**UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC**

**CURSO DE CIÊNCIA DA COMPUTAÇÃO**

**BRUNO ABEL MILANEZ**

# INTERPOLAÇÃO APLICADA À ANIMAÇÃO GERADA POR COMPUTADOR

**CRICIÚMA, DEZEMBRO DE 2009**

**BRUNO ABEL MILANEZ**

# INTERPOLAÇÃO APLICADA À ANIMAÇÃO GERADA POR COMPUTADOR

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: Prof. MEng. Evânio Ramos Nicoleit

**CRICIÚMA, DEZEMBRO DE 2009**

**BRUNO ABEL MILANEZ**

# Interpolação Aplicada à Animação Gerada por Computador

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

**Profa. MSc. Ana Claudia Garcia Barbosa**
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

**Prof. MEng. Evânio Ramos Nicoleit (UNESC)**
Orientador

**Profa. MSc. Leila Laís Gonçalves (UNESC)**

**Prof. Esp. Luciano Antunes (UNESC)**

Aos meus pais e minha irmã, por serem compreensivos, pacientes e me incentivarem nos momentos difíceis.

## AGRADECIMENTOS

Agradeço sobretudo a Deus por momentos que só Ele esteve presente.

À minha mãe Terezinha, ao meu pai Clédio e à minha irmã Laila, pois estavam sempre dispostos a ajudar.

Ao meu orientador Evânio por participar ativamente e acreditar que nós conseguiríamos. Agradeço também a ele pelas enriquecedoras reuniões nas quais construímos conhecimento.

À coordenação do curso e ao corpo docente, em especial a professora Merisandra e ao professor Fábio que foram importantes e atenciosos coadjuvantes.

Aos meus familiares e amigos que sempre demonstraram confiança em mim e com gestos simples me motivaram a continuar.

“Os sentidos se deleitam com coisas devidamente proporcionadas.”

São Tomás de Aquino

## RESUMO

Grande parte das animações produzidas é auxiliada de alguma forma pelo computador, por motivos de qualidade e rapidez no processo de produção. Uma das formas de uso mais importantes que lida com esses dois motivos é na interpolação. A possibilidade de diminuir o tempo de produção das seqüências animadas é tão relevante que já se tornou comum animadores utilizarem este recurso. No entanto a interpolação não é uma tarefa trivial. A escolha de determinada técnica tem impacto direto sobre a qualidade da animação. O animador precisa escolher a técnica que melhor se aplica em determinada ocasião. A pesquisa realizada neste trabalho discute três técnicas de interpolação: Linear; com base em *Splines* Cúbicas, representada pela Catmull-Rom *Spline* e; baseada em B-*splines*, na forma *B-spline* Cúbica Uniforme. O trabalho apresenta conceitos de animação e interpolação e suas relações. Também demonstra comparações entre os resultados das três técnicas de interpolação por meio da Razão Sinal Ruído de Pico e do Erro Médio Quadrático. Exemplos práticos de animações utilizando essas técnicas são apresentados. Finalmente são analisados os métodos de interpolação e os resultados, revelando características associadas às técnicas estudadas nas animações.



Palavras-chave: Animação, Interpolação Linear, Interpolação Polinomial, *Splines* Cúbicas, *B-Splines*

**ABSTRACT**

Much of the animations produced is aided somewhat by the computer, for reasons of quality and speed in the production process. One of the most important use that deals with these two reasons is the interpolation. The possibility of reducing the production time of animated sequences is important and has become common animators to use this resource. However, the interpolation is not a trivial task. The choice of a particular technique has a direct impact on the quality of the animation. The animator needs to choose the technique that best fits a particular occasion. The research described here discusses three techniques of interpolation: Linear; based on Cubic Splines, represented by Catmull-Rom Spline and; based on B-Splines, as B-spline Cubic Uniform. The work presents concepts of animation and interpolation and their relationship. It also shows comparisons between the results of three techniques of interpolation by means of Peak Signal to Noise Ratio and Mean Squared Error. Practical examples of animations using these techniques are presented. Finally, the interpolation methods and results are analyzed, showing characteristics associated with the techniques studied in the animations.

Keywords: Animation, Linear Interpolation, polynomial Interpolation, Cubic Splines, B-Splines

# LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| 2D | Duas Dimensões |
| 3D | Três Dimensões |
| 4D | Quatro Dimensões |
| AGL | Apple *Graphics Library* |
| AMD | Advanced Micro Devices, Inc. |
| API | *Application Programming Interface* |
| ARB | *Architecture Review Board* |
| AWT | *Abstract Window Toolkit* |
| CAD | *Computer Aided Design* |
| GLU | OpenGL *Utility Library* |
| GLUT | OpenGL *Utility Toolkit* |
| GLX | OpenGL *for* X Window System |
| IBM | International Business Machines Corporation |
| IDE | *Integrated Development Environment* |
| JOGL | Java OpenGL |
| JSR | Java *Specification Requests* |
| LERP | *Linear IntERPolation* |
| MSE | *Mean Squared Error* |
| NTSC | *National Television System(s) Committee* |
| NURBS | *NonUniform Rational B-Splines* |
| PAL | *Phase Alternating Line* |
| PDF | *Portable Document Format* |

| | |
|---|---|
| PSNR | *Peak Signal-to-Noise Ratio* |
| RGB | *Red, Green, Blue Color Model* |
| SGI | Silicon Graphics, Inc. |
| WGL | Windows OpenGL |
| WWW | *World Wide Web* |

**SUMÁRIO**

# 1 INTRODUÇÃO

A animação está associada à exibição de uma sucessão de imagens ou modelos, semelhantes, para criar uma ilusão de movimento, computacional ou não. Esta ilusão é conseqüência do fenômeno de persistência de visão: as seqüências são exibidas a taxas de quadros altas o bastante para produzir sensação de movimento contínuo em função da capacidade de retenção da imagem pela retina do olho humano (GOVIL-PAI, 2004, tradução nossa).

Há diversas técnicas para animação tais como tradicional, *stop-motion*, computacional, dentre outras. Na animação tradicional, os quadros individuais são inicialmente desenhados, capturados e transpostos para uma película. A *stop-motion* utiliza modelos reais em diversos materiais, tais como massa de modelar, figuras, desenhos, etc. Os modelos são movimentados e capturados quadro-a-quadro sendo posteriormente organizados e seqüenciados para criar a ilusão de movimento (PARENT, 2002, tradução nossa).

Na animação computacional, que é uma subárea de interseção entre a computação gráfica e a animação, as seqüências de imagens, os objetos da animação e a sensação de movimento são geradas por computador, envolvendo modelagem geométrica, controle de movimentos e renderização. Os quadros ou imagens são então registrados num processo denominado animação por quadros-chave (*keyframe animation*) (FOLEY et al, 1999, tradução nossa).

Para que seja possível a realização deste processo, quadros-chave precisam ser estabelecidos. Quadros-chave são aqueles que representam pontos bem definidos da animação, nos domínios de tempo e espaço, indicando valores de início e fim de cada objeto em movimento. E, com base nestes, quadros intermediários entre quadros-chave são obtidos

por técnicas de interpolação para que os movimentos sejam percebidos como contínuos. Este processo é conhecido como *inbetweening* (FOLEY et al, 1999, tradução nossa).

A interpolação, em um intervalo, resulta da inserção de amostras (quadros intermediários) que não constam em um conjunto de referência (quadros-chave), bem como do processo de aproximação dos valores de amostras (quadros intermediários) a partir dos valores da função no conjunto de referência (quadros-chave) (BUSS, 2003, tradução nossa).

Em uma seqüencia de quadros de animação, a interpolação visa inserir quadros (ou objetos) intermediários, entre quadros chave. A partir de um conjunto de quadros-chave (inicial, final e intermediários), geram-se quadros intermediários por meio de interpolação entre os pontos correspondentes dos objetos nos respectivos quadros-chave.

O método mais simples de interpolação é a interpolação linear (*Linear intERPolation - lerp*) (FOLEY et al, 1999, tradução nossa). Como característica, além da simplicidade, preserva a continuidade dos movimentos dos objetos. Entretanto, por considerar apenas dois quadros-chave a cada etapa de interpolação, apresenta limitações devido a variações abruptas de posição e velocidade nos movimentos, pois gera derivadas descontínuas no tempo e no espaço. Um exemplo clássico é o caso da bola saltitante (*bouncing ball*), que a cada vez que atinge o ponto máximo tem derivada e portanto velocidade nulas.

A interpolação baseada em *splines*, para o caso de animação, utiliza tipicamente polinômios suaves, continuamente diferenciáveis, de grau reduzido em cada subintervalo de quadros-chave (FOLEY et al, 1999, tradução nossa). Neste caso, são garantidas variações suaves de velocidade e posição (em tempo e espaço). Algumas desvantagens para utilização de interpolação baseada em *splines* são a necessidade de imposição de restrições a alguns pontos de controle, tais como vetores tangentes nulos, complexidade computacional elevada e dificuldade de implementação (KAPLAN, 2003, tradução nossa).

Outro caminho para interpolação é por meio de filtragem baseada em funções do tipo *splines* de base ou *basis-splines (b-splines*). Um grande atrativo para o uso de *splines* e *b-splines* é a suavidade que pode ser obtida pelas restrições de continuidade impostas para a função e suas derivadas (NICOLEIT; SEARA, 2001). Desta forma, obtém-se um alto grau de concordância nos pontos de amostragem no processo de construção de quadros intermediários.

A interpolação, dentro do processo de animação, não é, entretanto uma tarefa trivial. A qualidade da seqüência animada de imagens deve ser a melhor possível, segundo medidas objetivas, tais como Razão Sinal-Ruído de Pico (*Peak Signal-to-Noise Ratio* - PSNR) e Erro Médio Quadrático (*Mean Squared Error* - MSE) (NICOLEIT; SEARA, 2001), e complexidade computacional envolvida, sem contar a taxa de bits resultante, que não está no escopo desta pesquisa. Esta proposta visa avaliar comparativamente, em termos de qualidade resultante e complexidade computacional, técnicas de interpolação em seqüências de animação de quadros, bem como os algoritmos associados às implementações. Serão avaliados interpoladores lineares, baseados em *splines* cúbicas e em *b-splines*.

## 1.1 OBJETIVO GERAL

Avaliar comparativamente três técnicas de interpolação: Linear, com base em *Splines* Cúbicas, e em *B-splines*, utilizadas em animação por computador adotando como critérios a qualidade resultante e a complexidade computacional.

## 1.2 OBJETIVOS ESPECÍFICOS

a) Dominar os conceitos básicos de Computação Gráfica relacionados à animação computacional;

b) Descrever a interpolação em sua forma genérica, possibilitando a melhor compreensão das técnicas de interpolação;

c) Identificar as técnicas de interpolação linear, baseadas em *splines* cúbicas e em *b-splines* no contexto de animação computacional ressaltando as diferenças entre elas;

d) Avaliar e comparar características resultantes das técnicas de interpolação supracitadas quando aplicadas à animação gerada por computador, baseando-se em métricas objetivas, usando como critérios a qualidade e o custo computacional;

e) Demonstrar os resultados comparativos por meio da implementação dos algoritmos utilizados.

## 1.3 JUSTIFICATIVA

Atualmente a animação é vastamente utilizada em diversos setores, tais como indústria de entretenimento, simuladores de vôo, aplicações industriais, sistemas de controle, jogos, pesquisa científica e educação (FOLEY et al, 1999; MEALING, 1998, tradução nossa; AZEVEDO; CONCI, 2003).

Na animação computacional, a interpolação é uma tarefa importante. Segundo Neto e Melo (2004), o animador atém-se a produção dos quadros-chave, deixando os demais

quadros (intermediários) serem automaticamente calculados, contribuindo para agilizar o processo.

Entre as técnicas de interpolação, a lerp destaca-se pela sua simplicidade, mas gera variações abruptas de posição e velocidade nos movimentos dos objetos. Um exemplo de sua utilização é na animação conhecida como *Morphing*, onde um objeto se transforma em outro, semelhante a uma metamorfose (HEARN; BAKER, 1996, tradução nossa).

Das técnicas de interpolação baseadas em *splines* e em *b-splines* se esperam bons resultados, tratando-se de qualidade, na medida em que restrições de continuidade e suavidade são impostas para as *splines* de ordem *n* e suas derivadas até ordem *n-1*, nos pontos de amostragem. Exemplo de uso para *splines* é o movimento da câmera na cena e para as *b-splines* as animações detalhadas como expressões faciais (HOCH; FLEISCHMANN; GIROD, 1994; MEALING, 1998, tradução nossa).

A determinação da quantidade de quadros-chave necessários para se ter a impressão de continuidade nos movimentos depende do algoritmo a ser utilizado. Algoritmos mais simples, como os baseados em interpoladores lineares, necessitam de mais quadros-chave para apresentarem um resultado satisfatório em termos de qualidade resultante comparativamente aos interpoladores baseados em *splines* e *b-splines* que, por sua vez, apresentam maior complexidade (KAPLAN, 2003, tradução nossa). Portanto, a escolha do interpolador e do algoritmo associado está relacionada ao problema, sendo necessário optar por uma técnica com base nas características deste problema. Informações provenientes de comparações entre as técnicas são bastante relevantes na escolha da mais apropriada.

Geralmente uma animação utiliza mais de uma técnica de interpolação. Bons resultados são obtidos pela utilização do método apropriado em determinado tipo de movimento que faz parte da seqüência. O animador precisa conhecer o comportamento das

diversas técnicas de interpolação a que dispõe e aplicá-las na situação apropriada, de forma a atingir conscientemente os resultados esperados (GOVIL-PAI, 2004, tradução nossa).

Para evidenciar as características das técnicas, de forma a qualificar e a quantificar cada uma com respeito à qualidade e à complexidade computacional resultantes, bem como indicar a adequação para aplicação de cada uma das técnicas, serão implementados algoritmos representando os métodos de interpolação a serem avaliados.

## 1.4 ESTRUTURA DO TRABALHO

O presente trabalho apresenta cinco capítulos organizados de forma a situar o leitor no contexto macro onde a pesquisa está inserida e direcionando-o para o foco principal. O Capítulo Um demonstra essa evolução, contextualizando animação e interpolação para então apresentar o enfoque do trabalho. Neste capítulo também são descritos os objetivos e a justificativa da pesquisa.

O assunto que representa o suporte do trabalho está descrito no Capítulo Dois, abordando desde a origem da animação, a inserção do computador na sua produção até chegar à interpolação.

O Capítulo Três descreve as abordagens acerca de interpolação adotadas no trabalho, iniciando pela definição matemática, passando por sua conceituação na forma polinomial, por imagens digitais e culminando na sua aplicação em animação. São apresentadas as representações das técnicas de interpolação Linear, com base em *Splines* Cúbicas e por *B-splines* por meio destas abordagens.

Trabalhos relacionados ao assunto pesquisado são mostrados no Capítulo Quatro.

No Capítulo Cinco é apresentada a análise realizada. São demonstrados os aspectos de implementação bem como as tecnologias de programação utilizadas. Finalmente

foi abordada a analise comparativa, iniciando pela apresentação das ferramentas de medição adotadas e a situação montada para coleta dos dados e culminando na construção de uma tabela comparativa. Encerra-se o capítulo inferindo acerca dos resultados alcançados.

