UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE SOFTWARE PARA GERAÇÃO DE ANIMAÇÕES POR QUADROS-CHAVES UTILIZANDO A TÉCNICA DE INTERPOLAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

ADRIANA DOS SANTOS

BLUMENAU, NOVEMBRO/2000

2000/2-01

PROTÓTIPO DE ANIMAÇÃO DE SOFTWARE DE ANIMAÇÃO POR QUADROS-CHAVES UTILIZANDO A TÉCNICA DE INERPOLAÇÃO

ADRIANA DOS SANTOS

ESTE TRABALHO DE CONCLUSÃO DE CURSO FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____
Prof. Dalton Solano dos Reis — Orientador na FURB

_____
Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____
Prof. Dalton Solano dos Reis

_____
Prof. Paulo Cesar Rodacki Gomes

_____
Prof. Antonio Carlos Tavares

DEDICATÓRIA

A meu pai (in memorian), que fica sempre na lembrança e na saudade.

A minha mãe, que é meu eterno anjo da guarda, estando sempre ao meu lado, nos melhores e piores momentos da vida.

A meu namorado, que é meu porto seguro, e que foi meu esteio de amparo às dificuldades encontradas.

AGRADECIMENTOS

A Deus, que é o centro da vida e do universo do qual faço parte;

A meu orientador Dalton, que com profissionalismo, competência me orientou, auxiliou e conduziu-me de forma certa para o fim deste trabalho;

Aos meus mestres, que com carinho, amizade e muita sabedoria serviram de ponte entre o conhecimento, o aprendizado e o crescimento pessoal e universitário;

A todos que direta ou indiretamente, proporcionaram, contribuíram e ajudaram nesta longa caminhada;

Muito Obrigado!

SUMÁRIO

DEDICATÓRIA ..... **III**

AGRADECIMENTOS ..... **IV**

SUMÁRIO ..... **V**

LISTA DE FIGURAS ..... **VII**

LISTA DE QUADROS E TABELAS ..... **VIII**

RESUMO ..... **IX**

ABSTRACT ..... **X**

1 INTRODUÇÃO **1**

1.1 MOTIVAÇÃO 2
1.2 OBJETIVOS 3
1.3 RELEVÂNCIA 3
1.4 ORGANIZAÇÃO DO TEXTO 4

2 ANIMAÇÃO **5**

2.1 ANIMAÇÃO USANDO O COMPUTADOR 8

3 TIPOS DE ANIMAÇÃO **11**

3.1 ANIMAÇÃO TRADICIONAL 11
3.2 ANIMAÇÃO *BITBL* 17
3.3 ANIMAÇÃO POR *SCRIPTS* 17
3.4 ANIMAÇÃO POR DESLOCAMENTO 18
3.5 ANIMAÇÃO *SCAN* 19
3.6 ANIMAÇÃO BIDIMENSIONAL (2D) 19
3.7 ANIMAÇÃO TRIDIMENSIONAL (3D) 21
3.8 ANIMAÇÃO EM TEMPO REAL E TEMPO SIMULADO 23
3.9 ANIMAÇÃO POR QUADROS-CHAVES 24

4 ANIMAÇÃO UTILIZANDO INTERPOLAÇÃO ENTRE QUADROS-CHAVES **27**

5 DESENVOLVIMENTO DO PROTÓTIPO **30**

5.1 ESPECIFICAÇÃO DO PROTÓTIPO 30
5.1.1 DIAGRAMA DE CONTEXTO 31
5.1.2 DIAGRAMA DE FLUXO DE DADOS 31
5.1.3 MER 32
5.1.4 FLUXOGRAMA 33
5.1.5 DICIONÁRIO DE DADOS 35

5.1.6 ARQUIVOS DE SCRIPT 35
5.2 IMPLEMENTAÇÃO DO PROTÓTIPO 36
5.2.1 INTERPRETAÇÃO DOS COMANDOS 37
5.2.2 DESENHO DAS FIGURAS 38
5.2.3 GERAN DO A ANIMAÇÃO 39
5.2.3.1 INTERPOLAÇÃO LINEAR PONTO-A-PONTO 40
5.2.3.2 INTERPOLAÇÃO LINEAR PONTO-A-PONTO RGB 41
5.2.3.3 INTERPOLAÇÃO LINEAR DE PONTOS MÉDIOS 41
5.2.4 FUNCIONAMENTO DO PROTÓTIPO 43
6 CONSIDERAÇÕES FINAIS 54
6.1 CONCLUSÃO 54
6.2 EXTENSÕES 54
ANEXO A: INTERPRETAÇÃO DO ARQUIVO *SCRIPT* 56
ANEXO B: SALVANDO UM ARQUIVO *SCRIPT* 58
REFERÊNCIAS BIBLIOGRÁFICAS 60

LISTA DE FIGURAS

FIGURA 1 - FITA DE PROJEÇÃO DE IMAGENS 6
FIGURA 2 - TRECHO DE *STORYBOARD* 12
FIGURA 3 - PLANOS DE FUNDO 13
FIGURA 4 - QUADROS-CHAVES I - HISTÓRIA EM QUADRINHOS 14
FIGURA 5 - QUADROS INTERMEDIÁRIOS I - HISTÓRIA EM QUADRINHOS 15
FIGURA 6 - SCRIPTS PARA PLANETAS 18
FIGURA 7 - QUADROS-CHAVES II - EXEMPLO COM TRIÂNGULOS 23
FIGURA 8 - QUADROS INTERMEDIÁRIOS II - EXEMPLO COM TRIÂNGULOS 25
FIGURA 9- INTERPOLAÇÃO I- QUADRO-CHAVE A E B 27
FIGURA 10 - INTERPOLAÇÃO II - SEQUÊNCIA DE ANJOS 28
FIGURA 11 - DIAGRAMA DE CONTEXTO 31
FIGURA 12 - DFD DIAGRAMA DE FLUXO DE DADOS 32
FIGURA 13 – MER 33
FIGURA 14 - FLUXOGRAMA 34
FIGURA 15 - ARQUIVO DE *SCRIPT* (TEXTO) 35
FIGURA 16 - COORDENADAS **X** E **Y** EQUIVALENTES PARA AS DUAS FIGURAS 40
FIGURA 17 - INTERPOLAÇÃO LINEAR PONTOS MÉDIOS 42
FIGURA 18 – JANELA PRINCIPAL DO *SOFTWARE* 43
FIGURA 19 - CAIXA DE DIÁLOGO ABRIR 44
FIGURA 20 - CAIXA DE DIÁLOGO DA OPÇÃO NOVO 45
FIGURA 21 - TELA COM AS TECLAS DE ATALHO HABILITADAS 46
FIGURA 22 - FIGURA AMPLIADA PELO *ZOOM IN* 47
FIGURA 23 - EDIÇÃO DOS NODOS 48
FIGURA 24 - FIGURA TRIANGULARIZADA 49
FIGURA 25 - TELA PARÂMETROS 50
FIGURA 26 - QUADROS INTERMEDIÁRIOS DO MÉTODO INTER. LINEAR SIMPLES PONTO-A-PONTO 51
FIGURA 27 - QUADROS INTERMEDIÁRIOS PELO MÉTODO DE INTER. LINEAR PONTO-A-PONTO RGB 52
FIGURA 28 - QUADROS INTERMEDIÁRIOS PELO MÉTODO DE INTER. LINEAR DE PONTOS MÉDIOS 53

LISTA DE QUADROS E TABELAS

QUADRO 1 - DICIONÁRIO DE DADOS 35
QUADRO 2 - ALGORITMO PARA REDIMENSIONAR A FORM 38
QUADRO 3 - REDIMENSIONAR TELA COM O **ZOOM IN** 39

TABELA 1 - EXEMPLO DE COMANDOS EM ARQUIVO *SCRIPT* 36

RESUMO

Este trabalho consiste num estudo dos métodos utilizados para gerar animação dando maior ênfase no método de animação por interpolação de quadros-chaves para gerar quadros intermediários, utilizando técnicas de morfismo, onde as transformações são efetuadas por interpolação das imagens origem para a imagem destino. Apresenta também, a especificação e implementação de um protótipo de *software* para a geração da animação possibilitando a aplicação desta técnica. Através de uma interface gráfica simples, permite-se ao usuário gerar quadros intermediários relativos às imagens desejadas através da triangularização entre elas. A plataforma de desenvolvimento e execução é o Windows 98, utilizando-se o Borland Delphi 4.0 como ferramenta de desenvolvimento.

ABSTRACT

This work consists in a research of the methods used to produce animation, giving more emphasis to the animation method of interpolation of key-squares to create intermediate squares. It makes use of morphs technique, where the changes are made by interpolation of the origin images to the destiny image. It presents also, technique of morphs, the specification and implementation of a software prototype to the animation generation, making possible the use of this technique. Through a simple graphic interface, the software application permits the user to create intermediate squares derived the wanted images throughout the triangulation between then. The development and execution structure is the Windows 98, using the Borland Delphi 4.0 as a development tool.

1 INTRODUÇÃO

Atualmente a animação é sem dúvida, uma área de grande interesse de pesquisa para qualquer pessoa, desde os tempos da animação mais "precária". Antigamente fazer animação como nos desenhos animados na década de 60 era sem dúvida, muito trabalhosa, pois o desenhista teria que desenhar quadro a quadro cada cena, para depois então sobrepor uma cena a outra e então gerar a animação ([FIG1987]).

O computador trouxe inúmeros benefícios e vantagens para a animação como redução de tempo para gerar cenas, redução de desenhistas, maior perfeição e também facilitou em muito na própria escolha de personagens e cores, onde o artista tem maior diversidade de escolhas e opções a serem utilizadas.

Nas produções de desenhos animados, mantém-se praticamente todos os processos tradicionais de geração de desenhos animados, automatizando-os. Existem, no entanto, inúmeras técnicas para a produção de imagens que se movimentam, como por exemplo a técnica de interpolação. Veremos ao longo desse trabalho, dados relevantes sobre as técnicas mais utilizadas e mais detalhadamente sobre a técnica que usa a interpolação de quadros-chaves para gerar a animação ([FIG1987]).

A partir de imagens, cenas já modeladas, pode-se então criar quadros-chaves. Quadros-chaves são quadros que representam pontos importantes da animação indicando o início e o fim de cada movimento. E com base nestes, são criados então, quadros intermediários entre os quadros-chaves para que os movimentos sejam contínuos. Estes quadros são bem mais fáceis de serem animados já que vão incorporando pequenas alterações em um quadro-chave até chegar o próximo ([FIG1987]).

