Rafael Viana de Carvalho

Simulador de Superfícies Deformáveis

Orientador: Ernesto Prado Lopes

Co-Orientadora: Eliana Prado Lopes Aude

Rio de Janeiro

Janeiro/2008

FICHA CATALOGRÁFICA

C331 Carvalho, Rafael Viana
Simulador de Superfícies Deformáveis / Rafael Viana de Carvalho. - Rio de Janeiro, UFRJ/IM/NCE, 2008.
127 p. : il.

Dissertação (Mestrado em Informática) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Núcleo de Computação Eletrônica, 2008.

Orientador: Ernesto Prado Lopes, Co-Orientadora: Eliana Prado Lopes Aude

1. Modelos de Superfície. 2. Deformações de Superfícies 3. Computação Gráfica. 4. Modelagem e Simulação. 5. Simulador. I. Ernesto Prado Lopes (Orient.). II. Universidade Federal do Rio de Janeiro. Instituto de Matemática. Núcleo de Computação Eletrônica. III. Título.

CDD

Simulador de Superfícies Deformáveis

Rafael Viana de Carvalho

Dissertação submetida ao Corpo Docente do Núcleo de Computação Eletrônica - Instituto de Matemática da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para a obtenção do grau de Mestre.

Aprovada por:

Prof. Ernesto P. Lopes, Ph.D.
(Orientador)

Prof[a]. Eliana P. L. Aude, Ph.D.
(Co-Orientador)

Prof. José R. L. de Mattos, D.Sc.

Prof[a]. Luziane F. de Mendonça, D.Sc.

Rio de Janeiro, RJ – Brasil

Aos meus pais Jocundo e Jaidete, minhas irmãs Carol e Elaine e minha avó Nazinha

Agradecimentos

Primeiramente a Deus, que sempre iluminou meus caminhos, me dando saúde, conforto, força e paz, em todos os momentos de minha vida.

A minha família, meu pai, minha mãe, minhas irmãs e minha avó, que sempre me apoiaram, torceram e rezaram por mim, mesmo com a distância e com a saudade, estiveram sempre ao meu lado. A eles dedico cada vitória de minha vida.

Ao meu orientador, Ernesto P. Lopes por sua dedicação, conhecimentos transmitidos e confiança em meu trabalho.

A professora Eliana Aude por acreditar em meu potencial, ao convidar-me para realizar essa jornada que foi o período do mestrado, por seu exemplo, dedicação e carinho.

Aos meus amigos e companheiros de trabalho Julio e Henrique, que tanto me ajudaram com apoio, alegria e muitos ensinamentos, essenciais para que esse trabalho fosse realizado.

Aos meus grandes amigos Flávio, Silvia e Aberson que me deram força nos momentos difíceis, alegria nos momentos de descontração, e ensinamento nos momentos de reflexão.

A minha tia Elisa e toda sua família, que tão bem me acolheu aqui no Rio de Janeiro, me dando carinho, atenção e cuidados, amenizando a saudade de meus pais e a solidão.

Resumo

CARVALHO, Rafael V. **Simulador de Superfícies Deformáveis.** 2008. 127 p. Tese (Mestrado em Informática) - Instituto de Matemática, Núcleo de Computação da Universidade Federal do Rio de Janeiro, Rio de Janeiro.

Este trabalho está relacionado ao problema de simulação de superfícies deformáveis. É apresentado um estudo sobre os tipos de curvas e superfícies em computação gráfica e sua deformação. Através de um modelo de animação, são aplicadas técnicas de computação gráfica para obter a visualização da evolução das deformações das superfícies. Este modelo é baseado no modelo de sistemas de partículas e inspirado em Yi Wu (WU, 1995). Ele atua no controle das deformações mantendo a suavidade das superfícies. Desenvolveu-se um aplicativo que integra as técnicas de iluminação, textura e animação, e que, por meio de interfaces gráficas, são configurados diversos aspectos de luminosidade, posição do observador, características de fontes luminosas e perturbação das formas das superfícies. Através do simulador desenvolvido obtém-se uma animação tridimensional das deformações de forma dinâmica.

Abstract

CARVALHO, Rafael V. **Simulador de Superfícies Deformáveis.** 2007. 127 p. Tese (Mestrado em Informática) - Instituto de Matemática, Núcleo de Computação da Universidade Federal do Rio de Janeiro, Rio de Janeiro.

This work is concerned with the deformable surface simulation problem. A study of the types of curves and surfaces and its deformations are presented. Computer graphics techniques are applied for a visual solution of the deformation in the surfaces, allowing visualization of oscillation and evolution of the movement based on an animation model. We use a model of particles system proposed by Yi Wu (WU, 1995), which operates on the control of deformation and smoothness of the surface in order to provide a dynamic animation. An application that integrates the lightining techniques, texture and animation has been developed. Through the use of graphical interfaces, many aspects concerning luminosity, observer position, light source features, perturbation and shape of the surface can be configured. An animation of the three-dimensional dynamic deformation is obtained as a result from this simulation tool.

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

1.1 Motivação

A cada dia, observa-se o crescente uso do computador nos mais diversos meios. Como seu desempenho aumenta continuamente, diversas linhas de pesquisa que até então eram conhecidas somente na teoria, como o fractal (EBERT, 1998), encontram uma ferramenta que serve como base para seu desenvolvimento e estudo.

A Computação Gráfica é um exemplo de ciência que depende da capacidade de processamento dos computadores. Através dela, é possível representar, graficamente, situações complexas em diversas áreas da vida real. Ela se fundamenta no uso do computador para transformar regras de geometria e física em imagens que têm significado para seres humanos (GLASSNER, 1995). Seu uso estende-se a aplicações científicas, como: simulação de fenômenos físicos, visualização científica e aplicações domésticas como: filmes, comerciais, diversão eletrônica, entre outras.

No passado, as representações de objetos do mundo real eram feitas através de estruturas simples como os *wire frame*[1] que davam ao usuário a impressão do objeto, mas, obviamente, insatisfatória. Em busca de uma melhor representação gráfica, os algoritmos tiveram que evoluir para possibilitar a representação completa das superfícies.

[1] Representação 3D de um objeto através de vértices e arestas interligados por retas ou curvas formando polígonos cujas faces não são preenchidas ou coloridas

A modelagem de objetos deformáveis tem sido alvo de estudo pela comunidade de computação gráfica por mais de duas décadas. Ela tem evoluído através das aplicações, destacando-se aí a pura arte, que busca gerar imagens e animações em cenas virtuais de fina beleza e realismo.

Os modelos vêm sendo criados e usados nas mais diversas áreas. Poderíamos citar como exemplos: a manufatura e projeto de vestuário (animação de vestuário sobre figuras humanas, comportamento de tecidos sobre outros objetos ou suspensos em ganchos e cabides, etc); as animações de expressões faciais de figuras humanas e de animais; as análises de imagens médicas (incluindo segmentação, representação, registro e captura de movimento da forma); a simulação dos movimentos dos músculos; em simulações cirúrgicas e treinamento de pessoal na área de cirurgia, etc. A exigência de processamento em tempo real e modelagem fisicamente realística é uma constante em quase todos estes exemplos. Para sumários do uso de modelos deformáveis em análises médicas, recomenda-se a referência (MCLNERNY, 1996). Aplicações e panorama de modelagem e animação de vestuários são apresentados em (MCLNERNY, 1996; VOLINO, 2000; ETZEMUF, 2001). Para uma revisão sobre superfícies deformáveis: topologia, geometria e deformação, recomenda-se (MONTAGNAT, 2001).

Muitos objetos do mundo real apresentam formas ou contornos suaves. Para modelar os objetos deformáveis em computação gráfica, é necessário a utilização de curvas e superfícies suaves na sua representação computacional (ANGEL, 2004).

Para esse trabalho foi escolhido a modelagem de superfícies deformáveis, buscando uma representação computacional abordando as técnicas de computação gráfica para criar um ambiente tridimensional capaz de simular tais superfícies em movimento.

A motivação num primeiro momento é de sintetizar imagens de superfícies deformáveis, com características de cor, iluminação e suavidade, que irão ser deformadas usando um modelo físico de sistemas de partícula inspirado em Wu (WU, 1995).

1.2 Modelos Deformáveis

Podemos classificar os modelos deformáveis em três tipos: (1) modelos geométricos, (2) modelos físicos, (3) modelos híbridos que tentam unir uma abordagem geométrica com uma abordagem física. Os modelos geométricos não levam em consideração as propriedades físicas do objeto, fazendo uso de entidades e/ou transformações geométricas para obterem os efeitos desejados. São úteis para modelagem estacionária, objetos rígidos cuja forma não muda ao longo do tempo. Infelizmente são modelos inertes (TERZOPOULOS, 1988). Os modelos físicos buscam incorporar conceitos e propriedades físicas para a animação do objeto. Eles constituem um campo importante e tem atraído interesse da comunidade de computação gráfica, pela possibilidade de unir realismo e controle intuitivo em simulações estáticas e dinâmicas, o que não é possível fazer com os modelos geométricos. A desvantagem deste tipo de modelo está no custo computacional maior do que nos modelos geométricos (complexidade de tempo e memória). Finalmente, os modelos híbridos tentam suprir as deficiências dos modelos geométricos na simulação de distintos comportamentos de curvatura com as técnicas de modelagem física para aumentar o realismo nas simulações.

O uso dos modelos físicos na animação de superfícies deformáveis tem sido um importante campo de pesquisa na computação gráfica. Existem dois tipos de abordagem: os Modelos da Mecânica do Contínuo e de Sistemas de Partículas.

Os modelos de mecânica do contínuo, levam em consideração uma superfície de formação contínua, ou seja, sem espaços vazios ou fendas, desconsiderando sua constituição molecular, sendo representada parametricamente. Em torno de cada ponto da superfície é considerada uma região contínua (geometria diferencial) e nela são impostos princípios de equilíbrio para obter equações diferenciais parciais na forma integral ou pontual que governa o equilíbrio. Por menor que seja a vizinhança local de cada ponto, ainda assim será um elemento de dimensão maior do que o ponto, ou partícula, considerando elemento sem dimensão. Nessa abordagem, a incorporação dos parâmetros de controle do comportamento de deformação do objeto é feita de forma natural. Esses parâmetros já vêm embutidos no modelo contínuo e não há busca para modelar possíveis interações internas (AU, 2000; CARIGNAN, 1992; CHEN, 1995; TERZOPOULOS, 1987).

Os modelos de sistemas de partículas partem do menor elemento que constitui o objeto, as partículas, as quais são consideradas como pontos-massa que se movem por influência das forças que atuam sobre elas, tais como: gravidade, velocidade, força elástica, etc. Em computação gráfica, o sistema de partícula foi usado inicialmente para modelar fogo, fagulhas, explosão, quedas d'águas, oceano, chuva, neve, e outros fenômenos naturais (REEVES, 1983). Mais recentemente, o sistema de partícula tem sido utilizado para modelar superfícies deformáveis. Um trabalho interessante nessa área é a descrição do comportamento do caimento e dobras de pano proposto por Breen (BREEN, 1991). Nesse campo, os métodos de interação entre partículas são usados para desenvolver um modelo teórico para simular as dobras e decaimento de tecidos de pano como algodão, lã, etc. Hilton e Egbert (HILTON, 1994) propuseram o uso de campo vetorial para sistemas de partículas 3D.

Szeliski e Tonnesen (SZELISKI, 1992) definem um modelo de superfície orientada para sistema de partícula que discretiza as superfícies em trechos (polígnos) especificando suas

conectividades de forma automática, sem necessidade de cálculos manuais. Essa modelagem da superfície pode ser usada para partir, unir ou esticar superfícies deformáveis sem a necessidade do conhecimento da topologia da superfície. Os sistemas massa-mola, são essenciais para modelos de partículas, onde cada partícula é conectada aos seus vizinhos por meio de fios ou molas, para incorporar os parâmetros de controle do comportamento do material da superfície, tais como: esticamento, cisalhamento e curvatura. Estes fios ou molas constituem um mecanismo para modelar as reações ou interações de uma determinada partícula com as suas vizinhas.

Yi Wu apresentou em (WU, 1995) um método simples para criar e animar superfícies deformáveis usando modelo físico, representando a deformação de uma superfície através de sistemas de partículas. Esse método apresenta uma solução analítica para a equação de movimento utilizando métodos de discretização para resolver a equação da simulação. Esse método proporciona uma boa estabilidade além de ser rápido em seus cálculos, podendo facilmente ser usado em qualquer tipo de superfíce deformável proporcionando bons resultados.

Eischen e Bigliani (EISCHEN, 2000) realizaram uma bateria de testes e simulações para comparar dois modelos distintos: um modelo de contínuo e um modelo de partículas. Eles concluíram que o modelo contínuo é mais preciso, enquanto o modelo de partículas tem um melhor desempenho em tempo e apresenta resultados qualitativamente corretos.

1.3 Objetivos

Uma visualização de superfícies deformáveis ideal seria aquela que embutisse quatro características importantes: realismo, controle intuitivo, eficiência (tempo e memória) e precisão nas simulações. Esse trabalho visa prioritariamente desenvolver uma simulação

em um ambiente gráfico tridimensional da deformação de uma superfície e o seu comportamento quando aplicada a uma animação baseada em um modelo físico de sistemas de partícula de acordo com as funções de controle inspirada por Wu (WU, 1995). Acreditamos que o avanço tecnológico, que nos oferece PC's cada vez mais velozes e com grande capacidade de armazenagem, dissipará o problema de eficiência (tempo e memória). Embora exista uma forte correlação entre a precisão e o realismo, não é o nosso objetivo avaliar a qualidade do nosso ambiente gráfico na simulação sob o ponto de vista numérico.

Nessa dissertação, temos como objetivo apresentar o ambiente gráfico desenvolvido simulando a deformação de uma superfície embasado num modelo de sistemas de partícula, que possibilite um controle intuitivo e resultados visualmente realísticos.

1.4 Descrição do Trabalho

O conteúdo desse trabalho está dividido em seis capítulos, dispostos como segue.

No Capítulo 2 são fornecidos os fundamentos básicos das técnicas para processamento e visualização de objetos gráficos e como elas são representadas matematicamente. Técnicas de transformações geométricas, projeção, iluminação, reflexão e tonalização são essenciais na simulação de objetos, retratando suas características físicas o mais fiel possível.

A exposição do que são as curvas e superfícies e seus modelos é feita no Capítulo 3. A representação de um modelo de curvas e superfícies envolve o problema de como caracterizar objetos suaves e como converter esta caracterização na forma computacional.

No Capítulo 4 é apresentado o modelo físico, o qual servirá como base teórica para a animação das deformações.

Detalhes de implementação do ambiente gráfico e da simulação são feitos no Capítulo 5. Nele são apresentadas as técnicas de programação utilizadas na construção da simulação, as ferramentas e biblioteca gráfica utilizada, sua interface e características.

As considerações finais e sugestões de trabalhos futuros são abordadas no Capítulo 6.

2 Visualização Espacial

Um sistema de computação gráfica tem como entradas uma série de informações sobre as características geométricas dos objetos que se quer retratar, sobre suas propriedades óticas e sobre as fontes de luz que iluminam a cena. Como saída produz um quadro de *pixels*, cada qual com um valor de cor, que colocadas na tela do computador nos dá uma imagem da cena. As informações geométricas dos objetos são dadas na forma de uma coleção ordenada de vértices que definem primitivas que são, em geral, polígonos e retas. Estas informações sofrem vários tipos de transformações geométricas, correspondendo as transformações necessárias nos objetos e na câmera para a visualização do mesmo da maneira desejada. Inicialmente será discutido neste capítulo as transformações geométricas dos objetos, principalmente as transformações afins e projeções usando as coordenadas homogêneas. Em seguida, trataremos da caracterização das propriedades óticas dos objetos e das fontes de luz, abordando os modelos de fontes de luz e de reflexão das superfícies dos objetos que são informações fundamentais para se conseguir a sensação tridimensional na imagem.

2.1 Transformações Geométricas

As transformações geométricas são ferramentas importantes na manipulação das cenas tridimensionais. Elas são utilizadas, por exemplo, para transladar, rotacionar, escalar

e espelhar os objetos, atuando no conjunto dos vértices ou pontos que definem esses objetos, transformando-os em um novo conjunto de vértices ou pontos (SILVA, 2006). As *coordenadas homogêneas* são a representação adequada para pontos e vetores e para as suas transformações geométricas em um sistema gráfico. A operação de *Translação* é responsável pelo deslocamento de um objeto em uma dada direção através de uma distância fixada, já a *Rotação* gira o objeto em um determinado eixo cartesiano, a operação de *Escala* altera o tamanho de um objeto em relação ao seu tamanho original. Essas operações tanto aplicadas em separado como em conjunto são importantes na criação de cenas e alteração dos objetos tridimensionais.

2.1.1 Coordenadas Homogêneas

A representação de pontos e vetores na álgebra linear ordinária é ambígua, uma vez fixado um sistema de coordenadas retangulares, um elemento do R^3 representa ao mesmo tempo um ponto e um vetor. O uso das *coordenadas homogêneas* permite resolver este problema. Em coordenadas homogêneas, pontos de um espaço afim de dimensão 3 e vetores do espaço vetorial associado são representados univocamente por elementos do R^4. As transformações geométricas de rotação, translação e de escala assumem uma representação matricial como uma transformação linear em espaços vetoriais de dimensão 4.

Em um espaço afim, um marco é definido por um ponto, chamado origem e pelos vetores de uma base do espaço vetorial associado. No caso de dimensão três, o marco é representado pela especificação: $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{P}_0)$, onde os três primeiros elementos são vetores da base e o quarto é a origem. Qualquer ponto $\mathbf{P}$ do espaço afim pode ser escrito de uma maneira única como (BERG, 1998):

$$\mathbf{P} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3 + \mathbf{P}_0$$

Assim, é possível expressar esta relação, usando um produto formal de matrizes, como:

$$\mathbf{P} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{P}_0 \end{bmatrix}.$$

A matriz linha é a representação das coordenadas homogêneas do ponto $\mathbf{P}$ no marco determinado por $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ e $\mathbf{P}_0$. Equivalentemente, é possível dizer que $\mathbf{P}$ pode ser representado por:

$$\mathbf{P} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 1 \end{bmatrix}.$$

A representação de um vetor tridimensional segue uma lógica similar. No mesmo marco, qualquer vetor $\mathbf{w}$ pode ser escrito como:

$$\mathbf{w} = \delta_1 \mathbf{v}_1 + \delta_2 \mathbf{v}_2 + \delta_3 \mathbf{v}_3$$

$$\mathbf{w} = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{P}_0 \end{bmatrix}.$$

Equivalentemente, $\mathbf{w}$ pode ser representado por:

$$\mathbf{w} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ 0 \end{bmatrix}.$$

Existem numerosas formas de interpretar esta formulação geometricamente, entretanto esse trabalho irá ressaltar apenas a simplicidade das operações com pontos e vetores utilizando suas representações em coordenadas homogêneas e a álgebra matricial ordinária.

Considerando a operação de mudança de marco, sejam $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{P}_0)$ e $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{Q}_0)$ dois marcos, a expressão dos vetores de base e o ponto referencial do segundo marco em função do primeiro é:

$$\mathbf{u}_1 = \gamma_{11}\mathbf{v}_1 + \gamma_{12}\mathbf{v}_2 + \gamma_{13}\mathbf{v}_3$$

$$\mathbf{u}_2 = \gamma_{21}\mathbf{v}_1 + \gamma_{22}\mathbf{v}_2 + \gamma_{23}\mathbf{v}_3$$

$$\mathbf{u}_3 = \gamma_{31}\mathbf{v}_1 + \gamma_{32}\mathbf{v}_2 + \gamma_{33}\mathbf{v}_3$$

$$\mathbf{Q}_0 = \gamma_{41}\mathbf{v}_1 + \gamma_{42}\mathbf{v}_2 + \gamma_{43}\mathbf{v}_3 + \mathbf{P}_0$$

Estas equações, na forma de matrizes, podem ser escritas como:

$$\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{Q}_0 \end{bmatrix} = M \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{P}_0 \end{bmatrix},$$

Onde M é uma matriz 4×4

Considerando M' uma submatriz 3×3 da matriz M, cujo seus elementos são:

$$M' = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}.$$

A matriz M' é a matriz de mudança de base no R^3, portanto, ela é não singular e seu determinante é não nulo. Isto é verdadeiro se, e somente se, o determinante da matriz M 4×4 é não nulo. M é chamada de *matriz de representação*. É possível utilizá-la para computar diretamente as alterações na representação de pontos e vetores.

Supondo que $\mathbf{a}$ e $\mathbf{b}$ são representações em coordenadas homogêneas de um vetor em dois marcos distintos, então:

$$\mathbf{b}^T \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{Q}_0 \end{bmatrix} = \mathbf{b}^T M \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{P}_0 \end{bmatrix} = \mathbf{a}^T \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{P}_0 \end{bmatrix}.$$

$$\mathbf{a}^T = \mathbf{b}^T M$$

$$(\mathbf{a}^T)^T = (\mathbf{b}^T M)^T$$

Assim,

$$\mathbf{a} = M^T \mathbf{b}.$$

Onde M^T, é:

$$M^T = \begin{bmatrix} \gamma_{11} & \gamma_{21} & \gamma_{31} & \gamma_{41} \\ \gamma_{12} & \gamma_{22} & \gamma_{32} & \gamma_{42} \\ \gamma_{13} & \gamma_{23} & \gamma_{33} & \gamma_{43} \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

que, assim como M, é determinada por 12 coeficientes.

É importante observar que a composição das transformações geométricas pode ser representada pela multiplicação das representações matriciais dessas transformações. Embora se tenha que trabalhar em quatro dimensões para resolver problemas tridimensionais, o uso desse tipo de representação simplifica muito os cálculos e a programação.

2.1.2 Translação

A *translação* é uma operação que desloca um ponto, em uma dada direção, por uma distância fixada, como ilustrado na figura 2.1. Se o ponto **P** for deslocado na direção do vetor **d**, de uma distância $|d|$, então:

$$\mathbf{P}' = \mathbf{P} + d.$$

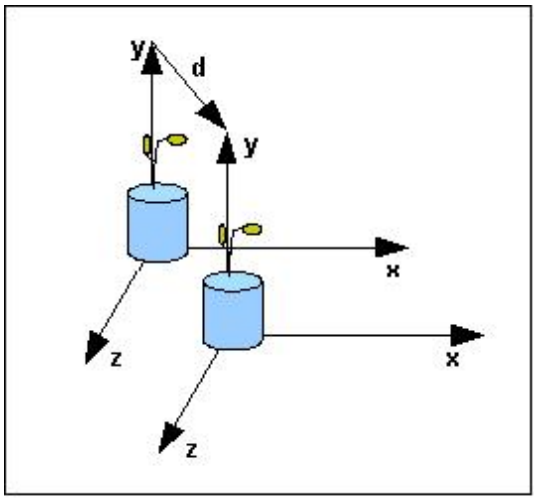

Figura 2.1: Translação de um objeto

Em coordenadas homogêneas a translação é representada por:

$$\mathbf{P} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \mathbf{P}' = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \text{ e } \mathbf{d} = \begin{bmatrix} \alpha_x \\ \alpha_y \\ \alpha_x \\ 0 \end{bmatrix}$$

Logo:

$$x' = x + \alpha_x$$

$$y' = y + \alpha_y$$

$$z' = z + \alpha_z$$

Uma translação é obtida pela multiplicação de uma matriz por um vetor:

$$\mathbf{P}' = T\mathbf{P},$$

onde

$$T = \begin{bmatrix} 1 & 0 & 0 & \alpha_x \\ 0 & 1 & 0 & \alpha_y \\ 0 & 0 & 1 & \alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

T é chamada *matriz de translação* e pode ser escrita como $T(\alpha_x, \alpha_y, \alpha_z)$ para enfatizar os três parâmetros de que ela depende, os demais estando fixados. É importante ressaltar que o ponto do marco que estiver sobre a origem do sistema de coordenadas será transferido, após a translação, para um novo ponto do universo. Esta característica em geral é utilizada na criação de modelos com o objetivo de posicioná-los no universo da melhor forma possível.

2.1.3 Rotação

A *rotação* define a orientação da imagem no universo. Sua especificação é mais complexa que a da translação, pois mais parâmetros estão envolvidos. A operação de rotação desloca um ponto em torno de um determinado eixo cartesiano de um ângulo θ. Essa operação inicia-se pela rotação de um ponto $\mathbf{P}$ em torno de outro ponto $\mathbf{Q}$ em um plano, especificando o ponto $\mathbf{Q}$ como origem, uma base ortogonal $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ e definindo um dos vetores, como por exemplo o vetor $\mathbf{v}_3$, como sendo um vetor perpendicular ao plano de rotação. Assim, fica definido um novo marco $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{Q})$.

Supondo que o ponto $\mathbf{P}$, cuja representação neste marco é $(x, y, 0, \mathbf{P} + (\mathbf{Q} - \mathbf{P}))$, é rotacionado em torno de $\mathbf{v}_3$ de um ângulo θ para a posição $\mathbf{P}' = (x', y', 0, \mathbf{P} + (\mathbf{Q} - \mathbf{P}))$. As equações desta rotação no plano são obtidas como segue:

(x, y) e (x', y') são escritos na *forma polar*:

$$x = \rho \cos \phi,$$

$$y = \rho \sin \phi,$$

$$x' = \rho \cos(\phi + \theta),$$

$$y' = \rho \sin(\phi + \theta),$$

Onde ρ é $||\mathbf{Q} - \mathbf{P}||$.

Usando as identidades trigonométricas para o seno e o cosseno da soma de dois ângulos, obtém-se:

$$x' = \rho \cos \phi \cos \theta - \rho \sin \phi \sin \theta = x \cos \theta - y \sin \theta,$$

$$y' = \rho\cos\phi\sin\theta + \rho\sin\phi\cos\theta = x\sin\theta + y\cos\theta.$$

Essas equações podem ser escritas na forma matricial como:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Essa matriz é a da rotação num plano em torno de um ponto. Para expressão tridimensional da rotação do ponto $\mathbf{P}$ do ângulo θ em torno do eixo perpendicular ao plano, no frame $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, Q)$, têm-se:

$$x' = x\cos\theta - y\sin\theta,$$

$$y' = x\sin\theta + y\cos\theta,$$

$$z' = z;$$

E na forma matricial

$$\mathbf{P}' = (x', y', 0, \mathbf{Q}) = M_z(\theta)\mathbf{P}$$

Onde

$$M_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A rotação pode ser feita em torno dos demais eixos ortogonais com uma argumentação idêntica. As demais matrizes para os outros eixos são:

$$M_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$M_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

A matriz de rotação em coordenadas homogêneas, como a de todas as transformações lineares no espaço tridimensional, tem a forma:

$$\begin{bmatrix} \alpha_{11} & \alpha_{21} & \alpha_{31} & 0 \\ \alpha_{12} & \alpha_{22} & \alpha_{32} & 0 \\ \alpha_{13} & \alpha_{23} & \alpha_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

quando

$$\begin{bmatrix} \alpha_{11} & \alpha_{21} & \alpha_{31} \\ \alpha_{12} & \alpha_{22} & \alpha_{32} \\ \alpha_{13} & \alpha_{23} & \alpha_{33} \end{bmatrix},$$

é M_x, M_y ou M_z, denota-se R_x, R_y ou R_z as respectivas matrizes de rotação em coordenadas homogêneas. A notação para a rotação em torno de um eixo genérico é R, e é obtida pela composição das rotações anteriormente descritas.

