 6.1, tendo como resultado um código de 4 *bits*. Se aplicarmos uma operação lógica *AND*, *bit* a *bit*, entre os códigos dos dois pontos extremos de um segmento

A	B	*AND*
0	0	0
0	1	0
1	0	0
1	1	1

,

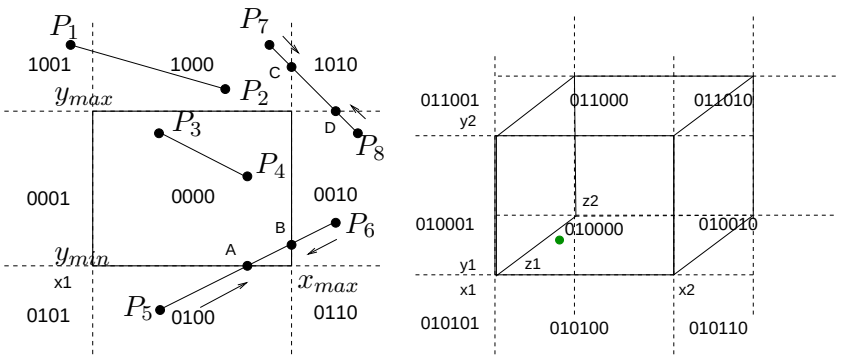


Figura 6.3: Algoritmo de Cohen-Sutherland aplicado em: (a) janela de exibição e (b) volume de visão.

teremos as seguintes possíveis situações:

1. os códigos dos pontos extremos e do resultado são 0000: o segmento está contida na região;

2. o resultado é diferente de zero: o segmento está totalmente fora da região; e

3. o resultado é 0000 embora os códigos dos pontos extremos não sejam: é indecidível a pertinência do segmento à região.

Na situação 3, o segmento P_1P_2 é subdividido (e descartado), da direção do ponto exterior P_1 para interior, até que todos fiquem decidíveis. Para subdividir, podemos substituir os valores x_{min}, x_{max}, y_{min} e y_{max} em uma das seguintes equações a fim de obter as intersecções sucessivamente

$$\begin{aligned} y &= mx + b \\ x &= \frac{1}{m}y - \frac{b}{m}, \end{aligned} \tag{6.1}$$

conforme ilustra o seguinte pseudo-código

Input: Segmento P_1P_2
Output: Resultado de recorte
$P_A = P_1$;
$P_B = P_2$;
foreach *lado da janela* x_{min}, x_{max}, y_{min}, y_{max} **do**
  if P_AP_B *é indecidível* **then**
    if P_A *não está fora da janela* **then**
      troca P_A com P_B;
    end
    Determine intersecção I entre P_AP_B com o lado da janela;
    Substitua P_A por I;
  else if P_A *e* P_B *tem código 0000* **then**
    Retorne P_AP_B como sub-segmento visível;
    break;
  else
    Retorne o estado de P_1P_2 como fora da janela;
    break;
  end
end

Algoritmo 1: Algoritmo de Cohen-Sutherland: Recorte de um segmento.

Aplicando o algoritmo nos segmentos da Figura 6.3.(a), temos

Segmento	Código de P_1	Código de P_2	Operação lógica	Comentário
P_1P_2	1001	1000	1000	totalmente fora
P_3P_4	0000	0000	0000	totalmente dentro
P_5P_6	0100	0010	0000	indecidível
P_7P_8	1000	0010	0000	indecidível

Como os resultados para os segmentos P_7P_8 e P_5P_6 e foram indecidíveis, aplicamos Algoritmo 1 neles e obtivemos, respectivamente, como resultados: P_7P_8 removido e o sub-segmento AB totalmente visível na janela. Em 1968, junto com o seu orientando Sproull, Sutherland projetou um algoritmo denominado **algoritmo de ponto médio** para determinar, de forma paralela, os sub-segmentos visíveis. Essencialmente, o algoritmo consiste em achar recursivamente as interseções de um segmento indecidível com os lados de uma janela, como mostra o seguinte pseudo-código da função Recorte_PM:

Input: Segmento P_1P_2
Output: P_AP_B
$P_A = P_1$;
$P_B = P_2$;
if *P_A e P_B tem código 0000* **then**
| Retorne P_AP_B como sub-segmento visível;
else if *AND(P_A,P_B) tem código 0000* **then**
| Retorne como sub-segmento visível;
else if $P_A = P_B$ **then**
| Retorne como sub-segmento visível;
else
| Ache o ponto médio $P_M = \frac{P_A+P_B}{2}$;
| Recorte_PM(P_1P_M, P_XP_Y);
| Recorte_PM(P_MP_2, P_ZP_W);
| $P_AP_B = P_XP_Y \cup P_ZP_W$;
end

Algoritmo 2: Função Recorte_PM: algoritmo de subdivisão por ponto médio.

Observamos que o procedimento recursivo Recorte_PM pode ser executado em paralelo. O algoritmo é, portanto, extremamente eficiente quando implementado em *hardware*.

A extensão do algoritmo de Cohen-Sutherland para um volume de visão delimitado por 6 planos é imediato. Basta adicionarmos mais 2 *bits* ao código para descarte rápido: quinto *bit* para $z < z_{min}$ e sexto *bit* para $z > z_{max}$, conforme ilustra Figura 6.3.(b). Por legibilidade, só mostramos os códigos dos sub-espaços $z < z_{min}$. Para $z_{min} \leq z \leq z_{max}$, os dois *bits* são 00 e para $z > z_{max}$, 10. Em casos indecidíveis, podemos utilizar o algoritmo de subdivisão por ponto médio (Algoritmo 2).

6.2.2 Algoritmo de Cyrus-Beck

Em 1978, Cyrus e Beck publicaram um algoritmo de recorte de segmentos em relação a qualquer janela convexa de n arestas. O algoritmo é fundamentado em um paradigma original. Eles utilizaram representação paramétrica para descrever segmentos (seção 3.2.1). Esta representação não só permite reduzir n coordenadas em um único parâmetro t, como também simplifica o procedimento para achar os sub-segmentos totalmente contidos na janela de exibição. Adicionalmente, eles definiram como **vetor normal de um lado** o vetor unitário perpendicular a este lado com direção para fora da janela e "traduziram" o problema de determinação de intersecção I entre um segmento e um lado E_i da janela, como determinação de um ponto I

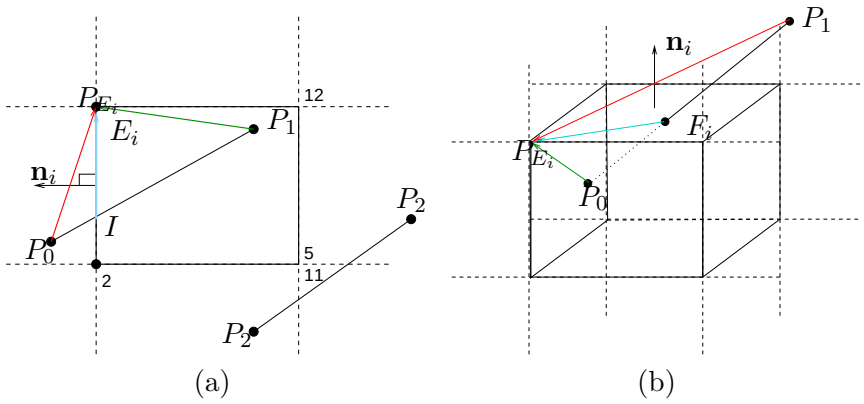


Figura 6.4: Algoritmo de Cyrus-Beck em: (a) janela de exibição e (b) volume de visão.

sobre este lado tal que IP_{E_i}, onde P_{E_i} é um outro ponto qualquer sobre o lado E_i, seja perpendicular ao vetor normal $\mathbf{n}_i$ de E_i (Figura 6.4.(a)).

Representando o segmento P_0P_1 parametricamente

$$P(t) = P_0 + (P_1 - P_0)t \tag{6.2}$$

e tomando um ponto fixo P_{E_i} da aresta E_i, o ponto de intersecção I entre $P(t)$ e E_i deve satisfazer

$$\mathbf{n}_i \cdot (P_{E_i} - P(t)) = 0.$$

Substituindo $P(t)$ pela Eq.(6.2), obtemos

$$\mathbf{n}_i \cdot (P_{E_i} - P_0) - \mathbf{n}_i \cdot (P_1 - P_0)t_i = 0,$$

do qual derivamos o valor do parâmetro t_i, $\mathbf{n}_i$ e $(P_1 - P_0)$ não forem perpendiculares

$$t_i = \frac{\mathbf{n}_i \cdot (P_0 - P_{E_i})}{\mathbf{n}_i \cdot (P_1 - P_0)}. \tag{6.3}$$

Se $t_i \in [0, 1]$, então o segmento intercepta a aresta E_i. O ponto de intersecção I é classificado de acordo com o produto escalar entre os vetores $P_{E_i} - P_0$ e $\mathbf{n}_i$

$$\theta_i = (P_{E_i} - P_0) \cdot \mathbf{n}_i \tag{6.4}$$

em

PL (sair da janela): se $\theta_i > 0$ e

PE (entrar na janela): se $\theta_i < 0$.

O procedimento é aplicado em relação a todos os lados da janela, a fim de determinar todas as possíveis intersecções. Estas intersecções são então ordenadas na ordem crescente através da ordenação crescente dos respectivos valores do parâmetro t. Por que? Basta lermos a Eq. 6.2 na seguinte forma: um ponto $P(t)$ é obtido deslocando a partir de P_0 em direção de um múltiplo t do vetor $(P_1 - P_0)$. Portanto, quanto maior for o valor de t, mais distante será o ponto $P(t)$ em relação ao ponto P_0. Em seguida verificamos, par a par, as classificações das intersecções ordenadas. O trecho entre as intersecções, classificadas como (PE,PL), (PE,dentro) ou (dentro,PL), é o trecho que está dentro da janela de recorte. Vamos ilustrar o procedimento com a Figura 6.4.(a), aplicando Eqs. 6.3 e 6.4 nos segmentos P_0P_1 e P_2P_3. Os resultados estão sintetizados na seguinte tabela:

	Lado i	$\mathbf{n}_i$	P_{E_i}	t_i	θ_i	PL/PE	Ordenação
$P_0 = (0,6)$	E_0	(-1,0)	(2,5)	$\frac{2}{9}$	-2	PE	3
$P_1 = (9,11)$	E_1	(0,1)	(2,12)	$\frac{6}{5} > 1$	6	PL	5
	E_2	(1,0)	(11,12)	$\frac{11}{9} > 1$	11	PL	6
	E_3	(0,-1)	(11,5)	$\frac{-1}{5} < 0$	1	PL	1
	P_0			0	-	(fora)	2
	P_1			1	-	(dentro)	4
$P_2 = (9,2)$	E_0	(-1,0)	(2,5)	$\frac{7}{-7} < 0$	7	PL	1
$P_3 = (16,7)$	E_1	(0,1)	(2,12)	$\frac{10}{5} > 1$	10	PL	6
	E_2	(1,0)	(11,12)	$\frac{2}{7}$	2	PL	3
	E_3	(0,-1)	(11,5)	$\frac{-3}{-5}$	-3	PE	4
	P_2			0	-	(fora)	2
	P_3			1	-	(fora)	5

A partir dos resultados, podemos ver que para P_0P_1 os valores $\{0, \frac{2}{9}, 1\}$ estão contidos no intervalo de parâmetros [0,1], mas somente o intervalo $\{\frac{2}{9}, 1\}$ (PE,dentro) está contido na janela. Em termos de coordenadas cartesianas, estes valores correspondem aos pontos $(2, \frac{64}{9})$ e $(9, 11)$. E para P_2P_3 não temos nenhum intervalo com classificação (PE,dentro), (PE,PL) ou (dentro,PL); portanto, o segmento está totalmente fora da janela.

O algoritmo Cyrus-Beck se aplica também para recorte de um segmento em relação a um volume de visão, para o qual cada lado é uma face F_i ao invés de ser um segmento E_i, como mostra a Figura 6.4.(b). Como o volume contém 6 faces, as Eqs 6.3 e 6.4 são executadas 6 vezes para cada segmento.

6.2.3 Algoritmo de Liang-Barsky

Liang e Barsky apresentaram uma versão mais eficiente do algoritmo de Cyrus-Beck para casos específicos de janelas retangulares ($x_{min}, x_{max}, y_{min}, y_{max}$). Observe na tabela da seção 6.2.2 que o fato dos vetores normais terem sempre uma coordenada nula e a outra com módulo igual a 1 simplifica tanto o cômputo de t_i quanto de θ_i, se escolhermos apropriadamente as coordenadas dos pontos P_{E_i} como mostra a seguinte tabela

Lado i	$\mathbf{n}_i$	P_{E_i}	t_i	θ_i
x_{min}	(-1,0)	(x_{min}, y)	$t_i = \frac{x_{min}-x_0}{x_1-x_0}$	$x_{min} - x_0$
x_{max}	(1,0)	(x_{max}, y)	$t_i = \frac{x_{max}-x_0}{x_1-x_0}$	$x_{max} - x_0$
y_{min}	(0,-1)	(x, y_{min})	$t_i = \frac{y_{min}-y_0}{y_1-y_0}$	$y_{min} - y_0$
y_{max}	(0,1)	(x, y_{min})	$t_i = \frac{y_{max}-y_0}{y_1-y_0}$	$y_{max} - y_0$

Compare as expressões para θ_i na última coluna da tabela com Eq. 6.4. Produto escalar foi reduzido em simples diferença entre dois valores reais!

Quando o recorte é em relação a um volume de visão, adicionamos mais dois lados: z_{min} e z_{max}. A derivação das expressões para obter t_i e θ_i é análoga aos outros lados.

6.3 Recorte de Polígonos

Embora um polígono possa ser descrito por uma sequência fechada de segmentos, utilizar os algoritmos dados na seção 6.2 pode tornar o "preenchimento" da área de um polígono recortado uma tarefa não trivial. Pois, ao aplicarmos os algoritmos de recorte de linhas, temos frequentemente uma coleção de segmentos não conectados, como ilustra Figura 6.5.(b). Para preencher o interior do polígono é necessário que a sequência de segmentos seja fechada, como mostra Figura 6.5.(c).

Nesta seção apresentamos dois algoritmos de recorte de polígonos. Adotamos a convenção de que os segmentos do contorno externo de um polígono sejam orientadas no sentido anti-horário e os segmentos do contorno interno orientados no sentido horário.

6.3.1 Algoritmo de Sutherland-Hodgman

Ao invés de determinar a parte do polígono que se encontra dentro da janela, Hodgman e Sutherland propuseram, em 1974, um procedimento de recortá-lo sucessivamente pelos lados da janela como se fosse cortá-lo com uma guilhotina ao longo das linhas da janela. Figura 6.6 ilustra esta idéia. Nesta

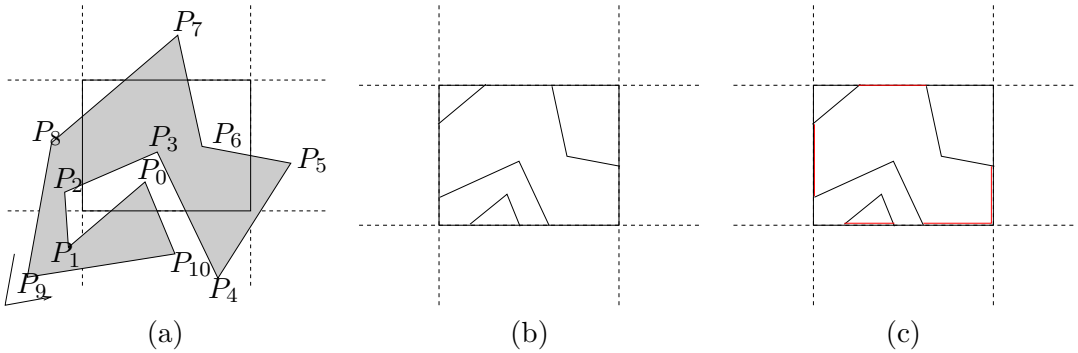


Figura 6.5: Recorte de um polígono: (a) original, (b) coleção de segmentos aberta, e (c) sequência fechada de segmentos.

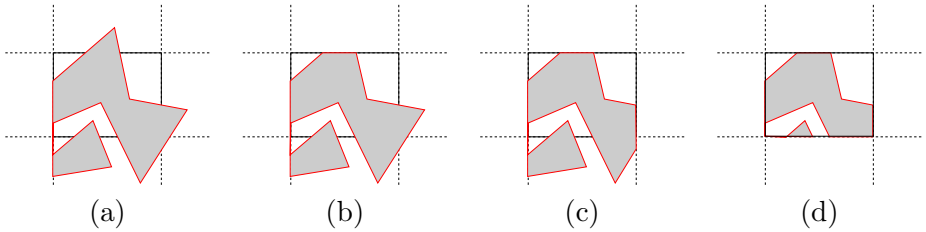


Figura 6.6: Algoritmo de Sutherland-Hodgman: (a) por x_{min}, (b) por y_{max}, (c) por x_{max} e (d) por y_{min}.

seção mostraremos como fazer um computador realizar esta tarefa simples, através de manipulação de números.

O polígono é originalmente definido como uma sequência fechada de vértices $P_0P_1 \ldots P_{n-2}P_{n-1}P_n$. Esta sequência é recortada, em primeiro lugar, com a linha $x = x_{min}$, gerando uma nova lista intermediária de vértices com base em quatro regras de inserção de vértices ao percorrer a sequência original de P_0 até P_n (Figura 6.6.(a)):

1. se o vértice P_i estiver no lado esquerdo da linha $x = x_{min}$, insere P_i na lista (Figura 6.7.(a));

2. se o vértice P_{i-1} estiver no lado esquerdo da linha e o vértice P_i no lado direito, determine a intersecção I (do tipo PL) e insere I na lista (Figura 6.7.(b));

3. se o vértice P_{i-1} estiver no lado direito da linha e o vértice P_i no lado esquerdo, determine a intersecção I (do tipo PE) e insere I seguido de P_i na lista (Figura 6.7.(c));

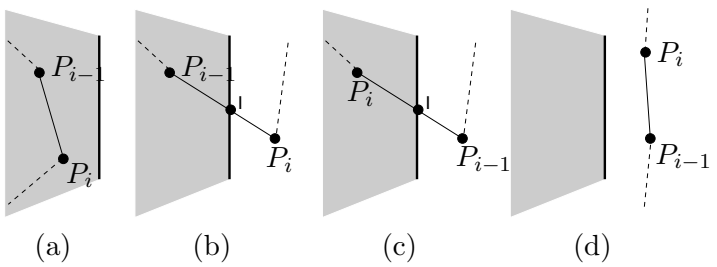


Figura 6.7: Quatro possíveis configurações: (a) dentro, (b) saindo (PL), (c) entrando (PE) e (d) fora.

4. se os vértices P_{i-1} e P_i estiverem ambos no lado direito da linha, nenhuma inserção é feita (Figura 6.7.(d)).

O processo inicia com a classificação de pertinência do último ponto da sequência, P_n, e rotulação deste ponto como P_{-1}. Se o ponto estiver dentro da janela, ele é inserido na nova lista. Esta rotulação é somente para uniformizar o tratamento do segmento P_nP_0 com relação ao restante dos segmentos.

A lista intermediária gerada é utilizada como entrada do procedimento de recorte em relação à linha $y = y_{max}$, gerando uma segunda lista intermediária (Figura 6.6.(b)). E o processo é repetido até que o polígono tenha sido "recortado" em relação a todos os lados da janela.

Resta agora verificar como se determinam as intersecções e como se "computam" os lados de uma linha. Para obter intersecções entre dois segmentos, podemos utilizar os algoritmos apresentados na seção 6.2.2. E para distinguir os lados, podemos utilizar a representação implícita da linha para avaliar o lado, conforme vimos na seção 3.2.1, ou determinar o ângulo entre o vetor normal da linha e o vetor formado pelo ponto P_i e um ponto da linha e avaliar o sinal do produto escalar, conforme vimos na seção 6.2.2.

O princípio do algoritmo de Hodgman-Sutherland vale também para um volume de visão. Neste caso, ao invés de passar 4 linhas cortantes sucessivamente, passaremos 6 planos cortantes sucessivamente.

6.3.2 Algoritmo de Weiler-Atherton

O algoritmo de Hodgman-Sutherland serve somente para janelas

convexas, ou seja, para janelas que são totalmente contidas no mesmo semi-espaço em relação a cada um dos seus lados, e

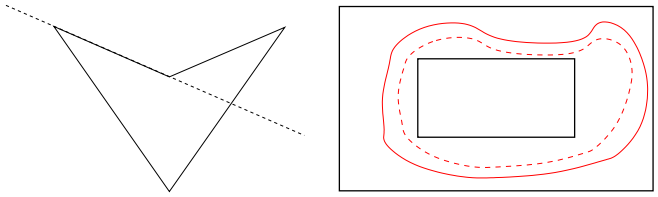


Figura 6.8: Janela: (a) côncava e (b) multiplamente conexa.

simplesmente conexas, ou seja, se desenharmos uma curva fechada no seu interior podemos "encolhê-la" continuamente em um ponto sem sair da janela.

Figura 6.8.(a) ilustra uma janela côncava e Figura 6.8.(b) uma janela multiplamente conexa. Como um computador conseguiria "recortar" uma figura plana "preenchida" por um contorno arbitrário?

Em 1977, Weiler e Atherton apresentaram um algoritmo genérico, capaz de determinar o recorte entre dois polígonos $\mathcal{P}$ e $\mathcal{Q}$ quaisquer. O polígono a ser recortado é denominado **polígono-sujeito** e o polígono recortante, **polígono-recorte**. O diferencial do seu algoritmo está no fato de substituir a parte da "borda" do polígono-sujeito pela "borda" do polígono-recortante, ao invés de tentar reconstruir a fronteira na parte removida.

Eles utilizam listas para representar os dois polígonos. Cada lista corresponde a uma sequência fechada de vértices de um contorno dos polígonos e ela já inclui todas as intersecções entre o polígono-sujeito e o polígono-recorte. Temos, portanto, um conjunto de listas L_S para polígono-sujeito e outro para polígono-recorte L_R, como ilustra Figura 6.9. Na inicialização são ainda criadas duas listas: uma lista L_{PE} de intersecções do tipo PE, entre os dois polígonos, e uma lista L_{PL} de intersecções do tipo PL. O procedimento de recorte, propriamente dito, é sintetizado no seguinte pseudo-código:

Input: L_S, L_R, L_{PE}, L_{PL}
Output: listas de contornos do polígono recortado
while L_{PE} *não é vazia* **do**
  Retire um vértice V_S de L_{PE} e o localize em L_S;
  $I = V_S$;
  $V_E = 0$;
  while $I \neq V_E$ **do**
    Percorre L_S a partir de V_S até encontrar um vértice V_E de L_{PL}, inserindo os vértices percorridos na lista de saída ;
    Procure em L_R o vértice V_E ;
    Percorre L_R a partir de V_E até encontrar um vértice V_S de L_{PE}, inserindo os vértices percorridos menos V_E e V_S ;
    Retire V_E e V_S das respectivas listas;
  end
end

Algoritmo 3: Algoritmo de Weiler-Atherton.

É difícil encontrar nas aplicações um volume de visão côncavo e/ou multiplamente conexo. Até onde temos informação, este algoritmo não é aplicado no fluxo de projeção.

6.4 Espaço de Recorte

Agora que mostramos diferentes alternativas para um computador "recortar" uma figura geométrica, vamos ver qual é o momento mais apropriado para remover as partes que estão fora do volume de visão. Chamamos o espaço onde ocorre recorte de **espaço de recorte**.

Comparando os tipos de geometria que os algoritmos de recorte conseguem trabalhar eficientemente e as formas que um volume de visão pode assumir em cada espaço (capítulo 5.4), o recorte deve ocorrer preferencialmente após os ajustes no espaço VRC quando as equações que representam os planos do volume de visão são mais simples. A pergunta é em qual ponto devemos inserir o espaço de recorte? Se for uma projeção paralela, a forma do volume de visão seria preservada até a projeção efetiva. Só o seu tamanho é alterado. Portanto, pode-se inserir, em princípio, o espaço de recorte em qualquer ponto após os ajustes no espaço VRC. O mesmo, infelizmente, não acontece com o fluxo de projeção perspectiva. Vamos verificar isso mais detalhadamente.

Vimos na seção 5.4.3 que o volume de visão canônico perspectivo é "radicalmente" deformado em um volume canônico paralelo após a divisão pela

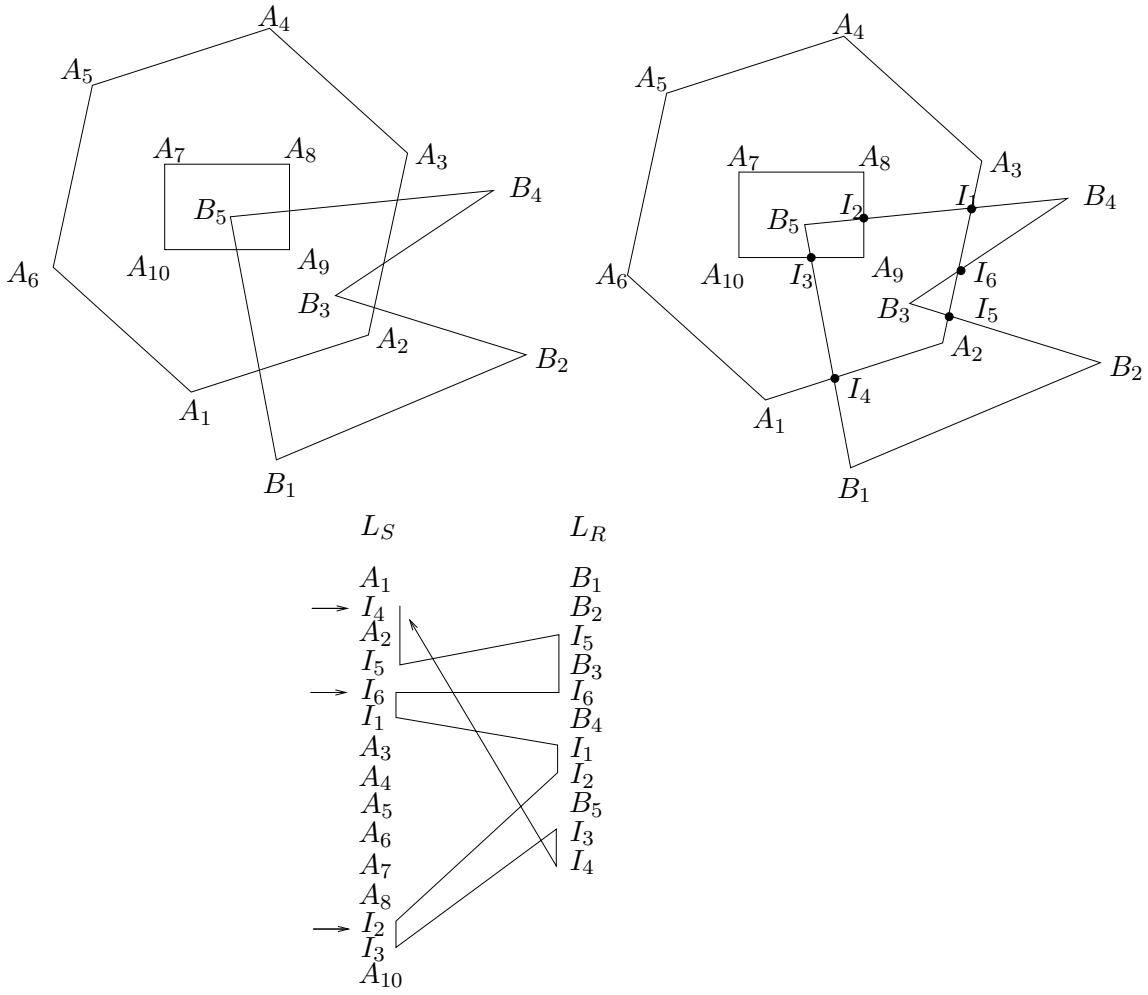


Figura 6.9: Recorte com uso de algoritmo de Weiler-Atherton.

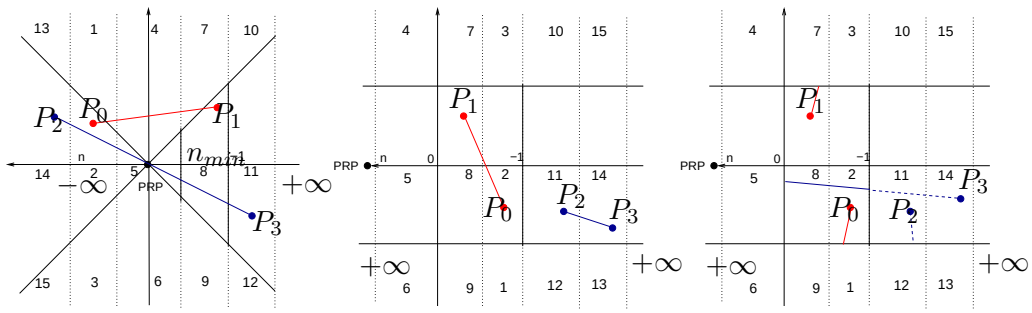


Figura 6.10: Sub-espaços associados ao volume canônico: (a) antes da divisão, (b) após a divisão e (c) recorte correto.

coordenada w. Tanto os pontos com as coordenadas que satisfazem

$$-w \leq u \leq w \qquad -w \leq v \leq w \qquad -w \leq n \leq 0 \quad w > 0$$

quanto os que atendem

$$-w \geq u \geq w \qquad -w \geq v \geq w \qquad -w \geq n \geq 0 \quad w < 0$$

estariam dentro do volume canônico paralelo após a divisão por $w \neq 0$

$$-1 \leq \frac{u}{w} \leq 1 \qquad -1 \leq \frac{v}{w} \leq 1 \qquad -1 \leq \frac{n}{w} \leq 0,$$

embora eles tenham posições bem distintas no espaço "homogêneo". Os primeiros pontos estariam em frente do centro de projeção PRP (sub-espaços 5, 6 e 10 da Figura 6.10.(a)) e os pontos do segundo grupo, atrás do centro de projeção (sub-espaços 2 e 14 da Figura 6.10.(a)). Ao dividirmos as coordenadas pela coordenada w, o centro de projeção PRP será "levado" para o "infinito" e os sub-espaços 1, 2, 3, 13, 14 e 15 "rebateriam" sobre os sub-espaços 9, 8, 7, 12, 11 e 10, respectivamente, conforme ilustra a Figura 6.10.(b). Isso significa que pontos que ficam atrás do centro de projeção podem ser transformados para frente dele e os segmentos serem descartados de forma equivocada após a divisão por w, como o segmento P_2P_3 na Figura 6.10.(b), ou serem conectados de forma incorreta pensando que o resultado seja um único segmento, como os pontos-extremos do segmento P_0P_1 na Figura 6.10.(b). Na Figura 6.10.(c) são esboçadas as projeções corretas.

Concluindo, o espaço de recorte deve ser inserido antes da divisão pela coordenada w ao longo do fluxo de projeção quando o interesse é somente

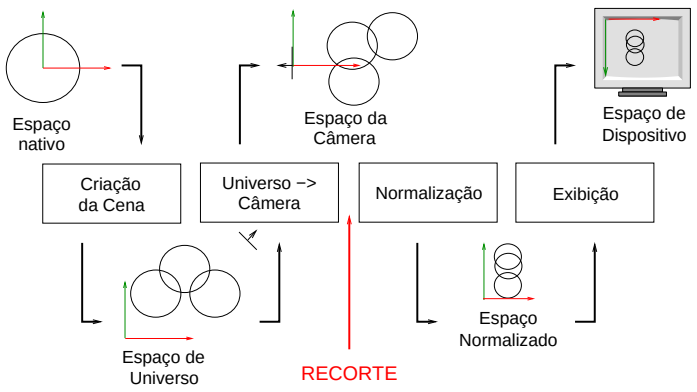


Figura 6.11: Fluxo de projeção com recorte.

a geometria das figuras da cena. Figura 6.11 apresenta o fluxo de projeção com espaço de recorte.

Precisaremos, então, utilizar as seguintes inequações para determinar as intersecções e decidir a pertinência de um ponto em relação ao volume de visão

$$-|w| \leq u \leq |w| \qquad -|w| \leq v \leq |w| \qquad -|w| \leq n \leq 0$$

Vamos supor que o plano de recorte seja $\frac{u}{w} = 1$, da qual deriva-se

$$u - w = 0$$

Para achar a intersecção de um segmento dado pela Eq. 6.2 com este plano de recorte, temos que achar t tal que

$$[(1-t)w_0 + tw_1] = [(1-t)u_0 + tu_1].$$

Separando t obtemos

$$t = \frac{w_0 - u_0}{(w_0 - u_0) - (w_1 - u_1)}.$$

Este procedimento é aplicado, de forma análoga, para obter intersecções com outros 5 planos de recorte: $\frac{u}{w} = -1$, $\frac{v}{w} = 1$, $\frac{v}{w} = -1$, $\frac{u}{w} = 0$, $\frac{n}{w} = -1$. As intersecções são, então, utilizadas nos algoritmos de recorte que vimos nas seções anteriores para obter recorte de segmentos e polígonos.

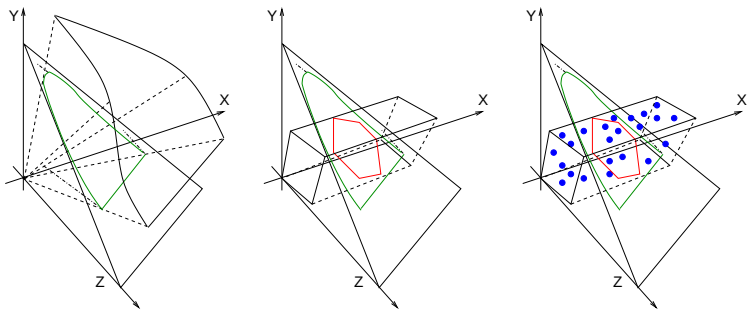
Capítulo 7

Modelos de Cor

O objetivo deste capítulo é apresentar alternativas para representar cores através de valores numéricos, de forma que um sistema digital possa "vê"-la e manipulá-la. Após a leitura deste capítulo, você deve ser capaz de

- definir os seguintes termos: matiz, saturação e brilho.
- distinguir os espaços de cor perceptiva, os espaços de cor reproduzível por um dispositivo (gamutes de cor) e os espaços de cor digitais.
- descrever o princípio de percepção de cor no sistema visual humano.
- descrever os principais modelos de cor: padrão de cor e gamute de cor.
- interpretar e aplicar o diagrama de cromaticidade.
- "ver" as cores através de números e manipulá-las com operações algébricas.
- mapear uma cor entre distintos espaços de cor.

A maioria das pessoas vive rodeada de uma natureza multicolorida: céu azul, nuvens brancas, folhas verdes, etc. Desde Antiguidade tentou-se reproduzir estas cores em pinturas e vestimentas utilizando produtos naturais, extraídos das plantas e insetos. Só em 1856 o químico William Henry Perkin inventou a primeira tinta artificial obtida com compostos químicos manipulados em laboratório. Independentemente de ser natural ou artificial, o fascínio pelas cores tem motivado pesquisadores de diversas áreas. Isso contribuiu ao uso de termos distintos para denotar uma mesma grandeza física, como veremos na seção 7.1. A história de Computação Gráfica não é



(a) (b)

Figura 7.1: Espaços de cor: (a) perceptível pela visão, (b) reproduzível pelo dispositivo de exibição e (c) representável em sistemas digitais.

muito diferente. Assim que foram geradas as primeiras imagens sintéticas, o apelo por imagens coloridas, mais próximas possíveis da nossa percepção, tem impulsionado muitas pesquisas relacionadas com a produção de cores.

Neste capítulo o nosso foco se limita à representação de cores no mundo digital, ou seja, às formas alternativas como um computador pode "ver" as cores de forma equivalente a um observador médio. Para isso, precisamos saber quais cores que um observador médio consegue perceber e como "ensiná-las" para um sistema digital. Na seção 7.2 daremos uma visão geral sobre o nosso processo de percepção de cor. Em seguida, apresentamos na seção 7.3 **modelos tricromáticos** para descrever numericamente o espaço de cores perceptíveis (Figura 7.1.(a)). Dois **padrões tricromáticos** serão apresentados. Veremos que podemos decompor uma cor em duas componentes: luminância e crominância. Esta última pode ser representada por um **diagrama de cromaticidade**. Na seção 7.4 mostraremos como se representa o espaço de matizes de cores através de um diagrama de cromaticidade. Na seção 7.5 descreveremos alguns **modelos de cor** mais utilizados para representar os gamutes de cores que são, de fato, um sub-espaço de cor como ilustra na Figura. 7.1.(b). As cores que são processadas em um sistema digital devem estar contidas no gamute de cores do dispositivo de saída para que elas sejam "exibíveis" (pontos cianos na Figura. 7.1.(c)). E, finalmente, na seção 7.6 mostraremos como transferir estes modelos para o mundo digital.

7.1 Terminologia

Uma **onda** é uma perturbação oscilante de alguma grandeza física no espaço e periódica no tempo. A oscilação espacial é caracterizada pelo **comprimento de onda** (λ) e a periodicidade no tempo (T) é medida pela freqüência da onda ($f = \frac{1}{T}$). Estas duas grandezas estão relacionadas pela velocidade de propagação da onda (v)

$$\lambda = \frac{v}{f}.$$

Energia radiante é a modalidade de energia que se propaga pelo espaço, mesmo vazio, por meio de ondas eletromagnéticas. **Radiação** é a energia radiante em uma mesma frequência. A frequência de uma radiação não se modifica quando ela passa de um meio para outro. Dá-se o nome de **espectro eletromagnético** ao conjunto de todas as radiações eletromagnéticas ordenadas segundo as suas frequências, ou os seus **comprimentos de onda**. O **espectro de luz visível** é um intervalo de espectro eletromagnético capaz de sensibilizar a visão humana, produzindo **percepção de cor**. A cada comprimento de onda é associada uma **cor espectral** ou **cor monocromática**. **Fotometria** trata do estudo das grandezas relativas à emissão e à recepção das radiações visíveis e de medição dessas grandezas.

A sensibilidade da visão humana varia de observador a observador. Para um mesmo observador, ele varia com a **distribuição espectral de potência** $I(\lambda)$, provocando distintos estímulos cromáticos. Figura 7.2.(a) apresenta uma típica distribuição espectral de potência de uma única cor. Sob o ponto de vista perceptivo, os termos importantes associados a uma cor são: matiz, saturação e brilho. Denomina-se **matiz** (*hue*) o atributo de sensação colorida que permite dizer se uma cor é verde ou azul. A **saturação** (*saturation*) é o atributo de sensação colorida que permite dizer se uma cor é mais clara ou mais escura. Chamamos de **claridade** (*lightness*) o atributo de sensação colorida que permite dizer se uma cor é mais ou menos "ofuscante". As setas na Figura 7.2.(b) indicam a direção do aumento do valor desses parâmetros. O matiz, a saturação e a claridade correspondem, respectivamente, às seguintes grandezas físicas da **Colorimetria**: **comprimento de onda dominante** λ, **pureza na excitação** ("percentagem da luz branca") e **luminância** $\mathcal{E}$ (Figura 7.2.(c)). Diferentemente da Fotometria, Colorimetria se ocupa dos métodos referentes à medição e à avaliação das radiações luminosas percebidas pela visão humana.

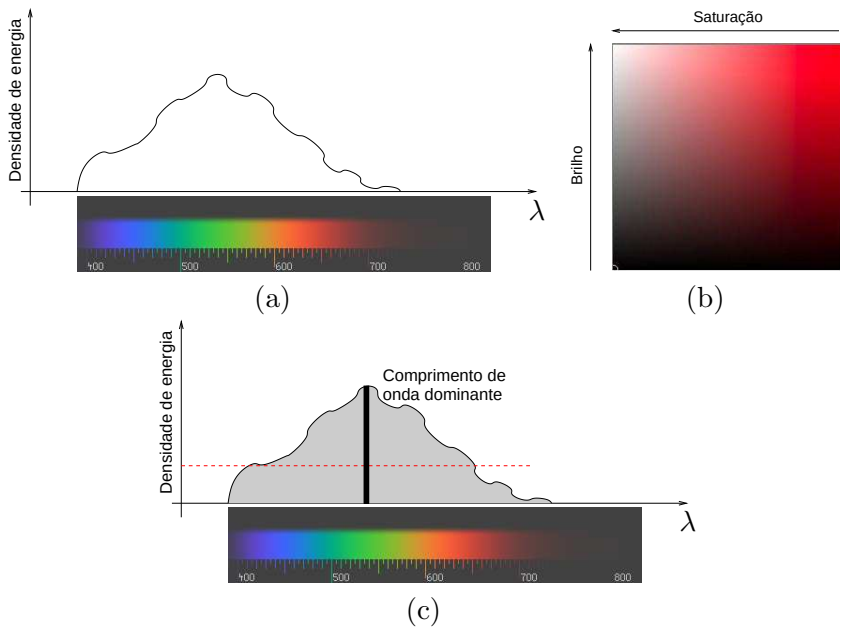


Figura 7.2: Uma cor em: (a) distribuição espectral típica, (b) medidas de percepção e (b) medidas colorimétricas.

7.2 Percepção Visual

O termo luz é usado em dois sentidos. No sentido físico, luz é um feixe de ondas eletromagnéticas/fótons de energia ao qual o olho humano é sensível. No sentido psicofísico, a luz é a sensação produzida num observador quando a sua retina é estimulada.

A faixa espectral de luz visível, embora limitada, compreende um número infinito de comprimentos de onda o que, à primeira vista, impossibilitaria a definição de um modelo computacionalmente factível. Felizmente, graças à limitada capacidade de resolução da visão humana, a quantidade de cores que um dispositivo com 24 *bits* consegue representar é suficiente para modelar e reproduzir uma variedade de cores comparável com a que vivenciamos no dia-a-dia. Quando um observador estiver adaptado ao estímulo de um espectro de cor, ele só vai perceber variação na cor se alterarmos o estímulo por uma **diferença somente distinguível**, em inglês *just noticeable difference*. Figura 7.3 ilustra a capacidade de discriminar duas cores espectrais, colocadas uma ao lado da outra, em três distintos níveis de iluminação medidos em *trolands*. Observe que a melhor discriminação ocorre em torno de

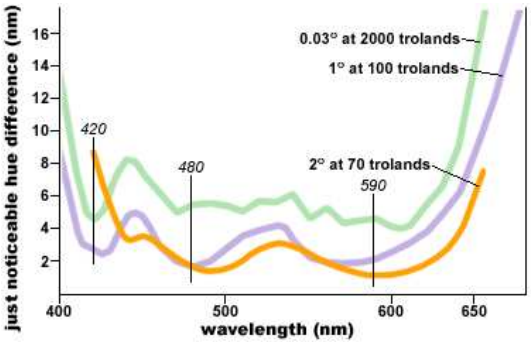
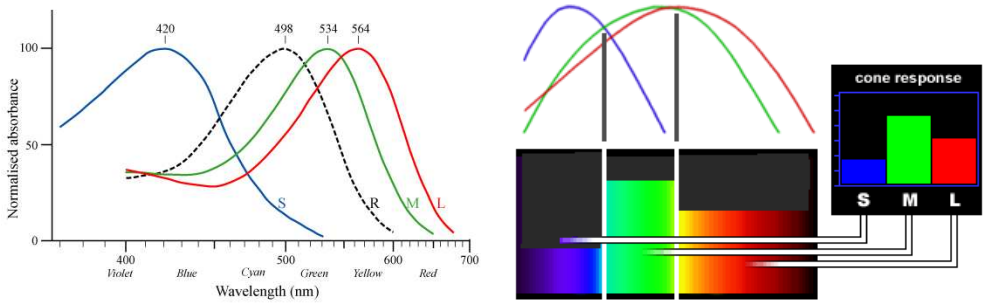


Figura 7.3: Discriminação de cores espectrais (Fonte: http://www.handprint.com/HP/WCL/color2.html).

490 nm (ciano) e 595 nm (laranja), quando a diferença é entre 1 a 2 nm.

Vimos no capítulo 5 que a imagem de um objeto se forma sobre o plano retinal e que se distinguem na retina dois tipos de células: os cones e os bastonetes. Os cones são destinados à **visão fotópica** (com intensa iluminação), sendo sensíveis às diferenças de cor. Os bastonetes, por sua vez, são destinados à **visão escotópica** (com fraca iluminação), sendo menos sensívesis às diferenças de cor. Com base em experimentos, foi proposto um modelo "tri-estímulo" para a visão humana (modelo de Young-Helmholtz). Este modelo se fundamenta no fato de que a retina possui essencialmente três tipos de cones que são, respectivamente, fotosenssíveis às faixas de baixa(S), média (M) e alta (L) frequência do espectro visível, como comprovam os gráficos das **funções (noramlizadas) de sensibilidade espectral**, em inglês *spectral sensitivity*, dos tipos de cones na Figura 7.4. (a). Observe na Figura 7.4.(b) que as duas intersecções entre as três curvas "dividem" o espectro em faixa azul, verde e vermelha, para as quais os cones S, M e L são, respectivamente, **receptores dominantes**.

Maxwell demonstrou através dos seus experimentos de casamento de cor que qualquer cor espectral se casa com uma específica combinação de somente 3 **cores primárias**, conforme mostra Figura 7.5.(a). Utilizando uma versão moderna do seu procedimento, o **método de saturação máxima**, é possível "levantar" as proporções de três cores espectrais, B=444nm, G=526nm e R=645nm, necessárias para reconstruir cada uma das cores espectrais. As letras $b_{10}(\lambda)$, $g_{10}(\lambda)$ e $r_{10}(\lambda)$ da Figura 7.5.(b) denotam as **funções de casamento de cor**, em inglês *color matching function*. Elas representam, respectivamente, proporções da mistura das luzes monocromáticas B, G e R para chegar a uma cor espectral. Por exemplo, para obter a cor mo-



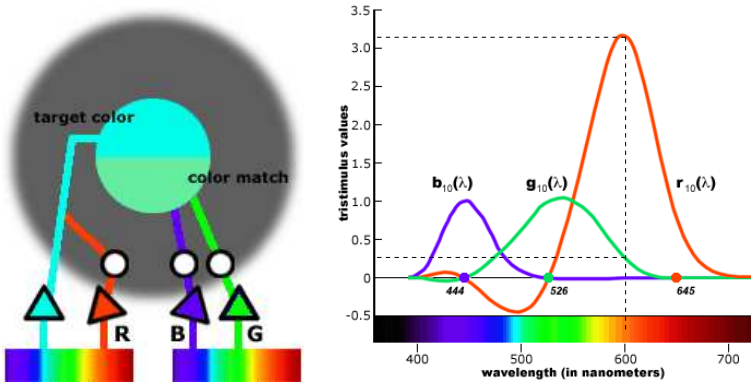
(a) (b)

Figura 7.4: Resposta espectral dos cones: (a) curvas de sensibilidade (Fonte: http://www.jpse.co.uk/sensory/colourtheory.shtml) e (b) faixa de dominância de cada cone (Fonte: http://www.handprint.com/HP/WCL/color3.html).

nocromática de 600nm é necessária a proporção 0:0.25:3.15. A teoria tricromática consegue explicar vários tipos de sensação de cor que vivenciamos. No entanto, há ainda casos observados experimentalmente que aparentemente ficavam sem resposta, entre os quais o seguinte fenômeno intrigante: por que não conseguimos perceber "vermelho esverdeado", "amarelo azulado" ou "preto esbranquiçado"?

