Clausius Duque Gonçalves Reis

Animação em tempo real de rugas faciais explorando as modernas GPUs

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Engenharia de Computação.

Orientador: José Mario De Martino
Co-orientador: Harlen Costa Batagelo

Campinas, SP
2010

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

R277a

Reis, Clausius Duque Gonçalves
Animação em tempo real de rugas faciais explorando as modernas GPUs / Clausius Duque Gonçalves Reis. – Campinas, SP: [s.n.], 2010

Orientador: José Mario De Martino.
Co-orientador: Harlen Costa Batagelo.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Computação gráfica. 2. Animação por computador. 3. Expressão facial. 4. Sistemas de tempo real. 5. Programa de computador. I. De Martino, José Mario. II. Batagelo, Harlen Costa. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título

Título em Inglês:	Real time animation of facial wrinkles exploring the modern GPUs
Palavras-chave em Inglês:	Computer graphics, Computer animation, facial expression, Real time systems, Computer software
Área de concentração:	Engenharia de Computação
Titulação:	Mestre em Engenharia Elétrica e de Computação
Banca Examinadora:	Siome Klein Goldenstein, Léo Pini Magalhães
Data da defesa:	24/05/2010
Programa de Pós Graduação:	Engenharia Elétrica e de Computação

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Clausius Duque Gonçalves Reis

Data da Defesa: 24 de maio de 2010

Título da Tese: "Animação em Tempo Real de Rugas Faciais Explorando as Modernas GPUs"

Prof. Dr. José Mario De Martino (Presidente): _____

Prof. Dr. Siome Klein Goldenstein: _____

Prof. Dr. Léo Pini Magalhães: _____

Resumo

A modelagem e animação de rugas faciais têm sido tarefas desafiadoras, devido à variedade de conformações e sutilezas de detalhes que as rugas podem exibir. Neste trabalho, são descritos dois métodos de apresentação de rugas em tempo real, utilizando as modernas GPUs. Ambos os métodos são baseados no uso de *shaders* em GPU e em uma abordagem de *normal mapping* para aplicar rugas sobre modelos virtuais. O primeiro método utiliza áreas de influência descritas por mapas de textura para calcular a exibição de rugas sobre o modelo, controlados por um "Vetor de Ativação", que informa ao shader a visibilidade das rugas em cada uma das áreas de influência. O segundo método apresenta rugas nos modelos faciais, utilizando as informações de deslocamento dos vértices em direções pré-definidas, informadas através de um "Vetor Direção de Rugas", que informa o sentido que o deslocamento de um vértice causa o surgimento de rugas.

Palavras-chave: Computação gráfica, animação por computador, expressão facial, sistemas de tempo real, programa de computador.

Abstract

The modeling and animation of facial wrinkles have been challenging tasks, due to the variety of conformations and detail subtleness that the wrinkles can display. In this paper, we describe two methods to present wrinkles in real time, using modern GPUs. Both methods are based on the use of GPU shaders and a normal mapping approach to apply wrinkles on virtual models. The first method uses influence areas described by texture maps to calculate the display of wrinkles on the model, controlled by an "Activation Vector", which tells the shader the appearance of wrinkles in each area of influence. The second method presents wrinkles on facial models using the vertex displacement information in predetermined directions, informed by a "Wrinkles Direction Vector", informing the direction that the displacement of a vertex causes the presentation of wrinkles.

Keywords: Computer graphics, computer animation, facial expression, real time system, computer software.

Agradecimentos

Ao meu orientador e co-orientador, Profs. José Mario De Martino e Harlen Costa Batagelo, sou profundamente grato pela orientação.

Aos colegas Charles, Tiago Coser e Mábia pela ajuda e horas de consultoria.

Aos demais colegas de pós-graduação, pelas críticas e sugestões.

Aos meus pais e família pelo apoio diante desta e outras jornadas, sempre acreditando em minha capacidade.

Ao meu amor Rachel, por estar sempre ao meu lado e me incentivar mesmo nas horas difíceis. Pela sua compreensão e pelo carinho todo dengoso que sempre demonstrou. Você é a razão por que estou aqui hoje e é por você que estarei aqui amanhã! Te amo!

À CAPES, pelo apoio financeiro.

A minha esposa, meus pais, irmãos, avós e tios

Sumário

Lista de Figuras

Lista de Tabelas

Trabalhos publicados pelo autor

1. C. D. G. Reis, H. C. Batagelo and J. M. De Martino. "Real-time simulation of wrinkles". *16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision - WSCG2008, Communication Papers*, University of West Bohemia, Czech Republic, pages 109-115, 2008. ISBN 978-80-86943-16-9.

Capítulo 1

Animação em tempo real de rugas faciais explorando as modernas GPUs

1.1 Motivação e relevância

A animação facial por computador é uma área da computação gráfica que abrange técnicas para gerar e animar modelos faciais, sejam estes bidimensionais, provenientes de imagens estáticas ou trechos de filmes, ou tridimensionais, utilizando tipicamente malhas poligonais. Atualmente a animação facial por computador encontra aplicação na indústria de jogos para computador e de consoles de vídeo game, cinema, teleconferência, multimídia educacional, medicina, entre outros.

As expressões faciais ajudam os seres humanos a se comunicar desde os primeiros dias de vida, com semblantes de tristeza, alegria, medo, prazer, entre outros. Portanto, o uso de expressões se faz necessário também em modelos virtuais, como em jogos eletrônicos, onde os personagens precisam transmitir emoções juntamente com as falas, fazendo com que a imersão do jogador no enredo seja muito maior. O mesmo acontece com sistemas multimídia educacionais, onde os avatares necessitam expressar visualmente as frases por eles afirmadas, como por exemplo um elogio ou exclamação ao usuário onde as sobrancelhas são elevadas, ou até mesmo uma correção ou reclamação onde os olhos são comprimidos, dando a impressão de desagrado.

Em computação gráfica, mais especificamente na área de animação facial, foram produzidos trabalhos com as mais variadas abordagens. Alguns trabalhos apresentando soluções mais simples, como o de Frederick Parke em 1972, que utilizou fotometria com marcadores espalhados pela face para gerar uma malha 3D de cada expressão que era animada por interpolação linear (PARKE, 1972). Outros buscaram parametrizar a face criando assim formas de se controlar partes independentes da face, como por exemplo, o movimento dos olhos, mandíbula, sobrancelhas, posição dos lábios e cantos da boca (PARKE, 1982; BERGERON; LACHAPELLE, 1985).

Ainda outros autores utilizaram simulação física para apresentar melhores resultados, como por exemplo a utilização de músculos virtuais baseados em modelos físicos (WATERS, 1987) e modelos empíricos (MAGNENAT-THALMANN; PRIMEAU; THALMANN, 1988). Alguns trabalhos cons-

truíram modelos faciais com camadas de ossos, músculos, gordura e pele interligados por um sistema massa-mola (CHADWICK; HAUMANN; PARENT, 1989; SEDERBERG; PARRY, 1986; PLATT; BADLER, 1981; TERZOPOULOS; WATERS, 1990; CORRêA, 2007), outros utilizaram simulação de elementos finitos (LUCERO; MUNHALL, 1999; BATHE, 1982; DENG, 1988; PARKE; WATERS, 1996).

A utilização de rugas em animação facial é fundamental para se adicionar realismo e expressividade em modelos virtuais. Com sua aplicação, além do aumento no grau de realismo, é possível acrescentar outros detalhes nas expressões faciais que não seriam possíveis sem seu uso, como por exemplo, sensação de força nas expressões e a idéia de envelhecimento.

O foco principal deste trabalho, apresentação de rugas faciais em modelos virtuais, tem sido estudado e aplicado por diversos autores (OLIVEIRA; POLICARPO, 2005; NORDENBERG, 2003; WU; THALMANN; THALMANN, 1994; BANDO; KURATATE; NISHITA, 2002; WU; THALMANN; THALMANN, 1994; WU; THALMANN, 1996; WU et al., 1999; BOISSIEUX et al., 2000). Há diferentes tipos de rugas, algumas delas são formadas de acordo com a idade e condições do ambiente em que as pessoas vivem, outras, de acordo com a compressão da pele humana. O rosto humano realiza por volta de 15 mil contrações por dia (COMBAZ; NEYRET, 2002), podendo estas ser resultados de risos, choro, alegria, preocupações, stress, entre inúmeras outras causas. Com o movimento constante, a pele humana perde sua resistência e flexibilidade, fazendo com que ocorram a formação de vincos ou rugas de expressão.

Neste trabalho foram utilizadas placas gráficas modernas para apresentar rugas em tempo real, apresentando-as em modelos virtuais tridimensionais com o auxílio de programas executados internamente às *GPUs* (PIEPER; ROSEN; ZELTZER, 1992), denominados *shaders* [1].

1.2 Objetivo do trabalho

O objetivo deste trabalho é o desenvolvimento de técnicas para a apresentação de rugas em tempo real em faces virtuais tridimensionais, explorando as características de processamento e memória suportadas pelas modernas *GPUs*.

Pretende-se utilizar as capacidades de hardware programável das placas gráficas, efetuando os cálculos de interpolação e apresentação das rugas, internos à *GPU* com a utilização de *vertex* e *pixel shaders*.

1.3 Resultados principais

Foram desenvolvidas duas técnicas para a apresentação de rugas em tempo real em faces virtuais. A primeira baseia-se em áreas de influência, que estabelecem as áreas onde as rugas devem aparecer na face, levando em conta as coordenadas de textura do modelo geométrico utilizado. Estas áreas

[1] Shaders: http://en.wikipedia.org/wiki/Shader

de influência são acomodadas em texturas 3D para facilitar o envio ao *shader*. A textura 3D, internamente ao *shader*, é percorrida pixel a pixel na etapa de *pixel shader* e multiplicada por um fator especificado em uma das posições de um Vetor de Ativação das áreas. Este fator define a porcentagem de apresentação das rugas através da interpolação entre valores especificados em dois mapas de normais. Um dos mapas descreve a face em seu estado relaxado, sendo que o outro define a configuração da face em uma postura extrema. A postura extrema, como o próprio nome sugere, representa uma situação extrema com a face exibindo todas as rugas passíveis de serem apresentadas. Entre as características dessa solução destaca-se a facilidade de gerar as áreas de influência necessárias para o funcionamento, facilmente trabalhadas por programas de edição de imagens.

A segunda técnica desenvolvida faz uso de "Vetores Direção de Rugas", que são vetores definidos pelo usuário com a finalidade de informar ao *shader* qual o sentido em que o deslocamento dos vértices irá resultar na apresentação de rugas. Para tanto, a técnica também utiliza informações do deslocamento de cada vértice em relação a uma posição inicial, com a face em repouso, assim como o ângulo formado entre esse vetor de deslocamento e o Vetor Direção de Rugas associado a cada vértice. Com essas informações é possível calcular os locais e intensidade com que serão apresentadas as rugas de acordo com as expressões faciais exibidas. Uma vantagem dessa técnica é que ela pode ser facilmente incorporada a diferentes abordagens de manipulação da geometria facial, como por exemplo sistemas massa-mola ou envolvendo simulação biomecânica, não sendo necessário alterar a lógica interna das mesmas já que os cálculos necessários para a exibição das rugas são realizados internamente ao *shader*. A única obrigatoriedade é o envio das informações extras à placa gráfica, como os Vetores Direção de Rugas e a posição original dos vértices.

1.4 Organização do trabalho

No Capítulo 2, são apresentados e discutidos os principais trabalhos encontrados na literatura voltados à modelagem e apresentação de rugas em faces virtuais, abordando as técnicas apresentadas, suas aplicações e utilização em sistemas em tempo real. Procurou-se passar pelos primórdios da animação facial, o uso da parametrização das faces, músculos virtuais e conceitos de simulação aplicados na área, como modelos massa-mola e de elementos finitos. Desta forma apresentando a evolução das técnicas de animação facial por computador e de modelagem e apresentação de rugas faciais. Por fim, são analisados os principais trabalhos envolvendo rugas faciais e situado o presente trabalho em relação às outras abordagens.

No Capítulo 3 é descrita a arquitetura das placas gráficas programáveis, o que são *shaders*, sua utilidade na melhoria da velocidade da aplicação e resultados finais de produção de imagens. São apresentados alguns *shaders* para situar a evolução de algumas técnicas de detalhamento de imagens, como o *bump mapping* (BLINN, 1978), *displacement mapping* (COOK, 1984) e *normal mapping* (KILGARD, 2000).

Um modelo conceitual das soluções é apresentado no Capítulo 4, onde são expostas as técnicas desenvolvidas neste trabalho e o motivo do uso de cada uma delas ao invés de outras existentes, e é descrito o uso de texturas 3D como áreas de influência e o conceito de "Vetores Direção de rugas". Também neste capítulo, são descritos os requisitos do sistema, suas etapas de pré-processamento,

entrada dos dados, e os cálculos realizados em cada quadro da animação. São apresentados os sistemas desenvolvidos para validar ambas as propostas descritas nesta dissertação, onde são definidas as linguagens e ferramentas utilizadas nas implementações e a motivação de suas escolhas.

Os resultados obtidos são comentados no Capítulo 5, juntamente com a descrição do equipamento utilizado para a obtenção dos mesmos.

Finalmente no Capítulo 6 é concluído o trabalho desenvolvido apresentando um resumo das técnicas propostas, assim como as vantagens e desvantagens de cada uma, e sugestões de melhorias que deverão ser levadas em consideração em futuros trabalhos.

Capítulo 2

Estado da técnica

Neste capítulo é feita uma revisão sobre os trabalhos realizados em animação facial na computação gráfica, sobretudo no tópico "rugas", objetivo principal desta dissertação.

Serão apresentados os principais trabalhos em animação facial, passando pelos pioneiros na área (Seção 2.1) até os mais atuais. Serão descritos abordagens utilizadas para o cálculo de rugas em modelos virtuais, assim como os principais trabalhos encontrados cada uma delas (seções 2.1 a 2.5), sendo realizado um comparativo entre as mesmas e as técnicas desenvolvidas nesta dissertação.

Na Seção 2.6 é feita uma classificação dos trabalhos analizados, de acordo com a abordagem utilizada.

Por fim, serão feitas as considerações finais a respeito do que foi discutido no capítulo.

2.1 Trabalho pioneiro em animação facial tridimensional

Frederick Parke, considerado pioneiro na área de animação facial, publicou seus primeiros trabalhos em 1972. Seu trabalho (PARKE, 1972), consistiu em modelar um rosto virtual humano utilizando técnicas fotogramétricas. Em uma face real foram posicionados diversos pontos de referência (Figura 2.1, A e B), também chamados de marcadores, e fotografados em diversos ângulos. Este mesmo procedimento foi repetido para cada uma das expressões faciais desejadas. De posse das fotografias foram medidas as distâncias entre os pontos de referência para a obtenção de malhas tridimensionais com as expressões fotografadas (Figura 2.1, C e D).

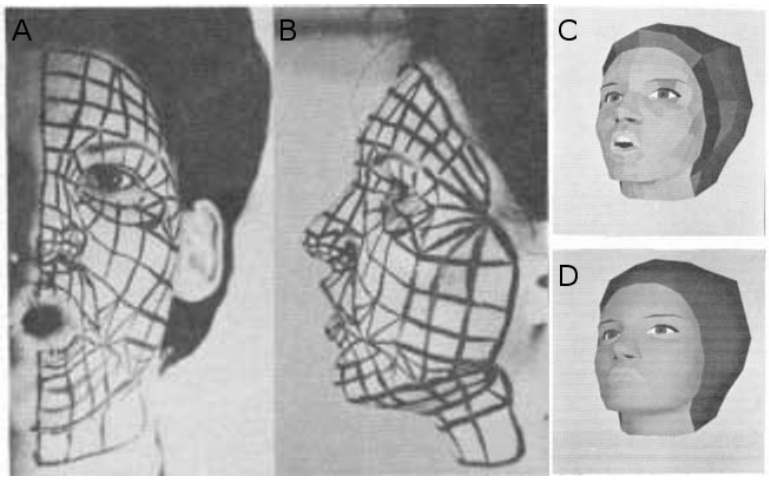

Fig. 2.1: Aquisição por fotogrametria da malha facial 3D (Adaptado de (PARKE, 1972)).

Este modelo facial foi animado com um esquema de interpolação por cosseno entre as poses-chave, que utilizava o algoritimo da figura 2.2 para mover os vértices do modelo geométrico das posições de uma expressão para as posições de outra.

```
      posição atual = posição na fase anterior + C * diferença

Onde:
      diferença = posição da próxima fase -
                  posição da fase anterior

      C = (1.0 - cos(ϕ)) / 2.0

e
      ϕ = fração de cada fase * 3,14159
```

Fig. 2.2: Algoritmo de interpolação utilizado por Parke (Adaptado de (PARKE, 1972)).

2.2 Modelos paramétricos

Com o objetivo de se obter modelos virtuais mais complexos e expressivos, surgiu a necessidade de se aprimorar a técnica para animar as faces virtuais. Como resultado dessa necessidade, foram desenvolvidos os modelos paramétricos. A vantagem da modelagem paramétrica é que não é necessário que o usuário possua um modelo geométrico para cada uma das expressões, como era normalmente feito. Nos modelos paramétricos, o rosto é dividido em áreas controladas de forma independente umas das outras. O usuário define parâmetros para cada área, gerando assim as expressões faciais desejadas na animação.

H] 

Fig. 2.3: Imagens da animação *Tony de Peltrie* (Adaptado de (BERGERON; LACHAPELLE, 1985)).

Em seu artigo (PARKE, 1982), Parke determinou que para se criar uma expressão eram necessários parâmetros para o movimento das pálpebras ao piscar os olhos, movimento de rotação das sobrancelhas, movimento vertical das sobrancelhas, rotação das mandíbulas, largura da boca, expressões da boca, posicionamento do lábio superior, posicionamento dos cantos da boca e movimento das pupilas dos olhos.

O trabalho de Parke foi aperfeiçoado em 1985, quando na animação *Tony de Peltrie* (BERGERON; LACHAPELLE, 1985) (Figura 2.3), os personagens, um pianista e várias faces virtuais, necessitaram passar seus sentimentos e emoções apenas com expressões faciais. Para que isso fosse possível fez-se uso de uma técnica chamada de soma de vetores, onde os personagens podiam além de utilizar expressões pré-definidas, criar novas expressões combinando as já existentes, definidas para uma única face, apenas somando ou subtraindo os vetores de distância entre os vértices.

2.3 Início do uso de músculos virtuais

Pode ser dito que o primeiro trabalho desenvolvido relacionado a músculos virtuais foi o de Platt e Badler (1981), no qual os autores apresentaram um modelo de fibra muscular utilizado para simular o comportamento da pele durante a movimentação dos músculos. Neste modelo, vértices da representação poligonal da face era conectado a uma determinada fibra muscular, que por sua vez recebia ação de uma força externa, realizando assim o deslocamento dos vértices. As fibras eram espalhadas pela superfície do modelo geométrico, e diferentes valores de força eram aplicados a cada uma delas, formando assim as expressões faciais.

Em 1987 Waters utilizou o conceito de músculos virtuais em seu trabalho (WATERS, 1987). Basean-do-se em modelos reais de músculos faciais, Waters propôs uma forma de simular a ação de músculos sobre o modelo geométrico facial (Figura 2.4 e 2.5). Foram distribuídos pelo modelo facial músculos virtuais lineares, utilizados para tração dos vertices, e músculos virtuais do tipo esfíncter, utilizados para compressão da malha em uma direção central ao músculo, como por exemplo o músculo ao redor dos olhos. A ação desses dois tipos de músculos aplicados sobre o modelo acarretavam a formação das expressões faciais (Figura 2.6).

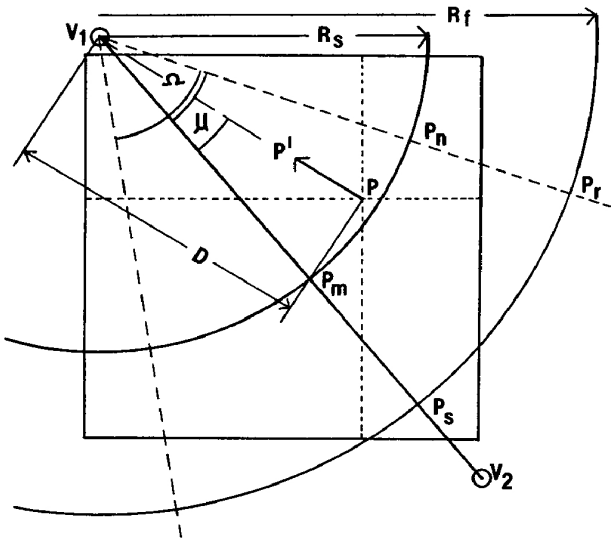

Fig. 2.4: Modelagem da região de influência do músculo virtual de Waters (Adaptado de Waters (1987)).

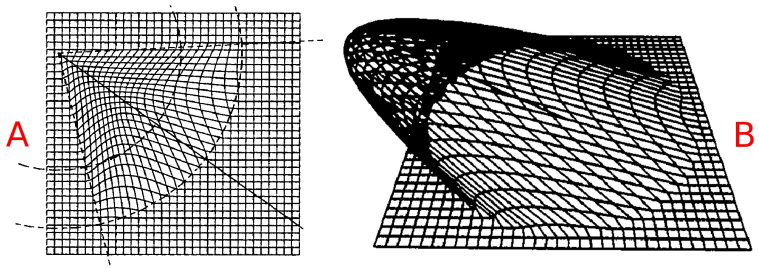

Fig. 2.5: Ação do músculo virtual de Waters sobre uma malha bidimensional (A) e sobre uma malha tridimensional (B) com um decaimento circular por cosseno (Adaptado de Waters (1987)).

Outra abordagem foi proposta no ano seguinte quando, no trabalho de Magnenat-Thalmann, Primeau e Thalmann (1988), foi desenvolvido um modelo de simulação denominado de “pseudomúsculos”, já que não se tratava de uma implementação de modelos musculares baseados rigorosamente na realidade. A abordagem apresentava um modelo empírico no qual para a formação das expressões era utilizado um método denominado ”Ação Abstrata de Músculos” (*Abstract Muscle Action*), que define expressões compostas de pequenas ações. Para cada ação, um movimento de contração ou relaxamento dos músculos deveria ser especificado.

Fig. 2.6: Expressões geradas pela ação dos músculos (Adaptado de Waters (1987)).

2.4 Modelos massa-mola

Nos modelos massa-mola, os vértices assumem o papel de pontos de massa conectados por um conjunto de molas ou elásticos. Todo o conjunto permanece em equilíbrio até que alguma força seja aplicada sobre os pontos de massa, quando um conjunto de equações diferenciais controla o movimento das massas para que o conjunto retorne ao equilíbrio.

A utilização desse tipo de sistema tornou-se muito comum em computação gráfica, sobretudo em simulação de tecidos, como por exemplo Platt e Badler (1981), em que foi proposto um tecido formado por uma malha de molas sem espessura. Essa malha era conectada por molas adicionais entre alguns vértices e os músculos abaixo da pele. O força do movimento dos músculos nesse sistema era propagada para os demais pontos conectados a ele na malha da face.

O trabalho de Waters (1987) sobre músculos virtuais, foi desenvolvido utilizando-se um modelo massa-mola para conectar os vértices da face geométrica. Isto proporcionou uma melhor visualização da ação dos músculos sobre a face, simulando a reação que ocorre nas áreas adjascentes ao deslocamento, assim como ocorre na face humana.

Terzopoulos e Waters (1990) apresentaram um modelo de três camadas de pele, que consistia de tecido cutâneo, tecido adiposo e uma camada de músculos. Sua abordagem conectava as massas à molas com constantes de espessuras variadas, modelando assim as propriedades elásticas das camadas de pele.

Baseando-se no trabalho de Terzopoulos e Waters (1990), Lucero e Munhall (1999) substituíram os coeficientes das molas e parâmetros dos músculos por valores mais aprimorados, utilizando conhecimentos sobre as propriedades biomecânicas da pele e músculos. A dissertação de mestrado de Corrêa (2007) também explorou a modelagem proposta por Terzopoulos e Waters (1990).

2.5 Análise de Elementos Finitos

Tradicionalmente utilizados em análise estrutural de materiais em engenharia, métodos utilizando elementos finitos (BATHE, 1982) têm se tornado bastante comum em aplicações de computação gráfica. Se comparado com sistemas massa-mola, abordagens utilizando elementos finitos são mais estáveis, mais precisas, porém são consequentemente mais caras do ponto de vista de processamento.

Um modelo de elementos finitos de três camadas visando o tecido facial foi idealizado por Deng (1988) onde a camada externa da pele era conectada por uma subcamada móvel, que por vez era conectada à camada de músculos logo abaixo.

Pieper, Rosen e Zeltzer (1992), por sua vez, utilizaram dados escaneados para planejamento de cirurgia plástica, onde a partir de linhas de incisão desenhadas diretamente no objeto geométrico, era gerada uma malha de elementos finitos para a simulação do corte.

2.6 Rugas faciais em computação gráfica

Técnicas para modelagem e apresentação de rugas são importantes em vários tipos de aplicações, como por exemplo, cinema, avatares virtuais e jogos. Seu uso contribui para aumentar o nível de realismo e expressividade em modelos faciais virtuais, principalmente em situações onde é necessário apresentar emoções na expressão dos personagens.

Os trabalhos envolvendo rugas em computação gráfica podem ser classificados de acordo com sua abordagem, por exemplo, trabalhos onde é utilizada simulação física para calcular a aparência e posicionamento das rugas sobre a superfície do modelo facial, ou então trabalhos onde apenas o efeito visual das rugas é reproduzido, por meio de texturas ou técnicas de tonalização, de acordo com a animação facial realizada. No caso das abordagens utilizando texturas, vale lembrar que a reprodução das rugas pode ser feita de forma estática por texturas, ou mesmo procedural, com equações para o cálculo de sua aparência.