Uma das técnicas que pode ser utilizada neste processo, é a técnica de interpolação de imagens, ou seja, com base no primeiro quadro-chave, faz-se a interpolação para gerar os quadros intermediários até o próximo quadro-chave. Esta técnica é, indubitavelmente, uma das mais importantes contribuições do computador à animação e continua sendo aperfeiçoada até hoje ([FIG1987]).

Nos próximos capítulos veremos sobre animação, seus tipos e em especial o método de animação por interpolação de quadros-chaves, como também a implementação destes quadros intermediários, através de um protótipo de software de animação. Para geração de imagens contínuas, será utilizada a linguagem de programação Delphi 4.0, utilizando-se de uma fonte de quadros-chaves definida por figuras do tipo *bitmap*. Na elaboração deste trabalho tomou-se por roteiro geral as seguintes etapas:

a) pesquisar técnicas de interpolação;
b) estudar a animação por computador utilizando a técnica de interpolação;
c) especificar e implementar um protótipo de software gráfico para gerar a interpolação entre os quadros que serão definidos como quadros-chaves;
d) realizar testes e validações sobre o desempenho obtido.

1.1 MOTIVAÇÃO

A animação já faz parte de nossas vidas, e a grande procura por essa área devido a publicidade, a Internet e filmes animados para adultos e crianças, tem gerado um crescimento espantoso nos últimos anos.

No entanto, esta procura pela animação computadorizada comprometeu, de certa forma, os hardwares e softwares existentes no mercado, exigindo a cada avanço da animação um avanço tecnológico eqüivalente, como por exemplo, processadores mais eficientes e ágeis, dispositivos de saída gráficos mais precisos e softwares mais completos e fáceis de serem manipulados. Por outro lado, com estes recursos, um outro problema se associa, a de custos muito altos, o que acaba por fazer uma seleção de profissionais que possam ter a mão material suficiente para trabalhar.

Deste modo, diferentes métodos de animação e principalmente a animação pela interpolação de quadros-chaves, tem se tornado uma ampla área de estudos, devido ao seu grande uso e principalmente por estar em evolução constante.

1.2 OBJETIVOS

O objetivo principal deste trabalho é especificar e implementar um protótipo de software que habilite um ambiente de animação, gerando imagens contínuas a partir de figuras do tipo *bitmap*, utilizando então uma técnica de interpolação matemática. O trabalho também possui os seguintes objetivos específicos:

a) revisar os diferentes tipos de animação computadorizada;

b) estudar mais detalhadamente a técnica de animação de imagens, a partir de quadros-chaves, definidos por figuras do tipo *bitmap*, utilizando um método de interpolação matemática;

c) especificar e implementar um protótipo de software que possibilite um ambiente de animação, gerando imagens contínuas a partir dos quadros-chaves definidos no ítem b;

d) realizar testes e validações.

1.3 RELEVÂNCIA

Este trabalho de conclusão de curso apresenta os seguintes aspectos tecnológicos relevantes:

a) realiza um estudo sobre os diferentes métodos de se fazer animação;

b) apresenta um aprofundamento da técnica de animação por computador através da interpolação matemática dos quadros previamente definidos como chaves;

c) envolve uma técnica não trivial da Computação Gráfica.

1.4 ORGANIZAÇÃO DO TEXTO

O texto a seguir organiza-se nos capítulos abaixo:

a) Capítulo 1- Introdução: o presente capítulo, descreve o trabalho de forma geral.

b) Capítulo 2 – Animação: a forma como surgiu a animação, as primeiras animações realizadas, as primeiras pesquisas, e a importância de Walt Disney para o mundo da animação estão escritas neste capítulo.

c) Capítulo 3 – Tipos de animação: neste capítulo estão descritas as técnicas mais utilizadas na geração de imagens, desde os métodos mais tradicionais até os métodos que estão ainda sendo pesquisados.

d) Capítulo 4 – Animação utilizando interpolação entre quadros-chaves e método matemático.

e) Capítulo 5 – Neste capítulo descrevem-se as principais características e os princípios de funcionamento do protótipo. Descreve-se nesta etapa o funcionamento do software incluindo a apresentação de suas telas e detalhes de implementação.

f) Capítulo 6 – Considerações Finais: relata-se aqui as conclusões, dificuldades encontradas no decorrer do trabalho, as limitações do software e por fim, sugestões para futuros trabalhos.

2 ANIMAÇÃO

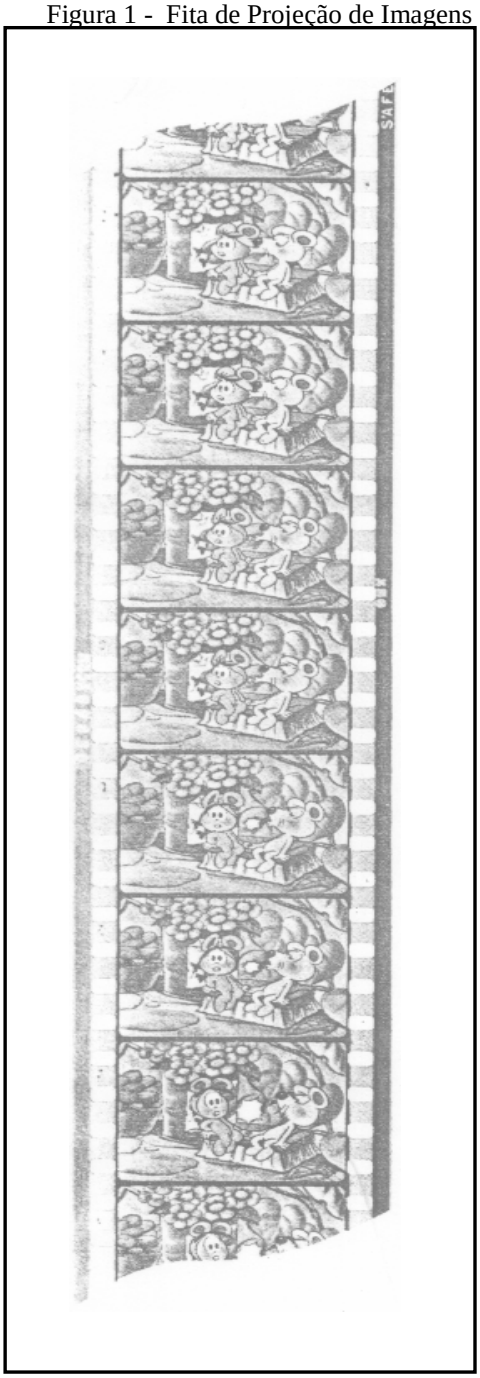
Imagens são melhores representantes da realidade do que um texto, mas se elas estiverem em movimento, podem resultar em uma compreensão ainda maior sobre o assunto em questão. No computador podemos colocar figuras em movimento através de algumas técnicas de animação.

O homem, sempre tentando copiar aquilo que era comum na natureza, criou algumas formas de movimento ilusório. Uma das primeiras técnicas para animar figuras, consistia em desenhar um objeto em cada folha de papel, sendo que cada folha possuía uma cópia mais adiantada do objeto. Essas folhas eram então movimentadas rapidamente, dando ao espectador a ilusão de movimento ([FIG1987]).

A animação foi realmente a precursora dos filmes e do cinema. Teve seu início em 1824, quando um estudante publicou um artigo chamado *The Persistence of Vision with Regard to Movig Objects*, o qual estabelecia que o olho humano retém uma imagem por fração de segundos a mais do que a exposição da imagem. Para demonstrar esse princípio, os cientistas inventaram muitos brinquedos óticos de salão no início do século 19 ([MOR1995]).

Graças a uma característica do olho humano, é possível que se produza a ilusão de um movimento contínuo através de projeção de cenas sem continuidade (discretas). Na realidade é possível para o homem observar qualquer movimento de forma contínua, já que o processo de imagens do seu sistema de visão é descontínuo, ou seja, o seu olho funciona como uma máquina fotográfica que tira fotos a intervalos regulares, as quais são enviadas para o cérebro que, analisando-as em seqüência inferem as imagens intermediárias entre uma “foto” e outra. Como o intervalo entre as fotos é bastante pequeno, cerca de 1/16 segundos, não é difícil para o cérebro ter a idéia de continuidade dos movimentos ([FIG1987]). Na figura abaixo (Figura 1), têm-se uma fita que projeta cenas discretas, onde pode-se perceber este movimento contínuo.

Figura 1 - Fita de Projeção de Imagens



Fonte: [FIG1987]

[a1] Comentário:

O processo de animação usado para criar desenhos é chamado de *cel anination.* A palavra *cel* vem das folhas de celulose (papel transparente) que eram usadas em 1915 pelos primeiros artistas de desenhos animados. Eles desenhavam nas folhas de papel transparente para permitir que os complexos planos de fundo fossem mostrados por trás dos personagens em movimento. Isso liberava os animadores de ter que desenhar o plano de fundo em cada quadro. Dessa forma os personagens do desenho animado poderiam se mover e pular em volta do plano de fundo estático e somente os personagens tinham que ser animados em cada quadro ([MOR1995], [FIG1987]).

A idéia básica da *cel animation* é que uma figura é criada para cada quadro. Alterações muito pequenas ocorrem em cada quadro. Quando esses quadros são reproduzidos em uma alta velocidade, a ilusão do movimento acontece.

O principal passo subseqüente na animação veio com o desenvolvimento do filme transparente à luz. Em 1887, um fotógrafo amador de New Jersey, criou uma emulsão sensível à luz em filme de celulóide. Pouco tempo depois, George Eastman desenvolveu um filme similar que poderia ser usado com a câmera Kodak que ele inventou. Esse novo filme de celulóide tinha muitas vantagens, e por fim, ele acabou fornecendo a base para o nascimento dos filmes para câmeras e projetores dos dias atuais ([MOR1995]).

Durante os primeiros tempos era necessário um desenho completo para cada quadro de animação. Esse problema foi resolvido por John Bray por volta de 1913. Bray passou a usar um papel translúcido para desenhar os planos de fundo do desenho animado. Ear Hurd também usou folhas de celulóide, mas ele pintava os personagens nas folhas de forma que os personagens podiam ser facilmente sobrepostos em planos de fundo complexos ([MOR1995]).

Alguns anos mais tarde, um dos mais memoráveis personagens do desenho animado fez o seu *debut:* Mikey Mouse. Em 1923, Walt Disney começou a criar histórias para crianças em desenho animado, e pouco tempo depois criou o curta metragem. Naquela época a tendência popular do cinema era concluir uma trilha sonora sincronizada ([MOR1995]).