Uma rotação R de um ângulo θ pode ser desfeita através da mesma rotação de ângulo $-\theta$, então

$$R^{-1}(\theta) = R(-\theta).$$

Além disso, em M como todos os termos com cosseno estão na diagonal e os termos com seno não estão na diagonal, pode-se utilizar as identidades trigonométricas $\cos(-\theta) = \cos\theta$ e $\sin(-\theta) = -\sin\theta$ para encontrar:

$$R^{-1}(\theta) = R^T(\theta).$$

Para construir a matriz de uma rotação qualquer, em torno de um ponto fixado como origem, basta fazer o produto das rotações individuais sobre os três eixos

$$R = R_z R_y R_x.$$

Como a transposta do produto de matrizes é o produto das transpostas na ordem inversa para uma rotação qualquer, obtém-se também:

$$R^{-1} = R^T.$$

Uma matriz cuja inversa é igual a sua transposta é chamada de *matriz ortogonal*. A vantagem desta propriedade é que não é necessário nenhum cálculo adicional para se obter a matriz inversa, bastando apenas alterar os índices de linhas e colunas da matriz.

A figura 2.2 ilustra uma rotação feita em torno do eixo z. Neste caso, os pontos originais são multiplicados por uma matriz de rotação ao redor do eixo definido pelo vetor z no sentido anti-horário.

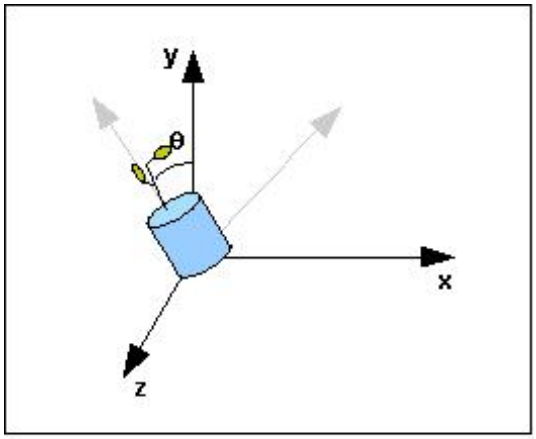

Figura 2.2: Rotação de um objeto em torno do eixo z

No caso da rotação em torno de um eixo genérico (e_x, e_y, e_z), basta transladar a figura a ser rotacionada para a origem, (calcula-se, para isto, a matriz de translação em relação ao eixo (e_x, e_y, e_z)), aplicar as rotações necessárias e, finalmente, transladar a figura para a sua posição inicial, aplicando a inversa da matriz de translação calculada inicialmente.

2.1.4 Escala

O *escalamento* altera o tamanho de um objeto, podendo torná-lo maior ou menor em relação ao seu tamanho original. As transformações com escala possuem um ponto fixo, e para especificar uma escala, é necessário, além deste ponto fixo, uma direção na qual deseja-se fazer a escala e um fator de escala (α). Para $\alpha > 1$, o objeto "cresce" em uma direção específica; para $0 \leq \alpha < 1$, o objeto "encolhe" nesta direção. Valores negativos de α fornece a *reflexão* sobre um ponto fixo, na direção de escalamento, como mostra a figura 2.3. Nesta figura o ponto fixo é a origem, $\alpha = -1$ e a direção de escalamento é $(0,1,0)$.

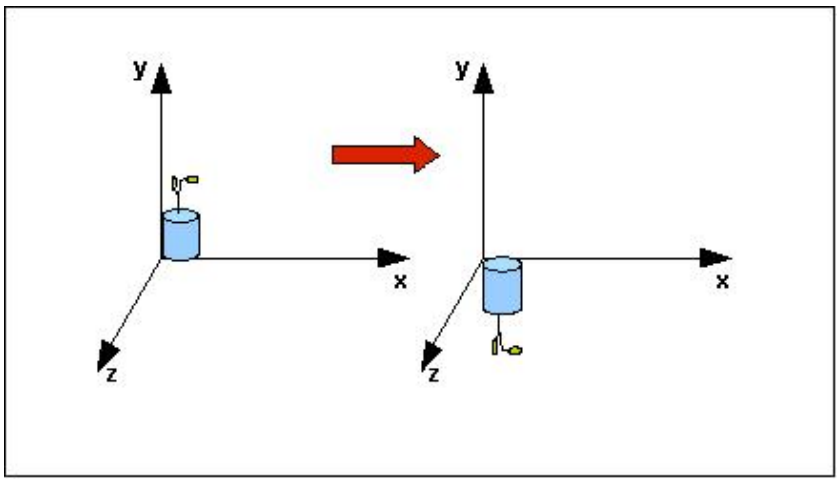

Figura 2.3: O escalamento de um objeto

Considerando o ponto fixo como sendo a origem, uma matriz de escalamento com um ponto fixo nessa posição permite o escalamento independente sobre os eixos das coordenadas. Isso quer dizer que todo o ponto que tiver sobre a origem, permanecerá nesta posição após a escala e os pontos que estão fora, irão sofrer um deslocamento.

As três equações são:

$$x' = \beta_x x,\ y' = \beta_y y,\ z' = \beta_z z.$$

Estas três equações podem ser combinadas na forma homogênea como $\mathbf{P'} = \mathbf{SP}$, onde

$$\mathbf{S} = \mathbf{S}(\beta_x, \beta_y, \beta_z) = \begin{bmatrix} \beta_x & 0 & 0 & 0 \\ 0 & \beta_y & 0 & 0 \\ 0 & 0 & \beta_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Note que, como para toda a transformação linear expressa em coordenadas homogêneas, a última linha da matriz força o quarto componente do ponto transformado a reter o seu valor.

Obter-se-á o inverso da matriz de escala aplicando os recíprocos dos fatores de escala:

$$\mathbf{S}^{-1}(\beta_x, \beta_y, \beta_z) = \mathbf{S}\left(\frac{1}{\beta_x}, \frac{1}{\beta_y}, \frac{1}{\beta_z}\right).$$

Para o escalamento de um objeto em um ponto $\mathbf{P'} = (x', y', z')$ arbitrário, é necessário utilizar a transformação na seguinte ordem:

1. Translada-se o ponto $\mathbf{P'}$ para o ponto de origem do marco;
2. Faz-se o escalamento do objeto em relação à origem e
3. Translada-se o objeto escalonado da origem para P'.

2.2 Projeção

Após a criação de cenas e objetos tridimensionais o próximo passo é efetuar a sua apresentação. Existe o problema de apresentar uma entidade tridimensional bidimensi-

onalmente, esse processo de transformação de objetos tridimensionais em um *plano de projeção* bidimensional é denominado *projeção*.

Esse processo tem sido tratado exaustivamente por desenhistas, artistas, arquitetos, que buscaram técnicas e artifícios para sistematizar e solucionar este problema. Pode-se dizer que o olho do observador coloca-se no *centro de projeção*, o plano, que deve conter o objeto ou cena projetada, é chamado de *plano de projeção*. Os segmentos de reta que saem do centro de projeção e atingem o objeto projetado no plano de projeção são chamadas de *projetantes* (ou *projetores*), a projeção de uma imagem pode ser vista na figura 2.4.

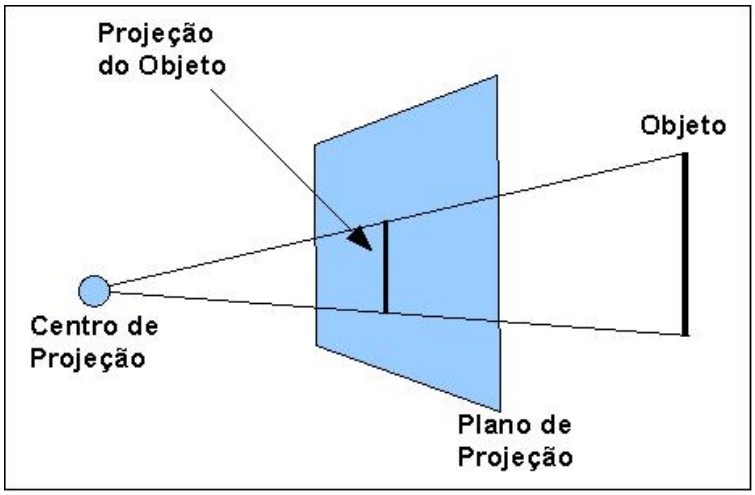

Figura 2.4: Projeção de uma imagem

A classe das projeções é conhecida como *projeção geométrica planar*, uma vez que a projeção é feita sobre um plano e não sobre uma superfície curva. Projeções geométricas planares, também referidas simplesmente como projeções, podem ser divididas em duas classes básicas: *perspectiva* e *paralela*. A diferença entre essas duas classes está na relação entre o centro de projeção e o plano de projeção. Se a distância entre eles for finita então a projeção é perspectiva, como ilustra a figura 2.5(a). Entretanto, se essa distância entre o centro de projeção e o plano for infinita então a projeção é paralela, como mostrado

na figura 2.5(b). Quando definimos uma projeção perspectiva, o centro de projeção é explicitado, enquanto que na projeção paralela deve-se informar apenas a direção da projeção (CAVALCANTI, 2000).

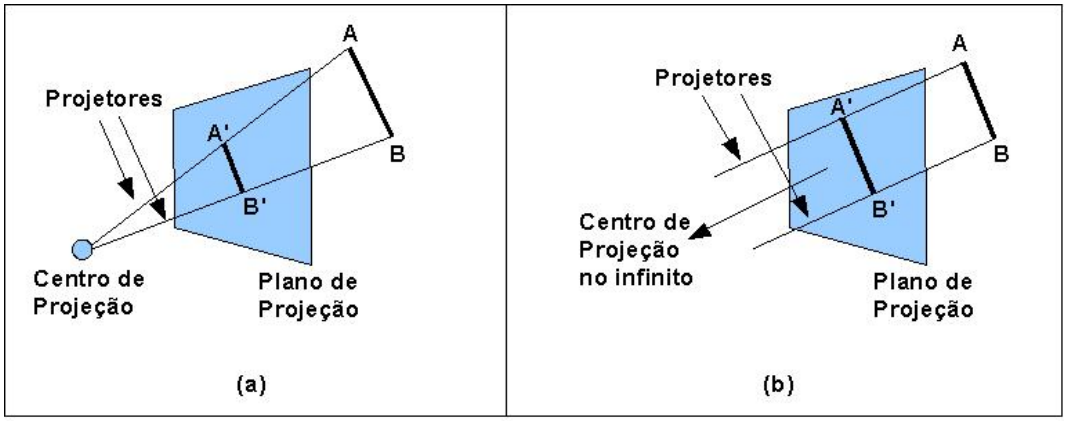

Figura 2.5: Classe das projeções: (a) Projeção Perspectiva, (b) Projeção Paralela.

As projeções perspectiva e paralela podem ser definidas através de matrizes 4×4, o que é interessante para a composição de transformações juntamente com a projeção.

2.2.1 Projeção Perspectiva

A projeção em perspectiva cria um efeito visual similar ao de sistemas fotográficos e ao da visualização humana. É utilizado quando se deseja um certo grau de realismo, uma vez que é possível se ter a percepção tridimensional da cena (FOLEY, 1990).

Os desenhos em perspectiva são caracterizados pelo *encurtamento perspectivo* e pelos *pontos de fuga*. O encurtamento perspectivo é a ilusão de que os objetos e comprimentos são cada vez menores à medida que sua distância ao centro de projeção aumenta. A projeção em perspectiva de qualquer conjunto de linhas que não são paralelas ao plano de projeção convergem para um ponto de fuga.

Projeções perspectivas são categorizadas pelo seu número de pontos de fuga principais, ou seja, o número de eixos que o plano de projeção intercepta. A figura 2.6 mostra uma projeção perspectiva de um cubo vista por dois ângulos diferentes. Nesse caso, a projeção possui apenas um ponto de fuga, pois somente as linhas paralelas ao eixo z converge, e as linhas paralelas aos eixos x e y continuam paralelas.

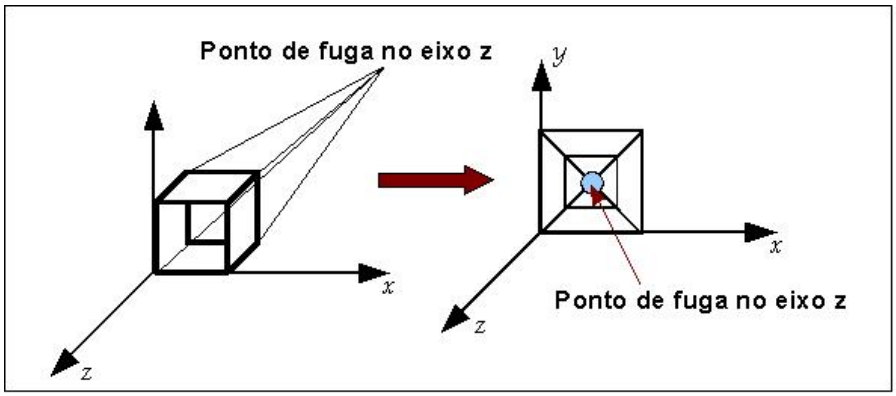

Figura 2.6: Projeções de um cubo com um ponto de fuga

Projeções perspectivas com dois pontos de fuga (quando dois eixos principais são interceptados pelo plano de projeção) são mais comumente usadas em arquitetura, engenharia, desenho publicitário e projeto industrial. Já as projeções perspectivas com três pontos de fuga são bem menos utilizadas, pois adicionam muito pouco em termos de realismo comparativamente às projeções com dois pontos de fuga, e são mais difíceis de serem construídas.

Supondo agora que, no caso de projeção perspectiva, o plano de projeção é normal ao eixo z, em $z = d$. Assim, seja o plano de projeção que se encontra a uma distância d da origem, e $\mathbf{P}$ o ponto que será projetado sobre ele. calcula-se o ponto $\mathbf{P}_p = (x_p, y_p, z_p)$, que é a projeção perspectiva de $\mathbf{P} = (x, y, z)$ sobre o plano de projeção em $z = d$ da seguinte forma:

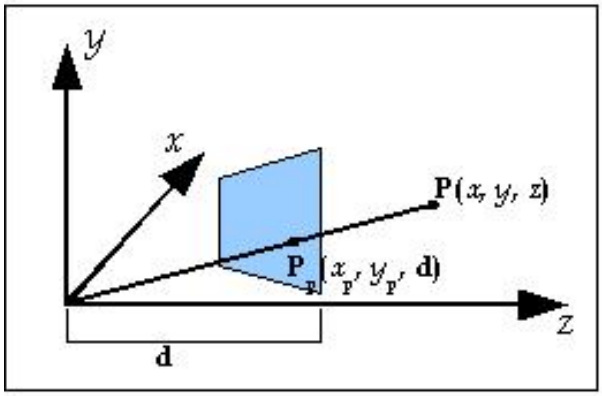

Figura 2.7: Cálculo da projeção perspectiva

Usando a semelhança de triângulos da figura 2.7, obtêm-se:

$$\frac{x_p}{d} = \frac{x}{z},\ x_p = \frac{dx}{z} = \frac{x}{z/d} \tag{2.1}$$

e

$$\frac{y_p}{d} = \frac{y}{z},\ y_p = \frac{dy}{z} = \frac{y}{z/d} \tag{2.2}$$

A distância d pode ser vista como o fator de escala que se aplica a x_p e y_p. Na projeção perspectiva, a divisão por z faz com que os objetos mais distantes do plano de projeção pareçam menores do que os que estão mais próximos. Além disso, z pode assumir qualquer valor (com exceção de $z = 0$).

Considerando agora a seguinte matriz, que é a matriz da projeção perspectiva:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}.$$

Multiplicando a matriz M pelo ponto $\mathbf{P}$ em coordenadas homogêneas, obtém-se um ponto $\mathbf{Q} = [x\, y\, z\, z/d]^T$:

$$\mathbf{Q} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = MP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Dividindo $\mathbf{Q}$ por z/d obtém-se a representação equivalente do ponto $\mathbf{Q}$:

$$\mathbf{Q}' = \begin{bmatrix} \frac{x}{z/d} \\ \frac{y}{z/d} \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}.$$

Logo, para obter as coordenadas cartesianas, usa-se:

$$(x_p, y_p, z_p) = \left(\frac{x}{z/d}, \frac{y}{z/d}, d \right)$$

Nota-se que a formulação deriva diretamente do resultado das equações 2.1 e 2.2, e a coordenada $z_p = d$ resulta da posição do plano de projeção sobre o eixo z (normal ao eixo z).

2.2.2 Projeção Paralela

A projeção paralela é uma visualização menos realística comparada à projeção perspectiva, a razão disto é que a projeção paralela não possui encurtamento perspectivo. No entanto, assim como na perspectiva, os ângulos são preservados somente nas faces do objeto que são paralelas ao plano de projeção.

As projeções paralelas são categorizadas em dois tipos, baseado na relação da direção de projeção e na *normal*[1] do plano de projeção. Nas projeções paralelas *ortográficas*, estas direções são as mesmas, enquanto que nas projeções paralelas *oblíquas* não são. Ou seja, na projeção ortográfica a direção da projeção é normal ao plano de projeção, enquanto que na projeção oblíqua, as projetantes são inclinadas em relação ao plano de projeção formando um ângulo α.

Para a projeção paralela, supõe-se que o plano de projeção é $z = 0$. Na projeção ortográfica, como a direção da projeção é a direção da normal ao plano de projeção, ela será o eixo z. Assim, o ponto $\mathbf{P}$ se projeta como:

$$x_p = x,\ y_p = y,\ z_p = 0.$$

Sua projeção é expressa pela matriz:

[1] A normal é um vetor que é perpendicular a um determinado plano

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

Escreve-se este resultado, em coordenadas homogêneas, como:

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Considerando agora a projeção oblíqua. Na figura 2.8 o cubo unitário é projetado no plano *xy*. O ponto $\mathbf{P}(0,0,1)$, que é um ponto atrás do cubo, é projetado no plano *xy*, como $\mathbf{P}'$.

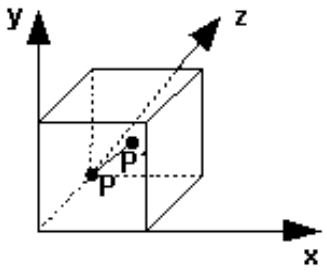

Figura 2.8: Projeção oblíqua de um cubo, $\mathbf{P}'$ é a projeção de $\mathbf{P}(0,0,1)$

A reta projetante, que não é perpendicular ao plano de projeção, deve passar por $\mathbf{P}$ e $\mathbf{P}'$, conforme podemos observar na figura 2.9. A direção desta reta projetante é dada por $\mathbf{P}' - \mathbf{P} = (l\cos\alpha, l\sin\alpha, -1)$ que faz um ângulo β com o plano xy.

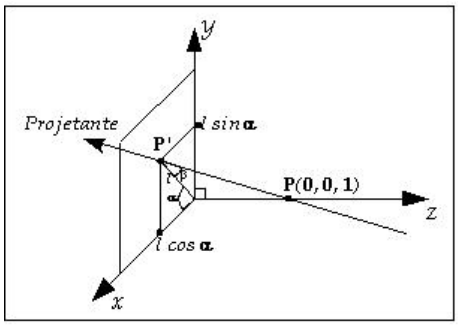

Figura 2.9: Projeção oblíqua de $\mathbf{P}(0,0,1)$ em $\mathbf{P}'(l\cos\alpha, l\sin\alpha, 0))$

Considerando agora um ponto genérico (x,y,z) e a sua projeção oblíqua dada por $(x_p, y_p, 0)$. As equações para os valores de projeção de x e y em função de z são obtidas por semelhança de triângulos como ilustrado na figura 2.10.

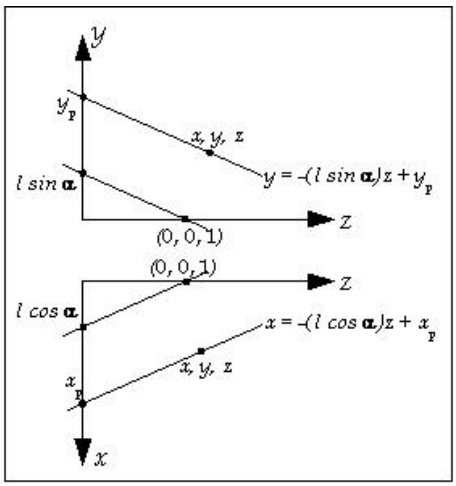

Figura 2.10: Projeção oblíqua de (x,y,z) em $(x_p, y_p, 0)$

$$\frac{x - x_p}{l\cos\alpha} = \frac{y - y_p}{l\sin\alpha} = \frac{z}{-1}.$$

Desta relação, obtêm-se:

$$x_p = x + zl\cos\alpha$$

e

$$y_p = y + zl\sin\alpha.$$

Logo, a matriz de projeção oblíqua será:

$$M = \begin{bmatrix} 1 & 0 & l\cos\alpha & 0 \\ 0 & 1 & l\sin\alpha & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

2.3 Iluminação

A iluminação é um recurso gráfico essencial para produzir a sensação de profundidade na visualização de uma cena tridimensional. O princípio da iluminação consiste em simular como os objetos refletem as luzes que incidem em sua superfície. É a interação entre luz e superfície que proporciona uma visualização mais realista de objetos tridimensionais (WOO, 1999).

Durante a interação da luz com um objeto, parte da energia luminosa é absorvida, parte é refratada, e parte é refletida pela sua superfície. A componente refletida é a responsável pela sensação de cor produzida no cérebro de um ser humano (LOPES, 2006). A construção de imagens realistas (que produzam a sensação de volume de objetos) baseia-se na análise desta interação entre fontes luminosas e superfícies.

A iluminação de um objeto pode ser classificada em dois tipos: **iluminação direta**, quando uma superfície recebe luz diretamente de uma fonte luminosa; e **iluminação indireta**, quando um objeto é iluminado por raios refletidos por outros objetos na cena. Em um mundo real, os objetos estão sujeitos sempre aos dois tipos de iluminação. A figura 5.6 ilustra as duas situações, onde raios luminosos partem diretamente da fonte luminosa para os pontos **P1** e **P2** (figura 5.6 **(a)**), e parte da luminosidade recebida por **P1** é refletida para o ponto **P2** (figura 5.6 **(b)**).

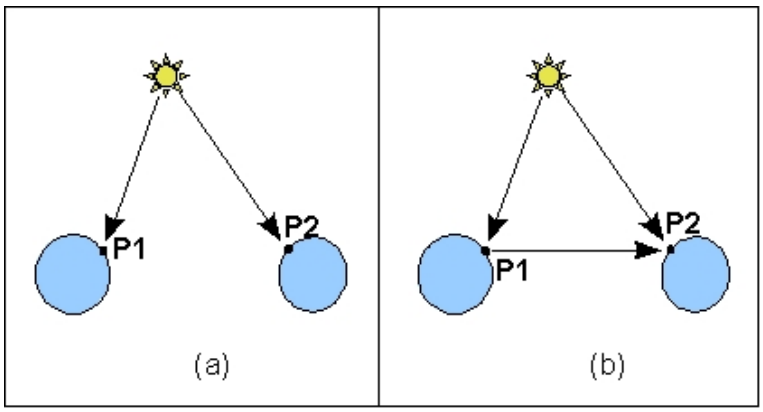

Figura 2.11: Formas de iluminação: **(a)** direta, e **(b)** indireta

A iluminação direta é relativamente simples de ser calculada, por depender unicamente da fonte de luz e do objeto iluminado. A luz indireta é muito mais complexa, pois é função do número de objetos da cena, afinal todos os objetos podem contribuir indiretamente para a iluminação de um único ponto da superfície de um objeto.

Segundo Watt (WATT, 1993), a modelagem de iluminação deve levar em consideração fatores relacionados à interação entre luzes e objetos. Os modelos de iluminação geralmente são divididos em: **modelos de fontes de luz**, que definem as características das luzes emitidas e sua propagação; e **modelos de reflexão**, que caracterizam onde e como diferentes tipos de superfícies reagem aos modelos de fontes luminosas.

2.3.1 Modelos de fontes de luz

A intensidade ou **luminância** da luz emitida por uma fonte é composta pelas intensidades de três componentes básicas:

$$\mathbf{I} = \begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix}$$

onde I_r, I_g e I_b são os componentes de intensidade das cores vermelha, verde e azul, respectivamente, e, em geral, caracterizadas por um valor real no intervalo [0,1]. Diferentes variações de cada uma destas componentes produzem diferentes sensações de cor na luz emitida por uma fonte ou refletida por uma superfície.

Esse trabalho utiliza um modelo composto por quatro tipos básicos de fontes luminosas: luz ambiente, fonte pontual, *spot* de luz e fonte distante. Esses tipos são suficientes para modelar cenas não muito complexas na qual os objeto são iluminados isoladamente por uma ou mais fontes de luz. Em cenas mais complexas é necessário levar em consideração as reflexões da luz entre os objetos presentes na cena.

- **Luz Ambiente**: geralmente oriunda de fontes largas que dissipam luz em todas as direções, possibilitando uma iluminação uniforme. A iluminação ambiente é caracterizada por uma intensidade

$$\mathbf{I}_a = \begin{bmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{bmatrix}$$

constante em todos os pontos da cena (embora diferentes superfícies possam refleti-la de formas distintas).

A utilização isolada da luz ambiente aplica uma coloração constante (ou seja, um vetor de intensidade constante) em todas as faces dos objetos não permitindo a visualização de profundidade do objeto na cena. Sua principal função é promover uma luminosidade em locais com pouca iluminação direta de outras fontes.

- **Fonte Pontual**: emite luz de mesma intensidade em todas as direções, a partir de um ponto específico, como mostra a figura 2.12.

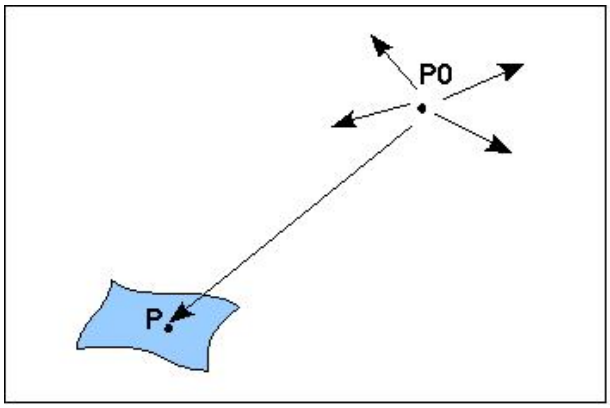

Figura 2.12: Fonte de luz pontual iluminando uma superfície

A intensidade luminosa de uma fonte pontual localizada em um ponto $\mathbf{p}_0$ pode ser escrita como:

$$\mathbf{I}(\mathbf{p}_0) = \begin{bmatrix} I_r(\mathbf{p}_0) \\ I_g(\mathbf{p}_0) \\ I_b(\mathbf{p}_0) \end{bmatrix}.$$

Embora constante em todas as direções, a intensidade luminosa sofre atenuação com a distância, sendo inversamente proporcional ao quadrado da distância entre a fonte

e o objeto iluminado. Desta forma, um ponto $\mathbf{p}$ da superfície recebe uma intensidade de luz de $\mathbf{p}_0$ dada por:

$$\mathbf{I}(\mathbf{p},\mathbf{p}_0) = \frac{1}{|\mathbf{p}-\mathbf{p}_0|^2}\mathbf{I}(\mathbf{p}_0).$$

Embora simples e intuitivo, este cálculo não apresenta bons resultados. Para distâncias pequenas, a intensidade recebida é muito amplificada, enquanto objetos relativamente distantes praticamente não recebem nenhuma luminosidade. Uma versão alternativa para o cálculo da atenuação é dada:

$$\mathbf{I}(\mathbf{p},\mathbf{p}_0) = \frac{1}{\left(\mathbf{c}_1+\mathbf{c}_2\mathbf{d}+\mathbf{c}_3\mathbf{d}^2\right)}\mathbf{I}(\mathbf{p}_0).$$

Onde $\mathbf{c}_1$, $\mathbf{c}_2$ e $\mathbf{c}_3$ são constantes definidas pelo usuário em função da atenuação desejada, e $\mathbf{d}$ é a distância entre a fonte de luz e o objeto.

