Manipulacao Direta em Geometria 3D

Vanio Fragoso de Melo e Mercedes R. G. Marquez
DCA/FEEC/UNICAMP

8 de fevereiro de 2001

Resumo

Esta monograa visa a explanacao sobre manipulacoes diretas em geometria 3D, dividida em tres partes: interacoes 3D, pontos de controle e arquitetura.

Sum `rio

1 Capitulo: Interacoes 3D

1.1 Introducao

Uma interação intuitiva e eficiente entre o usuário e o ambiente 3D, constitui uma das maiores preocupações no desenvolvimento de interfaces gráficas para aplicações 3D. A comunicação entre o usuário e o ambiente tridimensional através dos dispositivos de entrada e saída 2D é uma das principais dificuldades para se obter uma interação eficiente; isto devido à tridimensionalidade do espaço e dos objetos da cena (um grau de liberdade a mais, em relação aos dispositivos).

Em frente disso, duas abordagem têm sido dadas para o problema de interação usuário-ambiente 3D, e são:

- Abordagem por hardware
- Abordagem por software

Nas seguintes seções deste capítulo, descreveremos sucintamente três propostas. A primeira (O *mouse* de Rockin - Seção 1.2) pertencente a abordagem por hardware, e as duas próximas (Técnicas de manipulação direta para objetos 3D usando dispositivos *locator* 2D) - Seção 1.3 e *Snap-Dragging* em três dimensões - Seção 1.4) sendo técnicas de interação que permitem o mapeamento entre dispositivos de entrada bidimensionais e o espaço 3D (abordagem por software).

1.2 O mouse de Rockin: Manipulacao 3D integral no plano

Uma alternativa de solução dada ao problema de interação usuário-ambiente 3D tem sido o desenvolvimento de novos e até sofisticados dispositivos 3D. A justificativa para a criação de tais dispositivos tem sido o desejo de obter uma correspondência natural entre as ações dos usuários sobre os dispositivos e o movimento obtido no espaço 3D.

Ainda as aplicações 3D usualmente requerem uma quantidade substancial de interação 2D como manipular objetos 3D em vistas 2D, tarefas usuais de seleção de items de menu, digitação de um texto, etc. Em frente disso se faz necessário um dispositivo que permita a interação em 2 e 3 dimensões de forma eficiente.

Visto que o *mouse* 2D é de maneira indiscutível um bom dispositivo para interações 2D, neste artigo, propõe-se um *mouse* 3D, que incorpore as propriedades do *mouse* 2D usual e ao mesmo tempo forneça controle intuitivo sobre a terceira dimensão do espaço 3D.

O *mouse* de Rockin proposto (Figura 1).

Figura 1: O *mouse* de Rockin

como o *mouse* usual é capaz de sensar sua posição sobre a superfície de operação. Além disso ele tem a base arrendondada para permitir a sua inclinação em torno aos eixos x e z (conseguindo controlar dois graus extra de liberdade - Figura 2).

Figura 2: Inclinação do mouse

A implementação considera uma região plana no centro da base curveada para possibilitar manter a sua estabilidade e para poder eventualmente ser restringido ao uso de apenas dois graus de liberdade. Opera-se também sobre uma digitalizadora. Esta mesa é capaz de sensar a posição do sensor sobre o plano $x-z$ e também o grau de inclinação do sensor em relação aos eixos x e z. Um dos sensores é montado no centro do *mouse* de Rockin, possibilitando que as posições planar e angular sejam sensadas quando estejamos posicionados na mesa. A resolução corrente do sensor de inclinação é aproximadamente uma unidade por grau. A mesa pode também sensar o estado dos botões do *mouse* de Rockin, os mesmos que estão conectados ao sensor.

Experiência realizada

1. Objetivo:
   O objetivo da seguinte experiência foi avaliar a eficiência do *mouse* de Rockin comparado com o *mouse* regular, no contexto de tarefas de posicionamento 3D.

2. Recursos utilizados:
   Estação de trabalho Silicon Graphics Indigo2 Extreme e mesa digitalizadora da Wacom.

3. Tarefas e estimulo:
   A tarefa consistiu em deslocar um objeto 3D desde um extremo (quina) de

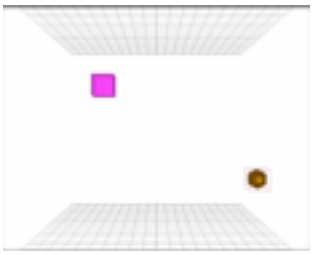


Figura 3: O *mouse* de Rockin

uma cena 3D e posicioná-lo dentro de outro objeto localizado na quina diagonalmente oposta da cena (Figura 3).

Clicando na face frontal do objeto (com botão direito) a caixa era selecionada ativando os eixos x-y.
Clicando no lado direito/esquerdo eram selecionados os eixos y-z.
Clicando na parte superior/inferior eram selecionados os eixos x-z.
Mantendo o botão da esquerda pressionado o *mouse* efetuava o movimento em 2D do objeto no plano ativo.
Movendo o *mouse* de Rockin para direita/esquerda ou frente/trás na mesa o objeto se movia na direção x e direção z respectivamente.
Inclinando o *mouse* no sentido horário/anti-horário movia o objeto para cima/ para baixo na direção de y.

4. Pessoas participantes:
   14 voluntários, todos destros. Três deles com experiência em cenas 3D com o *mouse* padrão e 11 familiarizados com o *mouse* 2D mas com experiência limitada com ambientes 3D.

5. Projeto e procedimentos:

   - 14 pessoas participantes
   - 2 dispositivos por pessoa
   - 6 blocos por dispositivo
   - 8 condiçoes por bloco
   - 4 provas por condição

   Foi considerado um treinamento inicial de 15 minutos aproximadamente. Para cada condição foram permitidos pequenos intervalos, porém foi requerido completar as quatro provas por condição, sem intervalos. A medição do tempo começava quando o objeto aparecia na tela e finalizava quando o objeto era satisfatóriamente posicionado no lugar destino. Entre cada prova houve pausas de 800 ms. As pessoas participantes eram designadas alternadamente, seja na ordem: *Mouse* de Rockin primeiro ou *mouse* regular primeiro. No final do experimento foi solicitado o preenchimento de um questionário para recolher a opinião sobre os dois dispositivos e sobre a técnica de interação associada.

6. Resultados piloto:
   A análise dos dados a partir dos testes piloto mostrou que a tarefa foi dividida em duas fases:

- Fase inicial (loop aberto).- Levar o objeto até uma vizinhança do objeto alvo.
- Fase seguinte.- Movimentos de loop fechado que posiciona com precisão o objeto dentro do objeto alvo.

A complexidade nos movimentos 3D é maior que nos movimentos 2D, porém a hipotese dos autores é que as pessoas eventualmente serão capaces de realizar as ações 3D automaticamente, não entanto precisa-se muita repetição e aprendizado.
A experiência foi insuficiente para permitir que os usuários adquiram uma experiência suficientemente alto de uso.
Após o terceiro teste o tempo fica aproximadamente constante devido ao maior treinamento que reforça o aprendizado.

Resultados

Três importantes resultados estadísticos do desempenho do *mouse* de Rockin em comparação ao *mouse* 2D, são mostrados nas Figuras 4, 5 e 6.

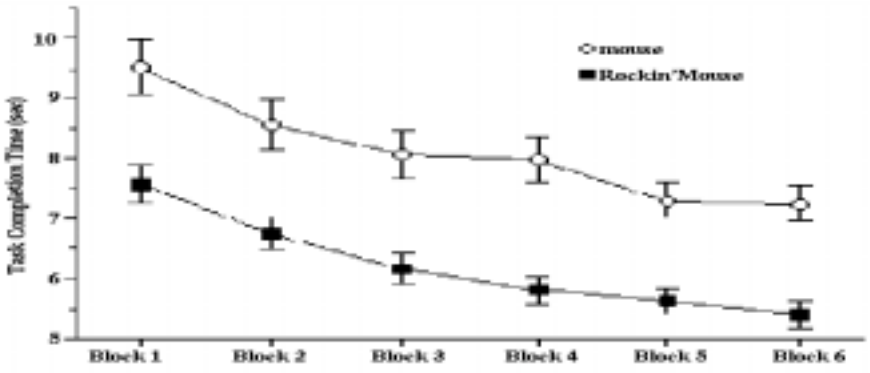


Figura 4: A média do tempo

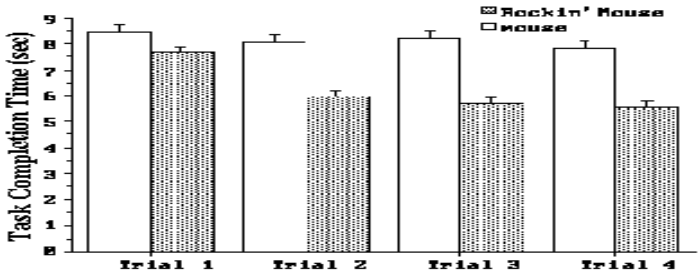


Figura 5: Média do tempo de

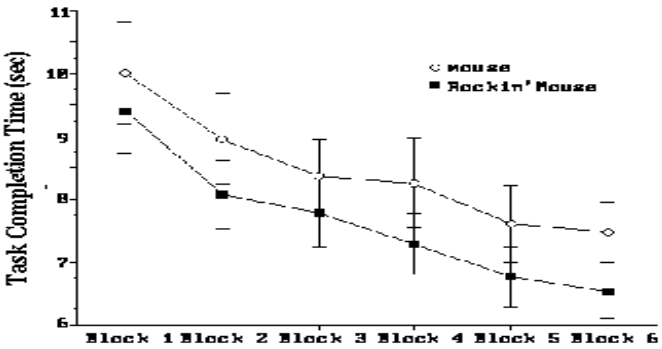


Figura 6: Média do tempo de

1.3 Snap-Dragging em tres dimensoes

Neste artigo propõe-se uma técnica para posicionar pontos e objetos em cenas tridimensionais. Esta ferramenta pela sua vez combina três técnicas interativas que trabalham juntas: funçoes de gravidade, alinhamento de objetos e tranformações interativas de movimentação suave. Desta maneira a composição da cena é realizada numa vista perspectiva única usando um *mouse* e um teclado.

A função de gravidade ativa um cursor 3D chamado *skitter* para ajudar o usuário a colocar pontos de objetos em vértices, arestas, superfícies e as suas interseções. Realiza-se uma atração de posição do cursor quando ele é posicionado numa vizinhança dos objetos. O conjunto de objetos alinhados (linhas, planos, esferas) podem ser construídos através de vértices de objetos ou pontos de controle, fornecendo o estilo regua e compás na construção em 3D. O *skitter* pode ser colocado nesses objetos alinhados e nos seus pontos e curvas de interseção, assim como objetos da cena. E transformações interativas (traslação, rotação e escalamento) rastream o movimento do *skitter* que continua a se fixar nos objetos durante as transformações permitindo transformações precisas a serem aplicadas nos objetos da cena.