Desde a criação do cinema então os processos para a criação de desenhos animados vem se desenvolvendo. A utilização do computador tem se mostrado de grande força nas empresas de efeitos especiais por diminuir em muito os gastos no aprendizado de produção de desenhos barateando muito os custos e permitindo que os artistas realmente façam arte ([MOR1995]).

Enfim, uma simples animação é a ilusão do movimento. Essa ilusão pode ser obtida então, com uma série de imagens produzidas de forma que o olho humano perceba como se estivessem se movimentando ([MOR1995]).

2.1 ANIMAÇÃO USANDO O COMPUTADOR

Nos últimos anos, a animação por computador tem se transformado na força motriz das empresas de efeitos especiais. Se quisermos, podemos ver dinossauros rosnando e atacando-se uns aos outros em um parque de diversões Jurástico. A animação por computador têm tornado a criação de ilusões fantásticas como esta, por exemplo.

A animação por computador é um campo exclusivo onde o usuário pode expressar o seu talento artístico mesmo que não tenha a destreza física requerida por formas de expressão artística mais tradicionais. Por outro lado, a sua aparente complexidade técnica pode parecer intimidante.

Ao longo da história, os artistas investiram horas incontáveis aprendendo a manejar um pincel, um lápis ou um instrumento musical, depois praticaram a sua arte por anos a fio para que conseguissem produzir algo de valor. A computação gráfica diminui o tempo gasto no aprendizado e permite que os artistas realmente façam arte ([FIG1987]).

Um escultor pode trabalhar no espaço tridimensional, mas a escultura representa um momento fixo no tempo e sofre limitação das cores baseadas em pigmentos. A animação por computador pode simular o movimento ou a passagem do tempo e pode fazer isso sem restrições de visualização ([MOR1995]).

A animação por computador é muito diferente de todos os outros meios artísticos. Ela permite que o artista crie mundos inteiros dentro do computador. Consegue-se não só criar modelos tridimensionais semelhantes à escultura, mas também pode-se especificar a cor e a posição das luzes, sombras e até mesmo efeitos atmosféricos tais como chuva e neve ([MOR1995]).

Com o computador, trabalhamos então com uma seqüência de imagens, o que pode tornar o processo de geração da animação uma tarefa bastante lenta dependendo da máquina usada para a implementação ([BIN1994]).

No cinema temos diversos quadros (ou fotografias) gravadas em seqüência. Quando exibimos estes quadros, uma pós o outro em determinada velocidade temos a impressão que as imagens na tela estão realmente se movimentando ([BIN1994]).

Os filmes de desenho animado consistem de seqüências de imagens que são projetadas à velocidade de 48 quadros por segundo. Como 24 imagens diferentes por segundo é o suficiente para a percepção do movimento contínuo, cada quadro é projetado duas vezes. Isso se faz necessário para que seja evitado o fenômeno denominado cintilação causado pelo branqueamento da tela na passagem de um quadro para o outro ([FIG1987]).

A velocidade com que são projetados os quadros é o maior responsável pela perfeição da imagem final. Por exemplo, se os quadros são exibidos em baixa velocidade, tem-se a impressão de que os movimentos pareçem desajeitados, por outro lado, se os quadros são exibidos com a velocidade adequada percebe-se a animação de forma contínua, sem defeitos ([MOR1995]).

As imagens de televisão são transmitidas na freqüência de 30 quadros por segundo. Mas para evitar a cintilação, a tela é varrida 60 vezes por segundo e cada imagem é projetada em duas partes, uma composta pelas linhas ímpares e outra, pelas linhas pares (técnica de entrelaçamento).

Isso representa um grande desafio, porque, mesmo um desenho animado de 15 minutos na televisão, exige mais de 27.000 mil quadros e um filme de 2 horas de duração exige aproximadamente 216.000 quadros ([MOR1995]).

Com a animação por computador pode-se acrescentar, modificar movimentos de forma que não se precise redesenhar todo o filme ou a cena para poder efetuar a alteração, agilizando o processo de criação ([BIN1994]).

Animando pelo computador, pode-se por fim, acrescentar movimento criando a ilusão de se estar caminhando dentro de uma cena. Pode-se mover as fontes de luz enquanto o computador atualiza automaticamente as reflexões e sombras dentro da realidade artificial. Os mundos artificiais criados dentro do computador podem ser explorados sob quaisquer pontos de vista ([MOR1995]).

Com a evolução tanto do hardware como do software, os estúdios de animação puderam desenvolver técnicas para aumentar a produtividade, passando por várias formas diferentes de fazer animação, como veremos no próximo capítulo.

3 TIPOS DE ANIMAÇÃO

A criação de animações por computador representa um dos campos mais avançados da computação gráfica atual. Ela exige muito dos animadores, pela complexidade dos métodos. Exige muito também dos computadores, pela carga computacional necessária. Mesmo assim o computador pode aumentar bastante a produtividade da animação, acarretando considerável economia de trabalho. Isso ocorre porque o computador pode ser usado para automatizar algumas tarefas repetitivas, que representam grande parte da carga de trabalho dos animadores.

A vantagem do computador se mostra mais evidente na animação tridimensional, a qual que abre novas oportunidades aos tradicionais métodos manuais. Veremos mais detalhadamente este método no sub-ítem Animação Tridimensional ([PAU2000]). Existem várias técnicas e métodos usadas por computador para gerar a animação. A seguir são descritos alguns tipos mais comuns.

3.1 ANIMAÇÃO TRADICIONAL

Existem inúmeros processos para a geração de desenhos animados, desde produções caseiras ou criações de pequenas seqüências para comerciais de TV até filmes de longa-metragem como por exemplo os filmes de Maurício de Souza ou Walt Disney. O processo tradicional básico, que vem sendo utilizado e aperfeiçoado desde os primeiros desenhos animados ([FIG1987]).

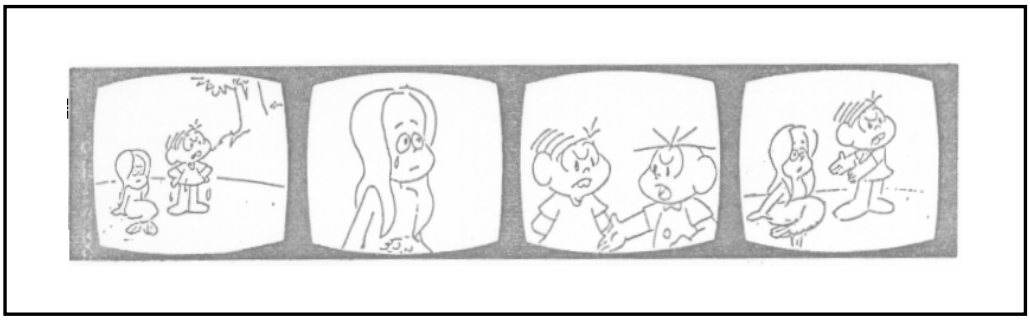
Segundo [PAU2000] o uso do computador na animação tradicional provavelmente começou com o emprego de microcomputadores apropriados para o controle de tempo real. Eles eram utilizados para controlar os movimentos dos mecanismos usados em animação em celulóide.

Tradicionalmente os desenhos animados são criados por uma equipe de artistas. A medida que uma idéia se desenvolve, os *esboços* são criados. Estes esboços são rascunhos que

mostram as roupas e como deve ser a aparência do fundo. Tudo é levado em consideração, desde cultura até idioma dos personagens ([LOP1990]).

Os conceitos da história são estruturados com uma técnica chamada de *storyboard*. Um *storyboard* é como uma fita cômica bastante detalhada, a qual ajuda o produtor a visualizar em como ficará o produto final ([MOR1995]). Na figura abaixo (Figura 2) temos um trecho de um *storyboard* que demonstra quadro a quadro a história.

Figura 2 - Trecho de *Storyboard*



Fonte: [FIG1987]

Sendo assim, antes que os desenhos comecem a ser produzidos, a história deve passar por um processo idêntico ao de produção de filmes, criando-se um roteiro com as seqüências, cenas e trilha sonora ([FIG1987]).

Depois que a história está refinada, a voz dos atores é representada e os efeitos sonoros e a música são gravados, os artistas de *layout* criam os desenhos a lápis para os fundos e os cenários dos desenhos animados. Estes vão para os artistas de fundo que criam a arte final para o fundo ([MOR1995]).

Os cenários, normalmente estáticos durante vários quadros, são definidos e pintados em folhas especiais. Geralmente, os cenários são bem maiores que os quadros que serão sobrepostos a eles, o que permite a movimentação dos personagens, aproximação o afastamento ([FIG1987]).

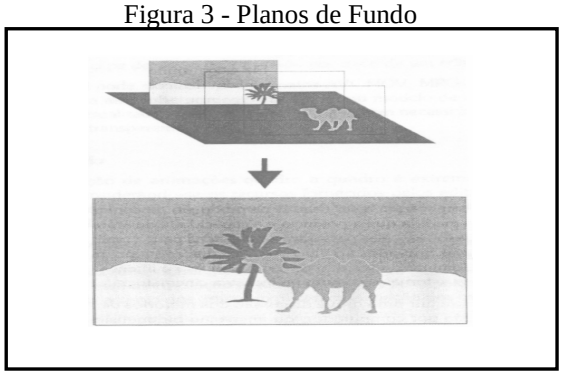
Um método bastante utilizado, para este tipo de animação, era uma câmera para a filmagem dos quadros, onde têm-se o cenário fixo em uma mesa horizontal, dispondo-se sobre ele os acetatos transparentes contendo os desenhos das partes animadas. Uma câmera disposta verticalmente e que possui vários graus de liberdade (aproximação, afastamento e rotação) registra quadro a quadro ([FIG1987]).

Devido ao fato de acetatos e cenários residirem em um único plano, os afastamentos e aproximações da câmera não conseguem transmitir um efeito de profundidade porque todos os elementos aumentam e diminuem na mesma proporção ([FIG1987] , [PAU2000]).

Já com o uso do computador, para a criação dos cenários, podemos trabalhar com desenhos compostos de múltiplas camadas, simulando movimentos na cena através de deslocamentos relativos destas camadas ([PAU2000]).

Na figura abaixo (Figura 3), têm-se um exemplo do deslocamento destas camadas, onde o plano de fundo (cenário) com a areia e o céu, é fixo. O plano da frente é uma folha transparente, onde o camelo está pintado. Para animar o movimento do camelo, basta trocar o plano da frente. Pode-se notar também que variando-se a distância entre as folhas, pode-se conseguir efeitos de *zoom*. O movimento do plano intermediário e do plano frontal para a frente dá uma impressão de aproximação da câmera. Este movimento é controlado por computador, em sistemas avançados de animação tradicional, para conseguir maior precisão ([PAU2000]).