# 2 ANIMAÇÃO

Animar significa dar vida, movimentar objetos ou imagens que não possuem autonomia para se moverem (MCKINLEY, 2006; PARENT, 2002, tradução nossa). Esta arte tão magnífica e desafiadora consiste em mostrar uma seqüência de imagens ligeiramente diferentes, conhecidas por quadros, de forma rápida o bastante para criar ilusão de movimento (GOVIL-PAI, 2004; KAPLAN, 2003; MCKINLEY, 2006, tradução nossa).

Desde que foi criada por Winsor McCay, a forma de criação das animações é essencialmente a mesma. Considerando a grande contribuição do famoso animador Walt Disney, que agregou cores e sons e acelerou o processo de produção da animação, ainda hoje continuam existindo dois conceitos bem definidos e trabalhados separadamente: o primeiro é a criação dos quadros principais, realizada pelos melhores animadores da empresa, onde são definidas as mudanças mais significativas nos movimentos dos personagens. O segundo conceito é a produção dos quadros intermediários, realizada por animadores assistentes e que proporcionam uma transição suave entre os quadros principais previamente produzidos (MULLEN, 2007; KUPERBERG et al, 2002, tradução nossa).

Houve muitos aperfeiçoamentos, na animação como um todo, dos seus primórdios até os dias de hoje, principalmente com o advento do computador. Destacam-se a automatização de alguns processos e a qualidade proporcionada pelos avanços tecnológicos (AVGERAKIS, 2004, tradução nossa).

Quanto à automatização, a trabalhosa tarefa de geração dos quadros intermediários pôde ser realizada totalmente pelo computador, por meio de interpolação, assunto principal deste trabalho (KUPERBERG et al, 2002; MULLEN, 2007, tradução nossa).

Sobre a qualidade gráfica das animações, as novas tecnologias no campo da computação gráfica, tanto em hardware quanto em software, alavancaram a indústria, criando

uma nova geração de animações, recheadas de efeitos especiais, modelagens e movimentos ricos em detalhes (MULLEN, 2007, tradução nossa).

No estágio atual, onde há ferramentas gratuitas de modelagem e produção de animação, como o projeto Blender, por exemplo, pequenas animações podem ser criadas por poucos ou até mesmo uma pessoa em tempo relativamente curto e com ótima qualidade final. Para isto, o animador concentra-se basicamente na criação dos quadros principais, pelos quais são expressas as suas idéias (AVGERAKIS, 2004, tradução nossa).

Entre as formas de utilização da animação destacam-se o entretenimento, como filmes e vídeos; marketing, instrumento de ensino, principalmente para crianças, estimulando a imaginação e as emoções; ferramenta científica, auxiliando com simulações de vários fenômenos, como por exemplo, reações químicas entre outras (FOLEY et al, 1999; PARENT, 2002, tradução nossa).

No campo da ciência, a animação está inserida em um grupo chamado de visualização científica, junto com outras disciplinas como processamento de sinal e geometria computacional (FOLEY et al, 1999, tradução nossa).

## 2.1 PERCEPÇÃO DE ANIMAÇÃO

O sistema visual humano, juntamente com o cérebro, possui uma característica denominada persistência de visão (VERTH; BISHOP, 2008, tradução nossa). Ao ver uma imagem, esta fica retida no cérebro por um curto espaço de tempo. Se a imagem seguinte for apresentada antes que a anterior seja perdida pelo cérebro, ocorre uma transição suave entre os quadros e conseqüentemente a sensação de movimento (GOVIL-PAI, 2004, tradução nossa).

A animação só é possível graças à amostragem de quadros ao longo do tempo. Uma medida de referência adotada em vídeo e em animação é a taxa de quadros apresentadas por segundo, também conhecida pelos termos em inglês *framerate* ou *frames per second* (fps). Para que haja percepção de movimento, a transição entre quadros deve ser suave, e os parâmetros associados serão de acordo com o suporte onde a animação será exibida. Para televisão analógica padrão *National Television System(s) Committee* (NTSC), adotado nos Estados Unidos, o sinal de vídeo é amostrado a 30 fps. Para o padrão *Phase Alternating Line* (PAL), usado no continente europeu, 25 fps. A amostragem para cinema é realizada com 24 fps (AVGERAKIS, 2004; GOVIL-PAI, 2004, tradução nossa).

## 2.2 FORMAS DE PRODUÇÃO DE ANIMAÇÃO

Há diversas técnicas para criar animação, dentre elas a convencional, a *stop-motion* e a computacional.

Na animação convencional, os quadros são desenhados a mão. Criam-se, primeiramente, os quadros principais, que apresentam alguma mudança significativa na seqüência da animação e posteriormente são produzidos os quadros intermediários, usados para promover uma transição suave entre os quadros principais. Finalmente os quadros são gravados em uma película, para serem tipicamente projetados (MEALING, 1998, tradução nossa).

A *stop-motion* utiliza modelos reais, como por exemplo, massa de modelar, bonecos ou qualquer material que possibilite construir uma cena. Os modelos são posicionados e a câmera registra uma imagem. Modifica-se a cena reposicionando os modelos e a câmera obtém outra imagem, seguindo desta forma até que a seqüência seja concluída (PARENT, 2002, tradução nossa).

Na animação computacional há elementos das técnicas acima mencionadas. Com a animação convencional foram aproveitados conceitos como a relação dos objetos e personagens com as leis da física, a manipulação do tempo, às vezes mais lento, outras vezes mais rápido, o fluxo da animação, para manter a continuidade através das cenas entre outros. Com relação à técnica *stop-motion*, a idéia de construir os modelos para registrar as imagens é vastamente usada (PARENT, 2002, tradução nossa).

## 2.3 ANIMAÇÃO COMPUTACIONAL

A execução de uma seqüência animada produzida com o auxílio do computador é denominada animação computacional. Sua forma de produção mais comum é conhecida como animação por quadro-chave, ou *keyframe animation*, onde são definidos quadros-chave e obtidos quadros intermediários baseados nestes (PARENT, 2002, tradução nossa).

Os quadros-chave, assim como na animação convencional, continuam sendo produzidos por intervenção humana, porém a trabalhosa tarefa da produção de quadros intermediários pode ser realizada totalmente pelo computador. Além disto, outro ponto positivo é a redução de memória requerida para armazenar as seqüências, pois somente os quadros-chave precisam ser guardados (PULLI et al, 2008, tradução nossa).

A forma de utilização do computador para produzir seqüências animadas costuma ser classificada em dois grupos: animação assistida pelo computador e animação gerada pelo computador.

## 2.4 ANIMAÇÃO ASSISTIDA PELO COMPUTADOR

Nesta forma de animação, o computador é usado apenas para auxiliar e agilizar a técnica convencional de animação feita a mão. O animador desenha os quadros principais e digitaliza-os. Com base nestes, o computador gera automaticamente os quadros intermediários em um processo conhecido como *inbetweening*, por vezes chamado de *tweening*. (GOVIL-PAI, 2004; KAPLAN, 2003; MCKINLEY, 2006, tradução nossa).

A produção da animação, na forma convencional, ou seja, sem a utilização do computador, costuma ser organizada seqüencialmente com os seguintes passos (CATMULL, 1978, tradução nossa):

- roteiro;
- storyboard;
- gravação da parte sonora;
- layout detalhado;
- inclusão dos sons;
- criação dos quadros-chave;
- geração dos quadros intermediários;
- execução de testes para ajuste de tempo;
- transposição dos quadros para a folha de acetato;
- pintura das imagens transpostas;
- verificação de erros;
- fotografia e edição dos quadros para a finalização do trabalho.

Muitos dos processos citados acima podem ser auxiliados pelo computador, entre eles:

- geração dos quadros intermediários;

- tratamento das imagens;
- edição e inclusão das trilhas sonoras;
- testes;
- edição final da animação.

Para que o computador faça parte do processo é necessário digitalizar os desenhos. Usam-se, para isto, *scanners*, *tablets*, ou até mesmo desenha-se diretamente usando softwares específicos. Realizada esta etapa inicial, os outros passos podem então ser executados. Para os propósitos deste trabalho, será abordado com maior ênfase o processo de geração de quadros intermediários (FOLEY, 1999, tradução nossa).

## 2.5 ANIMAÇÃO GERADA PELO COMPUTADOR

Também chamada de animação 3D, é a técnica padrão para a produção de animações e diferencia-se da anterior na forma de produção dos quadros-chave. Estes são criados pelo animador a partir de modelos geométricos desenvolvidos geralmente em softwares de modelagem 3D como Maya, 3DS Max e Blender utilizando um conceito semelhante ao da animação em *stop-motion*, obtendo os quadros-chave por meio do posicionamento dos modelos criados. Quadros intermediários, ou *inbetweening,* também são obtidos por meio de interpolação (MCKINLEY, 2006, tradução nossa).

Além do posicionamento dos objetos na cena, outros parâmetros podem ser usados para a produção de animações geradas no computador. Exemplos são mudanças nas propriedades da câmera, como posição, orientação e o foco. Efeitos de luz e sombra isoladamente também podem dar origem a animações (HEARN; BAKER, 1996, tradução nossa).

Animações geradas pelo computador podem ser usadas para as mais diversas finalidades. Para fins de entretenimento, simulação e treinamento, por exemplo, geralmente é desejável que a qualidade visual e os movimentos sejam realistas. No entanto, em alguns projetos científicos, os efeitos visuais são de pouca importância, pois os modelos são simbólicos e não fazem parte do objetivo principal do estudo (HEARN; BAKER, 1996, tradução nossa).

A criação de animações geralmente segue um planejamento bem definido e resulta em um trabalho estático com quadros pré-renderizados (produzidos de forma isolada). Porém há categorias de animação que exigem uma abordagem mais dinâmica. Exemplos destas exceções são jogos digitais, simuladores e visualizações científicas, onde a exibição dos quadros varia de acordo com os parâmetros que são passados dinamicamente, em tempo real (HEARN; BAKER, 1996, tradução nossa). Portanto, cada tipo de animação exige uma abordagem diferente na escolha da técnica de interpolação a ser utilizada.

# 3 INTERPOLAÇÃO

O termo interpolação varia de acordo com o contexto onde é aplicado. A definição mais abrangente remete a sua origem, do Latim *inter* (entre) e *polare* (polir) resumindo-se no processo de encontrar novos valores que estão localizados entre certos valores conhecidos (SALOMON, 2006, tradução nossa). Para que isso seja possível, a função original (desconhecida) é aproximada por outra função, substituindo-a. Além disto, a interpolação se aplica também em substituição a uma função muito difícil ou até impossível de realizar certas operações tais como diferenciação e integração, onde a simplicidade compensa o erro da aproximação entre as duas (RUGGIERO; LOPES, 1996). Outras formas de aplicação da interpolação são em imagens digitais e em animação, com destaque para esta última, apresentadas aqui em forma de polinômio.

## 3.1 INTERPOLAÇÃO EM IMAGENS DIGITAIS

As imagens digitais representam cenas, tanto em processos intermediários, por exemplo, a interface de um software de modelagem, quanto em resultados finais, como na exibição de uma animação (GOMES; VELHO, 1994).

Classifica-se uma imagem como um sinal bidimensional devido à sua forma de representação, com informações dispostas em linha e coluna, ou largura e altura (GOMES; VELHO, 1994). Adiante neste capítulo serão apresentados com mais detalhes os conceitos acerca de imagem.

Segundo Gomes & Velho (1994), como qualquer sinal, a imagem pode estar representada em um dos três níveis de abstração conhecidos: contínuo, discreto e simbólico.

No domínio contínuo, a imagem apresenta-se na forma original, mais abrangente, contendo todas as suas características.

O nível discreto é uma representação limitada do nível contínuo. As informações presentes na imagem original são mapeadas para um formato discreto utilizando modelos de representação finitos para cada uma de suas características, possibilitando que seja visualizada em equipamentos eletrônicos. Esta operação é chamada de discretização (GOMES; VELHO, 1994).

E a representação simbólica envolve o armazenamento das informações obtidas no processo de discretização em um formato de arquivo de dados (codificação), possibilitando que a imagem discretizada possa ser recuperada (decodificação).

O procedimento de conversão de uma imagem discretizada em sua representação contínua é chamado de reconstrução. A situação ótima seria poder reconstruir o sinal contínuo original sem perda de informações, porém isto geralmente não acontece. Para tratar destes casos, diversas técnicas podem ser aplicadas com a finalidade de obter um sinal próximo ao original. Usam-se, para isto, métodos de interpolação buscando uma razoável qualidade de reconstrução (GOMES; VELHO, 1994).

Para facilitar o entendimento sobre interpolação aplicada em imagens, é conveniente discutir primeiramente os princípios que envolvem imagens digitais, bem como suas características.

### 3.1.1 Imagens Digitais

Imagem digital é a representação discreta em duas dimensões de uma imagem contínua por meio de um conjunto de pontos definidos matematicamente ou por uma matriz (GOMES; VELHO, 1994; GREENBERG, 2007, tradução nossa).

Quando a determinação dos pontos é dada por meio de expressões, diz-se que a imagem é vetorial. Nesta forma de representação, os pontos da imagem são definidos por vetores e primitivas geométricas. Entre as suas vantagens estão o pouco espaço necessário para armazenamento e a capacidade de redimensionamento da imagem sem perda de qualidade. Em contrapartida, não é possível representar imagens muito complexas, tais como fotografias. Exemplos conhecidos de utilização são os arquivos em formato de documento portável, ou PDF e as animações em Flash da Adobe (GREENBERG, 2007, tradução nossa).

Se a imagem digital for representada por uma matriz de pontos, ela é do tipo *raster*, mais conhecida como *bitmap.* É a chamada representação matricial, onde ficam registradas no arquivo as informações referentes a cada ponto que a compõe. Desta forma, a quantidade de pontos é diretamente proporcional à qualidade da imagem e ao tamanho do arquivo. Outras características determinantes são as informações de cada ponto: localização espacial e cor (GREENBERG, 2007, tradução nossa).

Cada ponto de um *bitmap* é chamado de *picture element*, o popular *pixel,* que é a menor unidade de uma imagem digital. Eles estão diretamente ligados a duas informações determinantes para a qualidade da imagem criada: a resolução espacial e a resolução de cor (GOMES; VELHO, 1994).

A resolução espacial consiste na discretização do espaço físico da imagem na forma de uma matriz retangular. Na prática é o produto entre o número de linhas e o número de colunas da matriz. Sendo assim, quanto maior for este produto, maior será o nível de detalhamento da imagem digital. Entretanto, esta forma de medida de qualidade não é a mais correta, pois depende do tamanho do pixel suportado pelo dispositivo físico de captura da imagem. Uma métrica bastante utilizada, que aplica uma unidade de medida, expressa a quantidade de pixels presentes em uma polegada, ou 2,54 cm. Esta métrica chama-se *dots per inch* ou dpi (GOMES; VELHO, 1994).

Já a resolução de cor é definida pelo resultado do processo de discretização do espaço de cor da imagem, conhecido por quantização, em um espaço de cor suportado fisicamente pelo equipamento de visualização. Desta forma, a qualidade varia de acordo com a profundidade de cor, ou a quantidade de bits máxima que o dispositivo gráfico disponibiliza para cada pixel. Por exemplo, com 16 bits, é possível representar $2^{16}$ ou 65.536 cores distintas. Usando uma profundidade de 24 bits por pixel, podem-se visualizar até 16.777.216 cores (JUNIOR, 2006; GOMES; VELHO, 1994).