Outro aspecto importante que deve ser lembrado é a separação que pode ser realizada entre a forma como as rugas são aplicadas sobre o modelo virtual. Algumas técnicas trabalham com técnicas utilizando texturas para representar as rugas sobre a face do modelo, no entanto, a representação da silhueta desse modelo não será feita corretamente, como por exemplo, rugas da testa sendo vistas de lado não são representadas. No entanto existem técnicas que alteram diretamente a geometria, aumentando ou reduzindo a densidade de polígonos dos modelos finais para apresentar rugas com silhuetas corretamente, que dependendo do número de polígonos formados e equipamento utilizado, pode reduzir a performance do sistema.

Entre as abordagens utilizando texturas são encontradas estratégias que utilizam técnicas de tonalização, ou *shading*, que utilizam recursos de *hardware* modernos para apresentar em modelos de baixa densidade de polígonos, detalhes que só poderiam ser obtidos em modelos com alta densidade de polígonos. O uso desse tipo de técnica ao invés da alteração da geometria tende a aumentar consideravelmente a performance do sistema, porém na maioria das vezes a apresentação de silhuetas nos objetos é deixada de lado em prol desse ganho de velocidade.

A Tabela 2.1 divide os artigos apresentados nesta seção em quatro grupos: Trabalhos utilizando simulação física ou texturas para representar o efeito visual das rugas, ambos subdivididos em trabalhos utilizando alteração da geometria e trabalhos sem a alteração da geometria.

	Simulação	**Reprodução do efeito visual**
Alteração da geometria	**Grupo 1:** (VOLINO; MAGNENAT-THALMANN, 1999)	**Grupo 3:** (WANG; WANG; YUEN, 2006) (LARBOULETTE; CANI, 2004)
Sem alteração da geometria	**Grupo 2:** (BOISSIEUX et al., 2000) (MAGNENAT-THALMANN et al., 2002) (PASQUARIELLO; PELACHAUD, 2001) (WU; THALMANN; THALMANN, 1994)	**Grupo 4:** (BANDO; KURATATE; NISHITA, 2002) (BOISSIEUX et al., 2000) (VIAUD; YAHIA, 1992) (BICKEL et al., 2007) (NORDENBERG, 2003) (WU; THALMANN, 1996) (TU et al., 2003)

Tab. 2.1: Classificação dos trabalhos apresentados nesta seção.

2.6.1 Grupo 1: Rugas faciais com simulação e alteração da geometria

Existem diversas abordagens para se modelar ou apresentar rugas, algums autores optam por apresentar soluções mais voltadas para o lado da simulação física (VOLINO; MAGNENAT-THALMANN, 1999; BOISSIEUX et al., 2000; MAGNENAT-THALMANN et al., 2002; PASQUARIELLO; PELACHAUD, 2001), a fim de aproximar ao máximo o comportamento e a aparência final dos modelos virtuais à dos modelos reais. Neste grupo (1), são descritos os trabalhos que trabalham com simulação e alterações na geometria (VOLINO; MAGNENAT-THALMANN, 1999) dos modelos virtuais durante a execução, para a correta apresentação das rugas no resultado final.

Adota-se nesta dissertação que a alteração da geometria é a modificação do número de polígonos do modelo facial durante a execução do sistema.

Em Volino e Magnenat-Thalmann (1999), o autor apresenta um sistema baseado na compressão e alongamento da superfície dos triângulos de modelos geométricos, e de uma função constante para o padrão visual das rugas, eram encontrados os índices de amplitude evolutiva para cada vértice a fim de simular o resultado final por expressões matemáticas simplificadas. Foram apresentadas duas implementações do método proposto, uma voltada para a simulação de modelos têxteis, utilizando displacement mapping para aprimorar os detalhes visuais em sistemas de simulação de tecidos (Figura

2.7). A outra implementação apresentada, utilizando *bump mapping*, era direcionada à apresentação de rugas em faces virtuais animadas (Figura 2.8), além de possibilitar a simulação de envelhecimento, aumentando o comprimento em estado de repouso da malha em determinados pontos do modelo facial.

Fig. 2.7: Rugas aplicadas sobre uma simulação de tecidos através de *displacement mapping* (Adaptado de (VOLINO; MAGNENAT-THALMANN, 1999)).

Fig. 2.8: Rugas aplicadas sobre uma face virtual através de *bump mapping* (Adaptado de (VOLINO; MAGNENAT-THALMANN, 1999)).

2.6.2 Grupo 2: Rugas faciais com simulação sem alteração da geometria

Seguindo o mesmo modelo de trabalho, utilizando simulação, outros autores (BOISSIEUX et al., 2000; MAGNENAT-THALMANN et al., 2002; PASQUARIELLO; PELACHAUD, 2001) optaram por realizar a apresentação de rugas sobre modelos faciais, sem a alteração da geometria para alcançar esse objetivo.

Os artigos Boissieux et al. (2000) e Magnenat-Thalmann et al. (2002), focam na simulação do tecido da pele, visando tanto o aspecto visual quanto as propriedades biomecânicas da mesma. Em Boissieux et al. (2000) são apresentadas duas abordagens para a simulação de rugas em faces virtuais, apenas a segunda abordagem, explorada em maior detalhes no trabalho Magnenat-Thalmann et al. (2002), se encaixa nesse grupo (2), pois utiliza simulações físicas para os cálculos de apresentação de rugas. A abordagem proposta em ambos os trabalhos, objetiva a simulação de rugas em um modelo de tecido abstrato bidimensional, representando a pele humana. Este modelo abstrato é composto de várias camadas, com propriedades como viscosidade, elasticidade e preservação de volume diferentes para cada camada. O conjunto desas propriedades contribuem para o estado de equilíbrio da pele. O comportamento do tecido é controlado por deformações elásticas, onde cada triângulo da malha poligonal é considerado como um material elástico, linear e isotrópico, além de possuir uma memória que gradualmente era substituída durante a compressão do tecido, gerando eventualmente, depois de várias compressões, rugas de expressão (Figura 2.9).

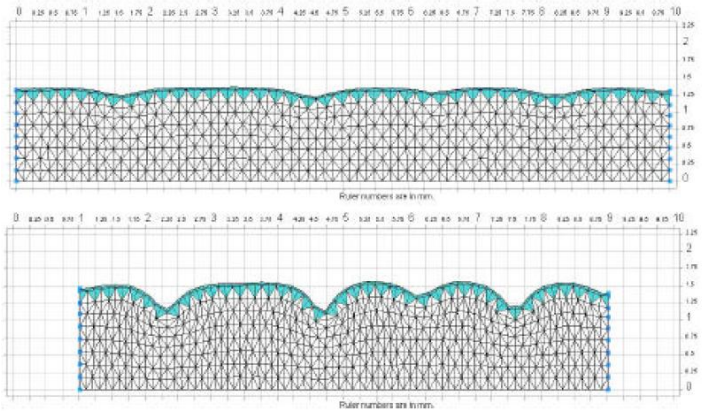

Fig. 2.9: Pedaço abstrato de tecido utilizado para gerar as rugas. Na parte superior é possível ver o tecido com as rugas obtidas através da memória de compressão, já na parte inferior, pode-se visualizar o mesmo tecido sendo comprimido (Adaptado de (BOISSIEUX et al., 2000)).

No trabalho "GRETA" (PASQUARIELLO; PELACHAUD, 2001), além do sistema de animação facial, que utilizava pseudomúsculos para animar uma face virtual feminina, causando o aparecimento de rugas nas regiões de maior acúmulo de tecido de pele, como por exemplo as rugas naso-labiais, formadas quando sorrimos, um sistema de apresentação de rugas baseado utilizando *bump mapping* foi desenvolvido, especificamente voltado para a renderização de detalhes na testa de GRETA. Neste sistema, de acordo com a compressão e deslocamento dos vértices da testa do modelo, linhas horizontais eram formadas por meio de texturas dinâmicas e aplicadas no modelo, simulando a aparência de acúmulo de tecido cutâneo (Figura 2.10).

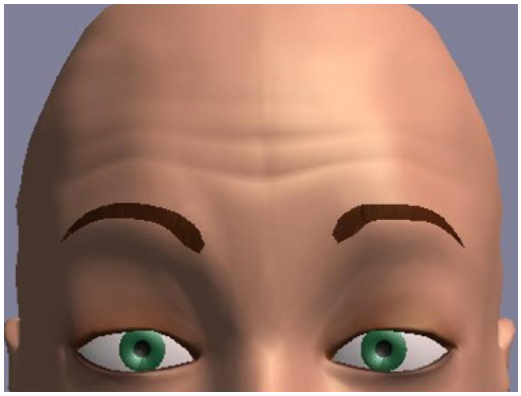

Fig. 2.10: Geração de rugas na testa de GRETA (Adaptado de (PASQUARIELLO; PELACHAUD, 2001)).

Uma proposta de simulação de rugas através de uma superfície multicamadas é apresentada em (WU; THALMANN; THALMANN, 1994), onde foi proposto um tecido com pele, tecido gorduroso e músculos (Figura 2.11). Na implementação, as camadas de músculos são construídas de acordo com os modelos anatômicos reais que caracterizam o movimento facial, e o tecido gorduroso entre a pele e os músculos é simulado como restrições de molas de Hooke agindo entre pontos. Com o processo elástico, a superfície da pele é deformada com as contrações musculares, produzindo as rugas de expressão.

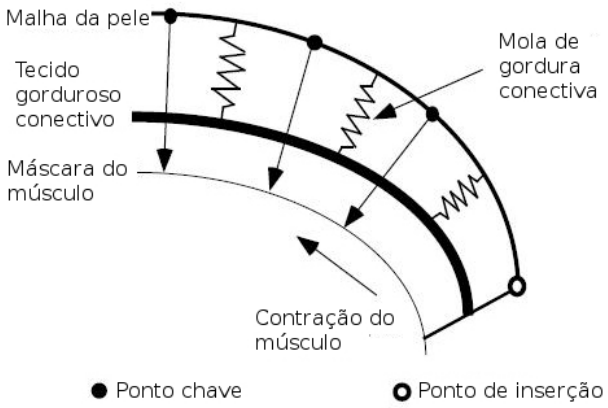

Fig. 2.11: Estrutura facial simplificada composta por pele, tecido gorduroso e músculos (Adaptado de (WU; THALMANN; THALMANN, 1994)).

2.6.3 Grupo 3: Rugas faciais por reprodução do efeito visual e alteração da geometria

Em Wang, Wang e Yuen (2006) os autores apresentam uma abordagem para apresentar rugas geometricamente em modelos não rígidos, utilizando para isso uma técnica de direcionamento por

curvas. Na implementaçao deste trabalho, as rugas da supefície são geradas por deformação da malha tridimensional em tempo de execução, de acordo com as alterações de uma curva governante posicionada na superfície do modelo. A curva em questão é capaz de imitar o comportamento de diferentes materiais, sendo necessário alterar algumas propriedades internas às funções da mesma. A propagação da amplitude da curva sobre a malha geométrica do modelo é realizada de forma paralela, limitada por uma área de influência (Figura 2.12).

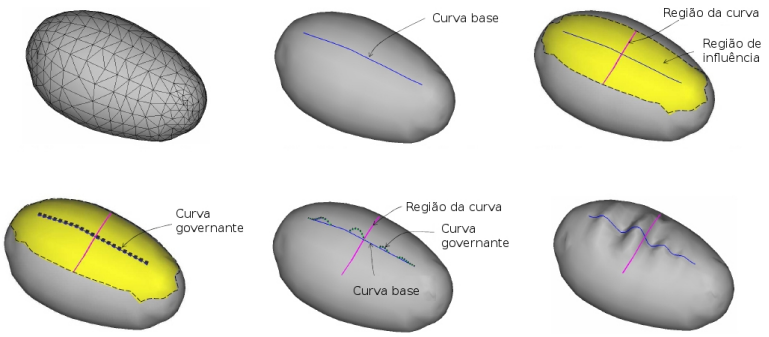

Fig. 2.12: Curva de controle com área de influência aplicado sobre um modelo geométrico não rígido (Adaptado de (WANG; WANG; YUEN, 2006)).

Durante o processo, a malha poligonal trabalhada é constantemente subdividida para que o relevo das rugas sejam apresentados de forma mais detalhada. (Figura 2.13)

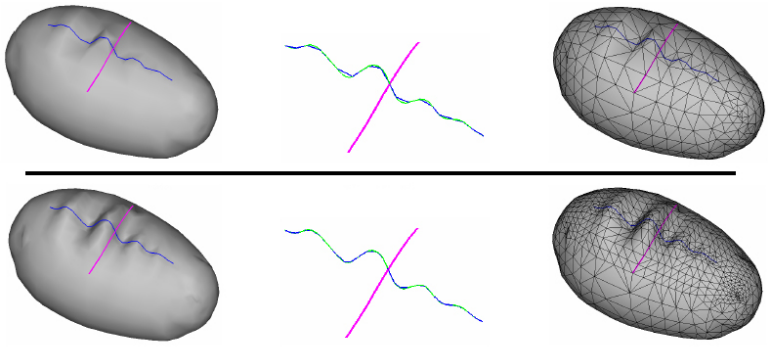

Fig. 2.13: Subdivisão da malha poligonal aplicada sobre o modelo geométrico não rígido (Adaptado de (WANG; WANG; YUEN, 2006)).

Outro método interessante para a apresentação de rugas geométricas de forma procedural foi introduzido em Larboulette e Cani (2004), onde os autores apresentaram uma forma de se imitar o comportamento de tecidos de acordo com sua compressão.

No método apresentado, o aspecto visual das rugas é animado a partir de uma curva de controle discreta bidimensional, de comprimento constante (Figura 2.14). Na definição desta curva são determinados o posicionamento de dois pontos extremos, um ponto de origem e um de destino, além de

n pontos de controle. Durante a animação, enquanto os pontos das extremidades se aproximam, o posicionamento dos pontos de controle é recalculado para que o comprimento da curva permanecesse constante, re-inserindo o comprimento perdido na forma de deslocamento ao longo do eixo *Y*.

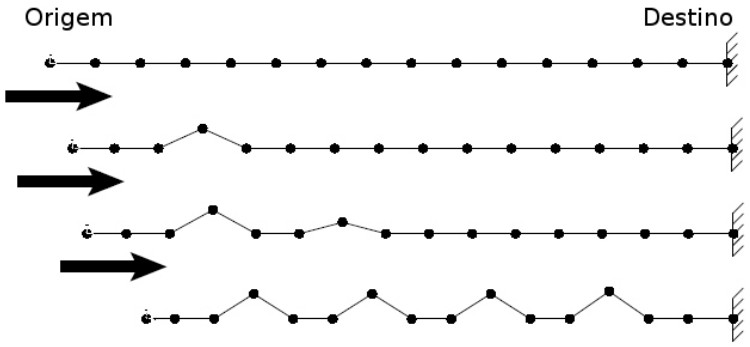

Fig. 2.14: Curva de controle utilizada para determinar a aparência do relevo das rugas (Adaptado de (LARBOULETTE; CANI, 2004)).

A apresentação das rugas na superfície do modelo é realizada utilizando-se a curva de controle, previamente descrita, juntamente com uma área de influência retangular utilizada para determinar a região afetada (Figura 2.15).

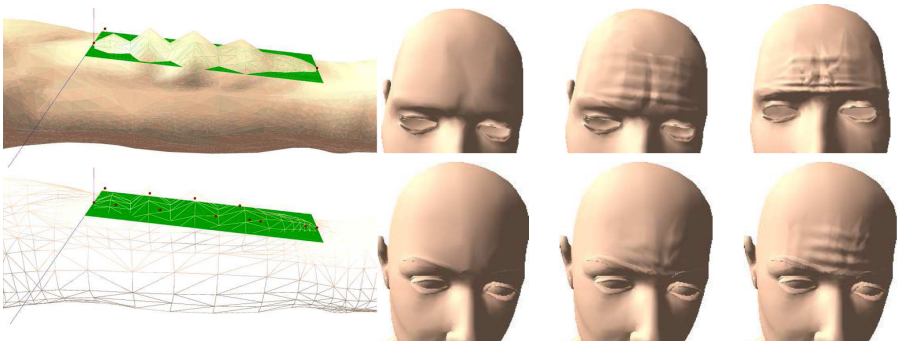

Fig. 2.15: Área de controle utilizada em associação com a curva de controle, e faces geradas utilizando a técnica (Adaptado de (LARBOULETTE; CANI, 2004)).

Durante todo o processo de animação das rugas, é realizada a subdivisão da geometria do objeto nas partes abrangidas pelas áreas de influência, a fim de melhor representar as rugas adicionadas.

2.6.4 Grupo 4: Rugas faciais por reprodução do efeito visual sem alteração da geometria

Entre um dos principais trabalhos já desenvolvidos, pode-se citar Bando, Kuratate e Nishita (2002), onde os autores propõem uma interface em que o usuário através de uma projeção bidi-

mensional da malha 3D, define curvas de *Bézier* nos pontos onde serão criados os sulcos das rugas (Figura 2.16). Estas curvas são utilizadas para se obter um mapa de deslocamento (*displacement map*) (COOK, 1984), utilizado para alterar o posicionamento dos vértices do modelo. No entanto, dependendo do refinamento da malha tridimensional do modelo utilizado, essa técnica necessita de um pré-processamento, a fim de adicionar polígonos nas áreas mais críticas. Esta etapa, dependendo das configurações definidas, tende a ser cara do ponto de vista de processamento computacional. No entanto, esse refinamento da malha não é realizado durante a execução do sistema.

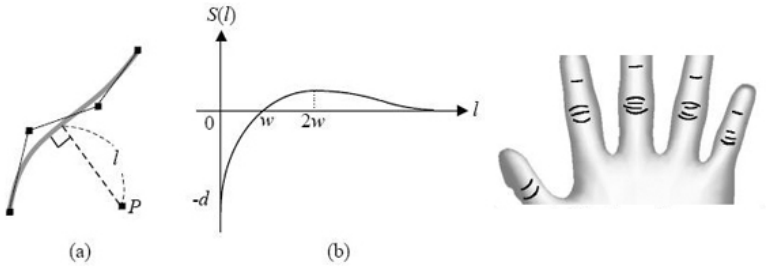

Fig. 2.16: Curvas de Bézier para criação de rugas em pontos específicos (Adaptado de (BANDO; KURATATE; NISHITA, 2002)).

O trabalho Boissieux et al. (2000) é citado também neste grupo (4) pois o mesmo apresenta duas abordagens para a apresentação de rugas em modelos virtuais. A primeira abordagem trabalha com oito texturas contendo máscaras genéricas de rugas, representando o relevo das rugas sobre o modelo geométrico, e o *shader* de *bump mapping*, a fim de apresentar detalhes sobre a superfície dos modelos em sistemas de animação facial. Estas máscaras são interpoladas para se apresentar em tempo real as rugas no modelo facial, imitando inclusive o envelhecimento da pele. Ao contrário da segunda aborgadem proposta em seu trabalho (BOISSIEUX et al., 2000), esta, utilizando texturas, não trabalha com simulação física das propriedades das rugas.

Em Viaud e Yahia (1992), é apresentada uma abordagem para se modelar rugas sobre uma face virtual, mapeadas por segmentos *C-Spline* e pontos de controle espalhados sobre a malha do modelo. Para a definição das rugas são desenvolvidas ferramentas para facilitar o processo. Neste trabalho, os autores ainda levam em consideração a influência que o avanço da idade tem sobre a aparência das rugas, adicionando esta variável à apresentação das mesmas.

No trabalho (BICKEL et al., 2007), é apresentado um método para a aquisição de animação facial e rugas em geometria de alta resolução, obtidas a partir de *scanners 3D* e arranjos de câmeras. Estas últimas utilizadas para aquisição dos detalhes faciais, como textura e rugas. No artigo apresentado, são escaneadas faces reais na melhor qualidade possível a fim de obter modelos geométricos estáticos de alta resolução, posteriormente suavisados para apresentarem uma superfície neutra, livre de quaisquer detalhes faciais como por exemplo, rugas adquiridas pela idade. Todo o processo de animação é obtido por *motion capture*, utilizando marcadores azuis previamente colocados nas faces dos atores (Figura 2.17). As rugas de expressão são obtidas pelo mesmo processo dos marcadores, porém são utilizadas as cores verde, amarelo e vermelho para sua determinação.

Fig. 2.17: Pontos de controle e marcadores espalhados pela face de um ator para coleta das informações faciais e geração do modelo tridimensional (Adaptado de (BICKEL et al., 2007)).

Os dados de rugas e animação facial são armazenados em um modelo paramétrico bibimensional. Combinação que é capaz de representar a aparência facial e seus detalhes em múltiplas escalas.

De acordo com os dados provenientes do processo de *motion capture*, a malha da face é interpolada a fim de se obter uma animação facial mais suave. Nesse processo são adicionadas as rugas obtidas no pré-processamento diretamente nos vértices do modelo, na forma de relevo (Figura 2.18).

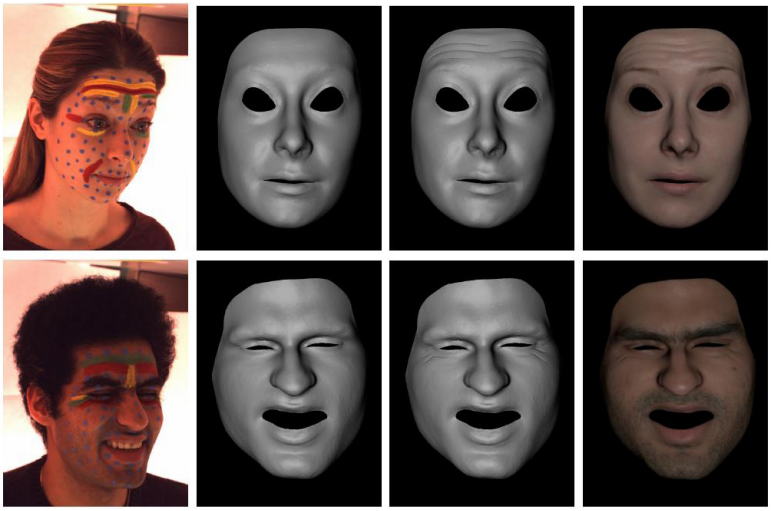

Fig. 2.18: Processo de utilização dos detalhes faciais obtidos em dois atores. A imagem mostra uma malha facial para cada modelo, sem rugas, com rugas e com e textura e rugas adicionadas (Adaptado de (BICKEL et al., 2007)).

Em sua tese (NORDENBERG, 2003), Mikael Nordenberg sugere uma forma de se adicionar rugas em tempo real à faces virtuais. Para isso ele propôe um sistema em duas partes, primeiramente é necessário calcular a compressão da malha facial quando esta é deformada, o que é obtido a partir de cálculos geométricos utilizando os triângulos do modelo. Após estes valores serem obtidos, a compressão é convertida na intensidade das rugas e apresentada em tempo real com o auxílio de placas gráficas, utilizando a técnica de *bump mapping* na renderização final (Figura 2.19).

Fig. 2.19: Rugas geradas por compressão dos polígonos da malha do modelo facial através de *bump mapping* (Adaptado de (NORDENBERG, 2003)).

No artigo Wu e Thalmann (1996), os autores apresentam uma abordagem para se gerar rugas tanto estáticas quanto dinâmicas. Para o modelo estático, foi desenvolvido um sistema utilizando triangulação planar de Delaunay, capaz de gerar a textura da pele (Figura 2.20). Sendo que as dobras visíveis na textura são definidas por arestas existentes na malha do objeto, que funcionam como uma restrição para o processo de triangulação hierárquica. Já para o modelo dinâmico de ruga foi implementado uma abordagem utilizando um modelo biomecânico da pele, fazendo com que rugas fossem expressadas de acordo com a animação facial ocorrida (Figura 2.21).

Fig. 2.20: Macro e micro estruturas da pele geradas por triangulação de Delaunay (Adaptado de (WU; THALMANN, 1996)).

Fig. 2.21: Rugas apresentadas por modelo biomecânico da pele (Adaptado de (WU; THALMANN, 1996)).

Uma proposta utilizando captura de animação facial e detalhes faciais como rugas, diretamente de vídeos foi apresentada em Tu et al. (2003), onde é descrito um sistema capaz de trabalhar tanto com animação bidimensional de faces quanto com animação tridimensional. Para isso, são capturadas faces com marcadores posicionados em locais chave (Figura 2.22), que servem de guia para que os dados obtidos possam ser aplicados em outros modelos faciais, além do original.

Fig. 2.22: Marcadores posicionados para captura de animação e detalhes faciais. Ao todo foram espalhados nessa imagem 49 marcadores (Adaptado de (TU et al., 2003)).

Os detalhes faciais são armazenados na forma de imagens médias (*Ratio images*), que são calculadas dividindo-se as cores de uma face realizando alguma expressão, pelas cores da face em estado expressivo neutro. No caso de animações faciais bidimensionais, o mapa obtido através desse cálculo é utilizado normalmente, aplicando-se o mesmo sobre o modelo, sem qualquer alteração, no entanto, no caso de modelos tridimensionais, é necessário realizar uma conversão desse mapa de imagens médias em mapas de normais, que são aplicados sobre o modelo 3D com a técnica de *bump mapping*, resultando em um nível de detalhes maior do que o original, sem a necessidade de subdivisão da malha poligonal (Figura 2.23).

Fig. 2.23: Aplicação dos detalhes faciais armazenados na forma de imagens médias (*Ratio images*) através de *bump mapping* (Adaptado de (TU et al., 2003)).

2.7 Considerações finais

Como foi apresentado neste capítulo, existem diversas soluções já propostas para o problema de apresentação e/ou simulação de rugas em modelos virtuais. Entre estas pode-se encontrar diversas abordagens como o uso de mapas de normais para apresentação das rugas, simulação física das propriedades da pele, rugas em textura e rugas modeladas diretamente na geometria dos modelos virtuais.