Figura 3 - Planos de Fundo

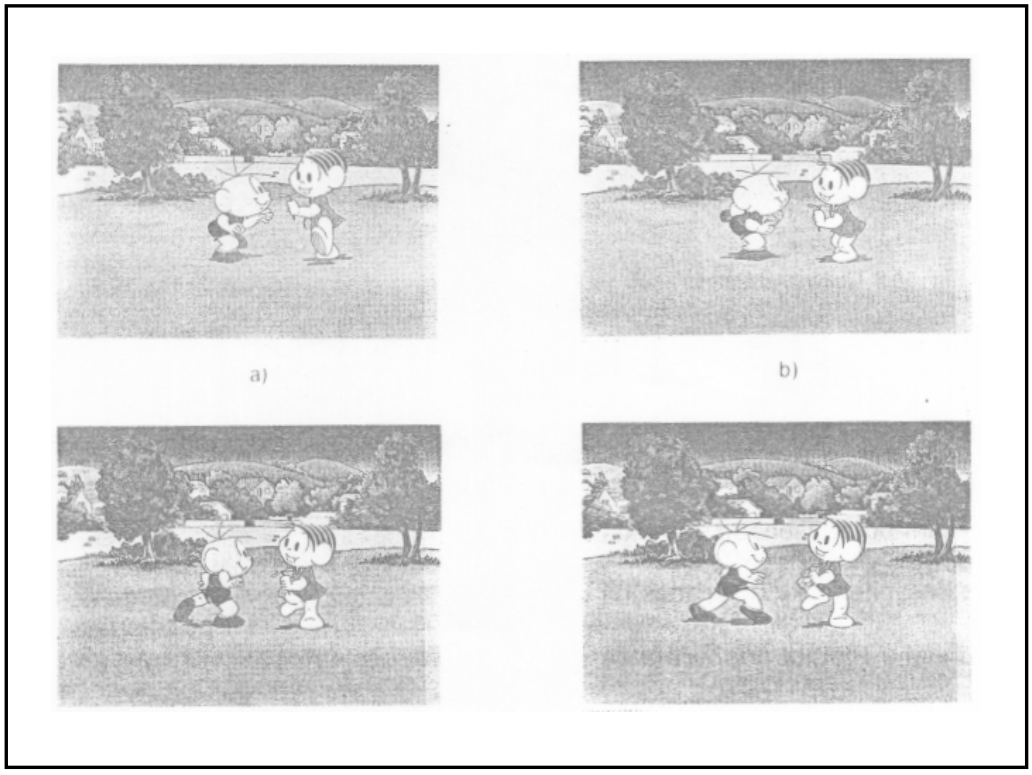


Fonte : [PAU2000]

Depois deste processo, então, os artistas principais começam a trabalhar criando os movimentos-chaves dos personagens animados ao longo do desenho animado. Os atores começam a criar os diálogos dos desenhos, o que permite aos artistas a análise da trilha sonora, e desenhar os personagens para corresponderem às palavras ([LOP1990]).

Embora vários quadros sejam necessários para cada segundo de movimento, os artistas geralmente desenham apenas os quadros que contêm os movimentos-chaves, ou principais, do personagem, como mostra a Figura 4 ([MOR1995]).

Figura 4 - Quadros-Chaves I - História em Quadrinhos

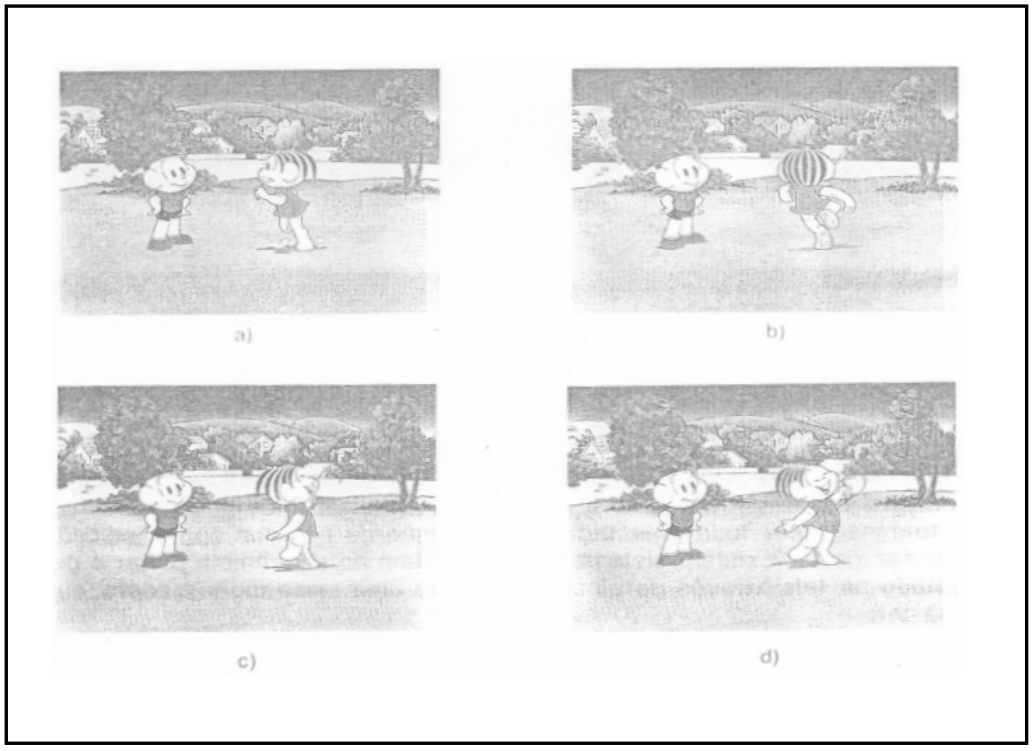


Fonte : [FIG1987]

Para preencher os quadros intermediários, os artistas assistentes desenham os quadros *in-between* (intermediários). Estes artistas executam o seu trabalho sobre uma mesa de luz, usando folhas de papel transparente. Como já visto anteriormente, à medida que desenham os quadros, eles sobrepõem cada página do quadro anterior e variam levemente o movimento representado no novo quadro, de forma que quando a seqüência é reproduzida em uma velocidade alta o movimento pareça natural ([LOP1990], [MOR1995], [PAU2000]).

Na figura abaixo (Figura 5), mostra-se um exemplo de geração de quadros-intermediários.

Figura 5 - Quadros Intermediários I - História em Quadrinhos



Fonte : [FIG1987]

Agora, o trabalho de animação passa para a produção final. Este processo é chamado de Desenhando e Pintando, onde os desenhos a lápis para cada quadro são limpos, desenhados à tinta e pintados. Um por um, os desenhos são transferidos para os celulóides. Os lados inversos dos celulóides são pintados para que os contornos de tinta permaneçam limpos. É um trabalho bastante demorado e deve ser feito com bastante cuidado, para que não haja mudanças drásticas de tonalidades entre as cores ([FIG1987], [MOR1995]).

As produções de animação tradicional, contam com o auxílio de computadores para um grande número de tarefas. A base destas animações continua a ser o desenho manual feito por artistas, ficando para o computador os processos mais demorados, e trabalhosos ([PAU2000]).

Atualmente, muitos produtores de animação usam computadores para melhorar estas técnicas tradicionais. Podendo digitalizar os rascunhos o computador pode reproduzir os rascunhos como 0animações de esboços. O computador também pode criar alguns quadros intermediários em muitos casos automaticamente. Tudo o que o artista precisa fazer é informar ao computador quais partes do personagem estão relacionados para cada quadro-chave. Depois o computador pode criar quantos quadros-chaves intermediários forem necessários para mover gradualmente os *pixels* das suas posições originais no primeiro quadro até a nova posição no segundo quadro ([MOR1995]).

Devido à espessura do acetato, a animação tradicional por celulóides é limitada ao número de celulóides que podem ser sobrepostos de uma só vez. Esta limitação não se aplica quando as celuloses são digitalizados em um computador. Outra grande vantagem que os computadores proporcionam aos artistas é no processo de Desenhando e Pintando, das celuloses individuais. No computador este processo é facilitado com uma ferramenta de preenchimento ([LOP1990]). E, por fim mais uma vantagem, é a capacidade de alterar as cores da animação depois que as celuloses são pintadas ([MOR1995]).

3.2 ANIMAÇÃO *BITBL*

Uma forma de mover objetos utilizada na produção de desenhos animados é denominada de animação por blocos de *bit* ou *bitbl*,. Esta técnica consiste em desenharmos um ambiente, chamado de imagem de fundo ou *background*, deixando-o fixo. Desenha-se então separadamente a seqüência do objeto a ser movido, também chamado de *foreground*. Por fim o objeto é mostrado na tela em suas diversas posições, apagando sempre a imagem anterior antes de mostrar a próxima imagem ([BIN1994]).

A operação de desenhar e apagar é necessária para que o desenho não deixe um rastro dele na tela. No entanto, se apagarmos a imagem da tela, o que sobrará no seu lugar será um quadrado preto e não a imagem de fundo que foi sobreposta.

Segundo ([FIG1987]), a solução para este problema consiste em salvar em outra região de memória, exatamente o espaço que será ocupado pelo bloco. Em seguida mostra-se o bloco e logo após, a mostra-se a imagem que foi salva anteriormente.

Assim, com esta seqüência, a operação de apaga e sobrepõe fica imperceptível ao usuário, dando a sensação de animação.

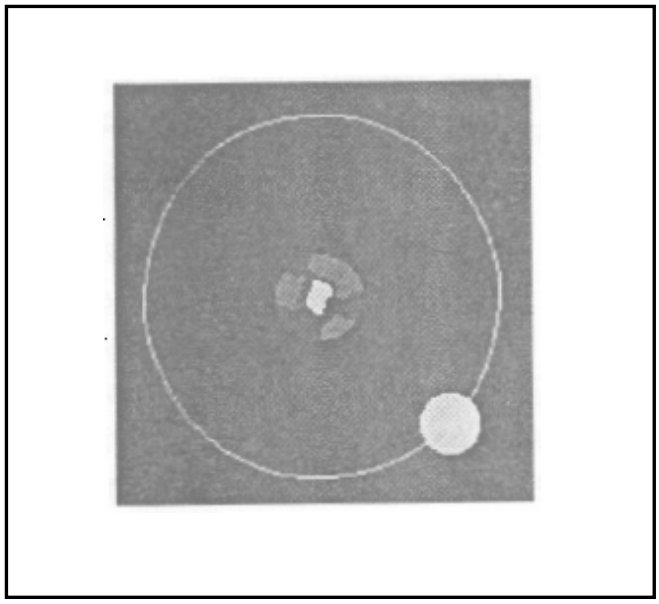
3.3 ANIMAÇÃO POR *SCRIPTS*

A animação por *scripts* utiliza-se de um *script* interpretado em tempo real para conseguir os efeitos desejados. Este *script* pode acionar a troca de imagens em uma seqüência animada, ou calcular o movimento de um objeto gráfico, possivelmente obedecendo a uma trajetória geométrica determinada, ou a leis da física ([PAU2000]).