### 3.1.2 Representação de Cores

As cores advêm de radiações eletromagnéticas com comprimentos de onda variados. O sistema visual humano é sensível somente a comprimentos de onda compreendidos entre 400 e 700 nanômetros, que correspondem às cores vermelha e violeta, respectivamente (AZEVEDO; CONCI, 2003).

Estudos realizados pelo cientista Thomas Young, posteriormente continuados pelo físico alemão Hermann von Helmholtz concluíram que a retina possui receptores com sensibilidade a três tipos diferentes de comprimento de onda; curtos, médios e longos. Os curtos respondem melhor em comprimentos de onda de aproximadamente 445 nm. Os médios trabalham na faixa de 535 nm e os longos próximos a 575 nm. Estes comprimentos determinam as cores azul, verde e vermelha, respectivamente, embora o comprimento de 575 nm esteja mais próximo da cor amarela (AZEVEDO; CONCI, 2003).

| Violeta | Azul | Ciano | Verde | Amarelo | Laranja | Vermelho |
|---|---|---|---|---|---|---|
| 380-450 | 450-480 | 480-490 | 490-560 | 560-580 | 580-600 | 600-700 |

Figura 1. Comprimentos de onda do espectro visível em nanômetros
Fonte: Adaptado de Azevedo e Conci, 2003

Os estudos demonstraram ainda que, pela soma destas três cores de referência é possível representar praticamente todas as cores do espectro visível apresentado na Figura 1.

Estas definições são a base para o conceito das cores primárias *red, green e blue (rgb),* que determinam alguns dos principais sistemas de cores utilizados atualmente (AZEVEDO; CONCI, 2003). Para os propósitos deste trabalho, será abordado somente o sistema RGB.

### 3.1.3 Sistema de Cores RGB

O modelo RGB, usado principalmente em monitores de vídeo e imagens digitais, é representado por um sistema cartesiano contendo três dimensões, conforme Figura 2. Nela, cada eixo do sistema assume uma das cores primárias *rgb* (FOLEY et al, 1999, tradução nossa).

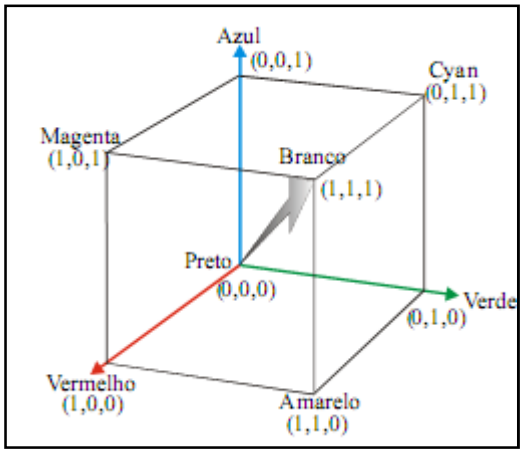


Figura 2. Eixos cartesianos do modelo RGB
Fonte: MIRANDA, J. I.; MARABITA, A. M. (2007)

Todas as cores representadas pelo modelo RGB podem ser obtidas por meio da adição das três cores primárias ( Figura 3). Sendo assim, para formar o branco são

combinados os valores máximos dos três eixos. Conseqüentemente, o preto é gerado pelos seus valores mínimos. Por estes motivos, o modelo RGB é um sistema de cores aditivo (AZEVEDO; CONCI, 2003).


Figura 3. Modelo de cores aditivas RGB
Fonte: Adaptado de Foley et al, 1999, tradução nossa

Analisando a Figura 2, percebe-se que, para formar a cor Azul, são usados os valores (0,0,1), que correspondem às cores Vermelha, Verde e Azul, respectivamente (considerando que a intensidade relativa da cor varia de zero a um no domínio dos números reais). Já o Magenta é a combinação dos valores extremos das primárias azul e vermelha (0,1,1). A flecha diagonal representa a escala de cinza, que é a variação de tonalidades entre preto e branco, formada por valores iguais nos três eixos (FOLEY et al, 1999, tradução nossa).

O valor máximo absoluto que cada eixo do sistema RGB pode assumir é igual à capacidade do dispositivo gráfico, medido pela quantidade de bits suportados por pixel. Quanto maior for esta quantidade, mais cores são geradas. Com 3 bits são representadas apenas as oito cores da Figura 2, pois cada eixo faz uso somente de 1 bit, ou seja, zero ou um. Usando 24 bits, cada eixo pode assumir 256 valores diferentes (8 bits por cor primária), sendo possível visualizar, conforme citado anteriormente, mais de 16 milhões de cores (JUNIOR, 2006).

### 3.1.4 Interpolação em Espaço e Cor

Eventualmente há a necessidade de aplicar transformações geométricas em uma imagem, em especial rotação e escalamento. Outras vezes é necessário adaptar a imagem as características de um determinado dispositivo gráfico (UNSER; ALDROUBI; EDEN, 1991, tradução nossa). Ao realizar tais procedimentos, as resoluções de espaço e cor da imagem geralmente são alteradas. Estas operações de processamento de imagem fazem parte do processo de reconstrução de imagem, tratado por métodos de interpolação, como fora anteriormente mencionado.

Para o caso do escalamento, quando as dimensões da imagem forem duplicadas, por exemplo, sua resolução espacial deve também ser aumentada. Para resolver este problema, criam-se novos pixels que preenchem os espaços vazios, culminando em outro problema envolvendo a resolução de cor: determinar a cor do novo pixel. São estas situações que a interpolação, por meio de diversos métodos, pode ser aplicada, buscando prover boa representação da imagem com custo computacional reduzido.

Outro aspecto importante é que o sistema visual humano define os objetos por meio dos seus contornos. Métodos de interpolação com elevados graus de continuidade provocam uma suavização na transição das cores entre os pixels em toda a imagem, incluindo os contornos. Esta peculiaridade caracteriza um ponto negativo a ser considerado, desencorajando o uso de polinômios de ordem elevada, com graus de continuidade muito altos. Com efeito, as imagens resultantes ficam com aparência de borradas. Por outro lado, interpoladores descontínuos tendem a criar novos contornos (artefatos), distorcendo a informação original da imagem (GOMES; VELHO, 1998).

## 3.2 INTERPOLAÇÃO EM ANIMAÇÃO

No contexto de animação computacional, interpolar significa inserir quadros intermediários com base em quadros-chave pré-determinados. Este processo varia de acordo com a classificação da animação quanto à forma de utilização do computador. Como previamente mencionado, há duas classificações: animação assistida pelo computador e animação gerada pelo computador.

Quanto à primeira, o *inbetweening* é processado com base nos quadros-chave renderizados, ou seja, os pixels são tratados diretamente para formar quadros ligeiramente diferenciados. Para um animador humano, esta é uma tarefa fácil no sentido de que a imagem desenhada é uma representação bidimensional de um modelo tridimensional construído na sua mente. Ao desenhar o quadro, informações deste modelo são perdidas quando, por exemplo, objetos são sobrepostos. Na imaginação sabe-se que o objeto está ali, mesmo sem estar aparecendo. Para o computador é difícil ter este tipo de percepção (CATMULL, 1978, tradução nossa).

Na animação gerada por computador, a interpolação deve considerar, além dos movimentos dos objetos, ou atores, todos os efeitos visuais envolvidos na cena, tais como variações nas transparências, cores, sombras, câmeras virtuais, dentre outros. Diferentemente da animação assistida pelo computador, que opera sobre pixels, os parâmetros interpolados são todas as características obtidas dos quadros-chave, como valores de posicionamento, cores e intensidades. Por esse motivo, o método de criação é chamado de animação por quadro-chave paramétrica. Dependendo do posicionamento de um objeto em vários quadros-chave consecutivos, ele percorrerá uma trajetória retilínea ou em curva (BUSS, 2003; FOLEY et al, 1999; GOVIL-PAI, 2004; MCKINLEY, 2006, tradução nossa).

Um grande desafio inerente à interpolação e à animação é a amostragem no tempo. Enquanto que nas representações de imagens em duas dimensões (2D) e três dimensões (3D) somente as características dos pontos são calculadas, na animação, há a preocupação com a velocidade de transição dos objetos, entendida como a quarta dimensão (4D). O método de interpolação deve também suprir estas variações de velocidade, pois em muitos casos ela não segue uma linearidade (FOLEY et al, 1999, tradução nossa).

Para auxiliar na escolha da técnica mais apropriada, há algumas características da seqüência animada a definir: interpolação ou aproximação, complexidade, continuidade e controle global ou local (PARENT, 2002, tradução nossa).

Na interpolação, a trajetória calculada deve passar pelos pontos usados como parâmetro. Já na aproximação, os pontos dados apenas controlam (orientam) a trajetória criada, sem fazer parte dela. É importante citar que o a interpolação citada até aqui difere deste conceito. O termo interpolação está sendo usado neste trabalho como conjunto de técnicas usadas para encontrar valores intermediários entre valores de referência, independentemente se o método adotado passa ou não pelos pontos de controle. No entanto, a definição aqui apresentada é importante para escolha do método apropriado a uma determinada situação (PARENT, 2002, tradução nossa).

A complexidade refere-se ao custo computacional. Dependendo da finalidade da animação, este quesito pode ser de suma importância. Polinômios são muito usados por serem facilmente computáveis e terem características que suprem as necessidades impostas, como trajetórias, acelerações e desacelerações suaves, principalmente nos pontos de transição. A utilização de polinômios cúbicos merece atenção especial, por promover satisfatório grau de suavidade e flexibilidade e com custo computacional reduzido comparado ao custo computacional para processar polinômios de grau mais elevado (FOLEY, et al, 1999; PARENT, 2002, tradução nossa).

Continuidade é o termo que indica a suavidade de uma curva, ilustrado na Figura 4. A análise da continuidade de uma curva expressa por uma equação pode ser realizada por meio das suas derivadas. Expressões sem derivada em todos os pontos apresentam continuidade de ordem zero. Equações Lineares são exemplos destas expressões. Caso a equação possua a primeira derivada, sua continuidade é de ordem um. Polinômios quadráticos possuem essa continuidade. Expressões que apresentam derivação até segunda ordem possuem continuidade de ordem dois, representadas, por exemplo, por polinômios cúbicos.

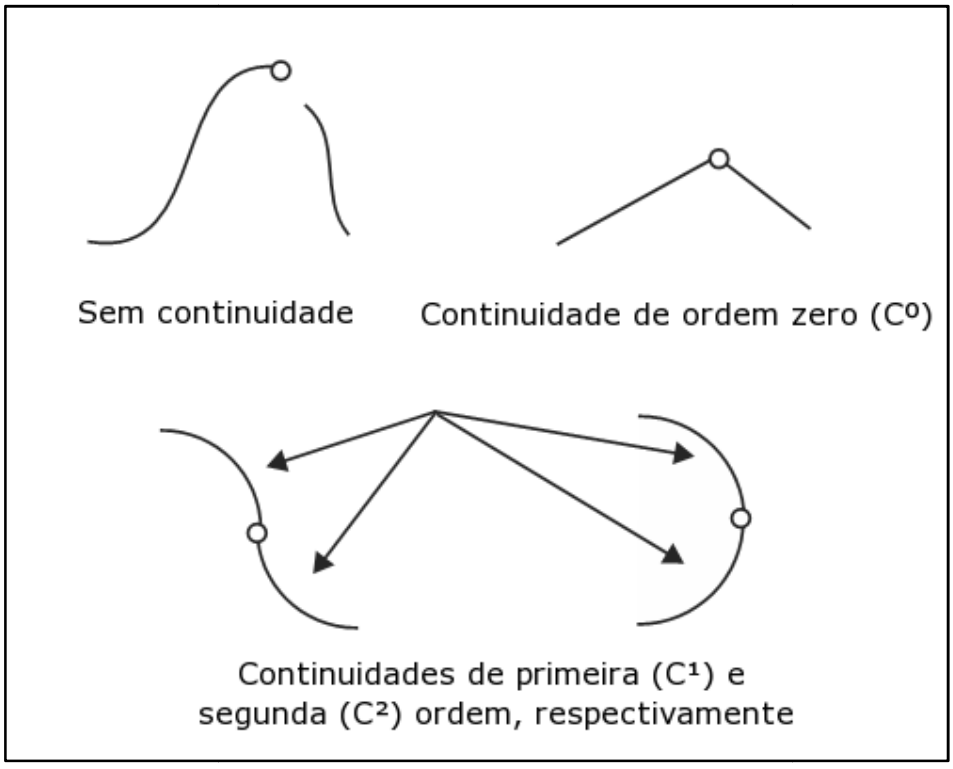


Figura 4. Grau de continuidade de uma curva
Fonte: PARENT, R. (2002, tradução nossa)

E quanto ao tipo de controle, a classificação varia de acordo com a influência que a alteração de determinado ponto exerce sobre a trajetória. Caso este ponto interfira no comportamento apenas dos pontos vizinhos, tem-se um controle local. Em contrapartida, o

controle global refere-se alteração do ponto refletir em toda a trajetória (PARENT, 2002, tradução nossa).

Os quadros-chave são quadros importantes onde o animador constrói uma cena. Estes quadros especiais são referências não somente para o animador, mas também para o processo de interpolação. É com base neles que o algoritmo cria quadros intermediários com a finalidade de promover uma transição suave entre dois quadros-chave consecutivos. Com a inclusão de mais quadros de referência, é possível a criação de animações bastante complexas (AVGERAKIS, 2004, tradução nossa).

A quantidade de quadros intermediários necessária depende da mídia usada na apresentação da animação. Para a televisão padrão NTSC, usam-se, ao todo, 24 quadros para cada segundo, incluindo os quadros-chave. Já filmes necessitam de 24 fps para que a animação seja visualizada (HEARN; BAKER, 1996, tradução nossa).

## 3.3 INTERPOLAÇÃO POLINOMIAL

Geralmente os métodos de interpolação existentes usam polinômios para aproximar funções. Justifica-se principalmente por facilitar os cálculos, tal que são facilmente representados no computador. Além disto, operações de derivação e integração geram novos polinômios (BURDEN; FAIRES, 2003). Resumindo, segundo o teorema de Weierstrass: "Toda função contínua pode ser arbitrariamente aproximada por um polinômio" (FRANCO, 2006).

Dados $n + 1$ pontos diferentes $(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)$ relacionados por $y=f(x)$, encontra-se um polinômio $P_n(x)$ tendo $n$ como grau máximo e que passa pelos pontos da função amostrada $f(x)$ em $(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)$ dado por:

$$P_n(x_0) = y_0,\ P_n(x_1) = y_1,\ P_n(x_2) = y_2,\ldots, P_n(x_n) = y_n.$$

Considerando a situação apresentada, onde os pontos são diferentes, o polinômio encontrado é único (FRANCO, 2006).

Existem diversas técnicas de interpolação, entre elas a Linear, Bilinear, Trilinear, com base em curvas de Bézier, em *Splines*, em *B-splines* entre outras (BUSS, 2003; SALOMON, 2006, tradução nossa). Para efeitos deste trabalho, recebem destaque a interpolação Linear, baseada em *Splines* Cúbicas e em *B-splines*. Todas elas utilizam polinômios.

## 3.4 INTERPOLAÇÃO LINEAR

A interpolação polinomial mencionada tem grau *n*, onde *n+1* é o número de pontos obtidos a partir da função a ser interpolada. Mesmo sendo facilmente computável, um polinômio com grau muito elevado exige cálculos em demasia, podendo ter um alto custo.