A escolha da abordagem a ser utilizada depende de vários fatores que em geral podem ser relacionados à memória disponível, capacidade de processamento, *hardware*, entre outros. No caso dos trabalhos apresentados nesta dissertação, os focos principais foram velocidade, visando a apresentação das rugas em tempo real, e portabilidade, visando a capacidade de se utilizar as técnicas apresentadas em outros sistemas de animaçção facial.

Os trabalhos desenvolvidos nesta dissertação podem ser classificados como sendo do grupo 4, ou seja, realizam a reprodução do efeito visual das rugas sem a necesidade de subdividir a malha poligonal do modelo facial durante a execução do sistema.

Capítulo 3

Shaders para mapeamento de detalhes 3D

Na área de computação gráfica, os *shaders* são utilizados na programação das unidades gráficas de processamento, ou GPU (*Graphic Processor Unit*) como são conhecidas. Normalmente, *shaders* nomeiam conjuntos de instruções enviadas à GPU com o objetivo de alcançar efeitos visuais, como por exemplo a alteração na iluminação e tonalização, ou adição de detalhes sem a necessidade de modificação da malha tridimensional dos objetos. No entanto, é importante ressaltar que as placas gráficas atuais também podem ser utilizadas para processamento de uso geral, e não somente para produção de imagens.

Com o advento das placas gráficas, no início estes equipamentos disponíveis não eram programáveis. Em outras palavras, não havia forma de se enviar um conjunto de instruções a fim de alcançar um objetivo específico. As placas gráficas possuíam basicamente a função de acelerar o processamento gráfico, e não o de alteração do resultado visual final. Com a evolução das placas gráficas, bibliotecas como OpenGL e DirectX passaram a incorporar recursos para programar as novas GPUs, adicionando funções especiais de tonalização em suas APIs.

3.1 Arquitetura do fluxo de processamento gráfico programável

A evolução do *hardware* das placas gráficas pode ser dividida em 4 gerações:

• **Primeira geração**: o hardware gráfico era capaz de trabalhar com transformações geométricas em vértices e recorte de polígonos.

• **Segunda geração**: passou a incluir a capacidade de processar iluminação por vértice.

• **Terceira geração**: acrescentou funcionalidades para amostragem de texturas.

• **Quarta geração**: inicia a era dos processadores de vértices e fragmentos capazes de serem programados através da API gráfica.

Neste trabalho foram utilizados hardwares atuais, de quarta geração. O avanço significativo ocorrido nessa geração em relação às gerações passadas está na introdução das GPUs programáveis, que permitem configurar o comportamento das etapas de transformação geométrica (*Vertex shader*) e iluminação (*Pixel shader*).

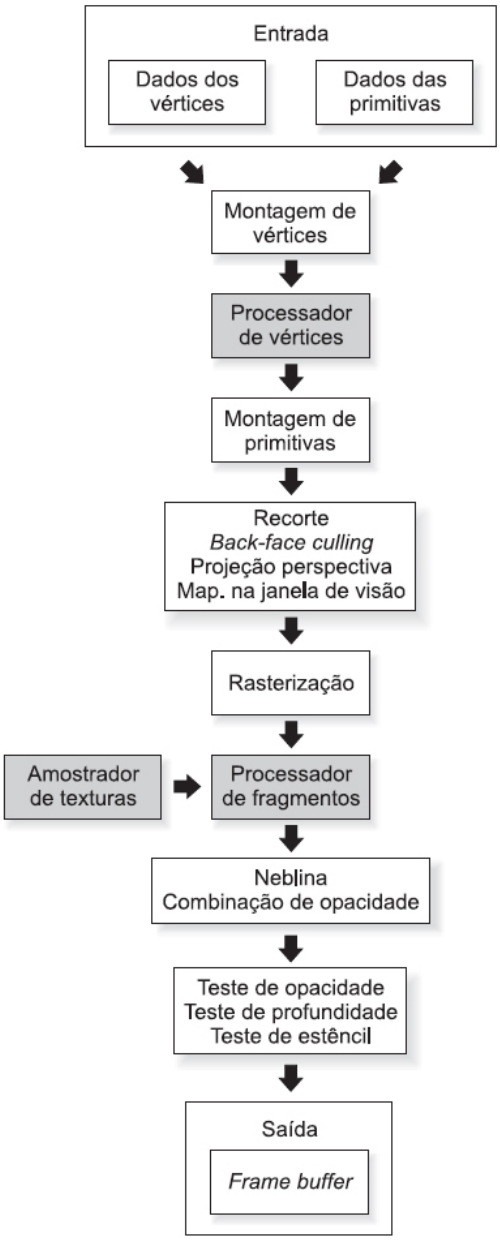

Fig. 3.1: Diagrama típico de um fluxo programável de síntese da imagem (Adaptado de Batagelo (2007)).

Na Figura 3.1, é mostrado o fluxo das placas gráficas programáveis para síntese da imagem. No caso dos *hardwares* compatíveis com o modelo de *shader* 3.0, existe também a capacidade de se

realizar amostragem de texturas dentro do processador de vértices. Apesar de nesta dissertação ser utilizado o modelo de *shader* 3.0, esta funcionalidade não foi explorada.

Entre as etapas do processamento gráfico das placas atuais, destacam-se:

• **Processador de vértices:** para cada vértice da geometria obtido após a etapa de montagem, um *shader* é executado pelo processador de vértices. Esta etapa possui o propósito de modificar os atributos dos vértices trabalhados, como posição, cor ou coordenadas de textura, para envio às próximas etapas do fluxo de processamento.

• **Processador de fragmentos:** esta etapa é responsável pela execução de um *fragment shader* sobre cada fragmento produzido pela etapa do rasterizador. Embora também seja conhecido como processador de *pixels*, é importante enfatizar que um fragmento nem sempre será um único *pixel*. Um fragmento corresponde a um ponto em coordenadas da amostragem da tela, que por sua vez é obtido pelo rasterizador a partir da amostragem de primitivas. Isto faz com que a amostragem possa produzir diversos fragmentos para um mesmo *pixel*, e a cor final desse *pixel* será uma combinação desses fragmentos. O propósito dessa etapa tipicamente, é realizar tarefas como iluminação de cena e efeitos relacionados a tonalização dos objetos.

As placas mais recentes, são capazes de trabalhar tanto como um processador de vértices, de fragmentos, ou como um processador de primitivas, de acordo com a demanda. O processador de primitivas consegue acessar as informações associadas a cada primitiva, como por exemplo os vértices que a compõem e as relações de conexão entre suas primitivas adjascentes.

O modelo de *shader* 4.0, utilizado nas placas gráficas mais recentes, apresenta uma arquitetura de *shader* unificada, onde a GPU utiliza o mesmo processador para implementar os cálculos de vértices, geometrias e fragmentos, o que no modelo anterior era realizado em processadores separados.

O uso de processadores separados para cada etapa causa um desempenho inferior em relação a uma arquitetura unificada, onde a GPU aloca dinamicamente recursos do processor para a manipulação de vértices, geometrias e unidades de pixels, dependendo da demanda.

Entre as vantagens da arquitetura unificada podem ser citadas:

- Balanceamento do uso de recursos dinâmico, devido à programação dos processos sob demanda.
- Maior poder ao processador de vértices, com a introdução da precisão total de pontos flutuantes.
- Maior performance graças ao acesso unificado aos cálculos e texturas.

Na Figura 3.2 é apresentada o modelo de arquitetura unificado das atuais placas gráficas.

3.2 *Bump mapping*

A técnica de bump mapping (BLINN, 1978) é utilizada em computação gráfica para se produzir efeitos visuais associados a perturbações na superfície de objetos, sem a necessidade de alterar a sua

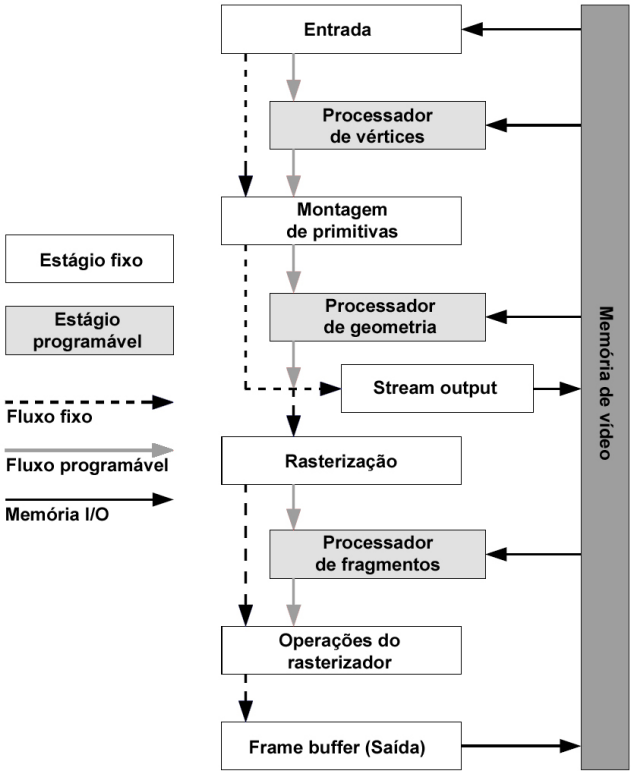

Fig. 3.2: Diagrama de um fluxo programável de síntese da imagem para arquitetura unificada (Adaptado de Patidar, Bhattacharjee e Narayanan (2007)).

geometria.

A forma mais comum desta implementação faz uso de um mapa de relevo (Figura 3.3-A[1]) em tons de cinza, utilizado para se calcular uma perturbação aos vetores normais da superfície do modelo geométrico (Figura 3.3-B[2]). Uma forma de se calcular essa perturbação dos vetores normais é aplicando a Equação

$$Gradiente\_X = Pixel(X-1, Y) - Pixel(X+1, Y)$$
$$Gradiente\_Y = Pixel(X, Y-1) - Pixel(X, Y+1)$$

sobre cada *pixel* da imagem.

Para cada *pixel* da imagem percorrida, são obtidos índices de perturbação para X e Y, representando respectivamenteas direções horizontal e vertical da textura. Na Equação anterior, esses valores são armazenados em Gradiente_X e Gradiente_Y. Após o cálculo dos gradientes para ambos os eixos, o vetor normal da coordenada UV em questão é ajustado, somando o vetor normal original com o resultado da multiplicação de cada gradiente pelo respectivo eixo, sendo armazenado o resultado no *pixel* da coordenada UV. Para isso aplica-se a Equação

[1] Adaptado de http://freespace.virgin.net/hugo.elias/graphics/x_polybm.htm
[2] Adaptado de http://freespace.virgin.net/hugo.elias/graphics/x_polybm.htm

$$Nova\_Normal = Normal + (U * Gradiente\_X) + (V * Gradiente\_Y).$$

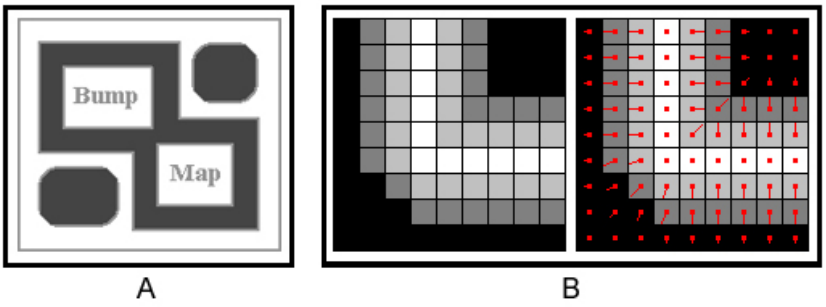

Fig. 3.3: Cálculo do bump mapping.

Com as normais calculadas por *pixel*, é possível realizar os cálculos de iluminação de forma mais precisa do que utilizando apenas as normais dos vértices, apresentando como resultado um objeto com maior nível de detalhes em sua superfície do que o original, apesar da sua geometria permanecer inalterada (Figura 3.4[3]).

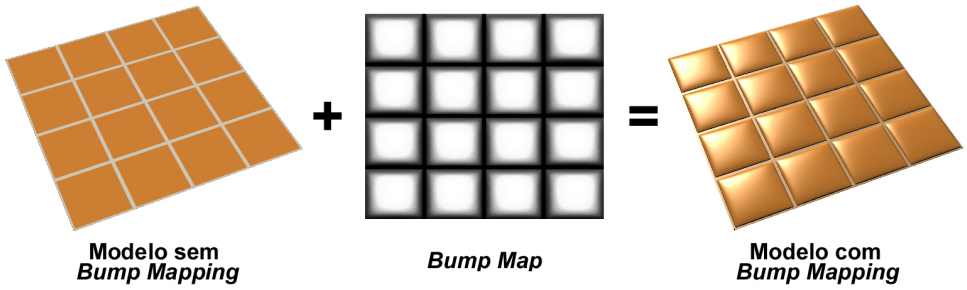

Fig. 3.4: Bump mapping shader.

3.3 *Displacement mapping*

A técnica de *displacement mapping* (COOK, 1984) ao contrário da técnica de *bump mapping*, produz alterações diretamente na malha 3D dos objetos de acordo com uma textura ou mapa de relevo (*height map*) (Figura 3.5[4]).

Esta técnica por muitos anos foi considerada uma técnica específica de sistemas de síntese de imagem de alto custo computacional, como *PhotoRealistic RenderMan*, já que formas para se sintetizar imagens em tempo real ainda não estavam disponíveis, como *Opengl* ou *DirectX*.

O motivo dessa exclusividade era que, em sua implementação original, a técnica de *displacement mapping* realizava uma subdivisão adaptativa da superfície para obter micropolígonos. Alguns deles

[3]Bump mapping: adaptado de http://en.wikipedia.org/wiki/Bump_mapping
[4]Adaptado de http://www.xbitlabs.com/images/video/matrox-parhelia/displacement-1.gif

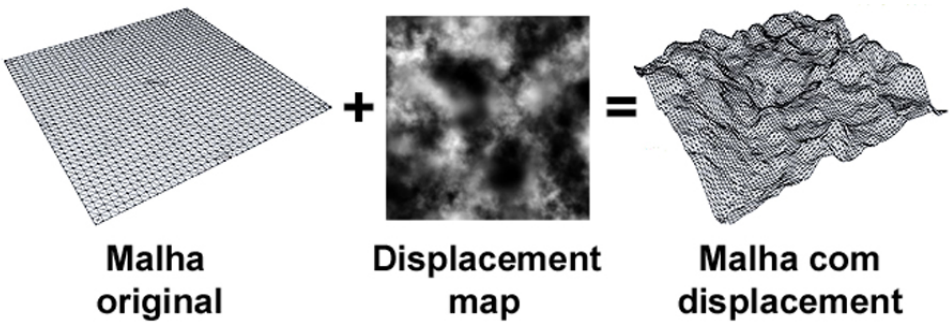

Fig. 3.5: Displacement mapping shader.

chegavam a ter o tamanho equivalente a um *pixel* na tela. Tal processo era computacionalmente caro e portanto, lento.

O elemento básico para a deformação da malha do objeto é o vetor normal obtido a partir do mapa de deslocamento, gerado de forma semelhante ao processo de geração do mapa de normais do *bump mappig* (Figura 3.2). São utilizadas texturas contendo as componentes RGBA (Vermelho (R), verde (G), azul (B) e transparência (A)), ou texturas em tons de cinza.

Para o primeiro tipo, RGBA, as informações de nível de deslocamento são armazenadas no canal de transparência (A) e as informações sobre a direção desse deslocamento são armazenadas nas componentes RGB. No caso de texturas em tons de cinza, também chamadas de escalares, o deslocamento segue o sentido da normal do respectivo vértice, utilizando as informações contidas na textura para determinar o nível desse deslocamento.

Algumas implementações de *displacement mapping* realizam subdivisões da malha original do modelo, causando um aumento considerável do número de polígonos e, consequentemente, do tempo necessário para síntese da imagem.

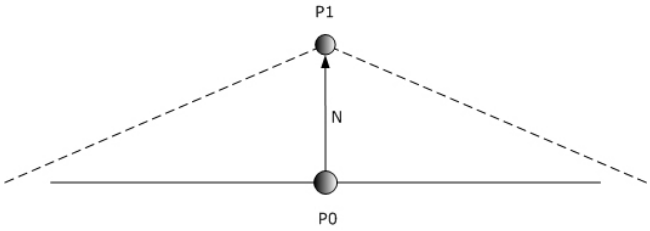

Fig. 3.6: Princípio do *displacement mapping*.

O esquema da Figura 3.6[5] demonstra como o deslocamento de cada vértice ocorre, governado pela Equação

$$P_1 = P_0 + (N \times df \times uf)$$

[5] Adaptada de http://www.ozone3d.net/tutorials/vertex_displacement_mapping_p02.php

onde P_0 é a posição original do vértice, P_1 a posição após o deslocamento, *N* é o vetor normal do vértice, *df* o fator de deslocamento normalizado e *uf* um fator de escala definido pelo usuário. O valor de *df* pode ser obtido pela Equação

$$df = 0,30 \times dv_x + 0,59 \times dv_y + 0,11 \times dv_z$$

que converte valores RGB para tons de cinza.

O valor de *dv* nessa equação é obtido do mapa de deslocamento para o corrente vértice processado, proveniente das componentes RGB, armazenadas nas coordenadas dv_x, dv_y e dv_z. Os valores 0,30, 0,59 e 0,11 determinam o uso de cada componente, a fim de tonalizar o resultado final de cada *pixel* da imagem.

3.4 *Normal mapping*

Pode-se dizer que o *normal mapping* (KILGARD, 2000), também conhecido como *Dot3 bump mapping* é uma evolução da técnica de *bump mapping*. Enquanto o *bump mapping* utiliza mapas de relevo, em tonalidades de cinza, para perturbar as normais da superfície do modelo geométrico, o *normal mapping* realiza uma completa substituição das normais, informadas por meio de uma textura gerada previamente, contento novos vetores normais.

A geração do mapa de normais ocorre tomando como base um modelo geométrico de baixa densidade de polígonos com coordenadas de textura já mapeadas, sendo feita uma cópia desse modelo geométrico para uso posterior.

Com uma das cópias do modelo, realiza-se várias subdivisões na malha poligonal, trabalhando o posicionamento dos vértices da malha subdividida, a fim de incorporar detalhes como rugas, cicatrizes ou deformidades. Um exemplo desses dois modelos podem vistos na Figura 3.7[6], com o modelo de alta densidade de polígonos posicionado à esquerda e o modelo de baixa densidade de polígonos posicionado a direita.

De posse desse modelo geométrico de alta densidade de polígonos, que possui as mesmas coordenadas de textura do modelo de baixa densidade de polígonos (Figura 3.8-A e B[7]), copiado previamente, toma-se como base as coordenadas de textura do modelo, iniciando a geração de uma nova textura, que irá receber normais.

São percorridos os *pixels* dessa imagem a ser gerada, calculando o vetor normal da superfície do modelo geométrico de alta densidade de polígonos, correspondente à coordenada de textura sendo trabalhada. O resultado do cálculo do vetor normal é armazenado no *pixel* dessa nova textura, como demonstrado na Figura 3.8-B[8].

[6] Adaptada de http://www.highend3d.com/maya/tutorials/texturing/229-2.html
[7] Adaptada de http://www.highend3d.com/maya/tutorials/texturing/229-2.html
[8] Adaptada de http://www.highend3d.com/maya/tutorials/texturing/229-2.html

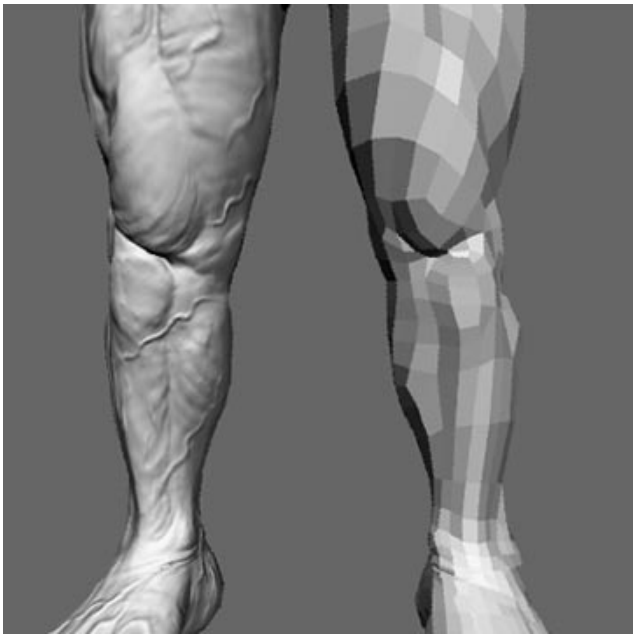

Fig. 3.7: Exemplo de uso dos mapas de normais.

Após percorrer todos os *pixels*, é obtida uma textura chamada de mapa de normais. Essa textura será aplicada sobre o modelo de baixa densidade de polígonos, utilizando para isso o shader de *normal mapping* (Figura 3.8-C[9]).

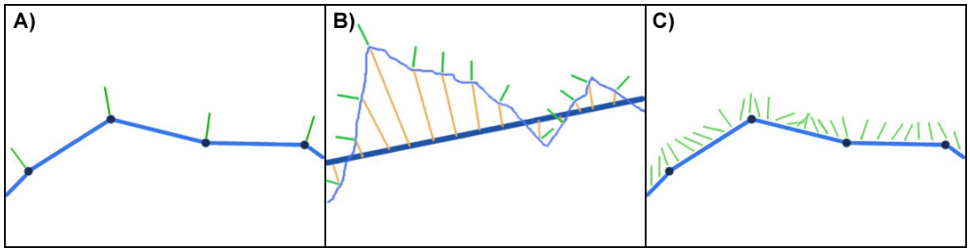

Fig. 3.8: Gerando os mapas de normais.

O *shader* de *normal mapping* trabalha utilizando as cordenadas de textura do modelo de baixa densidade de polígonos como referência. Ao invés de se utilizar as normais do modelo geométrico, estas são substituídas pelas vetores encontrados no mapa de normais, obtido a partir do cálculo das normais do modelo de alta densidade de polígonos.

Na Figura 3.9[10], são exibidos 3 modelos geométricos. O primeiro é um modelo de alta densidade de polígonos, possuindo 4 milhões de triângulos. O segundo modelo é uma representação aramada do modelo de baixa densidade de polígonos, visto no terceiro modelo. Este modelo com 500 triângulos não é capaz de expressar todas as nuances e detalhes que o modelo de alta densidade de polígonos consegue. No entanto, utilizando as normais contidas no mapa de normais para se calcular

[9] Adaptada de http://www.highend3d.com/maya/tutorials/texturing/229-2.html
[10] Adaptado de http://en.wikipedia.org/wiki/Normal_mapping

a tonalização final de cada *pixel*, obtem-se um resultado final que dá a impressão de se está trabalhando com modelos de número de polígonos bem maior do que os utilizados, acelerando os cálculos e consequentemente a velocidade final do sistema.

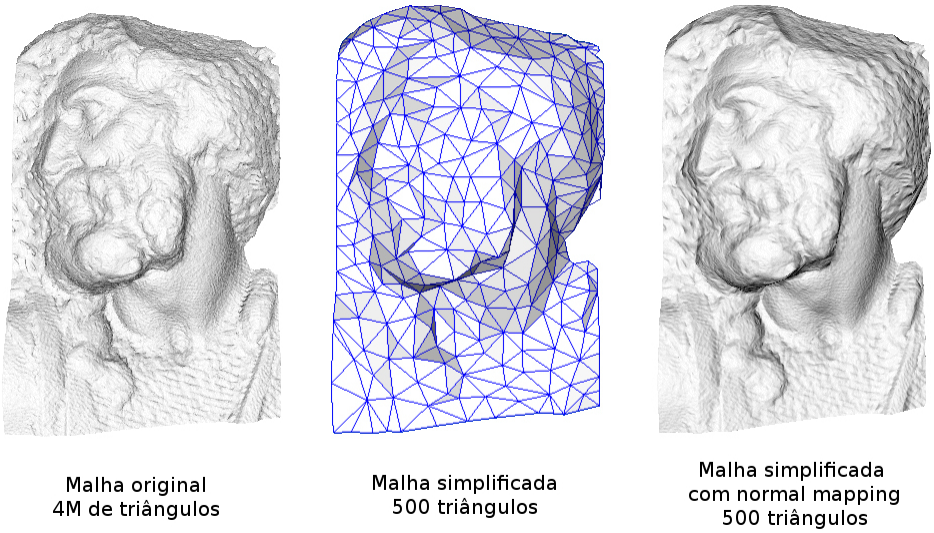

Fig. 3.9: Normal mapping.

3.4.1 Espaço tangente

A utilização de espaço tangente ou espaço de textura é necessário nessa forma de implementação porquê para que os cálculos de iluminação funcionem, os vetores de luz precisam ser transformados do espaço de câmera para o espaço de textura. Para isso é necessário um sistema de coordenadas local à textura em cada vértice. Os três eixos necessários são a normal, tangente e binormal (Figura 3.10-A), utilizados para se criar um sistema de coordenadas, para se transformar os vetores de luz para o espaço apropriado.

O espaço tangente é a matriz de rotação criada pela normal, tangente e binormal no raio de origem (Figura 3.10-B), ou seja, o vetor normal do modelo de baixa densidade de polígonos, onde será aplicado o mapa de normais.

3.5 Considerações finais

Neste capítulo foram apresentados conceitos sobre o funcionamento básico da arquitetura das placas aceleradoras gráficas, funcionamento genérico dos *shaders* e descritas em detalhes três técnicas para melhoria dos resultados gráficos: *bump*, *displacement* e *normal mapping*.