As três componentes, gravidade, alinhamento de objetos e transformações de movimentação suave têm sido usadas em editores de manipulação direta, porém *snap-dragging* apresenta melhoras sobre essas técnicas. Por exemplo muitas das funções de gravidade existentes, apenas atraem o cursor para um tipo de objeto de cada vez, exemplo vértices e arestas, mas não ambos, fazendo com que o usuário realice frequentemente troca de comandos para obter a função desejada. A técnica *snap-dragging* contorna esse problema ao utilizar 3 funções de gravidade ao mesmo tempo, numa ordem de preferência estabelecida pelo usuário.

Função de gravidade

Na técnica de *snap-dragging*, esta função é utilizada em todas as operações de interação com a cena. Dentro deste conjunto de atividades encontram-se os comandos utilizados para posicionar o *skitter* em um local determinado, acrescentar novos segmentos de linha e aplicar transformações afins suaves. O *skitter* é reposicionado fixando-se no objeto mais próximo que contém a função de gravidade, à medida que o mouse é movimentado. Se o *skitter* não está no domínio da gravidade do vértice, o programa vai tentar atraí-lo para uma aresta. Se não for possível, tentará atraí-lo

para uma face. Finalmente, se não for possível, o *skitter* vai ser atraído para um plano estabelecido por *default* e que é paralelo à tela.

O eixo z do *skitter*, quando é atraído, posiciona-se perpendicularmente à aresta ou à face para a qual o *skitter* foi atraído. O eixo x, será posicionado tangencialmente a essa aresta. Se o *skitter* está num vértice, seus eixos serão determinados pelas faces e arestas que terminam nesse vértice. Essa orientação será posteriormente utilizada na colocação de uma âncora para definir um eixo de rotação. Tal ancora terá a mesma orientação do *skitter*.

O ponto 3D onde é colocado o *skitter* é calculado pelo algoritmo responsável pela função de gravidade, e é tal que a posição (x, y) seja a mais próxima do cursor do mouse. Além disso, deve-se determinar uma lista de objetos que projetam na posição do cursor, de modo que um comando de menu possa ser invocado para permitir a escolha de objetos ocultos ou difíceis de serem selecionados por encontrar-se muito próximos.

Alinhamento de objetos

A implementação corrente considera três tipos diferentes de objetos de alinhamento: linhas, planos e esferas. O usuário ativa um conjunto de objetos de alinhamento escolhendo um menu de valores de alinhamento. A adição de objetos de alinhamento é realizada da seguinte maneira: primeiro o usuário seleciona os objetos da cena que criarão os objetos de alinhamento e aciona um comando do teclado para transformar todos os vértices dos objetos selecionados em *hot points*. Hot points são desenhados como quadrados brancos em cima desses vértices. Em seguida o usuário ativa um valor de um dos menus de alinhamento, fazendo com que em cada *hot point* o sistema construa um objeto de alinhamento de todos os valores ativados.

Há um objeto especial que serve igualmente para definir um eixo de rotação. Ele é chamado de âncora e o seu ponto central é um *hot point*. Define três planos de alinhamento que se cruzam no seu centro, e três retas de alinhamento. Esta âncora é colocada com a ajuda do *skitter*.

O algoritmo de eliminação de linhas ocultas não afeta os objetos de alinamento, o qual facilita a seleção de curvas de alinhamento em ocasiões onde normalmente estariam obscurecidas. Para evitar confusão na tela a causa da sobreposição de muitos objetos de alinhamento, cada um deles é desenhado como uma única curva fina de cor cinza, e as linhas de interseção são mostradas como curvas finas de cor preta. Quando o *skitter* é posicionado num ponto de interseção, todos os objetos de alinhamento e os *hot points* que contribuem para essa interseção, são desenhados com linhas grossas.

Transformações

Todas as tranformações *snap-dragging* seguem suavemente o *skitter*. Os objetos podem ser posicionados com precisão devido a que o *skitter* continua a se fixar aos objetos da cena e aos objetos de alinhamento durante as transformações. As transformações possíveis são translação, rotação sobre um ponto e rotação sobre um eixo.

Durante uma operação de translação, os objetos selecionados são transladados por um vetor partindo da posição inicial do *skitter* para a posição final. Colocaque-se o *skitter* na posição do cursor através de um comando de menu *place skitter*.

A âncora é usada como um centro de rotação, um eixo de rotação, um centro de escala ou um ponto de gravidade. Ela é posicionada sobre um *skitter*, seguindo a orientação dele. Durante a rotação em torno de um ponto, os objetos selecionados

são rotacionados em torno do ponto definido pela âncora. O ângulo de rotação é definido pela linha entre a âncora e a posição original do *skitter*. O eixo de rotação é a linha que passa pela âncora e é perpendicular ao plano determinado pelos três pontos que definem o ângulo de rotação. Dessa maneira, é a posição final do *skitter* a que determina o eixo de rotação. Durante a rotação em torno de um eixo, os objetos selecionados são rotacionados sempre em torno do eixo x da âncora, por um ângulo determinado pela posição do *skitter* em torno desse eixo.

Concluindo, a técnica de *snap-dragging* consegue simplicar substancialmente o trabalho de manipulação de objetos 3D ao incorporar funções de gravidade, alinhamento e movimentação suave em um só conjunto. Dessa forma o número de operações é reduzido, elimina-se a necessidade de utlizar várias janelas com perspectivas diferentes para edição de objetos, e a construção de objetos consegue transmitir melhor a analogia da utilização de réguas e compassos em 3D através do cálculo automático de interseções entre objetos de alinhamento.

1.4 Tecnicas de Interacoes 3D Usando Locator

A interação com objetos 3D não é trivial. Existem dispositivos para a manipulação e visualização em ambientes 3D, mas, temos alguns problemas para contornar, tais como: alto custo, fadiga visual e muscular e imprecisão que os mesmos ainda apresentam. Por outro lado, os dispositivos 2D são populares e de baixo custo, o que ratifica o grande uso dos mesmos na interação em ambientes 3D. Porém, é necessário abordagens especiais para a manipulação direta em 3D através de dispositivos 2D. É difícil a percepção da relação espacial entre os objetos e manipulá-los em ambientes 3D de forma intuitiva, confortável, eficiente e precisa, devido ao grau de liberdade a menos que os dispositivos 2D tem em relação ao ambiente 3D. Por causa dessa natureza dos dispositivos 2D, display e meios de interações, as técnicas de entrada 3D ainda continua sendo um problema em computação gráfica.

Várias técnicas de interação, segundo Nielson e Olsen, em [3], requerem locator 2D e tem as suas deficiências, como o fato do usuário manipular objetos de controle que são mapeados para a manipulação real desejada e o fato do retorno fornecido, a não ser o próprio objeto, ser de natureza 2D até certo ponto. Nesse mesmo artigo, Nielson e Olsen apresentaram técnicas de manipulação direta para tarefas interativas de especificação em 3D de um ponto, de uma translação, de uma rotação em torno de um eixo e de uma mudança de escala.

Especificando um ponto

Para especificar um ponto no espaço 3D usa-se um *mouse tríade (*graficamente é um objeto composto de três eixos ou é um cubo) cuja posição 3D é manipulada por um dispositivo locator 2D. Junto com o mouse tríade, cria-se uma armação cúbica na qual circundará o espaço no qual contém os objetos que se trabalhará, que servirá como estrutura de referência para a manipulação real do cursor tríade.

Mapeando movimento 2D para 3D

No mapeamento do movimento 2D para movimento 3D das posições do locator 2D, de $(x\prime, y\prime)$ para (x, y), usa-se as projeções dos eixos tríade (X_x, X_y), (Y_x, Y_y) e (Z_x, Z_y), e o vetor mudança de direção $(D_x, D_y) = (x - x', y - y')$.

O movimento do mouse 3D é ao longo de um de seus eixos tríades, que está relacionado o quão mais próximo a sua projeção está do vetor mudança de direção

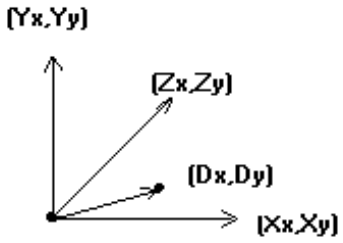


Figura 7: Proj. dos eixos tríades e vetor desloc.

(D_x, D_y). Basta pegar o eixo cujo o valor absoluto do coseno ângulo entre sua projeção e o vetor (D_x, D_y) seja o maior. Isto divide o espaço 2D em seis setores que são demarcados pelas retas bissetrizes dos ângulos entre as projeções dos vetores tríades(figura 8).

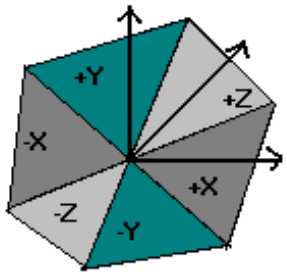


Figura 8: Zonas de movimento do cursor

Nas transformações de translação, rotação e mudança de escala é explorado a geometria do objeto como realmente sendo manipulada, tal como manter uma ou mais dimensões constantes enquanto manipula-se as outras. Os objetos são poliédricos. As transformações são baseadas nas arestas e faces dos mesmos.

Translação

A translação usando uma aresta do objeto é feita através da escolha de um ponto P_1, que seleciona a aresta do objeto, cuja projeção 2D está mais próxima do mesmo. Escolhe-se um segundo ponto P_2 que será projetado junto com P_1, sobre a aresta determinada acima, nos pontos $P_1^{'}$ e $P_2^{'}$, respectivamente(vê figura 9(a)). A translação é então $P_2^{'} - P_1^{'}$. A translação usando uma face, um ponto P_1 2D é usado para selecionar uma face e um outro ponto P_2 2D, junto com o primeiro, são projetados no plano da face selecionada, nos pontos $P_1^{'}$ e $P_2^{'}$ 3D. A translação será $P_2^{'} - P_1^{'}T$(vê figura 9(b)).

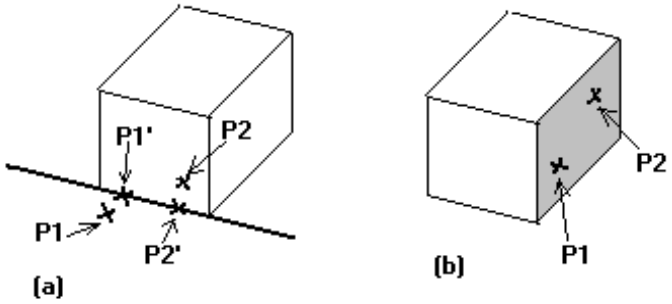


Figura 9: (a)Translação ao longo de uma aresta (b)Translação sobre um plano

Rotação

A rotação pode ser com relação a uma aresta ou com relação ao vetor normal a uma face. No primeiro caso, um ponto P_1 seleciona a aresta que será o eixo de rotação. Um segundo ponto P_2 é projetado sobre uma face F num ponto $P_2^{'}$ 3D. Um terceiro ponto P_3 é projetado num ponto $P_3^{'}$ 3D, sobre o plano α que passa por $P_2^{'}$ e é normal a aresta selecionada. Sendo $P \in \alpha \cap F$, o ângulo de rotação será o ângulo entre os vetores $u = P_3^{'} - P$ e $v = P_2^{'} - P$, no sentido de $P_2^{'}$ para $P_3^{'}$(vê figura 10(a)). No segundo caso, P_1 seleciona a face F, sendo projetado sobre a mesma no ponto $P_1^{'}$. O eixo de rotação R, passa por $P_1^{'}$ e é paralelo ao vetor normal a face F selecionada. O ângulo de rotação é determinado pelas projeções $P_2^{'}$ e $P_3^{'}$ 3D, de dois pontos P_2 e P_3, como sendo o ângulo entre os vetores $u = P_3^{'} - P_1^{'}$ e $v = P_2^{'} - P_1^{'}$, no sentido de $P_2^{'}$ para $P_3^{'}$, onde $P_1^{'} \in R \cap F$ 10(b)).