A interpolação linear também utiliza um polinômio, com a diferença de este ser limitado ao primeiro grau. Esta limitação acarreta um processamento mais simples, portanto, mais veloz que interpoladores que utilizam graus superiores. Conseqüentemente, o polinômio $P_1(x)$ substitui uma função $f(x)$ em intervalos compreendidos entre dois pontos $(x_0, y_0)$ e $(x_1, y_1)$, passando por eles. O polinômio será único se os pontos forem distintos, tal que $P_1(x_0) = f(x_0)$ e $P_1(x_1) = f(x_1)$ (FRANCO, 2006; BURDEN; FAIRES, 2003).

A equação associada à interpolação linear é a Equação 3.1, que para implementação pode ser transformada na Equação 3.2.

$$\boxed{y = y_0 \frac{(x-x_1)}{(x_0-x_1)} + y_1 \frac{(x-x_0)}{(x_1-x_0)}} \quad (3.1)$$

$$\boxed{y = (y_1 - y_0)u + y_0} \quad (3.2)$$

A variável *y* é o valor interpolado em função das extremidades $y_0$ e $y_1$. A incógnita *u* assume valores entre zero e um, para zero estando sobre $y_0$ e para um localizado sobre $y_1$.

### 3.4.1 Interpolação Linear em Imagens Digitais

Imagens digitais são matrizes de pixels com informação de cor associada a cada elemento. Este é espacialmente representado por um par ordenado *(x, y)* indicando sua posição horizontal e vertical, respectivamente. Quando as dimensões da imagem são aumentadas, novos pixels são intercalados para manter a semelhança com a imagem original.

O método de interpolação linear é o mais simples para realizar este procedimento, pois considera somente os dois pixels vizinhos mais próximos. Por isto é necessário que sejam realizados cálculos isolados tanto na direção horizontal quanto na direção vertical.

Considerando a interpolação na direção horizontal, a cor do pixel interpolado (*RGB*) é encontrada a partir de dois pontos cujos coeficientes de cores são $RGB_0$ e $RGB_1$. A Equação 3.2 resulta na Equação 3.3.

$$\boxed{RGB = (RGB_1 - RGB_0)u + RGB_0} \quad (3.3)$$

Para aplicar a interpolação na direção vertical, a mesma Equação 3.3 pode ser utilizada, alterando apenas os valores para $RGB_0$ e $RGB_1$ de forma que um pixel esteja localizado acima do outro.

### 3.4.2 Interpolação Linear em Animação

Sendo a técnica mais simples, a interpolação Linear, ou *Linear intERPolation (lerp)* considera apenas dois quadros-chave consecutivos e a função interpoladora determina um deslocamento em linha reta entre eles. Dados os quadros $Q_0$ e $Q_1$ como referência, determina-se um quadro qualquer entre eles por meio da Equação 3.4, onde *t* representa o tempo e conseqüentemente o deslocamento com relação ao ponto inicial e $t \in [0,1]$ (VINCE, 2006; KAPLAN, 2003, tradução nossa).

$$\boxed{Q(t) = (Q_1 - Q_0)t + Q_0} \quad (3.4)$$

A *lerp* é uma técnica útil em muitos casos, por sua simplicidade e conseqüente baixo custo computacional. Além disto, a taxa de variação entre dois quadros-chave é constante, sendo facilmente controlável. Entretanto não é possível alcançar resultados melhores em termos de realismo em algumas animações, pois são geradas derivadas descontínuas em todos os seus pontos ( Figura 5), causando mudanças abruptas de posição e velocidade. Um exemplo deste efeito é a bola saltitante da Figura 6 (KAPLAN, 2003; PULLI et al, 2008; SALOMON, 2006; FOLEY et al, 1999, tradução nossa).

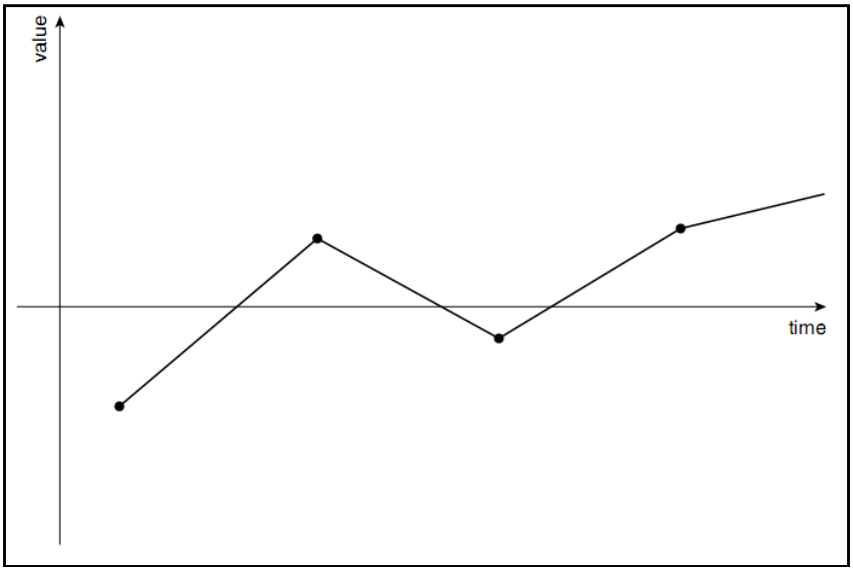


Figura 5. Continuidade Cº da lerp
Fonte: PULLI et al (2008)

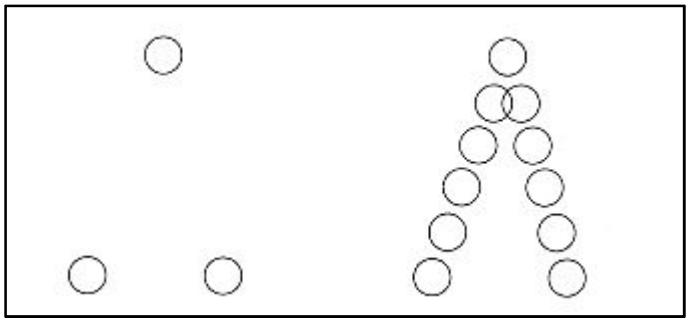

Figura 6. *Bouncing Ball*
Fonte: FOLEY et al. (1999)

## 3.5 INTERPOLAÇÃO BASEADA EM *SPLINES* CÚBICAS

*Spline* é um termo genérico adotado para um conjunto de técnicas utilizadas na representação de curvas e superfícies formadas por segmentos contínuos, como por exemplo, funções polinomiais, que passam por determinados pontos (MAZZOLA; MILMEISTER; WEISSMANN, 2005, tradução nossa).

Este nome se origina de uma régua flexível de madeira ou plástico usada em desenhos industriais. Ela é curvada de tal forma que passa por pontos pré-estabelecidos, também chamados de nós. Embora a régua *spline* seja usada desde o século XX, foi

matematicamente definida somente no final da década de 1960. Estas definições se baseiam nas relações entre os nós e a curvatura da régua formada entre eles (CLÁUDIO; MARINS, 2000; RUGGIERO; LOPES, 1996).

Técnicas de interpolação mais comuns baseadas em polinômios utilizam uma expressão para representar todos os pontos do intervalo (polinômio de grau igual a *n-1* pontos). Desta forma, quanto maior for a quantidade de pontos neste intervalo, maior será o grau do polinômio. Há algumas desvantagens nestes métodos de interpolação; a complexidade e o custo são diretamente proporcionais ao grau do polinômio interpolador e alterações em uma pequena parte do intervalo geram grandes variações para o restante do intervalo (BURDEN; FAIRES, 2003).

Utilizando *spline*, o intervalo inteiro é dividido em vários subintervalos onde cada um destes é representado por um polinômio. No caso de *splines* cúbicas, o polinômio é de terceiro grau, o que garante derivação contínua até a segunda ordem, permitindo uma transição suave entre os pontos do intervalo. Os polinômios conectados por um nó devem possuir as mesmas derivadas (BURDEN; FAIRES, 2003; SALOMON, 2006, tradução nossa).

Dados n pontos de uma função $(x_0, y_0) < (x_1, y_1) < ... < (x_n, y_n)$ a *spline* cúbica $S(x)$ que interpola a função $f(x)$ é definida sob as seguintes condições:

*a)* $S(x)$, indicado por $S_i(x)$ interpola a $f(x)$ no subintervalo $[x_i,x_{i+1}]$ para $i=[0,1]$;

*b)* $S(x_i) = f(x_i)$ para $0 \leq i \leq n$;

*c)* $S_{i+1}(x_{i+1}) = S_i(x_{i+1})$ para $0 \leq i \leq n-2$;

*d)* $S'_{i+1}(x_{i+1}) = S'_i(x_{i+1})$ para $0 \leq i \leq n-2$;

*e)* $S''_{i+1}(x_{i+1}) = S''_i(x_{i+1})$ para $0 \leq i \leq n-2$;

Aplicar estas condições sobre o polinômio de terceiro grau da Equação 5.1 não é suficiente para calcular os coeficientes *a, b, c e d*. A solução é obtida impondo mais duas

condições, que variam de acordo com o problema. Duas alternativas são o contorno natural e o contorno restrito.

$$\boxed{S_3(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \text{ para } i = 1, 2, 3, \ldots, n} \quad (5.1)$$

O contorno natural ou livre impõe as seguintes condições:

a) $S_3''(x_0) = 0$;

b) $S_3''(x_n) = 0$.

O contorno restrito requer as condições abaixo:

a) $S_3'(x_0) = f'(x_0)$;

b) $S_3'(x_n) = f'(x_n)$.

O contorno restrito é mais preciso, pois contém maiores informações sobre a função interpolada. Em contrapartida, requer que a derivada da função seja conhecida (ou bem próxima) nas extremidades (BURDEN; FAIRES, 2003; RUGGIERO; LOPES, 1996).

### 3.5.1 Interpolação por *Splines* Cúbicas Utilizada em Imagens Digitais

As equações apresentadas anteriormente neste capítulo são usadas para calcular o valor de uma incógnita. Sabendo que uma imagem é armazenada no computador em um vetor unidimensional, a mesma formulação pode ser utilizada, adotando como incógnita a cor do novo pixel. É importante frisar que, mesmo sendo um sinal unidimensional, a imagem armazenada no computador deve ser tratada como bidimensional, pois o último pixel de uma linha não possui relação com o primeiro pixel da próxima linha quanto a informações de cores. Portanto, os cálculos devem ser realizados dentro do domínio de cada linha da imagem.

*Splines* são comumente usadas em transformações geométricas que não envolvem alteração nas dimensões da imagem. Elas são empregadas para alcançar resultados de alta qualidade, podendo abordar separadamente as duas dimensões da imagem para reduzir o custo computacional (UNSER, 1999, tradução nossa). Operações de detecção de contornos, que localiza mudanças de intensidade nas cores dos pixels próximos, também podem ser implementadas usando *splines* cúbicas (POGGIO; VOORHEES; YUILLE, 1988, tradução nossa).

Para interpolar um pixel, são utilizados quatro pontos da imagem, sendo dois localizados antes e dois depois do pixel em uma mesma linha. Após este cálculo, consideram-se então os pixels da imagem na posição vertical, tratando os pontos de forma semelhante aqueles localizados na horizontal, para também influenciarem no valor do novo pixel.

### 3.5.2 Interpolação Baseada em *Splines* Cúbicas Aplicada à Animação

*Splines* são úteis no processo de animação, pois podem representar movimentos não lineares, como seqüência de curvas, por meio de funções que permitem adicionar mais pontos e obter valores ao longo de sua extensão.

Funções usadas para definir curvas no computador geralmente são apresentadas em sua representação paramétrica, no formato $P(t) = (x(t), y(t), z(t))$. Neste caso são calculadas três equações para encontrar o valor do ponto $(x,y,z)$, possibilitando então obter vários pontos por meio da variação do parâmetro $t$. Existem outras representações de curvas como a explícita ($y=f(x)$) e a implícita ($f(x,y)=0$) que são pouco usadas devido a necessidade de conhecer a função (SALOMON, 2006, tradução nossa).

Ainda com relação à construção de curvas paramétricas, principalmente na sua forma cúbica, uma representação comumente usada é a matricial, que facilita o cálculo de

derivadas. A fórmula geral (Equação 5.2) é formada pelas matrizes de variáveis *T*, de coeficientes *M* e de pontos da função *B* (PARENT, 2002, tradução nossa).

$$\boxed{P(t) = T^T MB} \quad (5.2)$$

Existem várias formulações baseadas em *splines* cúbicas, além da própria forma natural relatada anteriormente. A variante escolhida é a Catmull-Rom *Spline* ( Figura 7), que possui as seguintes características (PARENT, 2002; KOCHANECK; BARTELS, 1984, tradução nossa; NAMIKAWA, 2001):

a) os pontos de controle fazem parte da curva;

b) possui continuidade $C^1$, com segunda derivada interpolada linearmente;

c) há pontos do segmento fora do intervalo calculado (pontos de controle);

d) a tangente no ponto inicial da curva ($Q_i$) é paralela a secante que conecta os pontos de controle mais próximos ($Q_{i-1}$ e $Q_{i+1}$);

d) pontos exercem controle local na curva;

c) simplicidade nos cálculos.

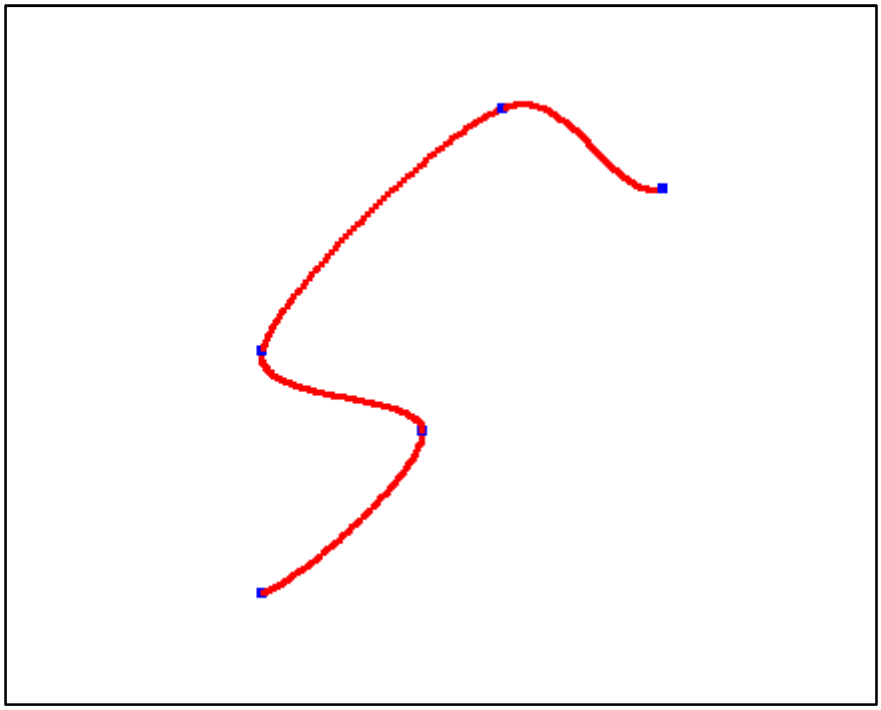

Figura 7. Exemplo de Catmull-Rom *Spline*
Fonte: Adaptado de Lengyel, 2004

Para aplicação na animação, parte-se informando dois quadros-chave $Q_i$ e $Q_{i+1}$, para $i+1$ igual ao número de quadros de referência, e os dois quadros-chave vizinhos $Q_{i-1}$ e $Q_{i+2}$. Variando o parâmetro $t$, conforme a Equação 5.3, normalizado para o intervalo ($0 < t < 1$), determina-se qualquer quadro intermediário $Q(t)$ entre $Q_i$ e $Q_{i+1}$ por meio de uma função cúbica, representada na formulação abaixo, aplicada sobre a Equação 5.2 (NAMIKAWA, 2001).

$$Q(t) = \frac{1}{2}[1, t, t^2, t^3] \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} Q_{i-1} \\ Q_i \\ Q_{i+1} \\ Q_{i+2} \end{bmatrix} \quad (5.3)$$

Usando diretamente a equação acima se interpolam quadros localizados em intervalos que não estão na extremidade inicial ($Q_0$, $Q_1$) e nem no extremo final ($Q_{i-1}$, $Q_i$), visto que para o cálculo, além de conhecer os quadros extremos do segmento, é necessário

informar os quadros imediatamente anterior e posterior. Como não existe quadro antes da extremidade inicial e nem depois da extremidade final, é necessário usar algum artifício. Uma alternativa é atribuir aos quadros do subintervalo inicial o valor do primeiro quadro e do segmento final o valor do último quadro (PARENT, 2002, tradução nossa; NAMIKAWA, 2001). Outra, utilizada no presente trabalho, consiste em gerar dois quadros-chave auxiliares que não fazem parte da animação exibida, somente para darem suporte à interpolação.