A técnica de *bump mapping* é capaz de adicionar detalhes visuais na superfície dos modelos geométricos, sem a necessidade de alteração de sua malha tridimensional. O cálculo de iluminação

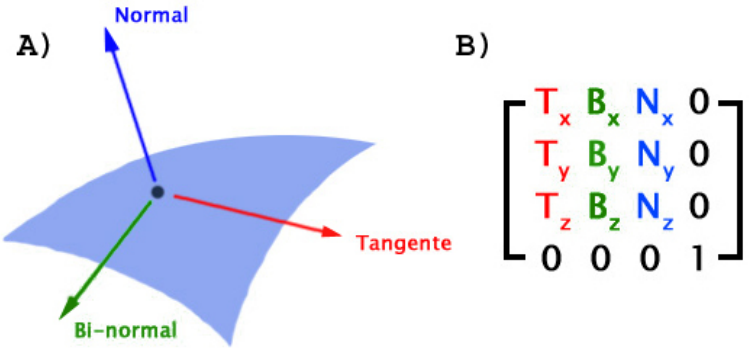

Fig. 3.10: Vetores normal, tangente e binormal no espaço tangente, criado a partir do raio de origem (Vetor normal do modelo de baixa densidade de polígonos), e matriz de rotação criada a partir desses vetores.

desta técnica no entanto, utiliza uma textura em tons de cinza, que é interpretada como informação de relevo. Esta técnica não foi aplicada nesta dissertação apenas por uma questão de escolha pessoal, no entanto, fica a critério do usuário final qual a melhor opção para cada caso em particular.

Na segunda técnica apresentada, *displacement mapping*, pode ocorrer a alteração da malha original do modelo, dependendo da quantidade de polígonos necessária para se apresentar todos os detalhes na superfície. Este fato pode fazer com que o sistema perca a capacidade de sintetizar imagens em tempo real, devido ao elevado número de cálculos necessários para o processamento dos vértices. Apesar de serem adicionados detalhes visuais diretamente na malha geométrica do modelo, o problema de perda de performance inabilita a escolha desta técnica para uso em tempo real.

A terceira técnica apresentada foi escolhida para o desenvolvimento deste trabalho por não realizar subdivisão da malha do modelo geométrico, mantendo a performance do sistema, e ainda assim apresentando resultados visuais satisfatórios, adicionando detalhes visuais a superfície do modelo tridimensional. A técnica utiliza modelos de alta densidade de polígonos para extrair as normais diretamente para uma textura, que é aplicada em um modelo de baixa densidade de polígonos, substituindo completamente as normais de cada pixel na tela.

Apesar do uso de *shaders* adicionar detalhes visuais à superfície dos modelos geométricos com baixa densidade de polígonos, dando a impressão de que modelos com alta densidade de polígonos estão sendo utilizados, é importante ressaltar que um número excessivamente pequeno de polígonos pode causar um problema chamado de “efeito silhueta”, onde o número de vértices e polígonos do modelo não são capazes de representar a silhueta básica dos modelos de alta poligonação. Este efeito pode ser visto na Figura 3.11, onde o modelo de uma esfera é representado com diferentes volumes de vértices.

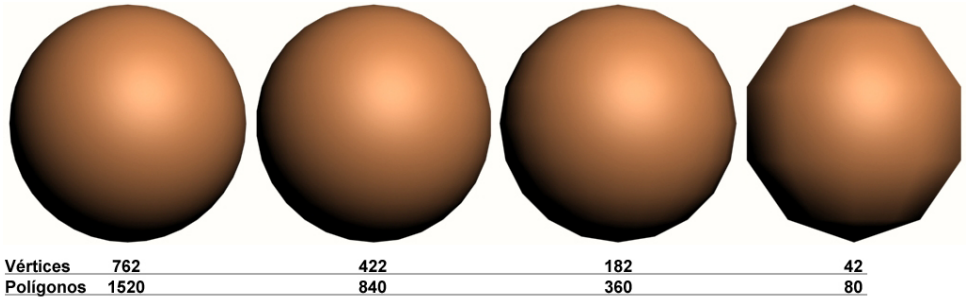

Fig. 3.11: Efeito silhueta causado pelo uso de modelos geométricos com número muito baixo de polígonos.

Pode-se reparar que um número muito pequeno de vértices, causa uma representação incorreta da silhueta da esfera, o que mostra a necessidade de um estudo sobre a quantidade mínima de vértices para que um modelo de baixa densidade de polígonos represente a silhueta de um modelo de alta densidade de polígonos.

Capítulo 4

Técnicas e protótipos desenvolvidos

Neste capítulo são apresentadas duas técnicas para a modelagem e apresentação de rugas em modelos faciais tridimensionais.

A primeira técnica baseia-se na utilização de áreas de influência para o controle em tempo real da apresentação de rugas. Estas áreas de influência são especificadas por mapas de textura que indicam os locais de apresentação das rugas. As imagens dos mapas de texturas são enviadas, agrupadas em arranjos de textura, em um único bloco para a placa gráfica para permitir que modificações do número de áreas de influência possam ser efetuadas de forma rápida e flexível, sem a necessidade de alteração do código do *shader* responsável pelo cálculo da exibição das rugas. Nesta técnica, o controle da apresentação das rugas se dá por um vetor, com índices diretamente relacionados com cada área de influência, que é populado e enviado à placa gráfica a cada quadro da animação.

A segunda técnica associa vetores a cada um dos vértice do modelo geométrico. Estes vetores informam o sentido em que o deslocamento do vértice, causará o surgimento de rugas sobre o modelo facial. O Vetor Direção de Rugas, além de informar um sentido para a apresentação de rugas, também estabelece uma região de influência para o deslocamento de vértices vizinhos, com o intuito de suavizar a troca entre expressões faciais. Ao contrário da primeira técnica, o controle de apresentação das rugas é feito internamente a placa gráfica, de acordo com o deslocamento dos próprios vértices no espaço tridimensional.

Para o desenvolvimento de ambas as técnicas, foram utilizadas linguagens e ferramentas de uso gratuito, a maioria delas sob licença de uso público (GPL), outras sob licença proprietária, porém disponíveis para *download* e uso pessoal no próprio *site* das empresas desenvolvedoras. Estas ferramentas foram utilizadas para facilitar o desenvolvimento e produção das informações necessárias ao funcionamento das técnicas, como modelos geométricos e texturas.

A seguir são apresentadas as duas técnicas, explicando a motivação de ambas, suas vantagens e desvantagens, os conceitos envolvidos e detalhes técnicos dos protótipos implementados, assim como das linguagens, bibliotecas e ferramentas utilizadas. Em seguida serão feitas algumas considerações finais sobre o que foi realizado, incluindo o posicionamento das soluções aqui propostas em relação ao estado da técnica.

4.1 Cálculo dos vetores normais e tangentes dos vértices.

Para ambas as técnicas é necessário o cálculo dos vetores normais e tangentes para os modelos faciais.

A formulação e algoritmos utilizados nesta dissertação para o cálculo destes vetores são baseados nas Seções 6.7.1 (vetores normais) e 6.8.3 (vetores tangentes) de Lengyel (2002). Na Seção 4.1.1 é apresentado o cálculo do vetor normal e na Seção 4.1.2 o cálculo do vetor tangente.

4.1.1 Cálculo dos vetores normais.

Para os cálculos do modelo de iluminação utilizado neste trabalho é necessária a definição de vetores normais para cada vértíce do modelo geométrico. Para o cálculo desses vetores, primeiramente é necessário encontrar o vetor normal de cada polígono, que neste trabalho são triângulos. Por convenção, neste trabalho é adotado que os vértices dos triângulos estão organizados no sentido anti-horário.

O vetor normal unitário N de um polígono descrito pelos pontos P_0, P_1 e P_2 é obtido por

$$N = \frac{(P_1 - P_0) \times (P_2 - P_0)}{|(P_1 - P_0) \times (P_2 - P_0)|}. \tag{4.1}$$

A partir dos vetores normais dos triângulos, a normal de um determinado vértice é calculada, como expresso pela Equação

$$N_{vertex} = \frac{\sum_{i=1}^{k} N_i}{|\sum_{i=1}^{k} N_i|} \tag{4.2}$$

tomando-se a resultante, devidamente normalizada, da soma vetorial das normais de todos triângulos que possuem este vértice em comum.

O código a seguir foi utilizado para o cálculo das normais em ambas as técnicas desenvolvidas nesta dissertação:

```
public void computeNormals() {
    int i, k;
    Vector3 v0 = new Vector3();
    Vector3 v1 = new Vector3();
    Vector3 v2 = new Vector3();
    Vector3 vv0 = new Vector3(0.0f, 0.0f, 0.0f);
    Vector3 vv1 = new Vector3(0.0f, 0.0f, 0.0f);
    Vector3 vv2 = new Vector3(0.0f, 0.0f, 0.0f);
```

```
    k = 0;
    for (i = 0; i < numFace; i++, k += 3) {
        vv0 = vertexPosition[facesV[i].p1];
        vv1 = vertexPosition[facesV[i].p2];
        vv2 = vertexPosition[facesV[i].p3];
        v0.setXYZ(vv0.x, vv0.y, vv0.z);
        v1.setXYZ(vv1.x, vv1.y, vv1.z);
        v2.setXYZ(vv2.x, vv2.y, vv2.z);

        v1 = v1.minus(v0);
        v2 = v2.minus(v0);

        Vector3 aux1 = v1.CROSS(v1, v2);
        Vector3 aux2 = module(aux1);

        Vector3 normalFace = new Vector3(
                aux1.x/aux2.x,
                aux1.x/aux2.x,
                aux1.x/aux2.x);

        facesV[i].normX = aux1.x;
        facesV[i].normY = aux1.y;
        facesV[i].normZ = aux1.z;
    }

    Vector3 pn = new Vector3(0.0f, 0.0f, 0.0f);

    for (i = 0; i < numVertex; i++) {
        vertexNormal[i] = new Vector3(0.0f, 0.0f, 0.0f);
        pn = new Vector3(0.0f, 0.0f, 0.0f);

        for (int wi = 0; wi < totalFacesAdjacentes[i]; wi++) {
            pn.x += facesV[facesAdjacentes[i][wi]].normX;
            pn.y += facesV[facesAdjacentes[i][wi]].normY;
            pn.z += facesV[facesAdjacentes[i][wi]].normZ;
        }

        pn.normalize();

        vertexNormal[i] = pn;
    }
}
```

4.1.2 Cálculo dos vetores tangentes.

As técnicas para apresentação de rugas desenvolvidas neste trabalho baseiam-se na abordagem de mapas de normais (*normal mapping*). Um mapa de normal é um mapa de textura, no qual cada elemento especifica um novo vetor normal para a superfície do modelo geométrico, relativo a respectiva coordenada de textura definida. Uma vez que na estratégia de mapas de normais o vetor $V = (0, 0, 1)$ representa a normal original da superfície do objeto é necessário utilizar um sistema de coordenadas, denominado espaço tangente, que permita transformar os vetores normais do espaço do objeto para este novo espaço e vice-versa.

Estando o eixo Z relacionado ao vetor normal de cada vértice, os eixos X e Y são mapeados para corresponder às coordenadas *u* e *v* do mapeamento de textura do mapa de normais. As coordenadas *u* e *v* correspondem respectivamente em um plano bidimensional às coordenadas X e Y, variando de 0 a 1 sobre o comprimento de ambos os eixos.

Para um ponto Q dentro de um polínono triangular, deve ser possível escrever a Equação

$$Q - P_0 = (u - u_0)T + (v - v_0)B, \tag{4.3}$$

onde T e B são vetores tangentes alinhados à textura do mapa de normais, P_0 é a posição de um dos vértices que compõem o polígono triangular, e $\langle u_0,v_0\rangle$ são as coordenadas de textura desse vértice. A letra B corresponde ao termo Binormal, porém não é um termo intuitivo, visto que neste caso específico, B representa a direção tangente à supefície, e não ao vetor normal. Então para que não haja equívocos, será utilizado o termo Bitangente.

Supondo um polígono triângular composto pelos vértices P_0, P_1 e P_2, cujas coordenadas de textura são dadas respectivamente por $\langle u_0,v_0\rangle$, $\langle u_1,v_1\rangle$ e $\langle u_2,v_2\rangle$. Os cálculos podem ser realizados de maneira simplificada, trabalhando de maneira relativa ao vértice P_0. Conforme as equações

$$\begin{aligned} Q_1 &= P_1 - P_0 \\ Q_2 &= P_2 - P_0 \end{aligned} \tag{4.4}$$

e

$$\begin{aligned} \langle u_1, v_1\rangle &= \langle u_1 - u_0, v_1 - v_0\rangle \\ \langle u_2, v_2\rangle &= \langle u_2 - u_0, v_2 - v_0\rangle. \end{aligned} \tag{4.5}$$

Então resolvendo as Equações

$$\begin{aligned} Q_1 &= u_1T + v_1B \\ Q_2 &= u_2T + v_2B \end{aligned} \tag{4.6}$$

para T e B, obtém-se um sistema linear com três T e três B desconhecidos, e as seis Equações, correspondentes às componentes X, Y e Z das duas Equações, visto em

$$\begin{bmatrix} Q_{1x} & Q_{1y} & Q_{1z} \\ Q_{2x} & Q_{2y} & Q_{2z} \end{bmatrix} = \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix} \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix}. \tag{4.7}$$

Multiplicando ambos os lados da Equação pelo inverso da Matriz $\langle u, v\rangle$, obtém-se a Equação

$$\begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix} = \frac{1}{s_1 t_2 - s_2 t_1} \begin{bmatrix} v_2 & -v_1 \\ -u_2 & u_1 \end{bmatrix} \begin{bmatrix} Q_{1x} & Q_{1y} & Q_{1z} \\ Q_{2x} & Q_{2y} & Q_{2z} \end{bmatrix} \tag{4.8}$$

onde são obtido os vetores tangentes não normalizados de T e B, para o polígono triangular cujos vértices são P_0, P_1 e P_2. Para calcular os vetores tangentes de um vértice específico, tira-se a média entre todos os polígonos triangulares que compartilham o vértice, de forma similar ao cálculo dos vetores normais.

De posse do vetor normal (N) e dos vetores tangente (T) e bitangente (B) para um vértice, é possível transformar o mesmo do espaço tangente para o espaço do objeto, utilizando a Matriz

$$M_1 = \begin{bmatrix} T_x & B_x & N_x \\ T_y & B_y & N_y \\ T_z & B_z & N_z \end{bmatrix}. \tag{4.9}$$

Para a transformação no sentido contrário, do espaço do objeto para o espaço tangente, que será utilizado para os cálculos da direção da luz, utiliza-se a inversa da Matriz M_1.

Não é necessariamente verdade que os vetores tangentes são perpendiculares entre si ou com o vetor normal, por isso o inverso de uma matriz não é obrigatoriamente a sua transposta. No entanto é seguro afirmar que os três vetores (TBN) serão pelo menos próximos da ortogonalidade, então pode-se aplicar o algoritmo de Gram-Schmidt (TREFETHEN; BLAU, 1997) sem que ocorram distorções inaceitáveis. Com este processo, novos vetores tangentes T' e B' serão obtidos pelas Equações

$$\begin{aligned} T' &= T - (N \cdot T)N, \\ B' &= B - (N \cdot B)N - (T' \cdot B)T'. \end{aligned} \tag{4.10}$$

Normalizando os vetores T' e B' e armazenando os mesmos como tangente e bitangente de um vértice, obtém-se a Matriz

$$M_2 = \begin{bmatrix} T'_x & T'_y & T'_z \\ B'_x & B'_y & B'_z \\ N'_x & N'_y & N'_z \end{bmatrix}, \tag{4.11}$$

com a qual é possível transformar o vetor direção da luz com amostragens do mapa de normais.

O código a seguir foi utilizado para o cálculo das tangentes em ambas as técnicas desenvolvidas nesta dissertação:

```
public void calculaTangentes2() {
    Vector3[] tan1 = new Vector3[numVertex];
    Vector3[] tan2 = new Vector3[numVertex];

    for (int a = 0; a < numVertex; a++) {
        tan1[a] = new Vector3();
        tan2[a] = new Vector3();
    }

    for (int a = 0; a < numFace; a++) {
        int i1 = facesV[a].p1;
        int i2 = facesV[a].p2;
        int i3 = facesV[a].p3;

        Vector3 v1 = vertexPosition[i1];
        Vector3 v2 = vertexPosition[i2];
        Vector3 v3 = vertexPosition[i3];

        Vector3 w1 = vertexTexCoord[i1];
        Vector3 w2 = vertexTexCoord[i2];
        Vector3 w3 = vertexTexCoord[i3];

        float x1 = v2.x - v1.x;
        float x2 = v3.x - v1.x;
        float y1 = v2.y - v1.y;
        float y2 = v3.y - v1.y;
        float z1 = v2.z - v1.z;
        float z2 = v3.z - v1.z;

        float s1 = w2.x - w1.x;
        float s2 = w3.x - w1.x;
        float t1 = w2.y - w1.y;
        float t2 = w3.y - w1.y;

        float r = 1.0F / (s1 * t2 - s2 * t1);
        Vector3 sdir = new Vector3(
                (t2 * x1 - t1 * x2) * r,
                (t2 * y1 - t1 * y2) * r,
                (t2 * z1 - t1 * z2) * r);
        Vector3 tdir = new Vector3(
                (s1 * x2 - s2 * x1) * r,
                (s1 * y2 - s2 * y1) * r,
                (s1 * z2 - s2 * z1) * r);

        tan1[i1].x = tan1[i1].x + sdir.x;
        tan1[i1].y = tan1[i1].y + sdir.y;
        tan1[i1].z = tan1[i1].z + sdir.z;

        tan1[i2].x = tan1[i2].x + sdir.x;
        tan1[i2].y = tan1[i2].y + sdir.y;
        tan1[i2].z = tan1[i2].z + sdir.z;

        tan1[i3].x = tan1[i3].x + sdir.x;
        tan1[i3].y = tan1[i3].y + sdir.y;
        tan1[i3].z = tan1[i3].z + sdir.z;

        tan2[i1].x = tan2[i1].x + tdir.x;
        tan2[i1].y = tan2[i1].y + tdir.y;
        tan2[i1].z = tan2[i1].z + tdir.z;

        tan2[i2].x = tan2[i2].x + tdir.x;
        tan2[i2].y = tan2[i2].y + tdir.y;
        tan2[i2].z = tan2[i2].z + tdir.z;

        tan2[i3].x = tan2[i3].x + tdir.x;
        tan2[i3].y = tan2[i3].y + tdir.y;
```

```
            tan2[i3].z = tan2[i3].z + tdir.z;
        }

        for (int a = 0; a < numVertex; a++) {
            Vector3 n = new Vector3(
                    vertexNormal[a].x,
                    vertexNormal[a].y,
                    vertexNormal[a].z);
            Vector3 t = new Vector3(
                    tan1[a].x,
                    tan1[a].y,
                    tan1[a].z);

            // Gram-Schmidt orthogonalize
            float dotNT = n.x * t.x + n.y * t.y + n.z * t.z;

            Vector3 nScaleDotNT = new Vector3(
                    n.x * dotNT,
                    n.y * dotNT,
                    n.z * dotNT);

            Vector3 tMinusnScaleDotNT = new Vector3(
                    t.x - nScaleDotNT.x,
                    t.y - nScaleDotNT.y,
                    t.z - nScaleDotNT.z);

            tMinusnScaleDotNT.normalize();

            vertexTangentU[a] = new Vector3(
                    tMinusnScaleDotNT.x,
                    tMinusnScaleDotNT.y,
                    tMinusnScaleDotNT.z);

            // Calculate handedness
            Vector3 t2 = new Vector3(tan2[a].x, tan2[a].y, tan2[a].z);
            Vector3 nCrosst = new Vector3(
                    n.y * t.z - n.z * t.y,
                    n.z * t.x - n.x * t.z,
                    n.x * t.y - n.y * t.x);

            float nCrosstDOTt2 = nCrosst.x * t2.x + nCrosst.y * t2.y + nCrosst.z * t2.z;

            vertexTangentU[a].w = (nCrosstDOTt2 < 0.0f) ? -1.0f : 1.0f;

            vertexTangentV[a] = new Vector3(
                    n.y * vertexTangentU[a].z - n.z * vertexTangentU[a].y,
                    n.z * vertexTangentU[a].x - n.x * vertexTangentU[a].z,
                    n.x * vertexTangentU[a].y - n.y * vertexTangentU[a].x);

            vertexTangentV[a] = new Vector3(
                    vertexTangentV[a].x * vertexTangentU[a].w,
                    vertexTangentV[a].y * vertexTangentU[a].w,
                    vertexTangentV[a].z * vertexTangentU[a].w);

            vertexNormal[a].normalize();
            vertexTangentU[a].normalize();
            vertexTangentV[a].normalize();
        }
    }
```

4.2 Técnica baseada em áreas de influência descritas por mapas de textura.

A primeira técnica desenvolvida neste trabalho é baseada na utilização de regiões de influência para informar os locais de apresentação das rugas no modelo facial. As áreas de influência são organizadas na forma de arranjos de textura para facilitar sua utilização, tanto no momento do envio das informações ao *shader*, quanto nos cálculos realizados internamente ao mesmo.

O controle da intensidade de exibição das rugas é realizado de acordo com informações passadas ao shader por meio de uma estrutura de dados do tipo vetor, chamado neste trabalho de Vetor de Ativação. Neste vetor, o programador associa cada índice a uma determinada região de influência, ou seja, o primeiro índice do vetor corresponde a primeira área de influência, o segundo índice corresponde a segunda área, e assim por diante.

O Vetor de Ativação recebe valores ponto flutuante que vão de 0 a 1, onde o valor 0 representa ausência de rugas, e o valor 1 corresponde a exibição total das rugas. Internamente ao *shader*, cada área de influência é multiplicada pelo valor contido no Vetor de Ativação correspondente. O somatório dessas multiplicações é armazenado em uma textura que será utilizada para interpolar entre o resultado de dois *normal mapping*, um deles contendo as rugas desejadas para apresentação sobre o modelo facial.

No protótipo implementado foi utilizada a técnica de *normal mapping* para apresentação das rugas sobre a superfície do modelo facial. Essa escolha se deve ao fato de que esta técnica possibilita um aumento do nível de detalhes visuais, como sombras e reflexos de iluminação, sem a necessidade de que o modelo facial 3D possua uma malha com número elevado de polígonos.

O protótipo foi desenvolvido em C Sharp.NET [1], em conjunto com o framework XNA [2], além da linguagem HLSL [3] para desenvolvimento de *shaders*. Esta escolha foi motivada pelo abrangente número de bibliotecas e funções disponíveis no ambiente C Sharp.NET e no *framework* XNA para o desenvolvimento gráfico, como tratamento de imagens, vetores 3D e suporte nativo a modelos tridimensionais.

4.2.1 Motivação

A utilização de texturas para definir as áreas de influência permite que a maioria dos *softwares* de edição de imagens, possa gerar o conteúdo necessário ao funcionamento da técnica aqui desenvolvida. A utilização de regiões de influência descritas por mapas de textura permite definir de forma flexível a forma e os locais nos quais as rugas podem ocorrer. A configuração desses mapas de textura é feita desenhando em um mapa de textura as regiões onde deverão ser apresentadas as rugas. Recomenda-se que as bordas das regiões das áreas de influência desenhadas no mapa de texturas não sejam abruptas para permitir transições suaves entre as áreas de influência.

[1] http://msdn.microsoft.com/en-us/vcsharp/default.aspx (Acessado em 26/06/2009)
[2] http://creators.xna.com (Acessado em 26/06/2009)
[3] http://msdn.microsoft.com/en-us/library/bb509561(VS.85).aspx (Acessado em 26/06/2009)

Outra característica importante deste técnica, é a possibilidade da justaposição e/ou sobreposição de diversas áreas de influência. Esta possibilidade é útil principalmente quando duas expressões distintas da face atuam sobre regiões próximas.

A técnica apresentada propõe o agrupamento das texturas como uma forma de tornar dinâmico o número de áreas de influência utilizadas. Este agrupamento é oportuno pela flexibilidade obtida, porém nada impede que as texturas sejam adicionadas uma a uma.

4.2.2 Vantagens e desvantagens

A configuração de áreas de influência em sistemas de animação facial, especificamente os que tratam da apresentação de rugas, pode ser realizada de diversas formas. Por exemplo, em (PASQUARIELLO; PELACHAUD, 2001), a partir de um ponto central, um raio de ação é definido para que rugas sejam modeladas geometricamente de acordo com as expressões realizadas pelo modelo geométrico. As áreas de influência formadas possuem formato de círculos, o que limita a configuração do formato das áreas de influência. Em (WATERS, 1987), o autor apresenta um sistema de músculos virtuais, utilizando uma aproximação biomecânica, onde sua própria estrutura constitui uma zona de influência, selecionando um grupo de vértices sobre a ação de uma força, da mesma forma que o exemplo anterior, a configuração da forma da área de influência é determinada pela forma da estrutura do músculo virtual.

Na técnica proposta neste trabalho, são utilizadas áreas de influência descritas por mapas de textura, o que possibilita uma liberdade na configuração das regiões de apresentação de rugas, podendo estas assumir qualquer formato sobre o modelo facial, além da facilidade de se realizar alterações sobre estas áreas de influência, já que é possível utilizar potencialmente qualquer pacote de edição de imagens que suporte a manipulação de imagens em tons de cinza.

O uso de *shaders* adiciona uma vantagem interessante, já que não são necessárias grandes alterações na lógica do sistema de animação facial, ou seja, a inclusão da técnica em sistemas já prontos pode ser feita adicionando o uso de *shaders* na síntese da imagem final, e o controle das áreas de influência através do Vetor de Ativação, que será explicado na Seção 4.2.3.1.