- ***Spot* de Luz**: caracterizado por um feixe luminoso em formato de cone, emitido por uma fonte localizada em um ponto $\mathbf{p}_s$ (vértice do cone). O feixe tem direção dada por um vetor $\mathbf{l}_s$, e abertura angular 2γ, como mostra a figura 2.13. Para $\gamma = 180$, o *spot* se comporta como uma fonte pontual.

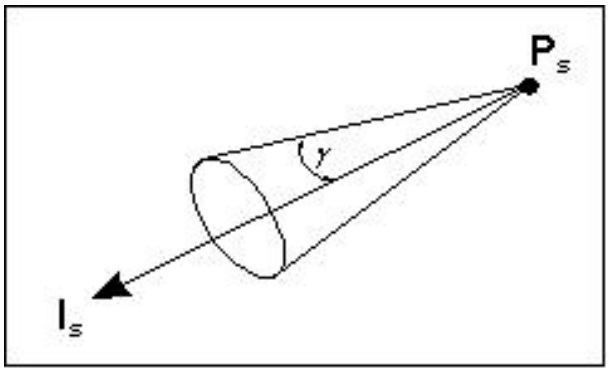

Figura 2.13: *Spot* de luz.

Pode-se definir a luminosidade recebida por um ponto $\mathbf{p}$ de uma superfície como uma função do ângulo ϕ entre a direção $\mathbf{l}_s$ do *spot* e a direção do vetor ($\mathbf{p}_s$,$\mathbf{p}$). Esta função usualmente é definida por $\cos^e \phi$. Desta forma, temos:

$$\mathbf{I}(\mathbf{p},\mathbf{p}_s) = \begin{bmatrix} I_r(\mathbf{p}_s) \cdot cos^e\phi \\ I_g(\mathbf{p}_s) \cdot cos^e\phi \\ I_b(\mathbf{p}_s) \cdot cos^e\phi \end{bmatrix} \quad \text{para} \quad \phi \in (-\gamma,\gamma),$$

sendo e uma constante maior que zero que determina o quão rapidamente a intensidade da luz decresce com $|\phi|$, como mostra a figura 2.14, para valores de ϕ no intervalo $[-\frac{\pi}{4},\frac{\pi}{4}]$.

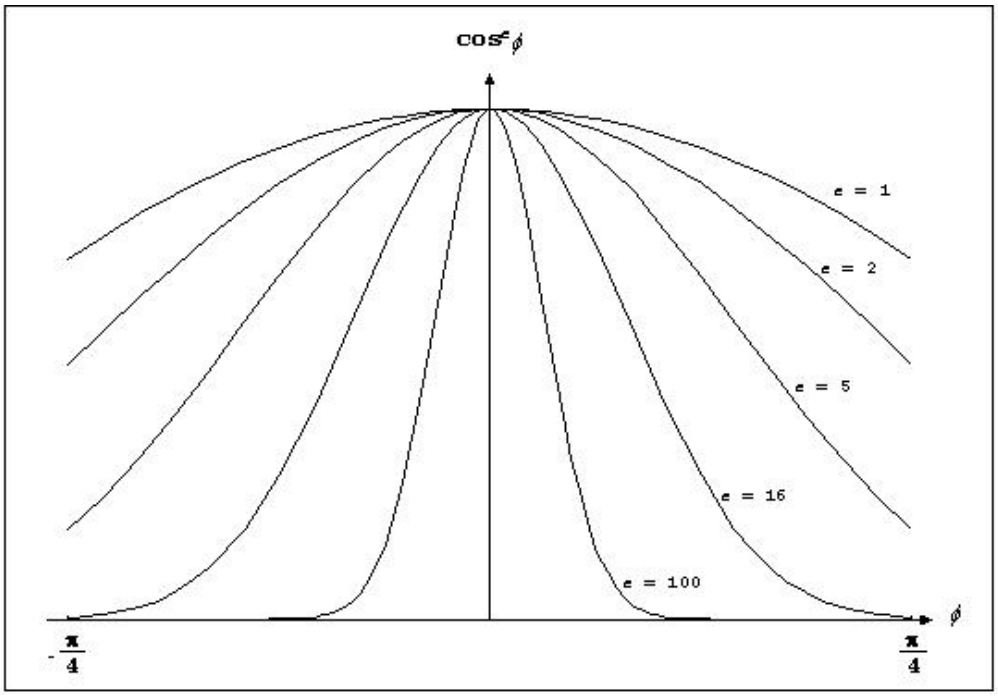

Figura 2.14: Intensidade do *spot* decrescente de acordo com o valor de e.

- **Fonte Distante de Luz**: também conhecida como fonte direcional, emite luz de direção e intensidade constantes. Cada fonte distante é caracterizada pelos raios paralelos que chegam nas superfícies da cena (ao invés de raios conicamente difusos como no *spot*. Deste modo, as superfícies com a mesma orientação da fonte, serão iluminadas da mesma forma, e as de orientação diferente não sofrerão essa iluminação.

  A fonte direcional é geralmente usada para simular uma fonte de luz distante como o sol. Ao contrário da luz pontual e de cone, as luzes direcionais não suportam o efeito de atenuação em relação à distância entre a luz e o ponto iluminado. A figura 2.15 ilustra uma fonte de luz distante.

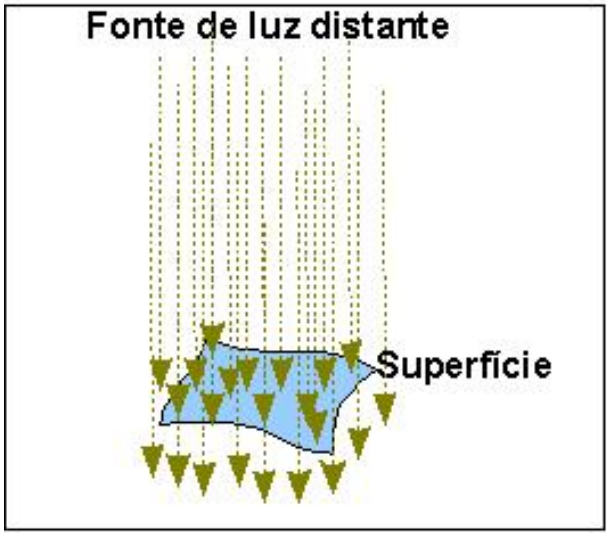

Figura 2.15: Fonte de luz distante.

2.3.2 Modelos de reflexão

Um modelo que represente a interação entre fonte luminosas e objetos deve levar em consideração diversos fatores, tanto da superfície iluminada: sua rugosidade, sua cor

e seu material; como das fontes luminosas: sua intensidade, seu ângulo de incidência e sua natureza. Desta forma, a qualidade das imagens geradas depende fortemente da capacidade do modelo de reflexão em incorporar estes fatores no processo de iluminação da cena.

É possível classificar basicamente três tipos principais de reflexão:

- reflexão ambiente: reflete a luz dispersa no ambiente que chega em todos os pontos de todos os objetos com intensidade constante e em todos os ângulos;
- reflexão difusa: reflete a luz incidente em todas as direções, independente da posição do observador;
- reflexão especular: reflete a luz com o mesmo angulo de incidência gerando pontos brilhantes na superfície do objeto.

A intensidade total de luz recebida por uma superfície é a soma das intensidades recebidas diretamente de fontes luminosas (iluminação direta), bem como de outras superfícies refletoras (iluminação indireta). De uma maneira geral, os modelos de reflexão podem ser classificados em locais ou globais. Enquanto um modelo de reflexão local trata somente a luz recebida diretamente de fontes luminosas, modelos globais consideram também as contribuições das reflexões de energia vindas de outros objetos sem luz própria, sendo estes mais complexos e de maior custo computacional do que os modelos locais (WATT, 1993). Exemplos de modelos globais de reflexão incluem a radiosidade e *ray-tracing* (GORAL, 1984).

Este estudo utiliza um modelo local que decompõe a reflexão em três tipos básicos: ambiente, difusa e especular.

Reflexão ambiente

Uma forma simples para um modelo local considerar as iluminações indiretas é definir uma iluminação ambiente. Essa componente adicional promove uma iluminação constante em toda a cena, representando as múltiplas reflexões de luz entre objetos como a luminosidade dispersa no ambiente. Todas as superfícies recebem a mesma iluminação ambiente, e devolvem parte desta intensidade luminosa em todas as direções.

O valor da intensidade luminosa refletida por essa componente em um dado ponto $\mathbf{p}$ de uma superfície, é dado pela equação:

$$\mathbf{I}_a(\mathbf{p}) = \begin{bmatrix} k_{ar} & 0 & 0 \\ 0 & k_{ag} & 0 \\ 0 & 0 & k_{ab} \end{bmatrix} \begin{bmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{bmatrix} = \mathbf{k}_a \mathbf{I}_a.$$

onde $\mathbf{I}_a(\mathbf{p})$ é a intensidade de luz ambiente refletida pelo ponto $\mathbf{p}$, e os valores I_{ar}, I_{ag} e I_{ab} determinam a intensidade de iluminação ambiente $\mathbf{I}_a$ presente na cena (frequências vermelha, verde e azul). O termo $\mathbf{k}_a$ é denominado de coeficiente de reflexão ambiente, composto pelas componentes k_{ar}, k_{ag} e k_{ab}, definidas no intervalo $[0, 1]$.

Estes coeficientes definem a atenuação da reflexão da luz ambiente, em função do material do objeto. Sua determinação é empírica, não tendo qualquer correspondência com as propriedades físicas reais dos materiais.

Reflexão difusa

Caracteriza-se pela reflexão de parte da luminosidade recebida em todas as direções. Uma superfície idealmente difusa reflete a mesma intensidade luminosa em todas as di-

reções, independente da posição do observador. Desta forma não existe a percepção de reflexo. Materiais foscos e sem brilho, como papel, giz, etc., são denominados refletores difusos ideais.

Em superfícies não ideais, a rugosidade não uniforme do material determina reflexões de diferentes intensidades em direções distintas. A figura 2.16 ilustra tal situação, onde os raios incidentes de direção $-\mathbf{l}$ são refletidos com variações de intensidade dependentes do tipo de rugosidade do material. Modelos de superfícies difusas ideais, como o utilizado neste trabalho, são mais simples e consomem menos cálculos computacionais.

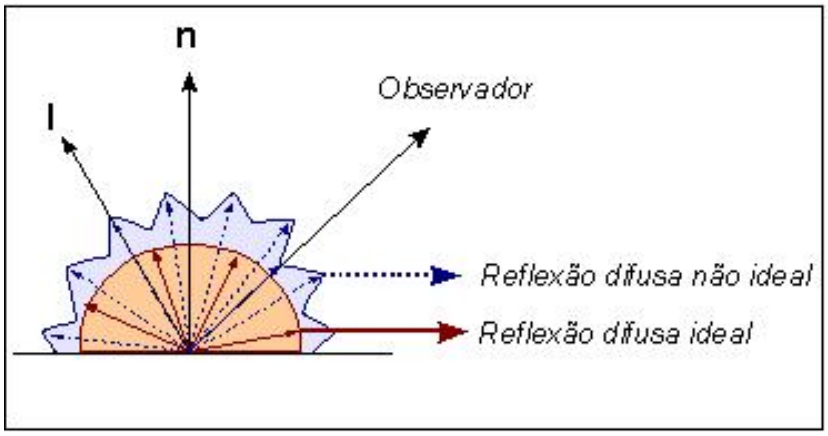

Figura 2.16: Modelo de reflexão difusa

Mesmo para superfícies ideais, o valor da intensidade luminosa refletida depende do ângulo entre a direção da incidência da luz e da superfície iluminada. A lei de Lambert estabelece que, considerando uma fonte pontual e um difusor perfeito, a intensidade da luz refletida é proporcional ao cosseno do ângulo $\boldsymbol{\theta}$ entre a direção da incidência da luz $\mathbf{l}$, e o vetor normal da superfície no ponto considerado, como mostra a figura 2.17.

Desta forma, o valor da intensidade luminosa refletida por um ponto $\mathbf{p}$, na forma difusa ideal, pode ser determinado pela equação:

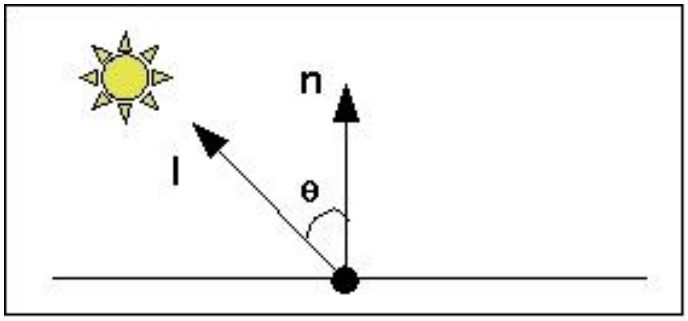

Figura 2.17: Lei de Lambert

$$\mathbf{I}_d(\mathbf{p},\mathbf{p}_0) = cos\theta \begin{bmatrix} k_{dr}(\mathbf{p}) & 0 & 0 \\ 0 & k_{dg}(\mathbf{p}) & 0 \\ 0 & 0 & k_{db}(\mathbf{p}) \end{bmatrix} \begin{bmatrix} I_r(\mathbf{p},\mathbf{p}_0) \\ I_g(\mathbf{p},\mathbf{p}_0) \\ I_b(\mathbf{p},\mathbf{p}_0) \end{bmatrix} = cos\theta\, \mathbf{k}_d(\mathbf{p})\, \mathbf{I}(\mathbf{p},\mathbf{p}_0).$$

Na equação acima, $0 \leq \theta \leq \frac{\pi}{2}$, e o coeficiente $\mathbf{k}_d(\mathbf{p})$ representa os coeficientes de reflexão para as frequências vermelha, verde e azul da superfície, todos definidos no intervalo [0,1]. Relebrando que o termo $\mathbf{I}(\mathbf{p},\mathbf{p}_0)$, definido anteriormente, representa a intensidade de luz incidente no ponto $\mathbf{p}$ proveniente do ponto $\mathbf{p}_0$, de componentes vermelha, verde e azul.

Modelo de Bougknight

Para simular a luminosidade indireta em modelos de iluminação local, Bougknight (BOUGKNIGHT, 1970) propôs um modelo que combinava as reflexões ambiente e difusa. Desta forma, a intensidade iluminosa $\mathbf{I}(\mathbf{p})$, refletida por um ponto $\mathbf{p}$ de uma superfície é dada pela equação:

$$\mathbf{I}(\mathbf{p}) = \mathbf{I}_a(\mathbf{p}) + \sum_i \mathbf{I}_d(\mathbf{p},\mathbf{p}_i) = \mathbf{k}_a\mathbf{I}_a + \mathbf{k}_d(\mathbf{p}) \sum_i cos\,\theta\ \mathbf{I}(\mathbf{p},\mathbf{p}_i),$$

que adiciona a reflexão da luminosidade ambiente à integração das contribuições de reflexão difusa, provenientes da iluminação direta de cada fonte luminosa.

Reflexão especular

Objetos polidos, como uma bola de bilhar ou superfícies metálicas, tendem a refletir a luminosidade gerando pontos brilhantes da mesma cor da fonte luminosa. Tal efeito caracteriza a reflexão especular, onde raios luminosos serão refletidos de uma maneira uniforme e simétrica, como ilustrado na figura 2.18.

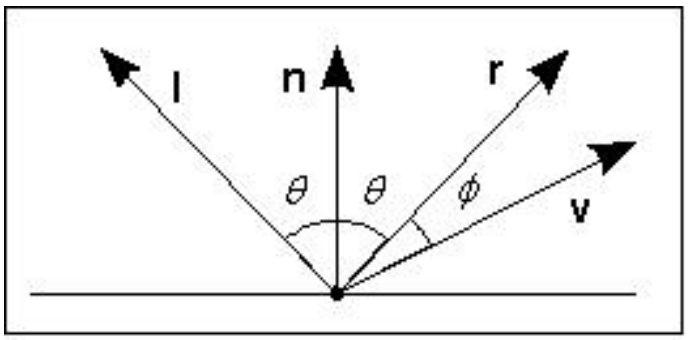

Figura 2.18: Reflexão especular

Observa-se que tanto o vetor de incidência **l** como o vetor de reflexão especular **r** formam o mesmo ângulo $\boldsymbol{\theta}$ com o vetor **n**, normal à superfície. O vetor **v** indica a posição relativa do observador, formando um ângulo $\boldsymbol{\phi}$ com o vetor **r**.

Somente superfícies refletoras ideais refletirão todos os raios paralelos (para vetores de incidência $-\mathbf{l}$) exatamente na mesma direção **r**. Neste caso, um observador somente perceberá os raios luminosos se os vetores **r** e **v** forem coincidentes, i.e. $\boldsymbol{\phi} = 0$.

Na prática, em superfícies bem polidas os raios concentram-se em uma vizinhança do vetor de reflexão especular **r** (cone preferencial de reflexão especular), como mostrado na figura 2.19(a) proporcionando um maior grau de reflexão especular. Superfícies menos

polidas (mais rugosas) tendem a divergir o feixe refletido, gerando mais reflexão difusa, como exemplificado na figura 2.19(b).

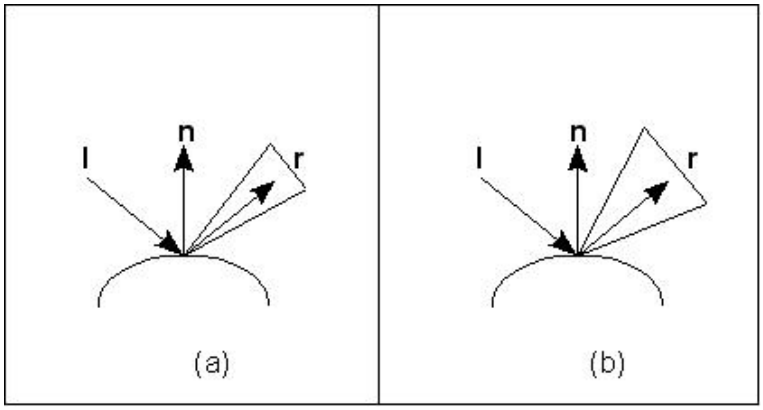

Figura 2.19: Reflexão Especular

Um modelo para incorporar a reflexão especular em iluminação de superfícies é dado pela fórmula:

$$\mathbf{I}_s(\mathbf{p}) = cos^{\alpha}\phi\, \mathbf{k}_s(\mathbf{p})\, \mathbf{I}(\mathbf{p}, \mathbf{p}_0).$$

Este modelo define a intensidade de reflexão especular $\mathbf{I}_s(\mathbf{p})$ como sendo proporcional a $cos^{\alpha}\phi$, com ϕ variando no intervalo $[-\frac{\pi}{2}, \frac{\pi}{2}]$. O valor de $\mathbf{k}_s(\mathbf{p})$ (*coeficiente de reflexão especular*) é determinado de forma empírica. O expoente α produz o efeito de atenuação da intensidade da luz refletida: quanto maior o expoente, maior o decaimento da intensidade para um mesmo ângulo ϕ, como mostra a figura 2.20.

Modelo de Phong

Phong (FOLEY, 1990) propôs um modelo que, além das componentes ambiente e difusa, incorpora também a reflexão especular, sendo conhecido como *Modelo Phong de*

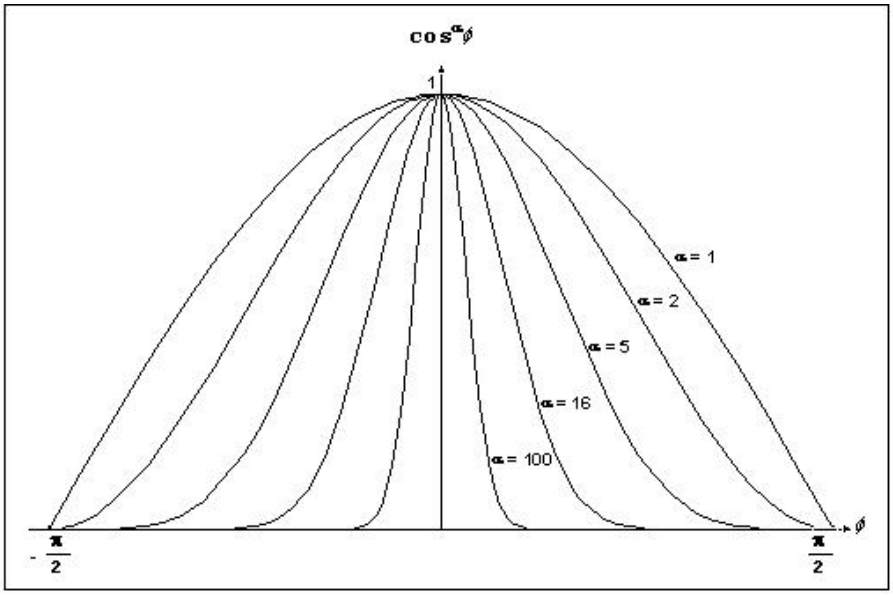

Figura 2.20: Variação do coeficiente de especularidade

Reflexão Especular. Um ponto $\mathbf{p}$ de uma dada superfície reflete com uma intensidade luminosa $\mathbf{I}(\mathbf{p})$, calculada na forma vetorial (componentes vermelho, verde e azul) por

$$\mathbf{I}(\mathbf{p}) = \mathbf{I}_a(\mathbf{p}) + \mathbf{I}_d(\mathbf{p}) + \mathbf{I}_s(\mathbf{p}) = \mathbf{I}_a\mathbf{k}_a(\mathbf{p}) + \mathbf{k}_d(\mathbf{p})\sum_i \mathbf{I}(\mathbf{p},\mathbf{p}_i)cos\,\theta + \mathbf{k}_s(\mathbf{p})\sum_i \mathbf{I}(\mathbf{p},\mathbf{p}_i)cos^{\alpha}\phi$$

Considerando os vetores $\mathbf{l}$, $\mathbf{n}$, $\mathbf{r}$ e $\mathbf{v}$ normalizados, é possível calcular os valores $cos\,\theta$ e $cos\,\phi$ utilizando o produto escalar entre eles:

$$cos\,\theta = \mathbf{l}\cdot\mathbf{n}$$
$$cos\,\phi = \mathbf{r}\cdot\mathbf{v},$$

o que resulta:

$$\mathbf{I}(\mathbf{p}) = \mathbf{I}_a\mathbf{k}_a(\mathbf{p}) + \mathbf{k}_d(\mathbf{p})\sum_i \mathbf{I}(\mathbf{p},\mathbf{p}_i)(\mathbf{l}\cdot\mathbf{n}) + \mathbf{k}_s(\mathbf{p})\sum_i \mathbf{I}(\mathbf{p},\mathbf{p}_i)(\mathbf{r}\cdot\mathbf{v})^{\alpha}$$

2.4 Modelos de Tonalização

Para o cálculo da tonalidade de um ponto de uma superfície, além da intensidade das fontes luminosas e propropriedades de reflexão do seu material, leva-se em consideração os ângulos de incidência de luz e de posicionamento do observador em relação ao vetor normal no ponto considerado.

Esse processo pode ser aplicado a cada ponto da superfície iluminada, o que pode ser considerado uma tarefa lenta para processamento gráfico interativo. Uma alternativa é decompor formas geométricas diversas em polígonos, aplicando a mesma tonalização para cada ponto do mesmo polígono. Dois fatores devem ser analizados para se obter um maior realismo: o grau de fragmentação de uma superfície, o que leva a um compromisso entre realismo e performance; e a tonalização nas vizinhanças dos polígonos. Os principais modelos de tonalização descritos na literatura são:

- ***flat*** ou Iluminação Constante: o modelo mais simples e eficiente, onde todos os pontos de um mesmo polígono serão igualmente iluminados (inclusive os de sua vizinhança). Isto decorre do fato de que o vetor normal é o mesmo para todos os pontos do polígono. Embora mais eficiente, este tipo de iluminação introduz um efeito denominado *match banding*, causado pela descontinuidade na intensidade da coloração nas vizinhanças entre polígonos adjacentes.

- ***Gouraud***: *Gouraud* propôs uma simplificação para uma maior continuidade da tonalização entre polígonos adjacentes. A idéia básica é calcular o vetor normal em um vértice comum a quatro polígonos adjacentes (dois a dois) como uma interpolação do vetor normal de cada um destes polígonos:

$$\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4}{|\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4|}.$$

- **Phong** propôs um novo modelo no qual as normais são interpoladas até mesmo em vértices interiores dos polígonos, o que elimina alguns problemas de perda de qualidade da reflexão especular existentes no modelo de *Gouraud*. Apesar de obter resultados melhores, o método de *Phong* acarreta em um aumento considerável no processamento, só atenuado por implementações em *hardware*, normalmente encontradas em equipamentos profissionais (ANGEL, 2004).

3 Curvas e Superfícies

Muitos objetos do mundo real apresentam formas ou contornos suaves. Para modelar tais objetos em computação gráfica, é necessário a utilização de curvas e superfícies suaves na sua representação computacional. Este tipo de modelagem é utilizado em diversas áreas como Medicina, Biologia, Química, na indústria do entretenimento, como o desenvolvimento de jogos em mundos virtuais, além de aplicações CAD/CAM[1]. A representação de uma curva como uma sucessão de trechos retos pode ser suficiente para várias aplicações. No entanto, curvas e superfícies complexas normalmente demandam uma maneira mais eficiente de representação. Tal representação é normalmente mais compacta que as formas discretas. Definir uma curva que passe por um conjunto determinado de pontos é um problema de **interpolação**, enquanto a definição de uma curva que passe próximo a um conjunto determinado de pontos é um problema de **aproximação**. Este capítulo irá tratar dos tipos de curvas mais comumente utilizadas como base para a construção de superfícies.

[1] **Computer Aided Design (CAD)**, ou desenho auxiliado por computador, é o nome genérico de sistemas computacionais (*software*) utilizados pela engenharia, geologia, arquitetura, e *design* para facilitar o projeto e desenho técnicos. O **Computer Aided Manufacturing (CAM)**, ou Manufatura Auxiliada por Computador, trabalham tendo como base modelos matemáticos provenientes do sistema CAD e estão ligados ao processo de produção.