O *script* pode ser usado também para comandar a ocultação sucessiva de diversos objetos, fornecendo a ilusão de que se trata do mesmo objeto sujeito a transformações. A ocultação pode ser feita em imagens, por exemplo, planetas, modificando-se a cor do objeto de modo que esta fique sendo a mesma do fundo ([PAU2000]).

Na Figura 6 no quadro abaixo temos um exemplo de animação por *scripts* em imagens com planetas.

Figura 6 - Scripts Para Planetas



Fonte : [PAU2000]

3.4 ANIMAÇÃO POR DESLOCAMENTO

Quando se possui uma imagem maior do que a tela pode representar, então tem-se que mover a imagem para que possa ser visualizá-la por inteiro. Esta animação consiste em movimentar pedaço por pedaço do desenho suavemente na horizontal ou vertical, conforme o caso. Esta técnica é chamada de animação por deslocamento, *planing* ou *clipping* ([BIN1994]).

A técnica consiste basicamente em mostrar a imagem sucessivas vezes na tela, variando o seu posicionamento de acordo com o que se deseja mostrar. Dessa forma, será realizado um movimento circular com a imagem, fazendo com que as partes que desaparecem do lado esquerdo reapareçam do lado direito, no momento adequado ([BIN1994]). Assim, a impressão que esta animação gera é que a tela está percorrendo a imagem e não a imagem se deslocando perante uma tela estática.

3.5 ANIMAÇÃO *SCAN*

Outro método bastante utilizado em processos convencionais, é a chamada animação *Scan*. Nele fotografa-se determinada cena e, alterando-se elementos de posição, mantém-se a exposição da câmera aberta. O efeito provocado é o de um rastro deixado pelo objeto movido. Suponha, por exemplo, que se deseje animar uma cena na qual um objeto parte de uma posição X, atinja o ponto Y, deixando um rastro atrás de si e, por fim, este rastro vai sumindo a partir de X ([FIG1987]).

Pelo processo convencional, deve-se fotografar passo a passo o objeto em transição para depois montar a cena. Conforme [FIG1987], com o computador, a animação *Scan* deve trabalhar com a memória da tela e não com exposição fotográfica. Então trabalha-se com sistemas de coordenadas, janelas, sombreamentos, transformações e misturas de imagens. O processo de mistura de imagens deve levar em conta os *pixels* da memória anterior e da atual. Como as possibilidades das cores são limitadas pelo número de *bits* reservados para ele, as intensidades das cores devem ser atribuídas de maneira a possibilitar várias sobreposições antes da saturação. Onde por fim, têm-se que o controle é efetuado sobre as intensidades dos *pixels*.

3.6 ANIMAÇÃO BIDIMENSIONAL (2D)

Os editores de animação bidimensional podem ser vistos como extensões de editores de pintura, que permitem a criação e o tratamento de seqüências de imagens. Freqüentemente, procura-se manter as mesmas características visuais da animação manual.

A animação bidimensional por computador é geralmente usada para desenhos animados e efeitos especiais tais como morfismo. A animação em 2D usa a terminologia da animação tradicional ([MOR1995]).

O processo é muito parecido com a pintura de um artista em um quadro, onde este processo pode manipular cada quadro que está sob o seu controle. O artista então pode usar o computador para manipular as suas imagens bidimensionais ([LOP1990]).

A animação bidimensional por computador pode ser dividida em quatro categorias principais: celulóides animados, efeitos óticos, morfismo e *color cycling* ([LOP1990]).

Nos celulóides animados, um personagem que foi criado e depois digitalizado, pode parecer caminhando na tela por uma série de quadros que mostram o personagem em várias etapas da caminhada. Os fundos podem ser criados e depois digitalizados como também podem ser criados diretamente no formato digital ([MOR1995], [LOP1990]).

Já os efeitos ópticos levam as celulóides animadas mais adiante. Seria então, utilizar o mesmo personagem caminhando em direção contrária da tela, fazendo-o diminuir até perdê-lo de vista. Para criar este efeito o computador diminuiria lentamente o tamanho do personagem ([MOR1995]).

O efeitos do morfismo são bastante similares ao efeitos ópticos. Pode então tentar gerar a animação entre dois personagens gerando quadros intermediários de forma a parecer suavemente a transição de um para o outro, e a medida que o primeiro vai se transformando no segundo personagem, o computador lentamente desaparece com o primeiro personagem ([MOR1995]).

E por fim, a última forma de animação por computador bidimensional é feita por *color cycling*. É o método pelo qual usa-se somente um quadro ou figura e o próprio computador gira em ciclos, ou troca as cores em um padrão pré-determinado. Ele é usado geralmente para efeitos de animações simples como a água correndo. Pode-se comparar com o mesmo efeito de “liga e desliga” das lâmpadas para fazê-las “correr” ([MIL2000]). É exatamente isso que o

computador faz, mas como cada *pixel* pode ter centenas, ou milhares de valores possíveis de cor, a diversidade é bastante grande dando espaço para a criatividade ([MOR1995]).

Todas as categorias mencionadas acima são manipuladas pelo método de interpolação que veremos no capítulo 4.

3.7 ANIMAÇÃO TRIDIMENSIONAL (3D)

Embora o uso de computadores na animação convencional seja hoje tão essencial quanto na editoração gráfica, é a animação tridimensional que permite explorar recursos e obter efeitos que não podem ser conseguidos de forma manual.

A animação tridimensional permite a visualização de objetos 3D por todos os ângulos, simulando filmagens no mundo real. Entretanto, ela exige muito tanto de animadores quanto de computadores. A modelagem tridimensional requer o emprego de técnicas avançadas de computação gráfica, com emprego de conhecimentos de geometria e de ótica. Por outro lado, a elaboração das imagens requer processamento muito intensivo, e precisa ser feita em computadores com a mais alta potência possível ([PAU2000]).

A animação tridimensional desempenha um papel cada vez mais importante em visualização técnica e científica, em que permite mostrar de forma intuitiva fenômenos e conceitos cuja representação exige grandes volumes de dados. As aplicações incluem *design* assistido por computador (CAD), *design* de novos produtos, estudos ambientais, e outros. Composições tridimensionais de imagens médicas são usadas, por exemplo, em tomografia, no planejamento de operações e ensino de medicina ([PAU2000]).

A animação tridimensional, por sua complexidade e custo de realização, é normalmente um trabalho de equipe, que deve ser realizado de forma disciplinada, e de acordo com uma metodologia precisa. O ciclo de criação de uma animação tridimensional pode ser decomposto nos seguintes passos: *modelagem*, *coreografia*, *elaboração* e *pós-produção* ([PAU2000]).

No processo de modelagem compreende a criação dos objetos geométricos que compõem as cenas, e a atribuição de suas propriedades de cores e materiais. Na coreografia

compreende a criação dos movimentos dos objetos, câmeras e luzes. Já no processo de elaboração compreende a produção de seqüências de imagens bidimensionais animadas a partir dos modelos e coreografias. E por fim, o processo de pós-produção é semelhante à edição de material de vídeo gravado ao vivo, incluindo a criação de arquivos ou a gravação de vídeo e áudio. Este ciclo pode ser percorrido várias vezes durante o desenvolvimento de uma animação ([PAU2000]).

Alguns estágios são bastante típicos durante estes ciclos:

- Animação em fio-de-arame, em que os objetos são exibidos apenas em representações de seus arcabouços, como contornos, esqueletos e armações. Permite verificar a modelagem geométrica, a coreografia e a sincronização com o som, sem muito investimento em tempo de computador.

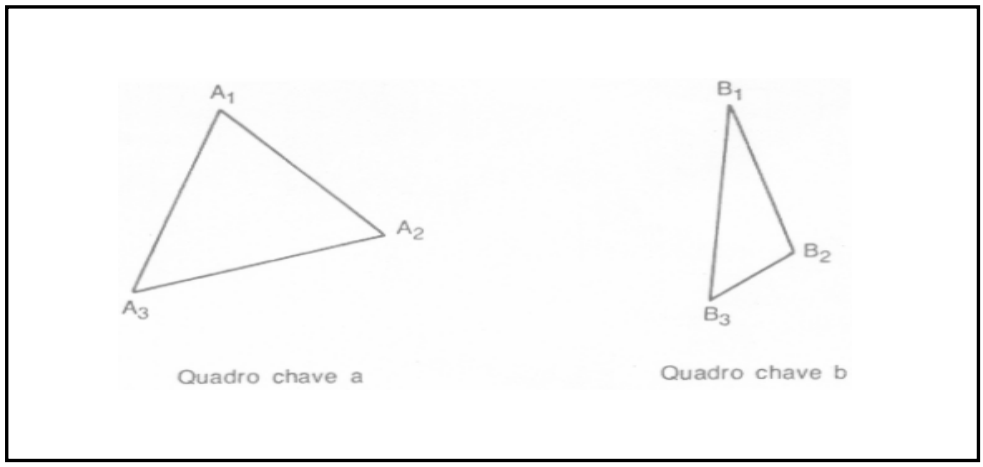
- Visualização preliminar, feita com seqüências de imagens estáticas (*storyboards*) e com seqüências animadas de baixa resolução.

- Visualização definitiva, feita na resolução e freqüência de quadros definidos pelos padrões da mídia definidas a ser usada (cinema, TV ou multimídia) ([PAU2000]).

Para gerar animação tridimensional criam-se então vários quadros-chaves (*keyframes*) ou também, simplesmente utilizar software onde se aplicam a cinemática inversa. Nos quadros-chaves, uma animação pode ser dividida em quantos quadros se acharem necessários ([LOP1990]).

A animação é gerada então definindo-se em que posição queremos que o objeto do quadro A tenha quando chegar no quadro B, por exemplo. Depois desta definição o próprio computador faz com que o objeto se movimente para que, quando chegar no quadro B, esteja na posição que foi definida inicialmente ([MOR1995]). A figura abaixo (Figura 7) nos mostra como acontece os movimentos chaves durante a transição de imagens.