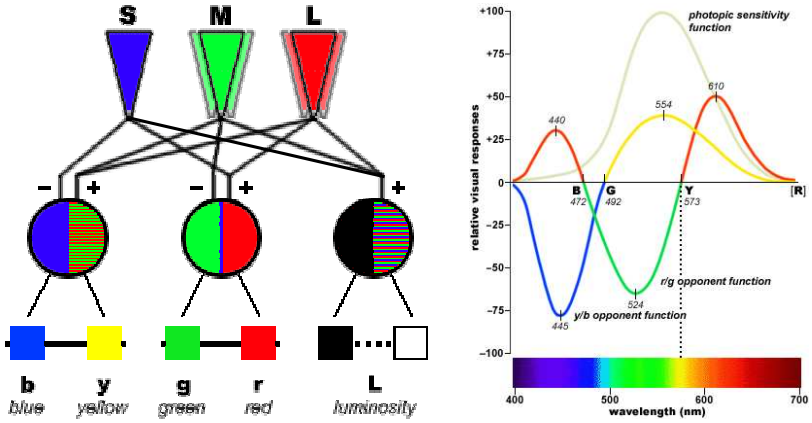
Em torno de 1874, Hering publicou uma monografia que defendia a **teoria de processo oponente**. Segundo a teoria, os sinais recebidos pelos cones e bastonetes são "interpretados" antes de produzir, de forma antagônica, uma específica percepção de cor como ilustra a Figura 7.6.(a): vermelho (L+S) × verde (M); azul (S) × amarelo (L+M); e preto × branco (L+M+S). Há fortes indícios de que exista um ponto de equilíbrio visual de cones L e M, no intervalo 570-575 nm. Estímulos neste ponto de equilíbrio causa a sensação de uma "nova" cor – "amarelo". Ao invés de um modelo de três cores, por causa de três tipos de cones, Hering propôs um modelo de quatro **cores puras**, em alemão *Urfarben*: além das três cores primárias "azul", "verde" e "vermelho", a cor "amarelo" é também considerada uma cor básica. Hurvich e Jameson demonstraram em 1950 que todas as cores espectrais podem ser decompostas em uma específica mistura de quatro cores: vermelho (em torno de 700nm), amarelo (em torno de 570nm), verde (em torno de 500nm) e azul (em torno de 470nm). Figura 7.6.(b) mostra a função de casamento de cor de um indivíduo médio.

A **eficiência luminosa** do olho é a sensibilidade do olho a uma luz de mesma potência, variando somente no comprimento de onda. Figura 7.7 mostra as **funções de eficiência luminosa** para uma visão fotópica, quando



(a) (b)

Figura 7.5: Funções de casamento de cor: (a) método de saturação máxima e (b) funções de casamento (Fonte: http://www.handprint.com/HP/WCL/color6.html).



(a) (b)

Figura 7.6: Processo oponente: (a) processamento ddos sinais S, M e L, e (b) funções de casamento (Fonte: http://www.handprint.com/HP/WCL/color2.html).

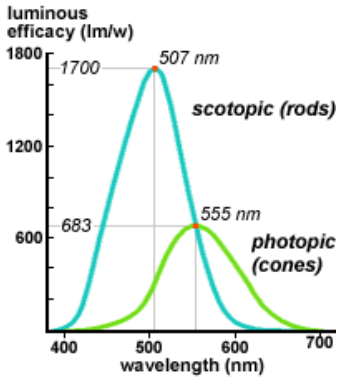


Figura 7.7: Funções de eficiência luminosa para visão fotópica e escotópica (Fonte: http://www.handprint.com/HP/WCL/color3.html).

os cones, em inglês *cones*, respondem melhor, e uma visão escotópica, quando os bastonetes, em inglês *rods*, apresentam melhor resposta. A cada composição espectral corresponde um estímulo cromático (uma única cor). Entretanto, um mesmo estímulo cromático pode ser causado por distintas composições espectrais. Tais composições espectrais são denominadas **metâmeras**. Como uma mesma radiação visível pode produzir sensação de cor diferente em observadores diferentes, duas cores metâmeras para um observador podem não as serem para outro.

7.3 Geometria de Cores

A breve introdução à percepção de cor dada na seção 7.2 é suficiente para percebermos a complexidade do processo. Precisamos agora ver como podemos modelar as mais diversas "sensações" de cor em valores numéricos para nos comunicar com um sistema digital e fazê-lo "perceber" tais cores também. A versão algébrica da teoria de percepção tricromática de cores foi baseada em uma série de observações do Grassmann, que hoje são conhecidas como **leis de Grassmann**:

Aditividade: dadas duas metâmeras x e y, isto é $x \sim y$. Se adicionarmos a elas uma mesma proporção de uma terceira cor z, então $x+z$ e $y+z$ são também metâmeras, ou seja, $x + z \sim y + z$.

Proporcionalidade: dadas duas metâmeras x e y. Se alterarmos a composição das três cores primárias nas duas cores pelo mesmo fator de proporção α, então αx e αy são também metâmeras, ou seja, $\alpha x \sim \alpha y$.

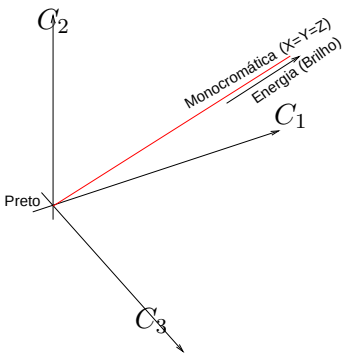


Figura 7.8: Geometria de cor: um espaço vetorial.

Transitividade: As cores x e a são metâmeras como as duas cores y e a são metâmeras, isto é $x \sim a$ e $y \sim a$, se e somente se, x e y são metâmeras.

A cada cor C é definida uma **cor complementar** $-C$ de forma que a mistura aditiva das duas cores resulta em uma **luz acromática** (cor "branco"). A luz acromática de um sistema é uma cor que adicionada a qualquer outra do sistema não altera o matiz da cor, mas sim só a sua saturação ou pureza. Estas propriedades junto com as leis de Grassmann nos permitem dizer que o conjunto de cores provido da operação de mistura aditiva e da multiplicação por um fator de potência α forma um **espaço (linear) de cor** gerado por três cores primárias $\{C_1, C_2, C_3\}$. Definimos como a base deste espaço as três cores primárias e a origem dele a luz acromática de luminância nula (luz "preta"), como esquematizado na Figura 7.8. Esta visão vetorial de cor nos abre a possibilidade de "traduzir" manipulações sobre as cores em operações vetoriais e transformações lineares (Seção 4.3), como veremos no restante deste capítulo.

A Comissão Internacional de Iluminação (CIE), órgão responsável pela padronização na área de Fotometria e Colorimetria, estabeleceu vários padrões que descrevem geometricamente as cores. O padrão mais encontrado em livros-texto de sistemas de informação gráfica é o CIE-XYZ 1931, que pode ser considerada uma transformação linear do espaço padrão CIE-RGB.

7.3.1 CIE-RGB

Baseado na teoria tricromática, Wright e Guild realizaram independentemente na década de 1920 uma série de experimentos para determinar as funções de casamento das três cores primárias R, G e B com um grande número de observadores. CIE padronizou, com base nos seus resultados,

as três cores primárias, selecionando as cores espectrais de comprimentos iguais a B=435.8 mn (azul), G=546.1 mn (verde) e R=700 mn (vermelho), e a partir das quais foram "levantadas" as funções de casamento $\overline{r}(\lambda)$, $\overline{g}(\lambda)$ e $\overline{b}(\lambda)$ com base em um ângulo de visão de 2^o de um observador-padrão CIE. Estas funções foram normalizadas de forma que as áreas sob elas sejam iguais, isto é,

$$\int_0^\infty \overline{r}(\lambda)d\lambda = \int_0^\infty \overline{g}(\lambda)d\lambda = \int_0^\infty \overline{b}(\lambda)d\lambda.$$

Em seguida estes valores foram multiplicados por um fator de escala para serem ajustados a uma fonte de luminância de cor (r,g,b) na proporção 1:4.5907:0.0601 e a uma fonte de radiações de cor (r,g,b) na proporção 72.0962:1.3791:1. Como resultado obteve-se as curvas de casamento mostradas na Figura 7.9.(a). Os gráficos são bem similares aos apresentados na Figura 7.5.(b).

Dada uma cor com a distribuição espectral de potência $I(\lambda)$, então os valores das coordenadas R, G e B podem ser calculadas por

$$\begin{aligned} R &= \int_0^\infty \overline{r}(\lambda)I(\lambda)d\lambda \\ G &= \int_0^\infty \overline{g}(\lambda)I(\lambda)d\lambda \\ B &= \int_0^\infty \overline{b}(\lambda)I(\lambda)d\lambda. \end{aligned} \tag{7.1}$$

Ao invés de especificarmos em termos da luminância que cada cor primária contribui para uma específica sensação de cor, normalizamos esta representação em relação à "luminância total" $R+G+B$ de forma a ter uma representação só de cromas de cor sem variações em luminância

$$r = \frac{R}{R+G+B} \tag{7.2}$$

$$\begin{aligned} g &= \frac{G}{R+G+B} \\ b &= \frac{B}{R+G+B}. \end{aligned} \tag{7.3}$$

Os valores r, g e b são denominadas **coordenadas de cromaticidade** de uma cor e o lugar geométrico dos pontos (r,g,b) é o plano $r+g+b=1$,

conhecido como **plano de Maxwell**, conforme mostra Figura 7.9.(b). As cores espectrais constituem uma curva aberta com um extremo em "roxo" (B=380nm) e outro em "vermelho" (R=700nm). O segmento de cores BR obtido pela combinação convexa entre "roxo" e "vermelho" é denominada **linha púrpura** (Seção 3.1.2). Pelas leis de Grassmann, todas as cores fisicamente realizáveis devem estar contidas nesta região convexa de formato de ferradura. Relembrando os conceitos que vimos na seção 4.5, as coordenadas (r, g, b) de um ponto sobre o plano são homogêneas a todas as "cores múltiplas" cujo lugar geométrico é a reta que passa pelo ponto e pela origem, pois

$$\begin{aligned} r &= \frac{\alpha R}{\alpha(R+G+B)} \\ g &= \frac{\alpha G}{\alpha(R+G+B)} \\ b &= \frac{\alpha B}{\alpha(R+G+B)}. \end{aligned} \tag{7.4}$$

(7.5)

Qual é a interpretação física para estes pontos? Eles representam as cores que tem a mesma proporção de cores primárias, diferindo apenas em luminância. Na Figura 7.9.(b) a linha em verde ilustra 1 conjunto de cores $(\alpha r, \alpha g, \alpha b)$ homogêneas a uma cor sobre o plano (r, g, b).

Projetando ortograficamente as cores contidas na "região de ferradura" do plano de Maxwell sobre o plano rg obtemos um **diagrama de cromaticidade**. Este diagrama é mostrado na Figura 7.9.(c). Observe que há uma grande área em que alguma das coordenadas assume valores negativos. Estes valores negativos indicam que a luz primária precisa ser adicionada à luz de teste: um pouco de verde para as cores que ficam no extremo "roxo" da faixa espectral luminosa e vermelho para as cores espectrais entre "azul" e "verde".

7.3.2 CIE-XYZ

De forma análoga ao padrão RGB, foram definidos três valores de estímulo X, Y e Z, os quais podem ser obtidos a partir da distribuição espectral de potência $I(\lambda)$ e das funções de casamento $\overline{x}(\lambda)$, $\overline{y}(\lambda)$ e $\overline{z}(\lambda)$

$$X = \int_0^\infty \overline{x}(\lambda) I(\lambda) d\lambda$$

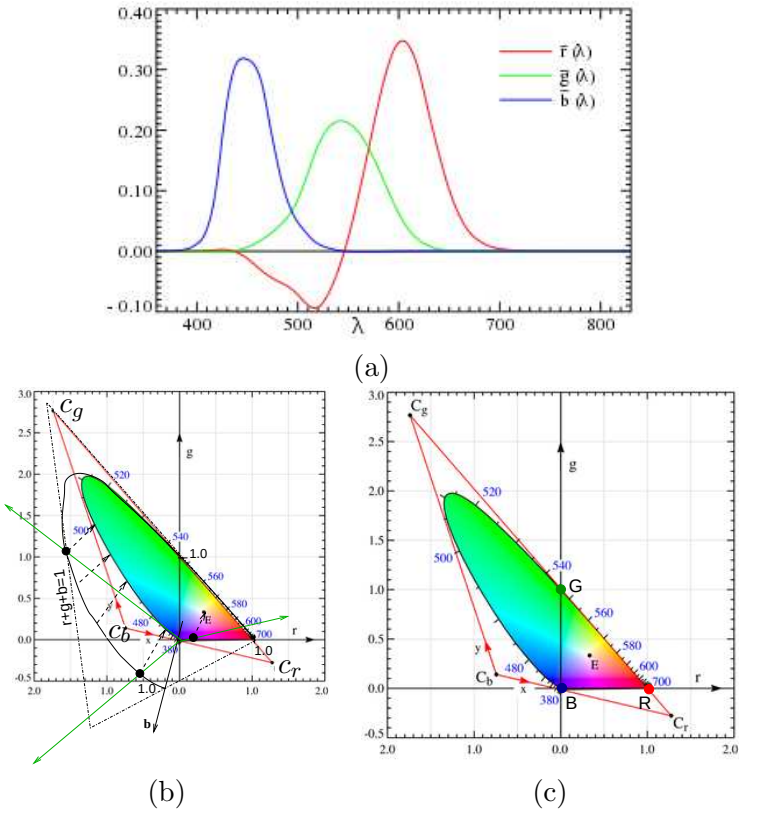
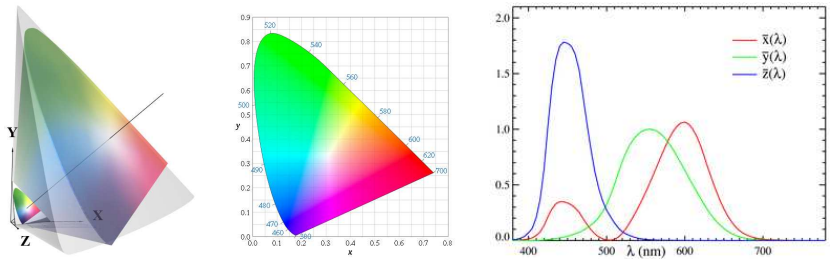


Figura 7.9: Padrão CIE-RGB: (a) funções de casamento; (b) espaço RGB; e (c) diagrama de cromaticidade.



(a) (b) (c)

Figura 7.10: CIE-XYZ: (a) espaço de cores; (b) diagrama de cromaticidade e (b) funções de casamento.

$$\begin{aligned} Y &= \int_0^\infty \overline{y}(\lambda) I(\lambda) d\lambda \\ Z &= \int_0^\infty \overline{z}(\lambda) I(\lambda) d\lambda \end{aligned} \tag{7.6}$$

e as coordenadas de cromaticidade x, y e z são obtidas através da normalização destes valores (Figura 7.10.(a))

$$x = \frac{X}{X+Y+Z} \tag{7.7}$$

$$\begin{aligned} y &= \frac{Y}{X+Y+Z} \\ z &= \frac{Z}{X+Y+Z}. \end{aligned} \tag{7.8}$$

O objetivo da proposta deste modelo é tornar os cálculos fotométricos e colorimétricos mais simples. Entre os requisitos para simplificação dos cálculos, temos

1. os valores numéricos deve ser não-negativos;
2. as grandezas fotométricas devem ser obtidas diretamente a partir das coordenadas;
3. o **ponto acromático** (quando os fotoreceptores são igualmente estimulados) deve ser tal que todas as três cores primárias tenham a mesma participação;

4. a combinação convexa das proporções das três cores primárias deve resultar em um maior número possível de cores fisicamente realizáveis;

5. uma das coordenadas deve ter valores nulos para cores com comprimento de onda maior que 650nm.

Vamos tentar "traduzir" os requisitos listados em grandezas geométricas. Para satisfazer a condição (1) o triângulo em vermelho na Figura 7.9.(c) deve ser transformado para o primeiro quadrante do novo referencial. Para satisfazer a condição (2) definiu-se a função de eficiência luminosa fotópica, mostrada na Figura 7.7, como a função de casamento $\overline{y}(\lambda)$, já que esta função descreve diretamente a relação entre luminância, uma grandeza fotométrica, e o comprimento de onda. Isso implica também que a luminância da mistura das cores primárias X e Z deve ser sempre nula. Para a terceira condição, podemos alterar as proporção das luzes primárias, de tal forma que o ponto acromático tenha as coordenadas de cromaticidade $E = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. Uma forma de satisfazer a condição (4) é fazer com que o envoltório triangular seja mais justo possível. Finalmente, para a última condição, o lado do triângulo deve tangenciar a borda da ferradura na região de "cores vermelhas". Por definição, este ponto representa a cor primária $Z = C_b$. Associamos o ponto C_g e C_r às cores primárias Y e X, respectivamente. Observamos que estes pontos estão fora da região de ferradura. Portanto, elas não são fisicamente realizáveis. Como se pode obter as funções de casamento $\overline{x}(\lambda)$, $\overline{y}(\lambda)$ e $\overline{z}(\lambda)$?

A solução seria via manipulações algébricas, reduzindo o problema em mudança de base do espaço CIE-RGB para o espaço CIE-XYZ. Definidas as três cores primárias normalizadas do novo espaço $c_r = (r_x, g_x, b_x = 1 - r_x - g_x) = (1.2750, -0.2779, 0.0029)$, $c_g = (r_y, g_y, b_y = 1 - r_y - g_y) = (-1.7395, 2.7675, -0.0280)$ e $c_b = (r_z, g_z, b_z = 1 - r_z - g_z) = (-0.7431, 0.1409, 1.6022)$, podemos representar as três cores primárias R, G e B, destacadas na Figura 7.9.(c), pelas coordenadas baricêntricas (x, y) com respeito a estas três novas cores primárias (Seção 3.1.2). Aplicando a Eq. 7.7, obtemos a terceira coordenada z pela expressão $z = 1 - x - y$. Resumindo, temos as seguintes correspondências

	(r,g,b)	(x,y,z)
R	(1,0,0)	$(0.73467, 0.26533, 0.0)$
G	(0,1,0)	$(0.27376, 0.71741, 0.00883)$
B	(0,0,1)	$(0.16658, 0.00886, 0.82456)$

Isso significa que para cada unidade de R, são necessários os estímulos $X_r = x_r C_r = 0.73467 C_r$, $Y_r = y_r C_r = 0.26533 C_r$ e $Z_r = z_r C_r = 0.0 C_r$, onde $C_r = X_r + Y_r + Z_r$, no espaço XYZ. Similarmente, são necessários $X_g = 0.27376 C_g$, $Y_g = 0.71741 C_g$ e $Z_g = 0.00883 C_g$ para uma unidade de G

e $X_b = 0.16658C_b$, $Y_b = 0.00886C_b$ e $Z_b = 0.82456C_b$ para uma unidade de B. Com isso, o problema de redefinir um novo espaço CIE-XYZ se reduz a achar uma transformação linear T do espaço CIE-RGB para CIE-XYZ tal que

$$\begin{bmatrix} x_rC_r & x_gC_g & x_bC_b \\ y_rC_r & y_gC_g & y_bC_b \\ z_rC_r & z_gC_g & z_bC_b \end{bmatrix} = T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

A transformação é, portanto,

$$\begin{aligned} T &= \begin{bmatrix} 0.73467C_r & 0.27376C_g & 0.16658C_b \\ 0.26533C_r & 0.71741C_g & 0.00886C_b \\ 0.0C_r & 0.00883C_g & 0.82456C_b \end{bmatrix} \\ &= \begin{bmatrix} 0.73467 & 0.27376 & 0.16658 \\ 0.26533 & 0.71741 & 0.00886 \\ 0.0 & 0.00883 & 0.82456 \end{bmatrix} \begin{bmatrix} C_r & 0 & 0 \\ 0 & C_g & 0 \\ 0 & 0 & C_b \end{bmatrix}. \end{aligned} \quad (7.9)$$

Agora falta determinar os valores de C_r, C_g e C_b. Para isso, podemos utilizar a condição (3) de que o ponto acromático tanto no espaço CIE-XYZ quanto no espaço CIE-RGB é $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, ou seja,

$$\begin{aligned} \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix} &= \begin{bmatrix} 0.73467 & 0.27376 & 0.16658 \\ 0.26533 & 0.71741 & 0.00886 \\ 0.0 & 0.00883 & 0.82456 \end{bmatrix} \begin{bmatrix} C_r & 0 & 0 \\ 0 & C_g & 0 \\ 0 & 0 & C_b \end{bmatrix} \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix} \\ &= \begin{bmatrix} 0.73467 & 0.27376 & 0.16658 \\ 0.26533 & 0.71741 & 0.00886 \\ 0.0 & 0.00883 & 0.82456 \end{bmatrix} \begin{bmatrix} \frac{C_r}{3} \\ \frac{C_g}{3} \\ \frac{C_b}{3} \end{bmatrix}. \end{aligned}$$

Multiplicando pela inversa da matriz, obtemos

$$\begin{bmatrix} C_r \\ C_g \\ C_b \end{bmatrix} = \begin{bmatrix} 0.73467 & 0.27376 & 0.16658 \\ 0.26533 & 0.71741 & 0.00886 \\ 0.0 & 0.00883 & 0.82456 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.66695 \\ 1.13241 \\ 1.20064 \end{bmatrix}$$

Substituindo estes valores na Eq. 7.9, chegamos em seguinte expressão de transformação

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.48999 & 0.31001 & 0.20000 \\ 0.17696 & 0.81240 & 0.01064 \\ 0.00000 & 0.01000 & 0.99000 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Aplicando esta transformação sobre todas as cores espectrais e normalizando-as, obtemos um diagrama de cromaticidade do padrão CIE-XYZ mostrado na Figura 7.10.(b).

A matriz inversa, que leva um ponto do espaço CIE-XYZ para o espaço CIE-RGB é dada por

$$\begin{aligned}\begin{bmatrix} R \\ G \\ B \end{bmatrix} &= \begin{bmatrix} 0.48999 & 0.31001 & 0.20000 \\ 0.17696 & 0.81240 & 0.01064 \\ 0.00000 & 0.01000 & 0.99000 \end{bmatrix}^{-1} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \\ &= \begin{bmatrix} 2.36466 & -0.89659 & -0.46807 \\ -0.51515 & 1.42641 & 0.08874 \\ 0.00520 & -0.01441 & 1.00920 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}\end{aligned}$$

Assim, pode-se obter, de forma indireta, as funções de casamento $\overline{x}(\lambda)$ e $\overline{y}(\lambda)$, que são esboçadas na Figura 7.10.(c).

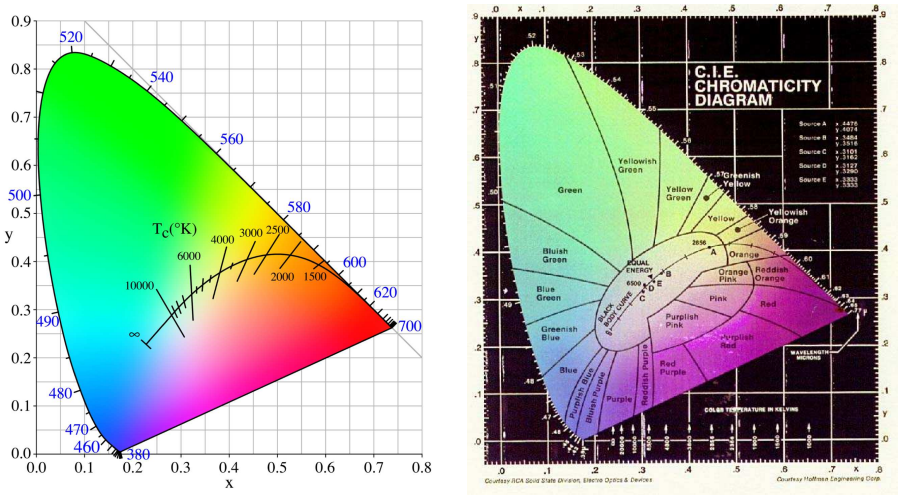
Observamos que a matriz de transformação padronizada pela CIEé ligeiramente diferente do que derivamos. A matriz, com a quantidade correta de dígitos significativos, é

$$\begin{aligned}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} &= \frac{1}{b_{21}} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \\ &= \frac{1}{0.17697} \begin{bmatrix} 0.489989 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.0 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (7.10)\end{aligned}$$

7.4 Diagrama de Cromaticidade

Vimos na seção 7.3.1 como se deriva um diagrama de cromaticidade. Nesta seção vamos explorar um pouco mais os dados que estão contidos nele.

O diagrama de cromaticidade CIE-XYZ apresenta todas as possíveis cores visíveis normalizadas em um específico nível de luminância. Ele representa, de fato, o **gamute da visão humana**. A parte curva da borda do gamute é o lugar geométrico de todas as cores espectrais ou cores monocromáticas. As "cores" que um corpo negro pode emitir a uma temperatura de em torno $1000^o\ K$ até infinito definem uma curva que é conhecida como **curva de corpo negro** (Figura 7.11.(a)). Os "brancos", ou **iluminantes**, que percebemos se encontram em torno desta curva. Alguns destes são fisicamente reproduzíveis pela tecnologia existente ou pela natureza. Na Figura 7.11.(b) são mostrados o ponto acromático $E = (\frac{1}{3}, \frac{1}{3})$, para o qual a distribuição espectral de potência é constante, e os iluminantes A=$(0.448, 0.408)$, B=$(0.349, 0.352)$, C=$(0.310, 0.316)$, D_{5500}=$(0.332, 0.348)$, D_{6500}=$(0.313, 0.329)$ e D_{7500}=$(0.299, 0.315)$. O iluminante A se aproxima



(a) (b)

Figura 7.11: Diagrama de cromaticidade: (a) curva de corpo negro (Fonte: http://upload.wikimedia.org/wikipedia/commons/b/ba/PlanckianLocus.png) e (b) alguns iluminantes (Fonte: http://www.imagingscience.com/cie1931_illum.htm).

da cor de uma lâmpada de tungstênio a $2856^o\ K$. Ele é um "branco avermelhado". O iluminante B se aproxima da iluminação solar ao meio-dia, enquanto o iluminante C, adotado como "branco" pelo Comitê Nacional do(s) Sistema(s) de Televisão (NTSC – *National Television Standards Committee*), é a iluminação de um céu coberto ao meio-dia. Os iluminantes D correspondem às radiações emitidas pelo corpo negro em torno de $5500^o\ K$, $5500^o\ K$ e $7500^o\ K$, respectivamente. O iluminante D_{6500} é adotado como "branco" pela Sociedade de Engenheiros de Filmes e Televisão dos EUA (SMPTE – *Society of Motion Picture and Television Engineers*), pela União de Transmissão Européia (EBU – *European Broadcasting Union*) e pelo sistema de transmissão televisiva Linha de Fase Alternante (PAL – *Phase Alternating Line*).

O diagrama de cromaticidade pode ser aplicado na determinação da pureza de uma cor, na identificação de cores complementares, na previsão da mistura de cores ou do gamute de cor de um sistema e na comparação de gamutes de cor.

7.4.1 Gamute de Cor

Seja uma cor C especificada pela sua cromaticidade e pela luminância, isto é, $C = (x, y, Y)$. Pelas Leis de Grassmann, a mistura de duas cores $C_1 = (x_1, y_1, Y_1)$ e $C_2 = (x_2, y_2, Y_2)$ pode ser obtida algebricamente como a soma das suas componentes

$$C_{12} = C_1 + C_2 = (X_1 + X_2) + (Y_1 + Y_2) + (Z_1 + Z_2).$$

Como se determina estas componentes? Tendo x_i e y_i, $i = 1, 2$, podemos obter z_i pela expressão $z = 1 - x - y$. Em seguida, determinamos $C_i = X_i + Y_i + Z_i$ utilizando o fato de $C_i = \frac{Y_i}{y_i}$. Substituindo x_i, z_i e C_i em Eq. 7.7, determinamos X_i e Z_i. Agora só resta determinar as coordenadas de cromaticidade da mistura

$$\begin{aligned} x_{12} &= \frac{(X_1 + X_2)}{C_{12}} \\ y_{12} &= \frac{(Y_1 + Y_2)}{C_{12}} \\ Y_{12} &= (Y_1 + Y_2). \end{aligned}$$

para representá-la em notação convencional $C_{12} = (x_{12}, y_{12}, Y_{12})$.

Quando se tem mais de duas cores, podemos aplicar sucessivamente o procedimento. Por exemplo, a mistura de duas cores $C_1 = (0.1, 0.3, 10)$ e $C_2 = (0.35, 0.2, 10)$ é $C_{12} = (0.25, 0.24, 20)$. Observe na Figura 7.12.(a) que o ponto da mistura C_{12} fica sobre o segmento que une as duas cores. Isso é esperado, porque algebricamente estamos fazendo uma combinação linear das duas cores (segmento cheio, em vermelho, na Figura 7.12.(a))

$$C_{12} = \alpha C_1 + (1-\alpha)C_2 = (\alpha X_1 + (1-\alpha)X_2) + (\alpha Y_1 + (1-\alpha)Y_2) + (\alpha Z_1 + (1-\alpha)Z_2). \tag{7.11}$$

e projetando as suas coordenadas sobre o plano de Maxwell (segmento pontilhado, em vermelho, na Figura 7.12.(a))

$$\begin{aligned} x_{12} &= \frac{(\alpha X_1 + (1-\alpha)X_2)}{C_{12}} \\ y_{12} &= \frac{(\alpha Y_1 + (1-\alpha)Y_2)}{C_{12}}. \end{aligned}$$

Se alterarmos apropriadamente os valores de Y (luminância) das duas cores, podemos obter todas as cores sobre o segmento. Por exemplo, a mistura de $C_1 = (0.1, 0.3, 10)$ e $C_2 = (0.35, 0.2, 0)$ resulta em $C_1 = (0.1, 0.3, 10)$.

Se adicionarmos mais uma cor, $C_3 = (0.2, 0.05, 10)$, teremos uma mistura $C_{123} = (0.215, 0.106, 30)$. Como esperado, C_{123} está sobre o segmento que une C_{12} e C_3. O lugar geométrico de todas as possíveis misturas entre estas três cores é o fecho convexo definido pelos pontos (X_1, Y_1, Z_1), (X_2, Y_2, Z_2) e (X_3, Y_3, Z_3) no espaço CIE-XYZ (Seção 3.1.2). A projeção deste fecho no plano Maxwell é representado pelo triângulo em linha pontilhada na Figura 7.12.(b).

Onde podemos aplicar isso? Na determinação do gamute de um dispositivo gráfico. Por exemplo, vimos na seção 2.1.1 que para televisões/monitores coloridos de tecnologia CRT há três tipos de fósforo, R, G e B, em cada *pixel*, de modo que cada fósforo, ao ser excitado, emite uma das cores primárias. Para identificarmos o gamute de cor que um específico monitor consegue reproduzir, é só identificarmos no diagrama de cromaticidade os pontos correspondentes às coordenadas de cromaticidade dos três fósforos. O triângulo definido por estes pontos cobre os possíveis cromas que eles conseguiriam reproduzir. A seguinte tabela exemplifica as especificações dos gamutes de alguns sistemas:

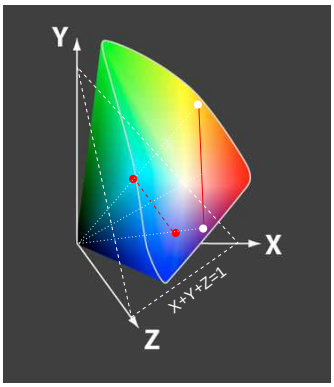
Sistema	R		G		B		"Branco"
	x	y	x	y	x	y	
NTSC	0.67	0.33	0.21	0.71	0.14	0.08	Iluminante C
EBU	0.64	0.33	0.30	0.60	0.15	0.06	D_{6500}
PAL	0.64	0.33	0.29	0.60	0.15	0.06	D_{6500}
Dell	0.625	0.340	0.275	0.605	0.150	0.065	9300K
SMPTE	0.630	0.340	0.310	0.595	0.155	0.070	D_{6500}

A partir destes dados técnicos, conseguimos identificar os gamutes dos quatro sistemas, conforme mostra a Figura 7.12.(c). Observamos ainda que, com os gamutes esboçados no diagrama de cromaticidade, podemos verificar a fidelidade da reprodução de uma cor exibida em um monitor pelo outro. Quanto maior for a área de sobreposição entre os gamutes, maior será a probabilidade dos dois sistemas serem "colorimetricamente" equivalentes.

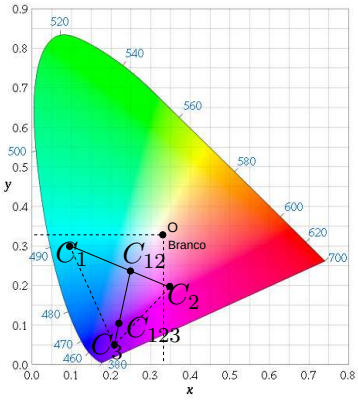
7.4.2 Pureza de Cor

Uma cor espectral é uma cor 100% pura, porque não contém componentes de outros comprimentos de onda, enquanto o "branco" de um sistema é considerado uma cor totalmente diluída, de 0% pureza. Podemos determinar a pureza de uma cor com auxílio do diagrama de cromaticidade.

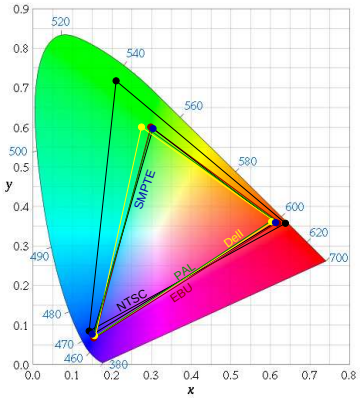
Dada uma cor $C_1 = (x_1, y_1, Y_1)$ qualquer, determinamos o seu **comprimento de onda dominante** traçando uma semi-reta que sai do "branco" $O = (x_w, y_w, Y_w)$ do sistema em direção de C_1. O ponto de intersecção $C_2 =$



(a)



(b)



(c)

Figura 7.12: Combinação de cores: (a) projeção; (b) mistura aditiva e (c) gamutes de cor.

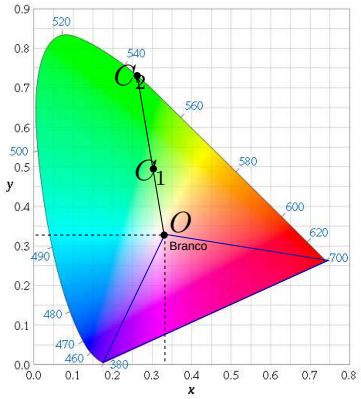


Figura 7.13: Pureza de cores.

(x_2, Y_2, Y_2) desta semi-reta com a "curva espectral" é o lugar geométrico da cor espectral cujo comprimento de onda é o comprimento de onda dominante de C_1. Observe que as cores contidas na região delimitada pela linha azul da Figura 7.13 não tem um comprimento de onda dominante, porque a semi-reta intercepta a linha púrpura. Neste caso, consideramos a intersecção da semi-reta com a linha púrpura.

Para obter a pureza da cor C_1, basta determinarmos a razão $\frac{OC_1}{OC_2}$, considerando todas as três cores O, C_1 e C_2 no mesmo nível de luminância.

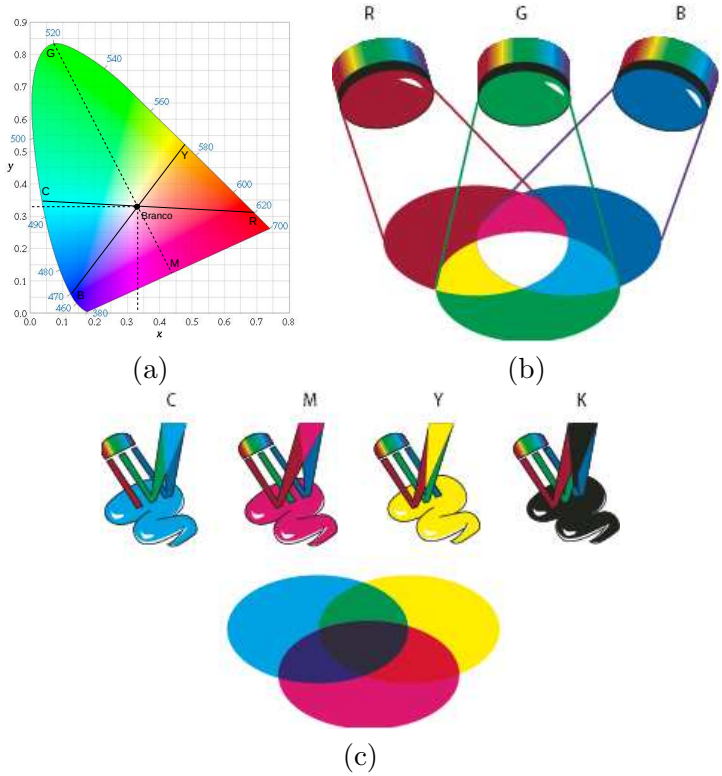
7.4.3 Cores Complementares

Por definição, duas cores C_1 e C_2 são complementares, se quando elas são misturadas em proporções apropriadas resulta em "branco" C_w do sistema (Seção 7.4.1).

$$C_w = \alpha C_1 + \gamma C_2 = \alpha(X_1 + Y_1 + Z_1) + \gamma(X_2 + Y_2 + Z_2)$$

A cor complementar de uma cor espectral C_1 pode ser obtida geometricamente, estendendo a reta que passa pelo "branco" do sistema e por C_1 até interceptar "outro lado da borda" do diagrama de cromaticidade. Na Figura 7.14.(a) os seguintes pares de cores são complementares: "vermelho" (R) e "ciano" (C); "azul" (B) e "amarelo" (Y); "verde" (G) e "magenta" (M).

O que significa fisicamente as cores complementares? Vimos que as cores R, G e B, quando combinadas em proporções adequadas, formam a cor "branco" (Figura 7.14.(b)). Uma cor que "complementa" a cor "vermelho"



(a) (b) (c)

Figura 7.14: Cores complementares: (a) solução gráfica; (b) sistema aditivo e (c) sistema subtrativo (Fonte: http://help.adobe.com/en_US/Photoshop/11.0/WSECCD001A-4989-4451-A752-63D5EF9D5619a.html).

para que o resultado fique "branco" seria uma cor que sensibilize os cones S e M (mistura G+B que é a cor "ciano"). Analogamente, "magenta" é uma mistura B+R que sensibiliza os cones S e L e a cor "amarelo" é uma mistura G+R. Em outras palavras, podemos dizer que "ciano" subtrai do "branco" a cor "vermelho", enquanto "magenta" e "amarelo" subtraem "verde" e "azul", respectivamente. Se sobrepormos um filtro "magenta" sobre um filtro "ciano", perceberemos "azul". Colocando um filtro "amarelo" sobre os dois, todas as radiações serão subtraídas. Consequentemente, perceberemos a cor "preto". Esta idéia é ilustrada na Figura 7.14.(c). O sistema de formação de cores por remoção de radiações de específicos comprimentos de onda é conhecido como **sistema subtrativo** de cor.

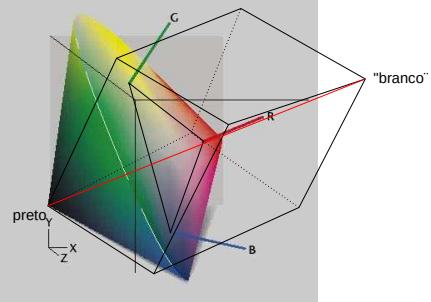


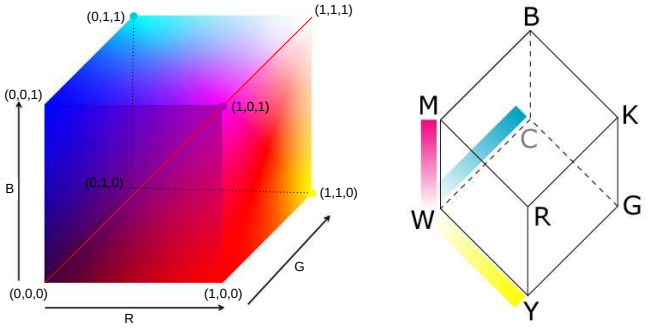
Figura 7.15: Gamute e espaço sólido de cor de um dispositivo.

7.5 Modelos de Cor em Sistemas de Informação Gráfica

Um gamute de cor apresentado no diagrama de cromaticidade não "carrega" a informação de luminância. Através dele não podemos identificar todas as possíveis cores geradas pelo dispositivo. Por conveniência, os aplicativos gráficos tem adotado modelos de cor 3D que incluem o sub-espaço de croma (gamute) e o sub-espaço de luminância, variando da luminância nula ("preto") até luminância máxima ("branco"), como o paralelogramo apresentado na Figura 7.15. Há modelos orientados à tecnologia de dispositivos de saída (Seção 2.1.1) e outros mais próximos da nossa concepção de cor.

Exemplos comuns de modelos orientados à tecnologia de dispositivos são: RGB, CMY e YIQ. O modelo RGB é utilizado para especificar cores de monitores. Isso se deve ao fato da formação aditiva das cores a partir de três tipos de fósforos. Aplica-se o modelo CMY para especificar processo de formação subtrativa de cores, como ocorre em impressão, tingimento e pinturas. Tintas, pigmentos ou filmes são materiais que refletem um subconjunto de comprimentos de onda da luz, produzindo uma específica percepção de cor diferente da cor da luz incidente. Na seção 7.4.3 vimos que três cores primárias "ciano", "magenta" e "amarelo" são suficientes para "subtrair totalmente" uma luz branca. O modelo YIQ é adotado para sistema de televisão colorida NTSC.

Outros modelos, que podem ser obtidos do modelo RGB por transformações lineares, como HSL, HSV e HSI, são mais apropriados para definir uma cor por parte do usuário, envolvendo os conceitos de tonalidade/matiz, saturação e o brilho. Alguns principais algoritmos de conversão de modelos de cores pode ser encontrados em `http://www.cs.rit.edu/~ncs/color/t_`



(a) (b)

Figura 7.16: Cubo de cor: (a) RGB e (b) CMY.

`convert.html`. Em `http://www.csit.fsu.edu/~burkardt/f_src/colors/colors.html` encontra-se uma lista de algoritmos de conversão implementados em C.

7.5.1 RGB

O modelo RGB especifica o espaço de cor dos monitores CRT e outros dispositivos gráficos *raster*. O gamute coberto por um monitor CRT depende da cromaticidade dos fósforos que ficam na camada interna da tela (Seção 2.1.1). Portanto, há uma grande variação de gamutes entre os monitores, como ilustramos na Figura 7.12.(b). O modelo RGB "uniformiza" esta diversidade de espaços através da definição de um cubo unitário em um sistema de referência cartesiano. Os vetores-base $(1,0,0)$, $(0,1,0)$ e $(0,0,1)$ representam, respectivamente, as cores primárias vermelha, verde e azul, em intensidade máxima, cujo croma é, respectivamente, igual a (x_r, y_r), (x_g, y_g) e (x_b, y_b). As coordenadas $(0,0,0)$ e $(1,1,1)$ representam, respectivamente, as cores "preto" e "branco" do sistema. Diferentes tons de cinza estão sobre o segmento em vermelho desenhado na Figura 7.16.(a). Pela Eq. 7.7, temos a seguinte correspondência entre esses pontos

RGB	CIE-XYZ
(0,0,0)	$(0,0,0)$
(1,0,0)	$(X_r, Y_r, Z_r) = (x_r C_r, y_r C_r, z_r C_r)$, onde $C_r = X_r + Y_r + Z_r$
(0,1,0)	$(X_g, Y_g, Z_g) = (x_g C_g, y_g C_g, z_g C_g)$, onde $C_g = X_g + Y_g + Z_g$
(0,0,1)	$(X_b, Y_b, Z_b) = (x_b C_b, y_b C_b, z_b C_b)$, onde $C_b = X_b + Y_b + Z_b$

Embora "uniformizados" em mesmo cubo unitário, os vetores-base que descrevem cada cubo podem corresponder a diferentes vetores no espaço

CIE-XYZ. Portanto, um ponto representado pelas coordenadas (R, G, B) em dois cubos distintos podem corresponder a duas cores distintas no espaço CIE-XYZ. A pergunta que naturalmente surge é como um processador pode mapear fielmente os pontos entre dois cubos unitários para que as imagens exibidas em dois dispositivos distintos, $\mathcal{D}_1$ e $\mathcal{D}_2$, sejam mais parecidas possíveis. Para isso, pode-se utilizar o padrão CIE-XYZ como referência, transformando a cor do gamute de $\mathcal{D}_1$ definido pelas cores primárias $(x_{r,1}, y_{r,1})$, $(x_{g,1}, y_{g,1})$ e $(x_{b,1}, y_{b,1})$ para XYZ através da transformação M_1

$$\begin{aligned}
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} &= M_1 \begin{bmatrix} R_1 \\ G_1 \\ B_1 \end{bmatrix} \\
&= \begin{bmatrix} X_{r,1} & X_{g,1} & X_{b,1} \\ Y_{r,1} & Y_{g,1} & Y_{b,1} \\ Z_{r,1} & Z_{g,1} & Z_{b,1} \end{bmatrix} \begin{bmatrix} R_1 \\ G_1 \\ B_1 \end{bmatrix} \\
&= \begin{bmatrix} x_{r,1}C_{r,1} & x_{g,1}C_{g,1} & x_{b,1}C_{b,1} \\ y_{r,1}C_{r,1} & y_{g,1}C_{g,1} & y_{b,1}C_{b,1} \\ z_{r,1}C_{r,1} & z_{g,1}C_{g,1} & z_{b,1}C_{b,1} \end{bmatrix} \begin{bmatrix} R_1 \\ G_1 \\ B_1 \end{bmatrix}
\end{aligned} \tag{7.12}$$

e depois da referência XYZ para o gamute do dispositivo $\mathcal{D}_2$ definido pelas cores primárias $(x_{r,2}, y_{r,2})$, $(x_{g,2}, y_{g,2})$ e $(x_{b,2}, y_{b,2})$ através de M_2

$$\begin{bmatrix} R_2 \\ G_2 \\ B_2 \end{bmatrix} = M_2^{-1} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x_{r,2}C_{r,2} & x_{g,2}C_{g,2} & x_{b,2}C_{b,2} \\ y_{r,2}C_{r,2} & y_{g,2}C_{g,2} & y_{b,2}C_{b,2} \\ z_{r,2}C_{r,2} & z_{g,2}C_{g,2} & z_{b,2}C_{b,2} \end{bmatrix}^{-1} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}.$$

As duas matrizes de transformação requerem o valor do fluxo luminoso máximo $C_{i,j}$ que cada fósforo i consegue emitir em cada dispositivo j. Como podemos obter estes valores? Há duas formas para determiná-los.

Uma forma é medir com um fotômetro o nível de luminância máxima $Y_{i,j}$, $Y_{i,j}$ e $Y_{i,j}$ de cada cor primária i do dispositivo j e substituí-los em

$$C_{i,j} = \frac{Y_{i,j}}{y_{i,j}}.$$

A segunda forma é determinar as coordenadas $(X_{w,j}, Y_{w,j}, Z_{w,j})$ da cor "branco" $(1, 1, 1)$ do dispositivo j e utilizar estas coordenadas para obter $C_{i,j}$, $C_{i,j}$ e $C_{i,j}$ através da expressão

$$\begin{bmatrix} C_{r,j} \\ C_{g,j} \\ C_{b,j} \end{bmatrix} = \begin{bmatrix} x_{r,j} & x_{g,j} & x_{b,j} \\ y_{r,j} & y_{g,j} & y_{b,j} \\ z_{r,j} & z_{g,j} & z_{b,j} \end{bmatrix}^{-1} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \tag{7.13}$$

Vamos considerar um monitor com branco $(0.310063, 0.316158, 10.0)$ e três cores primárias cujas coordenadas de cromaticidade são:

	R	G	B
x	0.67	0.21	0.14
y	0.33	0.71	0.08

Para obter as coordenadas RGB de uma cor $(x_C, y_C, Y_C) = (0.25, 0.2, 1.0)$ no espaço de cores do monitor, ou seja, determinar a "percentagem" da energia máxima de cada cor primária, C_R, C_G e C_B, devemos calcular em primeiro lugar C_R, C_G e C_B. Isso pode ser feito com uso da Eq. 7.13

$$\begin{bmatrix} C_R \\ C_G \\ C_B \end{bmatrix} = \begin{bmatrix} 0.67 & 0.21 & 0.14 \\ 0.33 & 0.71 & 0.08 \\ 0 & 0.08 & 0.78 \end{bmatrix}^{-1} \begin{bmatrix} 9.8072 \\ 10.0 \\ 11.8225 \end{bmatrix} = \begin{bmatrix} 9.061 \\ 8.265 \\ 14.304 \end{bmatrix}.$$

Em seguida, devemos escrever a cor (x_C, y_C, Y_C) na sua forma não normalizada, isto é, $(X_C, Y_C, Z_C) = (\frac{x_c}{y_c} Y_C = 1.25, 1.0, \frac{1-x_c-y_c}{y_c} Y_C) = 2.75$ e calcular a percentagem de contribuição de cada cor primária para obtê-la

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = \begin{bmatrix} 1.25 \\ 1.0 \\ 2.75 \end{bmatrix} = \begin{bmatrix} 0.67 & 0.21 & 0.14 \\ 0.33 & 0.71 & 0.08 \\ 0 & 0.08 & 0.78 \end{bmatrix} \begin{bmatrix} 9.061 & 0 & 0 \\ 0 & 8.265 & 0 \\ 0 & 0 & 14.304 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

Substituindo os valores, obtemos a representação da cor (X_C, Y_C, Z_C) no espaço de cores do monitor: $(0.106, 0.069, 0.243)$.