3.1 Tipos de Representação

Existem diversas formas de representação de curvas e superfícies na Geometria e na Geometria Diferencial. Nestas disciplinas, fundamentalmente, destacam-se três formas de representação de curvas e superfícies: a explícita, a implícita e a paramétrica. Nesta seção apresentaremos rapidamente os principais conceitos concernentes a representação de curvas e superfícies na Geometria e na Geometria Diferencial.

3.1.1 Representação explícita

A representação explícita de uma curva ou superfície é possível quando ela pode ser expressa como o gráfico de uma função. Ou seja, como um conjunto dado por:

$$G = \{(x,y)|y = f(x)\} \subset \Re^m, m = n+1 \text{ ou } 3;$$

onde f é uma função da **variável independente** $x \in \Re^n, n = 1,2$.

Este tipo de representação não é adequado para os sistemas gráficos. Espera-se da representação de uma curva ou superfície, que, no mínimo, ela defina uma bijeção entre um aberto de um espaço euclidiano (no caso o $\Re^1$ ou o $\Re^2$) e parte da curva ou superfície que se quer representar. Isto é fundamental, por exemplo, para que seja possível identificar os pontos da superfície e mapear sobre eles as propriedades óticas da superfície ou curva em cada ponto. Em geral, é extremamente difícil achar os abertos e as formas funcionais explicitas que estabeleçam esta relação biunívoca cobrindo toda superfície, mesmo quando elas existem.

3.1.2 Representação implícita

Na representação implícita a superfície é representada pelos pontos do $(x,y) \in \Re^{n+m}, n+m = 2$ ou 3, que satisfazem a equação:

$$f(x,y) = 0 \in \Re^m, m = 1 \text{ ou } 2.$$

Por exemplo, uma reta ou uma esfera centrada na origem podem ser representadas, respectivamente, por:

$$ax + by + c = 0 \quad \text{e} \quad x^2 + y^2 + z^2 - r^2 = 0$$

Nas condições do teorema da função implícita existe sempre uma representação explícita local para uma curva ou superfície definida implicitamente. No entanto, curvas e superfícies no espaço tridimensional não são fáceis de ser representadas de forma implícita. As vantagens desta representação, quando ela é possível, são a possibilidade de representar curvas e superfícies infinitas, p.ex. um plano dado pela equação $ax + by + cz = 0$, e a facilidade para se avaliar se um ponto pertence ou não a uma curva ou a uma superfície.

3.1.3 Representação paramétrica

A representação paramétrica expressa, localmente, os valores das coordenadas de cada ponto de uma superfície, com relação ao espaço em que ela está imersa, como função de uma variável independente $\boldsymbol{u}$, denominada parâmetro, definida em um aberto $U \subset \Re^n$, onde n é a dimensão da superfície.

Por exemplo, em um espaço tridimensional, cada ponto de uma curva parametrizada pode ser representado por:

$$\mathbf{x}(u) = (x, y, z), \quad \text{onde } x = x(u), \ y = y(u), \ z = z(u), \quad u \in I = (a, b).$$

Na geometria diferencial, que usa o cálculo diferencial para estudar as propriedades das superfícies, restringe-se o estudos das superfícies às chamadas superfícies regulares, que são definidas como (CARMO, 1976):

Um subconjunto $S \subset \Re^3$ é uma superfície regular se, para cada $p \in S$, existe uma vizinhança V no $\Re^3$ e uma aplicação $\mathbf{x} : U \to V \cap S$ de um aberto $U \subset \Re^2$ sobre $V \cap S \subset \Re^3$ tal que:

1. $\mathbf{x}$ é C^∞;
2. $\mathbf{x}^{-1}$: $V \cap S \to U$ existe e é contínua como restrição de uma função contínua, definida num aberto do $\Re^3$ contendo $V \cap S$ e a valores no $\Re^2$;
3. $\forall q \in U$, a derivada de $\mathbf{x}$ no ponto q, $\mathbf{x}'(q)$:$\Re^2 \to \Re^3$, é injetiva.

Por esta definição, uma superfície pode ser representada por um conjunto de aplicações diferenciáveis com inversas contínuas em todo seus pontos. Portanto, cada ponto da superfície pode ser referenciado de maneira única pela escolha correta de uma destas aplicações. Os domínios destas aplicações podem ser tomados sempre como sendo o conjunto aberto $(0,1)$ ou $(0,1) \times (0,1)$, de acordo com a dimensão da superfície.

No caso das curvas (superfícies de dimenção 1) define-se o vetor velocidade por $\mathbf{x}'(q) = \frac{dx(q)}{d(q)} = (x'(q), y'(q), z'(q))$; ele é tangente à curva em cada ponto. Pode-se representar $\mathbf{x}(q)$ e $\mathbf{x}'(q)$, na base do espaço em que a curva está imersa, como um vetor coluna:

$$\mathbf{x}(q) = \begin{bmatrix} x(q) \\ y(q) \\ z(q) \end{bmatrix} \quad \text{e} \quad \frac{d\mathbf{x}(q)}{du} = \begin{bmatrix} \frac{dx(q)}{du} \\ \frac{dy(q)}{du} \\ \frac{dz(q)}{du} \end{bmatrix}.$$

Na representação das superfícies (dimensão 2) se $q = (u, v)$, temos:

$$\mathbf{x}(u, v) = (\, x(u, v),\, y(u, v),\, z(u, v)\,).$$

O plano tangente em cada ponto $p = \mathbf{x}(q)$ da superfície é determinado pelos vetores linearmente independentes tangentes à superfície $\frac{\partial \mathbf{x}}{\partial u}(q)$ e $\frac{\partial \mathbf{x}}{\partial v}(q)$. Na forma matricial:

$$\mathbf{x}(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix}, \quad \frac{\partial \mathbf{x}}{\partial u} = \begin{bmatrix} \frac{\partial x(u,v)}{\partial u} \\ \frac{\partial y(u,v)}{\partial u} \\ \frac{\partial z(u,v)}{\partial u} \end{bmatrix} \quad \text{e} \quad \frac{\partial \mathbf{x}}{\partial v} = \begin{bmatrix} \frac{\partial x(u,v)}{\partial v} \\ \frac{\partial y(u,v)}{\partial v} \\ \frac{\partial z(u,v)}{\partial v} \end{bmatrix}.$$

A representação paramétrica de curvas e superfícies é a forma mais flexível e robusta para sua modelagem computacional. Ela não apresenta as limitações inerentes às formas implícita e explícita, permitindo o uso da álgebra linear nos cálculos necessários para a concatenação de fragmentos, prolongando-os em curvas ou superfícies contínuas e suaves, como requerido em modelagem de objetos. Isto, em geral, é feito pela escolha dos pontos de junção e pela imposição de restrições aos vetores tangentes e normais de cada fragmento de curva ou superfície.

Uma questão importante, que será discutida a seguir, são os métodos de escolha da forma aproximada das funções paramétricas que são usadas em computação gráfica.

3.2 Curvas Parametrizadas Cúbicas

A modelagem de objetos envolve a definição de curvas e superfícies que obedeçam determinadas condições de posição, tangência e curvatura, expressas em um conjunto de pontos denominados pontos de controle. Grande parte dos modelos geométricos utilizados em computação gráfica são baseados em curvas e superfícies parametrizadas que utilizam funções polinomiais, cujas principais propriedades são (PERSIANO, 1996):

- Polinômios de grau n admitem n raízes r_i (alguns pares podem ser complexo conjugados), e podem ser expressos como:

$$p(u) = p_0(u-r_1)(u-r_2)\dots(u-r_{n-1})(u-r_n);$$

- Polinômios possuem derivadas de qualquer ordem que são também polinômios de grau menor;
- A integral indefinida de um polinômio de grau n é um polinômio de grau $n+1$;
- Polinômios de grau $n > 0$ são funções ilimitadas;
- Polinômios de grau n admitem no máximo $n-1$ extremos (máximos ou minimos) locais.

Um fator a ser considerado na representação paramétrica com funções polinomiais é a definição do grau dos polinômios. Quanto maior o grau, mais parâmetros serão necessários para definir o formato da curva, podendo torná-la mais oscilante, e exigindo mais processamento computacional. Por outro lado, polinômios de menor grau podem não fornecer parâmetros suficientes para o ajuste de sua curvatura. Uma solução comumente utilizada na computação gráfica é a utilização de polinômios cúbicos (de grau 3).

Uma curva paramétrica cúbica por partes é definida pela junção de vários segmentos de curvas cúbicas. A suavidade de toda a curva é obtida pela continuidade em seus pontos de junção e das derivadas nestes pontos.

Um segmento de curva paramétrica cúbica pode ser escrita como $\mathbf{p}(u) = (x(u),\ y(u),\ z(u))$ onde:

$$x(u) = c_{0x} + c_{1x}u + c_{2x}u^2 + c_{3x}u^3,$$

$$y(u) = c_{0y} + c_{1y}u + c_{2y}u^2 + c_{3y}u^3,$$

$$z(u) = c_{0z} + c_{1z}u + c_{2z}u^2 + c_{3z}u^3.$$

Na forma matricial o polinômio cúbico paramétrico é escrito da seguinte forma:

$$[x(u)\ y(u)\ z(u)] = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \cdot \begin{bmatrix} c_{0x} & c_{0y} & c_{0z} \\ c_{1x} & c_{1y} & c_{1z} \\ c_{2x} & c_{2y} & c_{2z} \\ c_{3x} & c_{3y} & c_{3z} \end{bmatrix}$$

ou

$$\mathbf{p}(u) = \mathbf{c}_0 + \mathbf{c}_1 u + \mathbf{c}_2 u^2 + \mathbf{c}_3 u^3 = \sum_{k=0}^{3} \mathbf{c}_k u^k = \mathbf{u}^T \cdot \mathbf{c}$$

onde

$$\mathbf{p}(u)^T = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{bmatrix}, \text{ sendo } \mathbf{c}_k = \begin{bmatrix} c_{kx} \\ c_{ky} \\ c_{kz} \end{bmatrix}^T$$

Portanto, a forma básica que expressa o segmento de curva polinomial parametrizada é dada pela equação

$$\mathbf{p}(u) = \mathbf{u}^T \cdot \mathbf{c},$$

que será usada para gerar os pontos do segmento de curva quando o parâmetro u varia no intervalo $(0,1)$.

Resta ainda determinar os coeficientes que compõem a matriz $\mathbf{c}$. Para funções polinonimais cúbicas são necessárias quatro condições, em geral expressas por:

$$\mathbf{p}(u_i) = \mathbf{u}_i^T \cdot \mathbf{c}, \qquad i = 0,1,2,3.$$

O vetor $\mathbf{u}_i$ é composto por quatro valores específicos do parâmetro u. Em geral, os valores $\mathbf{p}(u_i)$ são determinados pelos pontos de controle, correspondendo aos valores u_i escolhidos . Este conjunto de quatro equações corresponde a um sistema que pode ser escrito da seguinte forma:

$$\mathbf{p} = \mathbf{A} \cdot \mathbf{c} \quad \Rightarrow \quad \mathbf{c} = \mathbf{A}^{-1} \cdot \mathbf{p}.$$

Esta forma básica apresenta versões específicas nos vários tipos de curvas polinomiais estudadas a seguir.

3.2.1 Interpolação

Uma curva cúbica por partes **interpolada** passa por uma sequência $\{\mathbf{p}_k\}$ de pontos de controle, onde:

$$\mathbf{p}_k = \begin{bmatrix} p_{kx} \\ p_{ky} \\ p_{kz} \end{bmatrix}.$$

Como são necessárias quatro condições (equações) para estabelecer os coeficientes $\mathbf{c}$ de cada segmento de cúbica, eles serão determinados de forma que a curva polinomial $\mathbf{p}(u) = \mathbf{u}^T\mathbf{c}$ interpole quatro pontos sucessivos $\mathbf{p}_k$, $\mathbf{p}_{k+1}$, $\mathbf{p}_{k+2}$ e $\mathbf{p}_{k+3}$.

Para $u \in [0,1]$, uma possível interpolação poderia utilizar os valores

$$u_k = 0,\ \frac{1}{3},\ \frac{2}{3},\ 1. \tag{3.1}$$

$$\begin{aligned}
\mathbf{p}_0 &= \mathbf{p}(0) = \mathbf{c}_0 \\
\mathbf{p}_1 &= \mathbf{p}(\tfrac{1}{3}) = \mathbf{c}_0 + \tfrac{1}{3}\mathbf{c}_1 + \tfrac{1}{3}^2\mathbf{c}_2 + \tfrac{1}{3}^3\mathbf{c}_3 \\
\mathbf{p}_2 &= \mathbf{p}(\tfrac{2}{3}) = \mathbf{c}_0 + \tfrac{2}{3}\mathbf{c}_1 + \tfrac{2}{3}^2\mathbf{c}_2 + \tfrac{2}{3}^3\mathbf{c}_3 \\
\mathbf{p}_3 &= \mathbf{p}(1) = \mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3
\end{aligned}$$

Reescrevendo as equações na forma matricial, obtém-se o sistema: $\mathbf{p} = \mathbf{A}\mathbf{c}$, onde:

$$\mathbf{p} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} \quad \text{e} \quad \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \frac{1}{3} & (\frac{1}{3})^2 & (\frac{1}{3})^3 \\ 1 & \frac{2}{3} & (\frac{2}{3})^2 & (\frac{2}{3})^3 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

A matriz dos coeficientes $\mathbf{c}$ pode, então, ser obtida da seguinte forma:

$$\mathbf{p} = \mathbf{A} \cdot \mathbf{c} \quad \Rightarrow \quad \mathbf{c} = \mathbf{A}^{-1}\mathbf{p} = \mathbf{M}_I\mathbf{p}.$$

A matriz

$$\mathbf{M}_I = \mathbf{A}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -5.5 & 9 & -4.5 & 1 \\ 9 & -22.5 & 18 & -4.5 \\ -4.5 & 13.5 & -13.5 & 4.5 \end{bmatrix},$$

é chamada matriz de **interpolação geométrica**, relativa aos valores u_k em (3.1). Cada segmento de cúbica $\mathbf{p}(u) = \mathbf{u}^T\mathbf{c}$, $u \in [0,1]$, interpola quatro pontos sucessivos.

A interpolação de todos os pontos é composta por segmentos cúbicos que interpolam subseqüências de quatro pontos de controle, cada um determinando uma parte da curva cúbica por partes.

Para garantir a continuidade nos pontos de junção, os extremos de segmentos vizinhos devem ser coincidentes. Ou seja, o último ponto $\mathbf{p}_k$ de um segmento deve coincidir com o primeiro ponto do próximo segmento. A Figura 3.1 ilustra a interpolação composta por dois segmentos de cúbicas adjacentes. Embora a curva cúbica por partes formada seja contínua, a derivada nos pontos de junção pode não existir.

Pode-se avaliar a influência de um ponto de controle sobre qualquer trecho do segmento que o interpola. Para isso, reescreve-se a equação:

$$\mathbf{p}(u) = \mathbf{u}^T\mathbf{c} = \mathbf{u}^T\mathbf{M}_I\mathbf{p},$$

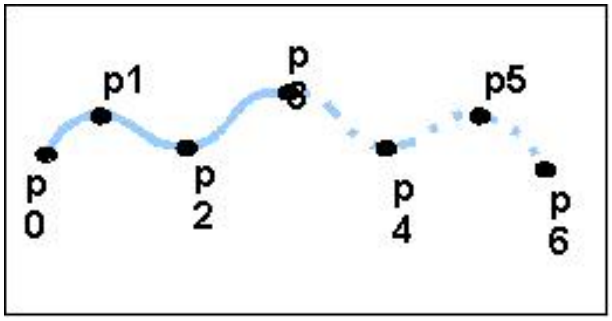

Figura 3.1: Junção de segmentos interpolados

ou

$$\mathbf{p}(u) = \mathbf{b}(u)^T \mathbf{p},$$

onde $\mathbf{b}(u)^T = \mathbf{u}^T \mathbf{M}_I$.

O vetor $\mathbf{b}(u) = (\mathbf{u}^T \mathbf{M}_I)^T = \mathbf{M}_I^T \mathbf{u}$ é um conjunto de funções cúbicas conhecidas como funções de mistura (*blending functions*):

$$\mathbf{b}(u) = \begin{bmatrix} b_0(u) \\ b_1(u) \\ b_2(u) \\ b_3(u) \end{bmatrix} = \begin{bmatrix} -4.5u^3 + 9u^2 - 5.5u + 1 \\ 13.5u^3 - 22.5u^2 + 9u \\ -13.5u^3 + 18u^2 - 4.5u \\ 4.5u^3 - 4.5u^2 + u \end{bmatrix}.$$

Reescrevendo $\mathbf{p}(u)$, obtém-se:

$$\mathbf{p}(u) = b_0(u)\mathbf{p}_0 + b_1(u)\mathbf{p}_1 + b_2(u)\mathbf{p}_2 + b_3(u)\mathbf{p}_3 = \sum_{i=0}^{3} b_i(u)\mathbf{p}_i.$$

Cada $b_i(u)$ expressa a influência do seu respectivo ponto de controle sobre a curva gerada. Pode-se depreeder da figura 3.2, que quando u distancia-se de $\frac{i}{3}$ (para $i = 0, 1, 2, 3$)

o ponto $\mathbf{p}_i$ diminui sua influência sobre a curva. As funções de misturas são polinômios de grau 3, portanto podem ser escritas como produtos envolvendo suas raízes. Observa-se que as raízes de cada b_i coincidem com os valores u_k definidos em (3.1).

$$
\begin{aligned}
b_0(u) &= -\tfrac{9}{2}(u-\tfrac{1}{3})(u-\tfrac{2}{3})(u-1),\\
b_1(u) &= \tfrac{27}{2}u(u-\tfrac{2}{3})(u-1),\\
b_2(u) &= -\tfrac{27}{2}u(u-\tfrac{1}{3})(u-1),\\
b_3(u) &= \tfrac{9}{2}u(u-\tfrac{1}{3})(u-\tfrac{2}{3}).
\end{aligned}
$$

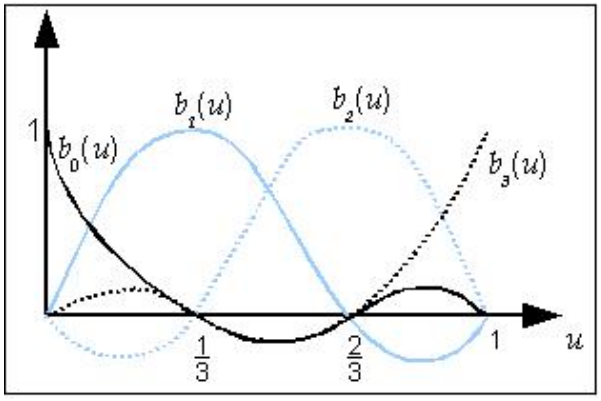

Figura 3.2: Interpolação dos polinômios das funções de mistura.

3.2.2 Curvas Hermite

Para resolver o problema da não existência da derivada nos pontos de junção dos segmentos de cúbica no método anterior, Hermite propôs usar apenas dois pontos de controle. As quatro condições necessárias para determinação dos coeficientes de $\mathbf{c}$ deveriam ser dadas pela interpolação do segmento de cúbica nos pontos de entrada e saída da curva, de forma que os vetores tangentes à curva neste pontos tivessem valores pré-estabelecidos. Notaremos $(\mathbf{p}_0, \mathbf{p}_3)$ e $(\mathbf{p}'_0, \mathbf{p}'_3)$, respectivamente, os pontos de entrada e saída do segmento de cúbica e a derivada nestes pontos.

Considerando $u \in [0,1]$ os pontos extremos são $\mathbf{p}(0)$ e $\mathbf{p}(1)$ e os vetores tangentes são determinados da seguinte forma:

$$\mathbf{p}'(u) = \begin{bmatrix} \frac{dx}{du} \\ \frac{dy}{du} \\ \frac{dz}{du} \end{bmatrix} = \mathbf{c}_1 + 2u\mathbf{c}_2 + 3u^2\mathbf{c}_3.$$

Logo, as condições de Hermite são:

$$\begin{aligned} \mathbf{p}_0 &= \mathbf{p}(0) = \mathbf{c}_0, \\ \mathbf{p}_3 &= \mathbf{p}(1) = \mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3, \\ \mathbf{p}'_0 &= \mathbf{p}'(0) = \mathbf{c}_1, \\ \mathbf{p}'_3 &= \mathbf{p}'(1) = \mathbf{c}_1 + 2\mathbf{c}_2 + 3\mathbf{c}_3, \end{aligned}$$

matricialmente representados por:

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_3 \\ \mathbf{p}'_0 \\ \mathbf{p}'_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \mathbf{c},$$

onde

$$\mathbf{q} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_3 \\ \mathbf{p}'_0 \\ \mathbf{p}'_3 \end{bmatrix} \quad \text{e} \quad \mathbf{A}_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix}.$$

A matriz dos coeficientes $\mathbf{c}$ é obtida resolvendo-se a equação: $\mathbf{c} = \mathbf{M}_H\mathbf{q}$, onde $\mathbf{M}_H$ é chamada matriz geométrica de Hermite

$$\mathbf{M}_H = \mathbf{A}_H^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}.$$

O segmento de cúbica é dada por $\mathbf{p}(u) = \mathbf{u}^T\mathbf{c}$, logo $\mathbf{p}(u) = \mathbf{u}^T\mathbf{M}_H\mathbf{q}$.

Na interpolação de Hermite, não só os pontos de junção de dois segmento de cúbica adjacentes são compartilhados, mas também o valor das derivadas nestes pontos. Para cada par de segmentos de cúbica adjacentes, o ponto de saída de um segmento e o vetor tangente neste ponto coincide com o ponto de entrada do próximo segmento e o vetor tangente neste ponto. Isto garante que a curva total seja de classe C^1. Ou seja, o controle preciso das tangentes de entrada e de saída dos segmentos da curva é essencial para a "suavização" da curva total. Para isso, os vetores tangentes de entrada e saída em cada ponto de controle devem possuir a mesma direção e módulo.

As funções de mistura são obtidas reescrevendo-se $\mathbf{p}(u)$ como:

$$\mathbf{p}(u) = \mathbf{b}(u)^T\mathbf{q}, \text{ onde}$$

$$\mathbf{b}(u) = \mathbf{M}_H^T \mathbf{u} = \begin{bmatrix} 2u^3 - 3u^2 + 1 \\ -2u^3 + 3u^2 \\ u^3 - 2u^2 + u \\ u^3 - u^2 \end{bmatrix}.$$

A figura 3.3 ilustra a continuidade da curva obtida pelo alinhamento das tangentes no ponto de união.

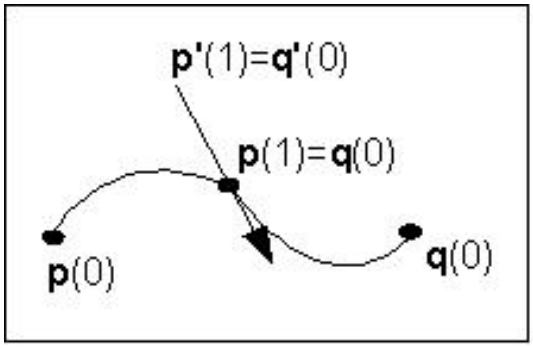

Figura 3.3: continuidade no ponto de junção de curvas Hermite.

3.2.3 Curvas Bézier

A curva de Bézier foi desenvolvida por Pierre Bézier durante seus trabalhos em projetos de automóveis para a *Renault* francesa no início da década de 1970. Bézier baseou sua curva no método proposto por Hermite, adicionando à definição original uma metodologia para a determinação das tangentes $\mathbf{p}'_0$ e $\mathbf{p}'_3$. Em seu método é usado mais dois pontos de controle $\mathbf{p}_1$ e $\mathbf{p}_2$, para obter uma aproximação linear de $\mathbf{p}'_0$ e $\mathbf{p}'_3$ definida da seguinte forma:

$$\mathbf{p}'_0 = \mathbf{p}'(0) \approx \frac{\mathbf{p}_1 - \mathbf{p}_0}{\frac{1}{3}} = 3(\mathbf{p}_1 - \mathbf{p}_0),$$
$$\mathbf{p}'_3 = \mathbf{p}'(1) \approx \frac{\mathbf{p}_3 - \mathbf{p}_2}{\frac{1}{3}} = 3(\mathbf{p}_3 - \mathbf{p}_2).$$

Através dessas aproximações, obtêm-se duas condições de controle, que, juntamente com as condições dos pontos extremos, determinam $\mathbf{p}(u) = \mathbf{u}^T\mathbf{c}$, onde:

$$\begin{aligned}
\mathbf{p}_0 &= \mathbf{p}(0) = \mathbf{c}_0 \\
\mathbf{p}_3 &= \mathbf{p}(1) = \mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3 \\
\mathbf{p}'_0 &= \mathbf{p}'(0) \approx 3\mathbf{p}_1 - 3\mathbf{p}_0 = \mathbf{c}_1, \\
\mathbf{p}'_3 &= \mathbf{p}'(1) \approx 3\mathbf{p}_3 - 3\mathbf{p}_2 = \mathbf{c}_1 + 2\mathbf{c}_2 + 3\mathbf{c}_3.
\end{aligned}$$

A matriz dos coeficientes $\mathbf{c}$ é obtida através da equação: $\mathbf{c} = \mathbf{M}_B\mathbf{p}$, onde $\mathbf{M}_B$ é chamada matriz geométrica Bézier, dada por:

$$\mathbf{M}_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}.$$

A polinomial cúbica de Bézier é dada por: $\mathbf{p}(u) = \mathbf{u}^T\mathbf{M}_B\mathbf{p}$.

As funções de mistura do método de Bézier, são polinômios de Bernstein, que se escrevem:

$$\mathbf{b}(u) = \mathbf{M}_B^T\mathbf{u} = \begin{bmatrix} (1-u)^3 \\ 3u(1-u)^2 \\ 3u^2(1-u) \\ u^3 \end{bmatrix}.$$

Os polinômios de Bernstein são dados por:

$$b_{id}(u) = \frac{d!}{i!(d-i)!}u^i(1-u)^{d-i}$$

e têm as seguintes propriedades:

- b_{id} não tem raízes no intervalo $(0,1)$.
- $0 < b_{id}(u) \leq 1,\ \forall u \in (0,1)$
- $\sum_{i=0}^{d} b_{id}(u) = 1,\ \forall u \in [0,1]$

A influência das funções de mistura na curva de Bézier são ilustradas na figura 3.4.