A interpolação baseada em *Splines* Cúbicas envolve um compromisso entre a modelagem da representação de curvas arbitrárias em 3D e a complexidade computacional na manipulação da representação dos objetos no processo de animação. A técnica baseia-se na utilização de polinômios de grau três, cada um suportado em um subintervalo, os quais se conectam em suas extremidades. Esses polinômios são obtidos de forma que as derivadas coincidam nas extremidades de subintervalos adjacentes, conduzindo a transições suaves na representação. A técnica de interpolação por *Splines* Cúbicas apresenta reduzida complexidade computacional, embora esta seja superior a da interpolação linear.

## 3.6 INTERPOLAÇÃO COM BASE EM *B-SPLINES*

As *Splines* de Base, ou *Basis Splines (B-splines)* são curvas muito úteis e bastante flexíveis. Diferentemente da definição das *Splines* Cúbicas, as *B-splines* (Figura 8) possuem controle local, tal que alterando um ponto de controle somente modifica a parte da curva que este ponto exerce influência e são caracterizadas por seus pontos de controle não estarem presentes ao longo da trajetória, sendo portando um método de aproximação (LENGYEL, 2004, tradução nossa).

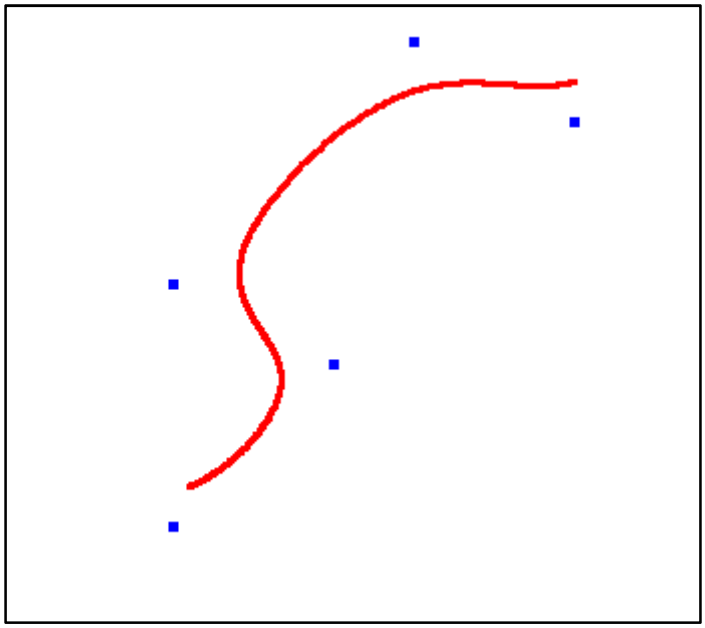

Figura 8. Exemplo de curva usando *B-spline* Cúbica Uniforme
Fonte: Adaptado de Parent, 2002

Dentre as várias formulações de *B-spline,* a que será utilizada neste trabalho é a *B-spline* Cúbica Uniforme. Elas possuem restrições de continuidade de ordem *n-1* ($C^{n-1}$), apresentando, portanto, continuidade $C^2$ em sua forma cúbica. Caracterizada como uniforme porque os pontos possuem espaçamentos iguais para valores de parâmetros iguais (LENGYEL, 2004, tradução nossa). Ela é constituída por vários segmentos de *spline* $P_i(t)$ conectados, onde cada segmento é definido por quatro pontos de controle consecutivos $P_{i-1}$, $P_i$, $P_{i+1}$, $P_{i+2}$. A cada novo ponto de controle adicionado é criado mais um segmento da curva; este controlado pelo novo ponto mais os três últimos pontos. Sendo assim, a curva completa conterá *n* segmentos com *n+3* pontos de controle. Cada segmento é gerado a partir da seguinte fórmula, baseada na Equação 5.2 (SALOMON, 2006, tradução nossa):

$$P(t) = \frac{1}{6}[t^3, t^2, t, 1]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}. \quad (6.1)$$

Existe outra formulação chamada *B-splines* Racionais Não-uniformes (*NonUniform Rational B-Splines* - NURBS), que é um padrão utilizados em aplicações do tipo *Computer Aided Design* (CAD) de modelagem de curvas e superfícies em computação gráfica. Elas apresentam ainda mais flexibilidade que as *b-splines* básicas, sendo usadas geralmente para modelagens de formas detalhadas com várias superfícies curvas, como expressões faciais. Para animação, tipos mais simples de *b-splines,* como as Cúbicas Uniformes são suficientes (HOCH; FLEISCHMANN; GIROD, 1994; PARENT, 2002; ROGERS, 2001, tradução nossa).

### 3.6.1 Interpolação Baseada em *B-splines* Usada em Imagens Digitais

No processamento de imagens, a Equação 6.1 é aplicada sucessivamente nas linhas e colunas. Os nós (termo *B* da Equação 5.2) são os pontos de amostragem da imagem discretizada, que possuem os coeficientes de cor usados no cálculo. O parâmetro *t,* em uma operação de escalamento, por exemplo, dependerá do fator de escala escolhido, pois representa a posição do novo pixel.

Os maiores atrativos para o uso de *b-splines* para a interpolação de imagens é a suavidade alcançada entre os nós, apresentando continuidade $C^2$ e o suporte compacto característico das suas funções de base.

### 3.6.2 Interpolação por *B-splines* Aplicada em Animação

Existem várias formulações para *b-splines.* Dentre elas, geralmente as mais básicas são suficientes para os propósitos da animação. Portanto a *B-spline* Cúbica Uniforme foi escolhida para ser avaliada. Sua formulação é baseada na Equação 6.1, onde os pontos da

função consideram quatro quadros-chave consecutivos. Para obtenção de um quadro pertencente ao intervalo interpolado, utiliza-se o parâmetro *t*, que representa a posição do quadro em função do tempo transcorrido em relação ao ponto inicial do intervalo [$P_i$, $P_{i+1}$]. Da mesma forma que a Catmull-Rom *Spline,* a *B-spline* Cúbica Uniforme necessita de algum artifício para gerar quadros intermediários nos intervalos extremos da seqüência. Na análise será verificada a qualidade com uma solução simples para este problema: geração de dois quadros-chave auxiliares posicionados fora da seqüência (um antes do primeiro quadro e outro depois do último).

O método de interpolação baseado em *b-splines* caracteriza-se por aproximar a representação do sinal, entretanto não garante a passagem nos pontos de referência. Um aspecto importante é a complexidade computacional associada que, uma vez que as funções de base são previamente conhecidas, não é requerido o cálculo dos coeficientes do polinômio interpolador. Isto reduz significativamente a complexidade computacional da interpolação baseada em *B-splines*, aproximando-a da interpolação Linear.

# 4 TRABALHOS CORRELATOS

No ano de 1978, Edwin Catmull publicou um artigo intitulado The *Problems of Computer-Assisted Animation*. Seu trabalho explora as dificuldades encontradas pelos animadores para incluir o computador no processo de criação das seqüências animadas, visando facilitar a técnica convencional, feita a mão. A idéia em torno da possibilidade do artista se preocupar apenas com os quadros-chave e deixar os quadros intermediários serem automaticamente gerados era muito tentadora (CATMULL, 1978, tradução nossa).

Alguns testes foram bem sucedidos, com imagens simples, porém longe da qualidade dos desenhos desenvolvidos na animação tradicional. Catmull (1978) atribuiu este problema ao fato do desenho ser uma projeção 2D de uma cena 3D imaginada pelo animador, que conseqüentemente provoca perdas nesta conversão. Este tipo de percepção é difícil de simular no computador. Ele ainda afirma que este problema é semelhante a questões abordadas pela Inteligência Artificial.

Edwin Catmull destaca, dentre as várias tarefas automatizadas pelo computador no processo de produção da animação, o *inbetweening*, tratando separadamente por julgar um problema bem complexo. Sua abordagem se baseia em apresentar alternativas para que o computador possa simular a imagem 3D em que o animador se baseia para desenhar um quadro em duas dimensões. Ele ainda propõe a análise da viabilidade dos métodos por meio do tempo gasto em todo o processo de animação. Catmull finaliza dizendo que para obtenção de resultados significativos é importante primeiro conhecer os problemas inerentes a esta forma de produção de animações e que o computador é naturalmente uma ferramenta usada para manipulação de imagens. O uso do computador na animação tradicional, apesar das dificuldades, pode proporcionar melhores resultados no trabalho final (CATMULL, 1978, tradução nossa).

Anos mais tarde, outro pesquisador, Geoffrey Slinker, em sua tese de mestrado, com o título Inbetweening Using a Physically Based Model and Nonlinear Path Interpolation, aborda basicamente os problemas de *inbetweening* na criação de animações. Ele inicia comentando algumas técnicas usadas para resolução deste problema e destaca uma técnica desenvolvida por Sederberg e Greenwood que gera excelentes resultados, porém o custo computacional é elevado. Slinker então se baseia nesta técnica para desenvolver seu trabalho (SLINKER, 1992, tradução nossa).

Geoffrey propõe um modelo com menos cálculos matemáticos e a utilização de interpolação não linear. Ele então demonstra os problemas encontrados na interpolação linear para movimentação de corpos rígidos e lista algumas técnicas de interpolação não linear para a resolução do problema. Entre as técnicas, Slinker (1992) enumerou curvas de Bézier, curvas *B-spline,* e ciclóides paramétricas. Após comparação entre as técnicas de interpolação linear e não linear, ele conclui que métodos não lineares de interpolação são mais adequados por não deformarem ou encurtarem os corpos.

Além da comparação, Slinker observa que na animação computacional não basta ter apenas corpos rígidos rotacionando completamente. Estes corpos geralmente são partes de uma estrutura maior, como um personagem. Nesse contexto, ele propõe um modelo interativo para unir corpos rígidos que se movimentam por meio de interpolação não linear, mostrando os resultados. Geoffrey Slinker finaliza seu trabalho concluindo que a animação tradicional é muito complexa e que os modelos de interpolação adotados não são capazes de resolver todos os problemas (SLINKER, 1992, tradução nossa).

Outro trabalho relacionado à animação foi produzido por Adriana dos Santos, em seu trabalho de conclusão de curso, que recebeu o título Protótipo de Software para Geração de Animações por Quadros-chaves Utilizando a Técnica de Interpolação. Nele, Adriana aborda técnicas de interpolação utilizadas na animação por quadros-chave para gerar quadros

intermediários. Ela utiliza três técnicas de interpolação linear: ponto-a-ponto, ponto-a-ponto RGB e de pontos médios. Estas técnicas foram usadas para na construção de um protótipo de software capaz de aplicar a animação conhecida por morfismo, onde uma figura se transforma em outra. Adriana apresenta os resultados obtidos em cada técnica mostrando os quadros intermediários gerados e conclui que o método que obteve melhores resultados foi a interpolação linear ponto-a-ponto RGB (SANTOS, 2000).

# 5 ANÁLISE E COMPARAÇÃO DAS TÉCNICAS DE INTERPOLAÇÃO

A validação da pesquisa realizada neste trabalho está representada nos resultados alcançados por meio de métricas objetivas aceitas na comunidade científica. Adicionalmente, os protótipos desenvolvidos utilizados na realização das medições demonstram visualmente as diferenças entre as técnicas de interpolação comparadas.

O restante do trabalho apresenta o planejamento realizado, demonstrando implementações geradas a partir da pesquisa feita e os resultados que foram alcançados.

## 5.1 METODOLOGIA

Os objetivos estipulados para este trabalho seguiram um planejamento basicamente dividido em três partes: descrição teórica, geração de seqüências animadas e análise comparativa.

A fundamentação teórica abordou a animação desde seus primórdios, com destaque para as relações e contribuições aos trabalhos produzidos atualmente, convergindo para o uso do computador no processo de criação, com referência às diversas partes da animação onde ele é empregado. Uma destas partes recebeu enfoque especial, sendo o objeto principal do presente trabalho: a interpolação de quadros.

Uma vasta pesquisa foi realizada acerca de interpolação, iniciando pela base matemática, em busca do entendimento do assunto. Este conceito foi descrito em três técnicas de interpolação: Linear, com base em *Splines* Cúbicas e em *B-splines*. Em seguida foram descritas as aplicações destes métodos em imagens digitais, pois são intrinsecamente ligadas à animação. Todo o estudo culminou na compreensão da interpolação no campo da animação, conceituando sua aplicação e definindo as técnicas a serem analisadas, representando os

grupos citados anteriormente: Linear, Catmull-Rom *Spline* e *B-spline* Cúbica Uniforme, respectivamente.

O referencial bibliográfico também abordou trabalhos que apresentam alguma relação com o assunto pesquisado.

A segunda parte do trabalho foi a geração de seqüências animadas, demonstrando a aplicação das técnicas comparadas. Foram descritas as ferramentas principais usadas na implementação das animações, com destaque para Java e Open Graphics Library (OpenGL). Estas tecnologias foram aplicadas em conjunto com toda a teoria pesquisada acerca de interpolação em animação.

A última parte descreveu as ferramentas usadas para a obtenção dos resultados, colhidos por meio de análises comparativas fornecidas pelas métricas objetivas PSNR e MSE. Estas ferramentas foram aplicadas sobre as animações geradas, produzindo subsídios para a apresentação dos resultados obtidos e conclusão do trabalho.

## 5.2 AMBIENTE DE DESENVOLVIMENTO

A parte prática do trabalho se iniciou com a preparação do ambiente de desenvolvimento. As tecnologias e ferramentas usadas foram escolhidas principalmente por serem livres, contendo outras características também relevantes para a pesquisa.

### 5.2.1 Ferramentas Utilizadas na Produção das Animações

As comparações entre os três métodos de interpolação até aqui conceituados necessitam do desenvolvimento de algumas seqüências animadas para visualização e coleta de dados. As tecnologias envolvidas neste processo foram principalmente Java e OpenGL. Além

do interesse do autor, outras características pesaram nesta escolha entre elas (DAVISON, 2007; WRIGHT; LIPCHAK; HAEMEL, 2007, tradução nossa):

a) são tecnologias livres;

b) são multiplataforma;

c) são robustas;

d) apresentam extensa bibliografia.