No entanto, a utilização de texturas possui uma limitação no que se refere a memória. Dependendo da capacidade da placa gráfica utilizada, o número máximo de áreas de influência pode ser maior ou menor, já que o uso excessivo de texturas tende a esgotar rapidamente a memória disponível.

4.2.3 A técnica

4.2.3.1 Dados necessários ao funcionamento da técnica

Para que a técnica possa realizar os cálculos, são necessários alguns dados. Entre estas informações, algumas devem ser geradas previamente em *softwares* específicos, outras serão geradas em tempo real pelo próprio sistema.

As texturas difusa e os mapas de normais são necessários em ambas as técnicas apresentadas nesta dissertação, por isso serão apresentadas separadamente, na Seção 4.4. As outras informações exigidas são: Áreas de influência descritas por mapas de textura e Vetor de ativação, descritas a seguir.

Áreas de influência descritas por mapas de textura:

As áreas de influência são responsáveis por informar ao sistema, as regiões onde as rugas devem aparecer quando houver uma movimentação dos vértices do modelo facial, ou seja, quando determinada região da face for movimentada. Essas áreas devem ser geradas coincidentes com as coordenadas de textura do modelo geométrico (Figura 4.1). Estas coordenadas são os componentes responsáveis por mapear as rugas na superfície do modelo facial.

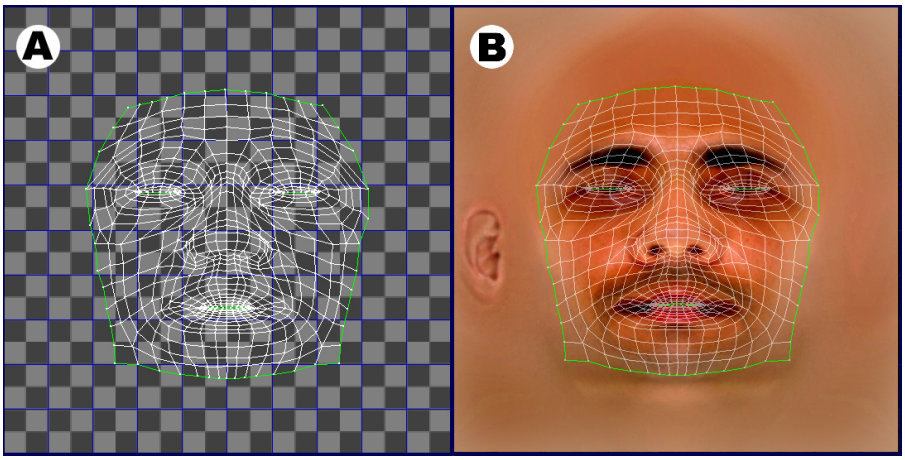

Fig. 4.1: Coordenadas de textura para criação das áreas de influência (A), e as coordenadas aplicadas sobre a textura difusa do sistema (B).

Na Figura 4.2 são definidas, como exemplo, as áreas de influência para as sobrancelhas direita e esquerda utilizando as coordenadas de textura existentes no modelo facial. Vale lembrar que apenas as texturas à direita na figura, em tons de cinza, representam áreas de influência, sendo as mesmas adicionadas ao sistema desenvolvido, para a criação da textura final a ser enviada ao *shader*. Os tons mais escuros nas texturas indicam os locais de maior influência na visibilidade das rugas, sendo que quanto mais escuro, mais visíveis as rugas serão de acordo com a animação facial realizada.

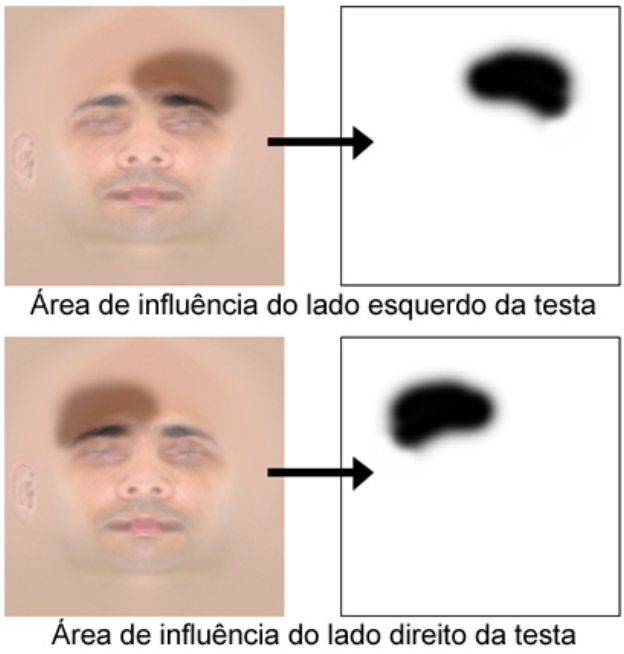

Fig. 4.2: Áreas de influência das sobrancelhas direita e esquerda.

Na técnica em questão, essas áreas de influência são empacotadas dentro de uma textura 3D para o envio à placa gráfica. Essas texturas são também chamadas de texturas volumétricas. No entanto, no caso deste trabalho em particular, as texturas 3D são lidas internamente ao *shader* como se fossem um vetor, composto de uma textura em cada índice.

No momento de inclusão de uma área de influência na textura tridimensional, são copiados todos os pixels da área para a primeira camada da textura 3D. Ao se incluir uma segunda área de influência, os dados são copiados para a segunda camada da textura 3D, e assim por diante até que todas as áreas de influência sejam adicionadas a textura final.

Para repassar as informações contidas em imagens bidimensionais para imagens tridimensionais, o seguinte trecho de código é aplicado:

```
01 Color[] myBits = new Color[
02     texture3D.Width *
03     texture3D.Height *
04     texture3D.Depth];
05
06 for (int d = 0; d < texture3D.Depth; d++)
07 {
08     Color[] bitsDiffuse = new Color[textures2D[d].Width * textures2D[d].Height];
09     textures2D[d].GetData<Color>(bitsDiffuse);
10
11     for (int x = 0; x < texture3D.Width; x++)
12     {
13         for (int z = 0; z < texture3D.Height; z++)
14         {
15             myBits[x + z * texture3D.Width + d * texture3D.Width * texture3D.Height] =
16                 bitsDiffuse[x + z * textures2D[0].Width];
17         }
```

```
18      }
19 }
```

Primeiramente, é criado um vetor onde serão armazenadas as informações de cada uma das áreas de influência. Para isso, é instanciado um vetor de bits de cores (RGB) de dimensão largura $\times$ altura $\times$ profundidade (Linha 01), onde a largura e altura indicam as dimensões das texturas contendo as áreas de influência, e a profundidade indica a quantidade de áreas a serem inseridas.

Para cada textura contendo áreas de influência a serem adicionadas (Linha 06), são lidos os *pixels* da imagem (Linhas 08 e 09) para que a mesma seja percorrida por completo, tanto na sua largura quanto na sua altura (Linas 11 a 13). Cada *pixel* da textura será copiado para o vetor gerado no início do algoritmo (Linhas 15). Este vetor é a textura tridimensional contendo todas as áreas de influência do sistema, que será enviado ao *shader*, para os cálculos de visualização das rugas.

Se forem utilizadas, por exemplo, cinco texturas monocromáticas contendo áreas de influência de dimensões 512$\times$512 *pixels*, a textura 3D final terá dimensões de 512$\times$512$\times$5 pixels. (Figura 4.3). No entanto, é importante lembrar que as dimensões das texturas contendo as áreas de influência, não necessitam ser iguais as texturas difusas ou mapas de normais. O objetivo de se enviar áreas de influência em mapas de texturas com dimensões reduzidas é a economia de memória da placa gráfica.

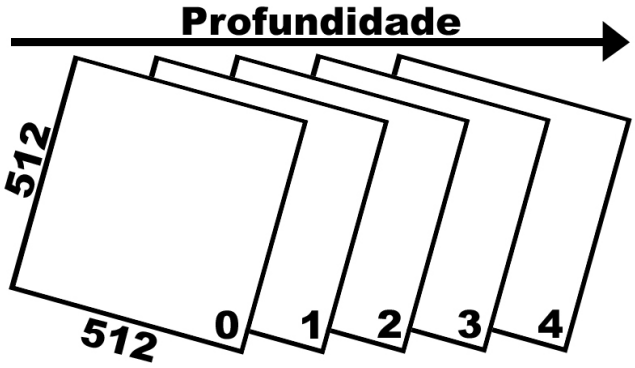

Fig. 4.3: Textura 3D de dimensões 512x512x5.

Caso fosse necessário que, a cada área de influência adicionada pelo usuário, uma nova variável fosse incluída no *shader* para que a área fosse suportada, seria trabalhoso do ponto de vista do programador, além de ir contra a portabilidade. Por esse motivo são utilizadas texturas 3D, já que o *shader* é capaz de determinar suas dimensões em tempo real, obtendo largura, altura e principalmente, a profundidade da imagem, que no caso deste trabalho correspondem ao número de áreas de influência inseridas pelo usuário.

Vetor de Ativação:

Além das áreas de influência, a técnica aqui descrita necessita que mais uma informação seja enviada para o *shader*, esta informação é o Vetor de Ativação, responsável por informar a porcentagem de visualização das rugas sob a influência de determinada área.

Ao passo que a textura contendo as áreas de influência são geradas apenas uma vez, normalmente em uma etapa de pré-processamento ou na construção do modelo facial, o Vetor de Ativação aqui descrito é alterado a cada quadro da animação. O motivo da separação dessa informação do restante a ser enviado ao *shader* deve-se ao objetivo de se obter maior velocidade no sistema, a fim de que o mesmo possa ser utilizado em tempo real.

O Vetor de Ativação é um vetor de números reais, com dimensão igual a profundidade da textura tridimensional que contém as áreas de influência. Cada posição do vetor corresponde a área de influência de mesma posição na textura 3D, e contém valores entre 0 e 1. Se a porcentagem de ativação da posição 0 for de 0,3, as rugas existentes na região da área de influência 0 serão apresentadas com uma porcentagem de visibilidade de 30%, se a porcentagem de ativação for de 0,8, a visibilidade será de 80% e assim por diante.

Para cada área de influência adicionada, o vetor deve sofrer um redimensionamento, ou seja, uma nova posição no vetor deve ser criada para controlar a respectiva área. Na Figura 4.4 é adicionada uma sexta área de influência de índice 5, e paralelamente é adicionada uma sexta posição no Vetor de Ativação, também com índice 5.

A conexão entre esse vetor e a animação facial do sistema deve ser implementada pelo programador diretamente no código, e pode ser realizado de diversas maneiras. Se, por exemplo, for utilizada uma abordagem de músculos virtuais, pode-se definir uma força mínima e máxima permitida por cada músculo, esta força será associada com a posição do vetor que melhor se encaixe à visualização. Se for um músculo da testa, será associada uma posição do Vetor de Ativação correspondente a uma área de influência localizada sobre a testa do modelo facial.

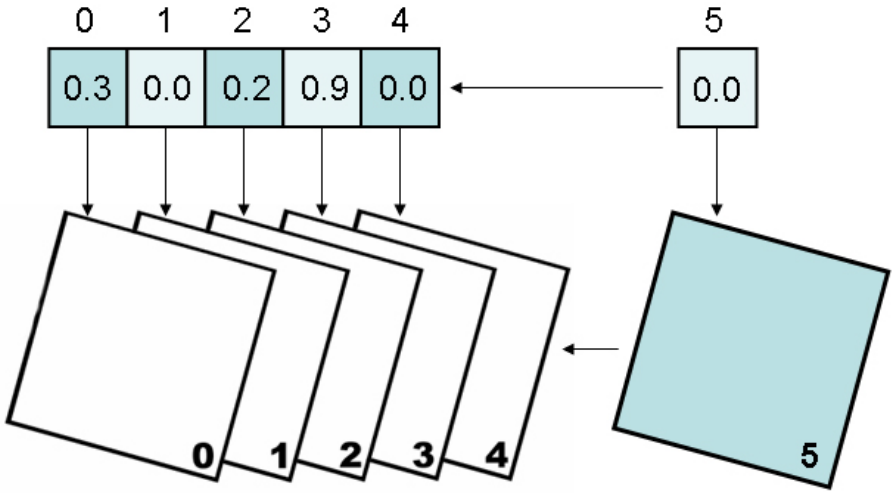

Fig. 4.4: Inclusão de uma área de influência e redimensionamento do Vetor de Ativação.

Na Figura 4.5 é apresentada uma proposta para uma configuração básica de áreas de influência contendo oito áreas distintas. Essas áreas podem ser modificadas para se adequarem as configurações de rugas definidas pelo usuário, inclusive, podendo ser sobrepostas umas às outras.

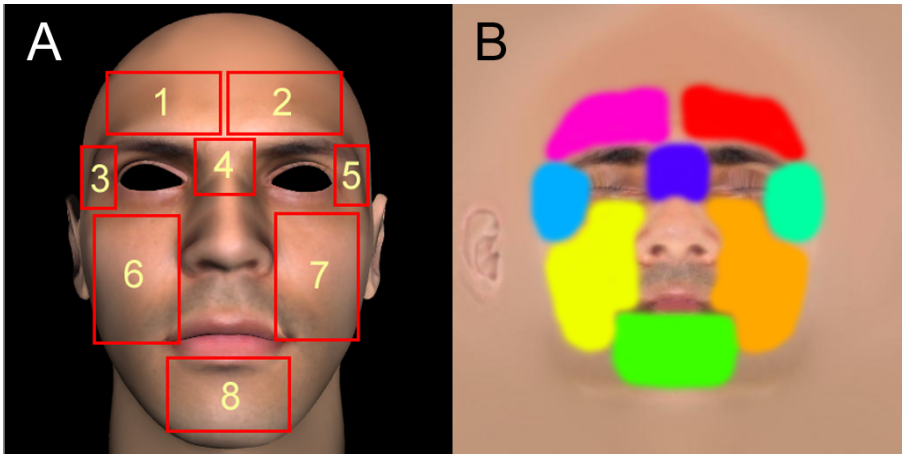

Fig. 4.5: Proposta de configuração das áreas de influência.

4.2.3.2 Funcionamento da técnica

Nas seções anteriores foram apresentadas as Áreas de Influência e o Vetor de Ativação, informações vitais para o funcionamento da técnica de aplicação de rugas em faces virtuais baseada em áreas de influência descritas por mapas de textura. O envio dessa informações à placa gráfica deve ser realizado com a mesma frequência que a informação da posição dos vértices, ou seja, a cada quadro da animação os dados necessários devem ser atualizados e reenviados.

Nesta seção são apresentados detalhes da implementação da técnica utilizando áreas de influência em texturas, sendo explicado como são geradas e organizadas as informações para o funcionamento da mesma, tais como: agrupamento das áreas de influência em uma textura 3D para facilitar o envio ao *shader*, o uso do Vetor de Ativação para envio das informações de porcentagem de exibição das rugas sobre determinada área de influência, os cálculos internos ao *shader* e as tecnologias utilizadas.

No protótipo desenvolvido, foi aplicado a técnica de *normal mapping* para a visualização das rugas, o que consequentemente torna necessária a inclusão de outras texturas, uma contendo a cor da pele do modelo facial, outra contendo o mapa de normais obtido a partir de um modelo de alta densidade de polígonos. No caso específico da técnica aqui apresentada, são necessários dois mapas de normais, um deles contendo as normais de uma face em expressão neutra, e outro contendo todas as rugas a serem trabalhadas no modelo facial. Estas texturas serão explicadas em mais detalhes na Seção 4.4, ainda neste capítulo.

As texturas contendo áreas de influência, no decorrer da execução, serão agrupadas em uma única textura 3D. Este agrupamento atende ao propósito de facilitar o envio das informações ao *shader*. No entanto, nada impede que o envio de várias texturas 2D seja utilizado ao invés de apenas uma textura 3D, porém a inclusão ou exclusão de uma área de influência não poderia mais ser realizada de forma dinâmica, ou seja, sem alterações internas ao *shader*. Em outras palavras, o agrupamento das áreas de influência atende um propósito de conveniência para deixar o *shader* mais dinâmico.

Foram adicionadas ao protótipo, funcionalidades para realizar o cadastro das áreas de influência,

fato que, por consequência, gera a adição de uma posição ao Vetor de Ativação. O total de áreas de influências adicionadas será igual à dimensão deste vetor, que deve ser associado de alguma forma, com a movimentação dos vértices do modelo geométrico. No caso deste trabalho, foi aplicada interpolação linear entre poses chave (Figura 4.6), e para o surgimento de rugas, foram associados a cada pose chave, os índices do Vetor de Ativação. Esses índices são modificados a medida em que ocorre a interpolação entre os modelos faciais.

Fig. 4.6: Poses chave utilizadas na interpolação do modelo facial. Da esquerda para a direita: expressão neutra, sobrancelha direita levantada, sobrancelha esquerda levantada, expressão de surpresa, compressão dos olhos.

No caso da expressão de levantamento da sobrancelha esquerda ou direita, apenas uma área de influência é ativada, então os índices respectivos recebem valores graduais durante a interpolação, causando o surgimento das rugas também de forma gradual, até o final da interpolação, onde as rugas estão 100% visíveis. Já na expressão de espanto e compressão dos olhos, duas áreas de influência foram utilizadas em cada expressão para exibir as rugas corretamente. No caso da expressão de surpresa , as áreas de influência da testa, e no caso da expressão de compressão dos olhos, as áreas de influência do canto dos olhos.

Uma vez que a associação entre as áreas de influência e as expressões é feita através do Vetor de Ativação, e as informações são enviadas ao *shader* a cada quadro da animação, o cálculo da visibilidade das rugas é realizado internamente a placa gráfica em duas etapas, no *vertex shader* e no *pixel shader*.

A etapa de *vertex shader* da técnica apresentada é responsável apenas por retransmitir as informações de posicionamento dos vértices à etapa de *pixel shader*. Nesta etapa, cada *pixel* da tela deve ser calculado, para cada quadro da animação, antes da exibição da imagem final.

Para cada *pixel*, o *normal mapping* é calculado duas vezes, uma vez utilizando o mapa de normais sem alteração, e outra vez utilizando o mapa de normais contendo as rugas definidas pelo usuário. Com isso obtêm-se as duas posições extremas possíveis para cada ponto da face, relaxada e enrugada, que serão interpoladas para obter o resultado final. Segue abaixo parte do código HLSL para o cálculo de cada *pixel*:

```
01 float4 PixelShader( VS_OUTPUT input ) : COLOR0
02 {
03     //Obtenho cor do pixel utilizando normal map neutro
04     ...
```

```
05     float4 finalColor1 = (diffuse + AmbientLightColor) * diffuseTexture + specular;
06
07     //Obtenho cor do pixel utilizando normal map com rugas
08     ...
09     float4 finalColor2 = (diffuse + AmbientLightColor) * diffuseTexture + specular;
10
11     //Calculo a visibilidade das rugas no pixel
12     finalColor2.a = 0.0;
13
14     //Percorro as áreas de influência
15     for (int i = 0; i < AllTexturesDepth; i++)
16     {
17        actualTexture = i;
18    float4 pixelAlpha =
19    tex3D(
20       AllTexturesSampler,
21            float3(
22               input.texCoord.x,
23               input.texCoord.y,
24               ((1.0 / AllTexturesDepth) * actualTexture) + 0.01
25            )
26        );
27
28        //Multiplico a visibilidade obtida pelo Vetor de Ativação da
29        // respectiva área, acumulando em finalColor2.a o resultado
30        finalColor2.a = finalColor2.a + (pixelAlpha.a * ativacao[i]);
31     };
32
33     //Interpolo a cor final do pixel utilizando como parâmetro o valor
34     // armazenado em finalColor2.a
35     float4 finalColor = lerp(finalColor1, finalColor2, finalColor2.a);
36
37     //Retorno a cor final para síntese da imagem
38     return finalColor;
39 }
```

Para a interpolação, é necessário calcular a porcentagem de visualização de cada *pixel*, utilizando a textura 3D e o Vetor de Ativação. Com uma estrutura de laço, são percorridas todas as camadas da textura 3D (Linha 15), o que dependendo da quantidade de áreas de influência pode impactar na performance da técnica, reduzindo a taxa de quadros por segundo da animação. O somatório dos *pixels* é multiplicado pelo Vetor de Ativação, o que resulta na porcentagem de visualização do *pixel* (Linha 30).

É realizada uma interpolação linear entre os dois valores previamente calculados para o *pixel*, obtidos com o primeiro e segundo mapas de normais, utilizando-se a porcentagem de visualização como parâmetro. O resultado desta interpolação será a cor final do *pixel* enviada à tela (Linha 35).

Após esse processo ser efetuado para todos os *pixels*, que depende das dimensões das texturas utilizadas, obtêm-se o modelo facial com rugas visíveis em pontos determinados pelo usuário, que podem variar de acordo com as configurações das áreas de influência e a forma como o programador implementou a conexão entre o Vetor de Ativação e a movimentação dos vértices propriamente dita.

4.3 Técnica baseada em vetores de direção das rugas

A segunda técnica desenvolvida utiliza vetores para representar a direção em que o deslocamento dos vértices do modelo facial produzirá rugas. Ao invés do usuário definir o momento de ativação das rugas, como é feito na Seção 4.2, a própria movimentação dos vértices realizada no modelo geométrico fornece as informações de posicionamento dos vértices necessárias para o cálculo de visualização das rugas, baseando-se no sentido e na amplitude do deslocamento de cada vértice. O processo de cálculo dessa amplitude de deslocamento é realizado internamente à placa gráfica, utilizando-se das informações de sentido das rugas, assim como o posicionamento original e atual dos vértices.

As texturas básicas utilizadas nesta técnica são as mesmas da técnica baseada em áreas de influência, ou seja, uma textura difusa e dois mapas de normais, no entanto, não são utilizadas texturas contendo áreas de influência. Essas texturas serão explicadas em detalhes na Seção 4.4.

O protótipo implementado foi desenvolvido em Java [4], em conjunto com a biblioteca OpenGL [5]. Para a programação da placa gráfica foi utilizada a linguagem GLSL [6]. A escolha dessas ferramentas justifica-se pela intenção de se utilizar tecnologias de uso livre, e por fornecer certa facilidade de se encontrar material de consulta na *internet*, tanto nas páginas dos desenvolvedores quanto em fóruns de discussão.

4.3.1 Motivação

Em ambas as técnicas apresentadas nesta dissertação, um dos focos principais é a exibição em tempo real de rugas. Com isso, o uso de vetores para determinar o sentido das rugas apresenta uma solução de baixo consumo de memória, se comparado com a técnica anterior (Seção 4.2), que utilizava texturas para passar as informações extras, e limitava a dimensão do arranjo de imagens de acordo com a placa gráfica utilizada.

O uso de Vetores Direção de Rugas, além de fornecer uma solução com baixo consumo de memória, proporciona a portabilidade entre diferentes técnicas de animação facial, uma vez definidos os vetores. Isso é possível pois todo o cálculo é realizado internamente à placa gráfica, diferente da técnica anterior (Seção 4.2) onde o programador é obrigado a desenvolver uma conexão entre o Vetor de Ativação e a animação realizada.

4.3.2 Vantagens e desvantagens

Para o correto funcionamento da técnica utilizando Vetores Direção de Rugas, são necessárias algumas informações extras passadas juntamente com os vértices, no caso, a posição original dos mesmos durante uma expressão facial relaxada, e os Vetores Direção de Rugas propriamente ditos.

[4] http://www.java.com (Acessado em 08/08/2009)
[5] http://www.opengl.org
[6] http://www.opengl.org/documentation/glsl/

Uma das vantagens desta técnica consiste na sua independência de processamento, uma vez que os dados para cálculo estejam todos presentes, em conjunto com os vértices. O que gera a capacidade de se integrar a técnica a outras implementações de animação facial, sem grandes alterações na lógica do sistema, além do envio das informações extras por vértice.

4.3.3 A técnica

4.3.3.1 Dados necessários ao funcionamento da técnica

Da mesma forma que na técnica anterior, são necessárias informações extras para que possam ser calculadas as rugas a serem apresentadas no modelo facial.

Além das informações necessárias apresentadas a seguir, também são necessárias a texturas difusa e mapas de normais, explicadas mais a frente na Seção 4.4. As outras informações exigidas são: Posição original dos vértices, Vetores de direção de rugas e Fator de expressão, descritas a seguir.

Posição original dos vértices:

Além do Vetor de Direção, para que se possa calcular o deslocamento propriamente dito, é necessária uma referência a um ponto inicial. Para esse fim é passada a posição original de cada vértice como informação extra, utilizando o modelo facial em expressão neutra como padrão, criando assim uma base de referência para se descobrir a amplitude dos deslocamentos ocorridos. Essa amplitude corresponde a distância entre o vértice original e o vértice deslocado, e é utilizada em conjunto com um ângulo pré-definido no *shader*, permitindo a suavização das rugas de acordo com deslocamentos laterais do vértice em relação ao eixo do Vetor Direção.

Vetores de direção de rugas:

Com o intuito de apresentar rugas no modelo facial de acordo com as expressões faciais realizadas, é proposto o uso de vetores de direção das rugas. Estes vetores indicam o sentido em que, havendo um deslocamento da malha geométrica, rugas serão apresentadas. É importante destacar que cada vértice deve conter seu Vetor de Direção correspondente, para que os cálculos fiquem independentes de áreas, e a apresentação das rugas na superfície do modelo resulte em uma transição suave.