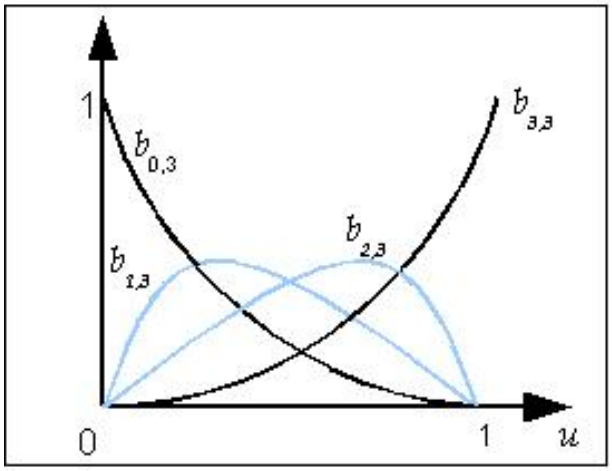

Figura 3.4: Polinômio de mistura que influencia a Bézier cúbica

A curva de Bézier pode ser escrita como: $\mathbf{p}(u) = \sum_{i=0}^{3} b_{i3}(u)\mathbf{p}_i$, onde b_{i3} são os polinômios de Bernstein de grau 3 e $\mathbf{p}_i$ são os pontos de controle.

Para cada $u \in [0,1]$, $\mathbf{p}(u)$ é uma combinação convexa dos quatro pontos de controle. Ou seja, a curva está contida no interior do fecho convexo definido por esses pontos, como mostra a figura 3.5.

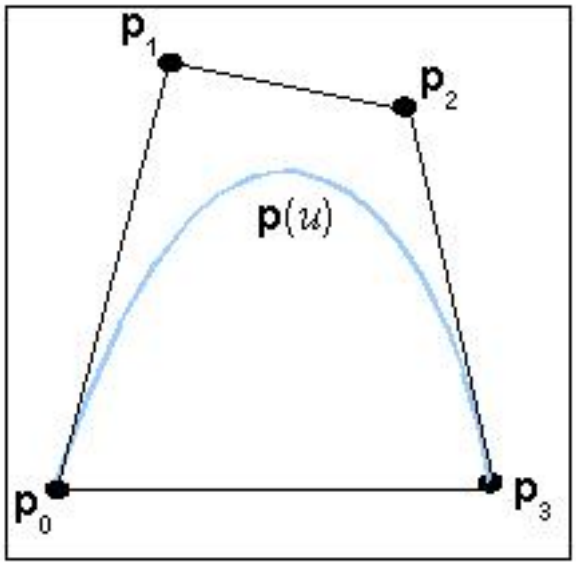

Figura 3.5: Curva de Bézier contida no fecho convexo formado por seus pontos de controle.

3.2.4 Curvas B-splines

O termo *spline*, em inglês, se refere originalmente às tiras de metal ou madeira que eram utilizadas nos projetos de embarcações, carros, etc. Pesos eram utilizados para ajustar a curvatura das *splines* às formas desejadas.

Inspirada nas *splines* físicas, a *Spline Cúbica Natural* é um polinômio cúbico por partes de classe C^2 que interpola todos os seus pontos de controle. Embora com suavidade maior do que as curvas Hermite e Bézier, sua principal desvantagem é que todos os pontos de controle devem ser considerados no cálculo dos coeficientes polinomiais: qualquer ajuste em um ponto de controle afeta toda a curva, exigindo muito cálculo computacional.

A B-Spline é uma versão simplificada da *Spline Cúbica Natural*, que implementa o controle local da curva. Qualquer alteração em um ponto de controle se propaga apenas para os vizinhos mais próximos. Entretanto, a B-Spline não interpola os pontos de

controle, apenas permite o ajuste de proximidade da curva gerada em relação a estes pontos.

O problema geral da *Spline Natural* é: dado um conjunto de pontos de controle $\{\mathbf{p}_i\}$, $i = 0, 1, ..., m$; deseja-se encontrar uma função suave

$$\mathbf{p}(u) = [x(u)\ y(u)\ z(u)], \quad u \in [u_{min}, u_{max}],$$

que aproxime, de alguma forma, dos pontos de controle. A partir de uma seqüência não decrescente de valores u_k, $k = 0, 1, ..., n$; denominados nós, tais que:

$$u_{min} = u_0 \leq u_1 ... \leq u_n = u_{max},$$

define-se então $\mathbf{p}(u)$ como uma função polinomial de grau d, entre cada par consecutivo de nós:

$$\mathbf{p}(u) = \sum_{j=0}^{d} c_{jk} u^j, \quad u_k < u < u_{k+1}.$$

Para definir a *spline* de grau d, é necessário calcular os $n(d+1)$ coeficientes c_{jk}. Desta forma, uma cúbica polinomial ($d = 3$) é definida por $4n$ coeficientes. Para calcular os coeficientes é necessário definir tantas condições quantos coeficientes exitem.

A B-Spline, assim como a curva Bézier, utiliza ajustadores (ou *blending functions*) no cálculo da curva gerada. Cada ajustador $B_{id}(u)$ expressa a influência do ponto de controle $\mathbf{p}_i$ na geração de toda a curva, dada por:

$$\mathbf{p}(u) = \sum_{i=0}^{m} B_{id}(u) \mathbf{p}_i.$$

Cada função $B_{id}(u)$ é um polinômio de grau d por partes no seu intervalo de atuação $(u_{i_{min}}, u_{i_{max}})$ e é igual a zero fora dele. O intervalo de atuação é determinado pelo grau d do polinômio, conforme veremos a seguir. O nome B-Splines vem do termo *basis splines*, porque para uma determinada seqüência de nós e um grau k fixado, as funções $B_{id}(u)$ formam uma base.

Um dos conjuntos de bases B-splines mais utilizadas e definida por recorrência é a proposta por **Cox-deBoor** (FOLEY, 1990):

$$B_{k,0}(u) = \begin{cases} 1, & u_k \leq u \leq u_{k+1} \\ 0, & \text{em outro caso} \end{cases} \tag{3.2}$$

$$B_{kd}(u) = \frac{u - u_k}{u_{k+d} - u_k} B_{k,d-1}(u) + \frac{u_{k+d+1} - u}{u_{k+d+1} - u_{k+1}} B_{k+1,d-1}(u).$$

A análise da recorrência acima permite um melhor entendimento sobre o papel de cada ponto de controle na geração da curva segmentada nos intervalos entre nós consecutivos.

A base da recorrência em (3.2); determina $B_{k,0}(u) = 1$ em (u_k, u_{k+1}), com valor nulo fora do intervalo, como ilustrado na Figura 3.6

Para $d = 1$, obtém-se:

$$B_{k,1}(u) = \frac{u - u_k}{u_{k+1} - u_k} B_{k,0}(u) + \frac{u_{k+2} - u}{u_{k+2} - u_{k+1}} B_{k+1,0}(u).$$

O que resulta em:

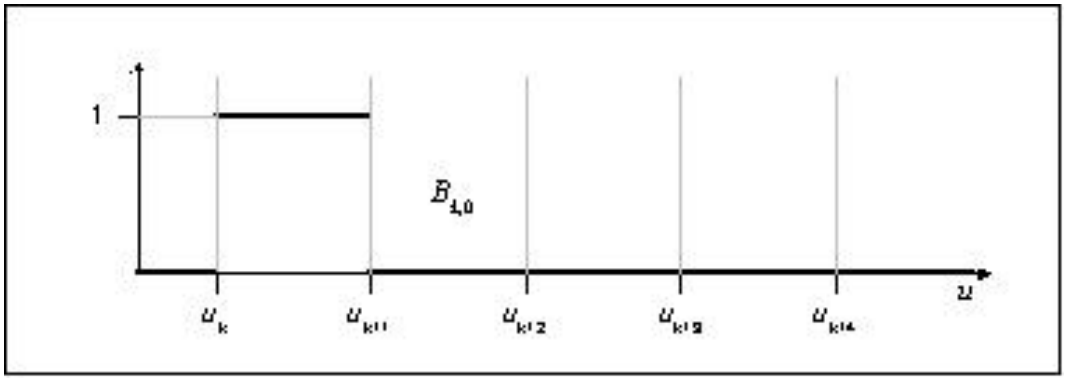

Figura 3.6: Ajustador (*Blending function*) B_{kd} para $d = 0$

$$B_{k,1}(u) = \begin{cases} \frac{u-u_k}{u_{k+1}-u_k}, & u_k \leq u \leq u_{k+1} \\ \frac{u_{k+2}-u}{u_{k+2}-u_{k+1}}, & u_{k+1} \leq u \leq u_{k+2} \\ 0, & \text{em outro caso} \end{cases}$$

A função $B_{k,1}(u)$ é ilustrada na Figura 3.7.

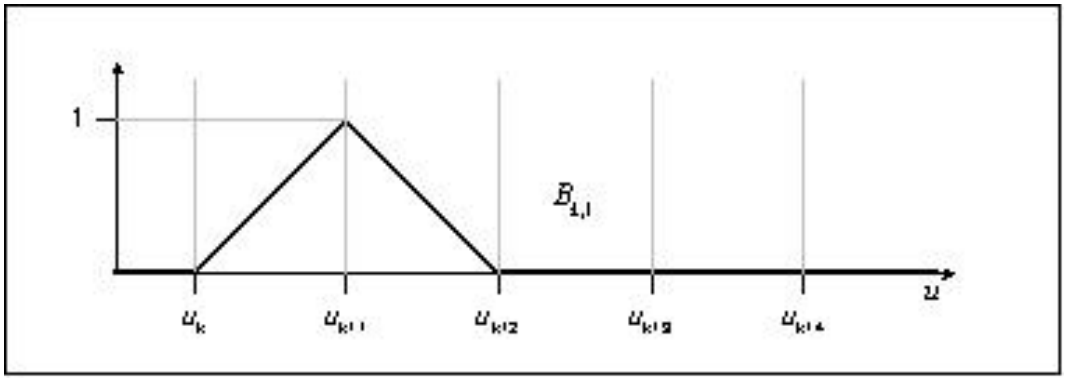

Figura 3.7: Ajustador (*Blending function*) B_{kd} para $d = 1$

Para $d = 2$:

$$B_{k,2}(u) = \frac{u-u_k}{u_{k+2}-u_k}B_{k,1}(u) + \frac{u_{k+3}-u}{u_{k+3}-u_{k+1}}B_{k+1,1}(u).$$

Resultando

$$B_{k,2}(u) = \begin{cases} \frac{u-u_k}{u_{k+2}-u_k} \cdot \frac{u-u_k}{u_{k+1}-u_k}, & u_k \leq u \leq u_{k+1} \\ \frac{u-u_k}{u_{k+2}-u_k} \cdot \frac{u_{k+2}-u}{u_{k+2}-u_{k+1}} + \frac{u_{k+3}-u}{u_{k+3}-u_{k+1}} \cdot \frac{u-u_{k+1}}{u_{k+2}-u_{k+1}}, & u_{k+1} \leq u \leq u_{k+2} \\ \frac{u_{k+3}-u}{u_{k+3}-u_{k+1}} \cdot \frac{u_{k+3}-u}{u_{k+3}-u_{k+2}}, & u_{k+2} \leq u \leq u_{k+3} \\ 0, \text{ em outro caso} \end{cases}$$

A função $B_{k,2}(u)$ é ilustrada na Figura 3.8.

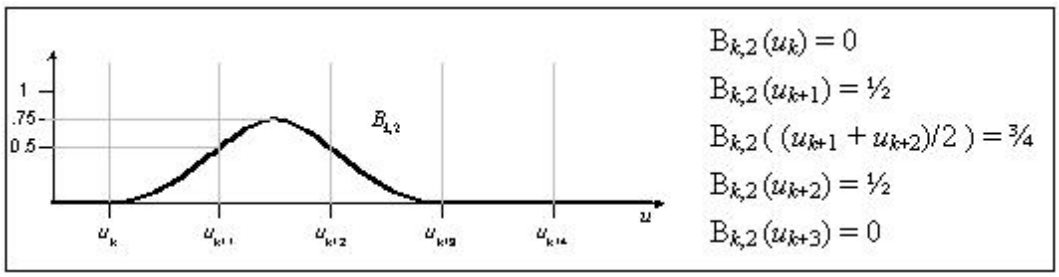

Figura 3.8: Ajustador (*Blending function*) B_{kd} para $d = 2$

Para $d = 3$, obtém-se:

$$B_{k,3}(u) = \frac{u-u_k}{u_{k+3}-u_k}B_{k,2}(u) + \frac{u_{k+4}-u}{u_{k+4}-u_{k+1}}B_{k+1,2}(u),$$

como ilustrado na Figura 3.9.

A figura 3.9 mostra a influência do ponto de controle $\mathbf{p}_k$, determinada pela função $B_{k3}(u)$, em quatro trechos consecutivos (de $[u_k, u_{k+1}]$ a $[u_{k+3}, u_{k+4}]$), da polinomial cúbica gerada. No caso geral, a geração de um polinômio de grau d determina a influência de cada ponto $\mathbf{p}_k$ sobre $d+1$ intervalos consecutivos, de $[u_k, u_{k+1}]$ a $[u_{k+d}, u_{k+d+1}]$.

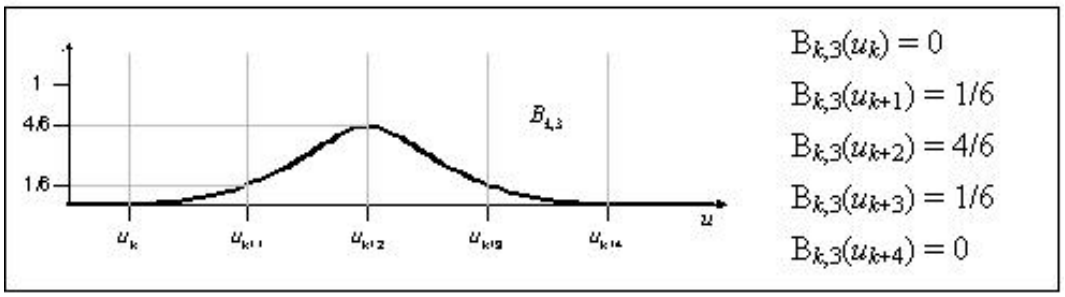

Figura 3.9: Ajustador (*Blending function*) B_{kd} para $d = 3$

Na polinomial cúbica, cada intervalo entre nós consecutivos é influenciado por quatro pontos de controle. A figura 3.10 ilustra tal situação, onde a geração da curva no intervalo $[u_k, u_{k+1}]$ leva em consideração quatro ajustadores consecutivos: $B_{k-3,d}$ a $B_{k,d}$.

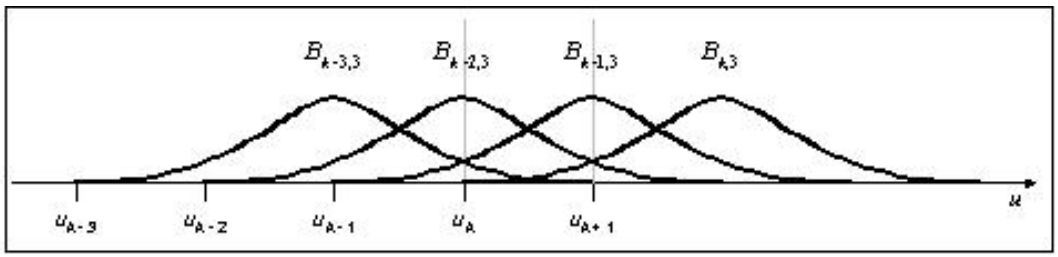

Figura 3.10: *Blending functions* consideradas entre dois nós consecutivos, $d = 3$

As bases B-splines de **Cox-deBoor** são de classe C^{d-1} nos nós e satisfazem a propriedade convexidade

$$0 \leq b_{id}(u) \leq 1 \quad \text{e} \quad \sum_{i=0}^{d} b_{id}(u) = 1, \ \forall u \in [0,1]$$

A figura 3.10 ilustra o caso em que os nós são igualmente espaçados pelo intervalo $[u_{min}, u_{max}]$; ou seja, todos os intervalos $[u_k, u_{k+1}]$, têm o mesmo valor $\Delta u = u_{k+1} - u_k$. As curvas em que todos os nós são igualmente espaçados são chamadas *splines* uniformes.

Pode-se também definir *Splines* não uniformes, que utilizam vetores de nós não uniformemente distribuídos. Esta flexibilidade permite até mesmo a definição de nós coincidentes ($u_k = u_{k+1}$). Neste caso, termos $0/0$ serão substituídos por 1. Por exemplo, o caso particular de uma *spline* com vetor de nós $u_k = \{0,0,0,0,1,1,1,1\}$ define uma curva de Bézier (ANGEL, 2004).

3.2.5 Curvas NURBS

As Curvas NURBS (Non Uniform Rational B-Splines), são assim chamadas por se tratarem de B-Splines não uniformes definidas por Funções Básicas Racionais, onde a cada ponto de controle é associado um valor real que determina o seu peso na geração da curva. Dessa forma, quanto maior o peso de um ponto de controle, mais influência este ponto exerce sobre o trecho da curva no qual atua.

Uma curva NURBS de grau d é definida por:

$$\mathbf{p}(u) = \frac{\sum_{i=0}^{n-1} w_i B_{i,d}(u)\mathbf{p}_i}{\sum_{i=0}^{n-1} w_i B_{i,d}(u)},$$

onde $\mathbf{p}_i$ são os pontos de controle da curva, $B_{i,d}(u)$ são as Funções Básicas B-Splines de grau d e cada valor real w_i é um peso associado ao ponto de controle $\mathbf{p}_i$. Observe-se que as B-Splines são um caso particular das NURBS, onde $w_i = 1$, $\forall\ i$.

Utilizando coordenadas homogêneas, uma curva NURBS definida em $\Re^3$ será representada como uma curva polinomial no espaço $\Re^4$. Pode-se incorporar o peso de cada ponto de controle em suas coordenadas homogêneas, definindo-se: $\mathbf{p}_i^w = w_i\mathbf{p}_i = (w_i x_i, w_i y_i, w_i z_i, w_i)$.

Dessa forma, a equação das NURBS será:

$$\mathbf{p}(u) = \frac{\sum_{i=0}^{n-1} B_{i,d}(u)\mathbf{p}_i^w}{\sum_{i=0}^{n-1} w_i B_{i,d}(u)}.$$

As curvas NURBS herdam todas as propriedades das curvas B-spline como envoltória convexa e serem de C^{d-1}. Além disso, incorporam a atribuição explícita de pesos aos pontos de controle. Este ajuste adicional tornam as NURBS uma poderosa ferramenta para modelagem de curvas e superfícies. A Figura 3.11 ilustra diferentes formas de uma curva, obtidas por variações do peso de um único ponto de controle.

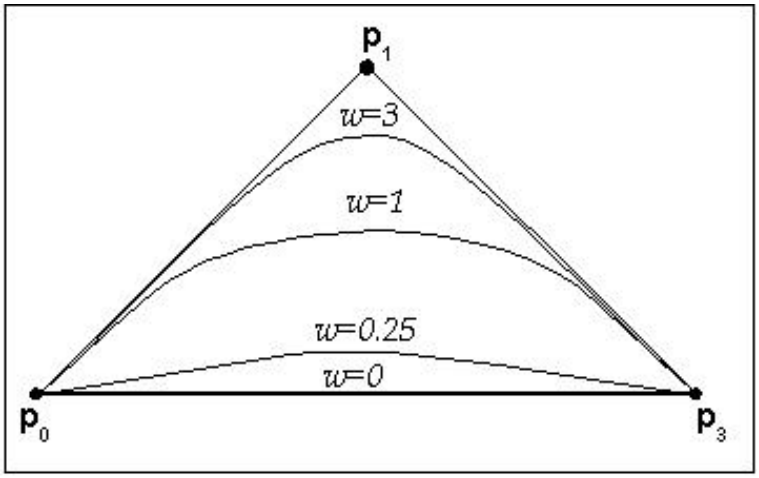

Figura 3.11: Ajustador (Curva NURBS com variação do peso associado ao ponto de controle $\mathbf{p}(1)$

Outra importante característica das NURBS trata da visualização em perspectiva. Determinadas operações como rotação, escala ou translação, podem ser obtidas aplicando-se transformações apenas em pontos específicos de uma forma geométrica. Por exemplo, pode-se rotacionar apenas os vértices de um cubo, e as novas coordenadas determinarão o novo cubo rotacionado. Tal processo é muito mais rápido do que rotacionar cada ponto do cubo previamente gerado.

Embora o mesmo processo se aplique a curvas e superfícies, a visualização em perspectiva para as B-Splines não segue o mesmo princípio: a aplicação da transformação

apenas nos pontos de controle não produz o mesmo resultado. Segundo Angel (ANGEL, 2004), as NURBS não apresentam esta limitação.

3.3 Superfícies Parametrizadas

Uma superfície polinomial parametrizada $\mathbf{p}(u,v)$ é dada por:

$$\mathbf{p}(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{c}_{ij} u^i v^j.$$

Ela é determinada por $3(n+1)(m+1)$ coeficientes e depende dos parâmetros u e v. Geralmente, define-se $n = m$ e $(u,v) \in [0,1]^2$, como ilustrado na Figura 3.12.

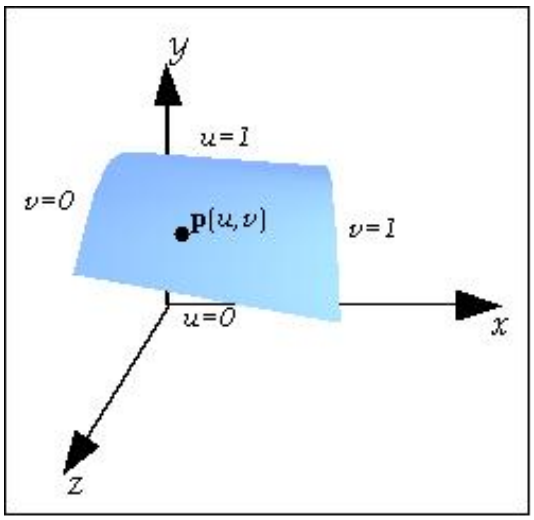

Figura 3.12: Trecho de superfície.

Qualquer superfície pode ser vista como uma coleção de curvas geradas fixando-se um dos parâmetros, u ou v, e variando-se o outro. Sob certas condições, pode-se gerar

superfícies a partir de curvas específicas (p.ex, polinomiais cúbicas), todas com as mesmas características. Nesse trabalho, como veremos a seguir, serão consideradas as superfícies de Bézier, as superfícies B-Splines e as superfícies NURBS, todas são superfícies paramétricas polinomiais por parte.

3.3.1 Superfícies de Bézier

Extendendo-se o conceito de curva de Bézier, pode-se definir uma superfície de Bézier a partir de uma matriz $n \times m$ de pontos de controle, $\mathbf{P} = [\mathbf{p}_{ij}]_{n \times m}$, e dos polinômios de *Bernstein* associados. Então,

$$\mathbf{p}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} b_i(u) b_j(v) \mathbf{p}_{ij}.$$

Fixando-se $u = u_0$, tem-se todos os $b_i(u_0)$ fixados, o que determina uma curva parametrizada em v. A curva $\mathbf{p}_{u_0}(v) = \mathbf{p}(u_0, v)$ é uma curva de Bézier denominada *curva isoparamétrica*.

As propriedades das superfícies de Bézier são semelhantes às das curvas de Bézier. Além de ter derivadas parciais contínuas, a superfície gerada está contida no poliedro determinado pela envoltória convexa dos pontos de controle. Quando $n = m = 3$, a superfície interpola os pontos $\mathbf{p}_{00} = \mathbf{p}(0,0)$, $\mathbf{p}_{03} = \mathbf{p}(0,3)$, $\mathbf{p}_{30} = \mathbf{p}(3,0)$ e $\mathbf{p}_{33} = \mathbf{p}(3,3)$. É possível interpretar as outras condições envolvidas na determinação dos $\mathbf{c}_{ij}$ das superífícies de Bézier como uma aproximação das derivadas parciais em seus vértices. Considerando o vértice $(0,0)$:

$$\mathbf{p}(0,0) = \mathbf{p}_{00},$$

$$\frac{\partial \mathbf{p}}{\partial u}(0,0) = 3(\mathbf{p}_{10} - \mathbf{p}_{00}),$$

$$\frac{\partial \mathbf{p}}{\partial v}(0,0) = 3(\mathbf{p}_{01} - \mathbf{p}_{00}),$$

$$\frac{\partial^2 \mathbf{p}}{\partial u \partial v}(0,0) = 9(\mathbf{p}_{00} - \mathbf{p}_{01} + \mathbf{p}_{10} - \mathbf{p}_{11}).$$

As três primeiras condições são extensões dos resultados para a curva de Bézier; a quarta condição pode ser vista como uma medida da tendência de curvatura da superfície na vizinhança do ponto $(0,0)$.

3.3.2 Superfícies B-Splines

Pode-se usar funções de base das curvas B-Splines para definir superfícies parametrizadas como:

$$\mathbf{p}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,j,d}(u,v)\mathbf{p}_{ij},$$

onde a *blending function* $B_{i,j,d}(u,v) = B_{i,d}(u)B_{j,d}(v)$ é definida como produto das funções de base das curvas B-Spline (Watt, 1993).

Além da matriz dos pontos de controle $\mathbf{p}_{ij}$, as superfícies B-Spline utilizam dois vetores de nós u_k e v_k, contendo sequências não decrescentes de valores para os parâmetros u e v (utilizadas nas funções de base para as curvas B-Splines).

Analogamente às curvas B-Spline, as sequências u_k e v_k determinam uma subdivisão de toda superfície gerada (*surface patches*), com sua continuidade C^2 sendo decorrente

do caso unidimensional. Tal como as curvas B-Spline, os valores dos nós podem ser irregularmente espaçados, originando superfícies não uniformes. Deve-se evitar pontos de controle duplicados, que criam descontinuidade.

Um exemplo simples de superfície B-Spline, determinada por 16 pontos de controle, é ilustrado na Figura 3.13.

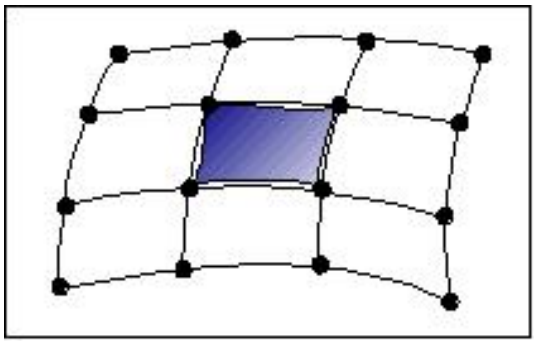

Figura 3.13: Trecho de superfície B-Spline

Nesse caso, a superfície gerada ocupa a região próxima aos quatro pontos de controle centrais. Devido a herança das propriedades da envoltória convexa e continuidade C^2, a superfície B-Spline é considerada mais suave que superfícies construídas a partir das curvas de Bézier e Hermite.

3.3.3 Superfícies NURBS

As superfícies NURBS apresentam características semelhantes às superfícies B-Splines.

Assim como as curvas NURBS, é possível definir superfícies a partir de uma malha de pontos de controle $\mathbf{p}_{i,j}$ com pesos $w_{i,j}$ associados, e utilizando coordenadas homogêneas. Uma superfície NURBS definida por curvas polinomiais em u e v (de graus p e q, respectivamente) e funções de base ($B_{i,p}(u)$ e $B_{j,q}(v)$) (WEISSTEIN, 2000), têm a forma geral:

$$\mathbf{p}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,p}(u)B_{j,q}(v)w_{i,j}\mathbf{p}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,p}(u)B_{j,q}(v)w_{i,j}}.$$

Essa equação pode ser reescrita como:

$$\mathbf{p}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{R}_{i,j}(u,v)\mathbf{p}_{i,j} \qquad 0 \leq u,v \leq 1$$

onde

$$\mathbf{R}_{i,j}(u,v) = \frac{B_{i,p}(u)B_{j,q}(v)w_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,p}(u)B_{j,q}(v)w_{i,j}}.$$

Utilizando coordenadas homogêneas, e definido-se $\mathbf{p}^w_{i,j} = w_{i,j}\mathbf{p}^w_{i,j}$, obtêm-se

$$\mathbf{p}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,p}(u)B_{j,q}(v)\mathbf{p}^w_{i,j}$$

sendo essa a fórmula mais utilizada nos algoritmos computacionais.