Os protótipos foram gerados em ambiente Microsoft Windows, porém, beneficiando-se das características apresentadas, podem ser visualizadas em outros sistemas. A seguir há mais detalhes sobre as duas tecnologias.

#### 5.2.1.1 OpenGL

A plataforma OpenGL é descrita como uma interface de software que atua sobre o hardware gráfico. É considerada um padrão na indústria gráfica. Criada pela Silicon Graphics (SGI) em 1992 foi mantida pelo *Architecture Review Board* (ARB) até 2006, quando a falida SGI entregou o controle da tecnologia OpenGL para um grupo chamado The Khronos Group que é um consórcio independente composto por mais de cem empresas, entre elas, AMD, Dell, IBM, Intel, Matrox, NVIDIA e Sun (WRIGHT; LIPCHAK; HAEMEL, 2007, tradução nossa).

A *Application Programming Interface* (API) OpenGL controla somente aspectos gráficos da aplicação por meio de comandos que acessam diretamente o hardware. Ela contém rotinas gráficas e de modelagem, bidimensional e tridimensional, extremamente portável e rápida, permitindo a geração de imagens com alta qualidade (AZEVEDO; CONCI, 2003). Desta forma, o código torna-se portável entre os diversos sistemas suportados.

Cada sistema possui bibliotecas próprias que agregam funcionalidades acessórias como, por exemplo, gerenciamento de janelas e rotinas de entrada e saída de dados. Destacam-se a OpenGL *Utility Toolkit* (GLUT) para várias plataformas, Windows OpenGL (WGL) para Microsoft Windows, OpenGL *for* X Window System (GLX) para sistemas baseados em Unix e Apple *Graphics Library* (AGL) para Mac OS X (WHITROW, 2008; WRIGHT; LIPCHAK; HAEMEL, 2007, tradução nossa).

Internamente a OpenGL trabalha como uma máquina de estados. Estados como cor e tipo de primitiva geométrica podem assumir diversos valores. Há também variáveis de estado que regem, por exemplo, efeitos de luz e neblina que podem ser ligados ou desligados por comandos do tipo *glEnable()* ou *glDisable().* A forma de utilização é muito simples. Os estados determinam como o desenho deve ser apresentado. Comandos alteram determinado estado que conseqüentemente modificam a forma que os próximos desenhos serão feitos. Por meio de comandos com *glGet* é possível obter o valor de uma variável de estado (SHREINER et al, 2006, tradução nossa).

Para o caso específico de animação, a OpenGL dispõe do recurso de *buffer* duplo. Antes de serem apresentadas, as imagens são transferidas para uma memória especial conhecida como *framebuffer*, que contém as informações necessárias para o processador exibir a imagem no monitor. Usando apenas um *framebuffer,* imagens complexas podem demorar a ser processadas e por conseqüência serem mostradas por partes. Com dois, as imagens serão apresentadas somente após estarem prontas. Enquanto o quadro referente ao um *framebuffer* está sendo exibido, o próximo é armazenado em um segundo *framebuffer*. Então os *framebuffers* são trocados, exibindo na tela a imagem do segundo enquanto o primeiro recebe o próximo quadro. É uma troca constante entre eles (WRIGHT; LIPCHAK; HAEMEL, 2007, tradução nossa).

A animação é apenas uma das várias áreas da computação gráfica que a API OpenGL oferece suporte. Ferramentas CAD, imagens médicas, visualização científica, realidade virtual e pacotes gráficos de modelagem usados para criar efeitos especiais em filmes como *Jurassic Park* e *Star Wars* são alguns exemplos (AZEVEDO; CONCI, 2003).

Além das funcionalidades presentes na especificação padrão, mais recursos podem ser oferecidos pela OpenGL por meio de extensões, onde fabricantes adicionam funcionalidades específicas. Para que isto seja feito não é necessário que a extensão seja aprovada pelo consórcio. Este módulo não altera o conjunto de operações padrão da API, pois a OpenGL é uma tecnologia livre, porém o seu código não é aberto. Para que novos recursos sejam usados, o fabricante disponibiliza um novo *driver* e o hardware compatível, além de software para os desenvolvedores explorarem as novas extensões. É uma forma dinâmica e rápida de adequação a evoluções na área. Algumas funcionalidades interessantes incluídas por este módulo de extensão chegam a ser agregadas a especificação principal da API, após aprovação pelos membros do consórcio (MARTZ, 2006; WRIGHT; LIPCHAK; HAEMEL, 2007, tradução nossa).

#### 5.2.1.2 Java

Java é uma linguagem de programação orientada a objetos baseada nas linguagens C e C++. Surgiu a partir de um projeto interno da Sun Microsystems de codinome Green, voltado para dispositivos eletrônicos inteligentes (DEITEL; DEITEL, 2001).

O crescimento do mercado destes dispositivos estava abaixo do esperado. Então a Sun decidiu mudar o foco do projeto, aproveitando a popularização do *World Wide Web* (WWW) (DEITEL; DEITEL, 2001).

Direcionado para aplicações w*eb*, a linguagem Java chamou a atenção da comunidade comercial. Hoje ela está sendo constantemente atualizada e abrange outros tipos de aplicação além da *web*, como aplicativos para *desktop,* dispositivos móveis e sistemas embarcados (DEITEL; DEITEL, 2001).

As aplicações Java são totalmente independentes de plataforma. O código criado em um sistema pode ser executado em qualquer outro que ofereça suporte a linguagem sem nenhuma modificação, precisando apenas ter a máquina virtual instalada. Java é portável tanto no contexto de software quanto de hardware. Esta característica é possível graças aos *bytecodes,* que são várias instruções interpretadas pela máquina virtual Java criadas a partir do código-fonte do programa e são independentes do sistema onde foi gerado. Por questões de segurança, a máquina virtual verifica os *bytecodes* antes de executá-los (ARNOLD; GOSLING; HOLMES, 2007).

Com relação ao OpenGL, a linguagem Java possui algumas bibliotecas que permitem a sua utilização. Destaca-se a Java OpenGL, conhecida como JOGL, que faz parte de um projeto da Sun em seu Game Technology Group sendo a implementação oficial de referência para ligação entre Java e OpenGL, de acordo com a Java *Specification Requests* (JSR) 231 (JSR-000231, 2006, tradução nossa).

#### 5.2.1.3 A Biblioteca JOGL

Definida na JSR 231, a JOGL é a especificação padrão utilizada pela Sun para ligar a linguagem Java à API OpenGL. Este é um dos motivos pelo qual ela se destaca em meio a outras bibliotecas que são usadas para a mesma finalidade. Apresenta também como características o acesso às interfaces gráficas AWT e Swing, controle de baixo nível,

acessando a OpenGL por meio de chamadas diretas aos seus comandos, além das funcionalidades de alto nível herdadas da própria linguagem Java.

A biblioteca JOGL é recente e sofreu algumas mudanças desde sua criação, principalmente após ser adotada pela Sun. Sua documentação está em desenvolvimento crescente, com exemplos adaptados de livros e sites famosos que abordam OpenGL com as linguagens C e C++. Além disso, programadores que utilizam a JOGL se beneficiam da vasta documentação sobre OpenGL existente, pois as chamadas aos comandos são muito semelhantes as implementações originais em C e C++.

A instalação da JOGL é muito simples, especialmente com o ambiente integrado de desenvolvimento (ou *integrated development environment* - IDE) Netbeans, cuja instalação pode ser feita por meio de um pacote de *plugins* preparado especialmente para ele. O pacote está disponível no site oficial da biblioteca JOGL, incluindo vários exemplos práticos bastante úteis e que revelam um pouco do que pode ser desenvolvido.

Qualquer aplicação desenvolvida com JOGL contém duas partes principais. A primeira é a criação da janela em si, que exibirá os gráficos gerados. A outra é a parte de tratamento de eventos, cuja origem deve ser ao menos do OpenGL.

É sabido que a JOGL trabalha tanto com AWT quanto SWING para o sistema de janelas. A forma de utilização entre elas apresenta pequenas diferenças. Neste trabalho foi utilizado apenas o sistema AWT, seguindo abaixo a apresentação da sua estrutura básica.

Cria-se uma janela que contém um objeto *GLCanvas*, responsável por tratar os eventos originados do OpenGL. Para isso, o *GLCanvas* precisa conter a instância de uma classe que implemente a interface *GLEventListener* nos métodos *init(), display(), reshape()* e *displayChanged()*. Cada método realiza as funções descritas a seguir:

a) no método *init()* são inicializados alguns estados da máquina de estados OpenGL, como por exemplo a cor usada para limpar a tela, a iluminação e

também variáveis e configurações fora do OpenGL que geralmente seguem inalteradas durante a execução da aplicação, como carregamento de arquivos;

b) em display() é onde os gráficos são apresentados na tela. Todas as funções que desenham estão neste método. Nele também podem ser realizadas animações, principalmente se a aplicação tenha configurado um objeto da classe Animator;

c) o redimensionamento da janela é monitorado pelo método *reshape()*. O plano de projeção também é configurado nele, pois este método é chamado uma vez na inicialização da aplicação;

d) por fim há o método *displayChanged()*. Trata de mudanças ligadas ao contexto gráfico oriundas do sistema operacional, como por exemplo, a alteração da resolução da tela. A implementação de referência não suporta este método, por isso não é obrigatório implementá-lo.

A estrutura básica de uma aplicação JOGL está ilustrada na Figura 9.

```
27  public class SkeletonAnimation implements GLEventListener {
28      Map<Integer, float[]> keyframes = new HashMap<Integer, float[]>();
29      int iAnim = 0; //controla o loop da animacao
30      int iInter = -1; //controla a geracao de quadros intermediarios
31      float[] qAtual = new float[9];
32
33      public static void main(String[] args) {
34          Frame frame = new Frame("Skeleton Animation");
35          GLCanvas canvas = new GLCanvas();
36
37          canvas.addGLEventListener(new SkeletonAnimation());
38          frame.add(canvas);
39          frame.setSize(640, 480);
40          final Animator animator = new FPSAnimator(canvas, 60, false);
41          animator.setRunAsFastAsPossible(true);
42          frame.addWindowListener(new WindowAdapter() {...});
58          // Center frame
59          frame.setLocationRelativeTo(null);
60          frame.setVisible(true);
61          //canvas.setAutoSwapBufferMode(true);
62          animator.start();
63      }
64
      public void init(GLAutoDrawable drawable) {...}
104
      public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {...}
122
      public void display(GLAutoDrawable drawable) {...}
172
      public void displayChanged(GLAutoDrawable drawable, boolean modeChanged, boolean deviceChanged) {...}
175
```

Figura 9. Estrutura da aplicação na JOGL

A classe *Animator* possibilita a geração de animações em tempo real. É a forma de animação usada neste trabalho. Seu funcionamento se baseia em criar uma instância da classe indicando um contexto do OpenGL (objeto *GLCanvas*). No caso ilustrado na Figura 9, foi criada uma instância da classe *FPSAnimator*, que estende a classe *Animator* provendo maior controle sobre a velocidade da aplicação. Após a inicialização do objeto *GLCanvas,* o método *start()* da classe *FPSAnimator* permanece redesenhando tudo por meio de chamadas ao método *display()* até que a aplicação seja finalizada.

## 5.3 ANIMANDO UM JIPE

O conhecimento adquirido acerca da biblioteca JOGL possibilitou a criação dos primeiros protótipos utilizando efetivamente a interpolação para gerar seqüências animadas. Estes protótipos inicialmente animavam imagens estáticas, ligeiramente diferentes entre elas, que foram carregadas seqüencialmente ordenadas possibilitando a aplicação das técnicas de interpolação.

Os primeiros exemplos utilizavam os pixels dos quadros-chave pré-carregados como base de cálculo. Os resultados obtidos não foram satisfatórios, visto que seria necessário algum artifício que permitisse controlar o deslocamento entre os pixels, e não somente a mudança de cores. Decidiu-se então direcionar os esforços para uma abordagem diferente: modelos geométricos. Animações simples, como deslocamento de formas geométricas, demonstraram a funcionalidade desta forma de animação, encorajando a continuação.

Passada esta etapa e constatado que o ambiente estava pronto para o desenvolvimento de seqüências mais interessantes, houve a necessidade de incluir no projeto um modelo pronto que oferecesse liberdade suficiente para permitir ser animado. Isto porque

seria inviável modelar um objeto diretamente via programação, tanto é que existem softwares de modelagem específicos para isto, que inclusive utilizam a API OpenGL.

A maior dificuldade para a inclusão do modelo na aplicação foi encontrar um algoritmo que fizesse a conversão do arquivo original para o formato da JOGL. Existem *Loaders*, como são chamados estes algoritmos, para diversos formatos de modelos 3D, porém em sua maioria destinados a implementações em C ou C++ do OpenGL. Felizmente um destes *Loaders* foi convertido de uma biblioteca em C++ para a JOGL e funcionou de forma satisfatória. A solução original foi desenvolvida por Porter (2000) e convertida para o Java por Ougaard (2006). O formato adotado neste algoritmo foi o *.ms3d*, gerado pelo software MilkShape 3D.

A seguir foi necessário encontrar um modelo que facilitasse a comparação entre as técnicas de interpolação em análise. O modelo escolhido foi um jipe ( Figura 10), criado por Psionic (2009). Este modelo possui as características abaixo relacionadas:

a) 2032 polígonos;

b) 6096 vértices;

c) utilização de textura;

d) composto por sete partes.

Figura 10. Modelo 3D de um jipe
Fonte: Adaptado de Psionic, 2009

Os primeiros testes animados utilizando o jipe foram feitos sem nenhuma técnica de interpolação. A seqüência foi realizada por meio de operações de translação e rotação disponíveis na OpenGL. Rotação para as rodas e translação para todo o modelo. A estrutura do jipe ajudou a completar esta etapa, porque ele foi modelado em sete partes: uma cada roda, os dois volantes e o restante do veículo. Então se utilizou um fator de rotação para todas as rodas e aplicou-se a translação em todo o modelo (inclusive as rodas), resultando em uma animação onde o jipe se movimentava para frente. O resultado alcançado esteve dentro do esperado. Era o momento de utilizar a interpolação linear.

## 5.4 INTERPOLAÇÃO LINEAR

Aparentemente esta etapa foi simples, pois o Java permite manipular os pontos da imagem individualmente. Desta forma, tendo dois quadros consecutivos é possível criar *n* quadros intermediários se valendo da interpolação linear.

### 5.4.1 Interpolação em 2D

O método se baseia no cálculo dos pixels que estiverem na mesma posição, varrendo toda a imagem para gerar cada quadro intermediário. Os quadros que estiverem mais próximos de uma das extremidades sofrem mais influência desta. Isto teoricamente garante a esperada transição suave entre os quadros chave da animação, porém os primeiros resultados obtidos não foram satisfatórios.

A interpolação utilizando a correspondência direta entre os pixels das imagens de destino e origem não resolveram o problema. É necessária a utilização de uma técnica que realize efetivamente o deslocamento dos pixels. Para isso é preciso relacionar pontos entre as imagens base que não estão na mesma posição, ou até mesmo criar novos pixels.

Depois de algumas tentativas sem sucesso, decidiu-se então mudar a abordagem do sistema proposto, sem sair, logicamente, do escopo do trabalho. A mudança recaiu principalmente sobre a origem dos quadros-chave, onde a interpolação foi realizada sobre vértices e não mais sobre pixels. Em outras palavras, foram utilizados modelos 3D ao invés de imagens estáticas para a base de cálculo. Esta abordagem, na animação computacional, chama-se Animação Gerada por Computador, e já foi explicada no Capítulo 2.