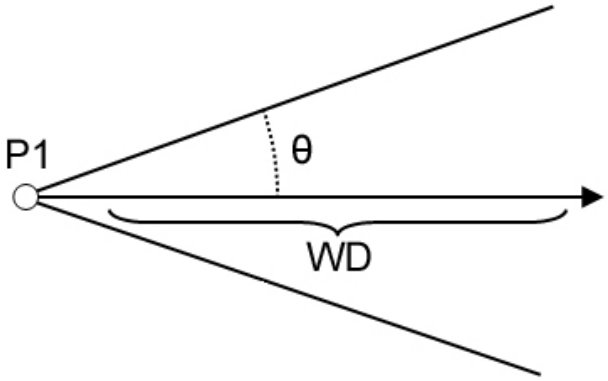

Fig. 4.7: Estrutura de um vetor de direção de rugas.

A estrutura de um Vetor Direção de Rugas (Figura 4.7) é composta de um ponto inicial (P1) que corresponde a posição do vértice no modelo geométrico em seu estado relaxado, ou seja, sem nenhuma expressão facial sendo executada. Outro elemento da estrutura é o vetor unitário (WD), que indica o sentido do deslocamento necessário para que ocorra a apresentação das rugas existentes nas coordenadas de textura correspondentes ao vértice trabalhado.

Além dessas duas componentes, ainda existe uma terceira, o ângulo θ (theta), que nos informa a área dentro da qual o deslocamento do vértice do modelo geométrico ainda é capaz de gerar rugas. No entanto, ao passo que um deslocamento de determinado vértice no sentido do Vetor Direção de Rugas resulta no aumento da visibilidade das rugas (Figura 4.8 - A), um deslocamento lateral, formando um ângulo maior que zero com o Vetor Direção de Rugas, fará com que a visibilidade dessa ruga seja atenuada, até o instante em que o ângulo formado desse deslocamento com o Vetor Direção de Rugas ultrapasse o ângulo θ (theta) definido pelo usuário, quando a visibilidade das rugas será igual a zero (Figura 4.8 - B).

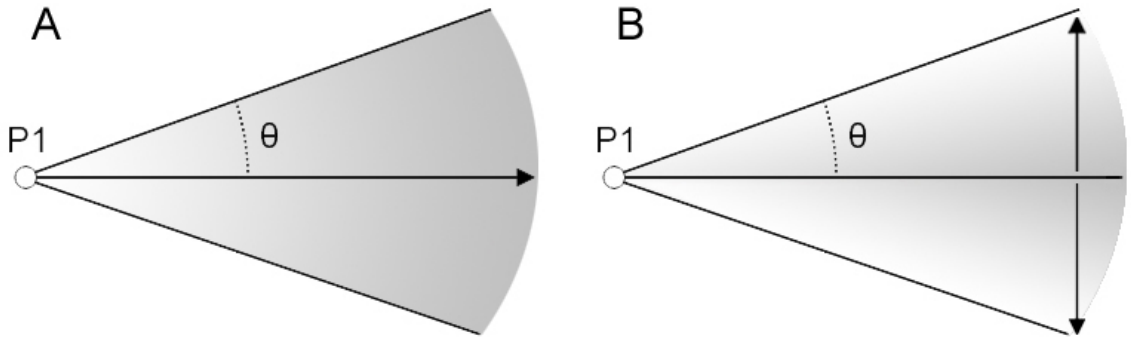

Fig. 4.8: Visibilidade das rugas de acordo com o deslocamento do vértice.

A Figura 4.9, representa o cálculo da porcentagem de visualização de rugas de acordo com o deslocamento dos vértices. No exemplo fornecido, o ponto inicial Pi é deslocado para dois pontos finais distintos: o primeiro ponto final (PF1) é deslocado para uma região interna ao cone formado

pelo Vetor Direção de Rugas e o ângulo θ (theta). Já o segundo ponto (PF2) é deslocado externamente a área delimitada pelo cone formado, não resultando na apresentação de rugas no modelo facial.

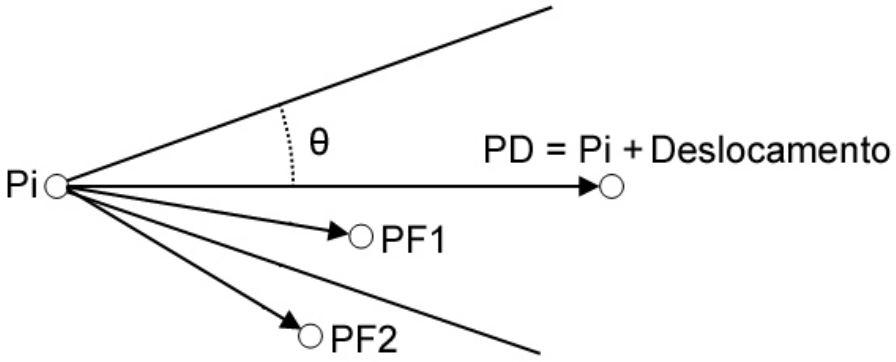

Fig. 4.9: Visibilidade das rugas de acordo com o deslocamento do vértice.

Qualquer deslocamento para uma posição interna ao cone resultará na apresentação de rugas no modelo facial, sendo que um deslocamento no sentido do Vetor Direção resultará em um aumento da visibilidade das rugas. Ao mesmo tempo, quanto maior o ângulo formado pelo vetor deslocamento em relação ao Vetor Direção, menor será a exibição dos detalhes faciais.

Todo o cálculo de apresentação das rugas é realizado internamente à placa gráfica pelo programa em *shader*, tornando o sistema portável entre diferentes implementações de animação facial. Este processamento será explicado na Seção 4.3.3.2.

Fator de expressão:

O fator de expressão é um valor passado a todos os vértices do modelo, utilizado como uma forma de suavizar a apresentação das rugas no modelo facial. A variável contendo o fator de expressão, internamente a placa gráfica, é utilizada para que seja suavizada a aparência das rugas. Em outras palavras, com esse fator de expressão é possível controlar o deslocamento necessário dos vértices para que sejam apresentadas as rugas.

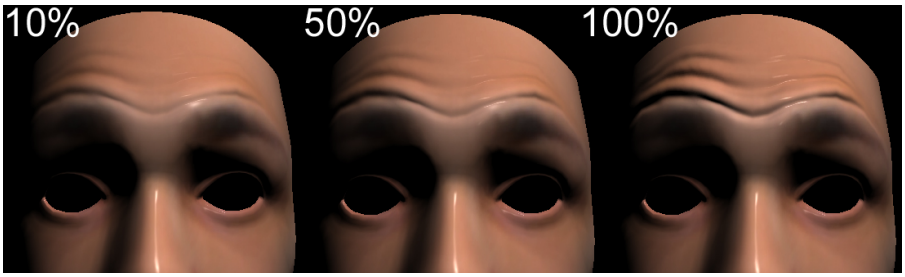

Fig. 4.10: Suavização da visibilidade das rugas pelo fator de expressão.

A Figura 4.10 demonstra a mesma expressão facial, consecutivamente exibindo as mesmas rugas, sendo suavizada pela variável contendo o fator de expressão. A esquerda é aplicado 10% de

visibilidade das rugas, no meio tem-se 50%, e na direita 100% de visibilidade.

4.3.3.2 Funcionamento da técnica

A técnica desenvolvida utiliza o processamento das placas gráficas para calcular a visibilidades das rugas utilizando o conceito de Vetores Direção de Rugas. São necessárias quatro etapas para que este cálculo seja realizado, a primeira ocorre antes do sistema ser iniciado, com a geração das informações necessárias como texturas, mapas de normais, que são descritas na Seção 4.4, e os Vetores Direção para o modelo tridimensional. No caso do protótipo implementado, foi desenvolvida uma ferramenta interna ao próprio sistema, para dar suporte à configuração dos Vetores Direção de Rugas.

A segunda etapa do funcionamento da técnica ocorre em tempo de execução, trata-se do envio das informações geradas à placa gráfica, o que dependendo das linguagens e tecnologias utilizadas pode ser feito de diferentes maneiras. No protótipo implementado foram criadas classes para dar suporte a todo este processo, sendo necessário realizar uma chamada as funções de envio das informações, repassando os dados.

A terceira etapa é realizada no *vertex shader*, onde é calculada a porcentagem de visibilidade propriamente dita. Isto é feito baseado na posição original e atual dos vértices, juntamente com o Vetor Direção de Rugas. Este cálculo será explicado em detalhes mais a frente nesta seção (Seção 4.3.3.2).

Finalmente, a quarta etapa é executada no *pixel shader*, onde utilizando o resultado da etapa de *vertex shader*, realiza-se uma interpolação entre os resultados de *normal mapping* da face relaxada e com rugas.

Ferramenta de suporte à configuração dos Vetores Direção de Rugas:

Os modelos geométricos devem ser modelados em outros *softwares* voltados para este propósito, o mesmo pode ser dito a respeito das texturas e mapas de normais. Quanto aos Vetores Direção de Rugas, no protótipo desenvolvido foi inclusa uma ferramenta para a definição dos mesmos, utilizando como base o deslocamento dos vértices. O motivo da inclusão desta ferramenta é simples: não existem ferramentas com este objetivo no mercado da mesma forma que existem para a geração de mapas de normais ou texturas.

No caso deste protótipo em particular, foi utilizada interpolação linear entre modelos geométricos com expressões faciais pré-modeladas. Não foi utilizada uma abordagem mais elaborada de animação facial, pois o objetivo principal deste trabalho consiste na técnica de apresentação de rugas faciais, e não na animação facial em si.

Para se definir os Vetores Direção das Rugas, foram criadas áreas para habilitar a seleção dos vértices de acordo com expressões faciais pré-determinadas. Na tela inicial de configuração do sistema, são escolhidas as áreas e qual modelo facial será utilizado na interpolação (Figura 4.11)

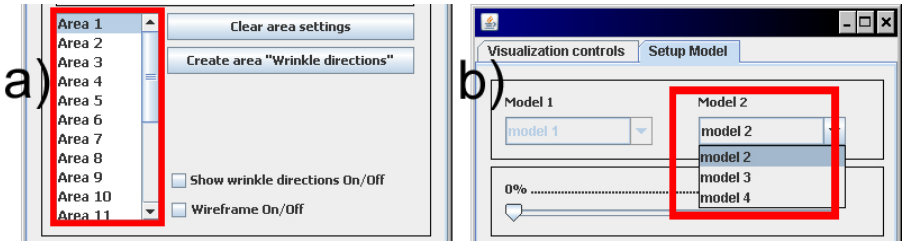

Fig. 4.11: Tela inicial de setup dos Vetores Direção de Rugas. A esquerda, a seleção de áreas (A), e a direita, a seleção do modelo facial utilizado na interpolação (B).

O modelo utilizado na interpolação deve ser escolhido de acordo com as rugas que o usuário deseje trabalhar. Uma vez selecionada uma área e o modelo a interpolar, sao selecionados no modelo geométrico os vértices onde o movimento dos mesmos, relativos a interpolação selecionada, irá gerar rugas (Figura 4.12).

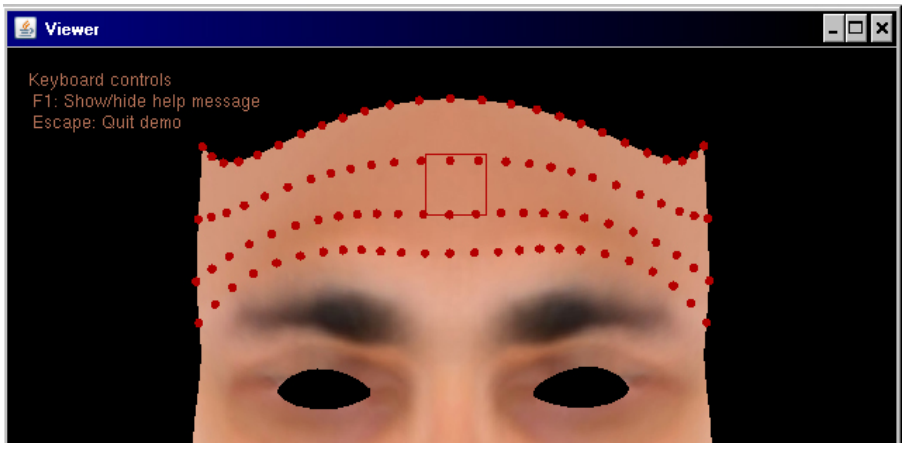

Fig. 4.12: Tela de seleção dos vértices por área escolhida.

Por exemplo, se o interesse for trabalhar uma expressão de surpresa, os vértices da testa deverão ser selecionados, pois nesta expressão são geradas rugas nesta região quando as sobrancelhas são levantadas. Para escolher o melhor modelo para esta etapa, é necessário levar em consideração a posição dos vértices no modelo selecionado em relação ao modelo com expressão relaxada, assim como as rugas a serem trabalhadas.

Os Vetores Direção de Rugas são gerados subtraindo a posição dos vértices do modelo facial relaxado dos vértices do modelo selecionado. O resultado deste cálculo é normalizado, obtendo o Vetor Direção de Rugas para cada vértice do modelo geométrico (Figura 4.13).

```
for (int i = 0; i < renderer.meshFinal.myModel.numVertex; i++) {
    if (renderer.meshFinal.extraArea[i] == renderer.selectedArea) {
        Vector3 vertexWrinkleDirection = new Vector3();

        vertexWrinkleDirection.setXYZ(
                renderer.mesh2.myModel.vertexPosition[i].x -
                   renderer.mesh1.myModel.vertexPosition[i].x,
                renderer.mesh2.myModel.vertexPosition[i].y -
                   renderer.mesh1.myModel.vertexPosition[i].y,
                renderer.mesh2.myModel.vertexPosition[i].z -
                   renderer.mesh1.myModel.vertexPosition[i].z);

        vertexWrinkleDirection.normalize();

        renderer.meshFinal.extraWrinkleDirection[i] = new Vector3(
                vertexWrinkleDirection.x,
                vertexWrinkleDirection.y,
                vertexWrinkleDirection.z);
    }
}
```

Fig. 4.13: Cálculo dos Vetores Direção de Rugas.

Se o usuário tiver necessidade de trabalhar com as rugas da testa e com os "pés-de-galinha" no canto dos olhos, basta selecionar os vértices da testa na área 1, alternar para a área 2, selecionar o modelo facial na expressão de olhos comprimidos e selecionar os vértices relativos a região do canto dos olhos, onde os pés-de-galinha são formados. Após isso, clicando em "Create wrinkle direction" os Vetores Direção dos vértices selecionados serão criados. Os vértices que não forem selecionados recebem o valor 0 (zero) para todas as componentes do vetor.

Após terminado este processo de criação dos Vetores Direção para o modelo facial, pode-se acessar o modo de visualização para verificar se os resultados são satisfatórios (Figura 4.14).

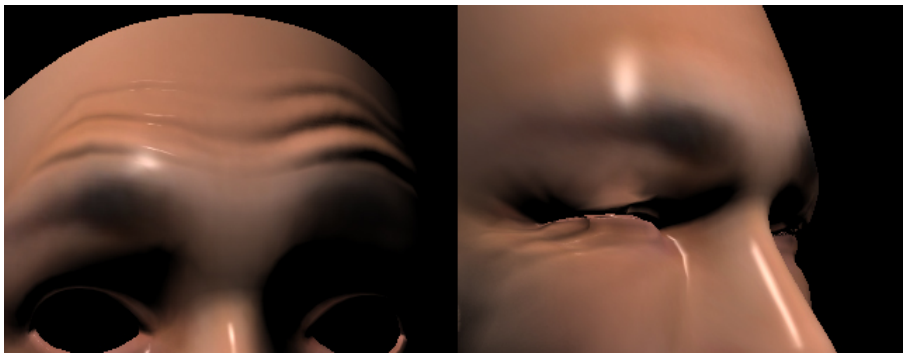

Fig. 4.14: Rugas formadas na testa (A), e no canto dos olhos (B).

Cálculo de exibição das rugas:

Após o cálculo dos Vetores Direção e do envio das informações ao *shader*, todo o cálculo de exibição de rugas é realizado internamente a placa gráfica durante o *vertex* e *pixel shader*. O algoritmo

abaixo demonstra como é feito o cálculo de visibilidade das rugas, na etapa de *vertex shader*:

```
01 início
02     ...
03     tipo vec3 = vetor[1..3] de real;
04     vec3 v1 <- wrinkleDirection;
05     vec3 v2 <- gl_Vertex.xyz - originalPosition.xyz;
06     real tetha <- vectorsAngle(v1, v2);
07     real maxAngle <- allowedAngle/100;
08
09     se (v1 != vec3(0.0, 0.0, 0.0)) então
10     início
11         se (1.0 - tetha < maxAngle) então
12        início
13             real angAtten <- angleAttenuation(maxAngle, tetha);
14             real dist <- 1 -
15                 (distance(gl_Vertex.xyz, originalPosition.xyz) *
16                     angAtten * expressionFactor);
17             dist <- clamp(dist, 0, 1);
18             compression <- dist;
19         fim
20  senão
21             compression <- 1.0;
22         fimse;
23     fim
24     senão
25         compression <- 1.0;
26     fimse;
27 fim.
```

Inicialmente é necessário calcular se o deslocamento do vértice está sendo feito para uma área interna do Vetor Direção. Para isso são necessários dois vetores: o Vetor Direção de Rugas do vértice respectivo (Linha 04), previamente enviado para a placa gráfica, e o vetor formado pelo deslocamento do vértice até sua posição original (Linha 05, informação também enviada durante a execução do sistema. Com esses dois vetores em mãos é possivel calcular o ângulo formado entre eles (Linha 06), utilizando uma função criada específicamente para isso.

O ângulo máximo permitido para um deslocamento lateral do vértice neste trabalho foi definido como uma constante de 20 graus, porém este valor poderia ser passado para o *shader* como uma variável acessível a todos os vértices, ou até mesmo de forma independente para cada vértice, aumentando as possibilidades de customização da apresentação das rugas. Como internamente ao shader os dados de ângulo estão sendo trabalhados entre 0 e 1, o ângulo máximo permitido é dividido por 100 para ficar no mesmo padrão (Linha 07).

Caso ocorra um deslocamento na posição do vértice (Linha 09), verifica-se se o ângulo entre os vetores é menor que o ângulo permitido. Caso o deslocamento estaja sendo realizado para dentro da área permitida, é calculado o índice de atenuação sofrido na visibilidade das rugas, devido ao deslocamento lateral dos vertices em relação ao eixo do Vetor Direção (Linha 13). Este índice de atenuação é obtido através de uma regra de três:

```
Ângulo_máximo <-> 1
Ângulo_atual  <-> Índice_atenuação

Índice_atenuação = (Ângulo_atual * 1) / Ângulo_máximo
```

Com esse índice de atenuação em mãos, é calculado o índice de deslocamento do vértice (Linha 14). Este deslocamento é obtido calculando-se a distância entre a posição do vértice atual e a posição original do mesmo, e multiplicando o resultado pelo índice de atenuação pelo ângulo, obtido anteriormente, e também pelo fator de expressão enviado pelo sistema ao *shader*. É realizado um enquadramento do índice de deslocamento para dentro dos limites de 0 e 1 (Linha 17), e o resultado é armazenado na variável *compression*, que é enviada para o *pixel shader*, e utilizada na interpolação entre o cálculo dos dos mapas de normais.

Na etapa de *pixel shader*, são calculados cada um dos *pixels* da imagem a ser sintetizada. Primeiramente são calculados os resultados de dois *normal mapping* para o *pixel* em questão, que foi a técnica de detalhamento de texturas escolhida neste trabalho. Após obtidos esses dois resultados, que são basicamente duas cores a serem exibidas na tela, é realizada uma interpolação linear entre as duas, utilizando como base de referência a variável *compression*, calculada na etapa anterior de acordo com o deslocamento dos vértices. Esta interpolação utiliza o seguinte cálculo para definir a cor final do *pixel*:

```
cor1 <- resultado_do_normal_mapping_neutro;
cor2 <- resultado_do_normal_mapping_com_rugas;

cor_final <- ((compression * cor1) + ((1-compression) * cor2));
```

Com esse cálculo, a medida em que ocorre um deslocamento do vértice, a variável *compression* é interpolada, e consequentemente a cor final do *pixel* trabalhado se altera, resultando em uma transição suave para o surgimento de rugas na superfície do modelo facial.

4.4 Texturas necessárias ao funcionamento das técnicas

Em ambas as técnicas propostas neste trabalho são utilizadas três texturas básicas, uma textura difusa e dois mapas de normais, um sem e outro com rugas. Estes mapas de normais são necessários para os cálculos de *normal mapping* (KILGARD, 2000), que no caso deste trabalho foi o *shader* escolhido para a apresentação das rugas na superfície do modelo facial.

- **Textura difusa:** Esta textura é responsável pela cor da pele do modelo geométrico. Normalmente pode ser gerada a partir de fotografias de faces reais e trabalhadas para que fiquem distribuídas sobre um plano, como se fosse um tecido a ser aplicado sobre o modelo facial. As posições dos elementos faciais presentes na imagem devem ser correspondentes às coordenadas de textura do modelo tridimensional (Figura 4.15), processo que pode ser realizado por exemplo em um software de manipulação 3D. No caso deste trabalho, os modelos faciais 3D foram criados com o software Autodesk MAYA [7].

[7] http://www.autodesk.com

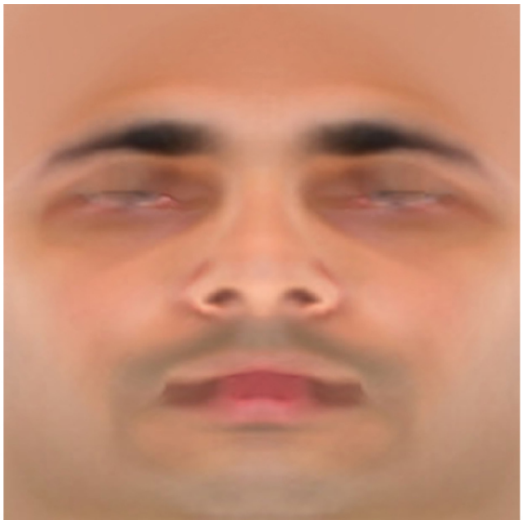

Fig. 4.15: Textura difusa.

- **Mapa de normal sem rugas:** O primeiro dos mapas de normais é uma imagem onde cada *pixel* é composto pelas componentes [X, Y, Z]={0.0, 0.0, 1.0}. Durante o processo de amostragem na etapa de *pixel shader*, cada *pixel* amostrado dessa imagem irá substituir a normal nos cálculos de iluminação.

  Esta textura, contendo os mapas de normais, será utilizada no cálculo do *normal mapping*, onde seu resultado será visível apenas nas áreas em que não estejam ocorrendo movimentação de vértices capazes de gerar rugas.

  Existem diversas formas de se criar essa textura. No caso deste trabalho, foram utilizados dois modelos geométricos, um sem a presença de rugas e outro com as seguintes rugas já modeladas em sua malha poligonal: rugas do canto dos olhos, também chamas de pés-de-galinha, rugas do meio dos olhos, rugas da testa e rugas das bochechas (Figura 4.16). Para se gerar o mapa de normal é necessário que todos os modelos tenham as coordenadas de textura já definidas. No caso do mapa de normal sem rugas, um modelo de alta densidade de polígonos sem a presença de rugas foi utilizado.

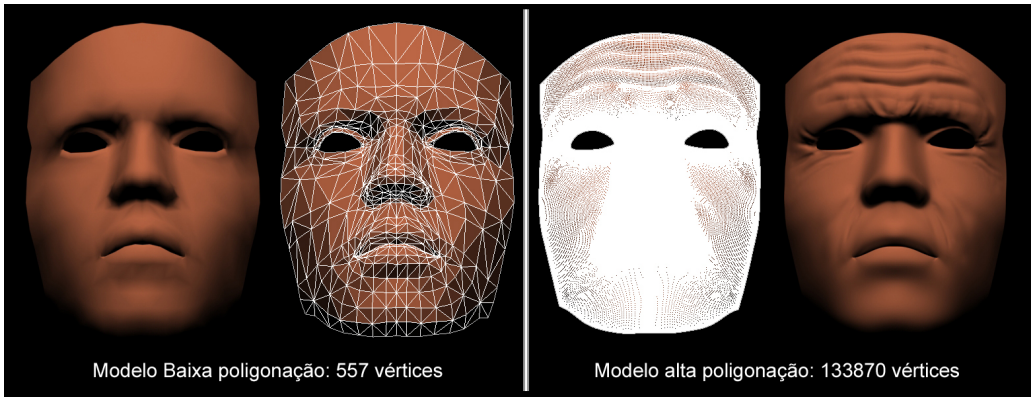

Fig. 4.16: Modelos geométricos utilizados para a geração dos mapas de normais. Modelos de baixa densidade de polígonos texturizado e aramado (esquerda) e modelos de alta densidade de polígonos com rugas aramado e texturizado (direita).

- **Mapa de normal com rugas:** O segundo mapa de normais é gerado da mesma forma que o primeiro. No entanto, ao invés de se utilizar o modelo sem a presença de rugas, utiliza-se um modelo com as rugas desejadas modeladas diretamente em sua malha 3D. As rugas escolhidas para serem aplicadas neste modelo são provenientes dos movimentos de levantamento das sobrancelhas, compressão do canto e meio dos olhos, movimento das bochechas, podendo existir outras, dependendo do interesse do usuário (Figura 4.17). Cada *pixel* amostrado desta textura contém o vetor normal mapeado equivalente ao encontrado no modelo geométrico de alta poligonação, na mesma coordenada de textura, conforme visto no Capítulo 3 (Figura 3.8).

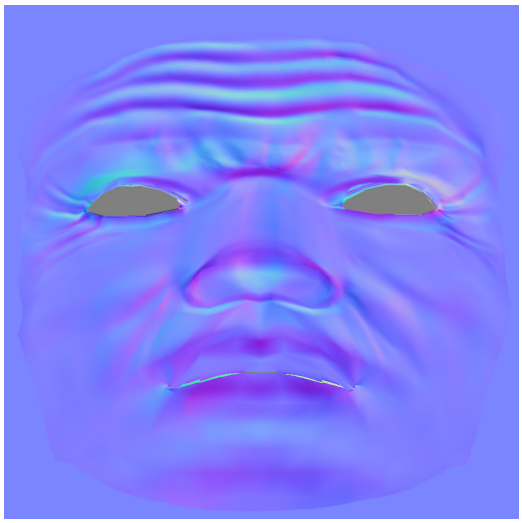

Fig. 4.17: Normal map com rugas.