As Funções Básicas Racionais apresentam as mesmas propriedades das Funções Básicas B-Spline. Consequentemente, as propriedades das superfícies NURBS são as mesmas das superfícies B-Spline. É importante observar que as superfícies NURBS generalizam as superfícies de Bézier e superfícies B-Spline.

4 Animação por Modelos Físicos

O uso de modelos físicos na animação de superfícies deformáveis tem sido um importante campo de pesquisa na computação gráfica, pois possibilitam unir realismo e o controle intuitivo nas simulações estáticas e dinâmicas das cenas. Esses modelos incorporam conceitos, propriedades e as leis físicas da mecânica. Existem dois tipos de modelos físicos: os que usam a mecânica do contínuo para derivar as equações necessárias à animação dos objetos e os que consideram os objetos com um conjunto de pontos discreto e usam a mecânica dos sistemas de partículas para derivar as equações relevantes. Esse capítulo descreve os dois tipos de modelos físicos, dando ênfase ao modelo de sistemas de partículas que será o método utilizado na animação das superfícies em nosso sistema.

4.1 Modelo de Mecânica do Contínuo

Os modelos da mecânica do contínuo, consideram um objeto como um meio contínuo parametricamente representado. Numa região contínua em torno de cada ponto do objeto são impostos princípios de equilíbrio para obter as equações diferenciais parciais que governam este equilíbrio. Por menor que seja a vizinhança local de cada ponto, ainda assim será um elemento de dimensão maior do que o ponto ou a partícula, considerada elemento sem dimensão. Em geral, os modelos deformáveis que usam mecânica do contínuo são baseado em energia ou em força.

Modelos baseados em energia calculam a energia de todo objeto e derivam um conjunto de equações correspondendo a um estado de energia mínima.

Modelos baseados em força representam a força nos pontos por equações diferenciais e realizam uma integração numérica para obterem as posições dos pontos a cada momento (NG, 1996).

Um exemplo de modelo contínuo baseado em energia, são os modelos de deformação elástica desenvolvidos por Terzopoulos (TERZOPOULOS, 1988). Esse método tem sido muito utilizado em animação de vestimento (CARIGNAN, 1992) e outros tipos de superfícies elásticas (TURNER, 1993). Nesse caso, as forças são derivadas de funções de energia cinética e pontecial pela formulação lagrangeana (TERZOPOULOS, 1987):

$$\frac{\partial}{\partial t}(\frac{\mu \partial r(a,t)}{\partial t}) + \gamma \frac{\partial r}{\partial t} + \frac{\delta \varepsilon(r)}{\partial r} = f(r,t),$$

onde $r(a,t)$ é a posição da partícula a no tempo t, μ é a densidade de massa da partícula, γ é a constante de amortecimento, e $f(r,t)$ representa o conjunto das forças externas que atuam em r no instante t. A função $\varepsilon(r)$ é uma função de aproximação da energia potencial da deformação elástica do corpo.

Para resolver essa formulação são usadas soluções numéricas e discretização das equações, como as técnicas de discretização por diferenças finitas. A discretização dos sistemas contínuos pode ser equivalente a um sistema massa-mola não linear.

Um problema dos sistemas contínuos é em relação a sua estabilidade, embora o controle passo-a-passo adaptativo possa melhorar a estabilidade (TERZOPOULOS, 1987). A alta complexidade é também um problema, especialmente durante o desenvolvimento do sistema, e é um limitante para a velocidade dos cálculos.

4.2 Sistemas de Partículas

O sistema de partículas é uma coleção de pontos-massa onde o ambiente dinâmico é dado pelas forças que as partículas exercem uma sobre as outras (ação e reação) e pela forças externas exercidas sobre os pontos. A aplicação das leis da mecânica clássica a este sistema determina o conjunto de equações diferenciais ordinárias que regem o movimento. É possível usar sistemas de partículas para modelar objetos sólidos deformáveis através de um *array* tridimensional de partículas , "interligadas" por molas (sistema massa-mola). Quando o objeto é submetido à forças externas, as partículas se movem e suas posições influenciam no formato do objeto (ETZMUF, 1998). Num sistema de partículas, quando as partículas se movem influenciadas por forças externas, as forças internas procuram restaurar o sistema à posição inicial. As forças de amortecimento do meio, que são função da velocidade, também exercem influência no movimento das partículas e, consequentemente, nas deformações.

Em nossa formulação de superfícies deformáveis, dado uma NURBS, consideramos seus pontos de interpolação ou de controle como pontos de um sistema de partículas.

Para obter a equação do movimento das partículas, escrevemos a formulação lagrangeana da equação do sistema de partículas:

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_k} - \frac{\partial T}{\partial q_k} = Q_k \qquad k \in 1,...,l,$$

onde l é o grau de liberdade para o sistema; $q_1,...,q_l$ são as coordenadas generalizadas; $T = \frac{1}{2}m\dot{q}_k^2$ é a energia cinética e Q_k são as forças generalizadas correspondentes a q_k.

Quando as partículas possuem três graus de liberdade ($l = 3$),

$$Q_i = F_i + P_i - c\dot{q}i - K(q_i - q_{i,0}), i \in 1, ..., 3$$

onde F_i é a soma das forças externas exercidas na partícula e P_i é a soma das forças internas aplicadas na partícula pelas demais partículas vizinhas, K é o coeficiente de *Hook*, presente na força elástica que mantém a integridade do sistema. O termo c é o coeficiente de amortecimento e $q_{i,0}$ é a posição inicial do equilibrio da partícula.

Com isso, obtêm-se a equação do movimento da partícula, através da equação:

$$m\ddot{q}_i + c\dot{q}_i + Kq_i = F_i + P_i + Kq_{i,0},$$

considerando $\{m, c, K\} \neq 0$, obtêm-se uma equação diferencial linear e não homogênea de segunda ordem com coeficientes constantes. Obtem-se uma solução analítica da equação do movimento das partículas quando considera-se os coeficientes e a direção e magnitude das forças P_i e F_i constantes. No caso não constante, a solução da equação é, em geral, obtida por métodos numéricos.

Na animação das superfícies deformáveis nesse trabalho, usamos somente a solução analítica da equação do movimento das partículas.

4.2.1 Solução da Equação do Movimento

Dada a equação do movimento de partículas:

$$m\ddot{q}_i + c\dot{q}_i + Kq_i = F_i + P_i + Kq_{i,0},$$

definindo

$$S_i(t) = F_i + Pi + Kq_{i,0}$$

e substuindo na equação do movimento de partículas, temos que:

$$m\ddot{q}_i + c\dot{q}_i + Kq_i = S_i. \tag{4.1}$$

Quando $S_i(t) = 0 \forall t$, temos:

$$m\ddot{q}_i + c\dot{q}_i + Kq_i = 0. \tag{4.2}$$

Obtem-se a equação auxiliar, substituindo-se na equação (4.2) $\ddot{q}_i$ por x^2, $\dot{q}_i$ por x e q_i por 1. Como a equação auxiliar é quadrática, ela possui duas raizes reais distintas, ou duas raizes reais iguais ou duas raizes complexas conjugadas.

Usando as raizes r_1 e r_2 da equação auxiliar para a solução da equação homogênea (4.2), encontra-se duas soluções independentes. Nesse caso, a solução analítica da equação é dada por:

$$\begin{array}{lll} r_1 \neq r_2; & q_i = C_1 e^{r_1 t} + C_2 e^{r_2 t} & i = 1,2,3 \\ r_1 = r_2 = r; & q_i = (C_1 + C_2) e^{rt} & i = 1,2,3 \\ r_1 = a + bi, r_2 = a - bi; & q_i = e^{at}[C_1 cos(bt) + C_2 sen(bt)] & i = 1,2,3 \end{array}$$

A solução geral da equação não-homogenea é obtida pela soma da solução geral da equação homogênea com uma solução particular da equação não-homogenea.

Para as simulações realizadas neste trabalho, considerou-se a equação do movimento não-homogênea, sendo que as partículas sofrem influência não só da aceleração e da velocidade, como também são aplicado à elas forças externas e internas que influenciam na deformação e animação da superfície.

5 Simulação

Este capítulo apresenta a estrutura básica do sistema de simulação desenvolvido para a visualização de deformação de superfícies, bem como os resultados obtidos. São apresentadas as técnicas de programação utilizadas para a visualização gráfica e animação de superfícies. A aplicação dos principais conceitos discutidos neste trabalho foi implementada utilizando-se a biblioteca gráfica OpenGL, que também será exposta neste capítulo.

5.1 Características do Simulador

O principal objetivo do sistema implementado é a simulação visual da deformação de superfícies em movimento. Este tipo de animação pode ser aplicado em diversas situações, tais como: o decaimento de tecidos de pano, animações de expressões faciais e figuras humanas, movimento da superfície de oceanos, etc.

O programa permite a construção de um ambiente tridimensional composto por um mundo, uma câmera de visualização e fontes luminosas, implementando primitivas gráficas que proporcionam uma visão mais realista dos objetos inseridos na cena. Além disso, opções como tipos de projeção e iluminação, bem como um controle manual ou automatizado da animação, podem ser configuradas pelo usuário.

Para a escolha de um subconjunto de todas as funcionalidades gráficas estudadas, o projeto do sistema levou em consideração alguns requisitos básicos: interface de fácil

comunicação, recursos para configuração do programa, e também uma estruturação que permitisse sua fácil compreensão e reutilização. Para atender a esses requisitos, foi utilizada a metodologia orientada a objetos, implementada na linguagem C++, em conjunto com a biblioteca gráfica OpenGL.

O sistema foi desenvolvido com o compilador do ambiente Microsoft Visual Studio.NET 2005, para a execução no sistema operacional Windows XP. Entretanto, por ser uma aplicação totalmente portável, o software pode ser compilado em qualquer outro compilador C++, juntamente com a biblioteca OpenGL, independente do Sistema Operacional. Para um melhor desempenho, é necessário um ambiente dotado de placa gráfica 3D que implemente primitivas OpenGL.

5.2 OpenGL

A biblioteca OpenGL é uma interface de software para utilização em hardware gráficos. Essa interface é composta por cerca de 150 funções distintas, utilizada para especificar objetos e operações necessárias para produzir aplicações 3D interativas. Um exemplo de cena criada com a OpenGL é ilustrada na figura 5.1.

A OpenGL foi desenvolvida para ser independente de hardware, sendo portável para inúmeras plataformas existentes. Para conseguir tal objetivo, a OpenGL não define nenhuma função para controle de janelas ou tratamento de entrada de dados; o programador deve utilizar as facilidades do sistema operacional para implementar tais funcionalidades.

Utilizando a OpenGL, o programador precisa construir os objetos da cena utilizando um pequeno conjunto de primitivas gráficas disponíveis: pontos, linhas e polígonos. Para cada primitiva pode ser especificada uma cor ou definido o tipo de material que irá refletir automaticamente a iluminação do ambiente. Além disso, funções específicas implementam

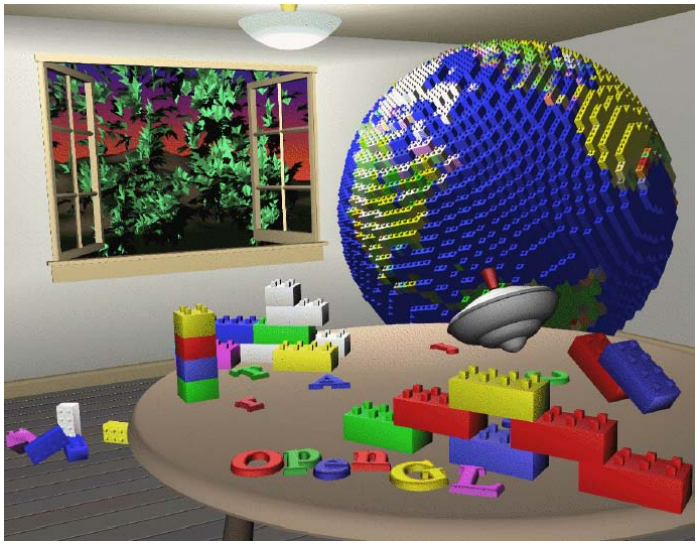

Figura 5.1: Exemplo de aplicação de OpenGL em uma cena 3D (Woo, 1999).

transformações lineares, tais como: rotações e translações, para o posicionamento de objetos na cena ou do observador (COHEN, 2006).

5.2.1 A máquina de estados da OpenGL

A OpenGL funciona, basicamente, como uma máquina de estados. O programador define determinadas configurações, cujos estados ficarão ativos até que novas funções reconfigurem o ambiente. Um exemplo de variável de estado é a cor de desenho: Todas as primitivas gráficas serão desenhadas com a cor atual até que uma nova cor seja selecionada (HILL, 2000).

A cor atual é apenas uma das muitas variáveis de estado definidas pela OpenGL. Outras controlam, por exemplo, o modo de desenho dos polígonos, propriedades das luzes de uma cena, transformações de projeção, etc. Muitas das variáveis de estado são habilitadas e desabilitadas utilizando os comandos `glEnable()` e `glDisable()`.

Duas importantes variáveis de estado são as matrizes utilizadas para as transformações referentes à visualização: a matriz de projeção (`mode = GL_PROJECTION`) e a matriz de visualização (`mode = GL_MODELVIEW`) do modelo. A função `glMatrixMode(mode)` indica qual delas estará ativa para as modificações subseqüentes. Desta forma, toda função de manipulação de matrizes implementa uma transformação específica sobre a matriz ativa, e as chamadas sucessivas efetuam concatenação de transformações. Funções adicionais permitem ainda: multiplicar a matriz ativa por outra, empilhar a matriz ativa, e restaurá-la da pilha quando necessário. Por fim, a chamada `glLoadIdentity()` restaura a matriz ativa com a matriz identidade.

5.2.2 Projeções 3D

Para mostrar uma cena 3D construída em OpenGL é necessário definir um *volume de visualização*, que além de especificar o tipo de projeção, também determina quais objetos ou suas partes estarão presentes na imagem final.

A OpenGL disponibiliza funções específicas para as projeções ortogonal (paralela) e perspectiva, que devem ser chamadas depois de `glMatrixMode(GL_PROJECTION)` e `glLoadIdentity()`. Na projeção ortogonal, o volume de visualização é um paralelepípedo, especificado pelos argumentos passados na chamada da função `glOrtho`:

`glOrtho(xmin, xmax, ymin, ymax, zmin, zmax)`.

Na projeção perspectiva, um tronco de pirâmide (cujo topo foi seccionado por um plano paralelo à sua base) define o volume de visualização para os objetos em seu interior, com o vértice superior da pirâmide definindo a posição do observador. A projeção perspectiva é ativada através da chamada da função

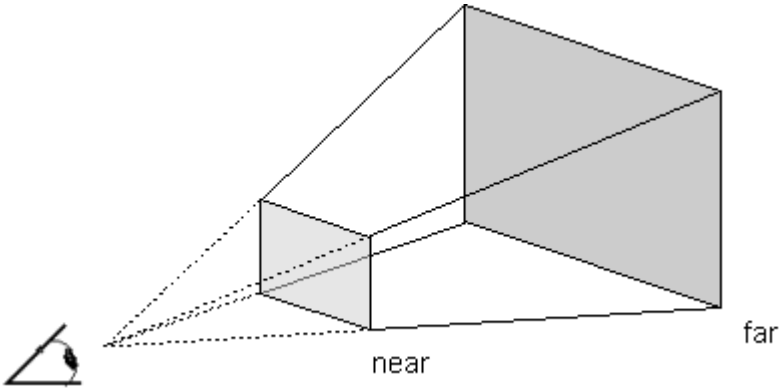

Figura 5.2: Planos de corte do tronco da pirâmide

```
gluFrustrum(xmin, xmax, ymin, ymax, zmin, zmax)
```

A figura 5.2 ilustra o volume de visualização para a projeção perspectiva, determinado pelos parâmetros passados, cujos vértices estão expressos em relação ao sistema de coordenadas da câmera de visualização.

A projeção perspectiva é muito utilizada em simulações visuais, animação e outras aplicações que necessitam de um certo grau de realismo, por melhor se assemelhar com a forma de visualização do olho humano.

Importante ressaltar que tanto `glOrtho()` quanto `gluFrustrum()` efetuam transformações sobre a matriz de visualização ativa.

5.2.3 Primitivas Gráficas

Cada objeto geométrico que compõe uma cena tridimensional é descrito em OpenGL por uma série de coordenadas de seus vértices, especificados por sucessivas chamadas à função `glVertex3f(x,y,z)` ou alguma de suas variantes. Todas as chamadas `glVertex3f()` são encapsuladas por um par `glBegin(mode)` e `glEnd()`.

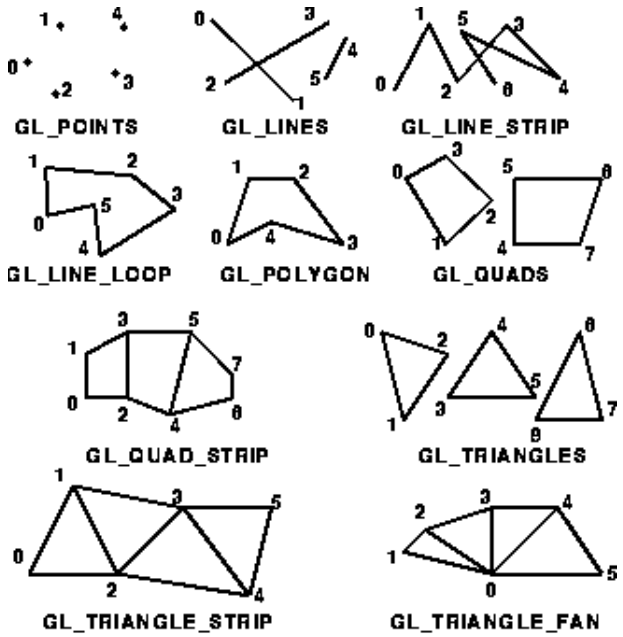

Figura 5.3: Primitivas gráficas em OpenGL

O parâmetro `mode` da função `glBegin()` especifica o tipo de objeto a ser desenhado. As várias constantes disponíveis pela OpenGL são enumeradas a seguir, com uma breve descrição dos objetos criados. Alguns exemplos das opções de desenho são ilustradas na figura 5.3, onde os números indicam a ordem de especificação dos vértices.

- `GL_POINTS`: pontos individuais
- `GL_LINES`: pares de vértices delimitadores de sucessivos segmentos de reta
- `GL_LINE_STRIP`: vértices consecutivos de uma linha poligonal
- `GL_LINE_LOOP`: vértices consecutivos de uma linha poligonal fechada
- `GL_POLYGON`: vértices consecutivos de um polígono simples e convexo
- `GL_QUADS`: sucessão de vértices que especificam, de quatro em quatro, vários quadriláteros
- `GL_QUAD_STRIP`: vértices que especificam sucessivos quadriláteros com lados em comum (dois a dois)

```
glBegin(GL_POLYGON);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(0.0, 3.0, 0.0);
    glVertex3f(4.0, 3.0, 1.0);
    glVertex3f(6.0, 1.5, 1.0);
    glVertex3f(4.0, 0.0, 2.0);
glEnd();
```

Figura 5.4: Exemplo de código para o desenho de um polígono

- `GL_TRIANGLES`: sucessão de vértices que especificam, de três em três, vários triângulos
- `GL_TRIANGLE_STRIP`: vértices que especificam sucessivos triângulos unidos por lados em comum
- `GL_TRIANGLE_FAN`: vértices que especificam sucessivos triângulos formando uma estrutura circular

A figura 5.4 exemplifica um trecho de código para desenho de um pentágono, cujos vértices são especificados dentro da seção: `glBegin(GL_POLYGON)` ... `glEnd()`.

5.2.4 Transformações

As funções da OpenGL que implementam transformações de translação, escala e rotações de objetos:

`glTranslatef(x, y, z)`, `glScalef(x, y, z)` e `glRotatef(angle, x, y, z)`.

Na função `glTranslatef()`, os parâmetros recebem os valores do deslocamento em cada um dos três eixos, em `glScalef()` os mesmos parâmetros indicam o fator de escala para cada eixo específico. Já em `glRotatef()`, o parâmetro `angle` indica o ângulo, em

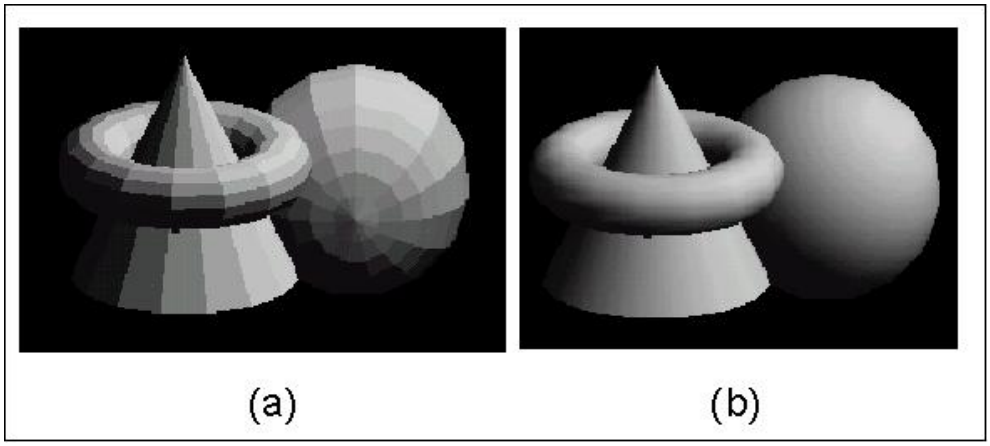

Figura 5.5: Diferentes modelos de tonalização em OpenGL: (a) *flat* e (b) *Gouraud*.

graus, de rotação no sentido anti-horário em relação ao vetor `(x, y, z)`. Como já dito, ambas as transformações são efetuadas sobre a matriz ativa.

5.2.5 Iluminação

A iluminação de cenas em OpenGL é definida em três passos: escolha do modelo de tonalização, criação de uma ou mais fontes luminosas, e definição do modelo de reflexão para cada material componente dos objetos criados.

A função `glShadeModel(mode)` especifica o modelo de tonalização em OpenGL, onde o parâmetro `mode` pode receber os valores `GL_FLAT` para iluminação constante, ou `GL_SMOOTH` para selecionar o modelo de tonalização de *Gouraud*[1]. A figura 5.5 ilustra a diferença entre os dois modelos. Este trabalho utiliza o modelo de *Gouraud*, que apresenta uma melhor definição da imagem gerada.

[1] A biblioteca OpenGL não implementa tonalização pelo método de *Phong*.

Tabela 5.1: Propriedades das fontes luminosas, configuradas por **glLightfv()**.

Constante	**Valor Default**	**Descrição**
`GL_AMBIENT`	(0.0, 0.0, 0.0, 1.0)	Valores RGBA da componente de luz ambiente.
`GL_DIFFUSE`	(1.0, 1.0, 1.0, 1.0)	Valores RGBA da componente de luz difusa.
`GL_SPECULAR`	(1.0, 1.0, 1.0, 1.0)	Valores RGBA da componente de luz especular.
`GL_POSITION`	(0.0, 0.0, 1.0, 0.0)	Posição da fonte. O 4^o parâmetro define se a fonte é direcional ou posicional.
`GL_SPOT_DIRECTION`	(0.0, 0.0, -1.0)	Direção da luz do tipo *spot*.
`GL_SPOT_EXPONENT`	0.0	Grau de atenuação angular da intensidade luminosa do *spot*.
`GL_SPOT_CUTOFF`	180.0	Ângulo de abertura da fonte de luz *spot*.
`GL_CONSTANT_ATTENUATION`	1.0	Termo constante da atenuação.
`GL_LINEAR_ATTENUATION`	0.0	Termo linear da atenuação.
`GL_QUADRIC_ATTENUATION`	0.0	Termo quadrático da atenuação.

A criação de fontes de luz é implementada pela função `glEnable(GL_LIGHTi)`, cujo parâmetro especifica qual a fonte luminosa está sendo habilitada. A OpenGL suporta no mínimo 8 fontes de luz[2], identificadas pelas constantes `GL_LIGHT0`, ..., `GL_LIGHT7`.

A configuração de cada fonte de luz criada é implementada pela chamada à função `glLightfv(light, pname, *param)`, onde o parâmetro `light` indica a fonte de luz a ser configurada. O parâmetro `pname` especifica qual propriedade será modificada, e `param` determina seu(s) novo(s) valor(es). A tabela 5.1 apresenta todas as propriedades configuráveis disponíveis na OpenGL.

Após a criação e ajuste das fontes de luz, é necessário definir como os objetos irão "reagir" aos raios de luz emitidos por essas fontes, ou seja, definir quais serão as características da reflexão do material de cada objeto. Para isso, a OpenGL possui a função `glMaterialfv(face, pname, *param)` que, em conjunto com as funções de iluminação

[2] A função **glGetIntegerv(GL_MAX_LIGHTS, *param)** fornece a quantidade de fontes luminosas disponíveis.

Tabela 5.2: Propriedades de reflexão dos materiais, configuradas por **glMaterialfv()**.

Constante	**Valor Default**	**Descrição**
`GL_AMBIENT`	(0.2, 0.2, 0.2, 1.0)	Valores RGBA da reflexão de luminosidade ambiente.
`GL_DIFFUSE`	(0.8, 0.8, 0.8, 1.0)	Valores RGBA da reflexão de luminosidade difusa.
`GL_SPECULAR`	(0.0, 0.0, 0.0, 1.0)	Valores RGBA da reflexão de luminosidade especular.
`GL_AMBIENT_AND_DIFFUSE`		Valores RGBA das reflexões de luminosidade ambiente e difusa.
`GL_SHININESS`	0.0	Expoente da reflexão especular (brilho). Varia entre [0.0,128.0]
`GL_EMISSION`	(0.0, 0.0, 0.0, 1.0)	Valores RGBA de objetos desenhados como fontes luminosas.

definem a cor e a característica da superfície do objeto. É possível definir materiais diferentes para o interior e exterior do objeto. O primeiro parâmetro, `face`, que pode receber as constantes `GL_FRONT`, `GL_BACK` ou `GL_FRONT_AND_BACK`, permite identificar quais faces do objeto possuem a propriedade do material que está sendo especificadas. O parâmetro `pname` indica a propriedade da reflexão do material. O último parâmetro `*parm` é um vetor que determina o valor para o qual `pname` é especificado. A tabela 5.2 apresenta alguns valores-padrão utilizados, bem como os tipos de características de reflexão do material que podem ser utilizados.