### 5.4.2 Interpolação em 3D

As técnicas de interpolação necessitam de quadros de referência como base de cálculo para geração dos quadros intermediários. Houve então a necessidade de gerar estes quadros-chave, que por ter a análise comparativa como objetivo principal, foram os mesmos utilizados por todas as técnicas abordadas no presente trabalho.

Na inicialização da aplicação são carregados todos os modelos (cada quadro-chave caracteriza um modelo, pois é o jipe em várias posições distintas) e dispostos de forma seqüencial. O cálculo do quadro intermediário ocorre dinamicamente, utilizando pares consecutivos dos quadros-chave pré-carregados. Isto é possível porque a equação da interpolação linear (Equação 3.4) está na forma paramétrica, cujo quadro interpolado varia em função do valor do parâmetro *t* que for passado, sendo $t = [0,1]$.

O parâmetro *t* representa a localização do quadro em um dado intervalo. Seu valor é determinado pela Equação 5.1, para $q_i$ igual ao quadro a ser calculado, $tq_i$ referente ao total de quadros interpolados no intervalo e $0 < q_i \leq tq_i$. Esse parâmetro indica o deslocamento no tempo e espaço partindo do ponto inicial, que é o termo $Q_0$ da Equação 3.4.

$$\boxed{t = q_i/(tq_i + 1)} \quad (5.1)$$

Após calcular o valor de *t*, aplica-se então a interpolação. A Figura 11 revela a fórmula utilizada na aplicação da interpolação Linear. A variável *IcurrentFrame* estabelece o parâmetro *t*. Os quadros $Q_0$ e $Q_1$ estão representados nas variáveis *start* e *end*, respectivamente. A interpolação é diretamente aplicada sobre os três eixos (x, y, z) de cada vértice do modelo atualizando o próximo quadro a ser exibido na tela, representado por *currentModel.* Desta forma não são usados recursos para armazenar as amostras interpoladas.

```
private void linearInterpolation(float IcurrentFrame,
                                 MS3DModel start, MS3DModel end) {

    for (int i=0; i<start.vertices.length; i++)
        for (int j=0; j<3; j++) {
            currentModel.vertices[i].location[j] = (end.vertices[i].location[j] -
                                                    start.vertices[i].location[j]) *
                                                    IcurrentFrame + start.vertices[i].location[j];
        }
}
```

Figura 11. Implementação da interpolação Linear

A Figura 12 mostra um quadro interpolado (direita) e um quadro de referência (esquerda) na mesma posição do quadro interpolado, obtido no software Milkshape 3D.

Figura 12. Interpolação Linear

## 5.5 INTERPOLAÇÃO BASEADA NO ALGORITMO CATMULL-ROM *SPLINE*

O segundo método de interpolação pesquisado foi escolhido com a promessa de fornecer melhores resultados que a interpolação Linear em termos de qualidade visual. Sua origem está apoiada sobre o conceito de *Spline* Cúbica, cujo objetivo principal é a construção de curvas ou trajetórias sendo formada por partes continuamente conectadas.

Uma característica interessante observada na Catmull-Rom *Spline* é o fato dos pontos de controle pertencerem a curva. Isto faz com que a trajetória construída para o movimento tenda para os quadros-chave, que são quadros de referência, portanto, desejáveis na animação.

O cálculo dos quadros interpolados por este algoritmo, sendo *spline* cúbica, necessita a indicação de quatro quadros-chave para interpolar pontos entre dois quadros consecutivos. Estes quadros, previamente carregados ordenadamente, percorrem a seqüência com deslocamento lateral, agrupados de quatro em quatro ($(Q_0, Q_1, Q_2, Q_3), (Q_1, Q_2, Q_3, Q_4),..., (Q_{n-2}, Q_{n-1}, Q_n, Q_{n+1})$, com $n$ representando o número de quadros-chave da seqüência e $Q_0$ e $Q_{n+1}$ os quadros auxiliares), gerando os quadros intermediários e conseqüentemente a

trajetória do movimento. Os quadros interpolados nos intervalos das extremidades foram calculados utilizando dois quadros-chave auxiliares para manter a qualidade do movimento.

Assim como a interpolação Linear, é possível gerar qualquer quadro intermediário dentro de um intervalo, especificando um valor para o parâmetro tempo. O cálculo deste foi realizado pela Equação 5.1.

A interpolação realizada é apresentada na Figura 13. Indicando quatro quadros-chave (vetor *keypoints*) mais o parâmetro tempo (variável *q*) é suficiente para a realização da interpolação. Os termos *q2* e *q3* representam o referido parâmetro elevado a segunda e terceira potências respectivamente. As variáveis *b1, b2, b3* e *b4* formam o cálculo do filtro base. O modelo *currentModel* comporta todos os vértices do quadro que está sendo interpolado para serem exibidos na tela. Este é mostrado logo após o término do cálculo, tanto que nenhum quadro interpolado fica retido na memória para uso posterior, economizando recursos da máquina.

```
private void catmullRomSplineInterpolation(float IcurrentFrame, MS3DModel[] keypoints) {

    float b1, b2, b3, b4;

    float q = IcurrentFrame;
    float q2 = (float)Math.pow(q, 2);
    float q3 = (float)Math.pow(q, 3);

    // funcao base
    b1 = -q3 + 2f*q2 - q;
    b2 = 3f*q3 - 5f*q2 + 2f;
    b3 = -3f*q3 + 4f*q2 + q;
    b4 = q3 - q2;

    for (int i=0; i<keypoints[0].vertices.length; i++)
        for (int j=0; j<3; j++) {
            currentModel.vertices[i].location[j] = (keypoints[0].vertices[i].location[j] * b1+
                                                    keypoints[1].vertices[i].location[j] * b2 +
                                                    keypoints[2].vertices[i].location[j] * b3+
                                                    keypoints[3].vertices[i].location[j] * b4) * 0.5f;
        }
}
```

Figura 13. Implementação da interpolação com base na Catmull-Rom *Spline*

Um exemplo de quadro interpolado por esta técnica é apresentado na Figura 14. O quadro da esquerda é o de referência e o quadro da direita é a amostra interpolada.

Figura 14. Interpolação baseada em Catmull-Rom *Spline*

## 5.6 INTERPOLAÇÃO COM BASE EM *B-SPLINE* CÚBICA UNIFORME

O método de interpolação utilizando *b-spline* também foi escolhido com o objetivo de obter bons resultados em se tratando da qualidade final da animação. Esta opção se revela por suas características, como a suavidade entre segmentos proporcionada pela continuidade de ordem dois e o controle local, presentes até mesmo nas formas mais simples, que é o caso da equação que foi usada neste trabalho, chamada *B-spline* Cúbica Uniforme.

Há outra propriedade deste tipo de curva que merece atenção. Os pontos de controle apenas se aproximam da trajetória, não fazendo necessariamente parte dela. Por esse motivo se diz que a curva é construída por aproximação, e não por interpolação. É importante relembrar que o conceito de interpolação tratado até aqui é diferente desse, pois se refere à geração automatizada de quadros intermediários, sem se preocupar com a forma de atuação dos quadros de controle. Até mesmo na produção destes quadros com *b-spline* utiliza-se interpolação, apesar da curva em sua essência ser por aproximação.

A implementação da interpolação de quadros por *B-spline* Cúbica Uniforme é praticamente a mesma da Catmull-Rom *Spline,* com alterações na função de base e no cálculo final dos vértices. A estrutura de variáveis é a mesma apresentada no método anterior. Especificam-se quatro quadros-chave consecutivos para gerar quadros intermediários em um intervalo. Deslocando-se para o próximo quadro-chave da seqüência, deixando para trás o primeiro, criam-se mais quadros suavemente conectados com o intervalo anterior, seguindo

desta forma até o penúltimo intervalo. Para a interpolação de quadros no primeiro e último intervalos, também foram criados dois quadros-chave auxiliares, um em cada extremidade.

A criação dos quadros intermediários, a exemplo das demais técnicas analisadas, foi controlada por meio do parâmetro tempo, cujo cálculo foi realizado pela Equação 5.1. A implementação completa da *B-spline* Cúbica Uniforme foi codificada conforme a Figura 15. Um exemplo de quadro interpolado (quadro da direita) é apresentado na Figura 16.

```
private void b_splineInterpolation(float IcurrentFrame, MS3DModel[] keypoints) {

    float b1, b2, b3, b4;

    float q = IcurrentFrame;
    float q2 = (float)Math.pow(q, 2);
    float q3 = (float)Math.pow(q, 3);

    // funcao base
    b1 = -q3 + 3f*q2 - 3f*q +1;
    b2 = 3f*q3 - 6f*q2 + 4f;
    b3 = -3f*q3 + 3f*q2 + 3f*q + 1;
    b4 = q3;

    for (int i=0; i<keypoints[0].vertices.length; i++)
        for (int j=0; j<3; j++) {
            currentModel.vertices[i].location[j] = (keypoints[0].vertices[i].location[j] * b1+
                                                    keypoints[1].vertices[i].location[j] * b2 +
                                                    keypoints[2].vertices[i].location[j] * b3+
                                                    keypoints[3].vertices[i].location[j] * b4) / 6f;
        }
}
```

Figura 15. Implementação da interpolação utilizando *B-spline* Cúbica Uniforme

Figura 16. Interpolação com base em *B-spline* Cúbica Uniforme

## 5.7 COMPARAÇÃO ENTRE AS TÉCNICAS DE INTERPOLAÇÃO

O objetivo principal deste trabalho é a comparação entre métodos de interpolação, no contexto de animação, de forma objetiva. Por esta característica foi necessário produzir dados provenientes da realização de medições por meio de ferramentas específicas. Foram utilizadas medidas objetivas sobre um ambiente preparado especificamente para a análise, com a mesma situação montada para todas as técnicas.

### 5.7.1 Métricas Objetivas

Em tarefas envolvendo processamento de imagens é essencialmente importante obter qualidade nos resultados. Uma forma de analisar a qualidade de uma imagem ou uma seqüência delas é por meio da opinião de várias pessoas. Este método tem a desvantagem de ser subjetiva, tal que cada pessoa possui uma percepção diferente da mesma imagem.

Existem outras formas de quantificar a qualidade resultante de um processo envolvendo imagens, baseada em medidas objetivas, relacionando as perdas resultantes do processamento das imagens. A mais usada é a Relação Sinal-Ruído de Pico (P*eak Signal to Noise Ratio* – PSNR), que será a métrica utilizada para avaliação das seqüências animadas geradas neste trabalho. Esta métrica é popular na comunidade científica por ser de compreensão simples e facilmente computável (WINKLER; MOHANDAS, 2008). Sua fórmula é apresentada na Equação 5.2 (NEMETHOVA et al, 2006).

$$PSNR = 10 * \log_{10}\left(\frac{255^2}{MSE}\right) \quad (5.2)$$

*Mean Squared Error* (MSE) é o erro médio quadrático (Equação 5.3) aplicado na comparação entre o quadro de referência *I(m,n)* e a imagem interpolada *J(m.n)* (NEMETHOVA et al, 2006).

$$MSE = \frac{1}{M * N} * \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [I(m,n) - J(m.n)]^2 \quad (5.3)$$

PSNR mede o ruído resultante do processamento em decibéis (dB). Valores acima de 30 dB são considerados aceitáveis, sendo que valores maiores representam melhores resultados. O valor 255 da Equação 5.2 representa o valor máximo do sinal, para o caso de 8 bits.

E para calcular a PSNR total da animação basta realizar a média simples entre todos os quadros envolvidos na análise.

### 5.7.2 Situação Analisada

O ambiente preparado para a coleta dos dados comparativos apresentou as seguintes características:

a) três quadros chave igualmente espaçados quanto à posição no tempo;

b) três quadros intermediários a cada intervalo entre dois quadros-chave;

c) total de quadros da seqüência: nove;

d) resolução espacial de 784 x 364 pixels.

Os quadros foram gerados a partir do posicionamento do modelo do jipe. A criação destes quadros de referência foi feita, por falta de solução *open source*, no software

Milkshape 3D, donde se origina o formato do modelo. Este software possui licença *shareware* e permite ser utilizado por trinta dias.

Criaram-se os quadros de referência com as especificações abaixo, sendo aplicadas as transformações geométricas sobre o posicionamento anterior:

a) rotação de trinta graus nas rodas em torno do eixo x;

b) translação de dois pixels em todo o modelo no eixo z.

Os dados obtidos estão relacionados na Tabela 1, cuja origem advém da aplicação da PSNR sobre cada quadro interpolado possibilitando a média simples entre os resultados. O valor obtido na média é representativo da PSNR para cada método de interpolação, com valores maiores indicando melhores resultados. Os quadros de referência de cada amostra interpolada foram gerados da mesma maneira que os quadros-chave, garantindo a comparação com o quadro de melhor qualidade possível.

Tabela 1. Resultados da aplicação da PSNR

| Quadro | Interpolação (Valores em dB) | | |
|---|---|---|---|
| | Linear | Catmull-Rom *spline* | *B-spline* Cúbica Uniforme |
| 1 | 36,5557 | 54,3896 | 34,6266 |
| 2 | 35,3799 | 52,9648 | 34,6247 |
| 3 | 36,8967 | 55,1563 | 34,7564 |
| 4 | 36,9633 | 55,0251 | 34,6427 |
| 5 | 35,5383 | 52,4994 | 34,6608 |
| 6 | 36,9807 | 55,4495 | 34,7665 |
| Média | 36,3858 | 54,2475 | 34,6796 |

Os dados apresentados nas Tabela 1 foram gerados por meio dos algoritmos ilustrados na Figura 17 e na Figura 18.

```
public static double PSNR (FloatBuffer reference, FloatBuffer compared, int width, int height) {
    double mse = MSE(reference, compared, width, height);

    if (mse == 0d)
        return 0d;
    else
        return 10d*Math.log10(1d/mse);
}
```

Figura 17. Implementação da PSNR

A Figura 17 revela como foi implementada a métrica PSNR. Os vetores *reference* e *compared* representam os quadros a serem analisados, onde o primeiro armazena os pixels do quadro de referência, ou quadro ideal e o segundo possui os pixels do quadro interpolado. *width e height* são a largura e altura do quadro, respectivamente.

A métrica objetiva PSNR realiza comparações considerando os pixels das imagens. Desta forma foi necessário converter o jipe em imagem. Este processo já é naturalmente realizado para que o modelo possa ser visualizado no monitor de vídeo. A API OpenGL possibilita a recuperação destes dados. Antes de enviar a imagem para a placa de vídeo, ela armazena as informações de cores de cada pixel que será exibido numa memória chamada *colorbuffer*, permitindo a leitura. Estas informações podem então ser guardadas em um vetor de dados e posteriormente manipuladas. Tal recurso foi usado para calcular o MSE (Figura 18), necessário para encontrar a PSNR.

```
private static double MSE (FloatBuffer reference, FloatBuffer compared, int width, int height) {
    double squaredError = 0d;

    do {
        squaredError += Math.pow(reference.get() - compared.get(), 2);
    }while(reference.remaining()>0);

    return  squaredError/(double)(width*height*3);
}
```

Figura 18. Implementação do MSE

Analisando a Tabela 1, os melhores resultados foram alcançados pela interpolação utilizando a Catmull-Rom *Spline*. Inesperadamente a segunda melhor técnica foi a interpolação Linear, superando a interpolação baseada em *B-spline* Cúbica Uniforme, que apresentou os piores resultados, ainda que por uma diferença bem pequena se comparada a segunda colocada.