Figura 7 - Quadros-chaves II - Exemplo com Triângulos



Fonte : [FIG1987]

No caso de animação utilizando a cinemática inversa, como em um caso bastante específico, o de um objeto caindo ao chão, não é necessário fazer o movimento do objeto passo a passo e modelando-o novamente. Neste caso a cinemática inversa permite que seja especificado que o objeto vai cair de uma altura X, com uma gravidade e fazendo com que o objeto caia ganhando velocidade ao longo do percurso e leve um tombo ao tocar o chão ([MOR1995]).

Embora seja uma técnica poderosa, não temos o total controle sobre o objeto, porque está sob o efeito da cinemática inversa, que por sua vez é controlada pelas leis da física ([MOR1995]).

3.8 ANIMAÇÃO EM TEMPO REAL E TEMPO SIMULADO

A geração de movimentos controlados diretamente por programas permite a criação de aplicativos bastante interessantes como videojogos, simuladores de vôo, ensaios mecânicos, programas educativos, entre outros.

Num simulador de vôo é indispensável que as imagens mostradas ao aprendiz de piloto sejam realistas e, portanto, animadas. Para que seja criada a ilusão de movimentos deve ser aplicado o mesmo princípio utilizado na animação de filmes, ou seja, várias imagens com pequenas variações entre si devem ser apresentadas em altas velocidades ([FIG1987]).

Independente de como os quadros de animação são criados no computador, eles ainda precisam ser produzidos em uma rápida sucessão para a ilusão de movimento. Esta reprodução é conseguida em alta velocidade de duas maneiras: em Tempo Real e em Tempo Simulado ([MOR1995]).

Tempo Real significa que o computador pode exibir os quadros a uma velocidade de 15 *fps* (*frames* por segundo)ou mais. Cada quadro é gerado a essa velocidade ou recuperado da memória e exibindo na tela do computador nesta velocidade. Pode-se ainda, se quiser, conectar um gravador de vídeo no computador e gravar a animação à medida em que ela se reproduz ([FIG1987]).

É o tipo de animação utilizado nos comerciais ou em filmes, onde devido a necessidade de alta resolução, a quantidade de dados aumenta consideravelmente, e para solucionar este problema os quadros individuais são armazenados na memória do computador e chamados um por um. Este método é também chamado de animação quadro-a-quadro. Este tipo de animação resulta na melhor qualidade possível porque cada figura pode ter um alto nível de detalhe ([MOR1995]).

Tempo Simulado é basicamente a compressão dos quadros individuais de forma que eles não ocupem muito espaço. Depois de comprimidos, estes quadros podem ser reproduzidos em velocidades e resoluções aceitáveis por algumas aplicações de vídeo ([MOR1995]).

3.9 ANIMAÇÃO POR QUADROS-CHAVES

A construção de animações quadro a quadro é extremamente trabalhosa, mesmo considerando-se os recursos fornecidos pelos editores de desenhos e imagens. Os quadros-

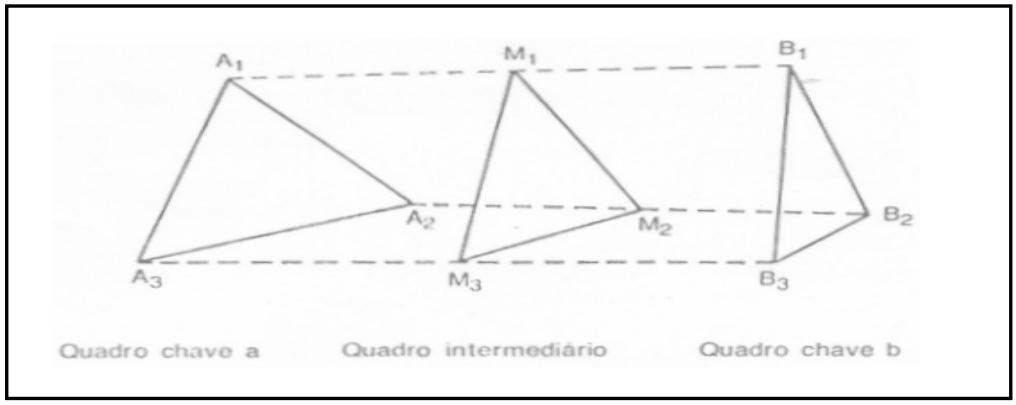
chaves são criados pelos animadores como desenhos estáticos, e vão trabalhando-os quadro a quadro ([PAU2000].)

Como foi descrito e explorado detalhadamente nos capítulos anteriores, após uma seqüência de esboços e informações sobre ação e enquadramento que mostra as principais passagens da história, os *storyboard* são definidos então a trilha sonora, o *layout* e o sincronismo com a trilha sonora ([FIG1987]).

O próximo passo é criar o cenário que normalmente é estático durante vários quadros. O processo mais importante depois de definido o cenário, são as criações dos quadros-chaves. Estes quadros são chamados de “chaves” porque representam os pontos importantes da animação indicando o início e o fim de cada movimento ([FIG1987])

Estes quadros-chaves definem claramente cada movimento. No entanto se projetados desta maneira, a animação ficaria cheia de saltos, com um realismo ainda mais pobre do que nos filmes mudos de antigamente. Para que os movimentos sejam contínuos é necessária a criação de quadros intermediários entre um quadro-chave e outro. Estes quadros são bem mais fáceis de serem gerados, já que vão incorporando pequenas alterações em um quadro-chave até chegar ao próximo ([FIG1987]). Este processo é bem representado na Figura 8 .

Figura 8 - Quadros Intermediários II - Exemplo com Triângulos



Fonte : [FIG1987]

Por fim, para a geração destes quadros intermediários têm-se várias formas de interpolação. Este trabalho ater-se-á no método de interpolação matemático para gerar a interpolação entre os quadros-chaves, como será descrito no próximo capítulo.

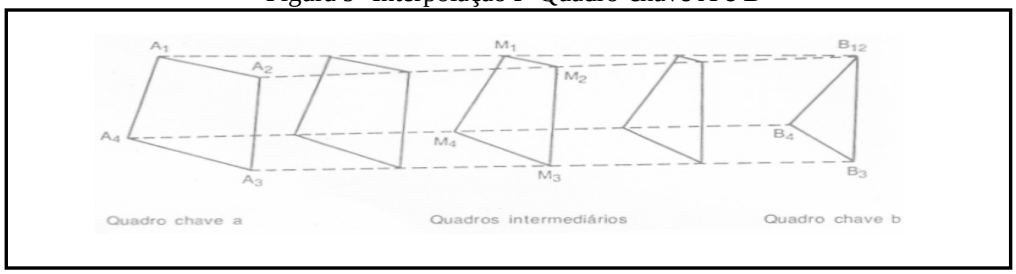
4 ANIMAÇÃO UTILIZANDO INTERPOLAÇÃO ENTRE QUADROS-CHAVES

O método de interpolação é uma técnica que prevê a continuidade derivada do movimento, em função de um quadro-chave. Tendo-se um quadro-chave inicial, que poderemos chamar de quadro de origem e um quadro-chave final que chamamos então de quadro destino. Fazendo então a geração de quadros intermediários através de interpolação entre os pontos correspondentes dos quadros iniciais e finais, este método de interpolação automatiza o processo tradicional, onde obtêm-se a animação feita passo-a-passo pelos artistas de criação de quadros-chaves ([PAU2000]).

Então o que se tem a fazer é interpolar cada ponto do quadro origem para um ponto correspondente do quadro destino, que chamamos de quadros intermediários. Para a geração dos quadros intermediários têm-se vários métodos, no entanto será focalizado o método de interpolação entre os quadros para representar a geração destes quadros. Quando todo o quadro destino estiver pronto, interpolado, então este deixa de ser destino e torna-se quadro de origem que será interpolado com um terceiro quadro-chave agora na posição de destino, e assim sucessivamente. Esta continuidade de movimento gera a interpolação de modo suave entre os quadros que definimos como chaves ao longo da animação ([WYV1989], [MAI1990]).

A Figura 9 mostra o processo de continuidade dos movimentos dos quadros intermediários entre o quadro-chave inicial mostrado como quadro-chave A, para o quadro-chave final mostrado como quadro-chave B

Figura 9- Interpolação I- Quadro-chave A e B



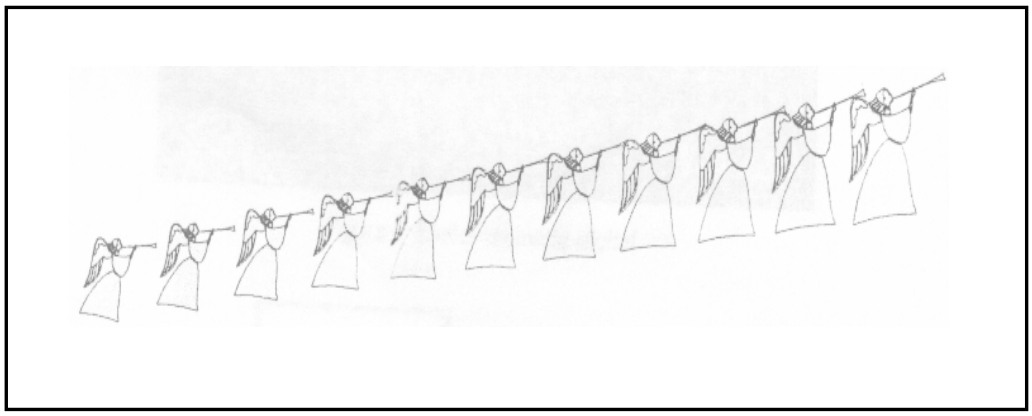
Fonte : [FIG1987]

Algumas dificuldades podem surgir, como por exemplo, a interpolação dos pontos de origem/destino, onde novos pontos têm que ser criados, a fusão de vários pontos de um quadro origem em um único ponto no quadro destino, para gerar um movimento sem “saltos”, ou ainda pelo fato de como a animação requer uma grande aceleração contínua do processo de interpolação, onde a habilidade de unir os movimentos se torna imprescindível. E por fim, uma outra grande dificuldade é processar estas imagens, de forma que pareçam uma animação natural onde o tempo é uma variável de muita importância a ser considerada ([WYV1989]).

A interpolação normalmente trabalha sob equações de transformações geométricas, utilizando algum modelo matemático para gerá-la, ou sob um modelo de parametrização, ou até sobre transformações de matrizes. Transformações de matrizes são as formas mais comuns para trabalhar-se com a interpolação, onde podemos associar o primeiro quadro-chave a matriz inicial e o segundo quadro-chave a matriz destino ([PEP1995], [WYV1989]).

Um exemplo de interpolação de quadros chaves, através de interpolação para gerar os quadros intermediários, pode ser claramente percebido na Figura 10, onde têm-se o primeiro anjo como quadro inicial e o último anjo como quadro final, ficando os anjos intermediários sendo construídos pela interpolação.