A biblioteca OpenGL implementa o modelo de iluminação proposto por *Phong*, conhecido como *Modelo Phong de Reflexão Especular*. A figura 5.6 ilustra diferentes composições de fontes luminosas e tipos de reflexão suportados pela OpenGL, utilizando-se o modelo de tonalização de *Gouraud*.

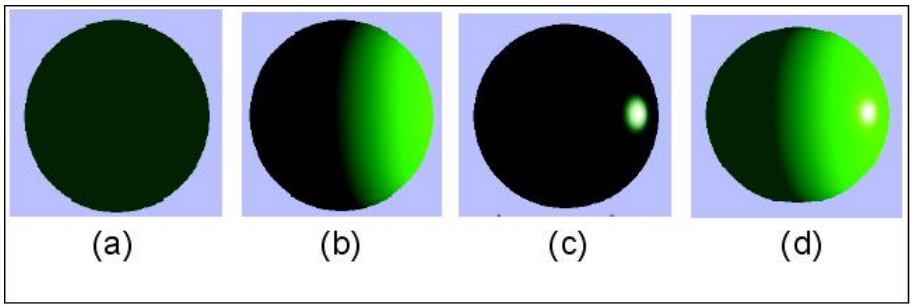

Figura 5.6: Tipos de fonte de luz e reflexão: (a)Ambiente, (b) Difusa, (c) Especular e (d) Todas as 3 componentes combinadas (*Modelo Phong de Reflexão Especular*).

5.2.6 GLUT

Conforme mencionado anteriormente, a OpenGL não provê funções que implementem facilidades específicas de cada sistema operacional, tais como: gerenciamento de janela ou manipulação de entrada e saída, como controle de teclado e mouse. Para desenvolver uma interface que permita a criação e manipulação de janelas ou entrada/saída de dados, foi criada a biblioteca *GLUT* (*OpenGL Utility Toolkit*), que também implementa funções para desenho de objetos 3D mais complexos, tais como superfícies, esferas, toróides e cubos (WRIGHT, 2000).

5.3 Estrutura do Simulador

Todo o sistema foi projetado de forma a permitir uma fácil criação, configuração e visualização de um ambiente tridimensional, atendendo também a requisitos básicos

como: implementação de uma interface amigável, e fácil compreensão e reutilização dos componentes criados.

O simulador é composto por vários módulos, que implementam inúmeras classes componentes do sistema. As principais classes modeladas podem ser classificadas em dois grandes grupos:

1. **Elementos de Ambiente**: São todos os componentes "físicos" da cena. São objetos que estão presentes em uma localização específica no mundo. Fazem parte deste grupo: o mundo criado, seus objetos componentes (curvas, superfícies, poliedros, etc.), as fontes de luz, a câmera de visualização, etc.

2. **Elementos de Configuração**: Referem-se a objetos que definem determinadas propriedades de visualização da cena, tais como: tipos de projeção, modelos de iluminação e tipos de reflexão luminosa dos materiais dos objetos visíveis.

O simulador implementa a criação e visualização de cada cena através da instanciação e configuração dos objetos que a compõem. Por exemplo, pode-se construir um determinado mundo, invocando métodos específicos para: criar e inserir objetos tridimensionais, posicionar e configurar fontes luminosas, definir seu modelo de iluminação, configurar o tipo de projeção desejada, e por fim visualizar toda a cena.

5.4 Descrição do Sistema

Esta seção apresenta a estrutura básica do simulador, descrevendo as principais classes implementadas. A definição das formas geométricas básicas manipuladas pelo simulador é estruturada pelo diagrama ilustrado na figura 5.7. Todas as formas geométricas são definidas por especializações da classe `Forma`, derivada da classe abstrata `Objeto`.

A classe `Forma` é que define o tipo de reflexão do material, herdado pelas suas classes filhas. A classe `Superficie` implementa a modelagem de superfícies parametrizadas utilizando as NURBS, discutidas no capítulo 3. O método `Desenho` utiliza as funções de desenho de NURBS disponibilizadas pela biblioteca GLUT (p.e., `gluNurbsSurface`). A classe `Superficie_Dinamica` especializa uma superfície, permitindo a simulação de seu movimento ou deformação, implementado pelo método `Executa`. Importante ressaltar que novas formas geométricas podem ser adicionadas pela especialização de `Forma`, implementando-se o método `Desenha` apropriadamente.

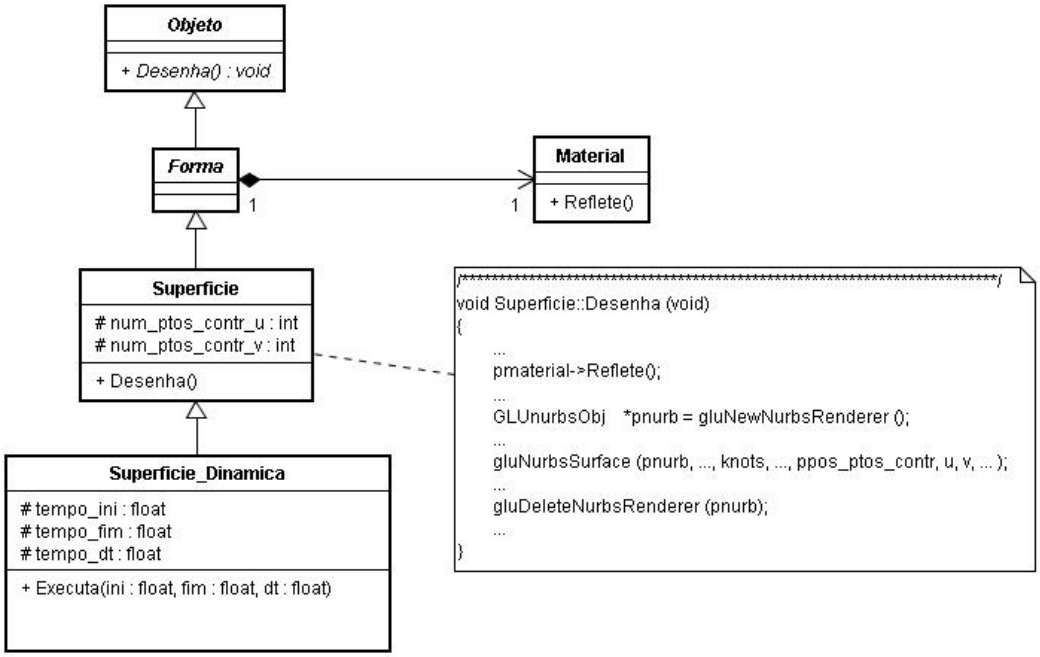

Figura 5.7: Estrutura de classes para a definição de uma superfície.

A figura 5.8 ilustra a estrutura do ambiente a ser construído. Toda a cena a ser visualizada e simulada é criada pela instanciação da classe `Mundo`, também herdada de `Objeto`. Métodos específicos permitem inserir formas geométricas ao mundo criado, bem como várias fontes de iluminação. A classe `Mundo` implementa o método `Desenha` invocando o desenho de suas formas componentes.

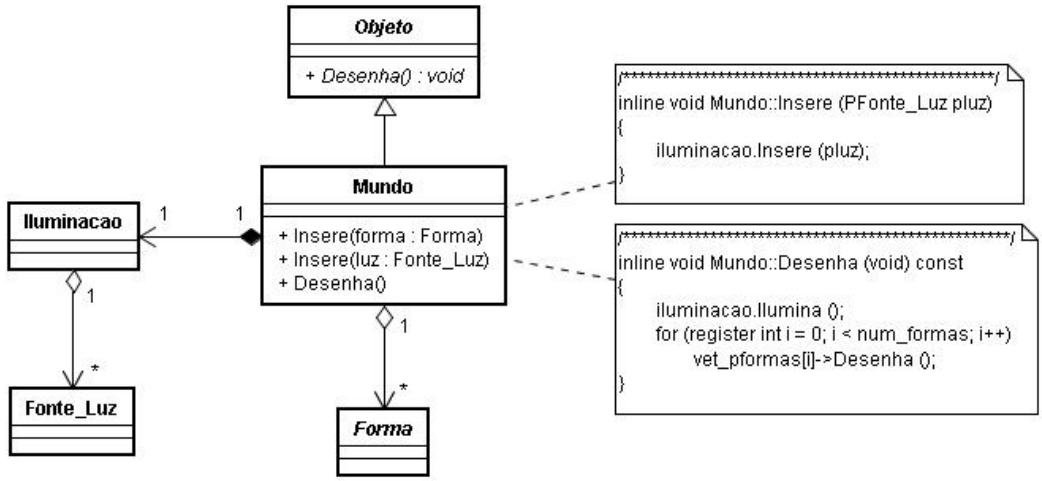

Figura 5.8: Estrutura de uma cena tridimensional.

A definição dos diferentes tipos de iluminação e reflexão de luminosidade é estruturada pelo diagrama ilustrado na figura 5.9. A classe básica `Luz3` define as componentes de luminosidade ambiente, difusa e especular, sendo herdadas pelas classes `Fonte_Luz`, que define fontes luminosas, e `Material`, que implementa a reflexão do material componente em objetos herdados da classe `Forma`. A classe `Iluminacao` implementa um conjunto de fontes criadas para iluminação de um mundo específico.

A figura 5.10 ilustra a estrutura geral do simulador. A classe `Tela_Principal` define a janela principal da aplicação. Vários objetos da classe `Janela` podem ser criados, cada um associado a um objeto `Mundo`. É possível definir exibições simultâneas de mundos distintos, ou mesmo diferentes visualizações do mesmo mundo. Tanto `Tela_Principal` como `Janela` implementam o método `Mostra`, herdado da superclasse abstrata `Janela_Exibicao`. Para a visualização da cena, a classe `Janela` também define associações para os objetos das classes `Camera` e `Projecao`, invocando-os antes do método de `Desenho` do mundo criado.

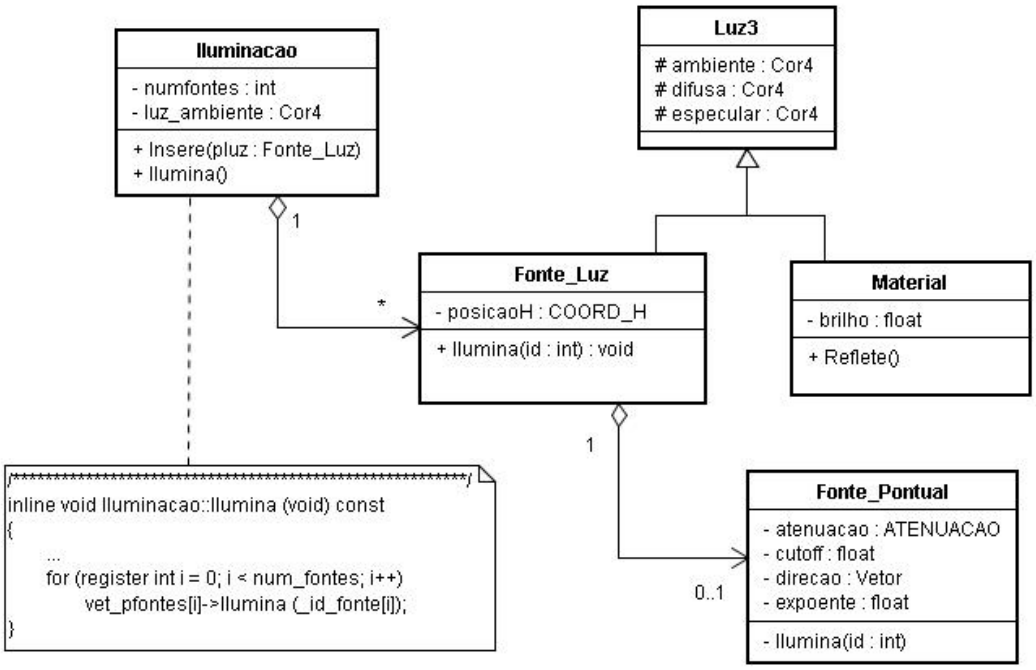

Figura 5.9: Estrutura de definição e utilização de modelos de iluminação.

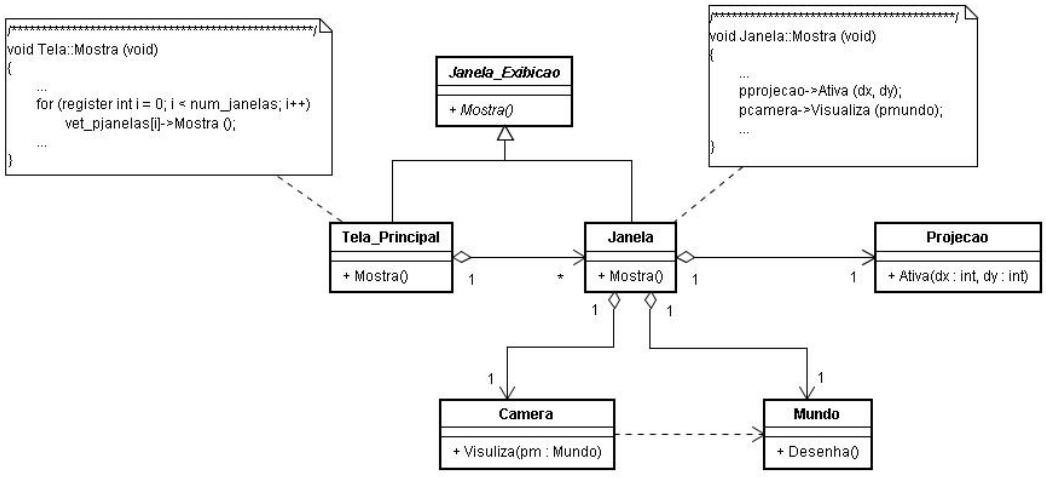

Figura 5.10: Estrutura básica do simulador.

5.5 Simulações e Resultados

Esta seção apresenta os principais resultados da utilização do sistema de simulação. São demonstradas suas funcionalidades mais relevantes e apresentados os principais re-

sultados de simulações envolvendo deformações de superfícies, objeto desse trabalho. As superfícies criadas podem ser manipuladas manualmente, através do reposicionamento dos seus pontos de controle, ou de forma automática, obedecendo a equações que regem a física do seu movimento. A principal aplicação visa simular as equações de movimento do sistema de partículas inspirado em Wu (WU, 1995).

A figura 5.11 ilustra a tela principal exibida pelo simulador em um dos primeiros testes de execução. Foi criada uma configuração simples, contendo uma única janela de visualização de um mundo onde são exibidos os eixos cartesianos (visando uma melhor compreensão do posicionamento dos objetos). Foi criada uma cena contendo um arranjo tridimensional de esferas de diferentes cores, distribuídas uniformemente no espaço, através da função `glutSolidSphere()`. O principal objetivo foi verificar a facilidade na criação de objetos e configuração de parâmetros de visualização: tipos de projeção, modelos de iluminação e reflexão, e movimentação da câmera. A maior dificuldade foi estabelecer os valores dos parâmetros de luminosidade e reflexão dos materiais, obtidos de forma empírica e utilizados nas simulações.

Foram realizados testes para definir qual a superfície a ser simulada que melhor apresentasse uma suavidade e que permitisse um melhor controle de sua deformação. Considerando que as superfícies podem ser vistas como uma coleção de curvas, simulamos algumas curvas para determinar que tipo de superfície melhor se aplicaria em nosso trabalho. A figura 5.12 ilustra dois tipos de curvas simuladas: uma curva interpolada, e uma spline. A curva Interpolada apresentou falhas na sua continuidade quando o deslocamento dos pontos de controle, já na curva spline, a suavidade manteve-se mais regular em suas conectividades.

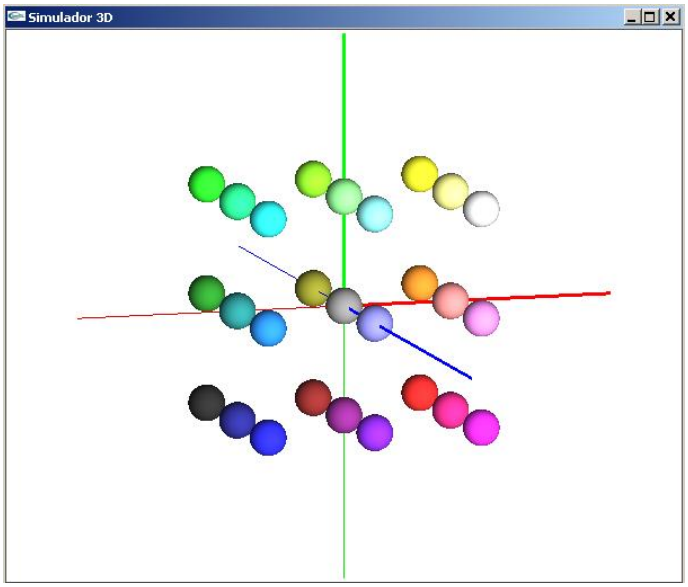

Figura 5.11: Teste inicial: arranjo de esferas coloridas.

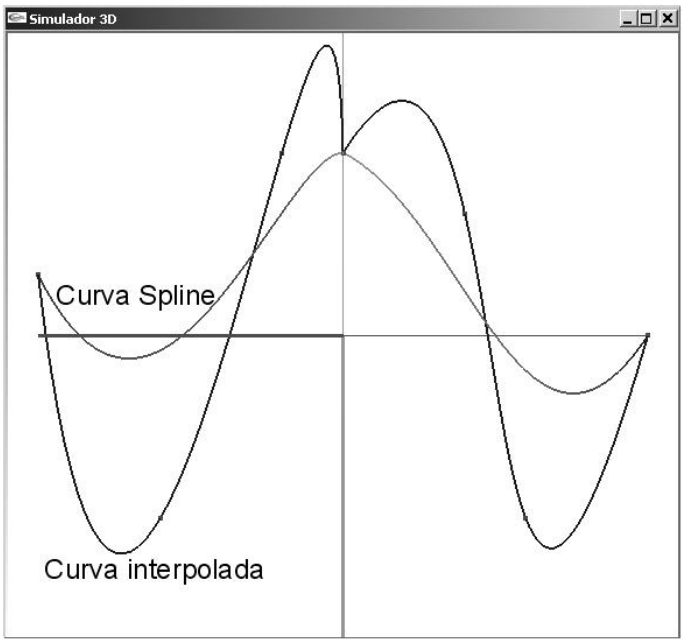

Figura 5.12: Teste inicial: Diferença entre curva interpolada e curva spline.

Sabendo-se que o cálculo das superfícies são extensões dos cálculos das curvas, escolhemos as superfícies NURBS, que são superfícies Splines que permitem um maior controle e suavidade na continuidade em seus pontos de conectividades.

A modelagem da NURBS é ilustrada na figura 5.13, onde uma mesma superfície é exibida em duas janelas distintas, ambas associadas ao mesmo mundo. Como pode ser observado, o simulador também permite a visualização dos pontos de controle da superfície, que foi especificada por uma malha de 16 pontos de controle e um vetor de nós, definidos como ilustra a figura 5.14. Como cada janela tem a sua própria câmera, pode-se "posicionar os observadores" em pontos distintos, como mostra a tela do programa ilustrada na figura 5.15.

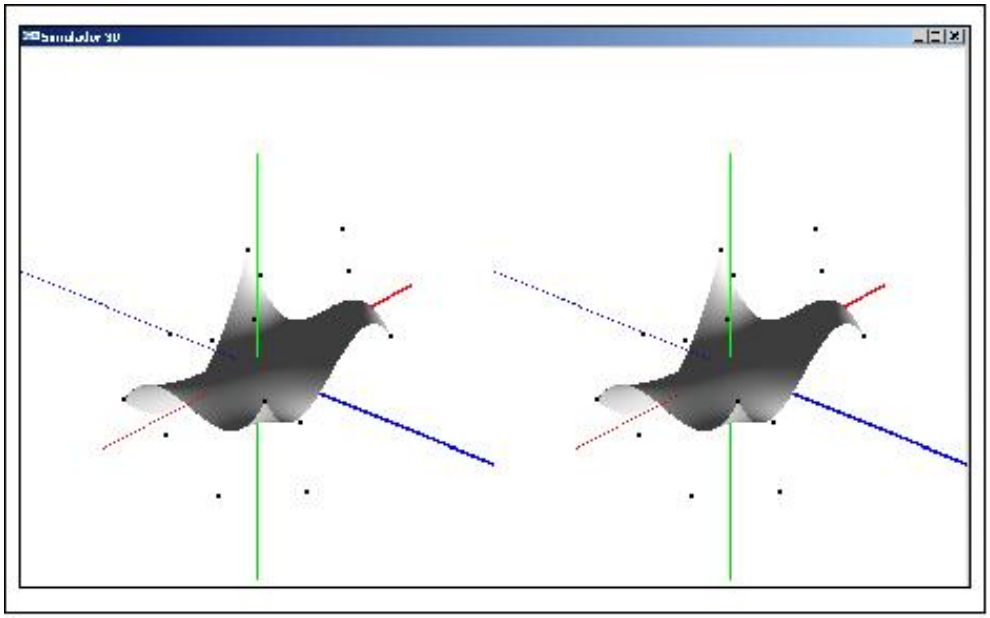

Figura 5.13: Visualizações simultâneas da mesma cena.

O simulador permite o reposicionamento manual de cada ponto de controle de uma superfície, possibilitando uma deformação controlada da mesma. A figura 5.16 ilustra tal situação, em que são exibidas duas superfícies distintas (ambas geradas pelos mesmos pontos de controle especificados na figura 5.14). Cada uma delas foi deformada por diferentes reposicionamentos de pontos de controle específicos.

```
#define   U    4
#define   V    4
static COORD_3D   pontos_superf[U][V] =
{
   { { -6.0, -2.0, -6.0 }, { -6.0,  4.0, -2.0 }, { -6.0, -6.0,  2.0 }, { -6.0,  2.0,  6.0 } },
   { { -2.0, -6.0, -6.0 }, { -2.0,  2.0, -2.0 }, { -2.0,  8.0,  2.0 }, { -2.0, -6.0,  6.0 } },
   { {  2.0, -6.0, -6.0 }, {  2.0,  2.0, -2.0 }, {  2.0, -4.0,  2.0 }, {  2.0,  8.0,  6.0 } },
   { {  6.0,  4.0, -6.0 }, {  6.0, -6.0, -2.0 }, {  6.0,  8.0,  2.0 }, {  6.0,  2.0,  6.0 } }
};

FLOAT     knots[8] = { 0, 0, 0, 0, 1, 1, 1, 1 };
```

Figura 5.14: Definição de pontos de controle para uma superfície específica.

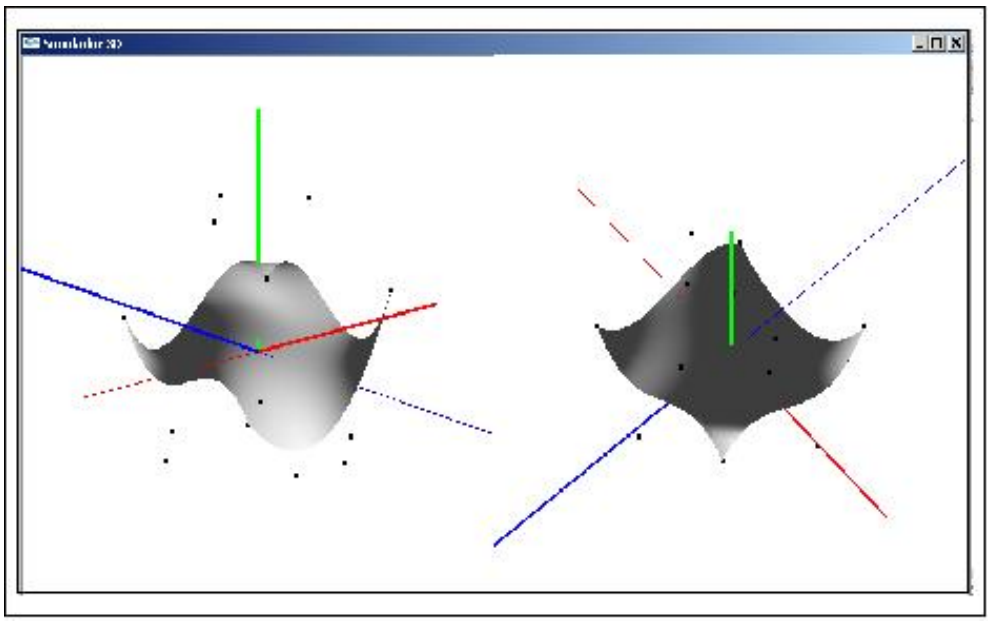

Figura 5.15: Visualizações distintas da mesma cena.

Para a animação da deformação da superfície, o deslocamento dos pontos de controle é realizado dinamicamente segundo a equação do movimento visto no capítulo 4. Foram realizadas duas simulações com intuito de analisar a deformação de uma superfície elástica. Com isso, atribuiu-se propriedades de massa aos pontos de controle que são interligados por molas. As partículas sofrem ação de forças externas que influenciam no seu movimento.

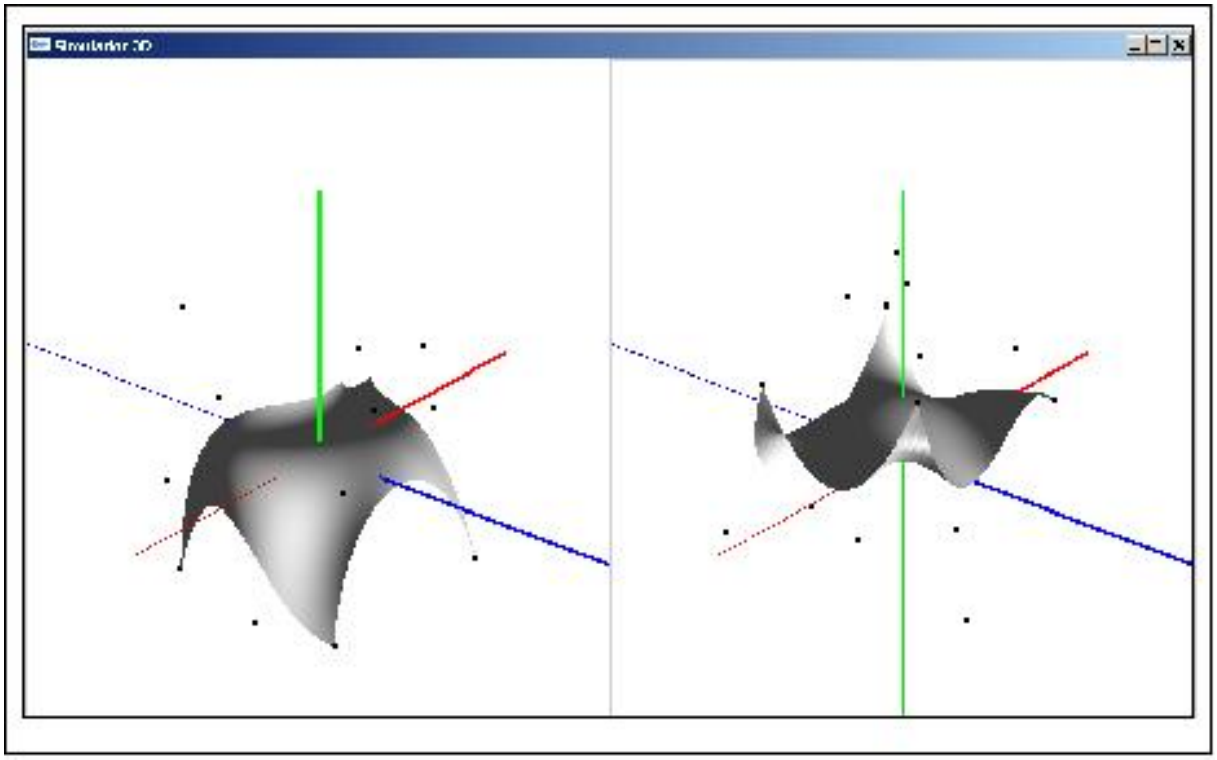

Figura 5.16: Deformação manual de superfícies.