7.5.2 CMY

É adequado para especificar as cores das impressoras coloridas que utilizam o sistema subtrativo para formação de cores. Nestes dispositivos, os pigmentos ou corantes são depositados sobre a folha durante a "impressão". Tais materiais funcionam como filtro, deixando somente que ondas de certos comprimentos sejam refletidas (Seção 2.1.1). A cor final percebida depende, portanto, muito da "cor original" da qual tais ondas foram subtraídas. Por exemplo, cor "ciano" sobre papel vermelho resulta em percepção de cor "preto" ou cor "ciano" sobre uma folha amarela aparece como "verde". Usualmente, considera-se como o referencial o "branco" (X_w, Y_w, Z_w).

De forma análoga ao modelo RGB, o modelo CMY "uniformiza" a representação dos espaços de cor de sistemas subtrativos por um cubo unitário definido em um referencial cartesiano. Para uma cor em coordenadas (R, G, B),

a sua representação (C, M, Y) no sistema CMY deve ser tal que

$$\begin{bmatrix} x_r C_r & x_g C_g & x_b C_b \\ y_r C_r & y_g C_g & y_b C_b \\ z_r C_r & z_g C_g & z_b C_b \end{bmatrix} \begin{bmatrix} R+C \\ G+M \\ B+Y \end{bmatrix} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}$$
$$= \begin{bmatrix} x_r C_r & x_g C_g & x_b C_b \\ y_r C_r & y_g C_g & y_b C_b \\ z_r C_r & z_g C_g & z_b C_b \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$

ou seja,

$$\begin{bmatrix} R+C \\ G+M \\ B+Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Em outras palavras, para um mesmo subespaço 3D de cor, as coordenadas do modelo RGB e as do modelo CMY satisfazem a seguinte relação (Figura 7.16.(b))

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

Vimos na seção 7.4.3 que, teoricamente, a partir das três cores primárias ciano (C), amarelo (Y) e magenta (M) é possível subtrair todos os comprimentos de onda. No entanto, a cor resultante da subtração de uma cor "branco" pelas três cores primárias tende a aparentar mais "marrom escuro" do que "preto", sem falar da quantidade de tinta necessária para chegar nesta cor. Com uma solução alternativa de baixo custo, é comum incluir uma quarta cor ao modelo: cor "preto". Daí, o popular modelo de cor CMYK (*cyan, magenta, yellow, and key (black)*).

7.5.3 NTSC YIQ

É utilizado na transmissão de sinais de televisão colorida NTSC nos Estados Unidos e no Japão. A coordenada Y é igual à coordenada Y do padrão CIE-XYZ, de forma que uma televisão preto-e-branco possa utilizá-la para exibir imagens coloridas em níveis de cinza. Esta coordenada indica, portanto, a luminância enquanto as coordenadas I e Q codificam a sua crominância: I, *In-phase signal* em inglês, contém informação da faixa laranja–ciano e Q, (*Quadrature signal*) em inglês, representa cromas do intervalo magenta–verde. Como a sensibilidade da visão humana é maior para a luminância e menor para as variações entre magenta e verde, esta decomposição contribui

adicionalmente para maximizar a transmissão de informação "relevante" numa largura de faixa menor que 4.2MHz.

Considerando uma cor especificada no modelo RGB do sistema NTSC (Seção 7.4.1) com o iluminante C ($(x_w, y_w, Y_w) = (0.31, 0.316, 100.0)$) como o ponto branco, ela pode ser mapeada para uma cor do modelo YIQ através da seguinte transformação

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ 0.595716 & -0.274453 & -0.321263 \\ 0.211456 & -0.522591 & 0.311135 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

7.5.4 HSV

Diferentemente dos modelos RGB, CMY e YIQ, o modelo HSV é orientado aos usuários, cuja especificação se baseia na forma como estamos habituados a definir uma cor – matiz (H), saturação (S) e "valor de brilho" (V). O modelo HSV é derivado do modelo RGB, olhando-o pela diagonal do cubo unitário, como ilustra Figura 7.17.(a). A coordenada H indica o ângulo, em grau, da cor em relação à cor "vermelho"; a coordenada S corresponde à pureza da cor: quando a cor é "pura", $S = 1$. E a coordenada V está relacionada com a claridade da cor: quando mais "ofuscante" for a cor, maior será o valor de V. O pseudo-código RGB_HSV ilustra um procedimento empírico de mapeamento de uma cor entre os dois espaços. Figura 7.17.(b) ilustra um cone HSV.

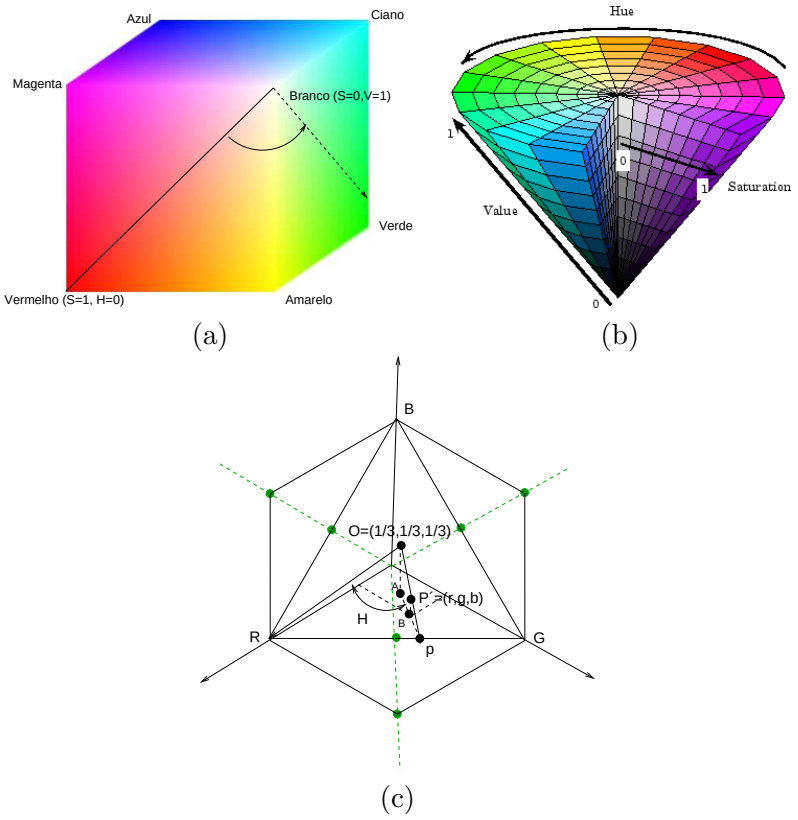


Figura 7.17: Cone HSV: (a) vista diagonal do cubo RGB; (b) sólido de cor; e (c) relação entre RGB e HSV.

Input: R,G,B
Output: H,S,V
max = Máximo (R,G,B) ;
min = Mínimo (R,G,B) ;
V = max ;
if $max \neq 0$ **then**
  S = $\frac{max-min}{max}$;
else
  S = 0;
end
if $S = 0$ **then**
  H = indefinido ;
  Retorne ;
else
  $delta = max$ - min ;
  if $R = max$ **then**
    H = $\frac{G-B}{delta}$;
  else if $G = max$ **then**
    H = 2 + $\frac{B-R}{delta}$;
  else
    H = 4 + $\frac{G-G}{delta}$;
  end
  H = H * 60 ;
  /* Converter para graus */ **if** H ¡ 0 **then**
    H = H + 360 ;

end

Algoritmo 4: RGB_HSV: Procedimento de conversão do modelo RGB para HSV.

Um procedimento analítico, mais preciso, consiste em considerar que o valor/a intensidade da cor $P = (R, G, B)$ seja

$$V \triangle \frac{1}{3}(R + G + B).$$

Projetamos P e o ponto "branco" sobre o plano $R + G + B = 1$ para achar o ângulo entre a reta projetada $\overline{OP'}$ e a reta de referência $\overline{OR}$. A projeção do ponto P é $P' = (r, g, b)$

$$r \quad \triangle \quad \frac{R}{R+G+B} = \frac{R}{3I}$$

$$g \triangleq \frac{G}{R+G+B} = \frac{G}{3I}$$
$$b \triangleq \frac{B}{R+G+B} = \frac{B}{3I}$$

e a projeção do ponto "branco" é $O = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$.

O produto interno entre os vetores normalizados $\Longrightarrow OR$ e $\Longrightarrow OP'$ nos fornece o cosseno do ângulo H entre os dois vetores

$$\begin{aligned} cos(H) &= \frac{\overrightarrow{OR} \cdot \overrightarrow{OP'}}{|\overrightarrow{OR}||\overrightarrow{OP'}|} \\ &= \frac{2R - G - B}{\sqrt{\frac{2}{3}}\sqrt{6(R^2+G^2+B^2-RG-GB-BR)}} \\ &= \frac{3R - (R+G+B)}{2\sqrt{R^2+G^2+B^2-RG-GB-BR}} \\ &= \frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}} \end{aligned}$$

Observe que se B > G, precisamos subtrair H de 360^o para adequar à convenção do matiz variar no intervalo $[0, 360]$.

Finalmente, para determinar a saturação S, usamos o fato de que no ponto "branco" temos S=0, e nos pontos da borda do triângulo $R+G+B = 1$ temos S=1. Assim, nos outros pontos o valor de saturação é dado por

$$S = \frac{\overline{OP'}}{\overline{Op}} = 1 - \frac{\overline{P'p}}{\overline{Op}}.$$

Pela semelhança de triângulos, $\triangle OAp \sim \triangle P'Bp$, obtemos o valor de saturação em termos das coordenadas r, g e b

$$S = 1 - \frac{min(r,g,b)}{\frac{1}{3}}.$$

7.6 Representação Digital de Cor

Mostramos como é possível "ver" cores que percebemos via valores numéricos na seção 7.3. Adicionalmente, mostramos na seção 7.5 que os dispositivos de saída de distintas tecnologias conseguem reproduzir um sub-espaço destas cores e eles conseguem "identificar" cada cor deste sub-espaço por um

triplo de valores numéricos. Isso significa que se quisermos que um dispositivo reproduza uma determinada cor, é suficiente que "digamos" para ele a identificação numérica de cada cor. Caso tal dispositivo for controlado por um processador, basta que o processador transmita para ele tal identificação. No caso de monitores, estes valores são armazenados em *frame buffer* (Seção 2.2.1). A resolução de cor nesta memória depende da quantidade de planos de *bit*, ou profundidade como já introduzimos na seção 2.1.1, associada a ela. Vamos ver diferentes alternativas de organização desta memória para representação de cores de uma imagem.

Um dispositivo é monocromático, quando há somente um *bit* associado a cada plano (Figura 7.18.(a)). Ele recebe apenas dois tipos de sinais de controle: 0 ou 1. Este sinal digital é convertido, pelo conversor digital–analógico DAC, em sinal analógico de controle da luminância das "luzes primárias", como já comentamos na seção 2.2.1. Um *frame-buffer* com somente um plano de *bit* é conhecido também como *bitmap*. Uma memória com 8 planos de *bit* consegue distinguir 2^8=256 níveis de cinza ou 256 cores, ou seja, gerar 256 distintos sinais de controle (Figura 7.18.(b)). Podemos ainda, com auxílio de um *look-up table*, aumentar a capacidade de discriminação de cores de um monitor colorido. Ao invés de utilizarmos os planos de *bit* para representar cores, eles são utilizados para armazenar as entradas em *look-up table* com $M > 8$ *bits* por entrada, conforme ilustra Figura 7.18.(c). Neste caso, aumentou-se a resolução de cor para 2^M ¿ 2^8 com a restrição de que somente 2^8 distintas cores podem estar disponíveis simultaneamente em cada instante. Para acessar outras cores, é necessário recarregar a tabela. Por isso, é comum chamar este tipo de memória de **cor falsa** ou **pseudo-cor**, em inglês *pseudo color*. Hoje em dia, é comum utilizar 24 planos de *bit*, 8 *bits* para cada uma das cores primárias R, G e B, ou seja 2^8 sinais de controle para cada um dos três canhões de feixe de elétrons. Ao todo, consegue-se diferenciar 2^{24} cores – maior do que a visão de um observador médio conseguiria distinguir. Por este motivo, este tipo de memória é conhecido como do tipo **cor real**, em inglês *true color*. A sua organização é bem similar à ilustrada na Figura 2.11 com 15 planos de *bit*.

No caso de dispositivos de tecnologia baseada em raios catódicos, os sinais digitais armazenados em *frame-buffer* são convertidos em tensões V que controlam a intensidade luminosa I a ser emitida por cada fósforo. A função de transferência entre o sinal de vídeo e a intensidade de luz perceptível na tela do monitor é dada por uma expressão não-linear

$$I = V^{\gamma}. \tag{7.14}$$

Isso significa que um incremento na tensão não produza necessariamente

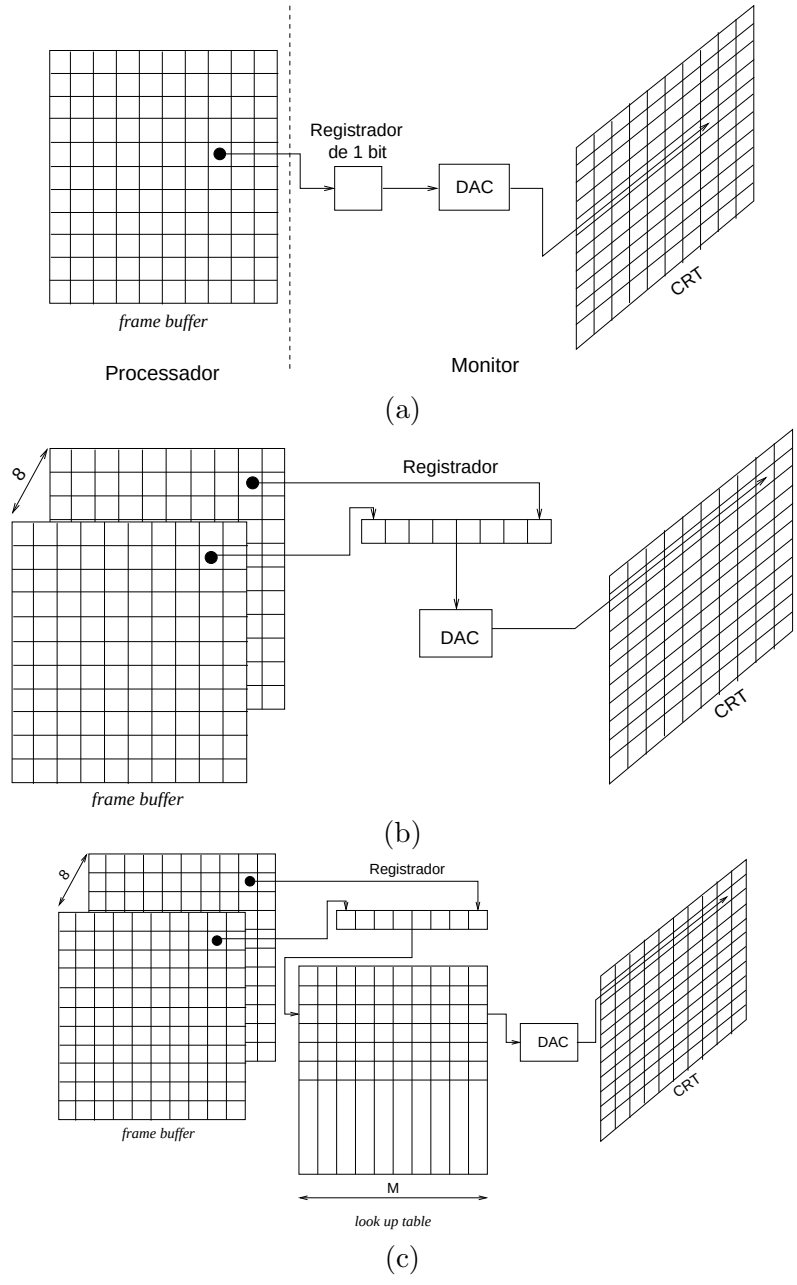


Figura 7.18: *Frame-buffer*: (a) *bitmap*; (b) 8 planos de *bit* monocromático; (c) pseudo-cor.

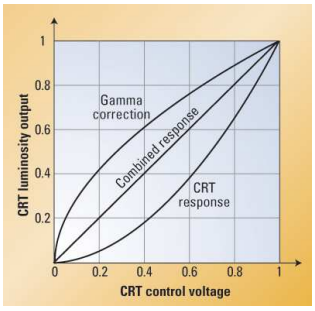


Figura 7.19: Correção gamma (Fonte: http://broadcastengineering.com/newsrooms/broadcasting_gamma_correction/).

uma variação em intensidade na mesma proporção, como ilustra (Figura 7.19). O expoente γ que aparece na Eq. 7.14 é uma constante que depende das características dos fósforos. Por exemplo, no sistema NTSC, o fator é 2.2 e no sistema PAL, 2.8. Para corrigir esta não-linearidade, é feita em muitos sistemas uma compensação através da "pré-correção" do sinal original V, aplicando um sinal $V' = V^{\frac{1}{\gamma}}$ em seu lugar

$$I = V'^{\gamma} = V^{\frac{1}{\gamma}\gamma} = V$$

Este ajuste é conhecido como **correção gamma**. Diferentemente dos monitores de tecnologia CRT, observamos que os monitores de tecnologia LCD ou plasma apresentam uma função de transferência linear. Isso tem gerado distorções de cores na exibição de sinais pré-compensados nestes dispositivos.

Para facilitar a especificação de uma cor por usuários de diferentes perfis, é comum encontrar na interface de aplicativos gráficos modelos alternativos ao modelo RGB. O modelo HSV é o mais popular. Procedimento de conversão do espaço HSV para o espaço RGB é automaticamente executado para que internamente a representação seja uma das quatro alternativas apresentadas nesta seção, em valores inteiros sem sinal.

Capítulo 8

Modelos de Iluminação

O objetivo deste capítulo é apresentar paradigmas através dos quais os computadores conseguem "ver" as radiações luminosas refletidas pelas superfícies, "sentir" cores próximas da nossa percepção e reproduzí-las em uma imagem. Após a leitura deste capítulo, você deve ser capaz de

- diferenciar um modelo de iluminação local de um modelo de iluminação global.
- caracterizar uma fonte de luz no contexto de modelos simplificados utilizados em síntese de imagens (como os computadores "vêem" fontes luminosas?)
- caracterizar o material e a rugosidade de uma superfície no contexto de modelos simplificados utilizados em síntese de imagens (como os computadores "vêem" as propriedades de uma superfície?).
- caracterizar os modelos de iluminação considerados em síntese de imagens (como os computadores "vêem" as interações das radiações luminosas em um ambiente?)
- aplicar o modelo de iluminação local para obter a cor de uma amostra da superfície.
- diferenciar traçado de raio da radiosidade.
- aplicar a técnica de traçado de raio para obter uma cor para uma amostra da superfície.
- caracterizar as diferentes técnicas de tonalização.

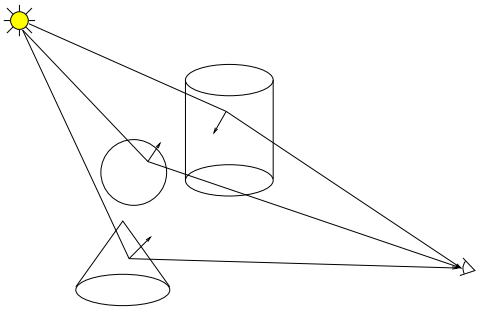


Figura 8.1: Interações de radiações luminosas.

- aplicar as técnicas de tonalização para computador preencher a área de uma faceta poligonal.

Como vimos na seção 7.2, a nossa percepção de cores se dá como resultado das interações entre as radiações/energia luminosas oriundas da superfície dos objetos de interesse com as células fotosenssíveis na retina. Estas radiações partem de uma ou mais fontes de energia luminosas, algumas são absorvidas e outras são refletidas pelas superfícies presentes em uma cena conforme o material e a geometria destas (Figura 8.1).

Se "pintarmos" um quadro com as cores percebidas, como ilustra Figura 8.2.(a), teremos uma imagem bem próxima da nossa realidade tridimensional. Neste capítulo estamos interessados em estudar como um computador "veria" as radiações e as "converteria" em cores, além de aplicá-las em cada *pixel* de uma tela de exibição de forma que o resultado final seja similar ao de uma tela produzida por um artista, como mostra Figura 8.2.(b).

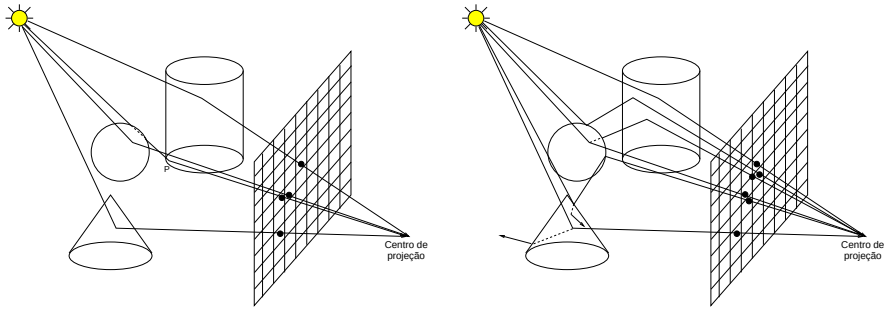
Teoricamente, para "colorir um *pixel*", é suficiente determinar as radiações luminosas que são refletidas pela área da superfície projetada nele. No entanto, as interações entre os raios luminosos, a superfície dos objetos de interesse e a visão de um observador é um processo complexo. O procedimento que nos permite prever a cor de um ponto sob o efeito de uma ou mais fontes de luz ou saber o que causou a formação de uma cor específica é denominado **modelo de iluminação**. Embora não exista ainda uma formulação teórica completa, os modelos existentes foram suficientes para inspirar algoritmos computacionalmente factíveis e capazes de fazer o computador "ver" cores "reais" no espaço de universo WC e produzir em telas de exibição (espaço DC) imagens que proporcionam a "ilusão" de uma cena real (Seção 5.3).

(a) (b)

Figura 8.2: Imagens próximas da realidade tridimensional: (a) pintura (Fonte: http://www.jesuspaintings.com/baltimore_oriole.htm), (b) gerada por computador (Fonte: http://tralvex.com/pub/rover/gg2.htm).

Para efeito de síntese de imagens, há dois paradigmas de "visão de cores no computador". Em um paradigma, denominado **modelo de iluminação local**, somente as interações entre as radiações oriundas diretamente das fontes de luz com as superfícies são "percebidas" pelo computador, como ilustra Figura 8.3.(a). Embora este modelo não condiga com o comportamento físico de radiações luminosas, ele é simples e gera, em muitos casos, imagens perceptivelmente convincentes. Veremos na seção 8.3 algumas soluções que contornam problemas apresentados por este modelo. O segundo paradigma se aproxima do modelo físico da nossa percepção. Neste paradigma o computador "veria" idealmente todas as possíveis radiações que fluem em um ambiente. Por esta consideração de uma ambiente como todo, a classe de algoritmos que seguem este paradigma é conhecida como **modelo de iluminação global** (Figura 8.3.(b)). Dependendo do modelo utilizado para representar a luz, distingue-se ainda duas grandes sub-classes de modelo de iluminação global: traçado de raio e radiosidade. Na seção 8.5 vamos detalhar a técnica de traçado de raio.

Aplicar um modelo de iluminação em todas as regiões de uma superfície que são projetadas em um *pixel* para estimar a cor que deveria ser atribuído



(a) (b)

Figura 8.3: Cor em imagens: (a) "percepção" local da máquina; (b) "percepção" global da máquina.

ao *pixel* de forma a resultar uma imagem próxima da nossa percepção é um processo muito custoso computacionalmente. Como uma solução de compromisso entre eficiência e qualidade das imagens, Gouraud propôs em 1971 **tonalizar** cada faceta projetada no espaço da imagem a partir das cores computadas nos respectivos vértices. Mais tarde, Phong apresentou uma outra forma de interpolar os parâmetros para atenuar alguns artefatos. Na seção 8.4 mostraremos como os dois algoritmos de tonalização operam.

No entanto, não basta ter apenas um procedimento de "ver" e "colorir". Para gerar uma imagem dentro da expectativa do usuário, o computador precisa interagir com este para "saber" a concepção que este tem em mente: os tipos de radiações, as propriedades das superfícies, e a disposição destes em relação ao observador. No capítulo 3 vimos como descrever as formas geométricas das superfícies na "linguagem do computador"; no capítulo 4 mostramos como especificar, em linguagem matemática, os movimentos básicos destas formas no espaço para que elas fiquem posicionadas conforme a nossa expectativa. Nas seções 8.1 e 8.2 deste capítulo mostraremos como definir, em linguagem matemática, os dois elementos restantes: fontes de radiações luminosas e o material das superfícies.

8.1 Modelo de Radiações Luminosas

A luz é um conjunto de radiações capazes de estimular o sistema de visão humano. O estudo da luz pode ser dividido em três partes:

1. Óptica geométrica: estuda as consequências do princípio de propagação retilínea dos raios luminosos,

2. Óptica energética: estuda o comportamento radiante dos raios luminosos e suas interações com a matéria.

3. Óptica física: estuda a dispersão, a interferência, a difração e a polarização da luz.

Existem vários modelos para descrever o comportamento da luz. A maioria dos algoritmos utilizados em sistemas de informação gráfica é ainda baseada no modelo de ondas eletromagnéticas. Uma onda eletromagnética é constituída de dois campos oscilantes perpendiculares entre si: elétrico e magnético. Ambos são perpendiculares à direção de propagação da onda. No vácuo, a velocidade de propagação é aproximadamente 3.0×10^8m/s. A energia da onda é proporcional ao quadrado da amplitude da onda. E o comprimento de onda, λ, determina a cor que percebemos, como vimos no capítulo 7.

Chamamos de **fluxo (de energia) luminoso** Φ (em lúmens) de um objeto a razão entre a quantidade de energia radiante que um objeto emite durante um certo intervalo de tempo. A **intensidade luminosa** (em candelas) de um objeto em uma determinada direção refere-se à razão entre o fluxo luminoso que ele emite através de um ângulo sólido w cujo eixo é a direção considerada

$$I = \frac{\Phi}{\omega}.$$

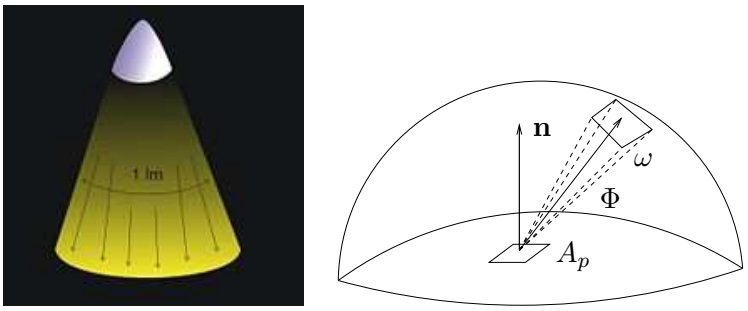
Só se pode falar em intensidade luminosa de uma fonte quando ela for pontual, isto é, quando as suas dimensões forem desprezíveis em relação à distância de que é observada.

A **emitância luminosa** (ou **radiância luminosa**) de uma fonte é a razão entre o fluxo luminoso que ela emite e a área da sua superfície.

$$I = \frac{\Phi}{A_p}.$$

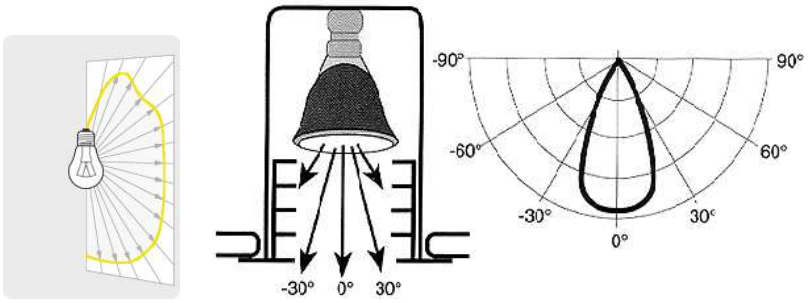
Denominamos como **fontes de luz** objetos em uma cena capazes de emitir ondas eletromagnéticas da faixa espectral visível. Estas ondas estimulam as células fotoreceptoras e causam sensação de cor. Vimos na seção 7.2 que esta sensação de cor pode ser representada por um vetor de três componentes primárias: vermelho R, verde G, e azul B. Portanto, podemos caracterizar as radiações emitidas por um vetor de três escalares.

Além da intensidade luminosa emitida, deve-se descrever também a distribuição desta intensidade para todas as direções, medidas em ângulo, a posição P e a direção $\mathbf{d}$ da fonte a fim de que o computador possa "ver"



(a) (b)

Figura 8.4: Fluxo luminoso: (a) emitido por uma fonte (Fonte: http://pt.wikipedia.org/wiki/Fluxo_luminoso), (b) esquema.



(a) (b)

Figura 8.5: Curva de distribuição de intensidade luminosa de: (a) uma lâmpada incandescente; (b) uma luminária *spot*.

a área de abrangência de cada fonte e determinar corretamente a claridade em cada ponto da superfície. Figura 8.5 mostra a curva de distribuição de intensidade luminosa de duas fontes luminosas.

Tanto a posição P_0 quanto a direção **d** da fonte podem ser representadas numericamente como vimos no capítulo 3. Resta ver como descrever as curvas de distribuição de intensidade luminosa. Para simplificar, aproxima-se usualmente as curvas para uma das curvas típicas de três classes de fontes de luzem sistemas de informação gráfica: direcional, pontual, e *spot*.

8.1.1 Fonte Direcional

É também conhecida como **fonte distante**. A energia luminosa flui uniformemente no espaço numa direção específica **d**, como se a fonte estivesse

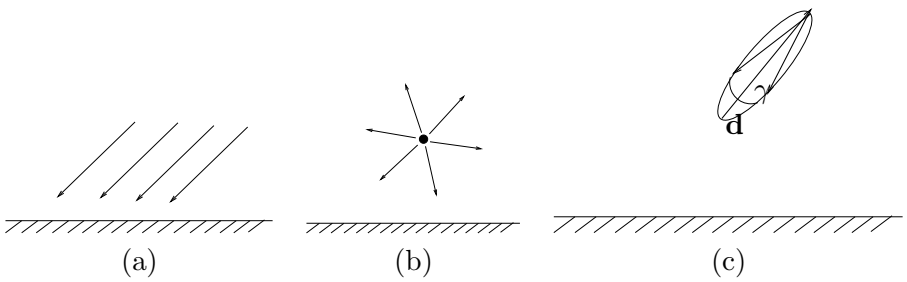


Figura 8.6: Fonte de luz: (a) direcional; (b) pontual; e (c) *spot*.

localizada no "infinito". Portanto, somente a intensidade luminosa da fonte

$$I = \begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix}$$

e a direção $\mathbf{d}$ são suficientes para especificá-la:

$$\mathbf{d} = \begin{bmatrix} d_x \\ d_y \\ d_z \\ 0 \end{bmatrix}. \tag{8.1}$$

Tipicamente, o fluxo luminoso irradiado por esta fonte é constante ao longo da trajetória (Figura 8.6.(a)).

8.1.2 Fonte Pontual

Também chamada **fonte puntiforme**. Esta fonte é uma aproximação para classes de fontes de luz que distribuem radialmente o fluxo luminoso a partir de um ponto P_0 (Figura 8.6.(b))

$$P_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}. \tag{8.2}$$

Tipicamente, considera-se que o fluxo atenue à medida que se afasta da fonte através de um **fator de atenuação**, cujo valor varia entre 0.0 e 1.0, para que objetos localizados mais longe de uma fonte de luz pontual recebam menos fluxo luminoso do que aqueles que estejam próximos. Em decorrência disso,

os primeiros vão aparentar mais escuros. Dada uma fonte pontual em P_0 com uma intensidade luminosa $I(P_0)$. A intensidade luminosa no ponto P seria

$$I(P) = f_{at} \begin{bmatrix} I_r(P_0) \\ I_g(P_0) \\ I_b(P_0) \end{bmatrix}.$$

Três expressões mais usuais para "representar" matematicamente o **fator de atenuação** são

- $f_{at} = \frac{1}{d}$, e
- $f_{at} = \frac{1}{d^2}$, e
- $f_{at} = min(\frac{1}{a_1+a_2d+a_3d^2}, 1)$.

Embora a segunda expressão simplifique o cômputo de f_{at}, evitando o cálculo da raíz quadrada, pois $(\sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2})^2 = (x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2$, ela tende a gerar contrastes fortes na iluminação de objetos relativamente próximos por causa da variação quadrática. Isso pode ser agravado com distorções perspectivas. A terceira expressão é uma generalização das duas primeiras expressões.

8.1.3 Fonte *Spot*

Ela modela uma classe de fontes de luz que apresentam curvas de distribuição de intensidade luminosa limitadas a um ângulo sólido. É caracterizada como um feixe cônico ou pincel de raios luminosos de intensidade $I = (I_r, I_g, I_b)$ que parte de uma posição P_0 em direção do vetor unitário $\mathbf{d}$, formando uma abertura de ângulo γ (Figura 8.6.(b)). O controle do decaimento da intensidade luminosa é através do **expoente da fonte spot** c, pois a intensidade luminosa que chega em um ponto P qualquer é dada por

$$I(P) = I(\frac{\mathbf{d} \cdot \overrightarrow{P_0P}}{|\mathbf{d} \cdot \overrightarrow{P_0P}|})^c.$$

Figura 8.7 mostra o efeitos nas curvas de distribuição de intensidade para distintos valores de c.

8.2 Modelo da Superfície

As propriedades elétricas do meio por onde a luz passa afetam a sua trajetória de propagação. O campo magnético da onda afeta os elétrons no

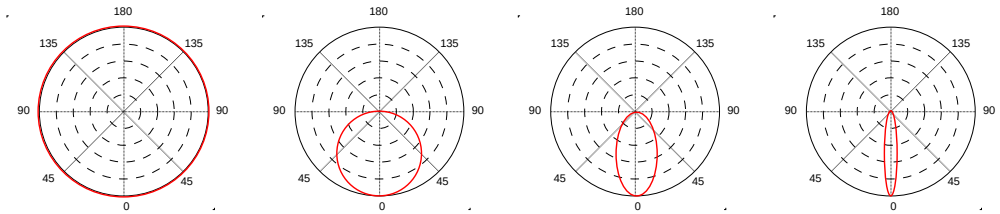


Figura 8.7: Influência de c na distribuição de intensidade luminosa: (a) c=0; (b) c=1; (c) c=4; e (d) c=32.

material e produz diferentes efeitos ópticos. De acordo com o movimento dos elétrons, distinguem-se duas classes de materiais:

dielétricos: os elétrons são bastante estáveis; portanto, afeta muito pouco a direção de propagação e desacelera a velocidade de propagação da onda, e

condutores: há muitos elétrons livres; portanto, novas ondas eletromagnéticas podem ser geradas e emitidas.

Os **índices de refração** são utilizados para caracterizar as propriedades ópticas dos meios. O **índice de refração absoluto** de um meio é a razão entre a velocidade de propagação da luz no vácuo e a velocidade de propagação da luz no meio considerado.

Ao mudar de meios de propagação de índices de refração distintos, as radiações luminosas sofrem **reflexões** e **refrações**. A reflexão corresponde ao fenômeno de que parte das radiações volta a se propagar no mesmo meio no qual a luz incide e a refração refere-se ao fenômeno da outra parte da luz atravessar a superfície de separação dos dois meios e propagar-se no outro meio. Os materiais condutores refletem praticamente todas as radiações incidentes; por isso eles são tipicamente opacos. Por outro lado, os materiais dielétricos sempre absorvem uma fração das radiações incidentes, podendo gerar uma sensação de transparência/translucidez. Figura 8.8.(a) apresenta um esquema de relação entre estas energias. Uma versão vetorial deste esquema é dada na Figura 8.8.(b). Em Fotometria, chamamos de **iluminância** (ou **aclaramento**) o total de fluxo luminoso que incide sobre uma superfície (de separação) por unidade de área.

A taxa de fluxo luminoso refletido $F_r = \frac{\Phi_r}{\Phi_i}$ e a taxa de fluxo refratado $F_t = \frac{\Phi_t}{\Phi_i}$ são conhecidas, respectivamente, por **refletância de Fresnel** e

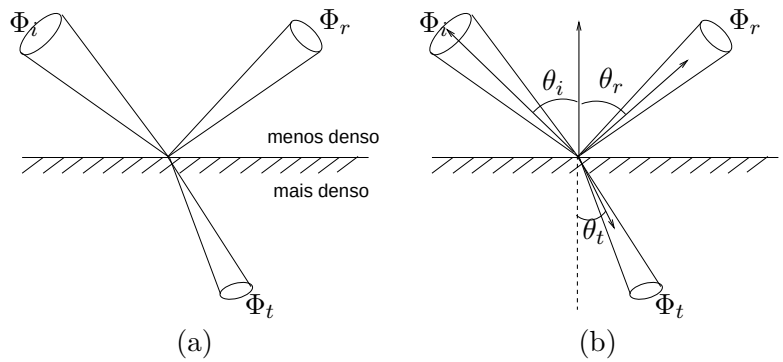


Figura 8.8: Energia luminosa × superfície: (a) interações e (b) notação vetorial.

refratância de Fresnel. Pelo princípio de conservação de energia

$$\Phi_i = \Phi_r + \Phi_t = F_r\Phi_i + F_t\Phi_t,$$

ou seja $F_r + F_t = 1.0$. Para dielétricos com luz não polarizada num meio de índice de refração próximo ao do vácuo, a reflectância de Fresnel é dada pela expressão

$$F_r = \frac{1}{2}[\frac{\sin^2(\theta_i - \theta_t)}{\sin^2(\theta_i + \theta_t)} + \frac{\tan^2(\theta_i - \theta_t)}{\tan^2(\theta_i + \theta_t)}].$$

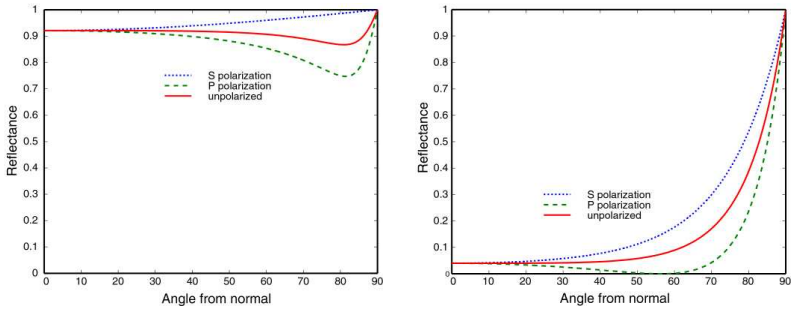
Diferentemente dos materiais condutores, os materiais dielétricos podem exibir uma alta especularidade quando o ângulo de incidência fique próximo de 90^o, como ilustra o gráfico da reflectância de distintos tipos de luz na Figura 8.9.

Figura 8.10 ilustra a diferença em imagens que levam em conta a reflectância de Fresnel e as imagens que não levam. Observe como fica mais especular a borda do vidro na Figura 8.10.(b).

As direções dos fluxos refletidos e refratados obedecem as **leis de Snell-Descartes**, que foram originalmente descobertas por Ibn Sahl no século VIII, redescobertas por Willebrord Snell em 1621 e, num trabalho independente, formuladas em termos de senos por René Descartes em 1637:

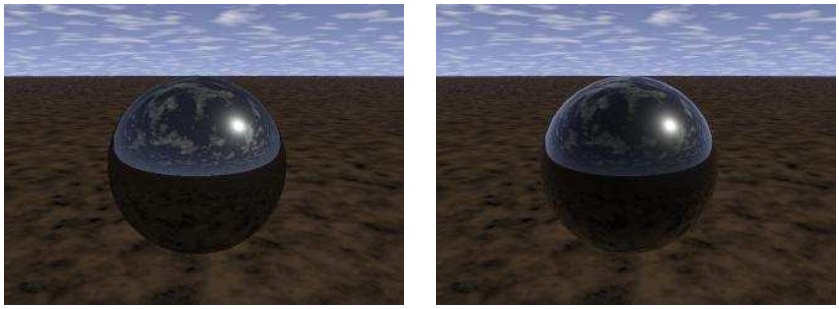
Reflexão : a direção do fluxo incidente, a direção do fluxo refletido e o vetor normal da superfície (de separação) S em P estão no mesmo plano e o ângulo de incidência θ_i e o ângulo de reflexão θ_r são iguais.

Refração : a direção do fluxo incidente, a direção do fluxo refratado, e o vetor normal de S em P estão no mesmo plano e os ângulos de



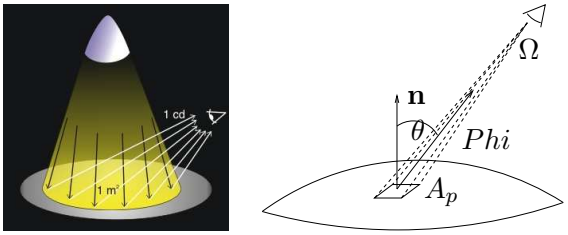
(a) (b)

Figura 8.9: Reflectância × θ_i: (a) condutores e (b) dielétricos (Fonte:http://www.graphics.cornell.edu/%7Ewestin/misc/fresnel.html).



(a) (b)

Figura 8.10: Imagens: (a) sem reflectância de Fresnel e (b) com reflectância de Fresnel (Fonte:http://www.graphics.cornell.edu/%7Ewestin/misc/fresnel.html).



(a) (b)

Figura 8.11: Luminância: (a) percebida por um observador (Fonte: http://pt.wikipedia.org/wiki/Ficheiro:Luminancia.jpg); (b) representação vetorial.

incidência e de refração satisfazem a seguinte relação

$$n_i sen\theta_i = n_t sen\theta_t, \tag{8.3}$$

onde θ_t é o ângulo de refração e n_i e n_t, os índices de refração dos dois meios. Observe na Figura 8.8.(b) que o meio onde o fluxo incide é menos denso do que o segundo meio, pois o ângulo em relação à normal ficou mais "agudo" .

A **luminância** L de uma superfície na direção θ em relação à sua normal **n** é o fluxo luminoso Φ naquela direção, subtendido no ângulo sólido Ω do campo de visão do observador, por unidade de área A_p (Figura 8.11)

$$L = \frac{\Phi}{A_p \cos\theta \Omega}.$$

Esta grandeza fotométrica indica quão "brilhante" uma superfície aparenta.

Se a superfície de separação de dois meios for perfeitamente especular ou polida, as radiações luminosas refletidas e refratadas seguirão **coerentemente** a relação de Snell-Descartes. Quando a superfície de separação for muito rugosa, como a apresentada na Figura 8.12.(a), a luminância percebida pelo observador seria algo "difuso", pois as radiações refletidas podem ficar não só **incoerentemente** orientadas, como também oclusas por protuberâncias locais (Figura 8.12.(b)). Torrence e Sparrow propuseram em 1967 modelar uma superfície rugosa por um conjunto de microfacetas e derivaram a partir dele uma **função de atenuação geométrica** para "corrigir" o fluxo ideal de reflexão. A função de atenuação geométrica descreve a proporção de microfacetas efetivamente vistas por ambos, o observador e a fonte de luz. Figura 8.12.(a) mostra os parâmetros da função.

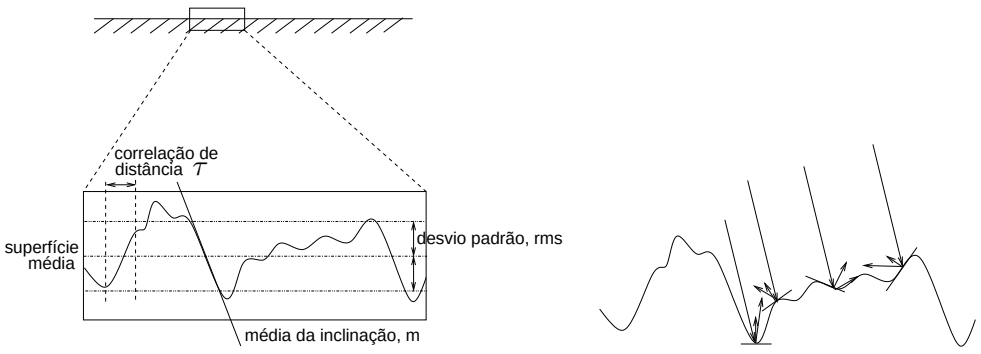


Figura 8.12: Visão microscópica de uma superfície: (a) parâmetros; e (b) orientação difusa das radiações refletidas.

Existe uma classe especial de superfícies, denominadas **superfícies lambertianas**.Elas podem ser emissoras e refletoras. As superfícies lambertianas refletoras caracterizam-se por refletirem radiações cuja intensidade por unidade de área é dependente somente do cosseno do ângulo de incidência. Ou seja, as radiações percebidas por um observador não dependem do ângulo entre a sua linha de visão e o vetor normal da superfície. Estas superfícies são ditas **perfeitamente difusas** e satisfazem a **lei de Lambert** de irradiação

$$i \;=\; I cos\theta_i, \tag{8.4}$$

onde i é a intensidade refletida em todas as direções, I a intensidade da fonte de iluminância e θ_i é o ângulo de incidência.

Apesar de dispor de modelos (incompletos) que representem matematicamente a influência das propriedades físicas e geométricas da superfície (de separação entre dois meios) na sua luminância, a aplicação destes resultados é ainda muito modesta. Uma das principais razões é a sua complexidade e incompletude (por exemplo, não leva em consideração nestes modelos superfícies desgastadas por ação de tempo). Na maioria dos sistemas de informação gráfica adota-se ainda uma solução bem empírica, porém satisfatória: modelar a influência da superfície por meio de constantes k que atuam como "redutores" das intensidades ideais, como veremos na seção 8.3.