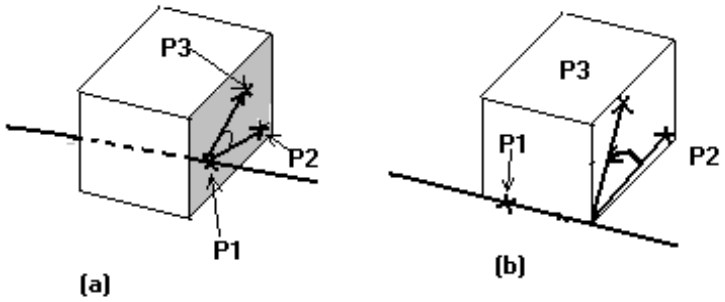


Figura 10: (a)Aresta de rotação (b)Face de rotação

Mudança de escala

A mudança de escala é proposta de três formas. A primeira, três pontos P_1, P_2 e P_3 são projetados sobre uma aresta (a mais próxima) nos pontos $P_1^{'}$, $P_2^{'}$ e $P_3^{'}$. O fator de escala será $\frac{\left\|P_3^{'} - P_1^{'}\right\|}{\left\|P_2^{'} - P_1^{'}\right\|}$ ao longo do eixo definido pela aresta(vê figura 11(a)). A segunda, o ponto P_1 determinará o vértice do objeto e duas arestas adjacentes

(o resultado será um tanto satisfatório se as arestas forem perpendiculares), que definirão o plano de projeção dos pontos P_2 e P_3, nos pontos $P_2^{'}$ e $P_3^{'}$. Transforme o vértice na origem, as arestas nos eixos x e y, e o vetor normal no eixo z. Sendo $P_2^{''}$ e $P_3^{''}$ os pontos $P_2^{'}$ e $P_3^{'}$ transformados, aplique a mudança de escala $\frac{P_{3x}^{''}}{P_{2x}^{''}}$ e $\frac{P_{3y}^{''}}{P_{2y}^{''}}$ para os eixos x e y, respectivamente. Em seguida, transforme de volta(vê figura 11(b)). A terceira, o ponto P_1 seleciona uma face e junto com um segundo ponto P_2 são projetados na mesma. As respectivas projeções, $P_1^{'}$ e $P_2^{'}$, determinam uma linha de projeção do terceiro ponto P_3cuja projeção será $P_3^{'}$. Transforme $P_1^{'}$ na origem, o vetor normal no vetor $e_3 = (0, 0, 1)$ e o vetor $u = P_2^{'} - P_1^{'}$ no vetor $e_1 = (1, 0, 0)$. Aplique a mudança de escala $\frac{P_{3x}^{'}}{P_{2x}^{'}}$ para os eixos x e y. Em seguida, transforme de volta(vê figura 11(c)).

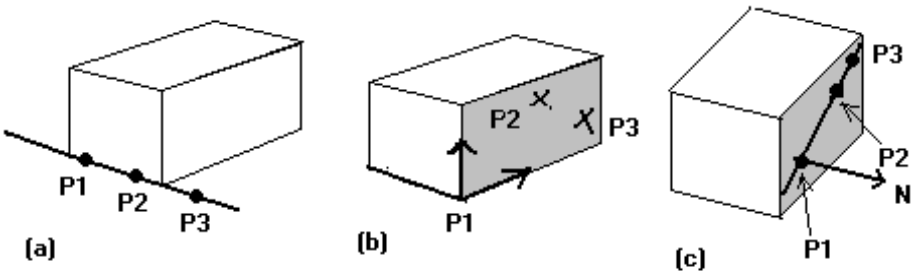


Figura 11: (a)Aresta de escalonamento (b) Face de escalonamento (c)esc. uniforme

Conclusão

Nota-se, com esta abordagem de Nielson e Olsen, as dificuldades do problema de realizar tarefas 3D com dispositivos 2D, display e meios de interações, como posicionar um ponto e mapear o seu movimento no espaço, tendo 3 graus de liberdade no espaço contra 2 graus de liberdade dos dispositivos. Têm-se que fazer aproximações, perdendo um pouco e as vezes muito com isso, dependendo da situação. Realizar tarefas 3D de rotação, translação e mudança de escala, as vezes está limitada ao tipo de objeto e, quando não, pode não ser tão intuitivas, podendo ter as duas, limitação e de difícil percepção.

2 Capitulo: Pontos de Controle

2.1 Introducao

Na modelagem e animação de objetos (curvas, superfícies e sólidos) em interfaces interativas 3D, um dos grandes desafios é o controle dos parâmetros de forma intuitiva e eficiente, que possibilitem determinar e mudar a estrutura geométrica, o posicionamento e o movimento de um objeto no espaço. Parâmetros que possibilitem controlar a forma do objeto(esticar, curvar e torcer), rotacionar, translandar, cuidar dos problemas de contatos, colisões e interpenetrações entre os objetos (*posicionamento do objeto na cena*) de forma intuitiva, com realismo visual e com uma boa eficiência de tempo. Não podemos esquecer dos parâmetros que controlam a câmera, que possibilitam determinar como a cena será visualizada. Este problema de controle de parâmetros é de grande interesse da comunidade de computação gráfica, que ainda não foi solucionado pela grande dificuldade do mesmo. Várias técnicas foram propostas para solucionarem parcialmente tal problema. Técnicas para manipulação da forma geométrica do objeto, técnicas para controlar o posicionamento dos objetos na cena (*evitar as interpenetrações, durante colisões e contatos entre os mesmos*) e técnicas para o controle dos parâmetros da câmera. É difícil conseguir, numa só técnica, realismo visual, eficiência de tempo e controle intuitivo. Ganha-se num e perde parcialmente noutro. Na secção 2.2, discutiremos alguns modelos de deformações geométricas denominados FFD (*Free-Form Deformation),* que são técnicas de manipulação direta em estruturas de forma livre, relacionadas ao objeto, para obtenção de deformações (*mudança na estrutura geométrica)* nos mesmos. Na secção 2.3, discutiremos uma técnica proposta para tratar dos problemas de contatos, colisões e interpenetrações entre objetos. Na secção 2.4, será discutida uma técnica de determinação dos parâmetros de controle da câmera apartir da cena.

2.2 Controle de Forma Livre

As técnicas de deformações de objetos, até então apresentadas pela comunidade que se interessa em solucionar ou propor soluções parciais para o mesmo, podem ser classificadas como: geométricas (que levam em consideração apenas conceitos e propriedades geométricas), físicas (que levam em consideração conceitos e propriedades físicas) e híbridas (usa o que há de melhor das duas anteriores). As técnicas geométricas são tidas como eficientes em tempo de processamento do que as técnicas físicas, pois, esta última, em geral, têm-se um sistema de EDP's de 2^aordem para resolver. Mas, as técnicas físicas tem mais realismo visual do que as geométricas. As técnicas híbridas, como dito acima, tenta aproveitar o que há de melhor das duas, usa uma técnica geométrica para obtenção da forma grosseira do objeto e uma técnica física para o refinamento do mesmo. As técnicas apresentadas a seguir são puramente geométricas.

Deformação usando Lattice

Apresentaremos, a seguir, uma técnica denominada Free-Form Deformation (deformação de forma livre), FFD, para análise das dificuldades inerentes ao problema.

Em 1986, Sederberg propôs um modelo de deformação denominado FFD, no artigo [9], onde um objeto é imerso numa LATTICE (treliça) de pontos de controle. Em seguida, define-se uma função que relaciona os pontos de controle e o objeto. Para a obtenção da deformação do objeto, manipula-se diretamente nos pontos de

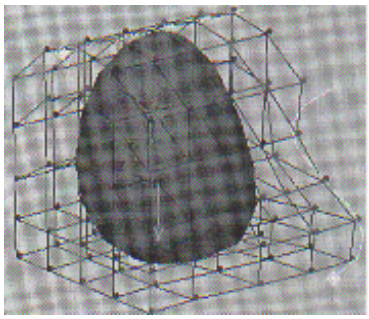


Figura 12: Lattice de pontos de controle

controle da lattice gerando uma mudança dos pontos do objeto, ou uma deformação do mesmo(vê figura 12).

Com essa técnica pode-se criar deformações locais ou globais. A forma mais simples de lattice é o paralepípedo. Criar outras formas de lattice, que dê um melhor controle da forma desejada para deformação, é embaraçoso e difícil. Para um melhor refinamento da superfície do objeto é necessário uma grande quantidade de pontos de controle que tornará mais difícil a sua manipulação para a obtenção do efeito desejado, como também aumentará o custo de processamento. Quais pontos e como manipulá-los para obter-se a deformação desejada? Para objetos simples talvez não seja um grande problema, mas, para objetos complexos é um problema difícil e em certos casos sem solução. Segundo Sederberg, não dá para realizar escalamento (em duas ou menos direções) e curvamento gerais. Neste artigo, ele usou uma lattice paralepipedal e a função que relaciona o objeto aos pontos de controle foi "Trivariate Tensor Product Bernstein Polynomial". Afirma ainda, que a função de deformação pode ser formulada em termos de outras bases polinomiais, tais como produto tensorial B-splines ou produto não-tensorial de polinomiais de Bernstein.

Em 1992, no artigo [10], Jonh F. Hughes e Henry Kaufman, apresentaram quatro problemas nas deformações via pontos de controle:

a. a forma exata é difícil de ser encontrada;

b. o posicionamento exato dos pontos do objeto é difícil de ser encontrado;

c. usuários não familiarizados com splines não compreenderão o propósito dos pontos de controle e os resultados de seus movimentos;

d. os pontos de controle tornam-se difíceis para manipular quando ocultados pelo objeto sendo deformado, ou quando existem muitos deles abarrotados na tela.

Neste artigo, Hungles e Kaufman fazem o processo inverso. O usuário seleciona um ponto de um objeto e então move-o para uma posição onde este ponto do objeto deverá estar após a deformação. Computa-se as alterações necessárias para os pontos de controle do spline FFD que induzirá esta mudança.

A função de deformação usada é um "Trivariate B-spline Tensor Product". Segundo eles, têm-se um melhor controle local e uma garantia da continuidade quando alguns pontos de controles são movidos. Com esta abordagem, elimina-se a necessidade de exibir os pontos de controle (e sua lattice de controle associada) e a interface fica mais transparente, permitindo ao usuário se concentrar em outros detalhes.