Figura 10 - Interpolação II - Sequência de Anjos



Fonte : [PAU2000]

Para a transformação de matrizes podemos usar vários métodos matemáticos. Neste trabalho usaremos o método de interpolação matemática pela equação paramétrica, conforme descrito no Capítulo 5.

5 DESENVOLVIMENTO DO PROTÓTIPO

Com base nos conceitos apresentados nos capítulos anteriores, tornou-se possível o desenvolvimento do protótipo de *software* que permite gerar animação de imagens. Neste capítulo serão abordados a especificação, a implementação e o funcionamento do protótipo.

O protótipo de software desenvolvido aborda a animação de imagens através do conceito de amostra mencionado nas diversas referências consultadas. Considera-se neste contexto amostra como sendo uma imagem digital armazenada em um arquivo *raster* no formato *bitmap* (.bmp), com nível de resolução alta o suficiente para o processo de animação.

O processo de animação utiliza-se de duas amostras de cada vez, e é feito em três etapas, sendo elas, (1) a etapa de edição dos nodos, (2) de triangularização dos nodos e por fim (3) a etapa de interpolação dos quadros. Na etapa de edição dos nodos são efetuadas as gerações de edição (`Novo`, `Alterar` e `Remover`) aos pontos a serem utilizados no processo de triangularização. Por fim, na etapa de interpolação dos quadros, escolhe-se o método de interpolação, gerando-se os quadros intermediários necessários à animação.

Após a geração da animação, pode-se então verificar como a primeira amostra se torna na segunda amostra, podendo inclusive acompanhar cada quadro interpolado e seu estado final. Pode-se também parametrizar a quantidade de quadros intermediários e o intervalo entre os quadros no momento da animação.

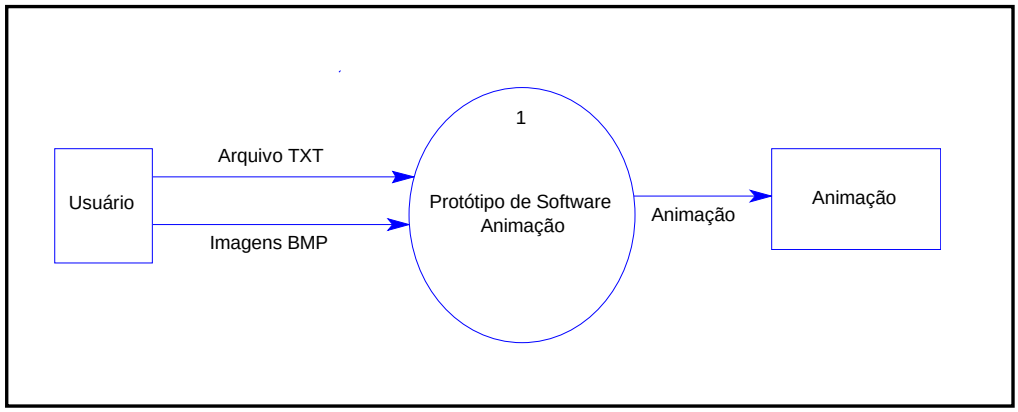
5.1 ESPECIFICAÇÃO DO PROTÓTIPO

Segundo [MEL1990], para o desenvolvimento de sistemas de informação, a prototipação representa uma boa solução para a maioria dos problemas. A metodologia de prototipação de sistemas utilizada neste protótipo é a **Prototipação Fundamental** ou **Básica**, originária do método Evolutivo, onde o produto final será o próprio sistema na sua forma mais aperfeiçoada. Conforme [MEL1990], o método evolutivo é usado na identificação do problema e na construção de modelos concretos, adaptados e corrigidos a medida que o usuário e o analista vão conhecendo a realidade e a solução do problema.

5.1.1 DIAGRAMA DE CONTEXTO

O Diagrama de Contexto é uma representação gráfica do sistema como um todo e os seus relacionamentos. Na Figura 11, tem-se como escopo deste protótipo as diferentes entradas e a visualização da animação pelo usuário.

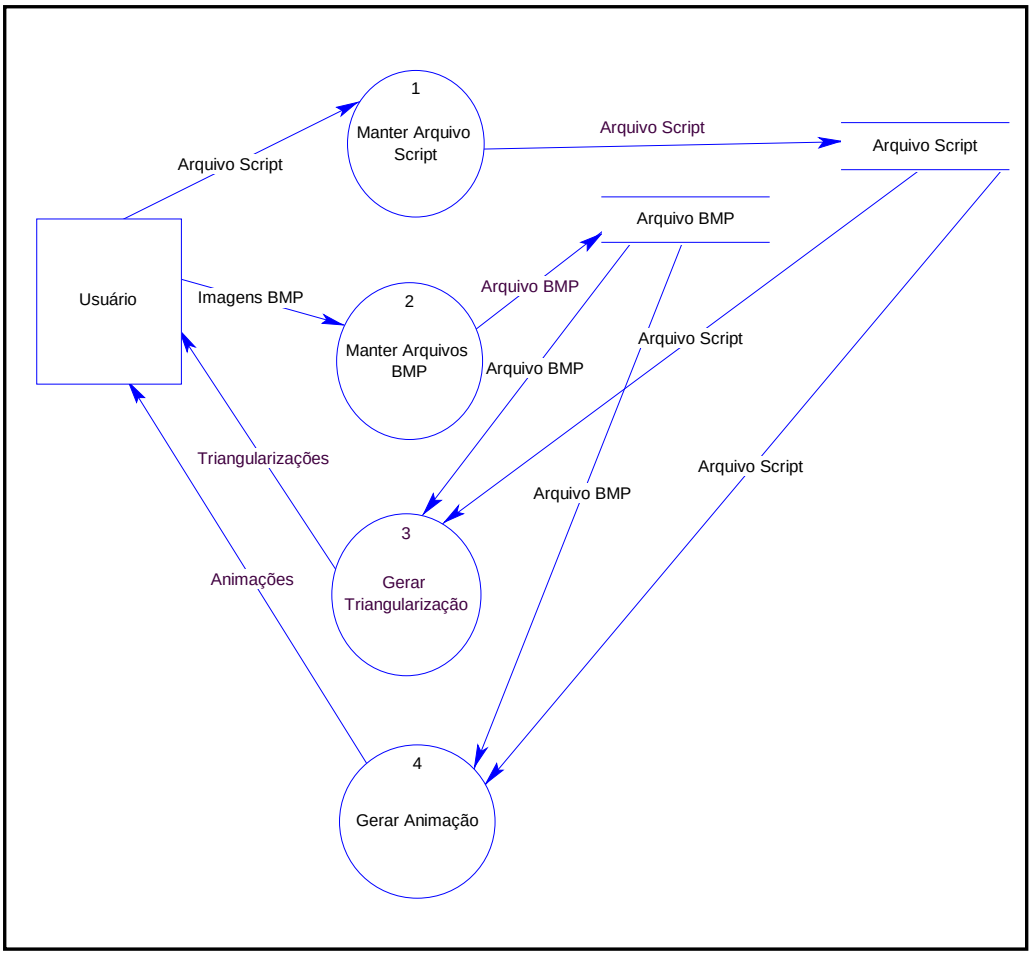
Figura 11 - Diagrama de Contexto



5.1.2 DIAGRAMA DE FLUXO DE DADOS

Na Figura 12, encontra-se o Diagrama de Fluxo de Dados de nível 1, que descreve o fluxo de informações e as transformações que são aplicadas à medida que os dados se movimentam da entrada para a saída.

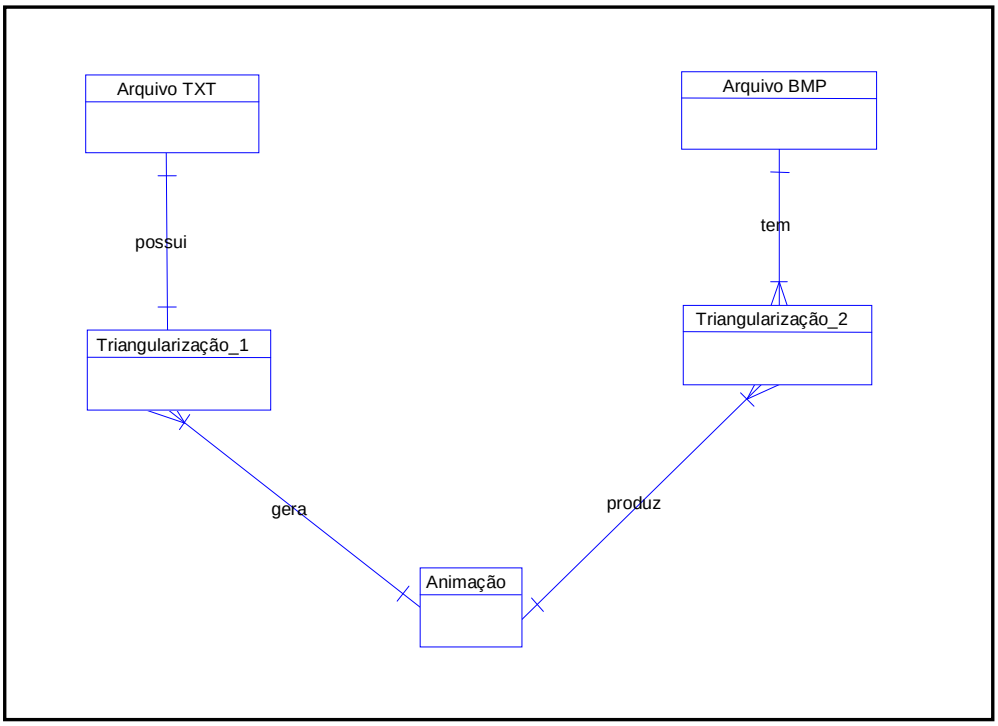
Figura 12 - DFD Diagrama de Fluxo de Dados



5.1.3 MER

Na Figura 13, encontra-se o Modelo de Entidade e Relacionamento.