4.5 Considerações finais

Neste capítulo foram descritas as duas soluções desenvolvidas neste trabalho para o problema de apresentação de rugas em faces virtuais, focando o uso em aplicações em tempo real. Foram apresentadas as texturas e outras informações necessárias para o funcionamento de ambas as técnicas, além de uma descrição do funcionamento das mesmas.

Para o desenvolvimento de ambas as soluções, foram levados em consideração os seguintes aspectos:

- **Velocidade:** Tendo por objetivo seu uso em tempo real, ou seja, com respostas imediatas aos comandos do usuário, as técnicas apresentadas neste trabalho não realizam a simulação das características físicas do tecido facial, já que este processo tende a ser oneroso do ponto de vista de processamento computacional. Um dos objetivos deste trabalho é desenvolver formas de se aumentar o nível de detalhes visuais em modelos faciais tridimensionais, especificamente rugas, sem a necessidade de se aumentar sua malha poligonal e principalmente, sem prejudicar a performance geral do sistema.
- **Flexibilidade:** Outro fator importante é a facilidade de integração das soluções aqui apresentadas em diferentes abordagens de animação facial. As placas gráficas têm acesso às informações dos modelos geométricos durante o processo de sintese da imagem, por esse motivo a utilização de *shaders* fornece uma forma de se adicionar pequenos detalhes visuais à superfície de modelos de baixa poligonação, sem que o sistema responsável pela animação facial seja alterado em sua lógica. Com o uso de *shaders* é possível deixar portável os cálculos de exibição de rugas em modelos faciais virtuais.

A primeira técnica apresentada neste trabalho, utilizando texturas como áreas de influência, provê uma forma relativamente simples de configurar quais rugas devem ser apresentadas, visto que as texturas podem ser geradas por sofwares de edição de imagens disponíveis nas distibuições mais comuns de sistemas operacionais (Windows, Linux, entre outros). Alguns destes softwares estando inclusive, sobre licença de código aberto. Apesar do fator limitador, com relação ao total de áreas de influência que podem ser utilizados simultaneamente, não são necessárias muitas áreas para se representar os diversos grupos de rugas faciais. Tudo depende do nível de detalhes que o programador deseja alcançar.

Já a segunda técnica apresentada, baseada no uso de Vetores Direção de Rugas, permite uma separação da animação facial e do cálculo de apresentação de rugas, visto que configurados e inseridos os dados necessários juntamente com os vértices, todo o cálculo para apresentação das rugas é realizado internamente à placa gráfica.

Ambas as técnicas desenvolvidas neste trabalho poderiam ser associadas ao Grupo 4 (Seção 2.6), visto que não é realizada alteração da geometria, nem são feitos cálculos de simulação para a apresentação das rugas sobre o modelo facial. A exibição das rugas é feita por uma reprodução do efeito visual, obtido por meio do uso de *shaders*.

Capítulo 5

Avaliação das técnicas e protótipos desenvolvidos

Neste capítulo é apresentada uma avaliação das técnicas desenvolvidas nesta dissertação, descritas nas Seções 4.2 e 4.3. Na Seção 5.1 é detalhada a metodologia de avaliação adotada, incluindo a descrição dos equipamentos utilizados para as medidas de desempenho, assim como os modelos faciais que compuseram os casos de teste.

Na Seção 5.2, são exibidos os resultados das avaliações, com quadros de desempenho para as técnicas desenvolvidas, com diferentes configurações como, o número de áreas de influência enviadas à placa gráfica para a técnica da Seção 4.2 e a quantidade de vértices nos modelos enviados à placa gráfica para a técnica da Seção 4.3.

Por fim, são apresentadas as conclusões a respeito dos resultados obtidos e metas alcançadas.

5.1 Metodologia de avaliação

Para a avaliação das técnicas, foi levado em consideração que as mesmas deveriam ter a capacidade de trabalhar em tempo real. Para isso, foram realizados testes de desempenho, com diferentes configurações do sistema, como por exemplo, a quantidade de áreas de influência passadas ao *shader*, e o número de vértices dos modelos faciais trabalhados.

Em cada uma das técnicas, foram gerados 1000 (mil) quadros, sendo esse valor utilizado para o cálculo das médias, dividindo-se o tempo total gasto (ms) por 1000, obtendo assim o tempo médio em quadros/ms. Para as técnicas das Seções 4.2 e 4.3, foram calculados a cada quadro da animação, a interpolação da posição dos vértices, e recalculados os vetores nomal, tangente e binormal de cada vértice dos modelos faciais dos casos de teste.

No contexto deste trabalho, entende-se como execução em tempo de real uma taxa, em quadros por segundo, de síntese e apresentação das rugas suficiente para transmitir a sensação de evolução suave da animação, sem que processo de síntese e apresentação individual dos quadros seja perceptível

a um observador humano. É considerado que uma taxa de síntese e apresentação acima de 24 quadros por segundos (24 FPS) corresponde à execução em tempo real.

5.1.1 Equipamento utilizado

Para os testes de desempenho das técnicas foi utilizado um computador portátil (*Notebook*).

Para detectar a configuração do computador, foi usado um aplicativo específico para este fim [1], que informa as configurações de processador, placa gráfica e memória, além de vários outros dados. Na Tabela 5.1 é apresentada a configuração do computador.

Notebook Hewlett-Packard Presario V6000 (Modelo: V6210BR)	
Processador:	AMD Mobile Sempron 3500+
Clock processador:	1808.0 MHz
Memória RAM:	2048 MBytes (2 x 1024 MBytes)
Clock da memória:	667 MHz (DDR2-SDRAM)
Placa gráfica:	nVidia GeForce Go 6150 (C51) + MCP51
Memória da placa gráfica:	64 MBytes SDRAM Compartilhada

Tab. 5.1: Configuração computador portátil.

Como pode ser observado, o computador utilizado nos testes não é de última geração em relação aos disponíveis atualmente no mercado.

5.1.2 Expressões faciais

No desenvolvimento dessa dissertação foram gerados quatro modelos faciais, cada um deles com 2172 vértices, pré-modelados em programas específicos [2], que posteriormente foram interpolados linearmente entre si para obter animações faciais. A partir desses modelos foram criadas cópias com número de vértices variando de 561 a 5.151, para o estabelecimento dos casos de teste. Procurou-se enfatizar nas técnicas, as expressões em que o maior acúmulo de rugas fossem apresentadas. Na figura 5.1 estão as expressões escolhidas no trabalho, sem adição de textura.

[1] HWiNFO32: http://www.hwinfo.com/ (Acessado em 02/04/2010)
[2] MAYA: http://www.autodesk.com (Acessado em 27/12/2008)

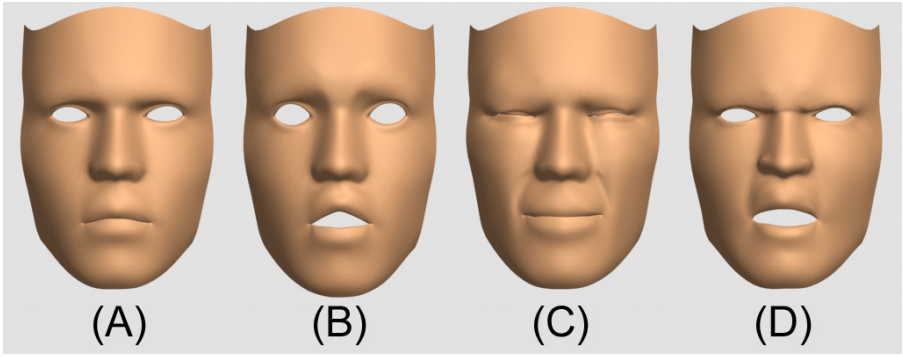

Fig. 5.1: Expressões utilizadas na avaliação do sistema.

A Figura 5.2 apresenta as mesmas quatro expressões trabalhadas nessa dissertação, como demonstrado na Figura 5.1, com textura difusa e mapas de normais aplicados, utilizando uma abordagem de *normal mapping*. É possivel observar o surgimento das rugas nas áreas onde a malha do modelo facial é comprimida.

Fig. 5.2: Expressões utilizadas na avaliação do sistema com textura difusa e *normal mapping* aplicados.

- **Expressão 1** (Figura 5.1-A): A expressão neutra é utilizada como referência para o cálculo do surgimento de rugas, onde a posição dos seus vértices determinam um estado relaxado da face, não apresentando rugas.
- **Expressão 2** (Figura 5.1-B): Esta expressão enfatiza o surgimento de rugas na testa do modelo facial, ocasionado pelo levantar das sobrancelhas, que comprimem a área da testa.
- **Expressão 3** (Figura 5.1-C): A compressão dos olhos é responsável pela ocorrência das rugas laterais à face, comumente conhecidas como "pé-de-galinha". Em alguns casos, como o de pessoas obesas ou idosas, o ato de comprimir os olhos pode gerar o acúmulo de tecido entre os olhos, resultando no surgimento de rugas nesta região. Nesta expressão também foi adicionado um pequeno sorriso, para que as rugas inferiores aos olhos pudessem ser demonstradas.
- **Expressão 4** (Figura 5.1-D): O ato de franzir a testa resulta no aparecimento de rugas na região central dos olhos, devido a compressão do tecido nessa região.

5.2 Resultados das avaliações

5.2.1 Técnica baseada em áreas de influência descritas por mapas de textura (Seção 4.2)

Utilizando as configurações do sistema descritas na Seção 5.1, foram realizados testes com diferentes números de áreas de influência, a fim de determinar a viabilidade da técnica para se trabalhar em tempo real.

O número de áreas de influência utilizadas nas amostragens foi alterado em cada teste, variando de 1 até 16 áreas de influência sendo trabalhadas. Lembrando que 16 áreas foi o limite máximo suportado pelo equipamento utilizado nos testes. Os modelos faciais utilizados nestes testes possuem 2172 vértices cada.

O uso de áreas de influência descritas por mapas de textura (Seção 4.2) apresentou uma solução onde cada uma das áreas pode ser definida por meio de programas de edição de imagens disponíveis na maioria dos sistemas operacionais. Exemplos: *Paint* (*Windows*) e *Gimp* (*Linux*).

A Tabela 5.2 apresenta os resultados obtidos com a técnica, realizando interpolação linear da posição dos vértices, assim como o cálculo das normais e tangentes para cada um deles existente no modelo geométrico a cada quadro da animação

Áreas de inf.	T. médio quadro(ms)	FPS médio	FPS mín	FPS máx	Desvio padrão
1	17.4	58.74	30.87	65.98	18.54
2	17.9	59.32	30.87	64.60	18.14
3	18.6	59.45	29.11	65.82	19.62
4	19.8	41.60	10.67	61.76	25.73
5	19.8	39.88	9.98	64.16	27.14
6	21.3	37.47	22.69	60.39	19.00
7	22.2	34.73	8.68	62.62	26.98
8	22.9	35.86	8.84	61.63	26.40
9	24.2	32.74	30.29	36.24	2.99
10	24.6	31.44	29.34	33.21	1.94
11	25.5	30.27	8.26	61.37	26.68
12	26.3	28.86	7.86	61.70	27.14
13	26.7	28.34	26.99	30.11	1.56
14	28.3	27.52	26.38	28.68	1.15
15	29.1	26.67	8.26	60.38	26.43
16	29.9	27.06	24.98	29.78	2.41

Tab. 5.2: Desempenho da técnica descrita na Seção 4.2.

Observa-se que a taxa média de quadros por segundo (*FPS*) diminui praticamente pela metade quando o número de áreas de influência cresce de 1 para 16, caindo de 57,42 *FPS* para 33,42 *FPS*.

Já o tempo médio do quadro sobe de 17,4 ms para 29,9 ms. É importante lembrar que cada textura adicionada ao sistema precisa ser percorrida por completa a cada quadro, o que significa que para cada área adicionada, o sistema consequentemente vai necessitar de um maior processamento. A tabela 5.2 também mostra o *FPS* mínimo e máximo obtidos em cada teste, assim como o desvio padrão[3] para os valores de *FPS*.

Com base nos resultados obtidos em cada os testes da Tabela 5.2, pode-se observar que apesar da técnica apresentar resultados de performance, que permitem a sua execução em tempo real com poucas áreas de influência, o acréscimo de mais áreas diminui a performance geral da animação facial. No entanto, mesmo acrescido de 16 áreas de influência, o sistema manteve a capacidade de ser executado a uma taxa de quadros por segundo interativa.

Uma melhor visualização dos resultados de performance pode ser vista no gráfico da Figura 5.3, onde a Tabela 5.2 é representada.

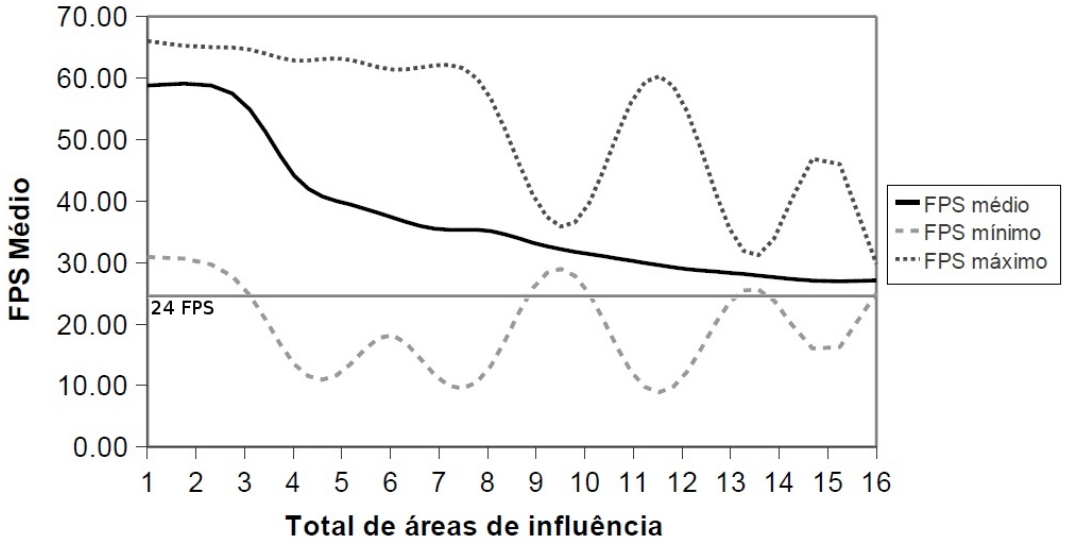

Fig. 5.3: Comparativo entre FPS médio pelo número de áreas de influência utilizadas, para a técnica da Seção 4.2, considerando as configurações propostas na Seção 5.1.

Observando o gráfico da Figura 5.3, pode-se verificar uma queda de performance a cada área de influência adicionada aos cálculos, e mesmo com 16 áreas trabalhadas, máximo suportado pelas placas gráficas utilizadas, o sistema é executado em tempo real a uma taxa média de 33 quadros por segundo.

5.2.2 Técnica baseada em Vetores Direção de Rugas (Seção 4.3)

Para a técnica descrita na Seção 4.3, os mesmos testes de performance da Seção 5.1 foram realizados. Como nesta técnica as áreas de influência não são descritas por mapas de textura, mas por

[3]Desvio padrão: http://www.inf.furb.br/sias/saude/Textos/desvio_padrao.htm

Vetores Direção de Rugas, enviados a placa gráfica juntamente com as informações dos vértice, os testes de performance foram baseados na quantidade de cálculos necessários em cada vértice para a apresentação de rugas.

Visto que os cálculos são realizados por vértice, foram gerados modelos faciais com diferentes configurações, para determinar a influência que o aumento do número de vértices nos modelos tem na performance da técnica, e se o mesmo é capaz de trabalhar em tempo real, mesmo com o consequente aumento no processamento.

A técnica da Seção 4.3 utiliza vetores aplicados sobre cada um dos vértices, na forma de informações extras. Estes vetores servem para indicar o sentido em que o deslocamento facial produz rugas sobre a superfície do modelo virtual. A solução encontrada no uso de vetores, resultou em uma forma flexível de se configurar o surgimento de rugas, não sendo necessário uma associação com elementos internos das abordagens de animação facial.

Ao passo que na técnica da Seção 4.2 cada uma das áreas de influência é associada a um elemento do sistema de animação facial, como por exemplo, a força aplicada de um músculo virtual sobre os vértices da face, ou o índice de interpolação entre duas expressões faciais, como realizado neste trabalho, na técnica da Seção 4.3, o surgimento das rugas é influenciado pela movimentação dos vértices no espaço tridimensional, independente do método de animação facial utilizado, sendo necessário apenas que o sistema onde a técnica será implementada, realize o envio das informações de referência, ou seja, os Vetores Direção de Rugas e posição inicial dos vértices.

Na Tabela 5.3 é apresentado o resultado dos testes de performance, sendo calculada a interpolação entre o posicionamento dos vértices das expressões e o cálculo de normais e tangentes por vértice a cada quadro da animação. Nos testes foram utilizados modelos geométricos com 561 a 5.151 vértices, para determinar a influência do número de vértices sobre a performance da animação, expressada em quadros por segundo (*FPS*).

Num. vért.	T médio quadro(ms)	FPS médio	FPS mín	FPS máx	Desvio padrão
561	16.06	62.26	31.25	66.67	19.30
1071	16.84	59.37	10.64	66.66	30.46
1581	19.00	52.63	10.63	66.66	29.16
2091	18.97	52.72	10.63	66.66	29.17
2601	21.37	46.79	9.09	66.66	29.24
3111	23.25	43.01	9.09	66.66	28.94
3621	29.22	43.23	8.20	66.66	29.42
4131	32.84	30.45	8.00	66.66	29.60
4641	37.34	26.78	7.09	66.66	30.35
5151	40.06	24.96	3.12	32.26	15.16

Tab. 5.3: Desempenho da técnica descrita na Seção 4.3.

Como pode ser observado, nos testes o sistema descreve uma queda de performance a medida em que o número de vértices presentes nos modelos faciais aumenta, iniciando a 62,26 *FPS* no modelo

com 561 vertices, decaindo até 24,96 *FPS* no modelo de 5.151 vértices, e tempo médio por quadro variando de 16,06 ms a 40,06 ms, o que representa praticamente 1/3 da performance inicial. Também são apresentados na tabela 5.3 os resultados de *FPS* mínimo e máximo obtidos em cada teste, e o desvio padrão[4] respectivo as taxas de *FPS* obtidas.

O gráfico da Figura 5.4 expressa visualmente essa queda de performance de acordo com o aumento do número de vértices, e consequente aumento do volume de cálculos por quadro da animação.

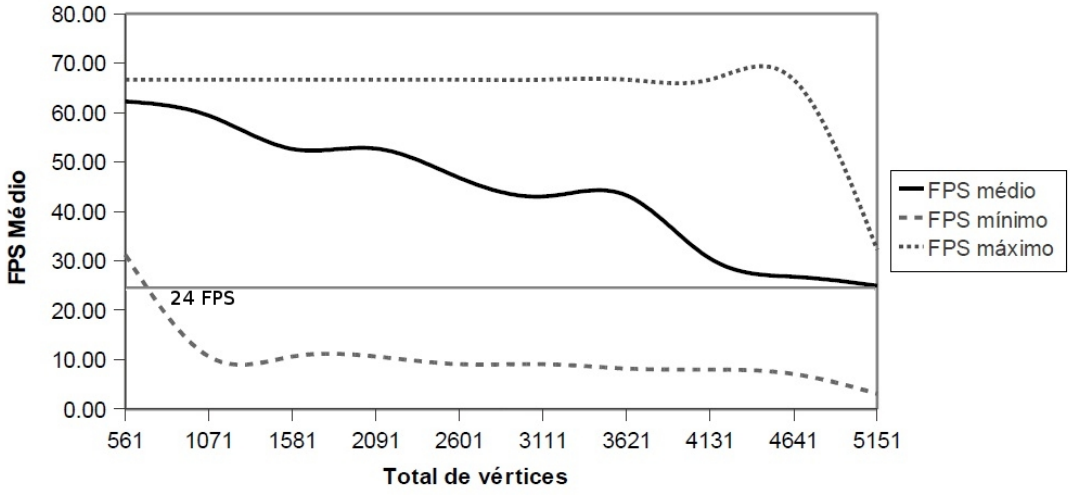

Fig. 5.4: Comparativo entre FPS médio pelo número de vértices nos modelos geométricos utilizados, para a técnica da Seção 4.3, trabalhando com as configurações propostas na Seção 5.1.

Mesmo com o processamento exigido para os cálculos, devido ao número elevado de vértices, o sistema ainda é capaz de trabalhar em tempo real. É interessante lembrar que dificilmente modelos faciais com muitos vérticies permitem sua execução em tempo real, sendo necessário a aplicação de técnicas como por exemplo, o uso de *shaders*, adicionando detalhes a modelos com baixa densidade de polígonos, utilizando mapas de normais, cálculos de iluminação, tonalização, entre outros, apresentando um resultado visual em modelos de baixa densidade de polígonos similar ao resultado expresso em modelos com alta densidade de polígonos.

5.2.3 Comparativo das técnicas

As duas técnicas desenvolvidas nesta dissertação, diferem em como o cálculo da apresentação de rugas é realizado. A técnica da Seção 4.2 utiliza texturas como áreas de influência, e concentra os cálculos de apresentação na etapa de *pixel shader* da placa gráfica. Já a técnica da Seção 4.3 utiliza informações extras em cada vértice para realizar os cálculos de apresentação de rugas, ficando os cálculos concentrados na etapa de *vertex shader*.

Para permitir a comparação entre as duas técnicas, primeiramente foi definida uma configuração

[4]Desvio padrão: http://www.inf.furb.br/sias/saude/Textos/desvio_padrao.htm

onde ambas as técnicas fossem semelhantes. Para a técnica da Seção 4.2 foram definidas 8 áreas de influência, como apresentado na Figura 4.5, de forma que toda a área da face fique sobre a atuação de pelo menos uma área de influência. No caso da técnica da Seção 4.3, cada área do modelo facial está sob atuação dos Vetores Direção de Rugas, que são enviados juntamente com cada vértice para a placa gráfica.

Com essas configurações, as duas técnicas apresentam uma configuração semelhante para exibição de rugas. Para realizar a coleta das informações de performance entre as mesmas, foram criados modelos geométricos com número de vértices variando de 561 até 5.151. O resultado dos testes expressa como ambas as técnicas se comportaram de acordo com o aumento do número de vértices.

Na Tabela 5.4 é apresentado um comparativo de desempenho entre as duas técnicas (4.2 e 4.3), variando o número de vértices dos modelos faciais.

Vértices	**Tempo quadro (ms) T1**	**Tempo quadro (ms) T2**	**FPS T1**	**FPS T2**
561	16,6	16,1	60,28	62,13
1071	16,6	16,8	60,27	59,48
1581	16,9	17,8	59,09	56,24
2091	17,8	18,8	56,33	53,15
2601	19,5	21,1	51,26	47,40
3111	21,4	22,9	46,77	43,59
3621	23,5	29,9	42,47	33,47
4131	25,6	32,3	39,13	30,97
4641	28,1	36,8	35,64	27,18
5151	30,0	40,0	33,29	24,98

Tab. 5.4: Resultado comparativo de performance entre as técnicas T1 (Seção 4.2) e T2 (Seção 4.3).

Pode ser observado que ambas as técnicas possuem performance para trabalhar em tempo real, mesmo com número elevado de vértices, como pode visto nos modelos com 5.151 vértices onde a performance da técnica da Seção 4.2 se mantém em 33,29 FPS, e a da técnica descrita na Seção 4.3 em 24,98 FPS.

Como é mostrado na Tabela 5.4, a técnica com Vetores Direção de Rugas (Seção 4.3), perde em performance para a técnica com áreas de influência descritas por mapas de textura (Seção 4.2). Isso ocorre pois além dos cálculos realizados em ambas as técnicas, interpolação da posição dos vértices e cálculo de normais e tangentes, a técnica da Seção 4.3 realiza cálculos para determinar o deslocamento e visibilidade das rugas para cada vértice do modelo facial, portanto quanto mais vértices o modelo possui, mais cálculos são necessários para se obter as normais, tangentes e índices de visibilidade para cada vértice.

Essa perda de performance, mostrada na Tabela 5.4, pode ser visualizada no gráfico da Figura 5.5.

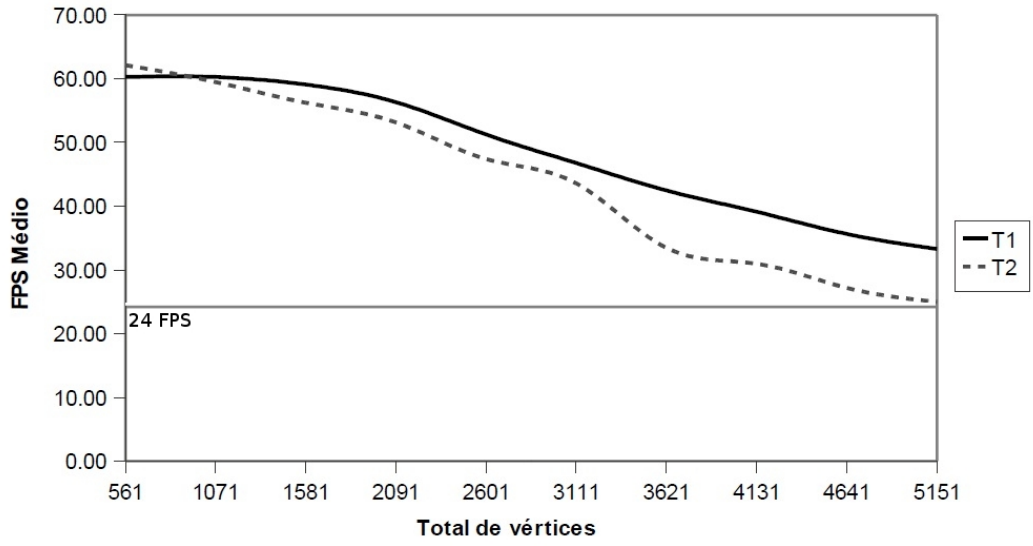

Fig. 5.5: Gráfico comparativo de performance entre as técnicas das Seções 4.2 (T1) e 4.3 (T2).