Deformação usando eixos

As deformações de eixos, proposto por Coquillart, Lazarus e Jancene em [11], segundo eles, fornece uma representação mais compacta na qual uma primitiva uni-dimensional, tal como um segmento de reta ou curva, é usado para definir uma deformação global implícita. Nessa técnica, usa-se: duas curvas W e R, que inicialmente são as mesmas, o cálculo do ponto mais próximo e referenciais de Frenet sobre W e R. Calcula-se os pontos de R mais próximos dos pontos do objeto. A deformação de um ponto P no objeto é obtida manipulando diretamente a curva W, ficando a sua cópia R como referência, e computando a transformação que transforma o referencial de Frenet, no ponto mais próximo de P em R, no referencial de Frenet, no ponto correpondente sobre W. Tal deformação é uma porção da transformação obtida acima. A proporção é baseada numa interpolação da distância do ponto mais próximo de P em R ($\|P - R(p)\|$), entre dois cortes R_{in} e R_{out}. Esta técnica tem um problema nos pontos onde a curva tem curvatura nula, pois, nesse caso o referencial de Frenet não está bem definido. Também, o escalamento, a rotação e a translação são tratadas numa só transformação, não são separadas, dificultando o refinamento da deformação, o tratamento das distorções e descontinuidades na superfície do objeto.

Deformação usando Arame

No artigo [4], Karan Singh e Eugene Fiume, propuseram uma técnica de deformação semelhante a técnica mencionada acima, deformações de eixo, porém, tratando o escalamento, a rotação e a translação de forma diferenciada, isto é, separadas. Explicaremos, a seguir, os detalhes dessa técnica.

Pâmetros Envolvidos (WIRE)

Um arame (wire) é uma curva cuja manipulação deforma a superfície de um objeto associado próximo a mesma. Com mais rigor, um arame é uma tupla $< W, R, s, r, f >$, onde W é a curva paramétrica de forma livre (wire), R é a curva de referência, cópia congruente de W, s é um escalar que controla o escalamento em torno da curva R, r é um valor escalar definindo um raio de influência em torno da curva R e f é uma função escalar $f : R^+ \rightarrow [0, 1]$, chamada função densidade, tal que f é no mínimo derivável, com derivada 1^acontínua, e monotonicamente decrescente, satisfazendo

$$f(0) = 1,\ f(x) = 0 \textit{ para todo } x \geq 1,\ f'(0) = f'(1) = 0.$$

Essa função servirá para definir a função F (fator de atenuação) que determinará a atenuação da deformação a medida que os pontos, no volume de deformação de raio r, se afastam da curva de referência R. F é definida como

$$F(P, R) = f(\frac{\|P - R(p)\|}{r}),$$

onde $R(p)$ é o ponto de R mais próximo de P (se existir mais de um pegue o de meno parâmetro). Observe que $F(P, R) = 0$, se P está fora ou sobre a superfície do volume de compensação, $F(P, R) \neq 0$, se P está no interior do mesmo, sendo que $F(P, R) = 1$, se P está sobre R.

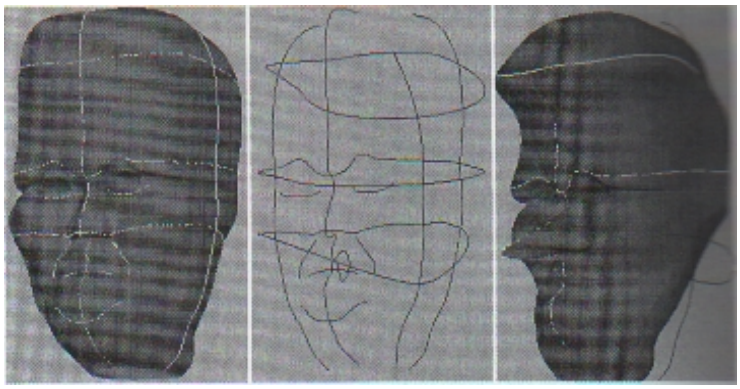


Figura 13: Objeto demarcado por arames

Deformação provocada por um arame

Quando um objeto é limitado ou demarcado pelo arame(figura 13 acima), o domínio de influência do arame está demarcado pela superfície do volume de compensação de raio r definida em torno da curva de referência R. Subsequente manipulação de W resulta numa mudança entre W e R, que é usada juntamente com F e s para definir a deformação. A atual deformação aplicada para um ponto é modulada pela sua influência, calculada quando o objeto estava limitado pelo arame. O algoritmo é o seguinte: o objeto O está limitado por um arame $< W, R, s, r, f >$, os parâmetros p dos pontos mais próximos (calcula-se uma vez, sendo recalculado caso a curva de referência mude) e $F(P, R)$ são calculados para cada ponto P do objeto. Manipula-se diretamente no arame W, ficando R (cópia de W) como referência, para gerar uma deformação no objeto, computada para $F(P, R) > 0$ como

1. **Escalamento**

   Escale P como

   $$P_s = P + [P - R(p)] \times \ [1 + (s - 1) \cdot F(P, R)]$$

   onde P_s é o ponto P deslocado na direção do vetor $P - R(p)$.

2. **Rotação**

   Rotacione P_s, em torno do eixo de direção $W'(p)$ X $R'(p)$, no ponto $R(p)$, de um ângulo $q \cdot \ F(P, R)$, sendo q o ângulo entre os vetores tangente de $W'(p)$ e $R'(p)$, obtendo o ponto P_r.

3. **Translação**

   Translade o ponto P_r proporcionalmente ao afastamento entre $W(p)$ e $R(p)$, gerando o ponto deformado P_{def}

   $$P_{def} = P_r + [W(p) - R(p)] \cdot \ F(P, R)$$

Veja figura 14 abaixo.

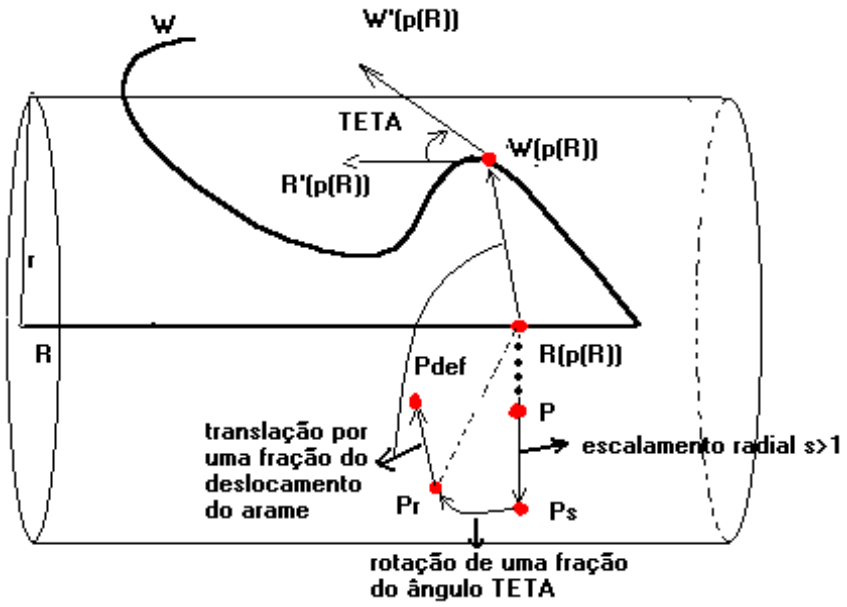


Figura 14: Deformação de um ponto provocada por um arame

Quando a curva arame W e a curva de referência R são criadas inicialmente, colocando $r = 0$ como default, não ocorrerá deformação, pois $R = W$. Outro detalhe é que pontos sobre a superfície do volume de compensação e exterior ao mesmo, não serão deformados. Pontos sobre a curva de referência R serão deformados nos pontos correspondentes sobre a curva arame W. A medida que o ponto P, dentro do volume de compensação, se afasta de R, a deformação irá diminuindo, tendo sua suavidade relacionada ao fator de atenuação $F(P, R)$.

Controle de descontinuidade

Apesar do uso do fator de atenuação $F(P, R)$, surgem problemas de descontinuidade da superfície do objeto como, por exemplo, decaimentos abruptos(veja figura 15 abaixo). Para preservar as propriedades de continuidade da mesma, exige-se um afastamento suave baseada na região selecionada. Na busca de solucionar tais problemas, posiciona-se *locators* ao longo da curva arame, possibilitando definir para cada locator, parâmetros r, s e f distintos, de acordo com a necessidade, o que antes não era possível. Entre dois locators consecutivos, interpola-se os atributos. Obtêm-se um controle local e simétrico radialmente e em torno de uma curva arame(veja figura 16 abaixo). Outro artifício é colocar *curva domínio* controlando qual parte do objeto será deformada, pois, a mesma tem a função de delimitar qual região, dentro do volume de compensação, sofrerá deformação e qual não sofrerá os efeitos da mesma. Rapidamente, se D é uma curva domínio, enquanto o ângulo entre os vetores $[D(p_d) - R(p_r)]$ e $[P - R(p_r)]$, onde p_d e p_r são os parâmetros do pontos mais próximos de P em D e R, respectivamente, faça

$$F(P, R) = f(\frac{\|P - R(p_r)\|}{\|P - D(p_d)\|})$$

e para pontos fora do domínio definido pela curva domínio, pode ser usado o raio de afastamento convencional r. Observe que os pontos onde $\|P - R(p_r)\| \geq \|P - D(p_d)\|$ não sofrerão deformação, delimitando uma região, no volume de compensação, que não sofrerá efeitos da deformação.

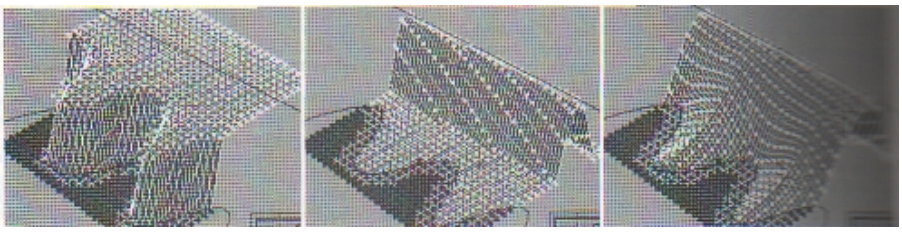


Figura 15: decaimentos: 1. vertical 2. horizontal 3. suave

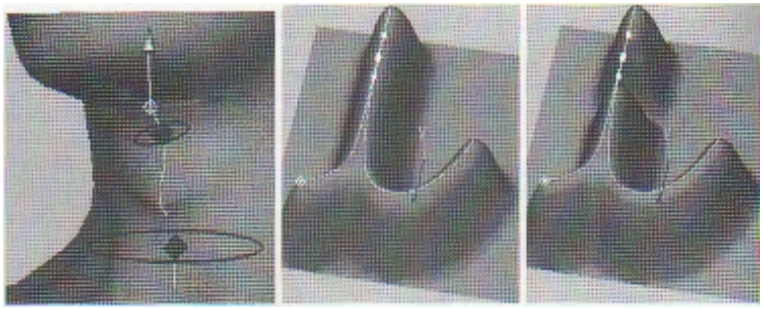


Figura 16: Usando locators para variar os parâmetros f, s e r.