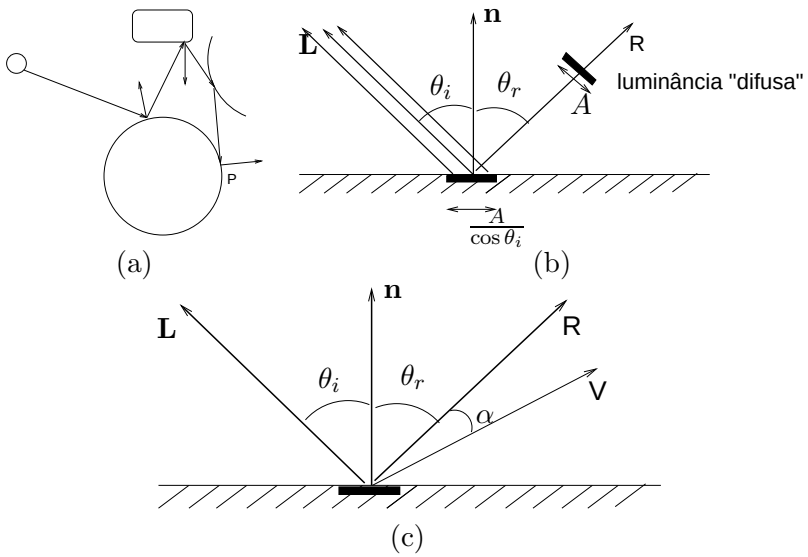


Figura 8.13: Componente: (a) ambiente; (b) difusa; e (c) especular.

8.3 Modelos de Iluminação Local

Dispondo da informação das fontes de radiações luminosas incidentes e da constante redutora da intensidade idealmente refletida de cada componente, como um computador consegue processar e "ver", em cada ponto da superfície, a intensidade luminosa que deve ser percebida por um observador médio? Como já comentamos na introdução, embora existam várias propostas para determinar esta intensidade com base em leis físicas, impera-se ainda o modelo empírico de Phong da década 1970. A principal razão da sua popularidade é produzir resultados satisfatórios com simplicidade. No modelo de Phong a intensidade luminosa incidente em cada ponto (x, y, z) é decomposta em três componentes: ambiente (I_a), difusa ($I_{d,i}$) e especular ($I_{s,i}$). Observamos que cada uma das componentes é, por sua vez, representado por um vetor de três cores primárias.

8.3.1 Reflexão Difusa

Assume-se que todas as superfícies tenham comportamento lambertiano, refletindo igualmente a parcela $I_{d,i}$ das radiações oriundas diretamente da fonte de luz i para todas as direções. A intensidade luminosa percebida por um observador pode ser aproximada pela Eq. 8.4 atenuada por um **coeficiente de reflexão difusa** k_d (Figura 8.13.(a)). Sabendo que $\mathbf{L} \cdot \mathbf{n} =$

$\cos\theta_i$, podemos escrever a contribuição difusa de m fontes de luz no ponto (x, y, z) como

$$I_d(x, y, z) = \sum_{i=1}^{m} k_d I_{d,i}.(\mathbf{L}_i \cdot \mathbf{n}), \tag{8.5}$$

onde $\mathbf{L}_i$ é a direção da fonte de luz i em relação ao ponto e $\mathbf{n}$ é o vetor normal da superfície no ponto.

8.3.2 Reflexão Ambiente

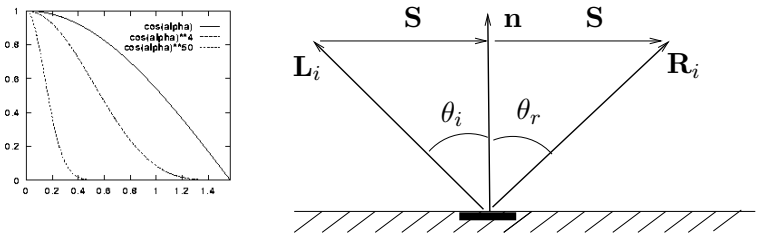
Pela Eq. 8.5 todos os pontos que não recebem radiações diretas das fontes de luz ficarão em preto, o que não condiz com a nossa vivência. A trajetória das ondas luminosas em um ambiente é complexa: um ponto pode receber radiações luminosas de uma fonte i de forma indireta, após multireflexões, como ilustra a Figura 8.13.(b). Observe que o ponto P na figura é iluminado indiretamente pela fonte de luz após o feixe luminoso sofrer 3 reflexões. Uma forma simples de incluir este fenômeno é adicionar uma parcela constante I_a atenuada pelo **coeficiente de reflexão difusa de ambiente** k_a

$$I_a(x, y, z) = k_a I_a \tag{8.6}$$

A "luz de fundo" não é suficiente para diferenciar duas formas geométricas posicionadas em distâncias distintas em relação ao observador, se elas forem iluminadas igualmente. Para contornar isso, podemos adicionar ainda um fator de atenuação à intensidade luminosa refletida em função da distância d do seu trajeto entre a superfície e o observador. Na prática, o fator de atenuação no trajeto entre a fonte e a superfície, como vimos na seção 8.1.2, é suficiente para muitos casos.

8.3.3 Reflexão Especular

No seu trabalho, Phong observou também que quando se trata de uma superfície lustrosa, há pontos de brilho cuja percepção depende da posição do observador em relação ao vetor normal da superfície. Para gerar este efeito, ele propôs considerar que uma parcela $I_{s,i}$ das radiações da fonte de luz i seja refletida especularmente, e que a intensidade luminosa percebida pelo observador varie com o ângulo α entre a direção do observador $\mathbf{V}$ e a direção do raio refletido $\mathbf{R}$ (Figura 8.13.(c)). A contribuição especular de m fontes de luz no ponto (x, y, z) atenuada pelo **coeficiente de reflexão**



(a) (b)

Figura 8.14: Componente especular: (a) $\cos^n \alpha$; (b) vetor de reflexão.

especular k_s é dada por

$$I_s(x,y,z) = k_s \sum_{i=1}^{m} I_{s,i}(\mathbf{V} \cdot \mathbf{R}_i)^n, \tag{8.7}$$

onde $\mathbf{V}$ é a direção do observador em relação ao ponto, $\mathbf{R}_i$, a direção do raio refletido do feixe incidente $I_{s,i}$ e n, o **expoente de reflexão especular**. O papel deste coeficiente é para controlar empiricamente os tamanhos dos pontos de brilho. Quanto maior o valor, mais "concentrado" fica o ponto de brilho conforme mostra o gráfico da Figura 8.14.(a). Tipicamente, atribui-se uma valor em torno de 100 para uma superfície muito "lustrosa" e em torno de 1 para uma superfície fosca.

Para determinar $I_s(x,y,z)$ utilizando Eq. 8.7, precisamos conhecer o vetor $\mathbf{R}_i$. Vamos mostrar que é possível obtê-lo a partir do vetor normal $\mathbf{n}$ e do vetor $\mathbf{L}_i$, pois (Figura 8.14.(b))

$$\begin{aligned} \mathbf{R}_i &= \mathbf{L}_i + 2\mathbf{S} \\ &= \mathbf{L}_i + 2((\mathbf{L}_i \cdot \mathbf{n})\mathbf{n} - \mathbf{L}_i) \\ &= 2(\mathbf{L}_i \cdot \mathbf{n})\mathbf{n} - \mathbf{L}_i \end{aligned} \tag{8.8}$$

8.3.4 Modelo de Iluminação de Phong

Combinando as Eqs. 8.5, 8.6 e 8.7, temos o seguinte modelo de iluminação local para determinar a intensidade luminosa em um ponto (x,y,z) iluminado pelas m fontes de luz

$$I(x,y,z) = I_a(x,y,z) + f_{at}(k_d \sum_{i=1}^{m} I_{d,i}.(\mathbf{L}_i \cdot \mathbf{n}) + k_s \sum_{i=1}^{m} I_{s,i}(\mathbf{V} \cdot \mathbf{R}_i)^n), \quad k_a, k_s, k_d \in [0,1]. \tag{8.9}$$

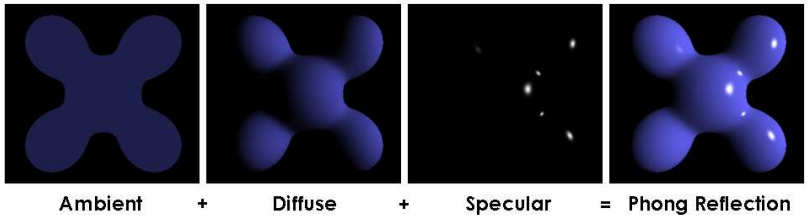


Figura 8.15: Influência de cada componente na cor final (Fonte: http://en.wikipedia.org/wiki/Phong_shading).

Observamos que como as radiações luminosas emitidas pelas fontes de luz I_a, $I_{d,i}$ e $I_{s,i}$ são representadas pelas três componentes primárias I_r, I_g, e I_b, é comum especificar os fatores de atenuação da superfície k_a, k_d e k_s para cada uma dessas componentes. Os coeficientes k_a, k_d e k_s são atribuídos de forma totalmente empírica. Uma regra prática é atribuir valores de tal forma que $k_a = k_d$ e $k_d + k_s = 1.0$. Quanto maior o grau de rugosidade de uma superfície, maior deve ser o valor de k_d e menor o coeficiente especular.

Figura 8.15 ilustra a influência de cada componente na cor final de uma imagem.

8.3.5 Modelo de Iluminação Blinn-Phong

Uma variante do modelo de Phong foi proposto por Blinn em 1977, que consiste em substituir o termo $(\mathbf{V} \cdot \mathbf{R}_i)^n$ pelo

$$(\mathbf{n} \cdot \mathbf{H})^{n'}, \tag{8.10}$$

onde $\mathbf{H}$ é a média dos dois vetores que não dependem da geometria da superfície (de separação): $\mathbf{L}_i$ e $\mathbf{V}$, isto é,

$$\mathbf{H} = \frac{\mathbf{L}_i + \mathbf{V}}{|\mathbf{L}_i + \mathbf{V}|}$$

A vantagem desta substituição é logo percebida, quando consideramos casos em que o observador e a fonte de luz estiverem distantes da superfície. Nestes casos, o vetor $\mathbf{H}$ será constante para todos os pontos, requerendo apenas o seu cálculo uma única vez. Além disso, veremo na seção 8.5 que esta formulação simplifica o cômputo de componentes especulares no algoritmo de traçado de raio.

Observamos que β e ψ tem valores diferentes, pois

$$\theta_i + \psi \quad = \quad (\theta_r - \psi) + \beta$$

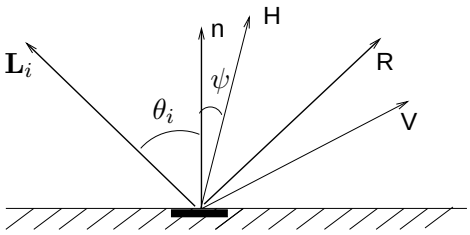


Figura 8.16: Vetor **H**.

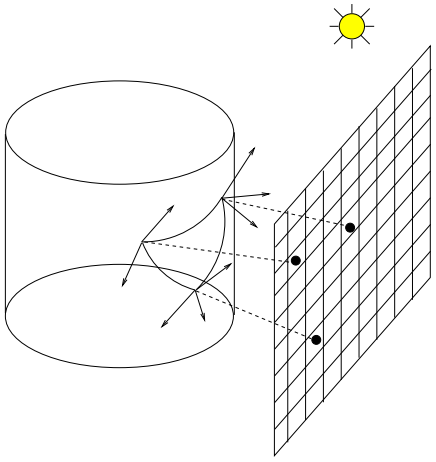


Figura 8.17: Tonalização.

$$\begin{aligned} \psi &= -\psi + \beta \\ 2\psi &= \beta. \end{aligned}$$

Consequentemente, para obter mesmos resultados visuais com os dois modelos, é necessário escolher valores distintos para o expoente de reflexão especular de cada modelo.

8.4 Tonalização

Sendo o processo de determinação da cor em cada ponto um processo relativamente mais demorado, é comum em sistemas de informação gráfica determinar a cor com uso da Eq. 8.9 somente em algumas amostras, tipicamente nos vértices, e interpolar os valores nos restantes *pixels* no espaço de imagens como está esquematizado na Figura 8.17.

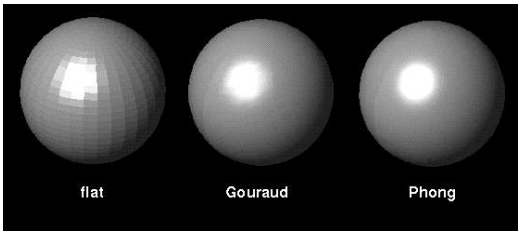


Figura 8.18: Tipos de tonalização.

Os três tipos de tonalização mais conhecidos são: constante, *Gouraud* e *Phong*.

8.4.1 Tonalização Constante

Consiste em simplesmente calcular a cor para um vértice de cada faceta no espaço de universo (WC) e propagá-la para todos os pontos restantes da face durante a rasterização (Seção 10.2.3). Este tipo de tonalização é conhecida por **tonalização constante**. Devido às variações abruptas da intensidade luminosa entre as facetas, a ação inibidora de cada fotoreceptor fora do seu centro de campo receptivo acentua ainda mais estas variações, como vimos na seção 1.1, causando a sensação de bordas "bem quebradas" entre as facetas como ilustra Figura 8.18.

8.4.2 Tonalização de *Gouraud*

Para suavizar estas variações, Gouraud propôs em 1971 um esquema de interpolação linear das intensidades entre os vértices, determinadas pela Eq. 8.9 no espaço de universo WC. Este procedimento é conhecido como **tonalização de Gouraud** e é aplicado durante a rasterização (Seção 10.2.3). Supondo que as intensidades dos vértices do triângulo da Figura 8.19 tenham sido calculadas no espaço de universo WC, podemos obter a intensidade I_4 pela interpolação linear

$$I_4 = \frac{y_2 - y_4}{y_2 - y_1} I_1 + \frac{y_4 - y_1}{y_2 - y_1} I_2,$$

e de forma análoga, a intensidade I_5

$$I_5 = \frac{y_2 - y_5}{y_2 - y_3} I_3 + \frac{y_5 - y_1}{y_2 - y_3} I_2.$$

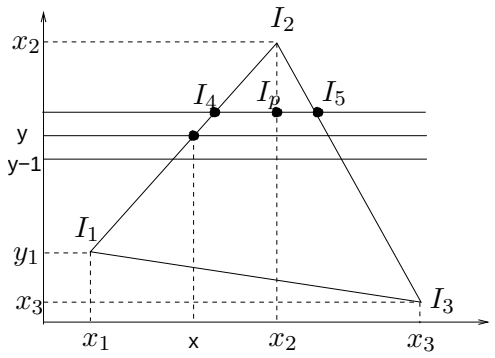


Figura 8.19: Tonalização de *Gouraud*.

Interpolando linearmente estas duas intensidades das arestas do triângulo, podemos obter a intensidade I_p em qualquer ponto no interior do triângulo

$$I_p = \frac{x_p - x_4}{x_5 - x_4} I_5 + \frac{x_5 - x_p}{x_5 - x_4} I_4.$$

A regularidade da organização dos *pixels* nos permite ainda elaborar um esquema recorrente de cálculo das intensidades, aproveitando os cálculos anteriores. Vamos supor que na linha y seja determinada a intensidade no ponto (x, y)

$$I_y = \frac{y_2 - y}{y_2 - y_1} I_1 + \frac{y - y_1}{y_2 - y_1} I_2,$$

a intensidade na próxima linha $y - 1$ seria

$$I_{y-1} = \frac{y_2 - y + 1}{y_2 - y_1} I_1 + \frac{y - 1 - y_1}{y_2 - y_1} I_2 = I_y + \frac{I_2 - I_1}{y_2 - y_1}.$$

8.4.3 Tonalização de *Phong*

Um dos problemas que a tonalização de *Gouraud* apresenta é perda de pontos de brilho no meio de uma faceta. Para evitar isso, recomenda-se aumentar o número de vértices na malha que a representa, ou seja, refinar mais a malha poligonal. Uma outra alternativa seria "estimar" os vetores normais em cada amostra da "faceta" através da interpolação linear dos vetores normais dos seus vértices durante a rasterização e aplicar a Eq. 8.9 em cada amostra mapeada em um *pixel*, a fim de obter uma intensidade mais próxima da percepção real. Chamamos este tipo de interpolação **tonalização de Phong**.

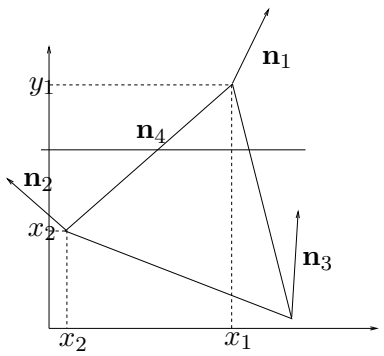


Figura 8.20: Tonalização de *Phong*.

O esquema de interpolação é similar ao de *Gouraud*, substituindo apenas as intensidades pelos vetores normais. Por exemplo, para obter o vetor normal $\mathbf{n}_4$ na Figura 8.20 utilizamos a seguinte expressão

$$\mathbf{n}_4 = \frac{y_4 - y_2}{y_1 - y_2}\mathbf{n}_1 + \frac{y_1 - y_4}{y_1 - y_2}\mathbf{n}_2,$$

A tonalização de Phong pode gerar imagens "falsas" quando a superfície não é suave.

8.5 Traçado de Raio

Diferentemente do modelo de iluminação local, um modelo de iluminação global considera idealmente que o computador consiga "ver" todo o fluxo luminoso que chegue em cada ponto da superfície. Tanto as radiações vindas diretamente das fontes de luz como também todas as radiações indiretas das fontes de luz, refletidas ou refratadas por outras superfícies presentes na cena de interesse, são consideradas no cômputo da cor de cada *pixel*. Adicionalmente, o modelo de iluminação global considera a possibilidade de uma radiação direta ser bloqueada por um outro objeto opaco ao longo da sua trajetória. Por exemplo, o ponto P da Figura 8.3.(a) receberia radiações diretas se a esfera não estiver no trajeto destas radiações. No modelo de iluminação local, como o computador só "vê" relações diretas entre luz–superfície–observador, incidirão sobre o ponto P as radiações diretas; enquanto no modelo de iluminsção global, o computador "veria" a esfera no trajeto das radiações e "perceberia" que estas radiações seriam bloqueadas, gerando sombra sobre o ponto P.

Figura 8.21: Modelo de iluminação global: (a) traçado de raio (Fonte: http://hof.povray.org/); (b) radiosidade (Fonte: http://blenderartists.org/cms/).

Nesta seção vamos apresentar as principais idéias sobre modelos de iluminação global. Existem essencialmente duas classes de algoritmos de iluminação global: **radiosidade** e **traçado de raio**. Enquanto o traçado de raio se baseia no mesmo princípio físico do modelo de iluminação local, a técnica de radiosidade adota um paradigma totalmente distinto. Ela se fundamenta na Teoria de Transferência de Calor para modelar a troca de fluxo luminoso entre as superfícies. Diferentemente do traçado de raio em que o cálculo da especularidade requer a posição do observador, a radiosidade não depende da posição do observador, já que ela considera que as superfícies sejam lambertianas. Grosso modo, podemos dizer que radiosidade é um modelo de iluminação (difusa) global e o traçado de raio, um modelo de iluminação (especular e de refração) global. Figura 8.21 apresenta duas imagens: uma gerada pelo algoritmo de traçado de raio e outra pela radiosidade.

Pela sua similaridade com o modelo de iluminação local em termos de fundamento teórico, vamos detalhar a seguir apenas a técnica de traçado de raio. Ela consiste em rastrear o percurso de um raio a partir do centro de projeção até a fonte de luz ou até um nível de propagação pré-estabelecido. Uma alternativa seria lançar um **raio primário** por *pixel*, em inglês *pixel ray*, e computar a superfície mais próxima do centro de projeção. Na Figura 8.22.(a), para o raio primário desenhado a superfície mais próxima seria S_1. Aplicando a lei de Snell-Descartes (Seção 8.2), lançam-se dois **raios secundários**: um na direção do raio de refração e outro na direção do raio de reflexão, denominadas, respectivamente, como T_1 e R_1 na Figura 8.22.(a). Cada superfície atingida pelo raio secundário é armazenada

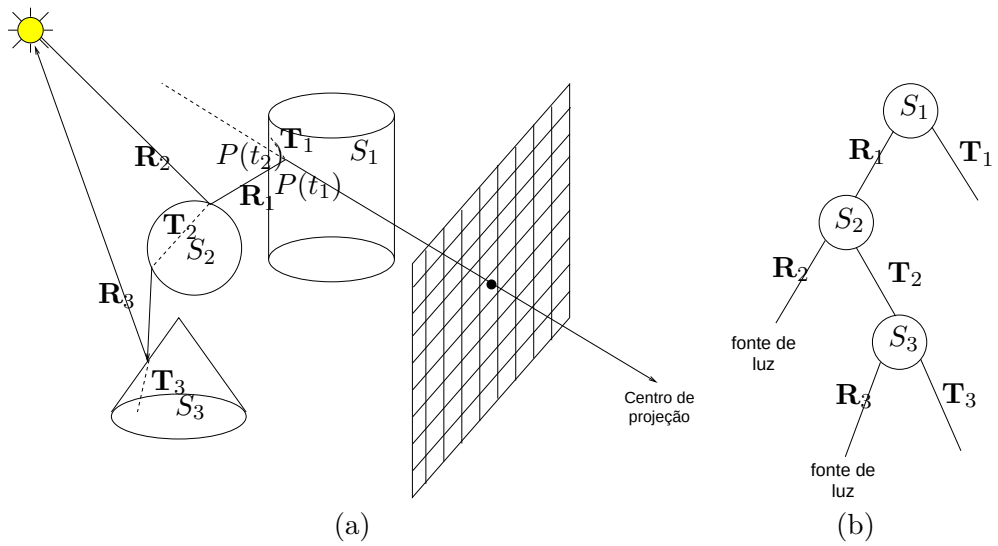


Figura 8.22: Traçado de raio: (a) caminho especular; (b) árvore binária.

numa árvore binária, cuja raíz é a superfície atingida pelo raio primário e cujos ramos correspondem aos dois raios secundários derivados, como ilustra Figura 8.22.(b). O procedimento é repetido para cada raio secundário até que as condições de parada sejam satisfeitas.

Para determinar a cor a ser atribuída para cada *pixel*, percorre-se a árvore no sentido de folhas até a sua raiz a fim de coletar a contribuição de intensidades em cada nó da árvore, devidamente atenuadas pelo fator de atenuação que leva em conta a distância percorrida entre o nó e o seu nó-pai. Se o raio primário não interceptar nenhuma superfície da cena, atribui-se ao *pixel* a "cor de fundo". As componentes ambiente e difusa em cada ponto podem ser computadas pelas Eqs. 8.6 e 8.4. Para evitar a determinação do raio de reflexão em relação ao vetor $\mathbf{L}$, é comum utilizar Eq. 8.10 para calcular a componente especular. Figura 8.23.(a) esquematiza a relação dos vetores necessários nos cálculos, denotando o raio secundário por $\mathbf{s}$. A intensidade luminosa em cada ponto é a soma das três componentes devidas às interações diretas com as fontes luminosas $\mathbf{L}_j$ e as intensidades oriundas das direções $\mathbf{R}_i$ e $\mathbf{T}_i$ ponderadas pelo coeficiente de reflexão especular e pelo **coeficiente de refração** da superfície, respectivamente. O coeficiente de refração k_t é uma aproximação empírica da refractância de Fresnel apresentada na seção 8.2. Sintetizando, a intensidade luminosa em cada ponto é

dado por

$$\begin{aligned} I(x,y,z) &= I_a(x,y,z) + f_{at}(k_d \sum_{j=1}^{m} I_{d,j}.(\mathbf{L}_j \cdot \mathbf{n}) + k_s \sum_{i=1}^{m} I_{s,j}(\mathbf{n} \cdot \mathbf{H}_j)^n) \\ &\quad + k_s I_{\mathbf{R}_i} + k_t I_{\mathbf{T}_i}, \quad k_a, k_s, k_d, k_t \in [0,1]. \end{aligned} \tag{8.11}$$

Agora só falta mostrar como o computador consegue "achar" a direção do vetor de refração $\mathbf{T}_i$ e a direção do vetor de reflexão $\mathbf{R}_i$ para rastrear um raio. Comparando o par de raios $(\mathbf{R}_i, \mathbf{s})$ da Figura 8.23.(a) com o par de raios $(\mathbf{R}_i, \mathbf{L}_i)$ da Figura 8.14.(b), a relação entre o par é igual se invertermos o sentido do raio $\mathbf{s}$. Portanto, podemos substituir $\mathbf{L}_i$ da Eq. 8.8 por $-\mathbf{s}$ para obter o vetor de reflexão $\mathbf{R}_i$

$$\mathbf{R}_i = \mathbf{s} - 2(\mathbf{s} \cdot \mathbf{n})\mathbf{n}.$$

Para determinar a direção do vetor de refração $\mathbf{T}_i$, vamos supor, sem perda de generalidade, que $\mathbf{s}$ e $\mathbf{n}$ sejam unitários. Definimos um terceiro vetor unitário $\mathbf{M}$, perpendicular a $\mathbf{n}$ de forma que valha a relação

$$\mathbf{s} = \mathbf{n}\cos\theta_i - \mathbf{M}\sin\theta_i.$$

Desta relação derivamos a seguinte expressão para $\mathbf{M}$

$$\mathbf{M} = \frac{(\mathbf{n}\cos\theta_i - \mathbf{s})}{\sin\theta_i}.$$

Com os vetores unitários $\mathbf{M}$ e $\mathbf{n}$, é imediato obter $\mathbf{T}_i$, já que

$$\mathbf{T}_i = \sin\theta_t \mathbf{M} - \cos\theta_t \mathbf{n}.$$

Substituindo Eq. 8.3 na expressão acima, temos

$$\mathbf{T}_i = \frac{n_i}{n_t}\mathbf{s} - (\cos\theta_t - \frac{n_i}{n_t}\cos\theta_i)\mathbf{n}$$

sendo

$$\cos\theta_t = \sqrt{1 - \sin^2\theta_t} = \sqrt{1 - (\frac{n_i}{n_t})^2 \sin^2\theta_i}.$$

Observe na Figura 8.22.(a) que o raio primário que parte do centro de projeção $P_s = (x_s, y_s, z_s, 1)$ em direção $\mathbf{V} = (x_V, y_V, z_V, 0)$ pode interceptar as figuras geométricas da cena em vários pontos. O ponto que interessa para a técnica de traçado de raio seria o primeiro ponto que o raio intercepta no seu trajeto, pois é neste ponto que o curso do seu trajeto pode ser alterado.

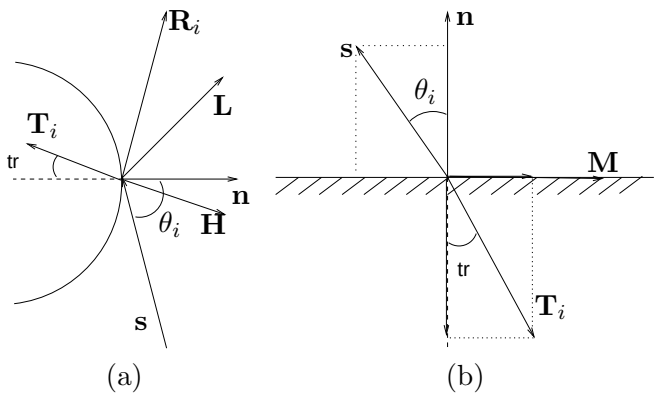


Figura 8.23: Cômputo de intensidades: (a) vetores envolvidos; (b) relação entre vetor incidente e vetor de refração.

Como o computador poderia distinguir este ponto? Uma solução simples é representar parametricamente um raio pela equação (Seção 3.2.1)

$$P(t) = P_s + t\mathbf{V}.$$

O resultado da interseção seria um escalar t. Este escalar reflete diretamente a distância de um ponto $P(t)$ em relação ao ponto C_P. Quanto maior for o valor do escalar, mais distante será o ponto. Como queremos que seja identificado o ponto mais próximo do ponto de onde partiu o raio, basta ordenar os pontos de intersecção obtidos pelos valores escalares e escolher o ponto que tiver o menor t. Por exemplo, na Figura 8.22.(a), foram obtidos dois pontos de intersecção com o raio primário: $P(t_1)$ e $P(t_2)$. Como $t_1 < t_2$, o ponto $P(t_1)$ é o ponto a ser atingido pelo raio e onde o raio mudará o curso do seu trajeto.

Um dos pontos críticos na implementação de traçado de raio é, portanto, a determinação de interseção entre um raio e as superfícies de interesse ao longo da trajetória de rastreamento. A seguir são apresentados alguns procedimentos mais conhecidos.

8.5.1 Raio com Plano

Seja um raio que sai do ponto P_s em direção $\mathbf{V}$ e um plano arbitrário definido pelo vetor normal $\mathbf{n} = (x_n, y_n, z_n, 0)$ e por um ponto $P_0 = (x_0, y_0, z_0, 1)$ sobre o plano. A interseção $P(t_*) = P_s + t_*\mathbf{d}$ deve satisfazer

$$(P(t_*) - P_0) \cdot \mathbf{n} = \mathbf{n} \cdot (P_s + t_*\mathbf{d}) - \mathbf{n} \cdot P_0 = t_*\vec{n} \cdot \mathbf{V} - \mathbf{n} \cdot (P_s - P_0) = 0.$$

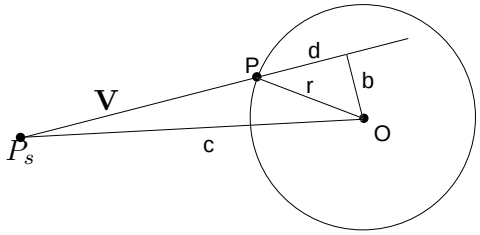


Figura 8.24: Interseção raio e esfera.

Segue-se que

$$t_* \mathbf{n} \cdot \mathbf{V} = \mathbf{n} \cdot (P_s - P_0)$$

e

$$t_* = \frac{\mathbf{n} \cdot (P_s - P_0)}{\mathbf{n} \cdot \mathbf{V}}.$$

Portanto, para $\mathbf{n} \cdot \mathbf{V} \neq 0$ (raio não paralelo ao plano), o raio intercepta com o plano no ponto $P(t_*) = P_s + t_* \mathbf{V}$ quando $t_* > 0$. Observe a semelhança entre a equação e Eq. 6.3. Se for um "plano limitado", como uma face retangular, é necessário verificar adicionalmente se $P(t_*)$ pertence à face retangular.

8.5.2 Raio com Esfera

Dada uma esfera de raio r centrado em O, então o ponto P da sua interseção com um raio $P(t) = P_s + t\mathbf{V}$ é (Figura 8.24)

$$P = P_s + (v - d)\mathbf{V},$$

onde $v = \overrightarrow{P_sO} \cdot \mathbf{V}$ e $d = \sqrt{r^2 - ((\overrightarrow{P_sO} \cdot \overrightarrow{P_sO}) - v^2)}$, se $r^2 - ((\overrightarrow{P_sO} \cdot \overrightarrow{P_sO}) - v^2) > 0$.

8.5.3 Raio com Superfície Implícita

Dada uma superfície implícita em forma de $f(x, y, z) = 0$. O problema se reduz a determinar as raizes da equação $f(x_s + tx_V, y_s + ty_V, z_s + tz_V) = 0$.

Capítulo 9

Algoritmos de Visibilidade

Oclusão é um dos fatores que contribuem a nossa percepção de profundidade. O objetivo deste capítulo é apresentar algumas soluções computacionais através das quais uma máquina consegue "discernir" o que são visíveis dos que não são visíveis em relação a um ponto de vista. Desta forma, ela consegue produzir imagens com oclusão, aproximando-se ainda mais da nossa percepção. Após a leitura deste capítulo, você deve ser capaz de

- explicar o papel de algoritmos de visibilidade em um modelo de iluminação local.
- explicar as coerências utilizadas para aumentar a eficiência dos algoritmos de visibilidade.
- utilizar a técnica de *backface culling* e *bounding box*.
- diferenciar algoritmos de linhas escondidas e de superfícies escondidas.
- aplicar algoritmo de pintor.
- aplicar algoritmo de *z-buffer*.
- construir e percorrer a árvore BSP para ordenar as facetas em relação a um observador.

Conforme já comentamos na seção 5.2, o sistema de visão humana faz uso de vários recursos para melhorar a sua percepção de profundidade. Um destes recursos é a **oclusão** de objetos, isto é, objetos mais distantes em relação a um observador ao longo de um raio de visão são escondidos pelos objetos opacos mais próximos, como ilustra Figura 9.1.(a). Nesta figura, a

superfície opaca S_A bloqueia totalmente os raios luminosos, impedindo que os raios refletidos pelas superfícies S_B e S_C cheguem no olho/na abertura de uma câmera pela direção $\mathbf{r}$. A cor percebida/captada nesta direção seria a da superfície S_A. Dizemos, então, que a superfície S_A é **visível em relação ao observador/à câmera** e as outras ficam **escondidas**, em inglês *hidden*.

Se compararmos a Figura 9.1.(b) com a Figura 8.22.(a), pode-se concluir que a técnica de traçado de raio inclui o algoritmo de visibilidade, já que ela seleciona, dentre todas as intersecções obtidas, o ponto mais próximo do observador/da câmera ao longo de cada raio primário (Seção 8.5). Esta forma de determinação do ponto visível, a nível de resolução do espaço de imagem (Seção 5.3), é conhecida como **ray-casting**. A sua complexidade é Nn, onde N é o número de *pixels* e n é a quantidade de formas geométricas existentes na cena. Se uma cena for "densamente povoada", com n muito grande, ficaria inviável obter a sua imagem na taxa interativa, ou seja, em torno de 30 fps. Sem falar que a cena pode conter figuras geométricas representadas por funções de grau muito elevado cujas intersecções com raios não tenham soluções algébricas como os casos apresentados nas seções 8.5.1 e 8.5.2.

No modelo de iluminação local somente as interações diretas são levadas em conta. Se as radiações refletidas por uma superfície estiverem na direção de visão, elas serão "vistas" independentemente se ao longo do trajeto existam elementos que possam bloquear tais radiações. Portanto, para que o computador gere imagens na forma como ele deveria perceber, é necessário tirar do seu "campo de visão" elementos oclusos. A questão que discutiremos neste capítulo é como "programar" um computador para que ele, antes de aplicar um modelo de iluminação local, diferencie as distâncias das superfícies em relação ao seu centro de projeção, "limitando" a sua "visão" somente para as superfícies mais próximas. Assim, ele conseguiria produzir imagens com correta oclusão. Por exemplo, no caso da Figura 9.1.(b) o computador deveria "ver" somente o ponto P_1 na direção $\mathbf{r}$. O processo de determinação das partes que devem estar no campo de visão de um computador é denominado **algoritmo de visibilidade**.

Pesquisas tem sido conduzidas no sentido de reduzir a complexidade temporal de um algoritmo de visibilidade e implementá-lo diretamente em *hardware* ou *firmware*. Em decorrência disso, pode-se encontrar uma grande variedade de algoritmos de visibilidade na literatura, que essencialmente são classificados em três grandes grupos: **técnicas baseadas em espaço de imagem**, **técnicas baseadas em espaço da câmera/normalizado** e **técnicas mistas**. A determinação de visibilidade por técnicas baseadas no espaço de imagem tem resolução a nível de *pixels*. As técnicas baseadas no

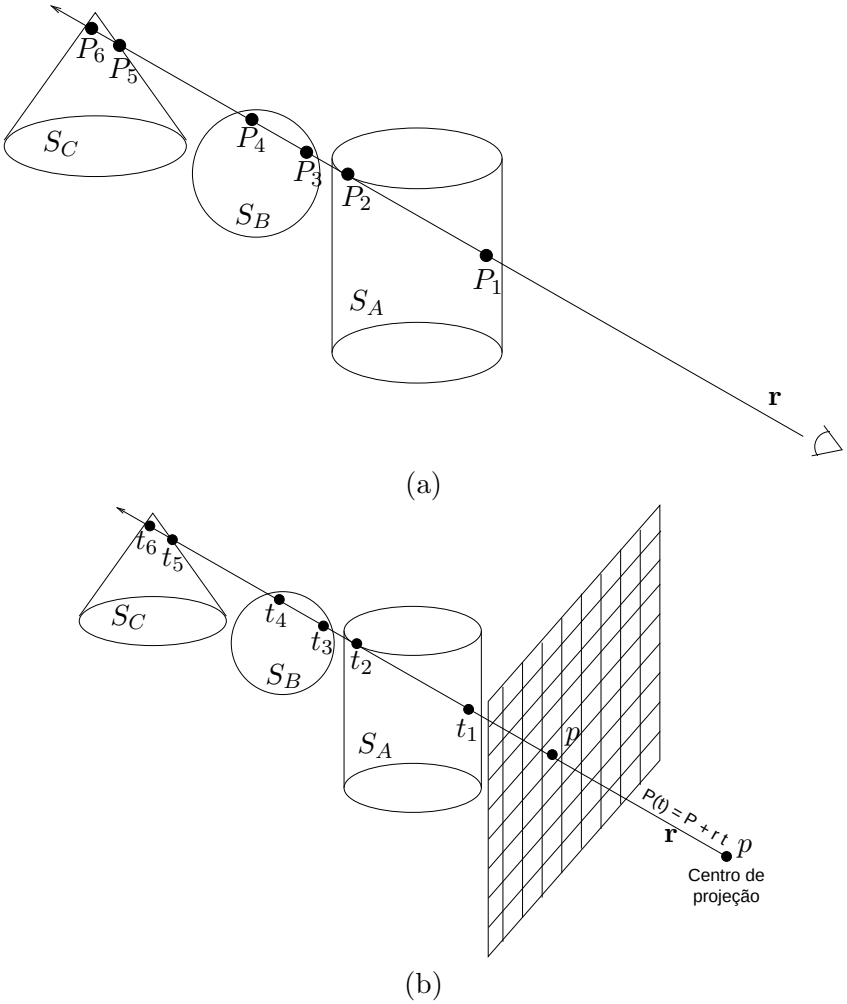


Figura 9.1: Visibilidade: (a) visão humana; (b) visão computacional.

espaço normalizado tem resolução a nível do espaço de representação das figuras geométricas. Uma solução a força bruta seria comparar cada uma das n figuras da cena contra as restantes $(n-1)$ figuras para saber se ela é a mais próxima do observador. A complexidade seria $O(n^2)$. Se $n \ll N$, o procedimento pode ser mais eficiente do que as baseadas no espaço de imagem.

Neste capítulo vamos apresentar dois algoritmo de visibilidade de linhas na seção 9.2. Dois algoritmos mais conhecidos de visibilidade de superfícies são descritos na seção 9.3. Antes, porém, vamos mostrar na seção 9.1 algumas técnicas amplamente utilizadas para aumentar a eficiência no cômputo de visibilidade.

9.1 Pré-processamento

A essência dos algoritmos de visibilidade é muito simples: remover as partes que "não devem ser vistas" pelo computador na produção de uma imagem sintética. O problema, no entanto, é implementar de forma eficiente esta idéia simples. Não provendo a inteligência visual, o computador precisa emular os raios de visão e computar os pontos que eles alcançam, um por um. Para reduzir o tempo de execução do volume de operações envolvidas, várias propriedades geométricas e características dos dispositivos de saída tem sido consideradas para reduzir o tamanho das figuras geométricas sobre as quais são aplicadas efetivamente as operações de maior custo.

Uma propriedade que os algoritmos de visibilidade exploram é a **coerência** dos atributos usando o fato de que sempre existe um escopo dentro do qual a variação dos atributos ocorre de forma bastante suave. Neste caso, ao invés de determinar os atributos "a partir do nada" em cada ponto da cena, podemos utilizar técnicas recorrentes para determiná-los.

Sutherland et al. identificaram em 1974 uma série de coerências que podem ser exploradas nos algoritmos de visibilidade:

1. Coerência de objetos: dois objetos disjuntos não podem apresentar intersecções a nível de faces, arestas ou pontos.

2. Coerência de face: as propriedades gráficas de uma face variam de forma suave. Transições bruscas são normalmente interpretadas como fronteira de duas faces.

3. Coerência de arestas: a transição da parte visível para a parte invisível de uma aresta, ou vice-versa, só pode ocorrer nos pontos de intersecção.

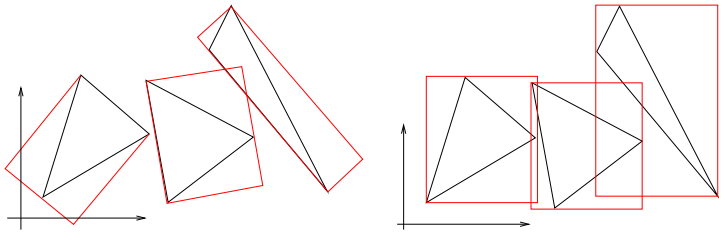


Figura 9.2: Caixa limitante: (a) justa; (b) paralela aos eixos.

4. Coerência geométrica: o segmento de intersecção entre duas faces planas pode ser determinado pelos dois pontos de intersecção.

5. Coerência das linhas de varredura: a variação das propriedades geométricas e físicas entre duas linhas de varredura é pequena.

6. Coerência na área de cobertura: um conjunto de *pixels* adjacentes é usualmente coberto por uma face.

7. Coerência na profundidade: os valores de profundidade das amostras adjacentes de uma mesma face variam pouco e os valores de profundidade das amostras de faces distintas usualmente são valores distintos.

8. Coerência nos quadros: dois quadros consecutivos de uma animação diferem muito pouco um do outro.

Baseado na coerência de objetos, é comum aplicar a técnica de **caixa limitante** (*bounding box*) para separar objetos trivialmente disjuntos de forma a reduzir a quantidade de pares de figuras geométricas a serem testadas. Há várias propostas para determinação de uma caixa limitante justa para um objeto (Figura 9.2.(a)), porém a mais difundida é a menor caixa com os planos paralelos aos eixos do sistema de referência capaz de envolver totalmente o objeto, como as caixas em vermelho na Figura 9.2.(b). Observe que neste segundo caso os resultados são mais conservadores, no sentido de que menos disjunções são identificadas, em troca de um procedimento mais simples.

Transformar as coordenadas da cena para o espaço normalizado pode simplificar as computações. Vimos na seção 5.4.3 que no espaço normalizado o raio de visão fica paralelo ao eixo n tanto em uma projeção paralela quanto em uma projeção perspectiva. Isso reduz comparações entre as distâncias dos pontos em relação ao observador em comparações entre

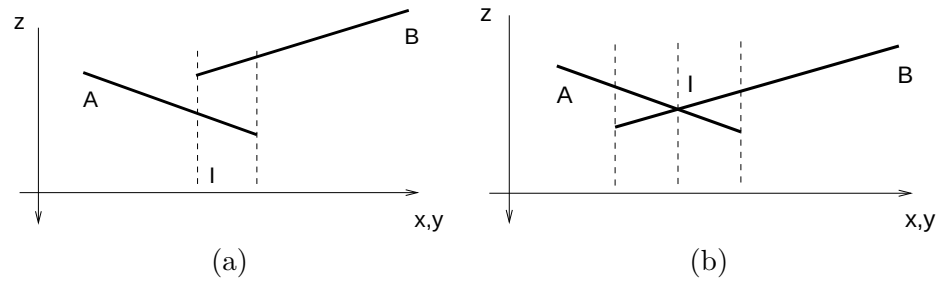


Figura 9.3: Descontinuidade de profundidade: (a) em sobreposições; (b) em intersecções.

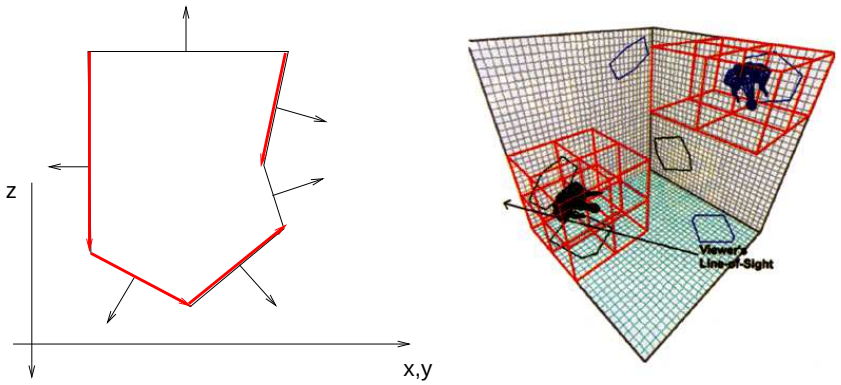
os valores das respectivas coordenadas n. A coerência de profundidade é tipicamente aplicada para reduzir a quantidade de comparações dos valores de profundidade. Observe na Figura 9.3 que se uma figura geométrica é classificada como a "mais próxima" do observador, ela retém esta classificação até ser sobreposta por uma outra ou até interceptar uma outra figura. Portanto, se pré-computarmos os pontos de transição entre as figuras geométricas, só será necessário efetuar um conjunto de comparações em cada extensão contínua de área.

Outro fato é que as facetas "de costas" para observador não são visíveis. Traduzindo em números este conceito, as facetas com vetores normais formando um ângulo menor que 90^0 em relação ao raio de visão (eixo n no espaço normalizado NDC definido na seção 5.3) não devem ser visíveis, como ilustra Figura 9.4. A verificação do ângulo entre um vetor normal $\mathbf{n} = (n_u, n_v, n_n, 0)$ e um raio de visão $\mathbf{V}$, que é igual a $(0, 0, -1, 0)$ no espaço normalizado, pode ser feita por um simples produto escalar

$$B_f = \mathbf{n} \cdot \mathbf{V} = -n_n. \tag{9.1}$$

Se $B_f > 0$ ou $n_n < 0$, a faceta está "de costas", portanto, não é visível (as facetas em linha preta na Figura 9.4.(a)); senão, ela é potencialmente visível (as facetas em linha vermelha na Figura 9.4.(a)). Isso permite reduzir a quantidade n de facetas a serem processadas pelo algoritmo de visibilidade. Esta técnica de remoção de facetas "de costas" é conhecida como **backface culling**.

Outro recurso que aumenta a eficiência do processamento de visibilidade é organizar previamente os dados em uma estrutura mais elaborada que permita inferir a visibilidade com base na coerência de inclusão. Figura 9.4.(b) ilustra uma cena particionada em uma árvore binária tridimensional através da qual podemos facilmente identificar e remover os octantes, e consequente-



(a) (b)

Figura 9.4: (a) *Backface culling* e (b) subdivisão espacial (Fonte: http://www.gamasutra.com/features/19970801/octree.htm).

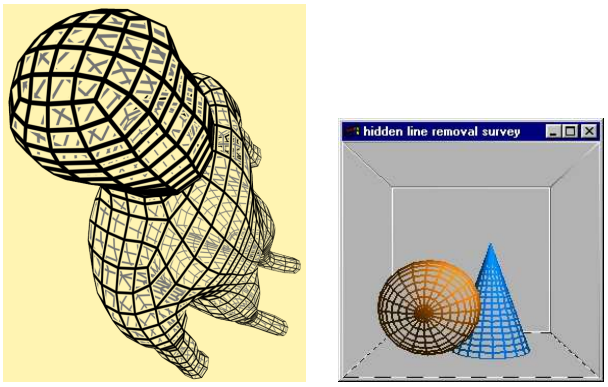
mente as figuras geométricas contidas neles, antes do processamento efetivo de visibilidade.

9.2 Algoritmos de Visibilidade de Linhas

Historicamente, estes algoritmos apareceram antes dos algoritmos de visibilidade de superfícies para remover segmentos que não devem aparecer na saída de um dispositivo vetorial. Figura 9.5 ilustra duas formas de desenhar as linhas não visíveis: (a) fazer uma auréola, ou (b) escondê-las. Estes algoritmos foram amplamente estudados para melhorar a legibilidade das imagens aramadas geradas em arquitetura de saída vetorial, como ilustrada na Figura 1.10.(d). Com a popularidade de imagens de facetas tonalizadas, eles perderam o destaque. No entanto, os problemas estudados para visibilidade de linhas continuam sendo importantes para processamento de informação gráfica.