O desempenho baixo da *B-spline* Cúbica Uniforme é explicado pelo tipo de animação onde a técnica foi aplicada. Por se tratar de interpolação *b-spline*, esta aproxima as imagens nos quadros-chave, entretanto não garante a passagem por estes quadros, promovendo portanto uma degradação do sinal. Contudo, subjetivamente, constatou-se uma transição entre quadros suave, tornando-se mais agradável a percepção visual.

Os resultados obtidos refletiram as características pesquisadas para cada um dos métodos. A interpolação Linear apresentou baixo desempenho, porque não é uma técnica apropriada para trajetórias curvas, como a rotação das rodas. O interpolador *spline* cúbico em Catmull-Rom *Spline* apresentou melhor comportamento, pois é essencialmente *spline*, próprio para construção de curvas. O seu diferencial em relação a *B-spline* Cúbica Uniforme é o fato dos pontos de controle, entendidos por quadros-chave, estarem presentes na trajetória. Como resultado prático a aplicação do método baseado em *b-spline* culminou na deformação da roda, reduzindo seu tamanho.

Os quadros-chave, na interpolação linear, também são parte da trajetória, porém esta característica faz com que os quadros interpolados mais próximos dos extremos do intervalo apresentem melhores resultados que os quadros localizados perto do centro, pois a trajetória é retilínea. Isso explica a variação da PSNR dentro do mesmo intervalo, fato este que ocorreu também com a Catmull-Rom *Spline*, evidenciando a suavidade superior alcançada pela *B-spline* Cúbica Uniforme ($C^2$).

As animações produzidas com as três técnicas analisadas auxiliaram na visualização dos resultados. A animação com Interpolação Linear apresentou os piores resultados quanto à qualidade visual. A roda deformou-se a cada novo quadro interpolado. Na seqüência utilizando a interpolação com Catmull-Rom *Spline* as deformações foram imperceptíveis. O movimento alcançado pela interpolação por *B-spline* Cúbica Uniforme foi muito bom, porém foi possível perceber que o pneu do jipe teve seu raio reduzido, caracterizando uma deformação.

# CONCLUSÃO

A animação é uma área que vem ganhando atenção cada vez maior com o emprego do computador. Diversas partes da produção das animações vêm passando por inovações. Os constantes avanços da tecnologia também têm sido importantes para o acréscimo de qualidade das técnicas envolvidas nas animações recentes.

Outro aspecto importante da relação computador e animação é o aumento da rapidez no processo de criação das seqüências animadas. A partir da definição de alguns quadros (quadros-chave) geram-se vários outros de forma automatizada (quadros intermediários), reduzindo esta trabalhosa e não raramente entediante tarefa.

A geração automática de quadros da animação não é uma tarefa trivial, tanto que há diversas formas de realizá-la, gerando resultados visualmente diferentes. Durante o trabalho foram abordadas técnicas baseadas em interpolação que podem ser aplicadas em situações apropriadas para obtenção dos resultados de animação.

A compreensão acerca das soluções pesquisadas exigiu conhecimentos sobre os princípios da animação juntamente com a matemática envolvida na interpolação. Essas definições foram imprescindíveis para entender como utilizar as técnicas de interpolação linear, baseada em *splines* e *b-splines* no contexto de animação, pois possuem aplicação em diversas áreas.

As técnicas de interpolação foram implementadas e os resultados foram comparados. Na análise apresentada, o interpolador que utiliza o algoritmo de Catmull-Rom *Spline* apresentou o melhor desempenho nos aspecto de qualidade resultante, bem superior a da interpolação Linear e da interpolação por *B-spline* Cúbica Uniforme. As duas últimas obtiveram resultados muito próximos.

A visualização das animações implementadas utilizando as três técnicas revelaram visualmente que a interpolação Linear alcançou resultados bem inferiores aos métodos baseados em *spline* e *b-spline*, ratificando o comportamento previsto. Esta diferença entre a medição objetiva e a avaliação subjetiva demonstra que a ferramenta PSNR não pode ser utilizada como único critério de comparação, e uma avaliação subjetiva também é requerida. No entanto, os dados gerados nas medições revelaram detalhes interessantes de características das técnicas, que no caso analisado, foram entendidos como indicativos de capacidade de suavização do movimento.

Os métodos de interpolação, especialmente baseados em *splines* são de fato boas opções, pois acompanham as inclinações associadas ao sinal de animação, possuem controle local, podem aproximar ou interpolar pontos de controle provendo suavidade ao sinal ou transição suave na representação (animação), são computacionalmente eficientes, possibilitam aumento de resolução por subdivisão e adição de pontos e são facilmente modificados por meio de transformações sobre os pontos de controle nos quadros-chave.

Dentre as possibilidades de trabalhos futuros envolvendo a pesquisa realizada estão a ampliação da análise efetuada considerando outros tipos de animação e sinais, abordagem do trabalho focado na animação assistida por computador, ou até mesmo uma combinação entre a animação assistida e a animação gerada pelo computador. Também podem ser feitas análises considerando os diversos fatores que compõe uma cena, tais como o posicionamento da câmera e efeitos de luz e sombras. Outra análise interessante se refere aos controles de velocidade e aceleração ao longo da trajetória do movimento.

Fora do contexto de animação poderiam ser feitas análises das técnicas de interpolação estudadas neste trabalho juntamente com outras em aplicações envolvendo processamento de imagem e codificação de vídeo.

# REFERÊNCIAS


ARNOLD, Ken; GOSLING, James; HOLMES, David. **A linguagem de programação Java**. Tradução de Maria Lúcia Lang Lisboa. 4. ed. Porto Alegre: Bookman, 2007. 800 p.

AVGERAKIS, George. **Digital animation bible:** creating professional animation with 3ds Max, LightWave and Maya. USA: McGraw-Hill, 2004. 330 p.

AZEVEDO, Eduardo; CONCI, Aura. **Computação gráfica**: geração de imagens. Rio de Janeiro: Elsevier, 2003. 353 p.

BURDEN, R. L.; FAIRES, J.D. **Análise numérica**. Tradução de Ricardo Lenzi Tombi. Revisão técnica de Leonardo Freire Melo. São Paulo: Pioneira Thomson Learning, 2003. 737 p.

BUSS, S.R. **3-d computer graphics**: a mathematical introduction with opengl. University of California, San Diego: Cambridge University Press, 2003. 371 p.

CATMULL, Edwin. The problems of computer-assisted animation. **Acm Siggraph Computer Graphics**, New York, v. 12, n. 3, p.348-353, ago. 1978.

CLÁUDIO, Dalcidio Moraes; MARINS, Jussara Maria. **Cálculo Numérico Computacional:** Teoria e Prática. São Paulo: Atlas, 2000. 464 p.

DAVISON, Andrew. **Pro Java 6 3D Game Development:** Java 3D, JOGL, JImput, and JOAL APIs. Berkeley, California, USA: Apress, 2007. 498 p.

DEITEL, H.M. DEITEL, P.J. **Java, como programar.** Tradução de Edson Furnankiewics. 3. ed. Porto Alegre: Bookman, 2001. 1202 p.

FOLEY, J. D. et al. **Computer graphics:** principles and practice. 2nd ed. Reading, MA, USA: Addison-Wesley, July 1999. 1175 p.

FRANCO, N. B. **Cálculo numérico**. São Paulo: Pearson Prentice Hall, 2006. 520p.

GOMES, Jonas; VELHO, Luiz. **Computação gráfica:** imagem. Rio de Janeiro: IMPA/SBM, 1994. 424 p.

GOMES, Jonas; VELHO, Luiz. **Computação gráfica:** volume 1. Rio de Janeiro: IMPA/SBM, 1998. 323 p.

GOVIL-PAI, S. **Principles of computer graphics: theory and practice using opengl and maya.** Sunnyvale, California, USA: Springer, 2004. 299 p.

GREENBERG, Ira. **Processing: Creative Coding and Computational Art**. New York: Springer, 2007. 810 p.

HEARN, D.; BAKER, M. P. **Computer Graphics: C version**. 2nd ed. New Jersey: Prentice Hall, 1996. 652 p.

HOCH, M.; FLEISCHMANN, G.; GIROD, B. Modeling and animation of facial expressions based on B-Splines. **The Visual Computer**, Berlin, v. 11, n. 2, p.87-95, feb. 1994. Mensal.

JSR-000231 Java Bindings for the OpenGL API: Final Release. Santa Clara, CA, USA: Sun Microsystems, Aug. 2006. Disponível em: <http://jcp.org/aboutJava/communityprocess/final/jsr231/index.html>. Acesso em: 22 jun. 2009.

JUNIOR, Annibal Hetem. **Computação gráfica**. Rio de Janeiro: LTC, 2006. 161 p.

KAPLAN, Craig. **Introduction to computer graphics.** Waterloo, Ontario, CA: Springer, 2003. 208 p.

KOCHANECK, Doris H. U.; BARTELS, Richard H. Interpolating splines with local tension, continuity, and bias control**. ACM SIGGRAPH.** California, v. 18, n. 3, p.33–41, 1984.

KUPERBERG, Marcia et al. **A Guide to Computer Animation for TV, Games, Multimedia and Web**. Woburn, MA, USA: Focal Press, 2002. 231 p.

LENGYEL, Eric. **Mathematics for 3D Game Programming and Computer Graphics.** 2nd ed. Hingham, MA, USA: Charles River Media, 2004. 551 p.

MARTZ, Paul. **OpenGL distilled.** Boston, USA: Addison Wesley Professional, 2006. 304 p.

MAZZOLA, Guerino; MILMEISTER, Gérard; WEISSMANN, Jody. **Comprehensive mathematics for computer scientists 2:** calculus and ODEs, splines, probability, fourier and wavelet theory, fractals and neural networks, categories and lambda calculus. Berlin: Springer, 2005. 355 p.

MCKINLEY, Michael. **The game animators guide to Maya**. Indianapolis, USA: Wiley Publishing, 2006. 252 p.

MEALING, S. **The art and science of computer animation**. Exeter, England: Intellect, 1998. 356 p.

MIRANDA, José Iguelmar; MARABITA, Antonio Marcello. Fusão Espectral de Imagens de Satélite no Espaço de Cores. **Comunicado técnico, 84**. Campinas: Embrapa Informática Agropecuária, 2007. ISSN 1677-8464. Disponível em: <http://www.repdigital.cnptia.embrapa.br/bitstream/CNPTIA/11920/1/comtec84.pdf>. Acesso em: 20 maio 2009.

MULLEN, Tony. **Introducing character animation with Blender**. Indianapolis, USA: Wiley Publishing, 2007. 494 p.

NAMIKAWA, Laércio Massaru. Suavização de Isolinhas por Meio de Spline de Catmull-Rom. **Anais X SBSR**. Foz do Iguaçu: INPE, p. 473-479, abr. 2001. Sessão Técnica Oral.

NEMETHOVA, Olivia et al. PSNR-based estimation of subjective time-variant video quality for móbiles. **Proceedings of MESAQIN 2006**. Prag, Tschechien, CZ, June, 2006.

NETO, Walter Dutra da Silveira; MELO, Andrey Krepsky de. Técnicas de Animação em Ambientes 3D. **Dapesquisa**: Revista de investigação em Artes, Florianópolis, v. 1, n. 2, ago. 2004. Anual.

NICOLEIT, E. R.; SEARA, R. Considerações Sobre o Uso de Filtragem B-Spline Aplicada a uma Estrutura de Codificação de Vídeo de Dois Estágios Escalável Espacialmente para Operação em Baixas Taxas de Bits. In: **Anais do XIX Simpósio Brasileiro de Telecomunicações,** Fortaleza, 2001.

OUGAARD, Nikolaj. **NeHe Java ports**. 2006. Disponível em: < http://pepijn.fab4.be/software/nehe-java-ports/comment-page-2/#comment-2521 >. Acesso em: 23 set. 2009.

PARENT, Rick. **Computer animation:** algorithms and techniques. San Diego, USA: Academic Press, 2002. 528 p.

POGGIO, T.; VOORHEES, H.; YUILLE, A. A regularized solution to edge detection. **Journal of Complexity**: Academic Press: Orlando, USA, v. 4, n. 2, p. 106-123, Jun. 1988.

PORTER, Brett. **Lesson**:31. Grapevine, TX, USA, Nov. 2000. Disponível em: <http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=31>. Acesso em: 25 set. 2009.

PSIONIC, Si. **Free 3d Models**. UK, Jul. 2009. Disponível em: < http://www.psionic3d.co.uk/?page_id=25>. Acesso em: 29 set. 2009.

PULLI, Kari et al. **Mobile 3d graphics with opengl ES and M3G**. Burlington, MA, USA: Elsvier, 2008. 445 p.

ROGERS, David F. **An introduction to NURBS:** with historical perspective. San Francisco, USA: Academic Press, 2001. 324 p.

RUGGIERO, M. A. G.; LOPES, V. L. R. **Calculo numérico:** aspectos teóricos e computacionais. 2. ed. São Paulo: Pearson Makron Books, 1996. 406 p.

SALOMON, David. **Curves and surfaces for computer graphics**. Northridge, California: Springer, 2006. 461 p.

SANTOS, Adriana dos. **Protótipo de Software para Geração de Animações por Quadros-chaves Utilizando a Técnica de Interpolação**. TCC (Graduação), Departamento de Ciências da Computação, Universidade Regional de Blumenau, Blumenau, SC, 2000.

SHREINER, Dave et al. OpenGL ARB. **OpenGL Programming Guide.** 5th ed. Boston: Addison-Wesley, 2006. 616 p.

SLINKER, Geoffrey. **Inbetweening using a physically based model and nonlinear path interpolation**. Tese (Mestrado), Department Of Computer Science, Brigham Young University, Provo, UT, 1992.

UNSER, Michael. Splines: A perfect fit for signal and image processing**. IEEE Signal Processing Magazine.** Maryland : IEEE Signal Processing Society, v. 16, p. 22–38, Nov. 1999. Bimestral.

UNSER, M.; ALDROUBI, A.; EDEN, M. Fast B-Spline Transforms for Continuous Image Representation and Interpolation. **IEEE Transactions on Pattern Analysis.** Los Alamitos, California: IEEE Computer Society, v. 13, n. 3, p. 277-285, Mar. 1991. Mensal.

VERTH, James M. Van, BISHOP, Lars M. **Essential mathematics for games and interactive applications:** a programmers guide. 2nd ed. Burlington, MA, USA: Elsevier, 2008. 673 p.

VINCE, John. **Mathematics for computer graphics**. 2nd ed. Bournemouth, UK: Springer, 2006. 248 p.

WINKLER, Stefan; MOHANDAS, Praveen. The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics. **IEEE Transaction on Broadcasting.** San Jose, California: IEEE Computer Society, v. 54, n. 3, p. 660-668, Sept. 2008.

WHITROW, Robert. **OpenGL Graphics Through Applications**. London: Springer-Verlag London, 2008. 330 p.

WRIGHT, Richard S. Jr; LIPCHAK, Benjamin; HAEMEL, Nicholas. **OpenGL Superbible:** Comprehensive Tutorial and Reference. 4th ed. Boston, USA: Springer-Verlag London, 2008. 330 p.