Mas, surge um problema de descontinuidade nas vizinhanças de pontos onde o ângulo entre os vetores é reto ($\frac{\pi}{2}$). Remove-se da seguinte maneira: tome $\delta \in (0, \frac{\pi}{2})$, para $\hat{a}ngulo \in [0, \delta]$,

$$F(P, R) = f(\frac{\|P - R(p_r)\|}{interpole(\hat{a}ngulo)})$$

onde $Interpole$ dá um valor interpolado suavemente de r para $\|P - D(p_d)\|$ quando varia de 0 a δ. Veja figura 17.

Deformação provocada por vários arames

Note que um ponto pode pertencer a n volumes de compensações, isto é, sofrer a influência das deformações de n arames. A deformação final é a combinação das deformações de cada arame, da seguinte forma:

$$P_{def} = P + \frac{\sum_{i=1}^{n} \Delta P_i \|\Delta P_i\|^m}{\sum_{i=1}^{n} \|\Delta P_i\|^m} \quad , \quad \text{onde } \Delta P_i = P - P_{def_i}.$$

Observe que para $n = 1$ a definição acima coincide com a que foi dada para um arame. A medida que m cresce positivamente a deformação de maior magnitude dominará as demais e quando m decresce negativamente a deformação de menor magnitude dominará as demais.

Conclusão

Concluindo, percebe-se algumas das dificuldades existentes nas definições e manipulações dos pontos de controles (parâmetros de controles) para a manipulação da estrutura geométrica do objeto, isto é, a dificuldade de se obter a forma desejada manipulando os mesmos, como também, que esta manipulação seja a mais intuitiva possível. Outra dificuldade perceptível, é que a técnica não seja customizada. É difícil encontrar uma técnica que tenha essas três qualidades de forma satisfatória.

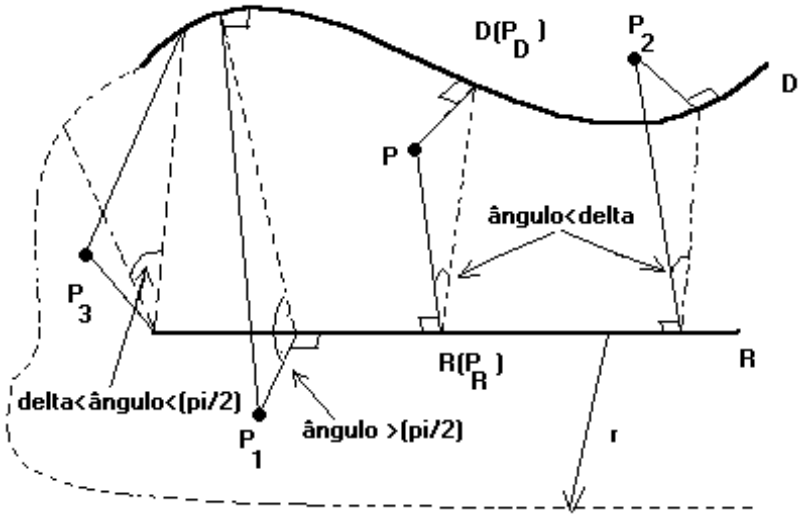


Figura 17: Curva domínio e região de influência da deformação

2.3 Interpenetracoes

Como mencionado na introdução deste capítulo, após a preocupação com a estrutura geométrica dos objetos, surge a preocupação de como tratar dos problemas de contatos entre eles, e das colisões que ocorrem, tendo que evitar as penetrações dos mesmos. Por exemplo, o problema de colocar um vaso sobre a mesa, temos que determinar os pontos de contato do vaso com a mesa, e vice-versa, e o tempo de detecção do contato, buscando evitar que haja penetrações. Deverá existir uma força que mantém os dois nesse estado depois do contato ter ocorrido. Um outro exemplo, um ator sintético vestido, em movimento, em alguns trechos do tecido ocorrerão colisões do mesmo com o corpo do ator sintético. Temos que determinar os pontos de colisões e detectar o tempo que as mesmas ocorrem, evitando que haja a penetração e fazendo que os mesmos repilam um ao outro. Fazer essa tarefa de posicionar um objeto 3D numa cena usando manipulação direta de um mouse, tracball ou introduzindo um conjunto de dados para orientação do objeto e parâmetros de translação, onde o usuário verifica pela visão se há ou não penetrações e estão fisicamente balanceados, é um tanto cansativo e enfadonho, principalmente quando se trata de objetos curvos. É difícil atingir uma configuração estável, isto é, atingir um grau aceitável de realismo.

Apresentaremos a seguir, uma técnica proposta por Jonh M. Snyder, no artigo [5], que busca o posicionamento de superfícies curvadas, numa cena, sem que haja penetrações.

Simulação Física

Este artigo, não se preocupa com a precisão física, mas, busca conseguir um estado(configuração) estável de alguns estados arbitrários, criando mudanças que são intuitivas e controláveis. A simulação da precisão física é a seguinte:

1. o corpo é movimentado através de uma série de estados discretos, cada um dos quais obedece limitações que asseguram a não penetração de corpos;

2. forças e torques não são as que um corpo dinâmico sofre, são escolhidas para obter a rápida convergência do corpo;

3. o tempo exato quando transições de pontos de contatos ocorrem não é calculado, fazendo-se as transições depois de cada etapa discreta;

4. para preservar as limitações de não peneteração são feitos pequenos ajustes não físicos de posicionamento e orientação;

5. somente um corpo de cada vez pode ser movido.

Colisão e Contato

Duas formas de detecção de colisão são requeridas para o posicionamento interativo. Detecção de colisão simples computa quando um objeto interfere com alguns outros objetos, e é usada na escolha de um estado de não interferência do qual inicia ou encerra a convergência para a estabilidade. A segunda forma de detecção de colisão é usada para computar quando um corpo já em contato com outros corpos intersecta em pontos adicionais.

Na detecção de colisões, a superfície é representada por uma malha triangular, com os vértices sobre a superfície analítica. Aproximar a superfície analítica por uma malha triangular pode nos dá o transtorno de ocorrer colisões entre as superfícies analíticas e não entre as malhas ou vice-versa (veja figura 18(a) e 18(b)). No primeiro caso, a penetração pode ocorrer até ocorrer a colisão na malha triangular ou criar um túnel, podendo ser ignorado, desde que os objetos não violem demais as condições de penetração, ou violam as condições de penetração por uma distância limitada pela aproximação de erro das duas malhas. No segundo caso, um erro na iteração numérica detecta a falha.

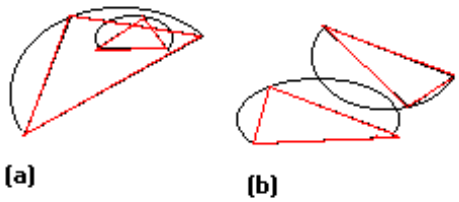


Figura 18: Inconsistência de colisões: superfície$X$malha

O posicionamento da colisão é aproximado encontrando as intersecções de dois triângulos e, para cada um deles, interpola-se baricentricamente as coordenadas paramétricas em cada vértice obtendo as coordenadas no ponto de intersecção. Os dois pontos são usados como pontos iniciais num método interativo (Newton-Rapson Multidimensional) que produz pontos de contatos sobre a superfície real. Nesse processo de ajuste é usado uma descrição funcional da superfície. A detecção de colisões é organizada na seguinte hierarquia de árvore: cada triângulo está numa caixa limitante, que é a primeira a detectar a possibilidade de colisão. Ocorrida a intersecção das caixas limitantes de dois triângulos, passa-se a computar as intersecções dos triângulos e, caso aconteçam, de posse dessas intersecções, computa-se o ponto de intersecção da superfície analítica, como mencionado acima.

Quando um ponto de contato entre dois corpos é determinado, define-se uma esfera de nigligência em torno dele para que o mesmo não seja computado novamente na busca de pontos de intersecções adicionais(figura 19). Quando a intersecção das caixas limitantes ou dos triângulos estiverem contidas na esfera de negligência, descartamos os mesmos. O raio da esfera de negligência é um pequeno valor determinado pela distância dos pontos de contatos movendo-se da aproximação poligonal para a superfície analítica durante o processo de criação de contato.

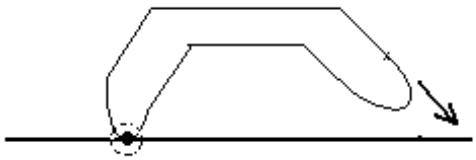


Figura 19: Esfera de negligência

As descrições funcional e poligonal são também usadas para rastrear o movimento dos pontos de contatos e para determinar quando o objeto não possui contato com outro depois de sofrer mudança de posição, respectivamente.

Pontos de Interface

A abordagem deste artigo cria só uma quantidade suficiente de pontos, chamados de pontos de interface, consistindo de pares de pontos sobre dois corpos, para prevenir a penetração e atingir um estado estável. Pontos de contatos são rastreados(incrementalmente atualizados) quando o corpo se move de estado para estado usando iteração numérica.

Durante a detecção de colisão, uma esfera de negligência é colocada ao redor de cada ponto de interface evitando a detecção de colisões já sendo manipuladas.

São dois os tipos de pontos de contatos: pares de contatos e pares extremais(figura 20). Os corpos realmente tocam-se em pares de contatos. Forças que balanceam as forças externas, como gravidade, são aplicadas aos pares de contatos. Essas forças não são usadas para grudarem os objetos, mas para repelí-los, evitando de interpenetrarem. As forças que grudam os objetos são usadas para transformarem pares de contatos em pares extremais.

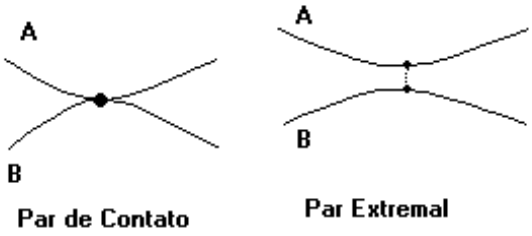


Figura 20: Pontos de Interface

Pares extremais são pares de pontos sobre os dois corpos que estão numa distância mínima. Os corpos não estão em contato numa vizinhança em torno de pares extremais e nenhuma força é aplicada aos mesmos. Quando uma força grudante é detectada pares de contatos são convertidos em pares extremais e quando a função de distância de separação nos pares extremais torna-se negativa os mesmos são convertidos em pares de contatos.

Pseudo-Forças

As pseudo-forças são usadas para mudarem incrementalmente a posição do corpo ativo, com o intuito da obtenção da convergência para o estado estável. São três as espécies de pseudo-forças: força externa, que representa a gravidade e cutucadas especificadas pelo usuário, força contato, que são aplicadas nos pares de contatos para balancear as forças externas, e força residual, que representa a força resultante (força externa - força contato) aplicada para ajustar a posição dos corpos. Quando a força residual é suficientemente pequena considera-se o corpo em repouso e automaticamente o mesmo é posicionado descansando.