O primeiro algoritmo de visibilidade de linhas foi desenvolvido pelo Roberts em 1963. Essencialmente, ele reduziu o problema de visibilidade em um problema de programação linear, ao comparar cada segmento P_1P_2 em relação a uma **figura geométrica convexa** definida pelas equações das suas m facetas, com o vetor normal $(n_{x(i)}, n_{y(i)}, n_{z(i)}, 0)$ orientado para o lado externo da figura geométrica. O plano que contém uma faceta é representada pela equação

$$n_{x(i)}x + n_{y(i)}y + n_{z(i)}z + d_i = 0,$$



(a) (b)

Figura 9.5: Linhas não visíveis: (a) aureolares e (b) escondidas (Fonte: http://www.sgi.com/products/software/opengl/examples/glut/advanced/).

e a figura geométrica $P = (x, y, z, 1)$, como um volume delimitado pelos m planos que satisfaz o seguinte sistema de inequações

$$\begin{bmatrix} n_{x(1)} & n_{y(1)} & n_{z(1)} & d_1 \\ n_{x(2)} & n_{y(2)} & n_{z(2)} & d_2 \\ n_{x(3)} & n_{y(3)} & n_{z(3)} & d_3 \\ \cdots & \cdots & \cdots & \cdots \\ n_{x(m)} & n_{y(m)} & n_{z(m)} & d_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

A matriz formada pelos coeficientes dos planos é denominada **matriz de volume**, denotada por $[VT]$.

Para reduzir o número de operações, Roberts introduziu a técnica de *backface-culling* (Eq. 9.1), amplamente utilizado até hoje em algoritmos de visibilidade de superfícies, para identificar os planos em relação ao qual o observador está "de costas". As linhas comuns de duas facetas contidas em planos escondidos são classificadas como escondidas. Elas são, então, removidas.

Em seguida, o restante das linhas é testado individualmente contra todos os volumes existentes na cena para verificar se cada linha é oclusa por algum dos volumes. Por simplicidade, vamos considerar apenas um segmento P_1P_2 e um volume $[VT]$.

A partir de um ponto $P(t) = v$ do segmento P_1P_2 dado por

$$P(t) = P_1 + (P_2 - P_1)t \leftrightarrow P(t) = P_1 + \mathbf{d}t \quad t \in [0, 1]$$

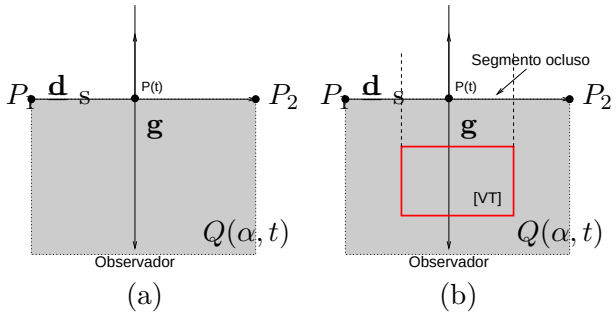


Figura 9.6: Algoritmo de Roberts: (a) plano de visão; (b) oclusão.

definem-se segmentos na direção de projeção $\mathbf{g}$, formando um semi-plano (Figura 9.6.(a))

$$Q(\alpha, t) = P(t) + \mathbf{g}\alpha = P_1 + \mathbf{d}t + \mathbf{g}\alpha, \alpha \geq 0.$$

Como os pontos interiores do volume $[VT]$ são oclusos, o problema pode ser reduzido em determinação de pontos do plano $Q(\alpha, t)$ que satisfaçam o sistema de inequações

$$\begin{aligned} Q(\alpha, t)[VT] &= (P_1 + \mathbf{d}t + \mathbf{g}\alpha)[VT] \\ &= P_1 \cdot [VT] + \mathbf{d}t \cdot [VT] + \mathbf{g}\alpha \cdot [VT], \quad \alpha \geq 0, t \in [0, 1]. \end{aligned}$$

Dentre os possíveis valores de t computados, procura-se pelo mínimo t_{minmax} dos valores máximos e o máximo t_{maxmin} dos valores mínimos que satisfazem as inequações. O segmento ocluso é o contido no intervalo $[t_{maxmin}, t_{minmax}]$. Observe que este é um clássico problema de programação linear.

Posteriormente, outros algoritmos mais simples e genéricos foram propostos, como o algoritmo de Appel em 1967 que explora a coerência de aresta e reduz o problema de visibilidade em intersecção entre arestas e facetas "de frente" para observador. O seu algoritmo não se limita para volumes convexos. Ele introduziu o conceito de **invisibilidade quantitativa** que é um contador para contar o número de vezes que um segmento de aresta passou por trás de uma faceta "de frente". Na Figura 9.7.(b) a borda destas facetas "de frente" foram destacadas com linha vermelha e as intersecções do segmento P_1P_2 com estas facetas destacadas com pontos pretos. Quando um segmento tem a invisibilidade quantitativa igual a zero, ele é visível; do contrário, ele fica invisível. Um pré-processamento é necessário para determinar o contorno das facetas "de frente". A técnica de *backface-culling* pode ser aplicada. Figura 9.7.(c) mostra os segmentos classificados com "1" em linha tracejada.

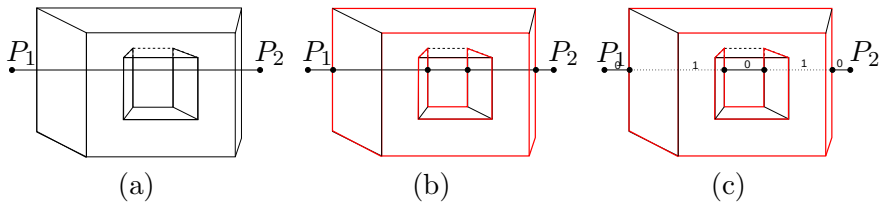


Figura 9.7: Algoritmo de Appel: (a) segmento e $[VT]$; (b) contorno das faces "de frente" do [VT]; (c) invisibilidade quantitativa.

9.3 Algoritmos de Visibilidade de Superfícies

Os algoritmos de visibilidade de superfícies determinam as áreas de uma superfície que devem ser visíveis a partir de um ponto de vista específico. Na seção 8.5 mostramos como o computador pode, através do rastreamento do raio de visão pelo princípio de óptica geométrica, resolver o problema de oclusão. O procedimento ocorre no espaço da câmera, embora a amostragem dos raios primários seja feita a nível de resolução do espaço de imagem. Nesta seção vamos apresentar mais dois algoritmos: dois baseados no espaço de imagem e um, híbrido.

Vale observar que dos três algoritmos a serem apresentados, dois deles, o algoritmo de pintor e o *z-buffer* que desenham polígono por polígono, podem ser adaptados para gerar efeitos de linhas escondidas em dispositivos de saída *raster*. Basta escrevermos na memória de exibição os *pixels* correspondentes ao interior de cada polígono com a cor do fundo. Desta maneira, os polígonos próximos do observador escondem naturalmente as linhas mais distantes.

9.3.1 Algoritmo de Pintor

O nome deste algoritmo se deve à sua analogia à técnica utilizada pelos pintores para pintar uma tela a óleo. Os pintores começam com um cenário distante e o cobre parcialmente com pinceladas para pintar objetos mais próximos. Mentalmente, os pintores ordenam a priori a sequência dos objetos a serem pintados, como ilustra a sequência de desenho na Figura 9.8. Desta forma, objetos mais próximos sobrepõem os mais distantes, solucionando o problema de visibilidade a custo de mais pinceladas em uma mesma área.

Para realizar isso no "computador", Newell, Newell e Sancha propuseram em 1972 um procedimento envolvendo três passos:

1. ordenar, de forma decrescente, os polígonos pelas distâncias, ou pro-

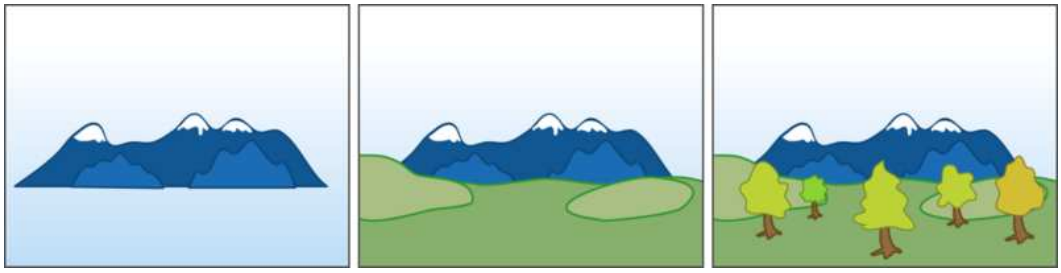


Figura 9.8: Técnica de pintores: as montanhas distantes foram pintadas em primeiro lugar, o gramado mais próximo foi adicionado e, finalmente, as árvores foram acrescidas (Fonte: http://en.wikipedia.org/wiki/Painter%27s_algorithm.

fundidades, em relação ao observador;

2. resolver as ambiguidades na ordenação, dividindo os polígonos; e
3. desenhar na memória de exibição os polígonos, dos mais distantes para os mais próximos, sendo os valores gravados anteriormente sempre sobrescritos pelos novos valores.

Como comentamos na seção 9.1, podemos tirar proveito se transformarmos as figuras geométricas para o espaço da câmera antes da ordenação, pois nestes espaços a coordenada n dos pontos corresponde à distância em relação ao observador.

Como a ordenação ocorre no espaço da câmera/normalizado e a solução de oclusão acontece no espaço de imagem, através da sobreposição de desenhos, este algoritmo pertence à classe de algoritmos de técnica híbrida.

Ordenação pela Profundidade

Figura 9.9 ilustra alguns casos típicos de ambuiguidade na ordenação pela profundidade. Na Figura 9.9.(a) o retângulo vermelho está simultaneamente acima e embaixo do polígono verde. E na Figura 9.9.(b) o retângulo azul está, ao mesmo tempo, acima do retângulo vermelho e embaixo do retângulo verde. Este, por sua vez, está embaixo do vermelho. Para resolver tais ambiguidades, subdivide-se os polígonos, como ilustram Figuras 9.9.(c) e (d).

Como o computador pode "ver" tais ambiguidades? Dado um polígono plano P, Newell et al. apresentaram uma sequência de 6 testes na ordem

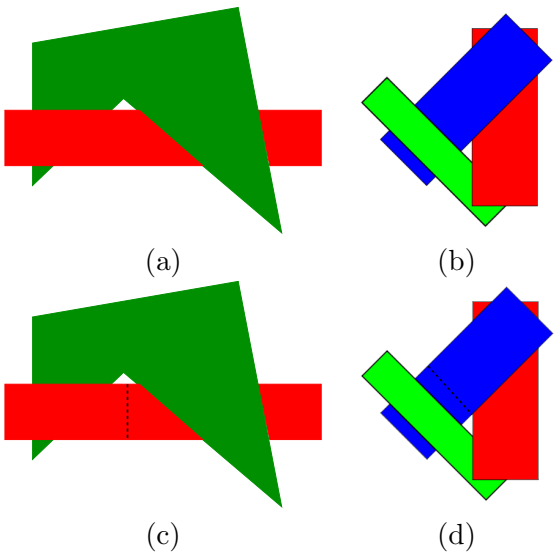


Figura 9.9: Ambiguidades na ordenação: (a) interpenetração; (b) sobreposição cíclica; (c) e (d) soluções.

crescente de complexidade computacional para verificar a ordenação de P em relação aos outros polígonos Q já desenhados na memória de exibição. Para cada par P e Q, verifica-se se

1. são disjuntos na extensão da coordenada z? (Figura 9.10.(a))
2. são disjuntos na extensão da coordenada x? (Figura 9.10.(b))
3. são disjuntos na extensão da coordenada y? (Figura 9.10.(c))
4. P está inteiramente contido no "lado" do plano Q oposto ao lado onde está localizado o observador? (Figura 9.10.(d))
5. Q e o observador estão inteiramente contidos no mesmo lado do plano P? (Figura 9.9.(a))
6. as projeções de P e Q se sobrepõem no plano de projeção?

Se todos os testes falharem, repetem-se os testes 4 e 5, trocando P por Q, e vice-versa. Caso falharem também estes testes, divide-se um polígono pelo outro e repete-se os testes com os novos polígonos. Do contrário, se um dos testes na sequência passar, "pinta-se" P na memória.

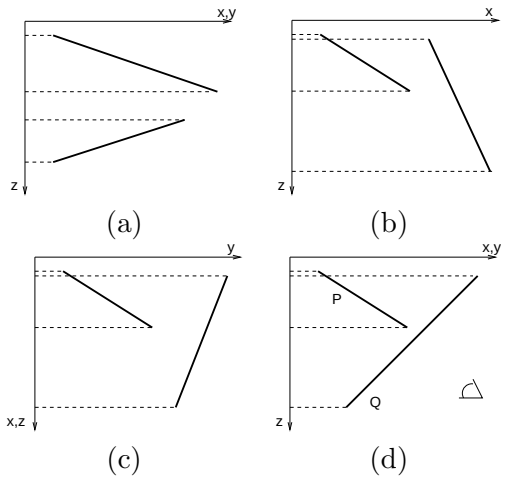


Figura 9.10: Testes de ambiguidades: (a) teste (1); (b) teste (2); (c) teste (3); (d) teste (4).

Para fazer os testes (1)–(3), simples comparações entre os valores mínimos e máximos de cada coordenada são suficientes. E para testes(4) e (5), basta utilizarmos a representação implícita do plano que contém um polígono

$$f(x, y, z) = n_x x + n_y y + n_z z + d$$

e avaliar o sinal desta função aplicada em cada vértice do outro polígono. Em relação ao teste (6), é necessário avaliar a continência das arestas de um polígono no outro. Para isso, podemos utilizar o algoritmo de recorte de Cyrus-Beck apresentado na seção 6.2.2.

Ordenação pela BSP

A ordenação proposta por Newell et al. é dependente do observador. Para cada alteração da posição do observador, os polígonos precisam ser reordenados, mesmo que a cena não tenha sofrido nenhuma modificação. Para reaproveitar os resultados de um pré-processamento, foram propostas várias estruturas de dados que consigam organizar as facetas de forma independente do observador. A **árvore de partição espacial binária** (*binary space-partitioning tree*), proposta por Fuchs, Kedem e Naylor em 1980, é uma estrutura que atende este requisito. Ela é extremamente eficiente para um cenário estático. O algoritmo se baseia em uma simples observação de que se há um plano que separa um conjunto de facetas em dois grupos, o grupo que estiver no lado do plano onde está o observador não pode ser

ocluso pelo grupo de facetas que estiver no outro lado do plano. Assim, a idéia é subdividir recursivamento as regiões até que cada uma contenha somente uma unidade dos elementos.

Para construir uma árvore binária de polígonos, podemos iniciar com um plano qualquer, por exemplo o plano que contém a faceta "A" na Figura 9.11.(a). Este plano será denominado o plano-raíz. Ele dividirá o espaço em dois semi-espaços, um para onde o vetor normal do plano aponta (frente) e o outro é o seu complemento Figura 9.11.(b). Note que a faceta "E" foi subdividida em "E1" e "E2" pelo plano de partição. Isso assegura que não haja ambiguidade na classificação de uma faceta em relação às regiões particionadas. A subdivisão continua recursivamente em cada novo semi-espaço enquanto houver mais de um elemento no semi-espaço, conforme ilustram Figuras 9.11.(c)-(g). Observe nas figuras a correspondência entre as partições espaciais e os acréscimos de nós na estrutura de dados.

Observe que a árvore foi construída independentemente da posição do observador. A ordenação das facetas é feita em tempo-real através de um percurso pela árvore com base na posição do observador especificada. Na Figura 9.12.(a) o observador está na região que contém a faceta "E1". Em relação ao nó "A", o observador está no semi-espaço "atrás", portanto o computador deve desenhar na sequência: o conjunto de facetas da subárvore "frente", "A" e o conjunto de facetas na subárvore "atrás". Ao processar o nó "G", verifica-se que o observador também está atrás da plano, será então desenhado na sequência: subárvore "frente", "G" e subárvore "atrás", e assim por adiante. O resultado da sequência completa de desenho é {F, E2, G, A, B, C, H, D, E}. Para a posição do observador mostrada na Figura 9.12.(b), obteremos o percurso {E2, F, G, A, D, E1, H, B, C}. O pseudo-código do algoritmo de pintor com uso da estrutura BSP é apresentado na função BSP_percurso.

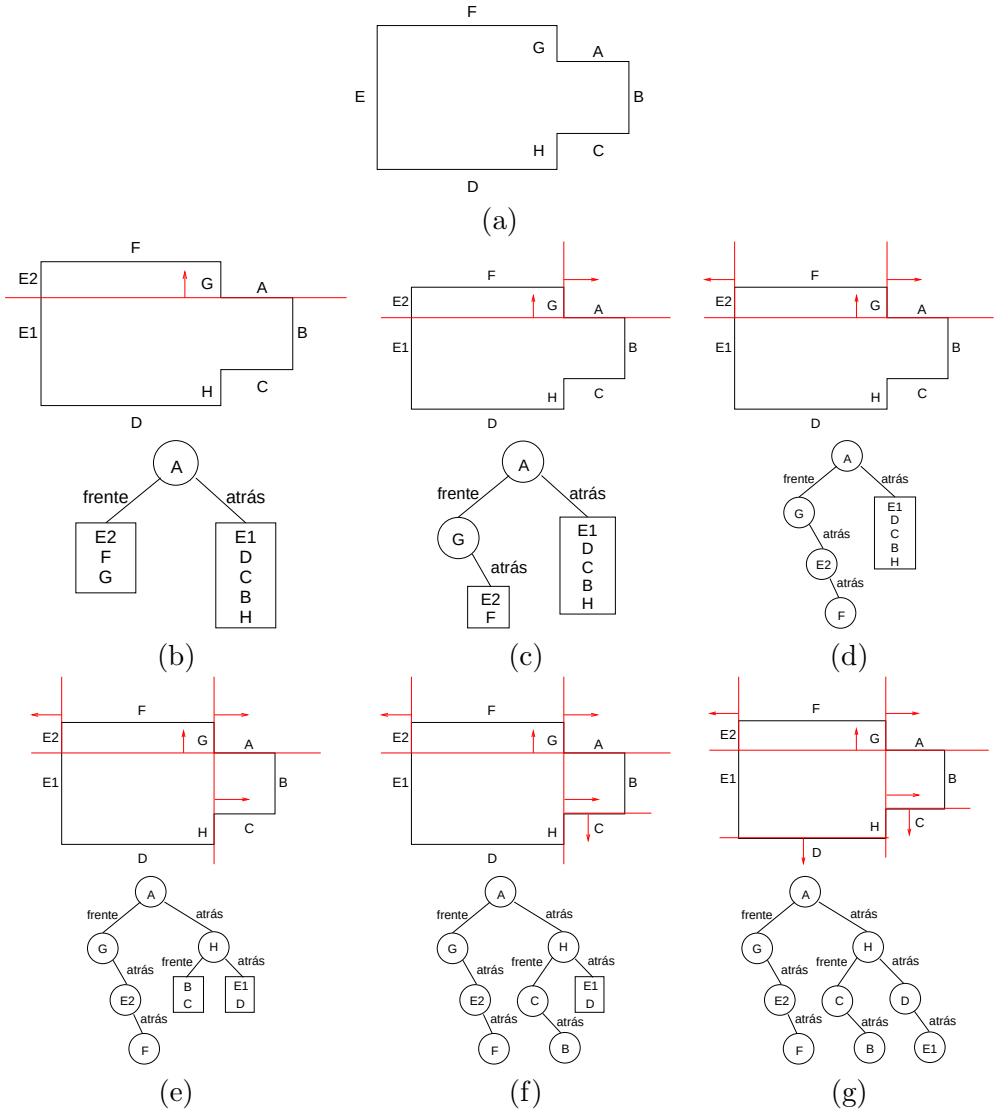


Figura 9.11: Partição do espaço e BSP: (a) cena; (b) primeira partição (c)-(d) partição do primeiro semi-espaço; (e)-(g) partição do outro semi-espaço.

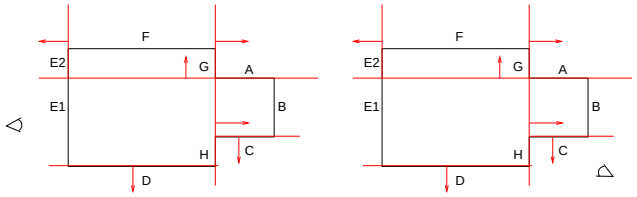


Figura 9.12: Testes de ambiguidades: (a) teste (1); (b) teste (2); (c) teste (3); (d) teste (4).

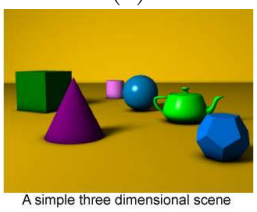
Input: BSP, observador
Output: desenho de polígonos
if *nó não é uma folha* **then**
  lado = Lado do "nó" onde fica o observador;
  if *BSP_percurso (no→frente);*
  desenha o polígono no nó ;
  BSP_percurso (no→atrás);
  then lado = "atrás"
    BSP_percurso (no→atrás);
    desenha o polígono no nó ;
    BSP_percurso (no→frente);
else
  desenha o polígono na folha ;
end

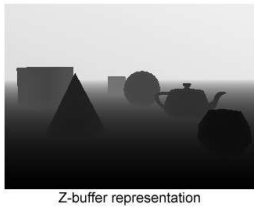
Algoritmo 5: Função BSP_percurso: algoritmo de desenho na ordem sequenciada pela profundidade.

9.3.2 Algoritmo de Z-buffer

O algoritmo de visibilidade de superfícies mais difundido entre as placas de vídeo é o algoritmo de *z-buffer* proposto por Edwin Catmull em 1975. Ele consiste em colorir cada *pixel* com a cor da faceta que estiver mais próximo do observador. Após transformarmos as figuras geométricas para o espaço da câmera/normalizado, a avaliação da profundidade de cada ponto em relação ao observador pode ser diretamente pelos valores da coordenada n. Portanto, é o algoritmo mais simples para ser implementado tanto em *software* quanto em *hardware*. Por outro lado, é o algoritmo que tem maior custo de memória. Além da usual memória de exibição, é necessária uma

(a)





(b)

Figura 9.13: Conteúdo da: (a) memória de exibição; (b) memória de profundidade (Fonte: http://en.wikipedia.org/wiki/Z-buffer).

área adicional, conhecida como **memória de profundidade** ou *z-buffer* em inglês, para armazenar o valor de profundidade do ponto desenhado em cada *pixel*. Antes de "desenhar" a cor de uma nova faceta em um *pixel*, compara-se o valor de profundidade do novo "ponto" com o valor de profundidade armazenado em *z-buffer*. Caso o valor for menor, a cor é substituída pela cor da nova faceta e o valor no *z-buffer* atualizado pelo valor de profundidade desse novo "ponto". Figura 9.13 visualiza o conteúdo das duas memórias de uma mesma cena. Observe na Figura 9.13.(b) que os *pixels* contendo pontos mais próximos do observador tem um tom de cinza mais escuro (com coordenada n menor).

O seguinte pseudo-código sintetiza o procedimento de desenho pela técnica de z-buffer.

Input: cena,memória de profundidade,memória de exibição
Output: desenho de facetas
Transformar a cena para o espaço da câmera/normalizado ;
Inicializa cada entrada da memória de profundidade com ∞ ;
Inicializa cada entrada da memória de exibição com a cor de fundo ;
foreach *polígono* **do**
  foreach *pixel (i,j) em que a faceta é projetada* **do**
    if *valor em (i,j) do z-buffer ¿ valor n do ponto projetado* **then**
      escrever a cor da faceta na posição (i,j) da memória de exibição;
      escrever o valor n na posição (i,j) da memória de profundidade;
    end
  end
end

Algoritmo 6: Função z_buffer: algoritmo de desenho pela técnica de *z-buffer*.

A grande vantagem do algoritmo de *z-buffer* é que, a custo de uma memória maior, consegue-se evitar ordenação no espaço da câmera. Adicionalmente, é possível explorar a coerência de profundidade para determinar recorrentemente os valores de profundidade da faceta projetada. Por isso, o algoritmo de *z-buffer* é muito utilizado em conjunto com o algoritmo de varredura, descrito na seção 10.2.3, para determinar corretamente a faceta que deve ser visível em cada *pixel*.

Considere que a equação do plano que contém uma faceta poligonal seja representada implicitamente pela expressão

$$n_x x + n_y y + n_z z + d = 0.$$

Segue-se que a coordenada z de cada amostra (x_k, y_k) é expressa por

$$z_k = \frac{-d - n_x x_k - n_y y_k}{n_z}. \tag{9.2}$$

A coordenada z do seu *pixel* vizinho $(x_k + 1, y_k)$ pode ser obtida de forma recorrente a partir de z_k

$$z_{k+1} = \frac{-d - n_x(x_k + 1) - n_y y_k}{n_z} = z_k - \frac{n_x}{n_z}.$$

E como podemos obter a coordenada z ao passar de uma linha de varredura para a outra? Basta substituir as coordenadas da amosta $(x, y_k + 1)$ na Eq. 9.2.

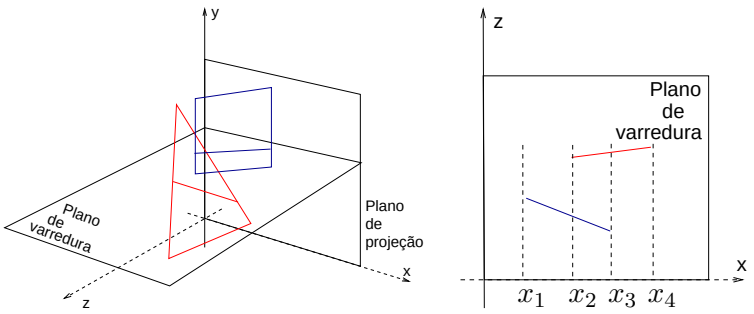


Figura 9.14: Scanline: (a) visão 3D; (b) visão 2D.

9.3.3 Algoritmo de Scanline com Z-buffer

O algoritmo de *scanline* com *z-buffer* pode ser considerado um aprimoramento do algoritmo de *scanline*. Veremos na seção 10.2.3 que a conversão da lista de exibição de figuras geométricas 2D (seção 1.3) em imagens discretas pode ocorrer, linha por linha, na ordem como é feita a varredura dos dispositivos de saída *raster*. Em cada linha, os pontos extremos dos intervalos dos polígonos que interceptam com ela são ordenados e os intervalos entre eles são desenhados alternadamente na memória de exibição constituída por uma única linha de varredura. Esta forma de desenho foi muito utilizada antes da memória ser economicamente acessível.

Ao desenhar, por linha, as figuras geométricas 3D disponíveis na lista de exibição, o problema de desenhar a projeção de uma cena 3D (Figura 9.14.(a)) pode ser também reduzido para um problema de desenhar os segmentos (Figura 9.14.(b)), com a diferença de que não se pode mais simplesmente desenhar alternadamente os intervalos entre os pontos extremos. Por exemplo, na Figura 9.14.(b), os pontos extremos dividem a extensão da coordenada x em intervalos $(-\infty, x_1)$, (x_1, x_2), (x_2, x_3), (x_3, x_4), (x_4, ∞). Se desenharmos alternadamente estes intervalos, o intervalo (x_2, x_3) não será desenhado. O resultado será incorreto. Baseado na proposta de Catmull, Myers sugeriu em 1975 alocar mais um vetor para cada linha em processamento. Este vetor é iniciado com valor máximo de profundidade e controla a escrita do conteúdo da memória de exibição com base nas comparações entre o valor de profundidade dos *pixels* e o valor de profundidade das amostras do polígono, como no algoritmo de *z-buffer*.

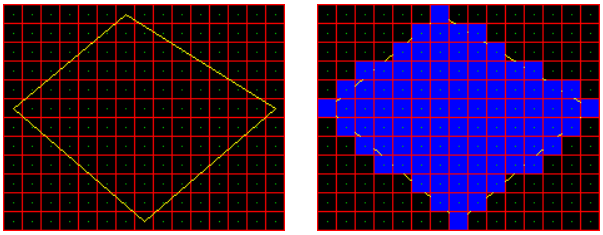
Capítulo 10

Amostragem

O objetivo deste capítulo é mostrar como um computador consegue transformar, com um número mínimo possível de artefatos visuais, a projeção de uma cena 3D em um padrão de arranjo de amostras compatível com o padrão de *pixels* dos dispositivos de saída. Após a leitura deste capítulo, você deve ser capaz de

- diferenciar os algoritmos de rasterização da técnica de traçado de raio;
- distinguir e justificar casos 3D para os quais algoritmos de rasterização conseguem se facilmente adaptar;
- listar as inovações "conceituais" introduzidas pelos algoritmos de rasterização;
- diferenciar o algoritmo de Bresenham e o algoritmo de ponto médio;
- aplicar o algoritmo de ponto médio e *scan-line com z-buffer*;
- distinguir em que espaço ocorrem os processamentos ao longo de um fluxo de imageamento, considerando os diferentes modelos de iluminação, tonalização e algoritmos de visibilidade.
- modelar uma imagem no domínio espacial e no domínio espectral;
- explicar o fenômeno *aliasing* e formas para atenuá-lo.

Na seção 1.3 vimos que, essencialmente, existem duas classes de dispositivos para exibição de imagens: os vetoriais e os *raster*. Um dispositivo



(a) (b)

Figura 10.1: Rasteriação: (a) imagem contínua; (b) imagem discreta. (Fonte: http://freespace.virgin.net/hugo.elias/graphics/x_polysc.htm

de saída vetorial consegue processar uma lista de exibição contendo primitivas gráficas aleatoriamente ordenadas; enquanto um dispositivo de saída *raster* requer que os dados sejam organizados em um arranjo bidimensional de pontos endereçáveis denominados *pixels* (*picture elements*). Assim, para exibir a **imagem contínua** ou **vetorial** de um polígono tonalizado e visível em um dispositivo *raster* precisamos transformá-la em um **conjunto de amostras** (i, j), onde i e j são coordenadas inteiras de cada elemento desse arranjo (Figura 10.1). Este processo particular de **amostragem** é denominado **rasterização**, em inglês *rasterization* ou *scan-conversion*, e ele envolve duas decisões: em qual *pixel* deve-se desenhar (seleção das amostras e truncamento das coordenadas em ponto flutuante destas amostras para coordenadas inteiras) e com qual cor. Veremos na seção 10.2 os principais algoritmos de rasterização.

Além da abordagem geométrica, é útil em sistemas de informação gráfica considerar a discretização de uma imagem sob o ponto de vista espectral. Na abordagem espectral a luminância $I(u, v)$ de uma imagem é representada como uma integral/um somatório de componentes de sinais senoidais e cossenoidais ao longo das coordenadas u e v, como ilustra Figura 10.2. Na seção 10.3 faremos uma breve introdução a este paradigma de representação. Ao "ver" uma imagem no espaço espectral, o computador consegue "saber" a causa das variações abruptas das cores entre *pixels* adjacentes e atenuar uma série de artefatos, aplicando técnicas de Processamento de Sinais. Na seção 10.4 são apresentadas três técnicas práticas.

Antes de apresentarmos algoritmos de conversão de imagens contínuas em imagens discretas, é conveniente introduzirmos alguns conceitos relacionados com imagens discretas na seção 10.1.

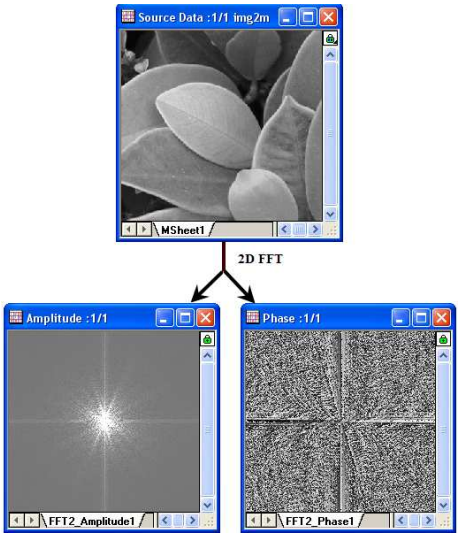


Figura 10.2: Imagem digitalizada (em cima) no espectro de Fourier (embaixo, esquerda) e seu ângulo de fase (embaixo, direita).

10.1 Imagens Discretas

Uma imagem discreta pode ser considerada um reticulado de *pixels*, a cada qual é associada uma luminância capaz de produzir um certo estímulo cromático no olho humano. O *pixel* é a menor unidade endereçável por um par de coordenadas inteiras (i, j). A discretização de uma imagem contínua, com pontos em coordenadas reais, em um espaço discreto, com pontos em coordenadas inteiras, requer adequação de alguns conceitos fundamentais no estudo de espaços reais para o estudo de um espaço de reticulado.

Análogo ao espaço métrico, podemos definir uma função de distância entre dois *pixels* (i, j) e (l, m) no espaço discreto. Entre as medidas de **distância** mais conhecidas citamos (Figura 10.3)

distância Euclideana : $D_E((i,j),(l,m)) = \sqrt{(i-l)^2 + (j-m)^2}$.

distância de quarteirão (*city-block distance*): $D_4((i,j),(l,m)) = |i-l| + |j-m|$.

distância de xadrez (*chessboard distance*): $D_8((i,j),(l,m)) = max\{|i-l|, |j-m|\}$

Em um espaço métrico M, a vizinhança de um ponto p é uma bola aberta $B(p;r) = \{x \in M / d(p,x) < r\}$, onde $d(p,x)$ é a distância euclidiana entre os

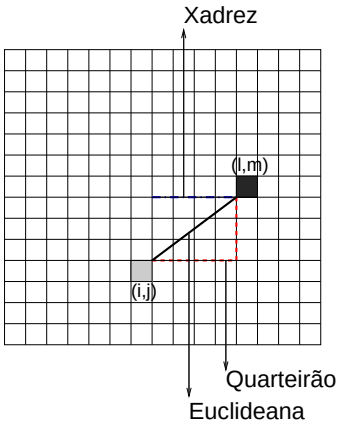
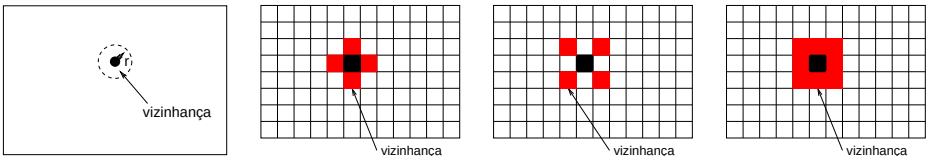


Figura 10.3: Medidas de distância enre dois *pixels* (i, j) e (l, m).



(a) (b) (c) (d)

Figura 10.4: Vizinhança: (a) disco aberto no espaço contínuo; (b) vizinhos-de-4; (c) vizinhos diagonais; (d) vizinhos-de-8.

pontos p e x . No caso de uma imagem bi-dimensional, a vizinhança de um ponto é um disco aberto (Figura 10.4.(a)). Por outro lado, em um espaço discreto, a **vizinhança** de um *pixel* é constituída por, no máximo, 8 *pixels*. Dizemos que dois *pixels* p e q são **vizinhos-de-4** quando eles satisfazem a relação $D_4(p, q) = 1$ (Figura 10.4.(b)). O conjunto de *pixels* vizinhos-de-4 de um *pixel* p é denominado **vizinhança-de-4** de p e representado por $N_4(p)$. Dois *pixels* são chamados **vizinhos-de-8** se eles satisfazem a igualdade $D_8(p, q) = 1$ (Figura 10.4.(c)). O conjunto de *pixels* vizinhos-de-8 associados a um *pixel* p é denominado **vizinhança-de-8** de p e representado por $N_8(p)$. Ele inclui $N_4(p)$ e os restantes vizinhos diagonais $N_D(p)$.

Um espaço métrico é conexo quando se pode passar de um qualquer dos seus pontos para outro por um movimento contínuo, sem sair do espaço. Em outras palavras, para dois pontos quaisquer existe sempre um caminho no espaço entre eles. Analogamente, no espaço discreto, um subconjunto S de *pixels* é um **componente conexo**, se para dois *pixels* quaisquer $p, q \in S$ existe um caminho em S entre eles. Como temos agora três tipos

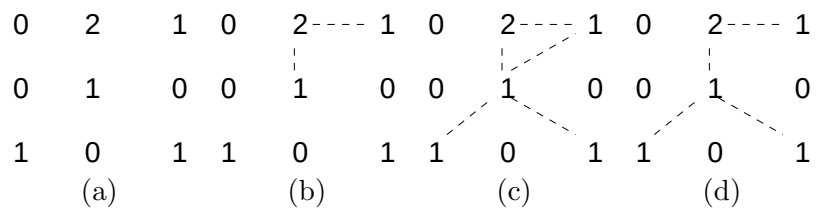


Figura 10.5: Conectividade: (a) subespaço S com valores {1,2}; (b) conectado-de-4; (c) conectado-de-8; (d) conectado-de-m.

de vizinhança, $N_4(p)$, $N_D(p)$ e $N_8(p)$, distinguem-se três tipos de caminhos, dependendo de como os *pixels* vizinhos do caminho estão conectados. Tipicamente, um subespaço S é caracterizado pelos valores dos *pixels*, como por exemplo a luminância. Dizemos, então que dois *pixels* p e q são

conectados-de-4 : se p e q possuem valores do conjunto V e $q \in N_4(p)$ (Figura 10.5.(b)).

conectados-de-8 : se p e q possuem valores do conjunto V e $q \in N_8(p)$ (Figura 10.5.(c)).

conectados-de-m : se p e q possuem valores do conjunto V, e $q \in N_4(p)$, ou então $q \in N_D(p)$ e $N_4(p) \cup N_4(q) = \emptyset$ (Figura 10.5.(d)).

Vale ressaltar aqui que um componente conexo sob a conectividade-de-8 pode não o ser pela conectividade-de-4, como ilustra a Figura 10.6. Observe que os *pixels* em preto, que correspondem a uma circunferência rasterizada, são conectados-de-8, mas não são conectados-de-4. Se a conectividade-de-4 tivesse sido escolhida para avaliar a conexidade da imagem formada pelos *pixels* pretos, teríamos concluído que ela é uma figura desconexa. Isso não corresponderia ao fato real, pois a circunferência é uma figura conexa. Assim, a escolha do tipo de conectividade a ser utilizado para avaliar a conexidade de uma figura discreta deve ser muito criteriosa para evitar conclusões falsas.

Dois *pixels* vizinhos são **adjacentes** se eles forem conectados. Como no espaço discreto são definidos três tipos de conectividade, distinguimos três tipos de adjacência: adjacência-de-4, adjacência-de-8 e adjacência-de-m. Dois componentes S_1 e S_2 de uma imagem são adjacentes, se eles tiverem *pixels* adjacentes.

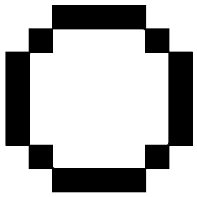


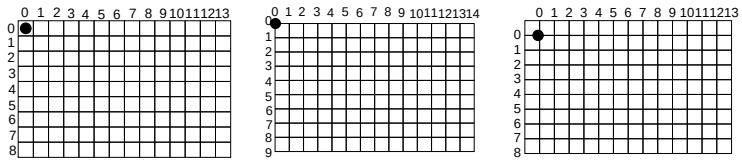
Figura 10.6: Circunferência.

10.2 Rasterização

Os primeiros algoritmos de rasterização foram desenvolvidos com o simples intuito de converter figuras geométricas representadas em um espaço vetorial em um reticulado de *pixels*. Essencialmente, a idéia consiste em "sobrepor" a figura vetorial sobre o reticulado de *pixels* e mapear o centróide de cada *pixel* para um ponto da figura geométrica, procurando preservar a conectividade e a adjacência entre as amostras como mostra Figura 10.1. Quem já não viu um trabalho em ponto-de-cruz ou um desenho em papel quadriculado? O procedimento de conversão é algo que os homens já conheciam há muito tempo. A pergunta que se colocava é como um computador consegue determinar esta sobreposição de forma eficiente? Vale ressaltar que, diferentemente da técnica de traçado de raio que ocorre no espaço de universo (Seção 8.5), os algoritmos de rasterização consideram que as figuras geométricas estejam no espaço de dispositivo DC (Seção 5.4.4). Nesta seção vamos apresentar os principais algoritmos de rasterização pelas suas inovações "conceituais".

10.2.1 Rasterização de Pontos

A conversão pode ser reduzido em um problema de arredondamento de valores reais para inteiros, quando as coordenadas do ponto tiverem a parte fracionária diferente de zero. O algoritmo de arredondamento depende da convenção utilizada para o endereçamento dos *pixels*. Podemos associar as coordenadas inteiras ao centro do *pixel* (Figura 10.7.(a)), ao seu canto superior esquerdo (Figura 10.7.(b)), ao canto inferior esquerdo (Figura 10.7.(c)), ou a qualquer um outro ponto. O importante é ter em mente que se deve optar por uma convenção que assegure maior similaridade entre a figura vetorial original e a figura discretizada, pois ao converter pontos de extensão nula para *pixels* com uma área finita, as coordenadas inteiras correspondem, de fato, a múltiplos de um *pixel* na vertical e na horizontal. Isso pode gerar distorções nas propriedades geométricas como comprimento, área e volume.



(a) (b) (c)

Figura 10.7: Endereçamento de *pixels*: (a) centro; (b) canto esquerdo superior; (c) canto direito inferior.

Um esquema de endereçamento de *pixel* é fixar a origem do referencial no canto inferior esquerdo e referenciar cada "área" de *pixel* pelas coordenadas inteiras associadas ao seu canto inferior esquerdo. Este esquema é equivalente ao esquema esboçado na Figura 10.7.(b). Ele evita mapeamento de valores fracionários de coordenadas reais em valores inteiros, o que propicia uma representação mais exata de figuras geométricas vetoriais. Adicionalmente, ele simplifica o controle de consistência, em termos de preservação das propriedades geométricas, durante o processamento de algoritmos de rasterização. Por exemplo, ao rasterizar um segmento de $(1, 1)$ a $(5, 5)$, podemos utilizar simplesmente as condições de $x \in [1, 5)$ e $y \in [1, 5)$ para assegurar que o segmento rasterizado tenha a extensão de 4 unidades de medida.

10.2.2 Rasterização de Segmentos

Na rasterização de curvas, além do problema de arrendodamento, deve-se preocupar com a posição relativa dos *pixels* adjacentes para que o resultado seja uma boa aproximação da forma original da figura geométrica. Para aumentar a eficiência, os algoritmos mais conhecidos são **recorrentes**: dado o primeiro ponto, os outros pontos subsequentes são obtidos em função do ponto anterior dando passos de um *pixel*, para evitar a geração de um conjunto de pontos de mesmos atributos gráficos (por exemplo, mesma cor) mapeado em um mesmo *pixel*.

Para digitalizar uma reta dada pela expressão

$$y = \frac{\Delta y}{\Delta x} x + b$$

dois algoritmos mais conhecidos são:

- algoritmo de **analisador do diferencial digital** (DDA – *digital differential analyzer*) e

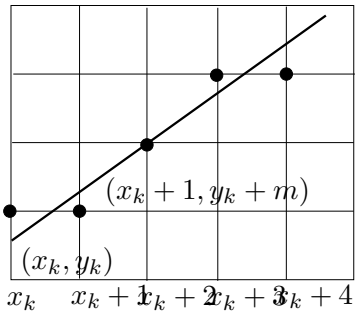


Figura 10.8: Algoritmo DDA.

- algoritmo de **Bresenham**.

Na prática, o algoritmo mais utilizado é uma variante do algoritmo de Bresenham conhecido como **algoritmo de ponto médio**.

Algoritmo DDA

No DDA, como o nome já disse, o incremento nas coordenadas de um ponto (x_k, y_k) para obter o ponto subsequente (x_{k+1}, y_{k+1}) é dado em função do diferencial

$$m = \frac{\Delta y}{\Delta x} = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}.$$

Para $m \leq 1$, as coordenadas x crescem mais rapidamente que as coordenadas y. Portanto, a amostragem é feita incrementando unitariamente na direção x. Com isso,

$$y_{k+1} = y_k + m.$$

Se $m > 1$ faz-se incremento unitário na direção y. Neste caso temos

$$x_{k+1} = x_k + \frac{1}{m}.$$

Algoritmo de Bresenham

O algoritmo de Bresenham, desenvolvido em 1965, consegue rasterizar uma linha somente com **computações inteiras incrementais**. O incremento é condicionado à comparação das "distâncias" entre os *pixels* vizinhos e a reta.

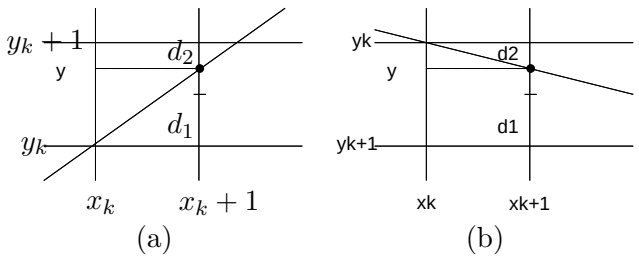


Figura 10.9: Inclinação de reta (a) positiva e (b) negativa.

Se $m \leq 1$, ou seja $|\Delta y| < |\Delta x|$, $\Delta y > 0$ e $\Delta x > 0$, o ponto $(x_{k+1}, y) = (x_k + 1, y)$, subsequente ao ponto (x_k, y_k), deve estar entre (x_{k+1}, y_k) e (x_{k+1}, y_{k+1}) (Figura 10.9.(a)).