Nas simulações realizadas com animação, foram atribuídos valores constantes para as forças externas, forças internas, velocidade e massa para ambas as superfícies simuladas. Tais valores foram definidos empiricamente e obedecem uma configuração que representa a sua direção em relação ao eixo da origem e a sua magnitude. A posição inicial dos pontos de controle $q_i(0)$ para ambas as superfícies, é ilustrado na figura 5.17.

As velocidades iniciais $\dot{q}_i(0)$ aplicadas aos pontos extremos da superfície, (i.e. $U \neq 0$ ou $V \neq 0$ para o vetor de pontos de controle), é nula: $\dot{q}_i(0) = (0,0,0)$. Para os pontos internos da superfície (i.e. $U \neq 0$ e $V \neq 0$ para o vetor de pontos de controle), foi atribuído: $\dot{q}_i(0) = (0,20,0)$(deslocamento de magnitude vinte no eixo y em relação à origem).

As Forças externas exercidas sobre as partículas, foram definidas da seguinte forma: $F_i = (0,0,0)$ para pontos extremos, e $F_i = (0,0,40)$(deslocamento de magnitude quarenta

```
#define   U    4
#define   V    4
static COORD_3D   pontos_superf[U][V] =
{
   { { -6.0, -6.0,  0.0 }, { -2.0, -6.0,  0.0 }, {  2.0, -6.0,  0.0 }, {  6.0, -6.0,  0.0 } },
   { { -2.0, -2.0,  2.0 }, { -2.0, -2.0,  8.0 }, {  2.0, -2.0,  8.0 }, {  6.0, -2.0,  0.0 } },
   { { -6.0,  2.0,  0.0 }, { -2.0,  2.0,  8.0 }, {  2.0,  2.0,  8.0 }, {  6.0,  2.0,  0.0 } },
   { { -6.0,  6.0,  0.0 }, { -2.0,  6.0,  0.0 }, {  2.0,  6.0,  0.0 }, {  6.0,  6.0,  0.0 } }
};

FLOAT     knots[8] = { 0, 0, 0, 0, 1, 1, 1, 1 };
```

Figura 5.17: Definição de pontos de controle para uma superfície específica.

no eixo z em relação à origem) para pontos internos. As Forças internas entre as partículas foram consideradas nulas $P_i = 0$.

Na primeira simulação de animação, atribuiu-se valores para a constante de Hook $K = 5.0$ e sem amortecimento aplicado às partículas $c = 0$. Nesse caso, a deformação da superfície seguirá um movimento harmônico simples nos eixos y e z, proporcionando um aspecto de movimento circulatório e infinito (pois não há amortecimento). A figura 5.18 ilustra uma visualização da superfície em sua posição de repouso, sobre dois ângulos de visão disitintos, antes de ser aplicada a animação.

Com o início da simulação, as partículas internas sofrem influência da força e da velocidade passando a se deslocar harmonicamente nos eixos y e z. A figura 5.19 ilustra o início do movimento e conseqüentemente a deformação da superfície.

O movimento das partículas alcança seu deslocamento máximo por conseqüencia do coeficiente de Hook, como ilustrado na figura 5.20, proporcionando a elasticidade máxima da superfície.

A figura 5.21 ilustra a continuação do deslocamento dos pontos de controle que se deslocam devido à força restauradora da mola. Por não possuir um amortecimento, a deformação da superfície permanece em um movimento harmônico intermitente.

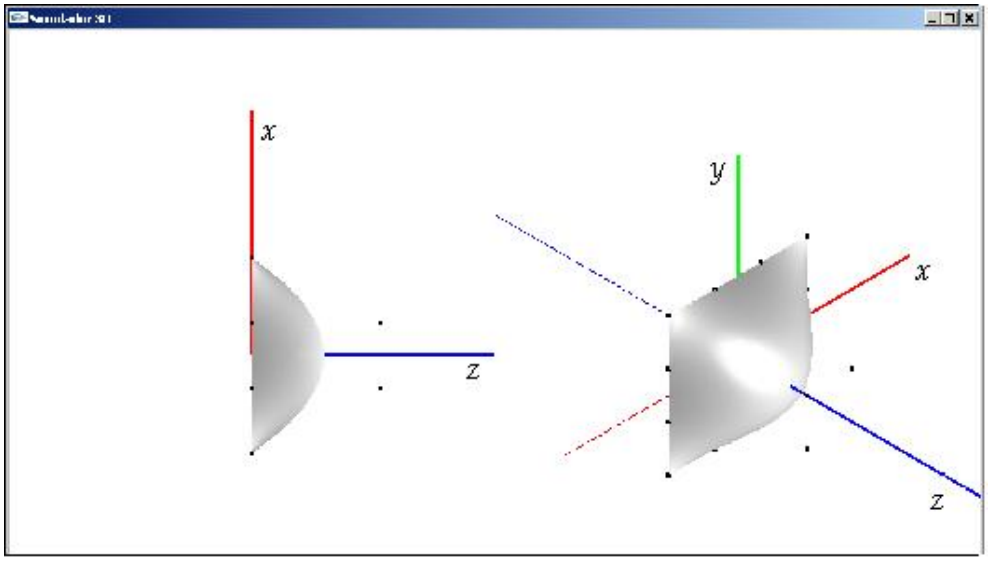

Figura 5.18: Superfície não amortecida em repouso.

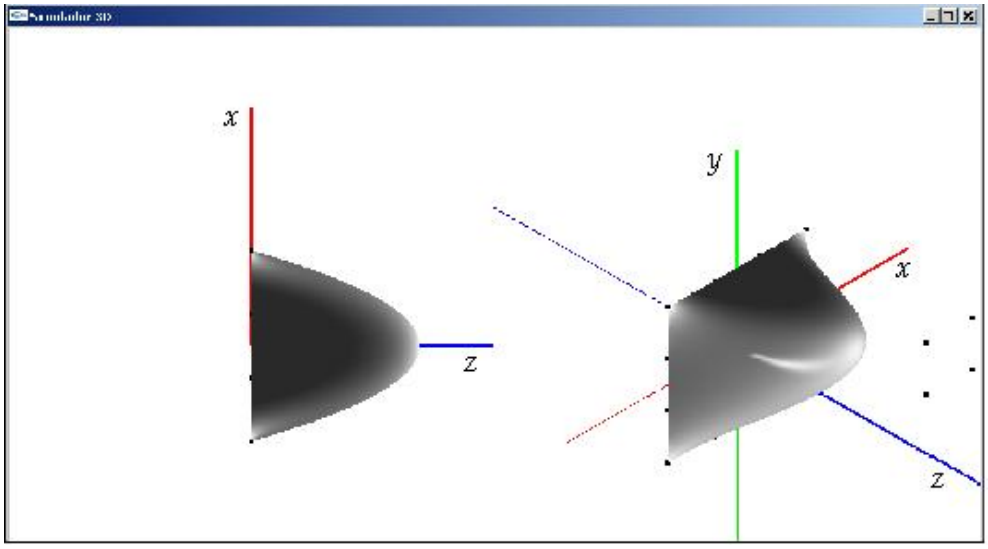

Figura 5.19: Inicio da deformação da superfície não amortecida.

Na segunda simulação da animação, adicionou-se um valor à constante de amortecimento $c = 10$ mantendo os mesmos valores para: a constante de Hook , as forças atuantes, e massa. A figura 5.22 ilustra uma visualização da superfície animada. Devido a constante de amortecimento ser maior que a constante elástica, a força amortecedora domina a força

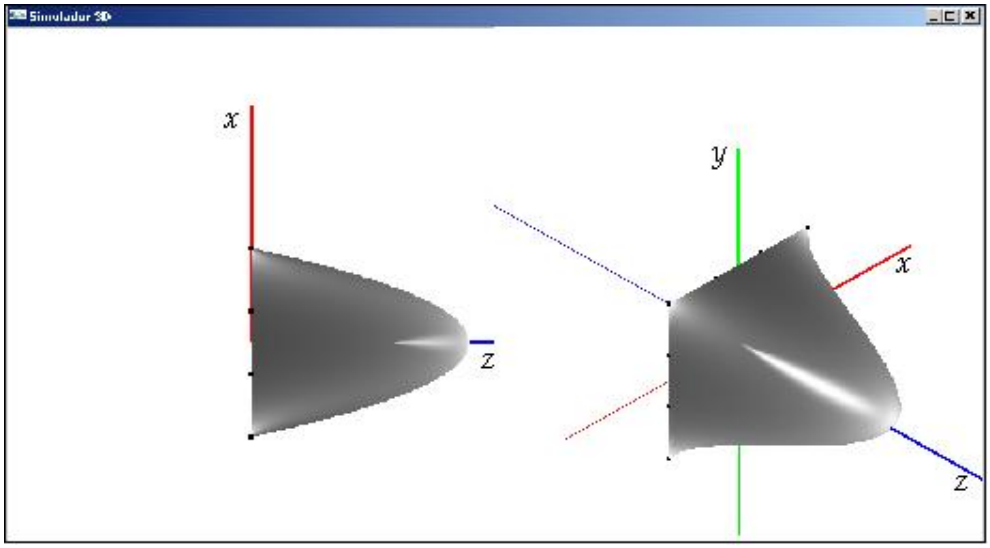

Figura 5.20: Superfície não amortecida em seu deslocament máximo.

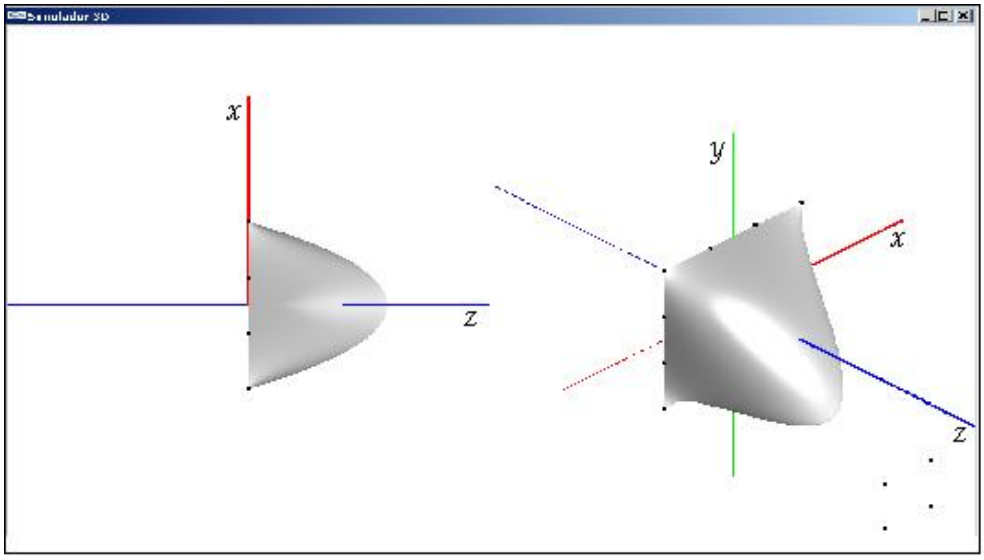

Figura 5.21: Superfície não amortecida em retornando seu deslocament máximo.

restauradora da mola, proporcionando uma estabilidade na deformação em um tempo relativamente curto, como ilustrado na figura 5.23. Para uma constante de amortecimento inferior à constante de Hook, o movimento harmônico é mais lento e a superficie oscila até a posição de equilíbrio em um tempo mais prolongado.

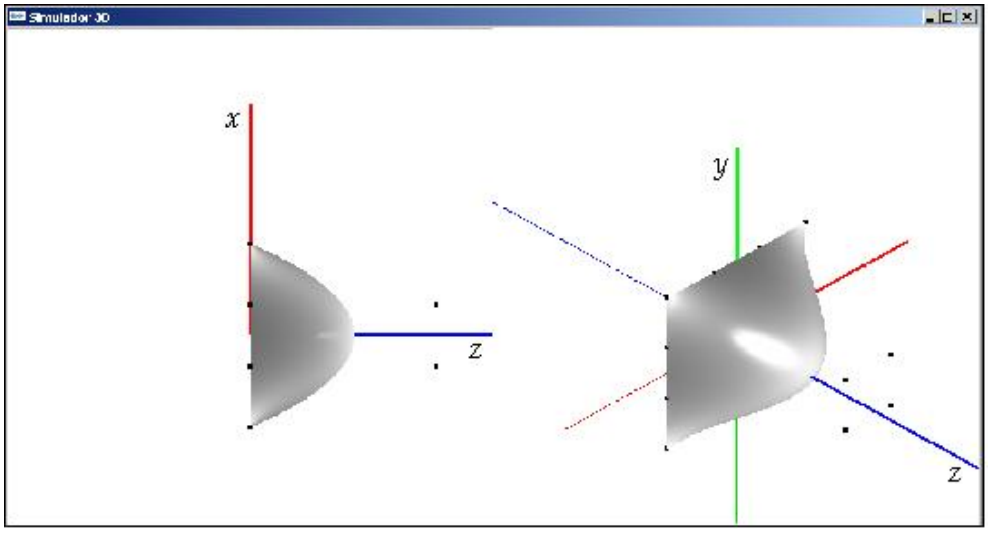

Figura 5.22: Deformação da superfície amortecida em seu deslocamento inicial.

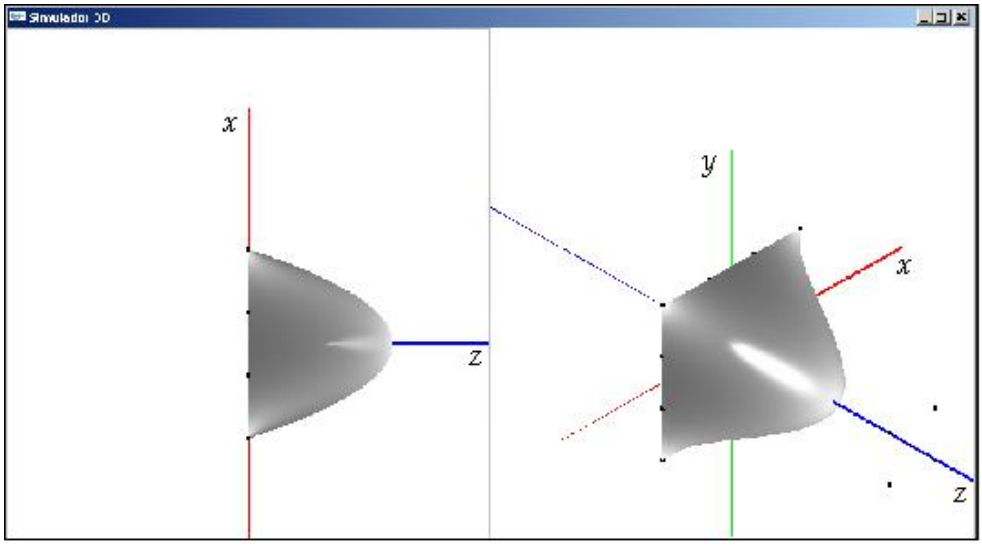

Figura 5.23: Deformação da superfície amortecida já estabilizada.

6 Conclusão

Este trabalho teve como objetivo o estudo e implementação de um ambiente gráfico capaz de simular superfícies deformáveis. Inicialmente foram discutidos os fundamentos básicos e técnicas para a representação, processamento e visualização de objetos gráficos. Um estudo sobre os principais modelos de curvas e superfície foi realizado. Este estudo contribuiu para um melhor entendimento dos mecanismos da visualização das deformações das superfícies. Um modelo físico baseado em sistema de partículas, inspirado em Wu (WU, 1995), foi desenvolvido e um sistema computacional para a simulação, visualização e animação das deformações implementado.

O programa permite a visualização tridimensional da deformação de uma superfície em tempo real, permitindo um controle manual ou automatizado da animação, através da alteração dos pontos de controle da superfície. O projeto do sistema levou em consideração alguns requisitos básicos: interface de fácil comunicação, recursos para configuração do programa, e também uma estruturação que permitisse sua fácil compreensão e reutilização. Para atender a esses requisitos, foi utilizada a metodologia orientada a objetos, implementada na linguagem C++, em conjunto com a biblioteca gráfica OpenGL.

Nos Experimentos realizados, buscou-se inicialmente testar as características de projeção, rotação, translação e iluminação de objetos em uma determinada cena. Notamos nesse caso a facilidade de criar objetos e configurar parâmetros de visualização, entretanto, houve uma certa dificuldade para estabelecer as configurações de luminosidade e

reflexão dos materiais de forma a proporcionar uma visão mais realística da cena. De forma empírica, diversos testes foram realizados até a obtenção da configuração de tais parâmetros ideal para o simulador. Para a definição das superfícies a serem simuladas, foram realizados testes com curvas interpoladas e curvas splines afim de analisar sua continuidade e suavidade quando alterado as posições dos pontos de controle. Através dessa análise, decidimos optar por simular superfícies NURBS que são uma generalização das superfícies splines e que possuem uma maior controle em sua suavidade. Encontramos dificuldades na geração de superfícies NURBS com malhas de pontos de controle muito grande, isso devido a distribuição dos pontos no espaço da cena, com isso, limitamos os testes a superfícies com 16 pontos de controle.

Obtivemos bons resultados em relação à animação da superfície elástica segundo a equação do movimento estudada. Os testes com amortecimento e sem amortecimento foram satisfatórios, proporcionando uma deformação animada e bem realística. O simulador desenvolvido está apto a simular não só superfícies deformáveis, como também proporcionar uma visualização e animação de objetos simples, isso devido a sua estrutura de fácil reutilização de código, proporcionando uma programação simples e intuitiva. Não foram realizados testes de desempenho comparativo com outros simuladores, haja visto não ter sido esse o objetivo do nosso trabalho. Entretanto, notamos que em nossas simulações, o tempo de resposta do simulador superaram nossas espectativas, isso devido a biblioteca OpenGL e a estruturação do programa.

6.1 Trabalhos Futuros

Pela avaliação das imagens geradas, nota-se que os parâmetros e técnicas usadas não são suficientes para dar todo o realismo necessário para imitar a deformação dos vários tipos de superfícies deformáveis. Por isso, algumas sugestões são dadas como trabalhos

futuros, que tem a função de adicionar ao simulador a capacidade de interagir com mais elementos para proporcionar uma melhor simulação de cenários reais.

- Melhoria na interface do simulador com o usuário, facilitando a entrada dos dados a serem simulados e permitindo um controle mais intuitivo da animação da superfície;
- Adicionar propriedades de textura, espelhamento e rugosidade, adcionando caracterísiticas mais reais aos objetos simulados;
- Técnicas de renderização como Radiosiade e Ray Tracing podem ser implementadas juntamente com outras técnicas de projeção e visualização, capaz de proporcionar uma simulação mais nítidas e real de uma cena;
- Simular não só superfícies como outros polígonos em movimento bem como iteração entre diversos objetos em uma mesma cena;
- Utilizar outros modelos físicos e até hibridos para simular o movimento e a deformoação das superfícies, afim de se obter melhores resultados nas animações.

Finalmente, o sistema de simulação de superfícies deformáveis desenvolvido através desse trabalho é um primeiro passo no desenvolvimento de um frame-work a ser implementado e utilizado em futuras simulações de pesquisas a serem realizadas pelo laboratório CONTROLAB.

Referências Bibliográficas

[ANGEL 2004] Angel, Edward. **Interactive Computer Graphics**. 3rd ed. Boston Massachusetts: Addison Wesley, 2004. 784 p.

[AU 2000] Au, C. K.; Wu, Z.; Yuen, M. M. F. **Effect of Fabric Properties on Cloth Draping Modeling**. In Proceedings IEE Conf. on Geometric Modeling and Processing 2000. pp. 69-79, Hong Kong, 2000.

[BOUGKNIGHT 1970] Bougknight, W. J. **A Procedure for Generation of Three-dimensional Half-tone Computer Graphics Presentations**. Comunications of the ACM, 1970.

[BERG 1998] Berg, M. T.; Kreveld, M. J. V.; Overmars, M.; Cheong, O. S. **A Procedure for Generation of Three-dimensional Half-tone Computer Graphics Presentations**. Comunications of the ACM, 1970.

[BREEN 1991] Breen, D. E.; House, D. H.; Getto, P. H. **A Particle-Based Computational Model of Cloth Draping Behavior**. In: Patrikalakis, N. M. **Scientific Visualization of Physical Phenomena** Springer-Velarg, Inc., Tokyo, 1991, pp. 113-134.

[CARIGNAN 1992] Carignan, M.; Yang, Y.; Magnenat-Thalmann N.; Thalmann, D. **Dressing Animated Synthetic Actors With Complex De-**

formable Clothes. Proc. SIGGRAPH'92 Computer Graphics, Vol.26, No.2, pp. 99-104, 1992.

[CAVALCANTI 2000] Cavalcanti, Paulo R. **Introdução à Computação Gráfica**. Notas do Curso de Computação Gráfica I, Universidade Federal do Rio de Janeiro, 2000.

[CHEN 1995] Chen, B.; Govindaraj, M. **A Physically Based Model of Fabric Drape Using Flexible Shell Theory**. Textile Journal, Vol. 65, No. 6, pp. 324-330, 1995.

[COHEN 2006] Cohen, Marcelo; Manssour, Isabel H. **OpenGL uma Abordagem Prática e objetiva**. São Paulo, SP, Novatec 2006, 478p.

[EBERT 1998] Ebert, D. et al. **Texturing & modeling**. 2nd ed. San Diego: AP Professional, 1998, 415p.

[EISCHEN 2000] Eischen, J. w.; Bigliani, R. **Continuum Versus Particle Representations**. In **Cloth Modeling and Animation** D. H. House and D. E. Breen, Eds. A. K. Peters Ltd. Natick, MA. pp. 79-122, 2000.

[ETZMUF 1998] Etzmuf, Olaf et al. **A Cloth Modelling System for Animated Characters**. In Proceedings of Graphiktag 2001.

[FOLEY 1990] Foley, J. D.; Dam, M. V. **Fundamentals of Interactive Computer Graphics**. 1nd ed. Boston Massachusetts: Addison Wesley, 1984. 348 p.

[GLASSNER, 1995] Glassner, A. S. **Principles of digital image synthesis**. San Francisco: Morgan Kaufmann, USA, 1995.

[GORAL 1984] Goral, C. M., Torrance, K. E., Greenberg, D. P. **Modeling the interaction of light between diffuse surfaces**. Computer Graphics, v.18, n.3, Jul. 1984, p.213-222.

[HILL 2000] Hill, Francis S. **Computer Graphics Using OpenGL**. 2nd ed. Upper Saddle River, New Jersey: Prentice-Hall, Inc., 2000. 922 p.

[HILTON 1994] Hilton, T. L.; Egbert, P. K. **Vector Field: an Interactive Tool for Animation Modeling and Simulation with Physically Based 3D Particle Systems and Soft Objects**. EUROGRAPHICS'94. Vol. 13, No. 3, pp. 399-412, 1994.

[LOPES 2006] Lopes, Ernesto P; Carvalho, Rafael V.; Silva, Marcela S. M.; Aude, Eliana P. L. **Notas Sobre Visualização**. Techinacal Report, Universidade Federal do Rio de Janeiro, Dezembro 2006.

[MELO 2004] Melo, Vanio F. **Modelagem e Controle de Caimento e Dobras em Superfícies Deformáveis**. Tese de Doutorado. Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação. Campinas, SP, 2004.

[MONTAGNAT 2001] Montagnat, J., Delingette, H, Ayache N. **A Review of Deformable Surfaces: Topology, Geometry and Deformation**. Image and Vision Computing, vol. 19, No. 14, pp. 1023–1040, 2001.

[MCLNERNY 1996] Mclnerny, T. and Terzopoulos, D. **Deformable models in medical image analysis: a survey**. Medical Image Analysis, 1(2):91.108, 1996.

[NG 1996] Ng, H. N. and Grimsdale, R. L. **Computer Graphics Techniques for Modeling Cloth**. IEEE Comput. Graph. Appl. 16, 5. Sep. 1996.

[PERSIANO 1996] Persiano, Ronaldo M. **Bases da Modelagem Geométrica**. Trabalho Apresentado na 10^a Escola de Computação, Instituto de Computação da UNICAMP, 141 p., 1996.

[REEVES 1983] Reeves, W. T. **Particle Systems: A Technique for Modeling a Class of Fuzzy Objects**. Proc. SIGGRAPH'83, Computer Graphics, Vol. 17, No. 3, pp. 359-376, 1983.

[SILVA 2006] DA Silva, Marcela S. M. **Simulação Gráfica de Modelos Estocásticos com Suporte Volumétrico**. Master's Thesis, Universidade Federal do Rio de Janeiro, 2006.

[SZELISKI 1992] Szeliski, R.; Tonnesen, D. **Surface Modeling with Oriented Particle Systems**. Proc. SIGGRAPH'92 Computer Graphics. Vol. 26, pp. 185-194.

[TERZOPOULOS 1987] Terzopoulos, D.; Platt, J.; Barr, A.; Fleischer, K. **Elastically Deformable Models**. In Proceedings Computer Graphics and Interactive Techniques, SIGGRAPH '87. ACM, New York, NY, 205-214, 1987.

[TERZOPOULOS 1988] Terzopoulos, D. and Fleischer, K. **Deformable Models**. Visual Computer, 4:306.331, 1988.

[TURNER 1993] Turner, R.; Thalmann, D. **The Elastic Surface Layer Model for Animated Character Construction** In Proceedings Computer Graphics International, Lausanne, Switzerland, Springer-Velarg, Tokyo, pp. 399-412, 1993.

[VOLINO 2000] Volino and Magnenat-Thalmann N. **Virtual Clothing, Theory and Practice**. Berlin:Springer, 2000.

[WATT 1993] Watt, Allan **3D Computer Graphics**. 2nd ed. Reading, Massachusetts: Addison Wesley, 1993. 500 p.

[WEISSTEIN 2000] Weisstein, Eric W. **NURBS Surface.** From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/NURBSSurface.html. Acessado em outubro 2007.

[WOO 1999] Woo, M., Neider, J., Davis, Tom; Shreiner, D. **OpenGL Programming Guide: the official guide to learning OpenGL version 1.2**. 3rd ed. Reading, Massachusetts: Addison Wesley, 1999. 730 p.

[WRIGHT 2000] Wright, R. S. Jr., Sweet, M. **OpenGL SuperBible**. 2nd ed. Indianapolis, Indiana: Waite Group Press, 2000. 696 p.

[WU 1995] Wu, Yi; Thalmann, D.; Magnenat-Thalmann N. **Deformable Surface Using Physically Based Particle Systems**. In Proceedings Computer Graphics International, pages 205-216. Academic Press., 1995.