Comentário Final

As ferramentas para configurações físicas de posicionamento de objetos curvados são custosas, porém tem requintes visuais melhores. Neste artigo, é proposta uma ferramenta que esqueçe um pouco os requinte visuais, isto é, os estados não representam o movimento real do objeto, mas ganha-se em custo, pois, os estados são feitos em etapas discreta de tempo e são usadas pseudo-forças. Chega-se mais rapidamente e de forma precisa no posicionamento adequado do objeto ativo na cena, enquanto os outros objetos permanecem estáticos.

2.4 Controle da camera atraves da lente

Problema

Como posicionar e controlar uma câmera virtual para obter uma imagem X a partir de um modelo Y ? (figura 21)

- Há diferentes parametrizações. Exemplo:
  3 parâmetros (para posicionamento), 3 (para orientação) e 1 do foco.
- Outros parâmetros são: Ângulo de visão, profundidade de campo, etc.

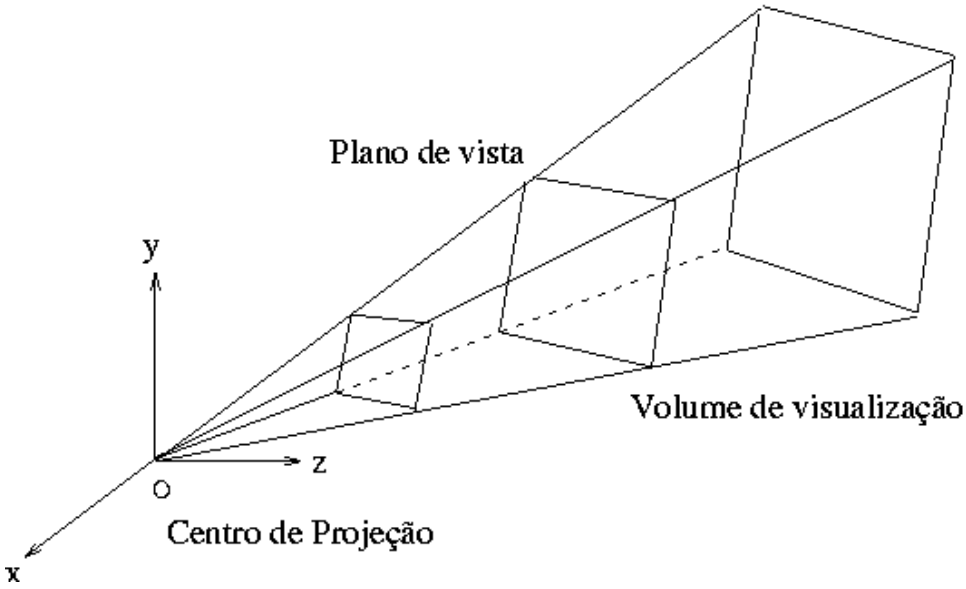


Figura 21:

Reformulando:
Como especificar os parâmetros da câmera para conseguir a imagem que desejo?

Especificação dos parâmetros

- Especificação diretamente relacionada com o problema de interface com o usuário.

1. Especificação Direta
   - Parâmetros são fornecidos explicitamente
   - Problema.- Quando o dispositivo de entrada é bidimensional o desenvolvimento de controles intuitivos para rotações 3D é um problema difícil.
2. Especificação inversa
   - Determinação indireta dos parâmetros da câmera.
   - Controle iterativo da câmera virtual através da manipulação direta sobre a imagem. Exm: figura 22.

   Gostaria que o ponto P fosse projetado no centro da tela. Como encontrar os parâmetros q da câmera para esse fim ?.

Modelo matemático

Seja $p \in R^3$ o ponto no espaço da cena a ser projetado na tela virtual (espaço imagem) e m o ponto imagem correspondente.

$$m = h(Vp) \qquad p = (kx, ky, kz, k)^t$$

onde h função que converte as coordenadas homogêneas para coordenadas imagem 2D, assim

$$h(p) = [\frac{kx}{k}, \frac{ky}{k}] = [x, y]$$

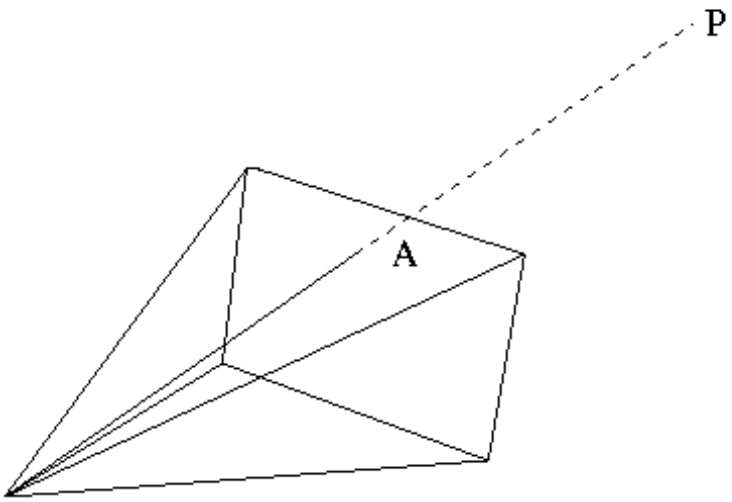


Figura 22:

A matriz V representa as transformações de visualização. É um produto de várias matrices cada uma sendo uma função de um ou mais parâmetros.

$$V = PRT$$

onde RT realizam a mudança do espaço da câmera para o espaço da cena.

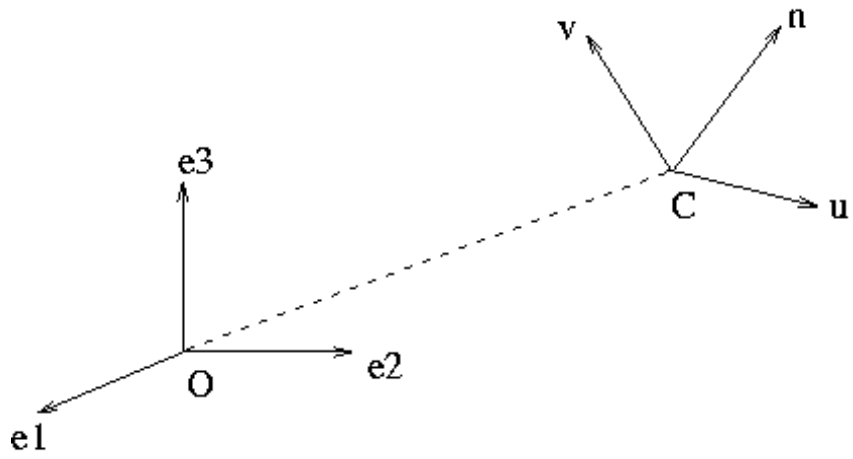


Figura 23:

P Matriz de Transformação perspectiva - Modelo Básico fig. 24.
Por similaridade de triângulos :

$$\frac{x}{f} = -\frac{x}{z-f} \rightarrow x = \frac{fx}{f-z} \qquad \frac{y}{f} = -\frac{y}{z-f} \rightarrow y = \frac{fy}{f-z}$$

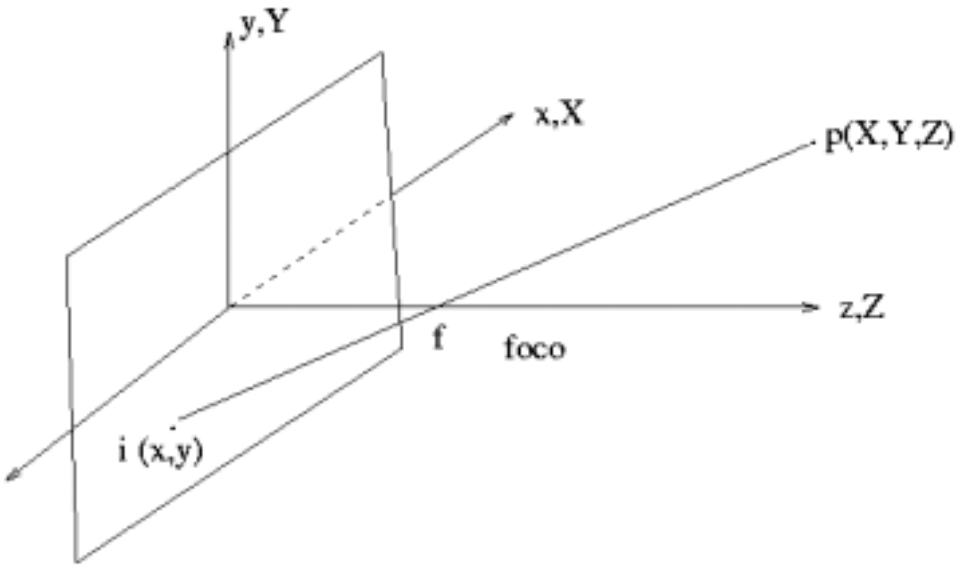


Figura 24:

em forma matricial

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 1 \end{bmatrix} \begin{bmatrix} kx \\ ky \\ kz \\ k \end{bmatrix} = \begin{bmatrix} kx \\ ky \\ kz \\ -\frac{kz}{f} + k \end{bmatrix}$$

$$p' = \begin{bmatrix} \frac{fx}{f-z} \\ \frac{fy}{f-z} \\ \frac{fz}{f-z} \end{bmatrix} \qquad h(p') = h(PRTp) = h(Vp) = \begin{bmatrix} \frac{fx}{f-z} \\ \frac{fy}{f-z} \end{bmatrix}$$

Problemas na especificação inversa

1. Não há problema de interação intuitiva usuário - sistema (tela 2D)
2. $m = h(Vp)$ Sistema não linear. Em geral a inversa não é garantida. Solução pode não existir ou não ser única.

Contornando os problemas

1. Procura-se as derivadas no tempo dos parâmetros da câmera a partir das derivadas no tempo dos controles sobre a imagem.

$$\dot{m} = h'(Vp)(\frac{\partial(Vp)}{\partial q})\dot{q}$$

fazendo $J = h'(Vp)(\frac{\partial(Vp)}{\partial q})$ temos $\dot{m} = J_{2\times n}\dot{q}_{n\times 1}$ que é um Sistema Linear. Porém J não é quadrada.

2. Usa-se otimização restringida
   Sujeito à restrição que $\dot{m} = \dot{m_0}$, minimiza-se a desviação de $\dot{q}$ em relação a um valor especificado $\dot{q_0}$. Minimizar $E = \frac{(\dot{q}-\dot{q_0})(\dot{q}-\dot{q_0})}{2}$ sujeito a $\dot{m} - \dot{m_0}$
   Chega-se a

$$JJ^t\lambda = \dot{m_0} - J\dot{q_0}$$

   e finalmente

$$\dot{q} = \dot{q_0} + J^t\lambda$$

   E.D.O de primeira ordem com valor inicial.