Definindo $d_1 = |y - y_k|$ e $d_2 = |(y_k + 1) - y|$, temos

$$\begin{aligned} d_1 - d_2 &= (\frac{\Delta y}{\Delta x}x + b) - y_k - y_{k+1} + (\frac{\Delta y}{\Delta x}x + b) \\ &= 2\frac{\Delta y}{\Delta x}(x_k + 1) - 2y_k + 2b - 1 \end{aligned}$$

Como $\Delta x > 0$, basta analisarmos o sinal da **diferença dos erros**

$$\begin{aligned} p_k &= \Delta x(d_1 - d_2) = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k - \Delta x + 2b\Delta x \\ &= 2\Delta y x_k - 2\Delta x y_k + (2\Delta y + 2b\Delta x - \Delta x). \end{aligned}$$

para concluirmos se devemos incrementar, ou não, a coordenada y, ou seja,

$$\begin{aligned} p_k \geq 0 &\rightarrow y_{k+1} = y_k + 1 \\ p_k < 0 &\rightarrow y_{k+1} = y_k \end{aligned} \tag{10.1}$$

Agora, vamos ver uma forma eficiente para determinar p_k para cada *pixel* k, envolvendo somente somas e subtrações de "constantes", Δx, $2\Delta x$, $2\Delta y$. Fazendo $c = (2\Delta y + 2b\Delta x - \Delta x)$, que é um valor constante, temos

$$p_k = 2\Delta y x_k - 2\Delta x y_k + c$$

e para p_{k+1}, podemos derivar uma expressão recorrente

$$\begin{aligned} p_{k+1} &= 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c \\ &= 2\Delta y(x_k + 1) - 2\Delta x y_{k+1} + 2\Delta x y_k - 2\Delta x y_k + c \\ &= (2\Delta y x_k - 2\Delta x y_k + c) - 2\Delta x(y_{k+1} - y_k) + 2\Delta y \\ &= p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k). \end{aligned} \tag{10.2}$$

Em especial,

$$\begin{aligned} p_0 &= 2\Delta y x_0 - 2\Delta x y_0 + c \\ &= 2\Delta y x_0 - 2\Delta x(\frac{\Delta y}{\Delta x}x_0 + b) + 2\Delta y + 2b\Delta x - \Delta x \\ &= 2\Delta y - \Delta x \ . \end{aligned} \tag{10.3}$$

Pela Eq. 10.1 podemos simplificar ainda a Eq. 10.2, pois

$$\begin{aligned} p_k \geq 0 &\rightarrow y_{k+1} = y_k + 1, \text{ então } p_{k+1} = p_k + 2\Delta y - 2\Delta x \\ p_k < 0 &\rightarrow y_{k+1} = y_k, \text{ então } p_{k+1} = p_k + 2\Delta y. \end{aligned} \tag{10.4}$$

Se $\Delta y < 0$ e $\Delta x > 0$, como na Figura 10.9.(b), temos $d_2 = |y - (y_k - 1)|$ e $d_1 = |y_k - y|$. Assim,

$$\begin{aligned} d_1 - d_2 &= (y_k - y) - (y - y_k + 1) \\ &= 2y_k - 2y - 1 \\ &= 2y_k - 2(\frac{\Delta y}{\Delta x}(x_k + 1) + b) - 1 \\ &= 2y_k - 2\frac{\Delta y}{\Delta x}x_k - 2\frac{\Delta y}{\Delta x} - 2b - 1 \end{aligned}$$

Analogamente, como $\Delta x > 0$, basta analisarmos o sinal da **diferença dos erros**

$$p_k = (d_1 - d_2)\Delta x = -2\Delta y x_k + 2\Delta x y_k + (-2\Delta y - 2b\Delta x - \Delta x).$$

para determinar o decremento, ou não, da coordenada y no ponto (x_{k+1}, y_{k+1}) em realação ao ponto (x_k, y_k):

$$\begin{aligned} p_k \geq 0 &\rightarrow y_{k+1} = y_k - 1 \\ p_k < 0 &\rightarrow y_{k+1} = y_k \end{aligned} \tag{10.5}$$

O erro p_k para cada *pixel* k pode ser também obtido de forma recorrente. Fazemos $c_1 = (-2\Delta y - 2b\Delta x - \Delta x)$, ou seja,

$$p_k = -2\Delta y x_k + 2\Delta x y_k + c_1$$

temos para p_{k+1}

$$\begin{aligned} p_{k+1} &= -2\Delta y x_{k+1} + 2\Delta x y_{k+1} + c_1 \\ &= -2\Delta y(x_k + 1) + 2\Delta x y_{k+1} + 2\Delta x y_k - 2\Delta x y_k + c_1 \\ &= (-2\Delta y x_k + 2\Delta x y_k + c_1) + 2\Delta x(y_{k+1} - y_k) - 2\Delta y \\ &= p_k - 2\Delta y + 2\Delta x(y_{k+1} - y_k). \end{aligned} \tag{10.6}$$

com

$$\begin{aligned} p_0 &= -2\Delta y x_0 + 2\Delta x y_0 + c_1 \\ &= -2\Delta y x_0 + 2\Delta x(\frac{\Delta y}{\Delta x}x_0 + b) - 2\Delta y - 2b\Delta x - \Delta x \\ &= -2\Delta y - \Delta x \ . \end{aligned} \tag{10.7}$$

Pela Eq. 10.5 podemos simplificar ainda a Eq. 10.6, pois

$$\begin{aligned} p_k \geq 0 &\rightarrow y_{k+1} = y_k - 1, \text{ então } p_{k+1} = p_k - 2\Delta y - 2\Delta x \\ p_k < 0 &\rightarrow y_{k+1} = y_k, \text{ então } p_{k+1} = p_k - 2\Delta y. \end{aligned} \tag{10.8}$$

Observe ainda que $|\Delta y| = -\Delta y$, então

$$\begin{aligned} p_k \geq 0 &\rightarrow y_{k+1} = y_k - 1, \text{ então } p_{k+1} = p_k + 2|\Delta y| - 2\Delta x \\ p_k < 0 &\rightarrow y_{k+1} = y_k, \text{ então } p_{k+1} = p_k + 2|\Delta y|. \end{aligned}$$

com

$$p_0 = 2|\Delta y| - \Delta x$$

Algoritmo de Ponto Médio

Pitteway apresentou em 1967 uma formulação ligeiramente diferente da do algoritmo de Bresenham, com a vantagem de ser facilmente extensível a todas as cônicas. Esta nova formulação é conhecida por **algoritmo de ponto médio**. Ao invés de computar recorrentmente a diferença dos erros para decidir o incremento, ou não, de uma variável enquanto a outra varia de uma unidade, avalia-se no algoritmo de ponto médio em que lado o ponto médio M da variável dependente fica. Se o ponto médio M estiver abaixo da reta, o *pixel* NE é escolhido; do contrário, o *pixel* E é escolhido (Figura 10.10). Seja $f(x, y) = ax + by + c$ a representação implícita da reta e $\Delta y < \Delta y$, podemos traduzir o critério de escolha em seguintes condições algébricas:

- Se $f(x_k + 1, y_k + 0.5) \geq 0$, $M = (x_k + 1, y_k + 0.5)$ está acima da/sobre reta; portanto, o *pixel* E é escolhido.
- Se $f(x_k + 1, y_k + 0.5) < 0$, $M = (x_k + 1, y_k + 0.5)$ está abaixo da reta; portanto, o *pixel* NE é escolhido.

Chamamos $d_k = f(x_k + 1, y_k + 0.5)$ de **variável de decisão**.

De forma análoga a diferença de erros, a variável de decisão também pode ser obtida recorrentemente. Basta observar que a variável na iteração $k + 1$ é

$$d_{k+1} = f(x_k + 2, y_{k+1} + 0.5).$$

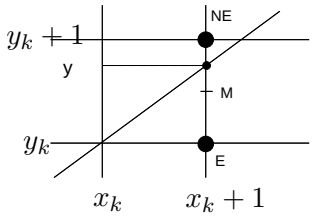


Figura 10.10: Alternativa de escolha: NE, ponto médio abaixo da reta e E, ponto médio acima da reta.

Se $d_k \geq 0$, então $y_{k+1} = y_k$. Isso significa que

$$\begin{aligned} d_{k+1} = f(x_k + 2, y_{k+1} + 0.5) &= a(x_k + 2) + b(y_k + 0.5) + c \\ &= a(x_k + 1) + a + b(y_k + 0.5) + c \\ &= f(x_k + 1, y_k + 0.5) + a \\ &= d_k + a. \end{aligned}$$

Para $d_k < 0$, temos $y_{k+1} = y_k + 1$, ou seja,

$$\begin{aligned} d_{k+1} = f(x_k + 2, y_{k+1} + 0.5) &= a(x_k + 2) + b(y_k + 1.5) + c \\ &= a(x_k + 1) + a + b(y_k + 0.5) + b + c \\ &= f(x_k + 1, y_k + 0.5) + a + b \\ &= d_k + a + b. \end{aligned}$$

O valor inicial d_0 pode ser obtido diretamente pela expressão

$$\begin{aligned} d_0 = f(x_0 + 1, y_0 + 0.5) &= a(x_0 + 1) + b(y_0 + 0.5) + c \\ &= a(x_0 + 1) + b(y_0 + 0.5) + c \\ &= f(x_0, y_0) + a + 0.5b. \end{aligned} \tag{10.9}$$

10.2.3 Rasterização de Polígonos

Rasterização de polígonos consiste em transformar uma região plana delimitada por uma sequência fechada de segmentos em um conjunto de *pixels* conexos. Tipicamente, os algoritmos para rasterização de polígonos exploram certos tipos de coerência entre os *pixels*, a fim de reduzir os passos de processamento.

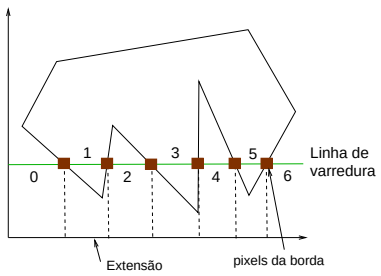


Figura 10.11: Linha de varredura dividida em vários intervalos.

Algoritmo de Scanline

O algoritmo mais conhecido é o **algoritmo de varredura por linha** (*scan-line*), provavelmente desenvolvido por Wylie et al. em 1967. Este algoritmo explora vários tipos de **coerência espacial** apresentadas na seção 9.1, e consegue determinar **incrementalmente**, linha por linha, os *pixels* dentro de uma região, com uso dos seguintes fatos:

1. os *pixels* das arestas da borda da região podem ser obtidos com uso dos algoritmos recorrentes de rasterização de linhas (seção 10.2.2);

2. numa linha de varredura os *pixels* podem ser divididos em distintas extensões (*extent*) separadas por *pixels* correspondentes às arestas da borda. A classificação de pertinência das extensões à área de interesse só muda nestes *pixels* "divisórios" ((Figura 10.11)); e

3. a classificação de pertinência das extensões identificadas em cada linha se alterna ao longo de uma linha de varredura.

O algoritmo inicia definindo os *y-buckets* para cada linha de varredura. Um *y-bucket* associado a uma linha contém informações de uma aresta do polígono que a intercepta. São armazenados no *y-bucket* a coordenada y do ponto extremo que tem maior valor de y (y_{max}), a coordenada x do ponto extremo que tem menor valor de y (x_{min}) e a inclinação da reta ($\frac{1}{m}$). O conjunto de *y-buckets* é denominado *edge table*. Figura 10.12 apresenta um esboço desta tabela.

Com uso das informações contidas nos *y-buckets*, é possível preencher a região à medida que se processe sequencialmente as linhas de varredura começando com a linha 0. Para cada linha de varredura y, os *y-buckets* ativos são ordenados de acordo com os valores x no seu campo x_{min} para poder fazer o preenchimento da linha de forma alternada entre os intervalos

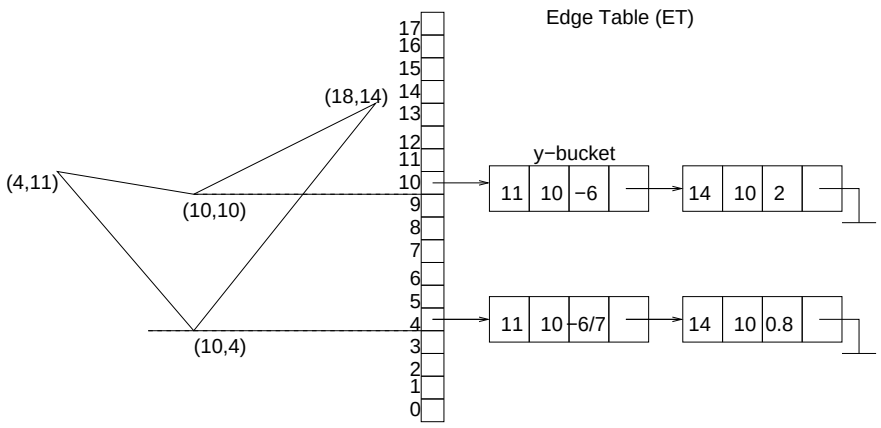


Figura 10.12: Algoritmo de *scanline*: tabela de arestas.

de segmento delimitados por estes pontos. O conjunto de *y-buckets* ativos associados à linha de varredura corrente é conhecida como **lista ativa** (*active edge table*). Finalmente, antes de passar para a próxima linha, remover da lista ativa os *y-buckets* que tiverem $y + 1 > y_{max}$. Para os *buckets* remanescentes, substituir o valor x no seu campo x_{min} por $x + \frac{\Delta x}{\Delta y}$. Ao passar para a próxima linha, é adicionada à lista ativa os *y-buckets* da linha, e o procedimento é repetido até que a tabela de arestas fique vazia.

Podemos integrar ao algoritmo de varredura por linha a tonalização de Gouraud. O cômputo da cor em cada *pixel* pode ser feito incrementalmente, se as cores nos pontos $P_1 = (x_1, y_1, z_1, 1)$ e $P_2 = (x_2, y_2, z_2, 1)$ forem conhecidas. Sejam $C_1 = (R_1, G_1, B_1)$ e $C_2 = (R_2, G_2, B_2)$ as cores em P_1 e P_2, respectivamente. Pela interpolação linear, obtém-se a cor

$$C(t) = (1 - t)C_1 + tC_2$$

para o ponto

$$P(t) = (1 - t)P_1 + tP_2.$$

Numa mesma linha de varredura, o valor t_{k+1} deve ser em função do incremento de X

$$X_k + 1 = x_1 + t_{k+1}(x_2 - x_1)$$

ou seja

$$(x_1 + t_k(x_2 - x_1)) + 1 = x_1 + t_{k+1}(x_2 - x_1).$$

Daí podemos chegar a

$$t_{k+1} = t_k + \frac{1}{\Delta x}.$$

E, de forma análoga, podemos derivar o valor t_{k+1} na linha de varredura $k+1$ em relação ao valor t_k da linha anterior k

$$t_{k+1} = t_k + \frac{1}{\Delta y}.$$

Algoritmo de Preenchimento

Quando uma região é rasterizada e queremos trocar, por exemplo, a sua cor de preenchimento por uma outra, podemos utilizar algoritmos recursivos com base em **preenchimento a partir de uma semente**. Um exemplo desta classe de algoritmos é o algoritmo *flood-fill*. O algoritmo percorre a partir da semente todos os *pixels* conectados que tiverem a cor antiga e trocá-la pela nova de acordo com o seguinte procedimento:

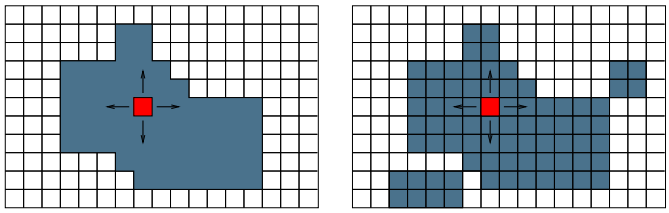
Input: Semente:(s_x, s_y)
Output: polígono preenchido
empilhe a semente (s_x, s_y);
while *pilha não é vazia* **do**
  Desempilhe o topo da pilha: (x,y) ;
  if *I(x,y)* $\neq$ *I*(s_x, s_y) **then**
    I(x,y) = I(s_x, s_y) ;
  end
  Empilhe os *pixels* adjacentes ;
end

Algoritmo 7: Função Floodfill: algoritmo de preenchimento de área.

O resultado do procedimento depende do tipo de conectividade considerado. Para a conectividade-de-4, no máximo 4 *pixels* adjacentes são empilhados em cada iteração e para a conectividade-de-8, 8 *pixels*. Um polígono, como o mostrado na Figura 10.13.(b), só poderá ser corretamente preenchido com o novo atributo se for considerada a conectividade-de-8.

10.2.4 Rasterização de Modelos 2.5D

Vimos na seção 5.4.4 que ao transformar os vértices do espaço NDC para NC podemos preservar a coordenada n, armazenando-a como um atributo adicional do *pixel*. Esta coordenada corresponde à profundidade da amostra projetada no *pixel* em relação ao observador (Figura 10.14). Conforme vimos na seção 9.3.3, com esta informação adicional o computador consegue resolver corretamente o problema de oclusão dos polígonos durante a sua rasterização.



(a) (b)

Figura 10.13: Algoritmo de preenchimento: (a) conectado-de-4; (b) conectado-de-8.

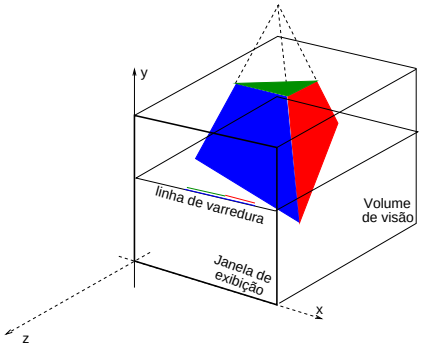


Figura 10.14: Raserização de modelos $2\frac{1}{2}$D.

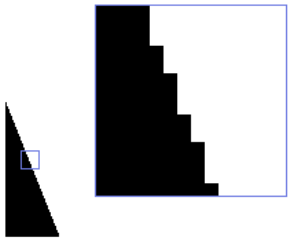
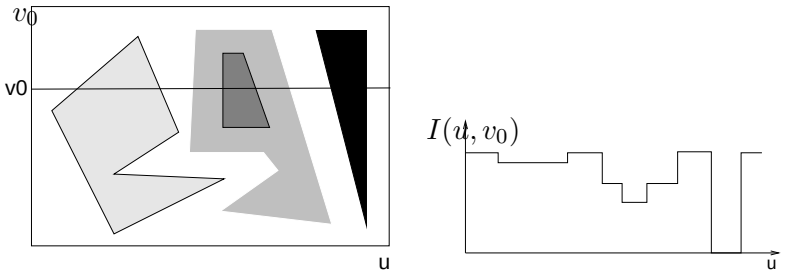


Figura 10.15: Artefatos na discretização: bordas serrilhadas (Fonte: http://msdn.microsoft.com/en-us/library/bb172267(VS.85).aspx.



(a) (b)

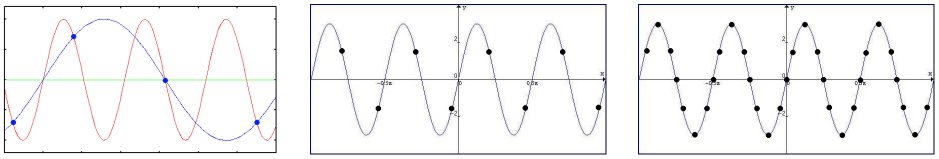
Figura 10.16: Imagem: (a) intensidade e (b) gráfico função de $I(u, v_0)$.

10.3 Análise Espectral

Um dos problemas encontrados nos processos de rasterização apresentados é a presença de **bordas serrilhadas**, *jagged* ou *stair step pattern* em inglês, nas imagens, devido às decisões binárias (acende–apaga) em relação à luminância de cada *pixel* da tela e às possíveis transições abruptas de luminância entre dois *pixels* vizinhos, como ilustra Figura 10.15. É natural se perguntar se há alguma forma de eliminar, ou pelo menos atenuar, tais efeitos. Veremos nesta seção que esta pergunta pode ser respondida com os conceitos do Processamento de Sinais.

Uma imagem pode ser representada como uma **função contínua** $I(u, v)$ que mapeia um domínio espacial $[u_{min}, u_{max}] \times [v_{min}, v_{max}]$ aos valores de luminância. Figura 10.16.(b) mostra o gráfico da função $I(u, v_0)$ ao longo da linha $v = v_0$.

No processo de rasterização, a função de intensidade luminosa é amostrada em alguns pontos (u, v), transformando-se numa **função discreta**. Esta função discreta contém somente as intensidades dos pontos amostra-



(a) (b) (c)

Figura 10.17: Amostragem: (a) sub-amostragem; (b) na frequência de Nyquist; e (c) super-amostragem.

dos, podendo haver perda da informação de variação das intensidades entre duas amostras vizinhas. Existe uma frequência de amostragem para a qual não haja esta perda de informação? Esta pergunta pode ser respondida mais facilmente com uso da **transformada de Fourier** de $I(u,v)$. Uma transformada de Fourier relaciona esta função definida no domínio espacial para o domínio de frequência de variação das luminâncias w_u e w_v

$$\mathcal{I}(w_u, w_v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(u,v)[\cos 2\pi(w_u u + w_v v) - i \sin 2\pi(w_u u + w_v v)]dudv. \tag{10.10}$$

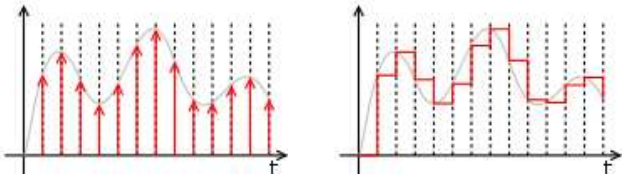
Uma breve revisão sobre a transformada de Fourier é dada na seção 10.3.1.

Sejam $[-U, U]$ e $[-V, V]$ as faixas espectrais para as quais $\mathcal{I}(w_u, w_v)$ não se anula. O **teorema de amostragem** nos diz que se

$$\frac{1}{\Delta u} \geq 2U \qquad \frac{1}{\Delta u} \geq 2V, \tag{10.11}$$

$I(u,v)$ pode ser completamente recuperada. $2U$ e $2V$ são chamados **limite (de frequência) de Nyquist** nas direções u e v, respectivamente. Δu e Δv são, por sua vez, conhecidos como **intervalos de Nyquist**. Quando a frequência de amostragem é acima da frequência de Nyquist, dizemos que houve uma **super-amostragem** e abaixo desta frequência, **sub-amostragem**. Figura 10.17 ilustra distintas frequências de amostragem de um sinal senoidal (uni-dimensional). Pelo teorema de amostragem, somente amostragem com frequência maior ou igual a frequência de Nyquist, o sinal poderá ser reconstruído fielmente. Observe o sinal reconstruído, em azul, a partir de sinais sub-amostrados. Na seção 10.3.2 justificaremos isso esboçando os sinais no espaço espectral.

Para visualizar as amostras num dispositivo de saída analógico, como um monitor CRT, precisamos **reconstruir** a partir das amostras sinais analógicos através de um conversor D/A. O esquema *sample and hold* por



(a) (b)

Figura 10.18: Reconstrução de sinal: (a) amostragem; (b) sinal reconstruído (Fonte: http://en.wikipedia.org/wiki/Sample_and_hold).

um período de "1 *pixel*" é tipicamente utilizado para esta finalidade (Figura 10.18). No caso de um monitor CRT, o valor da luminância associado a cada *pixel* é convertido em tensão a ser aplicada no canhão dos elétrons. Como a função de reconstrução possui tipicamente componentes de altas frequências, veremos que é difícil assegurar a reconstrução exata da imagem original. Na seção 10.3.3 mostraremos, sob o ponto de vista teórico, como se pode atenuar as componentes de altas frequências e produzir imagens "visualmente" agradáveis.

10.3.1 Transformada de Fourier

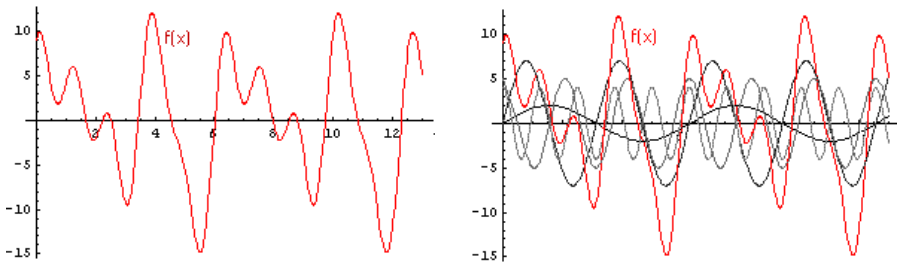
Jean Baptiste Joseph Fourier introduziu no início do século XIX uma forma simples de visualizar e manipular uma função complexa. Segundo ele, qualquer função periódica, por mais complicada que seja, pode ser representada como a soma de várias funções seno e cosseno com amplitudes, fases e períodos escolhidos convenientemente. Figura 10.19 mostra a representação da função periódica $f(x)$, em vermelho, como a soma de duas senóides e duas cossenóides de períodos distintos, em cinza

$$f(x) = 2\sin(x) + 7\sin(2x) + 5\cos(3x) + 4\cos(5x)$$

Quando a função $f(x)$ é muito complexa, os termos de senóides e/ou cossenóides podem ser estendidos indefinidamente para obter uma aproximação melhor de $f(x)$. Figura 10.20 ilustra as diferentes aproximações de uma função $f(x)$ plotada em azul por uma série de cossenóides. Observe que quanto maior o número de termos ou harmônicas, mais se aproxima o seu somatório da curva original.

Se uma função não-periódica $f(u)$

1. é contínua por parte num intervalo finito,



(a) (b)

Figura 10.19: Série de Fourier: (a) função periódica; (b) expansão em funções trigonométricas (Fonte: http://www.seara.ufc.br/tintim/matematica/fourier/fourier1.htm).

2. tem a derivada direita e derivada esquerda em todos os pontos, e
3. é absolutamente integrável ao longo do eixo x, ou seja, existe o limite

$$\lim_{a\to-\infty}\int_a^0 |f(u)|du + \lim_{b\to\infty}\int_0^b |f(u)|du,$$

ela também pode ser expandida em funções simples através da **integral de Fourier**

$$f(u) = \int_0^\infty [A(w_u)\cos 2\pi w_u u + B(u)\sin 2\pi w_u u]dw_u,$$

onde

$$A(w_u) = \int_{-\infty}^\infty f(u)\cos 2\pi w_u u du \qquad B(w_u) = \int_{-\infty}^\infty f(u)\sin 2\pi w_u u du.$$

Em forma complexa,

$$f(u) = \int_{-\infty}^\infty F(w_u)e^{j2\pi w_u u}dw_u, \tag{10.12}$$

onde

$$F(w_u) = \int_{-\infty}^\infty f(u)e^{-j2\pi w_u u}du. \tag{10.13}$$

Denominamos $F(w_u)$ a **transformada de Fourier** da função $f(u)$, $\mathsf{F}\{f(u)\}$, e dizemos que $f(u)$ é a **transformada de Fourier inversa** de $F(w_u)$. $f(u)$ e $F(w_u)$ são funções complexas, contendo dois componentes: um real $r(u)$ e $\Re(w_u)$ e um imaginário $i(u)$ e $\Im(w_u)$, respectivamente.

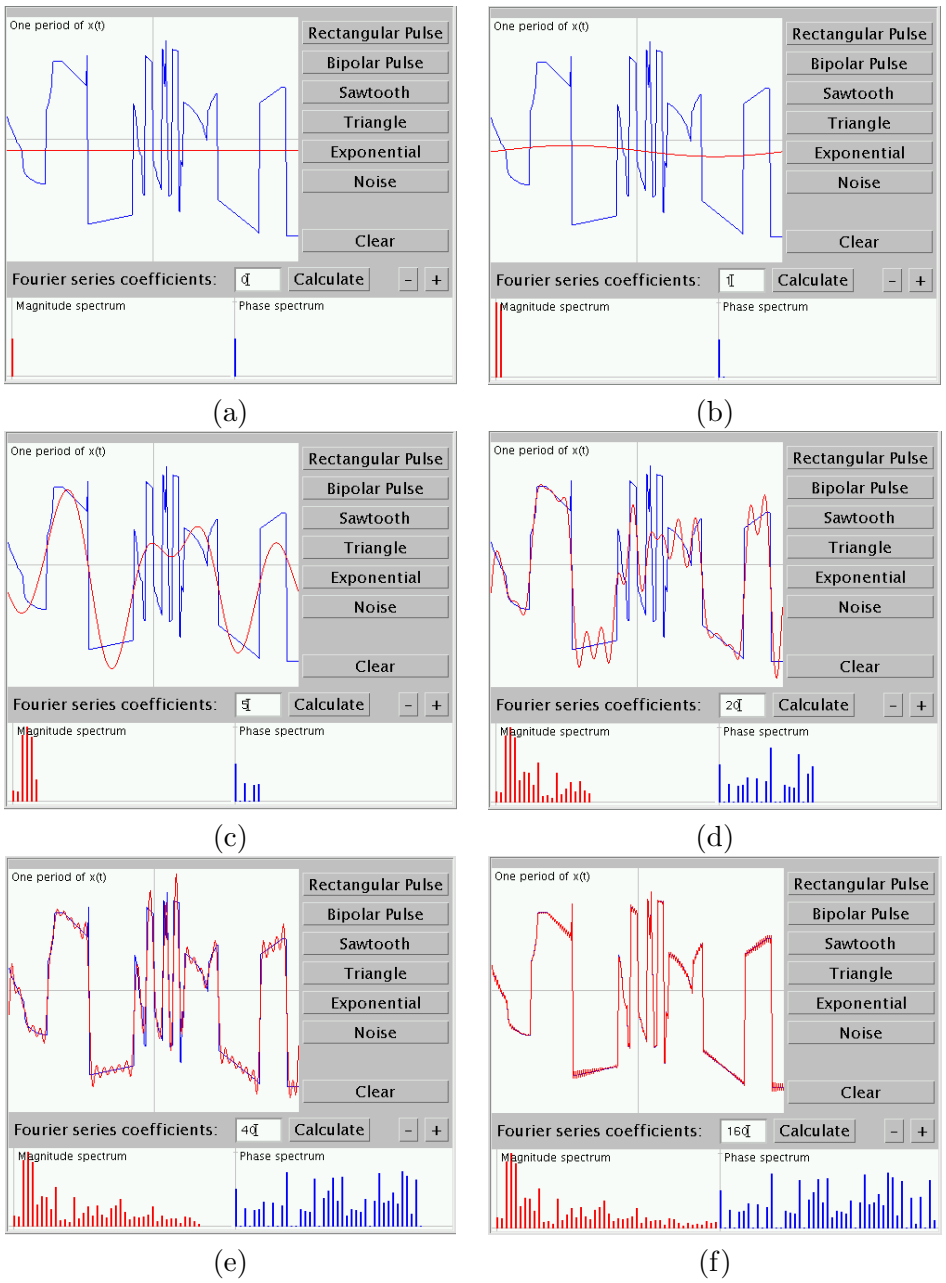


Figura 10.20: Aproximação de um sinal por soma de harmônicas: (a) componente dc; (b) 11 harmônica; (c) 5 harmônicas; (d) 20 harmônicas; (e) 40 harmônicas; (f) 160 harmônicas (Fonte: http://www.jhu.edu/ signals/fourier2/).

Sendo Eq. 10.12 uma superposição de sinais de todas as freqüências, ela é também conhecida como uma **representação espectral** de $f(x)$, cujo **espectro de Fourier** é

$$|F(w_u)| = \sqrt{\Re^2(w_u) + \Im^2(w_u)} \tag{10.14}$$

e o **ângulo de fase**,

$$\phi(w_u) = \tan^{-1} \frac{\Im(w_u)}{R(w_u)}. \tag{10.15}$$

O espectro de freqüência é muito utilizado para análise visual de $f(u)$.

A **densidade espectral** $|F(w_u)|^2$ mede a intensidade de $f(u)$ no intervalo de freqüência entre w_u e $w_u + \Delta W_u$ e a integral

$$\int_{-\infty}^{\infty} |F(w_u)|^2 dw \tag{10.16}$$

pode ser interpretada como a **energia** ou **potência** do sistema $f(u)$.

Algumas propriedades da transformada de Fourier:

Deslocamento: $\mathsf{F}\{f(u-a)\} = e^{j2\pi a w_u}\mathsf{F}\{f(u)\}$.

Linearidade: $\mathsf{F}\{af(u) + bg(u)\} = a\mathsf{F}\{f(u)\} + b\mathsf{F}\{g(u)\}$.

Derivação: $\mathsf{F}\{\frac{d^n f(u)}{du^n}\} = (j2\pi w_u)^n \mathsf{F}\{f(u)\}$.

Convolução: $\mathsf{F}\{f(u) * g(u)\} = \mathsf{F}\{f(u)\}\mathsf{F}\{g(u)\}$.

Multiplicação: $\mathsf{F}\{f(u)g(u)\} = \mathsf{F}\{f(u)\} * \mathsf{F}\{g(u)\}$.

No caso de imagens, que contem duas variáveis, é aplicada uma transformada de Fourier bi-dimensional. Considerando que todas as imagens $I(u,v)$ satisfaçam a condição de continuidade e de integrabilidade, a sua transformada de Fourier $\mathsf{F}(I(u,v)) = \mathcal{I}(w_u, w_v)$ é dada pela Eq. 10.10. Observe que, embora $I(u,v)$ seja uma função real, $\mathcal{I}(w_u, w_v)$ é geralmente complexa, isto é,

$$\mathcal{I}(w_u, w_v) = \Re(\mathcal{I}(w_u, w_v)) + j\Im(\mathcal{I}(w_u, w_v)),$$

cuja amplitude/magnitude é dada por

$$|\mathcal{I}(w_u, w_v)| = \sqrt{\Re^2(\mathcal{I}(w_u, w_v)) + \Im^2(\mathcal{I}(w_u, w_v))}$$

e o ângulo de fase por

$$\phi(w_u, w_v) = tg^{-1} \frac{\Im(\mathcal{I}(w_u, w_v))}{\Re(\mathcal{I}(w_u, w_v))},$$

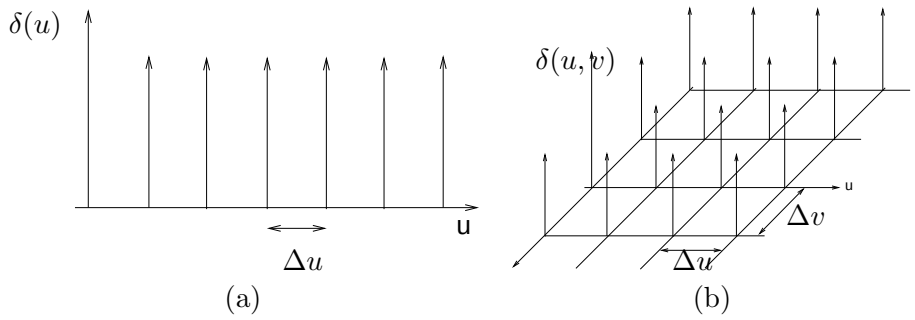


Figura 10.21: Função de pente: (a) uni-dimensional; (b) bi-dimensional.

A magnitude da transformada de Fourier de uma imagem indica a "quantidade" da componente de frequência (w_u, w_v) e o ângulo de fase indica "onde" se encontra tal componente na imagem. Muitas vezes, um rápido exame visual da transformada de uma imagem permite ter uma noção das harmônicas dominantes. Figura 10.2 mostra o gráfico de magnitude e de fase da transformada de Fourier da imagem apresentada, sendo o nível de cinza proporcional à amplitude de $|\mathcal{I}(w_u, w_v)|$ e ao ângulo de fase $\phi(w_u, w_v)$ para cada "par de frequências" (w_u, w_v).

10.3.2 Amostragem

Para obter as amostras uniformes separadas de $(\Delta u, \Delta v)$, ou seja uma **amostragem pontual uniforme**, basta multiplicarmos a função $I(u, v)$ por uma **função pente** bidimensional $\delta(u, v)$, em inglês *comb function* (Figura 10.21)

$$\sum_{k=-\infty}^{\infty} \sum_{j=\infty}^{\infty} \delta(u - k\Delta u, v - j\Delta v).$$

Ou seja,

$$I(u,v) \sum_{k=-\infty}^{\infty} \sum_{j=\infty}^{\infty} \delta(u-k\Delta u, v-j\Delta v) = \sum_{k=-\infty}^{\infty} \sum_{j=\infty}^{\infty} I(k\Delta u, j\Delta v)\delta(u-k\Delta u, v-j\Delta v).$$

Pelo **teorema da convolução**, a transformada deste produto equivale à convolução das suas transformadas correspondentes, $\mathsf{F}\{\delta(u, v)\}$ e $\mathcal{I}(w_u, w_v)$, ou seja,

$$\mathsf{F}(I(u, v)\delta(u, v)) = \mathsf{F}\{\delta(u, v)\} * \mathcal{I}(w_u, w_v).$$

Isso equivale a integrar as cópias de $\mathcal{I}(w_u, w_v)$, "transladadas" e ponderadas por $\mathsf{F}\{\delta(u, v)\}$. Se a função $\mathcal{I}(w_u, w_v)$ for de **banda limitada**, ou seja, ela

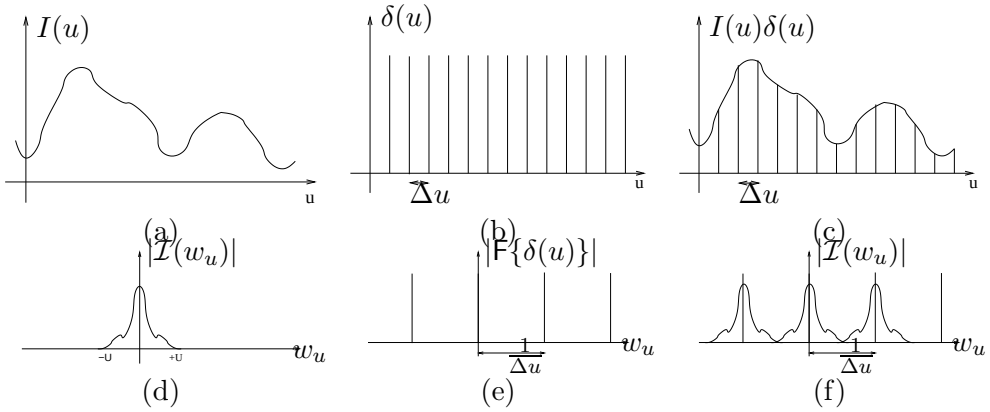


Figura 10.22: Amostragem: (a – c) multiplicação do sinal $I(u)$ pela função de pente no domínio espacial equivale a (d – f) convolução de $\mathcal{I}(w_u)$ com $\mathsf{F}\{\delta(u)\}$ no domínio espectral (Fonte: http://idav.ucdavis.edu/%7Eokreylos/PhDStudies/Winter2000/SamplingTheory.html).

tem $|\mathcal{I}(w_u, w_v)|$ nulos para u fora do intervalo $[-U, U]$ e v fora do intervalo $[-V, V]$ e o espaçamento de amostragem satisfizer Eq. 10.11, o resultado da convolução seria um "trem bidimensional" de réplicas de $\mathcal{I}(w_u, w_v)$, sem distorções, centradas nas frequências $(2mU, 2nV)$. Neste caso, a imagem poderá ser reconstruída integralmente sem distorções.

Figura 10.22.(c) ilustra o resultado da multiplicação de uma função unidimensional $I(u)$ com a função de pente $\delta(u)$ no domínio espacial. Esta operação corresponde à convolução das transformadas de duas funções no domínio espectral (Figura 10.22.(d,e)). Observe que neste caso, $\frac{1}{\Delta u} = 2U$ e o resultado é uma sequência de réplicas da transformada de $I(u)$ espaçadas de $\frac{1}{\Delta u}$, sem sobreposições.

Caso o inverso do espaçamento entre as duas réplicas no domínio de frequência for menor que a frequência de Nyquist do sinal de intensidade da imagem $I(u, v)$, componentes de altas frequências de uma réplica podem sobrepor as réplicas vizinhas, tornando-as indistinguíveis. Esta distorção impossibilita a reconstrução da imagem original. Este fenômeno é conhecido como *aliasing*. Figura 10.23 exemplifica uma situação em que as componentes de altas frequências se confundem com as componentes de baixas frequências no espectro de frequências, quando $\frac{1}{\Delta u} < 2U$.

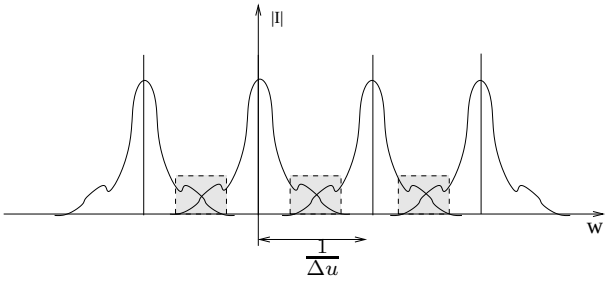


Figura 10.23: *Aliasing*.

10.3.3 Reconstrução

Para visualizar a imagem amostrada na tela de exibição, o sinal discreto é convertido em sinal analógico pelo esquema *sample and hold*. Isso corresponde a realizar convolução do sinal discreto por uma função pulso $Box(u, v)$ no domínio espacial. No domínio espectral esta operação equivale a multiplicar a sequência de réplicas por uma função $sinc(u, v)$, a fim de extrair uma única réplica. Mesmo que esta réplica seja intacta, sem distorções por sobreposições, a função de reconstrução pode introduzir componentes de altas frequências quando a sua frequência de corte é muito alta. Figura 10.24 ilustra o processo de reconstrução de um sinal uni-dimensional com uso de uma função pulso em ambos os domínios. Observe que a função pulso pode introduzir falsas componentes de alta frequência no sinal a ser reconstruído, alargando a sua largura de banda.

10.4 Antialiasing

Visualmente percebeu-se que os algoritmos de rasterização apresentados na seção 10.2 são passíveis à geração de bordas serrilhadas. Na seção 10.3 mostramos, com base em Processamento de Sinais, que estas bordas serrilhadas é um efeito de *aliasing*. Como podemos atenuar tais artefatos para gerar imagens visualmente mais agradáveis? Historicamente, a solução consiste em aumentar a resolução da tela de exibição, ou seja, aumentar a taxa de amostragem. Já que o "degrau" do serrilhado não pode ser maior que um *pixel*, quanto menor o tamanho de um *pixel*, menos perceptíveis ficam os serrilhados. Outra alternativa seria alterar o esquema de conversão digital–analógico, utilizando uma função de reconstrução de banda estreita. Infelizmente, ambas as soluções não são sempre factíveis, pois elas dependem do *hardware* disponível.

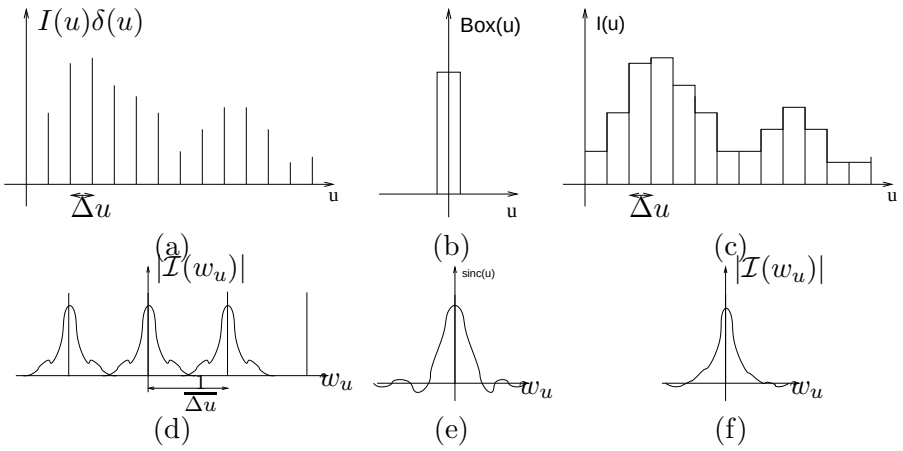


Figura 10.24: Reconstrução: (a – c) convolução das amostras pelo esquema *sample and hold* no domínio espacial equivale a (d – f) multiplicação do sinal *sinc* com "trem de réplicas".

Nesta seção apresentamos três outras alternativas. A idéia se baseia em uma observação simples: quando estivermos muito afastados da tela de exibição, não distinguimos as bordas serrilhadas. Isso decorre da nossa limitada acuidade visual, conforme vimos na seção 1.1. O que percebemos de fato é o resultado da combinação de cores de vários *pixels* em torno do *pixel* da borda quando a imagem estiver muito distante. Portanto, uma solução seria emular esta "combinação", eliminando as componentes de altas frequências ou atenuando as fortes transições de luminâncias nas bordas, com uso de mais de uma amostra por *pixel*.

Amostragem por Área : Esta solução se baseia no fato de que qualquer primitiva rasterizada ocupa, no mínimo, um *pixel*. Ela consiste em ponderar a luminância calculada para a primitiva pela razão da cobertura da primitiva no *pixel* e a área A do *pixel*

$$\frac{\int_P f(x,y)dxdy}{A}.$$

Esta luminância "suavizada" é então atribuída para o *pixel*. Observe as diferenças na rasterização da borda de um polígono sem e com uso da técnica de amostragem de área na Figura 10.25. A suavização consegue atenuar o padrão serrilhado.

Como a suavização reduz altas frequências na variação da luminância

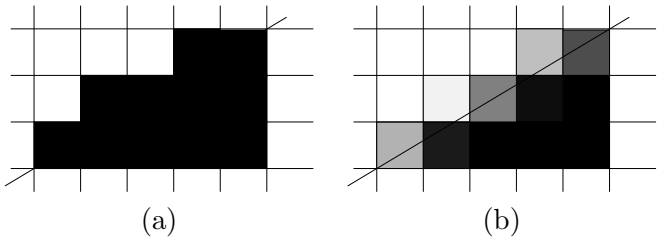


Figura 10.25: Amostragem por área: (a) sem *antialiasing*; (b) com *antialiasing*.

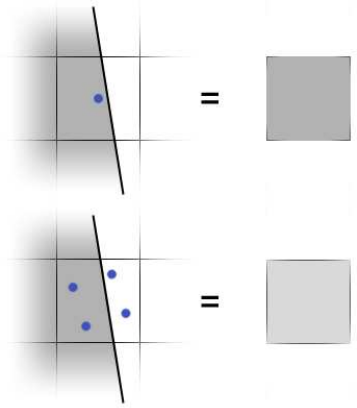


Figura 10.26: Amostragem: (a) centro de *pixel*; (b) super-amostragem.

dos *pixels*, a largura da faixa ocupada pelo espectro da imagem diminui. Isso diminui a probabilidade de *aliasing*.

Super-amostragem : ao invés de uma amostra, um conjunto de amostras p_1, p_2, $\cdots$, p_n são consideradas para cada *pixel* (usualmente, uma potência de 2) e a luminância média destas amostras é calculada. Podemos considerar esta amostragem uma forma discreta da amostragem por área, tendo um custo computacional menor. Figura 10.26 mostra a diferença entre uma amostragem por *pixel* e super-amostragem.

Multi-amostragem : é uma otimização de super-amostragem, a fim de reduzir o seu custo computacional. Esta otimização é alcançada através da diferenciação entre os *pixels* da borda e os interiores, limitando a aplicação da técnica de super-amostragem somente para os *pixels* da borda.

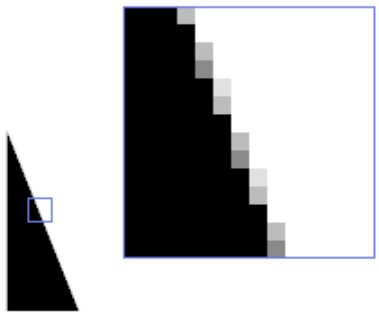


Figura 10.27: Imagem da Figura 10.15 com multi-amostragem.

Capítulo 11

Quantização

O objetivo deste capítulo é mostrar como um computador, condicionado a uma quantidade limitada de valores representáveis, consegue produzir, de forma mais próxima possível da nossa percepção, as cores de uma imagem. Após a leitura deste capítulo, você deve ser capaz de

- distinguir a amostragem e a quantização;
- distinguir os valores ou níveis de quantização e as células de quantização;
- caracterizar uma técnica de meio-tom;
- distinguir as técnicas de meio-tom digital e as técnicas de *dithering*/difusão de erro;
- quantizar uma imagem em tons de cinza para os níveis de quantização pré-definidos;
- aplicar técnicas de *dithering* ou de Floyed-Steinberg para quantizaruma imagem em tons de cinza

---

No universo digital a quantidade de *bits* disponíveis para representar um valor é limitada, requerendo aproximação de um intervalo infinito de valores reais por um conjunto finito de valores computacionalmente representáveis, como vimos na seção 7.6. Esta aproximação é denominada **quantização**. Figura 11.1 ilustra um sinal contínuo amostrado e quantizado. Compare-a com a Figura 10.22.(c) e observe que o valor q_i que cada amostra assume

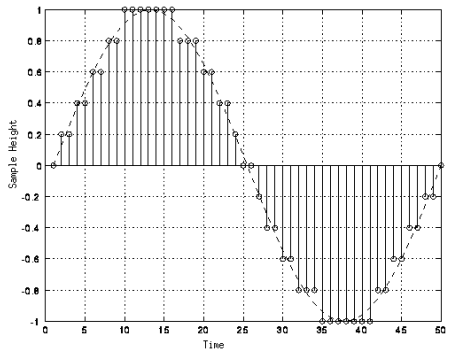


Figura 11.1: Amostragem e quantização de um sinal.

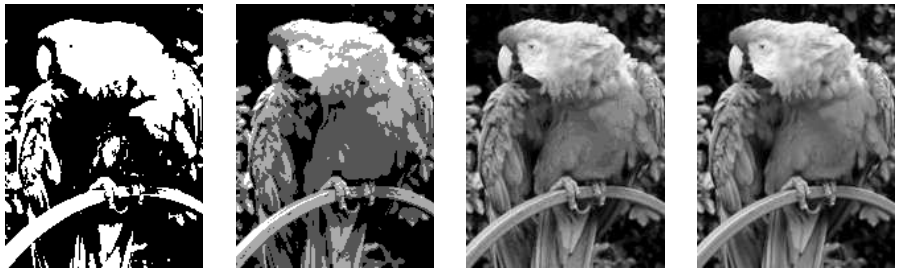
na Figura 11.1 não é exatamente igual ao seu valor original c_i como na Figura 10.22.(c). Os valores c_i foram aproximados para um dos 11 valores pré-definidos $q_i \in \{-1, -0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8, 1.0\}$.