É importante enfatizar que os cálculos de interpolação do posicionamento dos vértices e cálculos de normais e tangentes, são realizados no sistema de animação facial, antes de serem enviados à placa gráfica, onde é calculada a apresentação das rugas sobre a malha do modelo facial.

Mesmo com a perda de processamento, ambas as técnicas são capazes de trabalhar em tempo real, obtendo com o modelo de maior número de vértices (5.151 vértices) uma performance de 33,29 FPS para a técnica da Seção 4.2 e 24,98 FPS para a técnica da Seção 4.3, não sendo preciso utilizar equipamentos de última geração.

Na Figura 5.6 é apresentado um comparativo entre as duas técnicas desenvolvidas nesta dissertação. São geradas rugas na testa e canto dos olhos, utilizando uma área de influência comum entre ambas.

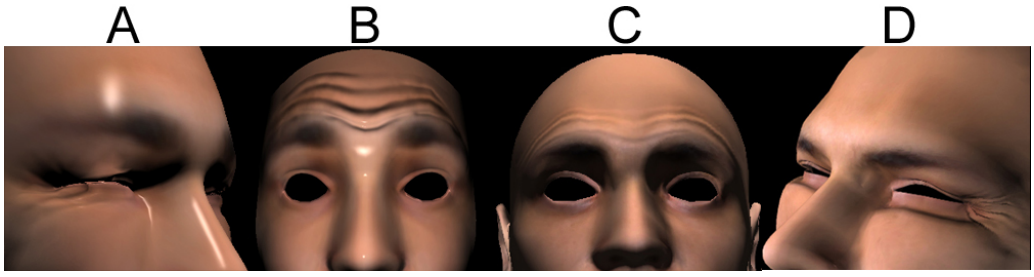

Fig. 5.6: Comparativo das rugas geradas com as técnicas da Seção 4.2 (C e D) e 4.3 (A e B).

Pode ser observado que em ambas as técnicas são capazes de apresentar rugas durante o processo de animação facial dentro da área de influência delimitada pelo usuário, o que na técnica da Seção 4.2 é feito por meio de texturas, e na técnica da Seção 4.3 é feito com vetores indicando o sentido do deslocamento para a formação de rugas.

No entanto, para a técnica da Seção 4.2 existe o limite imposto pelo equipamento utilizado e memória disponível para processamento gráfico, sendo que nos equipamentos utilizados esse limite foi de 16 texturas com dimensões 512x512 pixels, comprimidas em uma única textura 3D.

A técnica da Seção 4.3 é capaz de definir um vetor para cada vértice, tornando mais flexível a configuração do surgimento das rugas, como pode ser visto na figura 5.7 onde são definidos os Vetores Direção de Rugas para 4 vértices independentes. Para realizar a mesma tarefa com a técnica da Seção 4.2 são necessárias 4 áreas de influência definidas por meio de texturas.

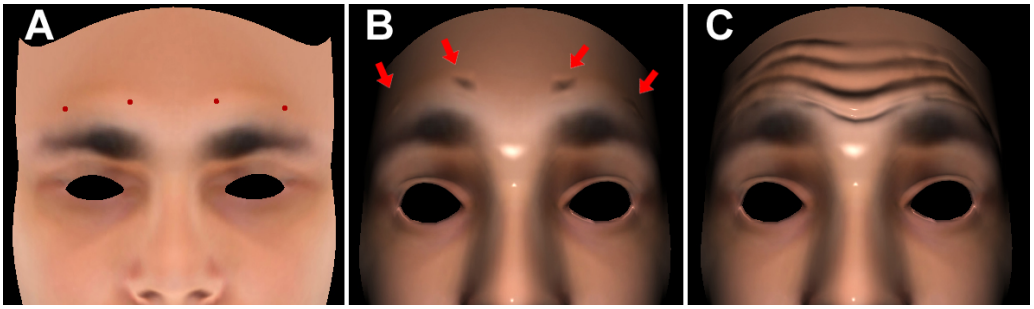

Fig. 5.7: Técnica da Seção 4.3 com Vetor Direção de Rugas definido sobre quatro vértices.

Na figura 5.7 são declarados 4 Vetores Direção de Rugas, trabalhando de forma independente entre si. Pode ser visto em 5.7-A os 4 vértices declarados, em 5.7-B esses as rugas sendo apresentadas na área delimitada por esses 4 vertices. É importante falar que estas áreas apresentam as rugas de forma independente, ou seja, se apenas um lado da testa for levantado, apenas as rugas desse lado serão exibidas. Em 5.7-C são apresentadas todas as rugas definidas na testa para fins de comparação.

No entanto, apesar da desvantagem em termos de definição da exibição de rugas, a técnica da Seção 4.2 apresenta melhores resultados de performance do que a técnica 4.3, quando trabalhando com poucas áreas de influência. Para modelos onde diversas áreas de influência devem ser definidas, ou uma sistema onde a apresentação das rugas deve ser feita independentemente em cada vértice, a técnicada Seção 4.3 representa uma melhor escolha.

5.2.4 Portabilidade das técnicas

Na área de informática, entende-se o termo portabilidade como a capacidade de um programa ser compilado ou executado em diferentes plataformas e hardwares [5].

Um dos objetivos desta dissertação era manter possível a portabilidade das técnicas desenvolvidas, entre diferentes abordagens de animação facial, como por exemplo, sistemas de animação facial utilizando músculos virtuais, interpolação linear, sistemas massa-mola, entre outros. Também sendo possível trabalhar com soluções híbridas, como por exemplo, uma abordagem de músculos virtuais que utilize um sistema massa-mola para calcular a posição dos vértices do modelo facial.

[5]http://www.babylon.com/definition/portabilidade/Portuguese

A solução encontrada para deixar as técnicas portáveis foi o uso de *shaders*, fazendo com que os cálculos da apresentação de rugas sejam realizados internamente a placa gráfica, deixando os cálculos de apresentação de rugas o mais independente possível do sistema de animação facial. As plataformas atuais de desenvolvimento gráfico, tanto de *hardware* quanto *software*, possuem suporte ao desenvolvimento com essa tecnologia.

Para a técnica da Seção 4.2, além das informações de texturas, mapas de normais, e as áreas de influência agrupadas na forma de uma textura 3D, é necessário que seja enviado ao *shader* um "Vetor de Ativação", contendo a porcentagem de ativação de cada área de influência.

No presente trabalho, foram realizados cálculos de interpolação linear entre quatro expressões faciais modeladas previamente, onde cada expressão era associada a visibilidade das rugas sobre determinada área de influência. Isto é feito associando a interpolação de um modelo facial a um ou mais índices do Vetor de Ativação, por exemplo: a interpolação entre a expressão 1 (Figura 5.2-A) para a expressão 2 (Figura 5.2-B). Enquanto o modelo facial da expressão 1 estiver sendo apresentado, os índices do Vetor de Ativação serão iguais a 0. A medida em que a interpolação do posicionamento dos vértices para transformar o modelo facial da expressão 1 no modelo facial da expressão 2 ocorre, os índices do Vetor de Ativação correspondentes às áreas da testa vão gradualmente recebendo valores entre 0 e 1, consequentemente apresentando as rugas. No momento em que a interpolação estiver concluída e o modelo facial da expressão 2 estiver sendo ap resentado, o valor contido nos índices do Vetor de Ativação, correspondentes às áreas de influência da testa, será igual a 1.

Se a técnica for integrada para uma abordagem de animação facial utilizando músculos virtuais, a associação entre a técnica e a animação facial será feita trabalhando o elemento ativo da abordagem, que no caso são os músculos virtuais. O músculo relaxado corresponde a um valor de ativação 0 no índice do Vetor de Ativação correspondente ao músculo. Da mesma forma, a contração total do músculo corresponde a um valor de ativação de 1 no índice do Vetor de Ativação.

Para outras abordagem de animação facial, a forma de se popular esse Vetor de Ativação será alterada, associando seus índices com quaisquer elementos necessários para o funcionamento. Isso permite que a técnica seja integrada em outras abordagens, mesmo sendo necessário o preenchimento de um vetor com índices de ativação.

No caso da técnica da Seção 4.3, além das informações básicas como texturas e vértices do modelo facial, são enviadas as informações de Vetor Direção de Rugas e posição inicial de cada vértice do modelo geométrico.

Essas informações são definidas pelo usuário previamente, e armazenadas em um arquivo texto da mesma forma como é feito com as informações de vértices, normais e coordenadas de texturas dos modelos tridimensionais. No caso dessa dissertaçao foram desenvolvidas ferramentas visuais para gerar esses Vetores Direção de Rugas. Durante a execução do sistema de animação facial, os Vetores Direção e posições originais dos vértices são carregados para a memória do computador, assim como o modelo facial, que é enviado ao *shader* juntamente com as informações extras.

Todos os cálculos de apresentação de rugas são realizados internamente ao *shader*, permitindo que os mesmos sejam executados independente do sistema de animação facial. Diferente da técnica da Seção 4.2, que necessita de um Vetor de Ativação associando os elementos ativos da abordagem

de animação com as áreas de influência, a técnica da Seção 4.3 não necessita de tal associação.

5.3 Considerações finais

Neste capítulo foram apresentados os resultados de desempenho para as técnicas propostas nesta dissertação (Seções 4.2 e 4.3), levando em consideração o número de áreas de influência enviadas para a técnica da Seção 4.2, e o número de vértices encontrados em cada modelo facial para a técnica da Seção 4.3. Foi descrito também, o computador utilizado para a coleta desses resultados, e os modelos faciais trabalhados na realização dos testes.

Como foi observado neste capítulo, ambas as técnicas foram capazes de produzir rugas em tempo real, mesmo não utilizando computadores de última geração em termos de processamento.

A técnica descrita na Seção 4.2 destaca-se em performance quando a mesma trabalha com poucas áreas de influência, no entanto, o mesmo não ocorre quando são acrescentadas mais áreas, ou utilizados modelos com maior número de vértices, aumentando consideravelmente o tempo que cada quadro da animação gasta para ser desenhado. O mesmo ocorre com a técnica descrita na Seção 4.3, perdendo performance quando utilizados modelos com maior densidade de polígonos, no entanto esta técnica não utiliza áreas de influência, sendo as informações calculadas de acordo com os Vetores Direção de Rugas em cada vértice do modelo facial.

Outra particularidade é vista na técnica da Seção 4.2, que possui a necessidade de associação de cada área de influência a um elemento do sistema de animação facial utilizado. Já a técnica da Seção 4.3 não necessita de tal associação, sendo que o deslocamento dos vértices realizado pela animação facial são suficientes para prover informações para o cálculo de apresentação de rugas.

Como é observado nos testes comparativos (Seção 5.2.3), o uso de Vetores Direção de Rugas aplicados sobre cada vértice do modelo facial exige maior processamento, logo a técnica da Seção 4.3 é indicada para sistemas de animação facial em que os cálculos de apresentação de rugas objetivam a independência entre os vértices para a exibição das rugas, pois a posição de cada vértice do modelo facial influencia no surgimento das rugas sobre o modelo. No entanto é importante enfatizar que em modelos com maior densidade de polígonos, a performance do sistema será prejudicada.

A técnica da Seção 4.2 se destacou em processamento ao utilizar modelos com maior densidade de polígonos, no entanto, foram utilizadas somente 8 áreas de influências ao invés de se obter controle sobre a apresentação de rugas em cada vértice, ou seja, uma área de influência é exibida com a mesma porcentagem de visibilidade sobre toda sua área delimitada, não sendo possível habilitar a visibilidade somente de parte da área de influência.

No caso do computador utilizado para os testes, o número máximo de áreas de influência suportado foi 16. Em outras palavras, ao passo que a técnica da Seção 4.3 consegue calcular a visibilidade das rugas sobre cada vértice indepentendemente, a técnica da Seção 4.2 só é capaz de fornecer um número limitado de áreas de influência, que são ativadas igualmente sobre toda sua área de atuação. O número de áreas suportadas irá variar de acordo com a memória disponível no equipamento utilizado.

Ambas as técnicas se mostraram capazes de trabalhar em tempo real, mesmo quando submetidas

a elevado processamento, devido ao número de áreas de influência, ou total de vértices utilizados.

Capítulo 6

Conclusão

6.1 Considerações gerais

Sistemas de animação facial são de grande importância em diversas mídias de hoje em dia, sendo amplamente utilizados em cinema, televisão, websites interativos, jogos eletrônicos, entre outras. Particularmente, a exibição de detalhes faciais, tais como rugas, tem sido trabalhada com a finalidade de se adicionar realismo às animações, como pode ser visto nos jogos da última geração, onde os personagens necessitam passar o que estão sentindo para o jogador, mesmo que utilizando apenas expressões para isso.

No entanto, a tarefa de se adicionar rugas em modelos virtuais não é trivial, principalmente quando esta deve ser realizada em tempo real e com alto grau de realismo. Com base nisso, foram desenvolvidas neste trabalho duas técnicas para a apresentação de rugas na superfície de modelos virtuais.

Com a primeira técnica (Seção 4.2) foi proposta a utilização de áreas de influência descritas por mapas de textura, permitindo flexibilidade na definição dessas áreas, visto que a maioria dos sistemas operacionais hoje em dia vem com pelo menos um software de edição de imagem. A técnica proposta necessita que o usuário final crie uma conexão entre a área de influência e o objeto de animação do modelo facial, que pode ser por exemplo um músculo virtual, ou a posição de um vértice.

Para a segunda técnica proposta neste trabalho (Seção 4.3), foram adicionados vetores extras, dispostos sobre todos os vértices do modelo facial, representando o sentido em que um deslocamento dos vértices produz rugas. Esta técnica soluciona algumas dificuldades da anterior, no que se refere a configuração das áreas de influência, que neste caso são definidas em cada vértice, permitindo maior flexibilidade. Outra melhoria foi a desassociação da necessidade de uma conexão com o objeto de animaçao facial, sendo que o próprio deslocamento dos vértices é capaz de prover informações para o cálculo da exibição das rugas, internamente à placa gráfica.

A partir dos resultados experimentais do Capítulo 5, pode-se concluir que ambas as técnicas (4.2 e 4.3) são capazes de adicionar rugas em sistemas de animação facial em tempo real, levando-se em consideração as devidas limitações de cada uma delas, como o limite do número de áreas de influência

na técnica da Seção 4.2 suportado pelo equipamento trabalhado, e a impossibilidade de definição de mais de um Vetor Direção de Rugas para cada vértice na técnica da Seção 4.3.

6.2 Contribuições

A contribuição deste trabalho é a proposta de duas técnicas que visam facilitar e melhorar a adição de rugas sobre modelos virtuais.

Uma outra contribuição deste trabalho são as ferramentas desenvolvidas para gerar o conteúdo necessário ao funcionamento das técnicas. No caso da primeira técnica (Seção 4.2), as áreas de influência foram agrupadas em texturas tridimensionais e foram criadas classes para controlar e enviar informações ao *shader*. Para a segunda técnica (Seção 4.3), foi desenvolvida uma ferramenta visual para definir os Vetores Direção de Rugas, facilitando a configuração dos mesmos. A partir dessas ferramentas foi possível obter dados para testar e validar o modelo.

6.3 Trabalhos futuros

Com relação à continuidade deste trabalho, ficam os seguintes casos a serem testados ou aplicados:

- Para o presente trabalho foi utilizado o *shader* de *normal mapping* para exibição dos detalhes visuais, para trabalhos futuros poderiam ser testadas técnicas de exibição de detalhes com apresentação de silhuetas, como *relief mapping* ou *displacement mapping* com subdivisão adaptativa. Estes testes são interessantes pois o trabalho atual, apesar de apresentar rugas sobre o modelo facial, não exibe suas silhuetas. Considera-se a apresentação de silhuetas sobre o modelo facial uma melhoria para o presente trabalho.

- Na técnica da Seção 4.2, um trabalho futuro seria o desenvolvimento de uma ferramenta visual para a criação das áreas de influência. No presente trabalho foram utilizados programas de edição de imagens disponíveis nas distribuições de sistemas operacionais, no entanto, seria interessante a possibilidade de delimitar as áreas de influência diretamente na superfície do modelo facial tridimensional. Essa ferramenta poderia também agrupar todas as informações necessárias diretamente em texturas 3D, passo este que é realizado via código no presente trabalho.

- Na técnica da Seção 4.3, adicionar a possibilidade de sobreposição de Vetores Direção de Rugas, assim como múltiplos mapas de normais para sentidos de rugas diferenciados. Esta melhoria seria interessante para dar mais liberdade para se produzir rugas em sentidos e formatos diferentes. No caso deste trabalho, foi trabalhado apenas um mapa de normais contendo todas as rugas possíveis do modelo facial, trabalhando com mais de um mapa de normais, essas rugas poderiam ser definidas juntamente com os Vetores Direção. Da mesma forma, com mais de

um Vetor Direção definido por vértice, pode-se produzir rugas para diferentes expressões. Um exemplo de onde esses Vetores Direção de Rugas adicionais podem ser úteis, são as rugas da bochecha, que podem ser formadas de acordo com diversas expressões faciais.

Referências Bibliográficas

BANDO, Y.; KURATATE, T.; NISHITA, T. A simple method for modeling wrinkles on human skin. In: *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*. Beijing, China: IEEE Computer Society, 2002. p. 166–175. ISBN 0-7695-1784-6.

BATAGELO, H. C. *Uma arquitetura de suporte a interaes 3D integrada a GPU*. Tese (Doutorado) — Universidade Estadual de Campinas - Unicamp, Campinas, So Paulo, Brasil, Junho 2007.

BATHE, K. J. *Finite Element Procedures in Engineering Analysis*. Englewood Cliffs, New Jersey: Prentice-Hall, 1982. ISBN 10: 0133173054.

BERGERON, P.; LACHAPELLE, P. Controlling facial expressions and body movements in the computer-generated animated short "tony de peltrie". In: *SIGGRAPH '85: Advanced Computer Animation - Tutorial*. San Francisco, CA: [s.n.], 1985.

BICKEL, B. et al. Multi-scale capture of facial geometry and motion. *ACM Transactions on Graphics*, ACM, v. 26, n. 3, p. 33, 2007. ISSN 0730-0301. Proceedings of ACM SIGGRAPH 2007.

BLINN, J. F. Simulation of wrinkled surfaces. In: *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*. Atlanta, Georgia, USA: ACM, 1978. v. 5, p. 286–292.

BOISSIEUX, L. et al. Simulation of skin aging and wrinkles with cosmetics insight. In: *Eurographics Workshop on Animation and Simulation (EGCAS 2000)*. Interlaken, Switzerland: [s.n.], 2000. p. 15–28.

CHADWICK, J. E.; HAUMANN, D. R.; PARENT, R. E. Layered construction for deformable animated characters. In: *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*. Boston, Massachusetts, USA: ACM, 1989. p. 243–252. ISBN 0-201-50434-0.

COMBAZ, J.; NEYRET, F. Painting folds using expansion textures. In: *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*. Washington, DC, USA: IEEE Computer Society, 2002. p. 176. ISBN 0-7695-1784-6.

COOK, R. L. Shade trees. In: *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. Minneapolis, Minnesota, USA: ACM, 1984. v. 18, n. 3, p. 223–231. ISBN 0-89791-138-5.

CORRêA, R. *Animação facial por computador baseada em modelagem biomecânica*. Dissertação (Mestrado) — Faculdade de Engenharia Elétrica e de Computação - Unicamp, Campinas, São Paulo, Brasil, Novembro 2007.

DENG, X. Q. *A Finite Element Analysis of Surgery of the Human Facial Tissue*. Tese (Doutorado) — Columbia University, New York, NY, USA, 1988.

KILGARD, M. *A practical and robust bumpmapping technique for today's GPU's*. Santa Clara, CA, USA, Maro 2000.

LARBOULETTE, C.; CANI, M.-P. Real-time dynamic wrinkles. In: *CGI '04: Proceedings of the Computer Graphics International*. Crete, Greece: IEEE Computer Society, 2004. p. 522–525. ISBN 0-7695-2171-1.

LENGYEL, E. *Mathematics for 3D game programming and computer graphics*. Rockland, MA, USA: Charles River Media, Inc., 2002. ISBN 0-89871-361-7.

LUCERO, J. C.; MUNHALL, K. G. A model of facial biomechanics for speech production. *Journal of the Acoustical Society of America*, v. 106, n. 5, p. 2834–2842, Novembro 1999.

MAGNENAT-THALMANN, N. et al. A computational skin model: fold and wrinkle formation. *Information Technology in Biomedicine, IEEE Transactions on*, v. 6, n. 4, p. 317–323, Dezembro 2002. ISSN 1089-7771.

MAGNENAT-THALMANN, N.; PRIMEAU, E.; THALMANN, D. Abstract muscle action procedures for human face animation. *Visual Computer*, v. 3, n. 5, p. 290–297, Maro 1988. MIRALab, HEC/IRO, Montreal Univ., Que., Canada.

NORDENBERG, M. *Modelling and Rendering Dynamic Wrinkles in a Virtual Face*. Dissertação (Mestrado) — Centro de tecnologia de Estocolmo (KTH, Department of Speech, Music and Hearing), Stockholm, Sweden, Abril 2003.

OLIVEIRA, M. M.; POLICARPO, F. *An Efficient Representation for Surface Details*. Porto Alegre, RS, Brasil, Janeiro 2005. Relatrio de Pesquisa RP-351.

PARKE, F. I. Computer generated animation of faces. In: *ACM'72: Proceedings of the ACM annual conference*. Boston, Massachusetts, United States: ACM, 1972. p. 451–457.

PARKE, F. I. Parameterized models for facial animation. *IEEE Computer Graphics and Applications*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 2, n. 9, p. 61–68, Novembro 1982. ISSN 0272-1716.

PARKE, F. I.; WATERS, K. *Computer facial animation*. Natick, Massachusetts, USA: A. K. Peters, Ltd., 1996. ISBN 1-56881-014-8.

PASQUARIELLO, S.; PELACHAUD, C. Greta: A simple facial animation engine. In: *In Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*. [S.l.: s.n.], 2001.

PATIDAR, S.; BHATTACHARJEE, S.; NARAYANAN, P. J. *Exploiting the Shader Model 4.0 Architecture*. [S.l.], 2007. Technical Report IIIT/TR/2007/145.

PIEPER, S.; ROSEN, J.; ZELTZER, D. Interactive graphics for plastic surgery: A task-level analysis and implementation. In: *I3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*. Cambridge, Massachusetts, USA: ACM, 1992. p. 127–134. ISBN 0-89791-467-8.

PLATT, S. M.; BADLER, N. I. Animating facial expressions. In: *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques*. Dallas, Texas, USA: ACM, 1981. p. 245–252. ISBN 0-89791-045-1.

SEDERBERG, T. W.; PARRY, S. R. Free-form deformation of solid geometric models. In: *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. Dallas, Texas, USA: ACM, 1986. p. 151–160. ISBN 0-89791-196-2.

TERZOPOULOS, D.; WATERS, K. Physically-based facial modelling, analysis, and animation. *Journal of Visualization and Computer Animation*, v. 1, n. 2, p. 73–80, Dezembro 1990.

TREFETHEN, L. N.; BLAU, D. *Numerical Linear Algebra*. [S.l.]: Society for Industrial and Applied Mathematics, 1997. ISBN 1-58450-037-9.

TU, P.-H. et al. Expression detail mapping for realistic facial animation. In: *Eighth Internal Conference CAD/Graphics'03*. Macao, China: [s.n.], 2003. p. 20–25.

VIAUD, M.-L.; YAHIA, H. *Facial animation with wrinkles*. [S.l.], Setembro 1992.

VOLINO, P.; MAGNENAT-THALMANN, N. Fast geometric wrinkles on animated surfaces. In: *7th International Conference in Central Europe on Computer Graphics and Visualization - Winter School of Computer Graphics 1999*. Plzen, Repblica Tcheca: [s.n.], 1999. p. 55–66.

WANG, Y.; WANG, C. C. L.; YUEN, M. M. F. Fast energy-based surface wrinkle modeling. *Computers and Graphics*, Pergamon Press, Inc., Elmsford, NY, USA, v. 30, n. 1, p. 111–125, 2006.

WATERS, K. A muscle model for animation three-dimensional facial expression. In: *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. Anaheim, California, USA: ACM, 1987. p. 17–24. ISBN 0-89791-227-6.

WU, P. K. Y.; THALMANN, N. M. Simulation of static and dynamic wrinkles of skin. In: *CA '96: Proceedings of the Computer Animation*. [S.l.]: IEEE Computer Society, 1996. p. 90. ISBN 0-8186-7588-8.

WU, Y. et al. Simulating wrinkles and skin aging. *The Visual Computer*, v. 15, n. 4, p. 183–198, Julho 1999.

WU, Y.; THALMANN, N. M.; THALMANN, D. A plastic-visco-elastic model for wrinkles in facial animation and skin aging. In: *Pacific Graphics '94: Proceedings of the second Pacific conference on Fundamentals of computer graphics*. Beijing, China: World Scientific Publishing Co., Inc., 1994. p. 201–213. ISBN 981-02-1896-6.