Vários controles

- Para r pontos imagem de controle, combina-se as r equações em uma única.
- O Sistema para um único ponto

$$\dot{m}_{2\times1} = J_{2\times n}\dot{q}_{n\times1} \quad \text{passa a ser}$$

$$\dot{m}_{2*r\times1} = J_{2*r\times n}\dot{q}_{n\times1}$$

- A função objetivo passa a ter r restrições.

Determinando o valor inicial q_0

- Pode ser uma estimativa inicial do usuário.
- Pode ser calculada pela formula

$$\dot{q_0} = k_c J^t(m_c - m)$$

  onde k_c é uma constante e m_c é a posição do cursor na imagem (previo ponto de controle).

- Minimiza-se a desviação dos novos parâmetros q em relação aos parâmetros obtidos a partir do prévio ponto de controle, quando o usuário está especificando dinamicamente r pontos de controle.

Realimentação da posição

- Especifica-se velocidade ao invés de posição, porém há a informação de posição.
- Usa-se esta informação para melhorar a precisão na especificação da velocidade de m.
- $\dot{m} = \dot{m_0} - k_f(m - m_0)$ onde k_f é uma constante de realimentação e m_0 a desejada posição para m no tempo corrente.
- Quando m está sobre m_0 o termo de realimentação é zero, quando não, a velocidade $\dot{m}$ é orientada na direção que reduz o erro.

Considerando um ponto em movimento

- Já que p depende agora do tempo, um termo adicional aparece na nossa equação.

$$\dot{m} = h'(Vp)(\frac{\partial(Vp)}{\partial q})\dot{q} + h'(Vp)V\dot{p}$$

3 Capitulo: Arquitetura

3.1 Introducao

Com a necessidade e uso de interfaces 3D, surgiu a necessidade de criação de ferramentas gráficas para prototipação e desenvolvimento desta interfaces. Várias ferramentas, até esta data, foram apresentadas. Algumas orientadas a objetos, como as que são apresentadas na secção 3.2 e 3.3, outras não. Ferramentas visuais que possibilitam não programadores desenvolverem tais interfaces, como a plataforma apresentada na secção 3.3, outras, como o Open Inventor, são de difícil uso para não programadores. Na secção 3.2, apresentaremos a plataforma gráfica "Open Inventor" que tem como objetivo principal a construção de objetos 3D, com detalhes de requintes visuais como: iluminação, textura, etc.. Na secção 3.3, apresentaremos uma plataforma gráfica, que foi apresentada no artigo "Arquitetura de uma Interface 3D Extensível", que cria e manipula diretamente objetos gráficos, dando um passo inicial na manipulação direta de objetos gráficos dentro da área de desenho.

3.2 Uma Toolkit graoca 3D orientada a objeto

Este artigo apresenta uma toolkit orientada a objetos para desenvolvedores de aplicações gráficas interativas. Já que técnicas de manipulação direta 3D em adição aos *widgets* convencionais, têm sido em geral ignoradas em previos trabalhos, os autores têm como objetivo principal facilitar para os programadores a criação de aplicações gráficas 3D que empreguem tais técnicas.

Segundo os autores uma toolkit interativa deve fornecer um conjunto rico e extensível de objetos 3D e deve suportar manipulação direta. Há três áreas fundamentais nas quais uma toolkit pode possibilitar o desenvolvimento de programas interativos 3D.

- Representação de objetos.- Dados gráficos devem ser armazenados como coleções de primitivas de desenho usadas para representá-los.
- Interatividade.- Um modelo evento para programação interativa direta deve ser integrado com a representação de objetos gráficos.
- Arquitetura.- As aplicações não devem ter que adaptar a representação do objeto ou políticas de interação impostas pela toolkit. Ao invés disso, os mecanismos de toolkit devem ser usados para implementar as políticas necessárias e desejadas.

A toolkit descrita neste artigo atende as necessidades mencionadas definindo um *framework* orientado a objetos para descrever cenas contendo objetos 3D e operações sobre eles. Objetos 3D são representações abstratas que podem ser renderizadas quando desejado. Os desenvolvedores são capaces de adicionar novos objetos de aplicação específica a ser incorporados à toolkit. Esta extensibilidade também inclue operações sobre objetos tais como novos métodos de renderização e cálculos geométricos.

A biblioteca toolkit 3D consiste de três secções principais:

1. Utilidades da Toolkit
    - Utilidades Xt e componentes
    - Utilidades GL

2. Toolkit 3D

    - Interação
    - Base de dados de cenas
    - Kits de nós

3. Suporte do sistema

    - sistema operativo
    - sistemas de janelas
    - biblioteca gráfica

O fundamento da toolkit é a base de dados da cena. Ela armazena representações dinâmicas de cenas 3D como grafos de objetos chamados de nós.
Várias classes de nós implementam geometrias, propriedades e comportamentos diferentes. A base de dados fornece um conjunto de ações que podem ser aplicadas a cenas ou partes de cenas, por exemplo: renderização, cálculo de uma *bounding box*, etc.

A secção de interação da toolkit introduz classes de eventos e nós que processam eventos. Um tipo especial de nó é um manipulador, que responde a eventos de interação e edita outros nós na base de dados.

A terceira secção da toolkit define kits de nós, que facilitar criar bases de dados consistentes estruturadas. Cada objeto kit de nó combina algum subgrafo base de dados da cena, regras *attachment* e outras políticas numa única classe.

3.3 Arquitetura de uma Interface 3D Extensível

No artigo [8], é apresentada uma plataforma para criação e para manipulação direta de primitivas 3D. Esta ferramenta se baseia na construção de objetos a partir de um conjunto de primitivas gráficas e restrições aplicadas a estas primitivas, conectadas por uma linguagem visual.

Segundo os autores, esta ferramenta permite que tanto programadores como não programadores possam prototipar rapidamente interfaces 3D, pois, não é implementada como bibliotecas de classes, facilitando a visualização de relações complexas. Também, o uso por não programadores é facilitado porque a manipulação é diretamente nas primitivas visuais. Outro detalhe importante, permite que o projeto seja extensível, isto é, novas primitivas e novos comportamentos entre primitivas possam ser facilmente criados. Isto é facilitado pelo *design* orientado a objeto.

Link - Classe - Slot -Métodos

Nesta plataforma, um conjunto de propriedades define uma primitiva de forma única. As relações entre propriedades de uma primitiva gráfica e propriedades de outra, chamadas de *links,* permitem criar um objeto com comportamentos de interações complexas. Quando um link é criado ou quando uma primitiva é manipulada pelo usuário, as propriedades são alteradas. No caso da manipulação, o link não deve ser violado.

Um objeto gráfico é uma instância de uma classe que representa a primitiva. Uma classe é composta de campos de dados chamados *slots*, que representam as propriedades características da primitiva.

O desenvolvedor fornece os métodos *Inquiry* e *Assignment* para obter e alterar o valor de um slot, e a medida que novas restrições são adicionadas a um slot, estes métodos devem ser atualizados. É introduzida uma linguagem que fornece a base para a construção visual de restrições(relações) entre primitivas, e consiste de classes representadas pelas suas primitivas 3D, onde os slots são usados para estabelecer as restrições. É usado o encapsulamento de coleções complexas de primitivas mais restrições para criar novas classes a serem adicionadas à ferramenta, podendo ser parametrizadas e utilizadas para recriar estrutura complexas.

A classe nada mais é que a abstração da geometria e comportamento de uma primitiva. A mesma define a representação visual de um objeto gráfico, cujo comportamento desta classe é definido pelos seus slots e suas formas de interações. Assim, cada primitiva tem uma representação geométrica e um ou mais slots que definem seu comporatamento. A relação de restrição entre duas classes são criadas conectando seus slots, isto é, a conecção é através da seleção do slot de uma primitiva destino e uma primitva origem, criando-se um fluxo de dados bi-direcional entre o slot de destino e a primitiva de origem. Esses slots representam quantidades da classe que podem sofrer restrições.

Primitivas

Nesta plataforma, as primitivas básicas são:

1. Ponto → slot: posição
2. Vetor → slot: posição,direção(sentido) e comprimento
3. Plano → slot: normal, centro, size1(largura) e size2(comprimento)
4. Objeto Gráfico → slot: os mesmos do plano + red, blue e green

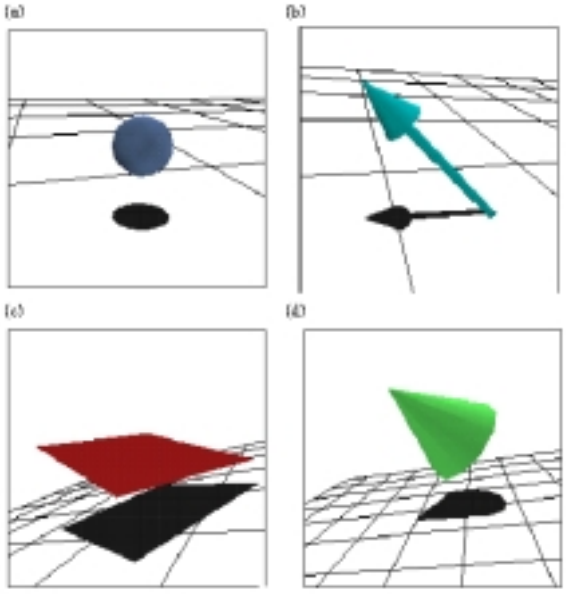


Figura 25: Primitivas: (a)Ponto (b)Vetor (c)Plano (d)Objeto geométrico

Técnicas de interação

As técnicas de interação desta plataforma especificam como modificar um slot durante interações do usuário, mantendo as restrições estabelecidas por outros slots.

Vejamos, com um exemplo, como obter e atribuir valores a um slot: se o slot de posição de um Vetor está conectado a um Ponto e o vetor está sendo resolvido, o método de pesquisa(Inquiry) da posição do vetor, que calcula o valor de um slot sob restrições, é invocado com o Ponto passado como seu parâmetro (figura 26).

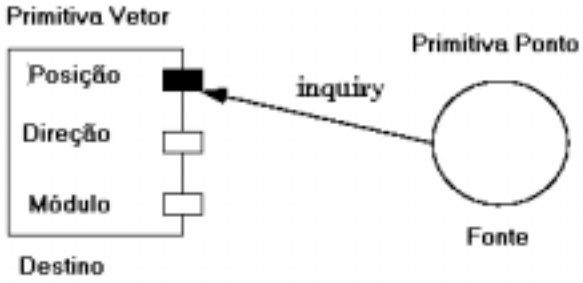


Figura — Fluxo de dados entre a Primitiva Vetor e a Primitiva Ponto quando vinculado o slot Posição do Vetor para um ponto

Figura 26: Fluxo de dados

Assim, o método inquiry obtém o slot de posição do Ponto (figura 27).