A técnica de quantização mais trivial é o **arredondamento**. No caso de luminância, simples arredondamento dos valores dos seus componentes, sem levar em conta a frequência de ocorrência de cada valor, pode resultar em distorções ou artefatos. Por exemplo, uma imagem, em tons de cinza, com todos valores menores que 0.5 ficaria completamente preta após o processo de arredondamento. É, portanto, desejável uma outra alternativa que apresente as seguintes características:

1. Todas as cores representantes devem corresponder a, pelo menos, uma cor do gamute da image, a fim de minimizar a perda de informação.

2. As cores representantes devem ser próximas das cores de maior frequência de ocorrência na imagem, a fim de evitar o erro de quantização.

3. As cores representantes devem ser indistintas perceptualmente, para evitar **bordas falsas** ou **contornos falsos**. Por exemplo, se o nível de quantização for próximo da resolução visual humana ($\approx$ 50 níveis de cinza ou $\approx$ 128 cores espectrais), poderemos reproduzir uma imagem equivalente à percebida pela nossa visão, sem nenhuma degradação perceptível. Figura 11.2 ilustra os efeitos produzidos pelo aumento de planos de *bit* para representar os níveis de cinza em uma imagem. Observe que para uma quantidade pequena de níveis de cinza surgem transições abruptas nas áreas de níveis de cinza suaves. Tais transições são conhecidas como contornos falsos. Tais contornos desaparecem



(a) (b) (c) (d)

Figura 11.2: Imagem com: (a) 1 plano de *bit*; (b) 2 planos de *bit*; (c) 4 planos de *bit*; (d) 8 planos de *bit*.

completamente na Figura 11.2.(d), com 2^8 níveis de cinza.

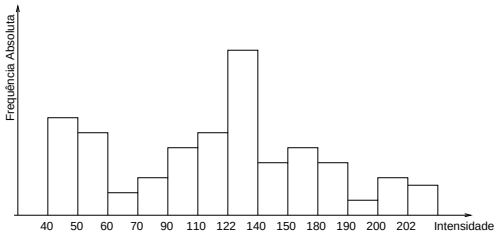
Para o primeiro e o segundo item, é importante que tenhamos o conhecimento das propriedades estatísticas relevntes da imagem, como a função de distribuição de probabilidade das cores/luminâncias em uma imagem. Uma aproximação desta distribuição pode ser representada graficamente por um **histograma**, como veremos na seção 11.1. Para o terceiro item, devemos particionar o gamute de cores em **células de quantização**, mutuamente exclusivas, que cubram cores perceptualmente indistingüíveis e que tenham **contornos de quantização** imperceptíveis. Na seção 11.2 veremos algumas técnicas de particionamento do gamute de cores em células e a escolha de uma cor representante para cada célula, ou seja, o seu **nível de quantização** ou **valor de quantização**. Finalmente, apresentamos na seção 11.3 algumas técnicas para atenuar impactos visuais da quantização nas imagens resultantes, dando impressão de um número maior de cores e reduzindo os efeitos de contornos falsos.

11.1 Histograma

Histograma é amplamente utilizado na análise e no processamento de imagens. Ele nos informa a **frequência de ocorrência de um determinado atributo** de interesse numa imagem. Um **histograma de intensidade** associa a cada nível de intensidade da imagem o seu número de ocorrências, ou seja, o número de *pixels* nos quais ele aparece. É comum representar um histograma por um gráfico de barras. Figura 11.3.(a) mostra o nível de cinza em cada *pixel* de uma imagem em 2^8 tons de cinza e Figura 11.3.(b), o seu histograma correspondente. Por exemplo, o número de ocorrências do

202	122	122	40	50	60	70	200	200	200	180	180
150	90	90	90	40	50	70	180	180	180	90	90
150	110	40	50	50	50	70	40	40	40	40	90
150	110	110	40	50	40	40	110	110	110	40	90
202	110	122	122	40	40	70	110	110	110	122	90
202	110	202	122	70	60	60	50	122	122	122	90
140	140	140	150	150	50	122	122	122	122	122	122
140	200	200	150	50	50	50	190	190	180	180	122
140	140	140	150	150	150	122	122	122	122	122	122

(a)



(b)

Figura 11.3: Imagem: (a) níveis de cinza; (b) histograma.

nível de cinza "122" é 22, enquanto o número de ocorrências do nível de cinza "40" é 13.

Através de um histograma de intensidades podemos visualizar a distribuição das intensidades de uma imagem como um todo, e determinar, por exemplo, as intensidades dominantes ou o grau de contraste. Em termos da distribuição dos valores de intensidade em um histograma, existem essencialmente quatro tipos básicos de imagens: ocorrências concentradas na extremidade esquerda (intensidade baixa) do histograma ou concentradas na extremidade direita (intensidade alta), concentradas numa faixa muito estreita de intensidades ou numa faixa muito larga, como ilustra Figura 11.4.

Para as imagens coloridas, é comum utilizar o histograma RGB para visualizar a união das ocorrências dos valores nos três canais de cor R (vermelho), G (verde) e B (azul). Ele pode estar acompanhado do histograma de luminância que nos proporciona uma melhor "sensibilidade" do contraste e da intensidade das imagens.

11.2 Células e Valores de Quantização

Como já mencionamos antes, o processo de quantização compreende duas etapas:

1. determinação de **células de quantização** mutuamente exclusivas; e
2. determinação do **nível de quantização** para cada célula.

Estas duas etapas podem ocorrer de forma sequencial ou de forma interdependente. A qualidade de um quantizador pode ser avaliada pela **medida**

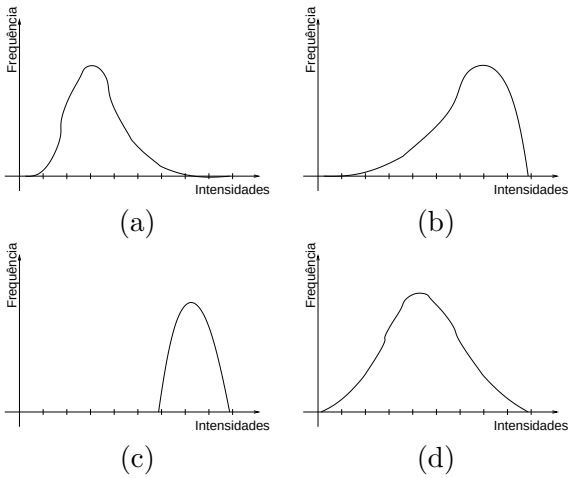


Figura 11.4: Imagem: (a) escura; (b) clara; (c) de baixo contraste; (d) de alto contraste.

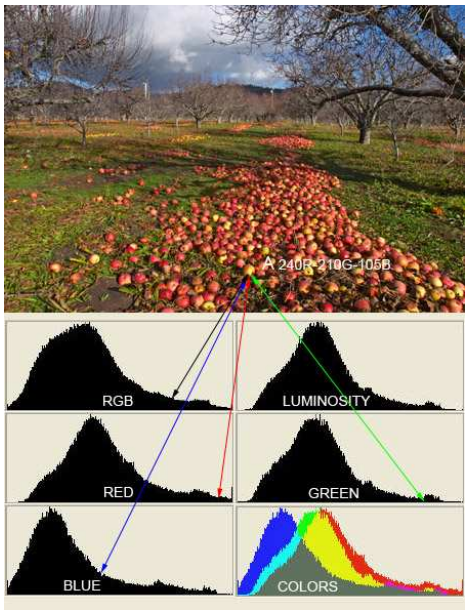
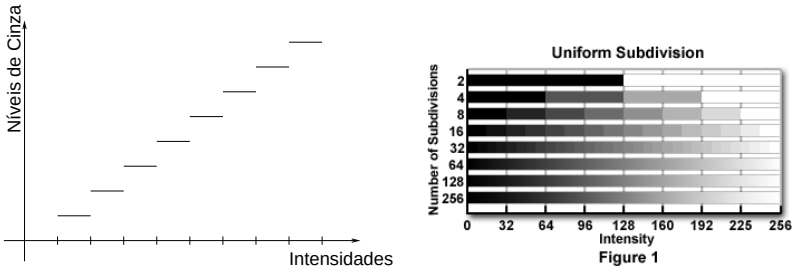


Figura 11.5: Imagem colorida: histogramas (Fonte: http://www.sphoto.com/techinfo/histograms/histograms2.htm.



(a) (b)

Figura 11.6: Particionamento uniforme: (a) histograma; (b) níveis de cinza (Fonte: http://micro.magnet.fsu.edu/primer/java/digitalimaging/processing/colorreduction/index.html.

de distorção de quantização, que é dada pelo erro médio quadrático dos **erros de quantização** Q de todas as cores que aparecem na imagem

$$Q = \int_{-\infty}^{\infty} p(c)\epsilon(c)dc, \tag{11.1}$$

onde $p(c)$ é uma função da frequência de ocorrência da cor c. Uma aproximação muito utilizada para o erro de quantização $\epsilon(c)$ de uma cor c é a sua distância euclidiana em relação ao valor de quantização q_i correspondente, isto é,

$$\epsilon(c) = |c - q_i|.$$

11.2.1 Quantização Uniforme

Uma forma mais simples e intuitiva para particionar o intervalo real $\mathcal{I} = [a, b]$ de níveis de cinza em k sub-intervalos é utilizar a razão $\Delta I = \frac{I}{k}$, de forma que $\mathcal{I} = [a, a + \Delta I) \cup [a + \Delta I, a + 2\Delta I) \cup \cdots \cup [a + i\Delta I, a + (i + 1)\Delta I) \cup \cdots \cup [a + (k-1)\Delta I, a + k\Delta I = b]$, como mostra Figura 11.6. Em cada célula podemos escolher como o valor de quantização a média aritmética da intensidade mínima e da intensidade máxima de cada subintervalo. Este tipo de quantização é conhecida como **uniforme**. Observe que os valores de quantização são escolhidos após o particionamento do gamute em células de quantização.

Estendendo para cores constituídas por 3 componentes, particionamos cada componente independentemente. Os planos de partição definem no espaço de cores várias células (cúbicas) de quantização, cujo valor de quantização pode ser dado pelo centróide de cada célula. Por legibilidade, Figura 11.7 mostra 5 fatias do cubo de cores particionado uniformemente.

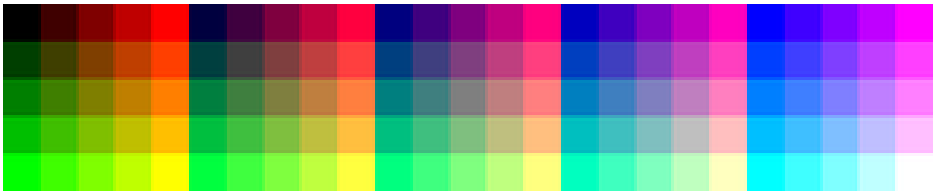


Figura 11.7: Particionamento uniforme de um cubo de cor.

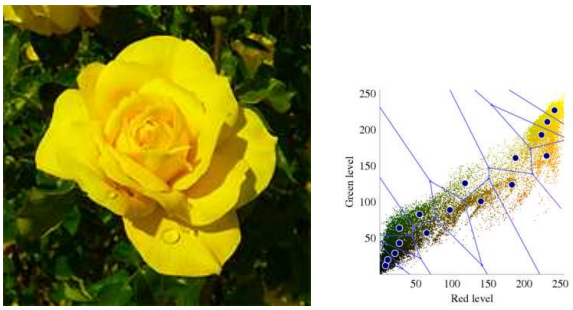
11.2.2 Quantização Adaptativa

Embora simples, a quantização uniforme tem a sua aplicação muito limitada. Dependendo da distribuição de frequências das cores, podemos ter uma ou mais células de quantização às quais nenhuma cor do gamute da imagem esteja associada, o que pode resultar em uma ocorrência maior dos efeitos de contornos falsos. Se fizermos um particionamento adaptativo, em que o tamanho de célula de quantização seja função da frequência de ocorrência das cores, podemos diminuir a medida de distorção dada pela Eq. 11.1.

O particionamento do gamute de uma imagem em K células de quantização pode ser feito com base em seu **histograma**. Pode-se, por exemplo, escolher as K cores de maior frequência de ocorrência e definí-las como K níveis de quantização, a priori. Neste caso, a célula de quantização seria a **célula de Voronoi** do **diagrama de Voronoi** dos K níveis de quantização, isto é, cada cor c numa célula de quantização de nível de quntização q_i deve ter o seu erro de quantização $|c - q_i| < |c - q_j|$ em relação a qualquer outro nível de quantização q_j, $i \neq j$. Esta técnica de quatização é conhecida como **algoritmo de populosidade** (*popularity algorithm*), proposto por Heckbert em 1982. Figura 11.8 ilustra o particionamento de uma imagem colorida, sem a componente azul, em regiões de Voronoi a partir de 16 níveis de quantização pré-selecionados.

11.2.3 Quantização por Corte Mediano

O algoritmo de populosidade desconsidera as cores muito distintas dos valores de quantização e que apresentam baixa frequência de ocorrência na imagem, podendo eliminar alguns efeitos visuais relevantes tais como *highlighting*. Como uma forma de contornar esta deficiência, Heckbert apresentou em 1982 o algoritmo de corte mediano (*median cut algoritm*), que consiste em particionar recursivamente o gamute de cores pela mediana da componente que tiver maior faixa de variações. O processo de recursão continua até que não existam mais cores da imagem original contidas em alguma



(a) (b)

Figura 11.8: Particionamento adaptativo: (a) imagem RG, sem o canal azul; (b) região de Voronoi (Fonte: http://en.wikipedia.org/wiki/Color_quantization).

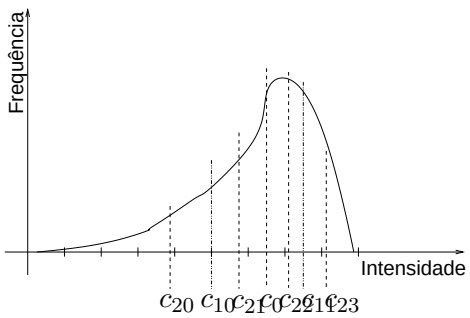
célula, ou até que seja alcançado o número pré-estabelecido de células de quantização. Após determinadas as células de quantização é computado o nível de quantização para cada uma delas. Ele pode ser a média aritmética, ou média ponderada em relação à frequência de ocorrência, das cores do gamute da imagem contidas na célula

Diferentemente do método de populosidade que parte da escolha dos níveis de quantização para determinar células de quantização, a técnica de corte mediano computa inicialmente as células de quantização para então estabelecer o nível de quantização para cada célula.

Figura 11.9 ilustra o resultado de uma partição por corte mediano de uma imagem em tons de cinza. Inicialmente, calcula-se a mediana c_0 que divide o intervalo de intensidades em dois sub-intervalos $[0, c_0]$ e $[c_0, c_{max}]$, onde c_{max} é a intensidade máxima. Aplica-se, então, o mesmo método de subdivisão para cada sub-intervalo e obtém-se $[0, c_{10}]$, $[c_{10}, c_0]$, $[c_0, c_{11}]$ e $[c_{11}, c_{max}]$. Em seguida, calcula-se a mediana de cada um destes intervalos c_{20}, c_{21}, c_{22} e c_{23} para alcançar as 8 células de quantização.

Para exemplificar, vamos quantizar a imagem da Figura 11.3.(a) em 4 níveis de quantização. A primeira mediana poderia ser o nível de cinza 120 que particiona o intervalo em $[0, 120]$ e $[120, 255]$. O primeiro intervalo com 56 ocorrências e o segundo com 52 ocorrências. Em seguida, particionamos cada um destes intervalos em $[0, 65]$ e $[65, 120]$, $[120, 145]$ e $[145, 255]$, respectivamente. Tendo as 4 células de quantização, escolhemos a média ponderada dos valores em cada célula como o seu nível de quantização. Como resultado, temos a imagem quantizada apresentada na Figura 11.9.(b).

Quando se trata de imagens coloridas, o procedimento é o mesmo. Só



(a)

177	126	126	59.3	59.3	59.3	94.8	177	177	177	177	177
177	94.8	94.8	94.8	59.3	59.3	94.8	177	177	177	94.8	94.8
177	94.8	59.3	59.3	59.3	59.3	94.8	59.3	59.3	59.3	59.3	94.8
177	94.8	94.8	59.3	59.3	59.3	59.3	94.8	94.8	94.8	59.3	94.8
177	94.8	126	126	59.3	59.3	94.8	94.8	94.8	94.8	126	94.8
177	94.8	202	126	94.8	59.3	59.3	59.3	126	126	126	94.8
126	126	126	177	177	59.3	126	126	126	126	126	126
126	177	177	177	59.3	59.3	59.3	177	177	177	177	126
126	126	126	177	177	177	126	126	126	126	126	126

(b)

Figura 11.9: Particionamento recursivo por corte mediano: (a) histograma; (b) imagem da Figura 11.3.(a) quantizada (Fonte: http://en.wikipedia.org/wiki/Color_quantization).

que, como tem mais de uma dimensão, procura-se escolher sempre o maior lado da célula para subdividir, como ilustra Figura 11.10.

11.3 Técnicas de Redução de Contornos Falsos

A percepção de contornos falsos depende do número de níveis de quantização. Em geral, para imagens em níveis de cinza, com uma quantização em 8 *bits* consegue-se evitar a percepção destes contornos; e para imagens coloridas, basta uma quantização em 24 *bits*, 8 *bits* para cada componente. Nesta seção apresentamos algumas soluções para casos em que tais números de níveis de quantização não são suportados, com a finalidade de aumentar o número de cores percebidas e atenuar os efeitos de **contornos falsos**, através da redistribuição dos erros decorrentes da quantização.

11.3.1 Aproximação do Meio-Tom

A técnica de meio-tom (*halftoning*) é uma técnica muito utilizada no passado para impressão de jornais, revistas e livros com apenas dois tons: preto e branco. A técnica consiste essencialmente em utilizar pontos pretos de tamanhos variados sobre o fundo branco de tal sorte que, quando olhados a uma certa distância, dão a sensação de vários níveis de tons de cinza. Observe na imagem detalhada da Figura 11.11 a variação dos pontos pretos. Nos jornais a densidade dos pontos varia de 60 a 80 pontos por polegada e na impressão de revistas e livros de qualidade melhor, é comum encontrar densidade entre 110 a 200 pontos por polegada.

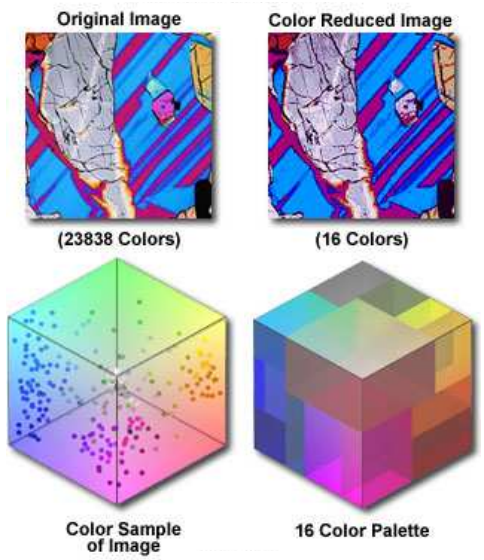


Figura 11.10: Quantização de uma imagem colorida por corte mediano (Fonte: http://micro.magnet.fsu.edu/primer/java/digitalimaging/processing/colorreduction/index.html.

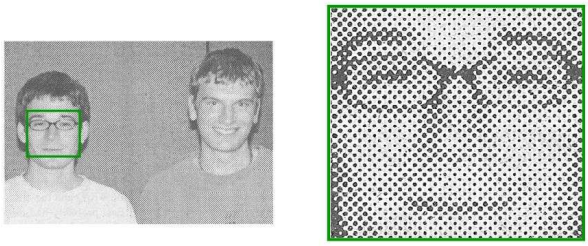


Figura 11.11: Imagem em meio-tom analógico (Fonte: http://cs.guc.edu.eg/courses/_Winter2007/DMET501/OnlineTutorial/ImageChapter.html.

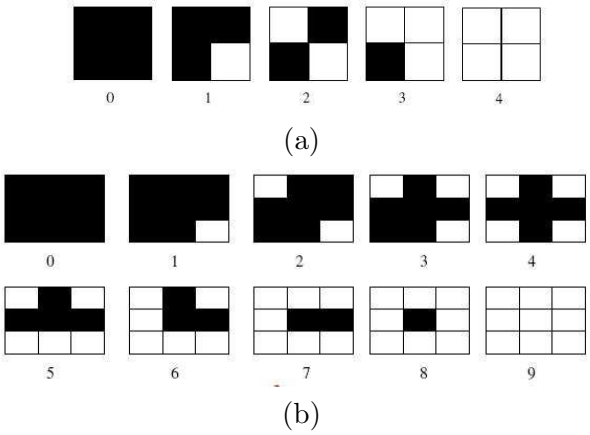


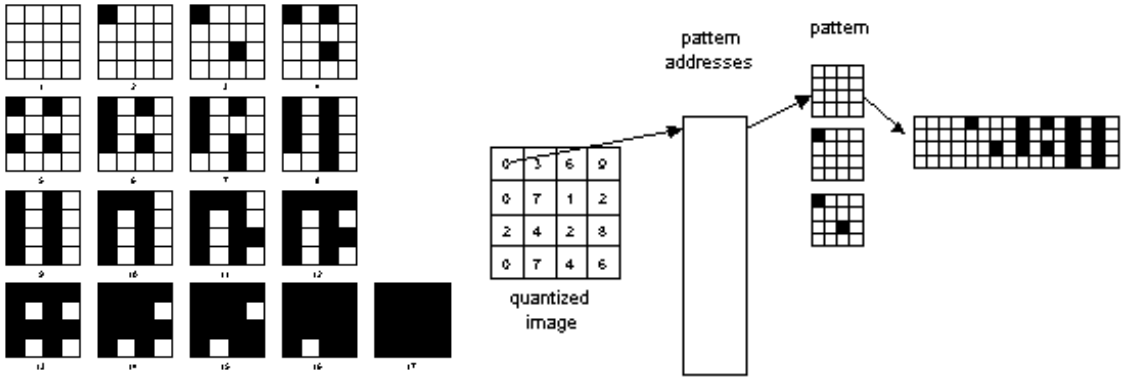
Figura 11.12: Meio-tom digital: (a) 5 níveis de cinza; (b) 10 níveis de cinza (Fonte: http://cs.guc.edu.eg/courses/_Winter2007/DMET501/OnlineTutorial/ImageChapter.html).

O processo de meio-tom analógico utiliza uma câmera fotográfica especial. A imagem é refotografada num filme de alto contraste sobrepondo a ela uma tela reticulada, de forma que o filme só é sensibilizado nos pontos correspondentes aos nós do reticulado. Cada nó do reticulado, por sua vez, atua como uma lente que focaliza a luz proveniente da imagem. Assim, a área a ser sensibilizada depende do valor da luminância recebida.

Os dispositivos utilizados para exibição de imagens digitais só conseguem, no entanto, produzir pontos de mesmo tamanho, como os *pixels* dos monitores CRT e os pontos de uma impressora matricial. Uma solução adotada é simular o método de meio-tom através da **aglomeração de** *pixels* (*clustered-dots*) em diferentes padrões de níveis de cinza. Um grupo de $n \times n$ *pixels* consegue emular $n^2 + 1$ níveis de cinza. Por exemplo, um aglomerado de 2×2 e um de 3×3 *pixels* consegue emular 5 e 10 níveis de cinza, respectivamente (Figura 11.12). Observe que há um compromisso entre a resolução espacial e a resolução radiométrica a ser emulada, uma vez que a sensação de diferentes tons se deve a diferentes relações entre áreas escuras e áreas brancas.

Certamente, para $n \times n$ *pixels*, quanto maior é n, maior é o número de possibilidades que temos para aglomerar estes *pixels* para formar os $n^2 + 1$ padrões de cinza. Algumas regras úteis para dispor estes *pixels* num padrão são:

1. não devem produzir artefatos visuais numa área grande de mesma



(a) (b)

Figura 11.13: Meio-tom digital: (a) padrões; (b) substituição de um nível de quantização por um padrão (Fonte: http://www.geocities.com/ResearchTriangle/Thinktank/5996/techpaps/introip/manual04.html).

intensidade;

2. o aglomerado dos *pixels* deve formar uma sequência monotônica crescente. Um *pixel* que fizer parte de um nível de quantização i, faz parte dos níveis (de cinza) maiores que i;

3. o aglomerado dos *pixels* deve crescer do centro para fora para dar impressão de pontos de tamanhos variáveis; e

4. os *pixels* escuros devem ficar adjacentes.

Figura 11.13.(a) ilustra um conjunto de 17 padrões correspondentes a 17 níveis de cinza quantizados e Figura 11.13.(b) mostra como se substitui o nível quantizado em cada *pixel* de uma imagem por um padrão composto de 16 *pixels*. Observe que a imagem final tem uma resolução espacial 16 vezes maior do que a imagem original.

Figura 11.14 ilustra uma imagem quantizada em 10 níveis de cinza exibida pela técnica de meio-tom digital com uma resolução espacial 3 vezes maior.

11.3.2 Dithering

A aglomeração dos *pixels* é importante para dispositivos que não conseguem exibir pontos isolados, como impressoras *laser*. No caso dos monitores CRT

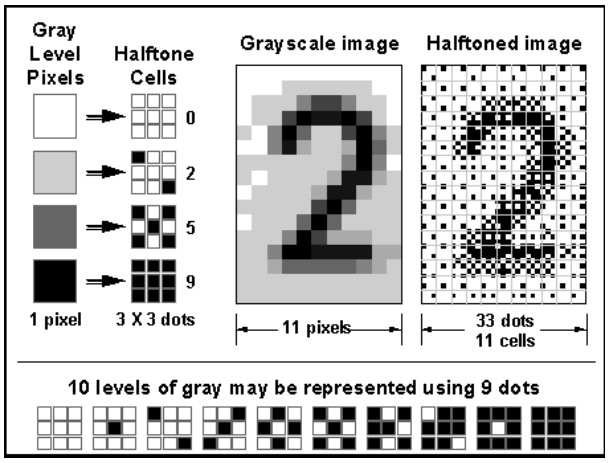


Figura 11.14: Imagem quantizada em 10 níveis de cinza exibida pela técnica de meio-tom digital (Fonte: http://fhctech.org/fhc/imaging/halftone.htm).

que conseguem exibir os *pixels* isoladamente, os *pixels* "escuros" não precisam estar adjacentes para produzir uma dada percepção de tom de cinza. Quando se aproxima o método de meio-tom por conjunto de *pixels* escuros não necessariamente adjacentes, dizemos então, que a aproximação é por **dispersão de pixels**. A eliminação da restrição dos *pixels* escuros ficarem adjacentes permite dar à imagem uma aparência com maiores variações nos níveis de cinza.

A técnica mais conhecida de dispersão de *pixels* é a de *dither*, que consiste em introduzir ruídos na imagem para que o erro de quantização seja distribuído de forma aleatória. Com isso, evita-se a formação de padrões de textura originalmente inexistentes nas regiões de intensidade constante. O erro é adicionado à imagem através do arrendondamento dos valores de cada *pixel* após a comparação destes valores com limiares pré-estabelecidos. Uma forma compacta de representar os limiares a serem utilizados para inserir os erros é através de uma máscara de ordem n contendo a sequência dos n^2 valores de quantização a serem utilizados como limiar de comparação em cada *pixel*. Os valores de quantização, por sua vez, podem ser determinados por uma das técnicas apresentadas na seção 11.2. Ao ladrilharmos uma imagem com uma destas máscaras, podemos decidir eficientemente o erro de quantização a ser introduzido ao fizermos o arredondamento do valor original do *pixel* para um dos valores exibíveis.

Vamos aplicar o seguinte padrão de ordem dos valores de quantização para gerar uma imagem binária que aparenta conter 4 níveis de cinza na

1	1	1	0	0	0	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	0	0
1	1	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	1	1	1	1	1	0
0	0	1	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1

Figura 11.15: *Dithering* ordenado: imagem da Figura 11.3.(a) representada por 2 níveis de intensidade, mas aparentando 4 níveis.

imagem da Figura 11.3. Os quatro níveis de quantização que utilizaremos são $\{59.3, 94.8, 126, 177\}$:

0	1
2	3

Isso corresponde a ladrilhar a seguinte máscara

59.3	94.8
126	177

sobre a imagem e comparar, *pixel* a *pixel*, os valores dos *pixels* com o limiar da máscara. Se o valor for menor que o limiar, atribui-se "0" ao *pixel*, senão, o valor "1" é setado. O procedimento é sintetizado no seguinte pseudo-código:

Input: Imagem em tons de cinza I, máscara $n \times n$
Output: Imagem binária filtrada O
foreach $0 \leq x \leq x_{max}$ **do**
  foreach $0 \leq y \leq y_{max}$ **do**
    i = x mod n ;
    j = y mod n ;
    if *I(x,y) > D(i,j)* **then**
      O
    (x,y) = 1 ;
    else
      O
    end
    (x,y) = 0 ;
  end
end

Algoritmo 8: Algoritmo de *dithering* ordenado.

Figura 11.15 mostra o resultado deste processamento.

Bayer mostrou em 1973 que uma dispersão feita de forma que a distribuição de *pixels* escuros correspondentes a cada nível de cinza ser mais uniforme possível minimiza a formação de texturas em regiões da imagem com intensidade constante. Ele estabeleceu ainda condições necessárias e suficientes para construir matrizes de *dither* de qualquer ordem. As matrizes são conhecidas como **matrizes de Bayer**.

A matriz de Bayer de ordem 2 é dada por

$$D^{(2)} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

Matrizes de Bayer de ordem superior podem ser obtidas de forma recorrente usando a relação

$$D^{(n)} = \begin{bmatrix} 4D^{(n/2)} + D_{00}^{(2)} U^{n/2} & 4D^{(n/2)} + D_{01}^{(2)} U^{n/2} \\ 4D^{(n/2)} + D_{10}^{(2)} U^{n/2} & 4D^{(n/2)} + D_{11}^{(2)} U^{n/2} \end{bmatrix},$$

com

$$U^{(n)} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$

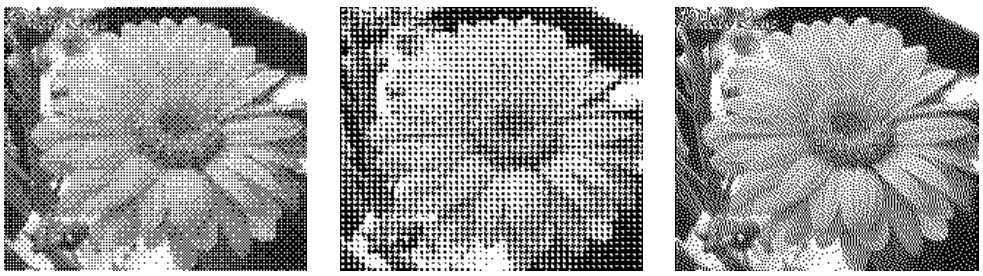
Por exemplo, a matriz de Bayer de ordem 4 é dada por

2	16	3	13
10	6	11	7
4	14	1	15
12	8	9	5

Figura 11.16 apresenta uma mesma imagem filtrada com (a) *dither* ordenado e (b) *dither* de Bayer.

Em geral, temos vários tipos de matrizes de *dither*. Só para citar, uma matriz de *dither* com a característica de distribuir uniformemente *pixels* escuros em todas as direções é a matriz conhecida como quadrado mágico, pelo fato de que as somas dos elementos das colunas, das linhas e das diagonais são todas iguais a 34

$$\begin{bmatrix} 0 & 6 & 9 & 15 \\ 11 & 13 & 2 & 4 \\ 7 & 1 & 14 & 8 \\ 12 & 10 & 5 & 3 \end{bmatrix}$$



(a) (b) (c)

Figura 11.16: *Dithering*: (a) ordenado; (b) de Bayer; (c) Floyed-Steinberg. (Fonte: http://www.pl32.com/tutorial/sraster/sraster.htm).

11.3.3 Difusão de Erro

Diferentemente das técnicas da seção 11.3.2, que incluem erros de aproximação em cada *pixel* ao "oscilar" de forma aleatória o seu valor, as técnicas por difusão de erro propagam tais erros para os *pixels* adjacentes procurando balanceá-los. Assim, ao selecionar o nível de quantização de uma *pixel*, leva-se em conta a soma do valor original do *pixel* e os erros herdados. Além disso, o erro de quantização em relação a esta soma é computada e distribuído para os *pixels* vizinhos. As técnicas existentes na literatura diferem na distribuição deste erro de quantização.

A técnica mais clássica é a proposta por Floyd e Steinberg em 1976 que utiliza a seguite matriz de distribuição

$$Peso = \frac{1}{16}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 7 \\ 3 & 5 & 1 \end{bmatrix}$$

sempre centrada no *pixel* em processamento. Esta matriz de distribuição nos indica que será distribuído

- $\frac{7}{16}\sum_i \sum_j Peso_{i,j}$ do erro de aproximação para o *pixel* do lado direito;
- $\frac{5}{16}\sum_i \sum_j Peso_{i,j}$ do erro de aproximação para o *pixel* imediatamente abaixo;
- $\frac{3}{16}\sum_i \sum_j Peso_{i,j}$ do erro de aproximação para o *pixel* abaixo, em diagonal no lado esquerdo; e
- $\frac{1}{16}\sum_i \sum_j Peso_{i,j}$ do erro de aproximação para o *pixel* abaixo, em diagonal no lado direito.

Figura 11.16.(c) mostra uma imagem em 2 níveis de cinza com filtragem de Floyed-Steinberg. Em pseudo-código, temos o seguinte procedimento:

Input: Imagem em tons de cinza I
Output: Imagem em tons de cinza I com erro difundido
foreach $0 \leq y \leq y_{max}$ **do**
  foreach $0 \leq x \leq x_{max}$ **do**
    antigo = I(x,y) ;
    novo = nível de cinza que se aproxima de I(x,y) ;
    I(x,y) = novo ;
    erro = antigo - novo ;
    I(x+1,y) = I(x+1,y) + $\frac{7}{16}$*erro ;
    I(x-1,y+1) = I(x-1,y+1) + $\frac{3}{16}$*erro ;
    I(x,y+1) = I(x,y+1) + $\frac{5}{16}$*erro ;
    I(x+1,y+1) = I(x+1,y+1) + $\frac{1}{16}$*erro ;
  end
end

Algoritmo 9: Algoritmo de Floyed-Steinberg.

Outras duas matrizes de distribuição de erro encontradas na literatura são

$$Peso = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{bmatrix}$$

$$Peso = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$$

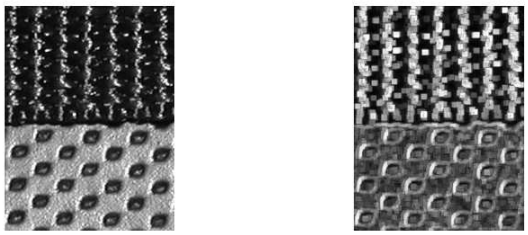
Capítulo 12

Textura

O objetivo deste capítulo é introduzir os principais conceitos de textura em Computação Gráfica. Após a leitura deste capítulo, você deve ser capaz de

- relacionar o conceito de "textura" empregado em Computação Gráfica e Processamento de Imagens.
- diferenciar a aplicação do modelo de iluminação e o modelo de textura em síntese de imagens para obter imagens com aparência mais próxima possível da nossa percepção.
- conceituar os termos: *texel*, espaço de textura, mapeamento de textura, pré-imagem de um *pixel*.
- relacionar espaço de textura com outros espaços.
- descrever diferentes formas de geração de textura e como os atributos das texturas podem afetar os parâmetros ou resultados de um modelo de iluminação.
- distinguir método direto e método inverso de mapeamento de textura.
- descrever um fluxo típico de mapeamento de textura bi-dimensional.
- entender problemas de alinhamento entre *pixels* e *texels* no mapeamento de textura e algumas das suas soluções.

De modo geral, textura é a característica de uma superfície que pode ser percebida pelos estímulos sensoriais visuais e tácteis. Quando tocarmos ou



(a) (b)

Figura 12.1: Análise de imagens: (a) imagem original e (b) imagem segmentada em duas regiões (Fonte: http://www.mathworks.com/products/image/demos.html?file=/products/demos/shipping/images/ipextexturefilter.html).

olharmos para um objeto, os nossos órgãos dos sentidos nos permitem distinguir se a sua superfície é lisa, rugosa, macia ou áspera. Uma emulação típica deste reconhecimento em sistemas computacionais é particionar a imagem em regiões com propriedades similares usando medidas **estatísticas**, **estruturais**, **espectrais**, ou combinação das três. Os resultados obtidos são valores numéricos através dos quais é possível, em muitos casos, fazer um sistema digital "sentir" a textura dos objetos contidos na imagem (Seção 1.1). Neste contexto, uma textura é entendida como uma representação que facilite a **identificação digital** das propriedades das superfícies contidas em uma imagem (Figura 12.1).

Outro interesse em sistemas de informação gráfica é produzir imagens mais próximas da nossa percepção, ou seja, aquelas que consigam despertar sensações de distintos "aspectos de superfície". Utilizando o modelo de iluminação de Phong (Eq. 8.9), as imagens ficaram esteticamente melhores do que as geradas pelos modelos anteriores, mas tinham uma aparência plástica. A sensação produzida era muito aquém do realismo desejado. Isso é porque a percepção da aparência de uma superfície está intimamente relacionada com a sua interação com as radiações luminosas. Além do material da superfície ser anisotrópico, a geometria da superfície é usualmente bem irregular. Se uma superfície plana for perfeitamente polida, as radiações seriam coerentemente refletidas numa mesma direção (Figura 12.2.(a)); caso contrário, as radiações luminosas que chegam ao observador são bem "irregulares", aparentando que em alguns pontos a intensidade luminosa refletida é maior e em outros, menor. Neste último caso, tem-se a sensação de uma

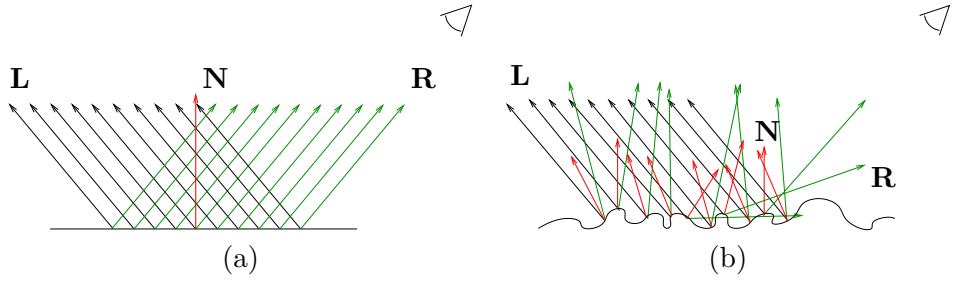


Figura 12.2: Interação luz – superfície.

aparência áspera ou rugosa que um modelo de iluminação simplificado, como o de Phong, ainda não leva em conta (Figura 12.2.(b)).

Isso motivou pesquisas em diferentes paradigmas para considerar estes "detalhes" de interações. Técnicas como traçado de raios, em inglês *ray tracing*, e radiosidade, em inglês *radiosity*, propostas na década 1980 se baseiam no paradigma de iluminação global, acreditando que o realismo possa ser alcançado considerando todas as possíveis interações de radiações/energias luminosas entre os objetos em uma cena. Em paralelo, com o trabalho pioneiro de Catmull, iniciou-se o desenvolvimento de técnicas de textura. Estas técnicas, a preço de um maior consumo de memória, são muito mais atrativas em termos de custo temporal. Além disso, elas aproveitam vários passos do fluxo de síntese de imagens baseado em modelos de iluminação local já bem consolidados na época.

A engenhosidade da solução está a forma como a "modelagem de aparência" é integrada ao fluxo de síntese de imagens. Ao invés de modelar "microscopicamente" a geometria e a refletância das superfícies para computar de forma mais precisa as direções das radiações refletidas, "alteram-se" os parâmetros do modelo de iluminação através de uma **textura**. Os resultados visuais são, em muitos casos, similares aos da nossa percepção. Neste contexto, uma textura consiste de um conjunto de valores que afetam os atributos de cada fragmento de uma imagem, com a finalidade de aumentar realismo sem onerar o procedimento sob o ponto de vista temporal. Estes valores pré-computados são tipicamente organizados em um arranjo multidimensional de *texels* (*texture elements*) em um espaço próprio, denominado **espaço de textura**. Figura 12.3 ilustra o mapeamento de uma textura bi-dimensional (s, t) sobre a superfície de um objeto, proporcionando sensação de um tampo de mármore.

O foco deste capítulo é a textura em síntese de imagens. Veremos na seção 12.1 vários tipos de textura e o seu modelamento. Em seguida, veremos

Figura 12.3: Síntese de imagens: mapeamento de textura.

na seção 12.2 uma arquitetura geral de mapeamento do espaço de textura sobre os fragmentos de uma imagem de interesse para enriquecer os seus detalhes. Pelo fato do espaço de textura e do de imagem serem discretos, neste mapeamento inerem problemas de alinhamento "exato" entre os *pixels* e os *texels*. Analisaremos com mais detalhes soluções para mapeamento de imagens 2D na seção 12.3, por este ser o mais popular.

12.1 Textura

Grosso modo, podemos dizer que textura em Computação Gráfica é uma técnica simples e barata para modelar a aparência visual de superfícies através da modulação dos valores dos parâmetros de um modelo de iluminação local. Vimos na seção 8.3 que a cor em cada amostra de uma superfície é computada como soma de três componentes: ambiente ($I_a = k_a I_a$), difusa ($I_d = k_d I_d(\mathbf{N} \cdot \mathbf{L})$) e especular (por exemplo a de Phong, $I_s = k_s I_s(\mathbf{V} \cdot \mathbf{R})^n$). Além de dependerem do material (k_*) e da fonte luminosa (L_*), as componentes podem depender da geometria da superfície (vetor normal $\mathbf{N}$) e da posição do observador ($\mathbf{P}$). Assim, temos várias alternativas para alterar a cor de cada amostra. Na ordem de popularidade destas alternativas, temos

mapeamento de cor : substitui a componente difusa da equação de iluminação pela cor especificada em textura, de forma a reproduzir a aparência da textura (Figura 12.4.(a));

mapeamento de reflexões especulares , em inglês *environment mapping*: substitui a componente especular da equação de iluminação pela

cor especificada em textura, de forma a produzir uma aparência especular (Figura 12.4.(b));

mapeamento de bossagem , em inglês *bump mapping*: perturba a direção dos vetores normais com o valor especificado em textura, de forma a gerar uma aparência rugosa (Figura 12.4.(c));

mapeamento de deslocamento , em inglês *displacement mapping*: perturba a posição do ponto da superfície na direção do vetor normal com o valor especificado em textura, de forma a causar sensação de protuberância (Figura 12.4.(d));

mapeamento de transparência : perturba a opacidade da superfícies, de maneira a dar impressão de translucidez (Figura 12.4.(e)).

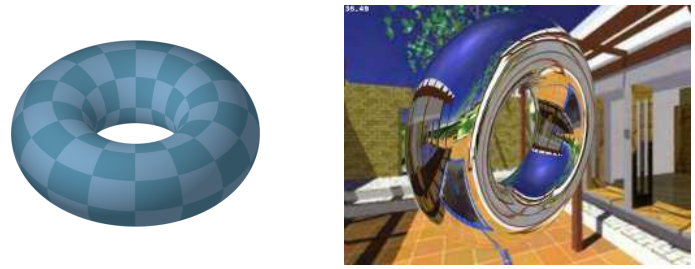
Para distinguir a modelagem de aparência de uma superfície da modelagem da sua geometria, é introduzido um novo espaço, denominado **espaço (homogêneo) de textura**. A dimensão deste espaço pode ser uni-, bi- ou tridimensional, com uso de até quatro **coordenadas homogêneas** s, t, r e q, e os seus pontos são passíveis a todas as transformações geométricas apresentadas no capítulo 4, como os pontos do espaço geométrico. Além disso, podemos aplicar tais transformações para mudar as coordenadas dos pontos entre os dois espaços. Finalmente, é comum normalizar as coordenadas dos pontos no espaço de textura para o intervalo $[0, 1]$. Isso facilita o mapeamento entre os espaços.

Veremos nesta seção alguns tipos de textura.

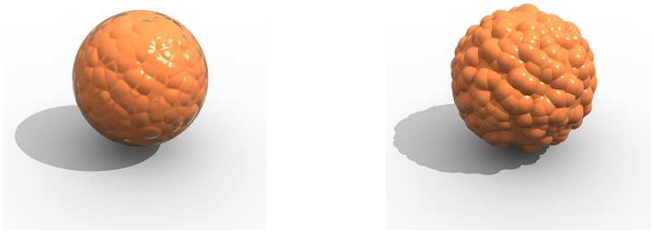
12.1.1 Imagens

Em aplicações de tempo-real, utiliza-se predominantemente imagens sintéticas ou imagens capturadas por dispositivos, como câmaras fotográficas, vídeos e escaneadores. Cada uma dessas imagens é, de fato, um arranjo de radiações refletidas pelas superfícies reais em uma cena. A idéia consiste, então, em utilizar estas imagens para obter as radiações que deveriam ser refletidas em cada ponto de uma imagem sintética, ao invés de determiná-las numericamente por meio de equações de modelos de iluminação, como as apresentadas na seção 8.3.

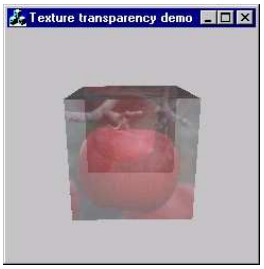
A forma mais direta para transformar uma imagem digital de $m \times n$ *pixels* em uma textura de $m \times n$ *texels* é associar a cada *texel* um *pixel*. Figura 12.5 exemplifica o uso de uma foto na dimensão 400×322 como textura de cor. Em primeiro lugar, define-se um referencial do espaço de textura (s, t) de 400×322 *texels*. Em seguida, normaliza-se as coordenadas



(a) (b)

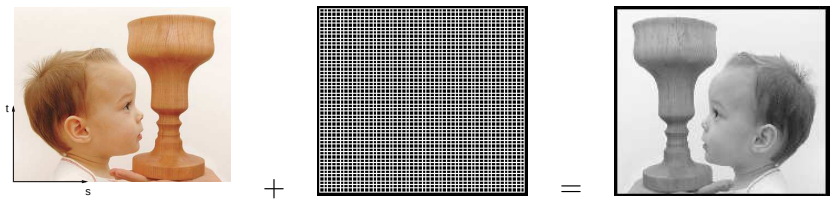


(c) (d)



(e)

Figura 12.4: Tipos de textura: (a) mapeamento de cor; (b) mapeamento de ambiente em um toro; (c) mapeamento de bossagem; (d) mapeamento de deslocamento em uma esfera (Fonte: http://www.spot3d.com/vray/help/150R1/examples_displacement.htm); e (e) mapeamento de transparência em um cubo (Fonte: http://www.codeguru.com/cpp/g-m/opengl/article.php/c2687/) .



(a) (b) (c)

Figura 12.5: Textura de cor: (a) imagem digital; (b) forma geométrica em malha retangular; e (c) retângulo texturizado.

s e t para o intervalo $[0, 1]$, de forma que o espaçamento entre dois *texels* adjacentes na direção s e na direção seja, respectivamente, $\frac{1}{400} = 0.0025$ e $\frac{1}{322} \sim 0.0031$ como mostra Figura 12.5.(a). Após esta normalização, podemos referenciar cada elemento da imagem digital por valores numéricos. "Projetando ortograficamente" esta textura sobre um retângulo de mesmas dimensões da textura, ilustrado na Figura 12.5.(b), "transferimos" para a superfície do retângulo a imagem. A sensação que a imagem final, mostrada na Figura 12.5.(c), nos produz é uma cena 3D contendo um cálice e o perfil de uma criança.

Observamos que em Medicina aparelhos, como ressonância magnética e tomografia computatorizada, conseguem formar uma sequência de fatias de imagens internas de corpos humanos, alinhadas entre si. Esta sequência de imagens é conhecida por **imagem 3D**. Figura 12.6 ilustra uma imagem 3D.