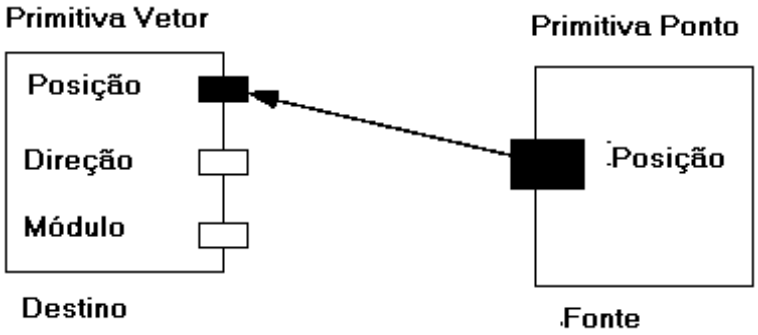


Figura — Resultado da pesquisa(Inquiry) para um slot posição do vetor para um ponto

Figura 27:

Agora, quando a posição do vetor é alterada, o método de atribuição(assignment) determina como a restrição do slot é mapeada de volta para a primitiva a qual ele está restrito, isto é, a nova posição é atribuída de volta ao Ponto. Neste caso, deve-se selecionar o Vetor como primitiva destino e o Ponto como primitiva origem. A interface gráfica indicará um link pendente, entre as duas primitivas, através de uma seta (figura 28). O método de atribuição dá à retrição o comporatamento bi-direcional. Esses métodos podem também chamar funções de outros slots para resolverem suas restrições.

Em seguida, o slot de interesse na primitiva destino (Posição) deve ser selecionado (na janela MOTIF) e escolhe-se a restrição dentro de uma lista de opções.

Um método Establish estabelece a restrição e, então, um método Resolution da primitiva Ponto é chamado. O método Resolution é definido para cada primitiva

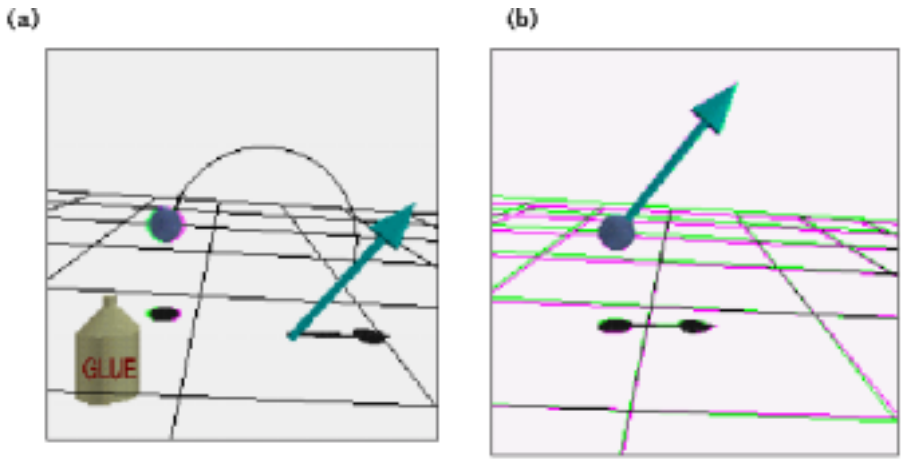
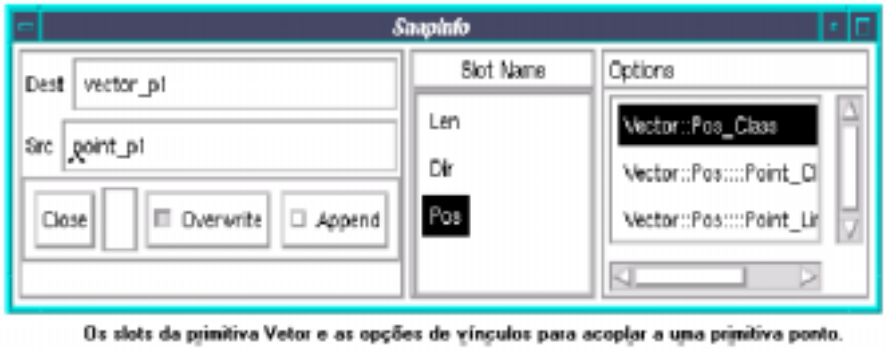


Figura — Primitivas ligadas (a) link pendente do slot posição do vetor para uma primitiva ponto (b) link estabelecido

Figura 28: Primitivas acopladas



Os slots da primitiva Vetor e as opções de vínculos para acoplar a uma primitiva ponto.

Figura 29: Janela Motif

para resolver seus slots, atualizar sua representação gráfica e chamar os métodos Resolution das outras primitivas conectadas a ele.

Interação com o Ponto
↓
Altera slot Posição do Ponto
↓
Método Resolution do Ponto
↓
Método Resolution do Vetor
↓
Resolve o slot Posição do Vetor

Encapsulamento

Nesta plataforma existem três tipos de encapsulamentos:

1. Encapsulamento Estrutural - reproduz um conjunto de primitivas conectadas. Por exemplo, conectar um vetor a dois pontos(figura 30).

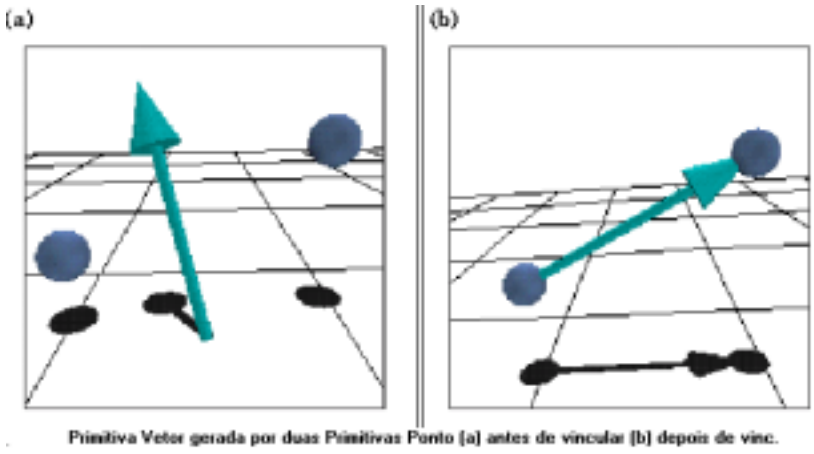


Figura 30: Encapsulamento Estrutural

Essa construção usada repetidas vezes cria o objeto geométrico chamado *line(*linha*).*

2. Encapsulamento de Classe - o que diferencia este do anterior é que este cria uma nova classe, como se estivéssemos colocando uma nova primitiva na plataforma. No exemplo da linha três slots: start-point, end-point e center-span(figura 31).

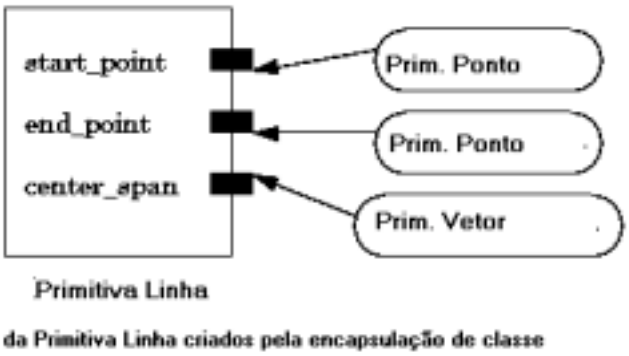


Figura 31: Encapsulamento de Classe

3. Encapsulamento Parametrizado - são encapsulamento recriados a partir de um conjunto de primitivas usadas como parâmetros. Por exemplo, o objeto do tipo linha poderia ser criado a partir de seus dois Pontos extremos.

É possível a inserção de restrições que limitam as variáveis do slot a intervalos desejados como, por exemplo, restringir Pontos a segmentos de retas(figura 32).

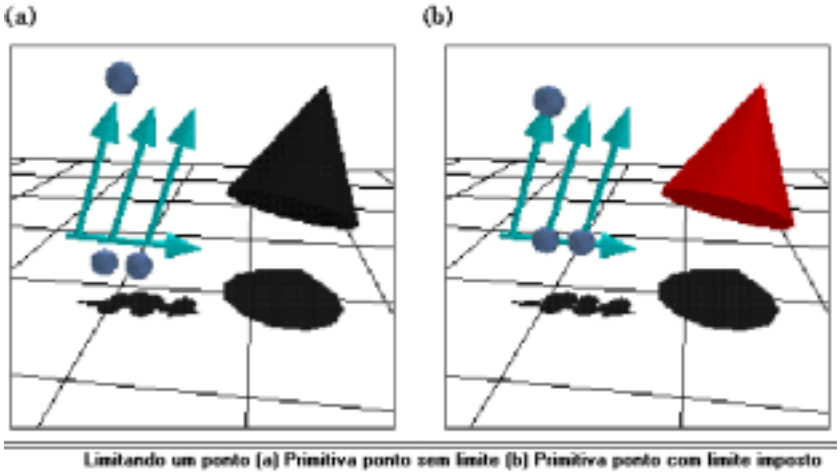


Figura 32: Encapsulamento Paramétrico

Referências

[1] The Rockin' Mouse: Integral 3D Manipulation on a Plane

Ravin Balakrishnan - Dept. of Computer Science - Univesity of Toronto & Alias\Wavefront

Thomas Baudel, Gordon Kurtenbach, George Fitzmaurice - Alias\Wavefront

[2] Snap-Dragging in Three Dimensions

Eric A. Bier - Xerox Parc

1990, pp. 193 - 204

[3] Direct Manipulation Techniques for 3D Objects Using 2D Locator Devices

Gregory M. Nielson - Arizona State University

Dan R. Olsen Jr. - Brigham Young University

Proceedings of the 1986 workshop on Interactive 3D graphics (October - 24, 1986, Chapel Hill, NC USA) pp. 175 - 182. 23

[4] Wires: A Geometric Deformation Technique

Karan Sing and Eugene Fiume - Alias\Wavefront

Computer Graphics, 1998, pp. 405 - 414.

[5] An Interactive Tool for Placing Curved Surface without Interpenetration

Jonh M. Snyder - Microsoft Corporation

Computer Graphics, 1995, pp. 209 - 218.

[6] Through-the-Lens Camera Control

Michael Gleicher and Andrew Witkin - School of Computer Science - Carnegie Mellon University

Computer Graphics, 1992, pp. 331 - 340.

[7] An Object-Oriented 3D Graphics Toolkit
Paul S. Strauss and Rick Carey
1992, pp. 341 - 347.

[8] An Architecture for an Extensible 3D Interface Toolkit
M. P. Stevens, R. C. Zeleznik and J. F. Hughes
Proceedings of UIST'94, ACM SIGGRAPH, 1994.

[9] Free-Form Deformation of Solid Geometric Models
Thomas W. Sederberg and Scott R. Parry
Computer Graphics, 1986, pp. 151 - 160.

[10] Direct Manipulation of Free-Form Deformations
William M Hsu, Jonh F. Hughes and Henry Kaufman
Computer Graphics, 1992, pp. 177 - 184.

[11] Axial Deformations: An Intuitive Deformation Technique
F. Lazarus, S. Coquillart and P. Jancene
Computer-Aided Design, 26(8):607-613, 1994.