12.1.2 Textura de Perturbação

Uma textura bi-dimensional pode ser um conjunto de ruídos que "perturbem ligeiramente" os valores de um parâmetro do modelo de iluminação, como a **textura de bossagem**, em inglês *bump map*, proposto em 1978 pelo J. Blinn. Esta textura teve como principal motivação reduzir a aparência "plástica" das imagens geradas com o modelo de Phong. Cada *texel* (s, t) da textura contém um valor $b(s, t)$, que é utilizado tanto para deslocar ligeiramente a direção do seu vetor normal quanto a posição de um ponto da superfície. Se projetarmos ortograficamente a textura sobre um retângulo e mapearmos os valores de ruído em níveis de cinza, veremos um mosaico de várias tonalidades de cinaza (Figura 12.7.(a)). Quanto mais escura for a tonalidade, menor será o ruído ou a perturbação, como ilustra Figura 12.7.(c). Os *texels* são também utilizados para perturbar a posição dos pontos da superfície. Veremos na seção 12.2.4 que, junto com a perturbação de vetores

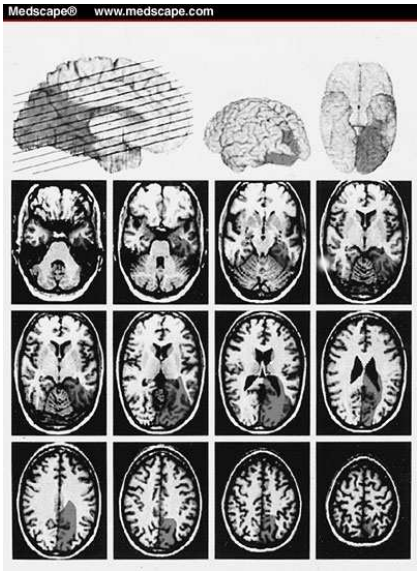
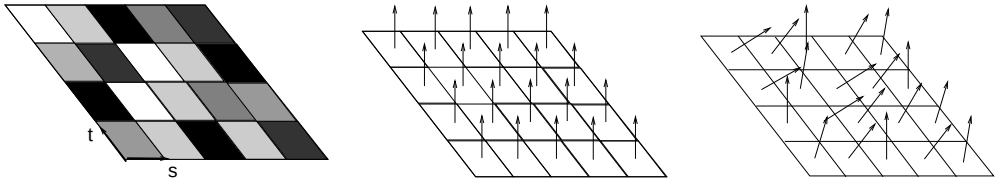


Figura 12.6: Imagem 3D (Fonte: www.medscape.com).



(a) (b) (c)

Figura 12.7: Textura de *bump*: (a) visualização do mapa; (b) vetores originais; e (c) vetores perturbados.

normais, consegue-se produzir uma variedade de efeitos realistas de terreno.

Exemplos de textura são apresentados na Figura 12.8.

12.1.3 Textura de Sombra

Sombra é fundamental para aumentar realismo de uma cena. Em 1978, Lance Williams propôs uma maneira eficiente de criar efeitos de sombra com uso de **textura de sombra**. Uma textura de sombra contém, essencialmente, a distância d de cada amostra (s, t) em relação a **uma** fonte luminosa.

Para gerar uma textura de sombra, basta posicionar a câmera na fonte

(a) (b) (c)

Figura 12.8: Exemplos de textura de *bump*.

luminosa com os raios de projeção alinhados com as radiações luminosas e determinar a matriz de projeção correspondente. Se a fonte de luz for distante (Seção 8.1), a projeção M_{par} será paralela (Figura 12.9.(a)). A distância r em cada *texel* (s,t) pode ser computada através de

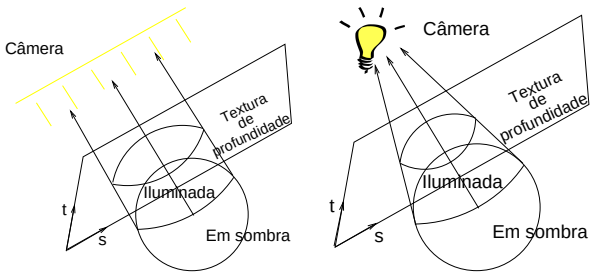
$$\begin{bmatrix} s \\ t \\ r \end{bmatrix} \sim \begin{bmatrix} s \\ t \\ r \\ 1 \end{bmatrix} = M_{par} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Se a fonte de luz for pontual (Seção 8.1), a projeção M_{per} será perspectiva (Figura 12.9.(b)). Neste caso, as coordenadas no espaço de textura e no espaço 3D são relacionadas por

$$\begin{bmatrix} s \\ t \\ r \end{bmatrix} \sim \begin{bmatrix} qs \\ qt \\ qr \\ q \end{bmatrix} = M_{per} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

12.1.4 Textura de Reflexão

É muito utilizada para produzir, em tempo-real, aparência de reflexões especulares em superfícies metálicas bem polidas. A idéia básica consiste em pré-computar a cor a ser refletida por cada amostra da superfície quando ela é observada na direção **d** e organizar essas cores em uma **textura de reflexão**. Na verdade, há várias formas para gerar uma textura especular. Ela poderia ser uma foto convencional de uma esfera espelhada, altamente reflexiva, como ilustra a Figura 12.10.(b), ou uma foto de uma câmera de lente de peixe. No primeiro caso, a câmera apareceria na foto (observe o fotógrafo na imagem da Figura 12.10.(b)) e no segundo caso, os efeitos



(a) (b)

Figura 12.9: Textura de sombra: (a) paralela e (b) perspectiva.

(a) (b) (c)

Figura 12.10: Efeito de especularidade: (a) cena sintética; (b) foto de uma bola de Natal; e (c) mapeamento (Fonte: http://www.debevec.org/ReflectionMapping/.

são aproximados, já que não há ainda uma lente com um campo de visão de 360^0. Uma terceira alternativa seria gerar computacionalmente uma textura de reflexão.

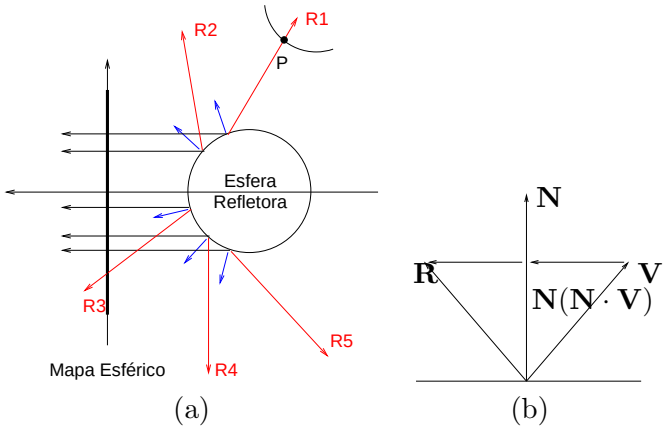
Seja $u^2 + v^2 + n^2 = 1$ uma esfera altamente especular, definida no referencial do observador, ou seja, referencial em que o observador esteja na origem e o raio de visão esteja na direção $\mathbf{V} = (0, 0, 1)$ (Figura 12.11.(a)). Neste caso, a direção do raio $\mathbf{R}$ que chega em cada ponto P e reflete na direção $\mathbf{V}$ é (Figura 12.11.(b))

$$\mathbf{R} \;=\; 2\mathbf{N}(\mathbf{N} \cdot \mathbf{V}) - \mathbf{V}, \tag{12.1}$$

onde $\mathbf{N}$ é o vetor normal da esfera em P.

Como a esfera é unitária e centrada na origem, as coordenadas do vetor normal em cada ponto coincidem com as coordenadas do ponto. Portanto, podemos escrever as componentes do raio $\mathbf{R}$ como

$$\begin{bmatrix} R_x \\ R_y \\ R_z \end{bmatrix} \;=\; 2 \begin{bmatrix} u \\ v \\ \sqrt{1-u^2-v^2} \end{bmatrix} \left(\begin{bmatrix} u \\ v \\ \sqrt{1-u^2-v^2} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) - \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$



(a) (b)

Figura 12.11: Textura de reflexão: (a) raios de reflexão e (b) geometria de reflexão.

$$= \begin{bmatrix} 2\sqrt{1-u^2-v^2}u \\ 2\sqrt{1-u^2-v^2}v \\ 2(1-u^2-v^2)-1 \end{bmatrix} \tag{12.2}$$

Observe que

$$\begin{aligned} R_x^2 + R_y^2 + (R_z+1)^2 &= 4(1-u^2-v^2)u^2 + 4(1-u^2-v^2)v^2 + (2(1-u^2-v^2)-1+1)^2 \\ &= 4(1-u^2-v^2)u^2 + 4(1-u^2-v^2)v^2 + (2(1-u^2-v^2))^2 \\ &= 4(1-u^2-v^2)u^2 + 4(1-u^2-v^2)v^2 + 4(1-u^2-v^2)^2 \\ &= 4(1-u^2-v^2)(u^2+v^2+1-u^2-v^2) \\ &= 4(1-u^2-v^2). \end{aligned} \tag{12.3}$$

Seguem-se então

$$\sqrt{R_x^2 + R_y^2 + (R_z+1)^2} = 2\sqrt{1-u^2-v^2}$$

e as seguintes igualdades

$$R_x = \sqrt{R_x^2 + R_y^2 + (R_z+1)^2}u \rightsquigarrow u = \frac{R_x}{\sqrt{R_x^2 + R_y^2 + (R_z+1)^2}}$$

$$R_y = \sqrt{R_x^2 + R_y^2 + (R_z+1)^2}v \rightsquigarrow v = \frac{R_y}{\sqrt{R_x^2 + R_y^2 + (R_z+1)^2}}.$$

Partindo do ponto P em direção do raio $\mathbf{R}$ é procurada na cena a superfície mais próxima com que o raio se intercepta. A cor da superfície na

Figura 12.12: Uma textura de reflexão sintética (Fonte: http://www.opengl.org/resources/code/samples/sig99/advanced99/notes/node179.html.)

intersecção, por exemplo o ponto P da Figura 12.11.(a), é armazenada na entrada (u, v) do arranjo de textura. O procedimento é repetido para todos os pontos $u^2 + v^2 \leq 1$. Observe que com este procedimento é gerado um mapa de *texels* no domínio $[-1, 1] \times [-1, 1]$. Para passar para o espaço de textura usual – domínio $[0, 1] \times [0, 1]$ – são aplicadas ainda as seguintes transformações de coordenadas do espaço (u, v) para o espaço (s, t):

$$\begin{aligned} s &= \frac{u+1}{2} = \frac{R_x}{2\sqrt{R_x^2 + R_y^2 + (R_z+1)^2}} + \frac{1}{2} \\ t &= \frac{v+1}{2} = \frac{R_y}{2\sqrt{R_x^2 + R_y^2 + (R_z+1)^2}} + \frac{1}{2} \end{aligned} \tag{12.4}$$

Figura 12.12 exemplifica uma textura sintética.

12.1.5 Mapa Cúbico

É uma alternativa para mapa esférico, tendo a vantagem de um custo bem menor. Um mapa cúbico é constituído por seis imagens/mapas obtidas através de 6 distintos ângulos a partir de um mesmo ponto do ambiente. Estas imagens correspondem às vistas que teríamos, olhando para a direção das quatro paredes, do teto e do chão de um cômodo em forma de um cubo, como mostra Figura 12.13. Cada uma dessas imagens é conhecida também como um **mapa de ambiente**. Ao invés de traçar raio por raio para obter as cores dos *texels* de um mapa esférico, pode-se aplicar o procedimento convencional de síntese de imagens para gerar cada uma das 6 imagens de um mapa cúbico, ou simplesmente utilizar as fotos capturadas por uma

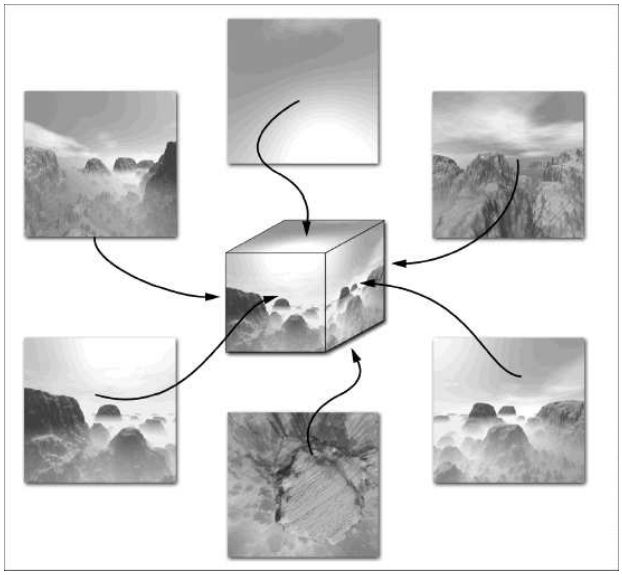


Figura 12.13: Um mapa cúbico (Fonte: http://www.developer.com/lang/other/article.php/2169281.)

câmera digital convencional. Certamente, o segundo procedimento é o mais rápido no estado-da-arte.

12.1.6 Textura Procedural

Em 1985, Perlin propôs uma função que mapeia diretamente uma n-upla coordenadas $\mathbf{P}$ de um ponto no espaço de n dimensões em um valor escalar. Textura gerada deste modo é conhecida como **textura procedural**. Ela se caracteriza por não precisar mais da definição de um mapa de textura para relacionar as coordenadas paramétricas (u, v) de ponto $\mathbf{P}(u, v)$ a um mapa de escalares. A função é bem genérica, podendo ser aplicada para modular a cor, o vetor normal ou qualquer outro parâmetro de iluminação. Duas propriedades devem ser satisfeitas por tal função:

1. as medidas estatísticas da região perturbada devem ser preservadas sob transformações rígidas (Seção 4.2.4); e

2. deve possuir uma banda limitada de frequência.

A primeira propriedade é para garantir que os efeitos de ruído sejam invariantes sob transformações rígidas e a segunda, assegurar que a variação do ruído não seja abrupta nem suave demais.

A proposta de Perlin é simples, eficiente e é uma boa aproximação para a função "ideal". Vamos detalhar o seu procedimento original para uma textura 3D. O seu gerador de ruído 3D começa com um volume discreto representado por um arranjo tri-dimensional. Para cada entrada (x_i, y_i, z_i) do arranjo é armazenada uma tripla de números aleatórios pré-calculados (a_i, b_i, c_i). Esta tripla é conhecida como **vetor de gradiente G**. Para uma arquitetura de 32 *bits* (MAXINT = 7fffffff) é típico utilizar a função Random(x) para gerar números aleatórios, em ponto flutuante, no intervalo

`Random(`*inteiro x*`)`

x = XOR_{bit}(shift(x,13),x) ;
x = AND_{bit}((x * (x * x * 15731 + 789221) + 1376312589),MAXINT);
r = 1.0 - x*2.0/MAXINT ;
return r ;

Função `Random(`x`)`.

[-1.0,1.0] e construir este vetor gradiente

$$\begin{aligned} a_i &= Random(64 * x_i + 56 * y_i + 71 * z_i) \\ b_i &= Random(73 * x_i + 79 * y_i + 83 * z_i) \\ c_i &= Random(89 * x_i + 97 * y_i + 101 * z_i). \end{aligned} \tag{12.5}$$

Dado um ponto $P = (x, y, z)$, determina-se o seu deslocamento Δd em relação à amostra $Q_5 = (\lfloor x \rfloor, \lfloor y \rfloor, \lfloor z \rfloor)$ da Figura 12.14, isto é,

$$\Delta d = (d_x, d_y, d_z) = (x - \lfloor x \rfloor, y - \lfloor y \rfloor, z - \lfloor z \rfloor)$$

e as outras 7 amostras adjacentes

$$\begin{aligned} Q_8 &= (\lfloor x \rfloor + 1, \lfloor y \rfloor, \lfloor z \rfloor) \\ Q_6 &= (\lfloor x \rfloor, \lfloor y \rfloor, \lfloor z \rfloor + 1) \\ Q_7 &= (\lfloor x \rfloor + 1, \lfloor y \rfloor, \lfloor z \rfloor + 1) \\ Q_1 &= (\lfloor x \rfloor, \lfloor y \rfloor + 1, \lfloor z \rfloor) \\ Q_2 &= (\lfloor x \rfloor, \lfloor y \rfloor + 1, \lfloor z \rfloor + 1) \\ Q_4 &= (\lfloor x \rfloor + 1, \lfloor y \rfloor + 1, \lfloor z \rfloor) \\ Q_3 &= (\lfloor x \rfloor + 1, \lfloor y \rfloor + 1, \lfloor z \rfloor + 1) \end{aligned}$$

Dados os ruídos de duas amostras adjacentes $R_A = (P - Q_A) \cdot \mathbf{G}_A = a_A x + b_A y + c_A z - (a_A x_A + b_A y_A + c_A z_A)$ e $R_B = (P - Q_B) \cdot \mathbf{G}_B = a_B x +$

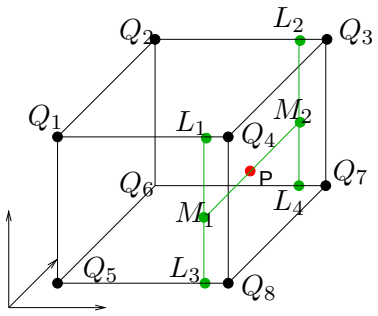


Figura 12.14: Interpolação trilinear.

$b_B y + c_B z - (a_B x_B + b_B y_B + c_B z_B)$, estima-se o ruído $R(t)$ de um ponto intermediário entre as amostras A e B pela interpolação linear

$$R_{AB}(t) = R_A + t(R_B - R_A). \tag{12.6}$$

Com uso de Δd, estima-se os fatores (f_x, f_y, f_z) de interpolação linear nas três direções por uma função cúbica $3t^2 - 2t^3$, ou seja,

$$\begin{aligned} f_x &= 3d_x^2 - 2d_x^3 \\ f_y &= 3d_y^2 - 2d_y^3 \\ f_z &= 3d_z^2 - 2d_z^3 \end{aligned}$$

Interpolando linearmente na direção x, teremos

$$\begin{aligned} R_{L_1} &= R_{Q_1} + f_x(R_{Q_4} - R_{Q_1}) \\ R_{L_2} &= R_{Q_2} + f_x(R_{Q_3} - R_{Q_2}) \\ R_{L_3} &= R_{Q_5} + f_x(R_{Q_8} - R_{Q_5}) \\ R_{L_4} &= R_{Q_6} + f_x(R_{Q_7} - R_{Q_6}) \end{aligned}$$

A partir de R_{L_1}, R_{L_2}, R_{L_3} e R_{L_4}, aplica-se interpolação linear na direção y

$$\begin{aligned} R_{M_1} &= R_{L_3} + f_y(R_{L_1} - R_{L_3}) \\ R_{M_2} &= R_{L_4} + f_y(R_{L_2} - R_{L_4}) \end{aligned}$$

e finalmente na direção z para obter o ruído de Perlin em P

$$RuidoPerlin(P) = R_P = R_{M_1} + f_z(R_{M_2} - R_{M_1}).$$

Utilizando este ruído, podemos por exemplo "perturbar" a cor (R, G, B) em um fragmento P, multiplicando cada componente da cor pelo valor $RuidoPerlin(P)$.

(a) (b) (c)

Figura 12.15: Ruído de Perlin na geração de textura de: (a) nuvens; (b) fogo; e (c) mármore (Fonte: http://mrl.nyu.edu/ perlin/doc/oscar.html).

Várias versões de ruído de Perlin foram propostas posteriormente. Hoje em dia aplica-se estas funções de ruído para gerar proceduralmente textura 3D de vários fenômenos da natureza com um realismo impressionante, como ilustra Figura 12.15.

12.2 Mapeamento

Por um lado, a separação entre o espaço de representação geométrica e o espaço de descrição da aparência visual (textura) aumenta a reusabilidade dos modelos. Por outro lado, ela requer que seja determinada a correspondência T entre os dois espaços para associar corretamente um *pixel* a um *texel*. Para simplificar, nós vamos referir o procedimento de correspondência entre *pixels* e *texels* por **texturização**. O problema de texturização é similar ao problema que temos no dia-a-dia quando tentamos encapar uma superfície de geometria arbitrária com papel contact, com a vantagem de termos em texturização um "papel contact moldável" que se estica ou encolhe facilmente. Por simplicidade, consideremos nesta seção que o espaço de textura seja "contínuo". Na seção 12.3 discutiremos considerações adicionais para situações práticas, em que texturas são mapas discretos.

Essencialmente, há dois paradigmas para mapeamento entre dois espaços: **método direto** e **método inverso**. O método direto é orientado ao espaço de textura. Cada *texel* sofre contrações ou esticamentos, como uma lâmina de borracha, para cobrir os *pixels* aos quais ele deve ser associado. Usualmente, usa-se uma **superfície parametrizada**, isto é, uma aplicação diferenciável de $(s,t) \in \Re^2$ em $\Re^3$, ou uma transformação projetiva entre as coordenadas (U,V) do espaço da imagem DC e as coordenadas (s,t) do espaço de textura(Figura 12.16.(a)). Tal trasformação pode ser representada

por uma matriz 3×3

$$\begin{bmatrix} U' \\ V' \\ n \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} s \\ t \\ 1 \end{bmatrix}, \tag{12.7}$$

onde $(U, V) = (\frac{U'}{n}, \frac{U'}{n})$. Se estabelecermos 4 correspondências, poderemos obter os 9 elementos da matriz de transformação por técnicas clássicas, como o método de eliminação de Gauss. Este procedimento de mapeamento é também conhecido como *warping*.

O método inverso é orientado ao espaço do dispositivo. A partir da posição (U, V) de um *pixel*, determina-se as coordenadas (s, t). Embora o método direto seja mais fácil para entender, na prática o método inverso é o mais utilizado. Ele acessa cada *texel* em função do *pixel* de interesse, permitindo que ele seja prontamente integrado aos algoritmos de varredura e de *z-buffer*, a fim de constituir um eficiente fluxo de síntese de imagens foto-realistas.

Em princípio, o método inverso é uma transformação inversa da Eq. 12.7, isto é,

$$\begin{bmatrix} s' \\ t' \\ q \end{bmatrix} = \begin{bmatrix} ei - fh & ch - bi & bf - ce \\ fg - di & ai - cg & cd - af \\ dh - eg & bg - ah & ae - bd \end{bmatrix} \begin{bmatrix} U' \\ V' \\ n \end{bmatrix}, \tag{12.8}$$

No entanto, por requerer, no mínimo, 4 correspondências conhecidas e por preservar a colinearidade, os resultados nem sempre são visualmente satisfatórios. Uma segunda alternativa seria estabelecer uma correspondência por meio de uma superfécie parametrizada intermediária. Determina-se, em primeiro lugar, para cada *pixel* o ponto $P = (x_w, y_w, z_w)$ correspondente no espaço geométrico através da transformação de projeção inversa. Com uso de um **projetor**, é "mapeado" P sobre uma superfície parametrizada cujo domínio pode ser facilmente "ajustado" em uma textura. Por isso, o *texel* (s, t) correspondente a P é também conhecido como **pré-imagem** do *pixel*. O valor da pré-imagem é lido, processado e, finalmente, utilizado para modular os parâmetros de geometria ou de material da superfície visível através do *pixel* (Figura 12.16.(b)).

Nesta seção vamos detalhar cada etapa deste método inverso.

12.2.1 Projeção Inversa

Vimos na seção 5.4 que as matrizes de projeção paralela e perspectiva são, respectivamente, N_{par} (Eq. 5.17) e N_{per} (Eq. 5.20), e que a matriz de transformação do volume canônico para *viewport* é $\mathcal{V}$ (Eq.5.21). Considerando

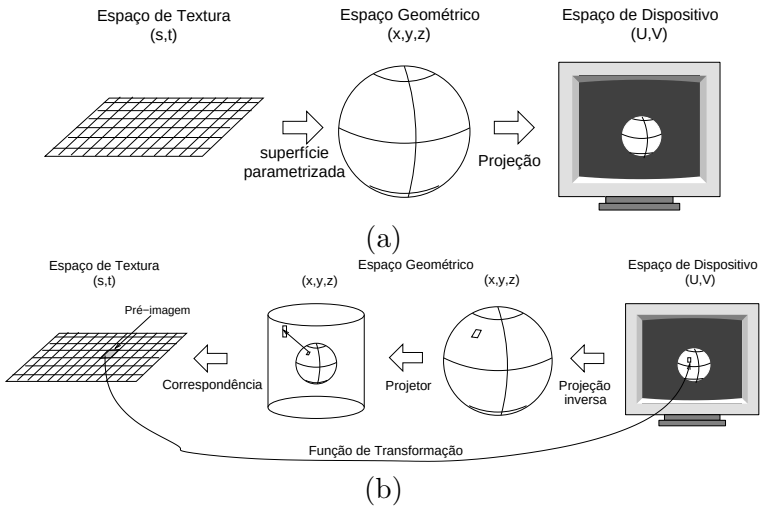


Figura 12.16: Mapeamento: (a) direto e (b) inverso.

a divisão por coordenada homogênea em projeções perspectivas e aplicando na sequência inversa estas transformações sobre as coordenadas (U, V, n) de um *pixel*, onde (U, V) são as coordenadas da tela e n é a profundidade, obteremos as coordenadas (x_w, y_w, z_w) do ponto correspondente no espaço de universo.

12.2.2 Projetor

O principal objetivo deste estágio é gerar as coordenadas de textura. Quando a figura geométrica é uma superfície parametrizada a correspondência é imediata, como veremos na seção 12.2.3. Em casos de superfícies "complexas", Bier e Sloan propuseram em 1986 "projetar" as figuras geométricas em superfícies intermediárias. Eles consideraram 4 superfícies: plano, cilindro, cubo e esfera. Quais são as alternativas de projeção de um ponto (x_w, y_w, z_w) sobre estas superfícies?

Raio Refletor

A projeção pode ser na direção do raio refletor do raio de visão$\mathbf{V}$ em relação ao vetor normal $\mathbf{N}(x_w, y_w, z_w)$ da superfície em cada amostra (x_w, y_w, z_w) (Figura 12.17.(a))

$$\mathbf{R}_i = \mathbf{V} - 2(\mathbf{V} \cdot \mathbf{N}(x_w, y_w, z_w))\mathbf{N}(x_w, y_w, z_w)$$

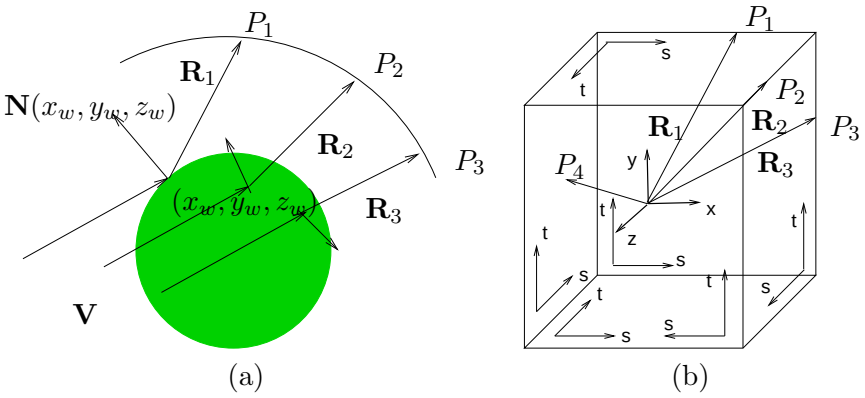


Figura 12.17: Projeção na direção do raio refletor: (a) mapa esférico e (b) mapa cúbico.

sobre o ponto $P_i = (x_i, y_i, z_i)$ da superfície intermediária. Em particular, quando se trata de texturização de uma superfície especular, podemos utilizar o vetor $-\mathbf{R}_i$ para determinar diretamente as coordenadas (x, y) da esfera (especular) intermediária pela Eq. 12.2. Se a textura for um mapa cúbico, conforme ilustra Figura 12.17.(b), o procedimento se desdobrará em dois passos, mapeando o raio refletor sobre um cubo unitário centrado na origem como superfície intermediária:

1. determinação da face i do cubo que o raio projetor $R_i = (r_x, r_y, r_z)$ intercepta; e

2. "projetar" ortogonalmente as coordenadas (r_x, r_y, r_z) sobre a face i, zerando uma das coordenadas. Por exemplo, se o ponto de interseção estiver sobre a face $x = 0.5$, como P_3 na Figura 12.17.(b), a projeção será (r_y, r_z) sobre a face $x = 0.5$.

Centróide

A projeção pode ser na direção definida pelo centróide (C_x, C_y, C_z) e o ponto $P_w = (x_w, y_w, z_w)$ (Figura 12.18.(a)). O ponto (x_i, y_i, z_i) é a interseção da superfície intermediária e o raio de projeção

$$P(t) = C + t(P_w - C).$$

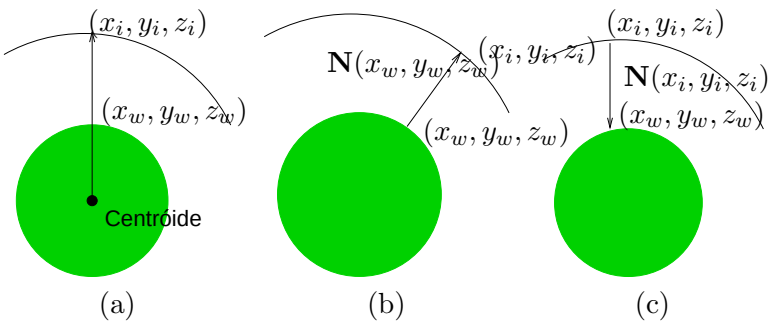


Figura 12.18: Projeção: (a) ao longo do centróide; (b) na direção do vetor normal da figura geométrica; (c) na direção do vetor normal da superfície intermediária.

Normal da Superfície Original

A projeção pode ser na direção da normal $\mathbf{N}(x_w, y_w, z_w)$ (Figura 12.18.(b)). Neste caso, o ponto (x_i, y_i, z_i) é a interseção da superfície intermediária e o raio de projeção

$$P(t) = p_w + t\mathbf{N}(x_w, y_w, z_w).$$

Normal da Superfície Intermediária

A projeção pode ser na direção da normal $\mathbf{N}(x_i, y_i, z_i)$ (Figura 12.18.(c)). Se a superfície intermediária for um plano com vetor normal $\mathbf{N}_P$, o ponto (x_i, y_i, z_i) é a interseção dessa superfície e o raio de projeção (Figura 12.19.(a))

$$P(t) = p_w - t\mathbf{N}_P.$$

Se a superfície for um cilindro, converte-se as coordenadas cartesianas (x_w, y_w, z_w) para coordenadas cilíndricas (r, θ, h). O ponto (x_i, y_i, z_i) é a interseção dessa superfície e o raio de projeção na direção $\mathbf{d} = (\cos\theta, \sin\theta, h)$ (Figura 12.19.(b))

$$P(t) = p_w - t\mathbf{d}.$$

E se for uma esfera, converte-se as coordenadas cartesianas (x_w, y_w, z_w) para coordenadas esféricas (r, θ, ϕ). O ponto (x_i, y_i, z_i) é a interseção dessa superfície e o raio de projeção na direção $\mathbf{d} = (\cos\theta\cos\phi, \sin\theta\cos\phi, \sin\phi)$ (Figura 12.19.(c))

$$P(t) = p_w - t\mathbf{d}.$$

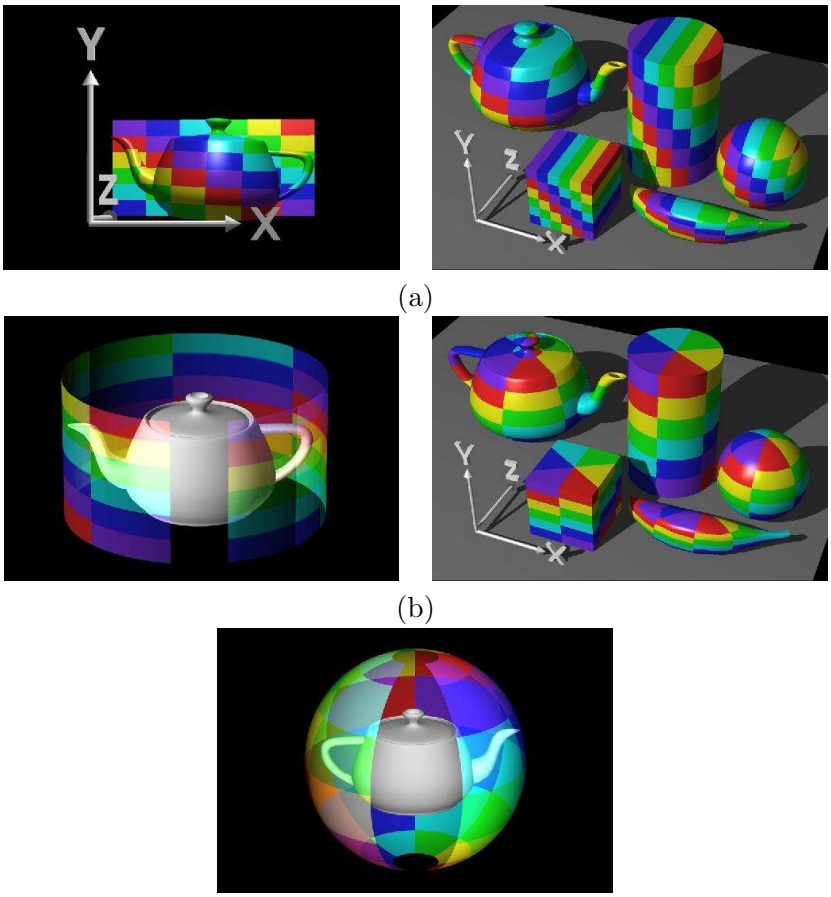


Figura 12.19: Projeção na direção da normal da superfície intermediária: (a) plano; (b) cilindro; (c) esfera (Fonte: http://www.siggraph.org/education/materials/HyperGraph/mapping/r_wolfe/-r_wolfe_mapping_2.htm.

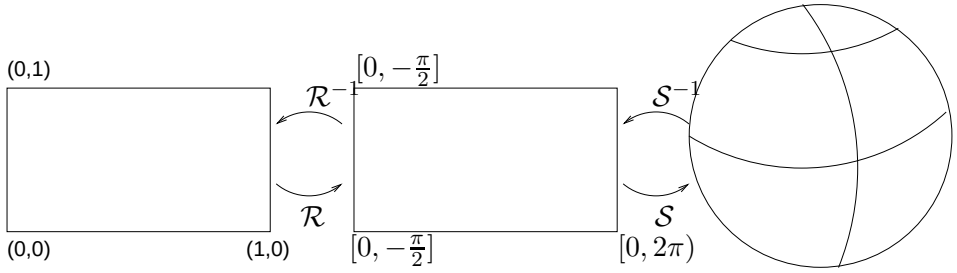


Figura 12.20: Correspondência de uma superfície parametrizada.

12.2.3 Correspondência

No estágio de correspondência, tendo uma superfície parametrizada, é imediato achar a correspondência entre as coordenadas do espaço paramétrico e as coordenadas (s,t) do espaço de textura. Vamos ilustrar o procedimento com uma esfera representada por parâmetros $\theta \in [0, 2\pi)$ e $\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$

$$\begin{aligned} x &= r\cos\theta\sin\phi \\ y &= r\sin\theta\sin\phi \\ z &= r\cos\phi. \end{aligned} \tag{12.9}$$

Podemos obter as coordenadas de textura através de uma aplicação inversa $\mathcal{R}^{-1}\mathcal{S}^{-1}$ sobre as coordenadas (x,y,z) do espaço de universo (Figura 12.20). A função $\mathcal{S}^{-1}$ seria

$$\begin{aligned} \phi &= arccos(\frac{z}{r}) \\ \theta &= arccos(\frac{z}{r\sin\phi}). \end{aligned} \tag{12.10}$$

Substituindo θ e ϕ em $\mathcal{R}^{-1}$, chegamos nas coordenadas (s,t) do espaço de textura

$$\begin{aligned} s &= \frac{\theta}{2\pi} \\ t &= \frac{\phi}{\pi} + \frac{1}{2}, \end{aligned} \tag{12.11}$$

Mesmo no caso de projeção de raios refletores, podemos aplicar a Eq. 12.4 para determinar as coordenadas (s,t) do espaço de textura de "ambiente

esférico" a partir das coordenadas (x, y) de um ponto (x, y) sobre o plano de projeção da esfera especular. Quando se trata de uma textura de "ambiente cúbico", a transformação é composta por um deslocamento e, eventualmente, uma reflexão. Por exemplo, para o ponto P_3 da Figura 12.17, a correspondência seria

$$\begin{aligned} s &= z + \frac{1}{2} \\ t &= y + \frac{1}{2} \end{aligned} \tag{12.12}$$

e para o ponto P_4 da mesma figura,

$$\begin{aligned} s &= -x + \frac{1}{2} \\ t &= y + \frac{1}{2} \end{aligned} \tag{12.13}$$

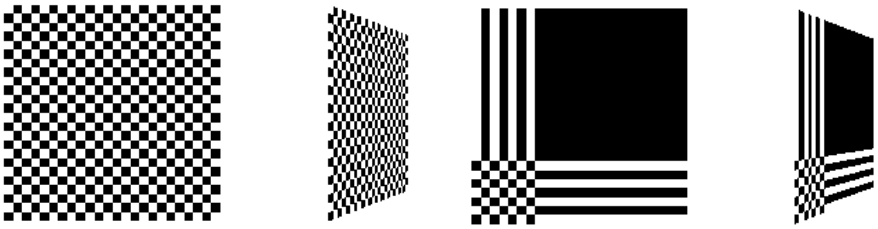
Tipicamente, o espaço de textura bi-dimensional é definido no domínio $[0, 1.0] \times [0, 1.0]$. No entanto, se precisarmos de mais "retalhos de textura" para cobrir uma área extensa, utilizando coordenadas de textura fora deste domínio, como ficará a aparência da figura geométrica? Ela dependerá de um dos dois seguintes "modos de embrulho", em inglês *wrapping mode*:

repetição: a textura será repetida até cobrir toda a extensão da área. Figura 12.21.(a) mostra o efeito da repetição da textura do canto esquerdo inferior da Figura 12.21.(b) nas duas direções.

supressão: a textura será suprimida fora do domínio "normalizado". Figura 12.21.(b) mostra o efeito de "omissão" da textura fora do intervalo $[0, 1.0]$ na direção x e na direção y.

A técnica de superfície parametrizada não consegue gerar efeitos satisfatórios em superfícies fechadas, como um cubo, uma esfera, ou um cone. Sem esticar ou comprimir alguma parte da textura, não é possível ajustá-la sobre a superfície. Isso pode provocar distorções indesejáveis, como ilustra Figura 12.22. Nesta figura mostramos o resultado da imagem apresentada na Figura 12.5.(a) aplicada sobre uma esfera. Compare o resultado com o da Figura 12.5.(c). Observe as distorções nos pólos da esfera!

Uma forma de solucionar as distorções e problemas de "costura" nas junções dos "retalhos de textura" é o uso da textura tri-dimensional. Em



(a) (b)

Figura 12.21: Modo de correspondência: (a) repetição e (b) supressão.

Figura 12.22: Distorção de uma textura retangular sobre uma esfera.

analogia à cinzelação, a textura representaria o material bruto do qual a forma geométrica de interesse seria "esculpida". Textura tri-dimensional mais conhecida é a do ruído de Perlin apresentado na seção 12.1.6. Utilizando o procedimento proposto pelo Perlin, computa-se diretamente a partir das coordenadas (x, y, z) as coordenadas de textura (s, t, r).

12.2.4 Função de Transformação

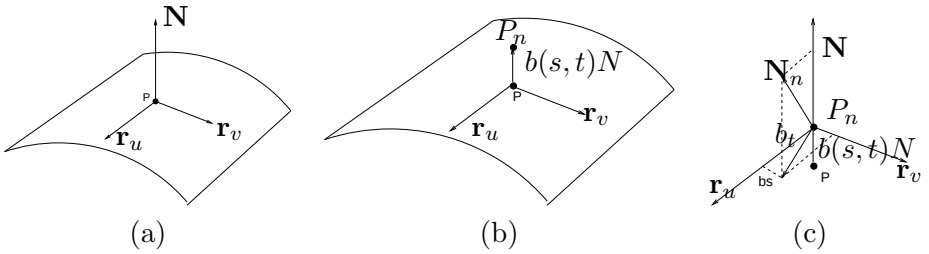
Tipicamente, os valores armazenados em cada entrada (s, t) da textura são as três componentes de cor (R, G, B) que podem alterar os valores do *pixel* correspondente em um dos seguintes modos de combinação

substituição , em inglês *replace*, a cor do *pixel* é substituída pela armazenada no *texel*;

decalque , em inglês *decal*, a cor do *pixel* é combinada com a cor do *texel*;

modulação , em inglês *modulate*, a cor do *pixel* é modulado pela cor de *texel*.

Há ainda texturas que armazenam valores que modificam outros parâmetros do modelo de iluminação, como textura de bossagem. Neste caso, é necessário ainda aplicá-los por funções modificadoras em algum atributo da



(a) (b) (c)

Figura 12.23: Textura de bossagem: (a) visualização do mapa; (b) vetores originais; e (c) vetores perturbados.

superfície antes de determinar a nova cor do *pixel*. Vamos ilustrar a idéia com a textura de bossagem da seção 12.1.2.

Supomos que o *texel* (s,t) do mapa de textura seja correspondente ao ponto da superfície $P = \mathbf{r}(x(u,v), y(u,v), z(u,v))$ com normal igual a $\mathbf{N}$ (Figura 12.23.(a)). A sua posição é perturbada com uso da textura ao longo do vetor normal, obtendo uma nova posição P_n (Figura 12.23.(b))

$$P_n = P + \frac{b(s,t)\mathbf{N}}{|\mathbf{N}|}.$$

Blinn mostrou que uma boa aproximação para o vetor normal $\mathbf{N}_n$ do ponto deslocado seria (Figura 12.23.(c))

$$\mathbf{N}_n = \mathbf{N} + \frac{b_s(\mathbf{N} \times \mathbf{r}_u) - b_t(\mathbf{N} \times \mathbf{r}_v)}{|\mathbf{N}|},$$

onde $\mathbf{r}_u$ e $\mathbf{r}_v$ são as derivadas parciais da superfícies $\mathbf{r}$ e b_s e b_t são derivadas parciais da função de ruído $b(s,t)$ que podem ser obtidas ou com uma máscara de convolução (p. ex., operador de Prewitt) ou diretamente com as diferenças finitas

$$b_s = \frac{b(s+\Delta s, t) - b(s,t)}{\Delta s} \qquad b_t = \frac{b(s, t+\Delta t) - b(s,t)}{\Delta t}.$$

Ao invés de uma imagem, os ruídos $b(s,t)$ podem ser definidos proceduralmente. Um exemplo simples seria uma função senoidal.

12.3 Mapeamento de Espaços Discretos

Até agora, consideramos que o espaço de textura seja contínuo, ou seja, para cada *pixel* existe sempre um *texel* correspondente de forma exata. Na

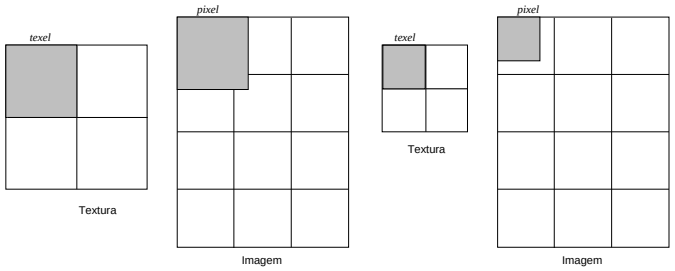


Figura 12.24: Correspondência *texel* e *pixel*: (a) magnificação; (b) minimização.

prática, o espaço de textura é discreto. Nem sempre a correspondência é 1:1. Quando as dimensões de um *texel* são maiores do que um *pixel*, ele cobre mais de um *pixel* (magnificação). E se forem menores, ele cobrirá uma fração dos *pixels* (minimização). A pergunta é como deve ficar a aparência da superfície nestes casos?

12.3.1 Magnificação

Quando se trata de magnificação, uma solução mais simples é escolher a amostra de coordenadas mais próximas possíveis do valor (s,t) obtido, resultando em uma aparência de "quadriculada", conhecida como *pixelation* em inglês. Outra solução, com efeitos visuais mais agradáveis, seria aplicar uma interpolação bilinear entre os atributos dos quatro *texels* vizinhos do texel (s,t) para obter o seu atributo $A(s,t)$. Os fatores f_s e f_t utilizados para interpolação bilinear são, respectivamente, $s - \lfloor s \rfloor$ e $t - \lfloor t \rfloor$

$$\begin{aligned} A(s,t) &= f_t(f_s A(s_r,t_b) + (1-f_s)A(s_l,t_b)) + \\ &\quad +(1-f_t)(f_s A(s_r,t_t) + (1-f_s)A(s_l,t_t)) \\ &= (1-f_s)(1-f_t)A(s_l,t_b) + f_s(1-f_t)A(s_r,t_b) + \\ &\quad (1-f_s)f_t A(s_l,t_t) + f_s f_t A(s_r,t_t) \end{aligned} \tag{12.14}$$

12.3.2 Minimização

Se a textura é minimizada, vários *texels* podem cobrir um único *pixel*; portanto, a aparência do *pixel* deve ser uma cor integrada das cores destes

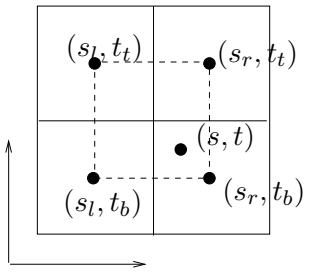


Figura 12.25: Interpolação bilinear.

texels. A pergunta é como obter este "efeito integrador" de forma eficiente? Escolher a amostra mais próxima das coordenadas (s, t) resulta em uma aparência serrilhada. Para atenuar este artefato, pode-se aplicar uma técnica de filtragem, removendo sinais de alta frequência. No entanto, este processo é computacionalmente caro, já que, em casos extremos, podem ocorrer milhares de *texels* em um único *pixel*. Diversas técnicas foram propostas com o objetivo de baratear o processo.

Hoje em dia, a técnica mais popular é a técnica conhecida por *MIP* (*multum in parvo*) que tem como premissa de que a pré-imagem de cada *pixel* tenha uma forma quadrada de tamanho igual a uma potência de 2. Ao invés de uma única imagem original de nível $i = 0$, uma sequência de imagens filtradas de níveis de resolução monotonicamente decrescente é pré-calculada, de tal sorte que a imagem de nível de resolução $i+1$ é o resultado da filtragem e subamostragem da imagem de nível de resolução i. Daí o nome sugestivo de **pirâmide de texturas** (Figura 12.26.(a)). Como o tamanho de cada imagem é uma potência de 2, um espaço de memória adicional correspondente a $\frac{1}{3}$ do espaço ocupado pela imagem original é suficiente para armazenar o restante dos níveis de detalhe (Figura 12.26.(b)).

Durante rasterização, deve-se selecionar dentre o conjunto de imagens pré-computadas a que tiver uma resolução mais próxima a do *pixel*. William propôs em 1983 estimar a imagem i a ser utilizada como pré-imagem de um *pixel* através da verificação da variação D das dimensões originais ∂s e ∂t da pré-imagem de um *pixel* $\partial U = \partial V = 1$

$$D = max[\sqrt{(\frac{\partial s}{\partial U})^2 + (\frac{\partial t}{\partial U})^2}, \sqrt{(\frac{\partial s}{\partial V})^2 + (\frac{\partial t}{\partial V})^2}]$$

Se $D = 2^i$, o nível de detalhe i é selecionado para mapeamento; senão, dois níveis de detalhe adjacentes são selecionados e interpolados linearmente para obter um *texel* com dimensões próximas das do *pixel*.

(a) (b)

Figura 12.26: Mipmap: (a) pirâmide e (b) organização de dados (Fonte: http://www.relisoft.com/science/graphics/mip.html).