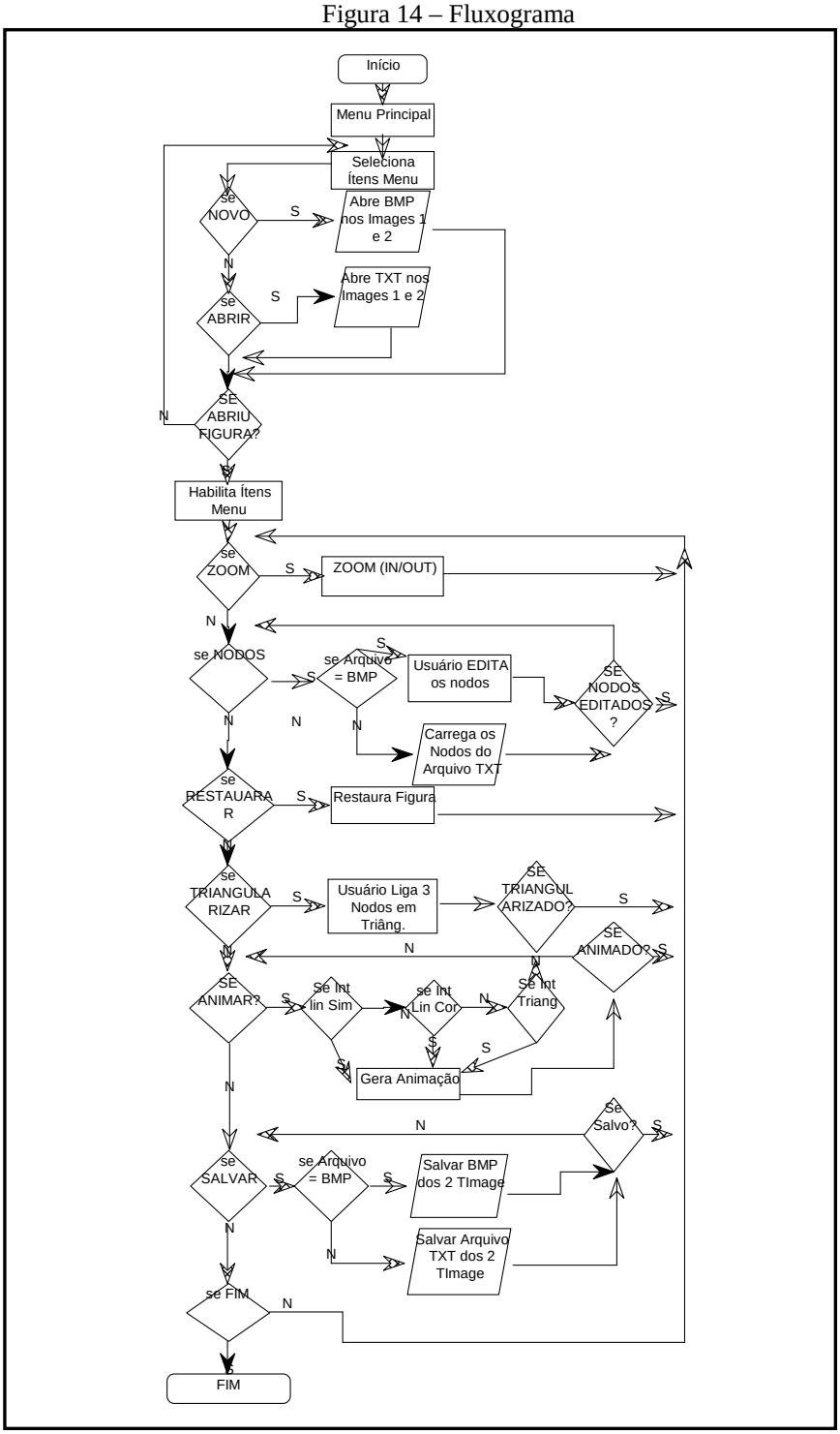
Figura 13 – MER



5.1.4 FLUXOGRAMA

Na figura abaixo, Figura 14, encontra-se o Fluxograma geral deste protótipo.

Figura 14 – Fluxograma



5.1.5 DICIONÁRIO DE DADOS

No quadro abaixo, Quadro1, encontra-se o Dicionário de Dados.

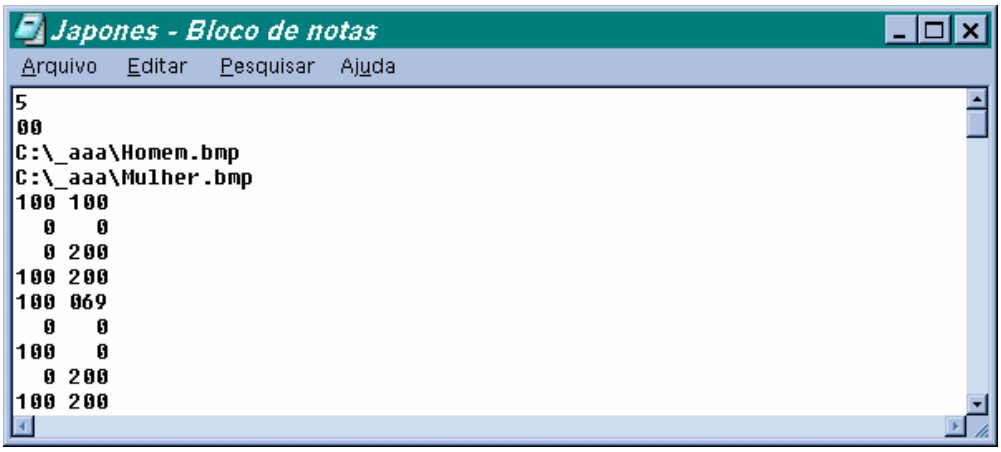
Quadro 1 - Dicionário de Dados

Nome	Tipo	Tamanho (em bytes)	Descrição
QtdePontos	Integer	4	Quantidade de pontos das figuras
QtdeTriang	Integer	4	Quantidade de triang das figuras
NomeFig1	String	256	Nome da primeira figura
NomeFig2	String	256	Nome da Segunda figura
PtosFig1x	TipoVetor	400	Pontos da primeira figura coordenada x
PtosFig1y	TipoVetor	400	Pontos da primeira figura coordenada y
PtosFig2x	TipoVetor	400	Pontos da Segunda figura coordenada x
PtosFig2y	TipoVetor	400	Pontos da Segunda figura coordenada y
TriangFig1	TipoVetor	400	Triângulos da primeira figura
TriangFig2	TipoVetor	400	Triângulos da segunda figura
TipoVetor	Array[1..100]	400	Coordenadas **x** e **y** das Figuras

5.1.6 ARQUIVOS DE SCRIPT

Os arquivos de *script* são de formato tipo texto, que são identificados com a extensão (*.txt). Cada linha de comando no *script* representa um dado relevante sobre as figuras que estão nele descritas, como mostra a figura abaixo (Figura 15).

Figura 15 - Arquivo de *Script* (Texto)



Este arquivo traz todos os dados sobre as imagens do arquivo texto, tendo na sua primeira linha a quantidade de nodos (pontos) das duas figuras; na segunda linha a quantidade de triângulos das figuras; seguidas pelas linhas três e quatro, que contém o nome dos arquivos de imagens. Nas linhas subseqüentes até o final do arquivo, estão as coordenadas dos nodos das figuras, sendo x e y, respectivamente, vindo por primeiro as coordenadas da primeira figura, seguida pelas coordenadas da segunda figura.

Como representação das linhas do *script* tem-se o seguinte exemplo, "50,90,Caminho1, Caminho2, Pontos1, Pontos2 ", representado na tabela abaixo (Tabela 1).

Tabela 1 - Exemplo de Comandos em Arquivo *Script*

Comandos Linhas *Script*	Significado dos Comandos
50	Quantidade total de pontos existentes nas figuras
- 90	Quantidade total de triângulos existentes nas figuras
Caminho1	Caminho para carregar primeira figura no `Timage` 1
Caminho2	Caminho para Carregar segunda figura no `Timage` 2
Pontos1	Sequência de coordenadas (**x,y**) dos postos da primeira figura
Pontos2	Sequência de coordenadas (**x,y**) dos postos da segunda figura

O arquivo *script*, pode ser alterado, acrescentando-se mais nodos (pontos), trabalhando diretamente com as coordenadas de cada figura, inclusive, mudando os seus valores.

5.2 IMPLEMENTAÇÃO DO PROTÓTIPO

Este protótipo teve a sua implementação no ambiente *Delphi* 4.0, onde utilizou-se programação estruturada composta praticamente por três *unit´s* distintas, sendo elas:

`Anima1`, que contém dados referente ao menu principal e aos controles das janelas que estão ativas; `Novo`, que contém todos os dados referentes às figuras que estão sendo trabalhadas, os arquivos texto e todas as outras variáveis para o controle do software, e por fim a *unit* `Tipos` onde contém todos os tipos de dados utilizados pelo protótipo.

Utilizou-se a propriedade `Timage` para a inserção da primeira e segunda figura. E a propriedade `TCanvas`, para a coloração dos nodos (pontos) em tonalidades de vermelho quando editados e a tonalidade de azul para quando somente selecionado um nodo.

Utilizou-se ainda a propriedade `TMemo`, que é um arquivo temporário para efetuar a leitura e escrita do arquivo *script,* guardar os seus dados nas respectivas variáveis.

Durante a execução do protótipo, a medida que o usuário vai interagindo, fazendo todos os passos necessários para gerar a animação, os arquivos utilizados podem ser atualizados.

Basicamente, o protótipo possui três fases principais, sendo elas: a fase de Edição dos Nodos, de Triangularização e por fim a fase de Animação, que serão descritas mais detalhadamente no capítulo de Implementação do Protótipo.

5.2.1 INTERPRETAÇÃO DOS COMANDOS

Se as imagens utilizadas para gerar a animação forem de arquivos *script´s,* estes precisam ser interpretados. No Anexo A, têm-se o algoritmo para a interpretação deste comando. Lê-se cada posição na linha até encontrar um caracter para leitura. Definiu-se para as coordenadas das figuras caracteres de três posições (000 até 999). Após cada linha lida do arquivo de *script,* é adicionada na propriedade `TMemo` para futuros acessos.

No Anexo B, têm-se o algoritmo inverso, ou seja, a leitura do `TMemo` para montar novamente o arquivo de *script.* Este processo é utilizado, para salvar as alterações efetuadas no arquivo, sendo acessado somente nestas duas situações, `Abrir` e `Salvar` arquivos.

5.2.2 DESENHO DAS FIGURAS

Para o desenho das Figuras nos `Images` respectivos, independente de ser arquivo *script* ou arquivo *bitmap*, precisa-se reestruturar as propriedades da `Form` ativa, para que suas dimensões sejam compatíveis com o tamanho das figuras. O redimensionamento destas propriedades pode ser visto no quadro abaixo (Quadro 2), onde são utilizadas de forma semelhante para as opções `Zoom In` e `Zoom Out`, que são mostradas no Quadro 3.

Quadro 2 - Algoritmo Para Redimensionar a `Form`

```
if ( FmNovo.LarguraFig1 <> FmNovo.LarguraFig2) or
     ( FmNovo.AlturaFig1  <> FmNovo.AlturaFig2 ) then
  begin
      ShowMessage ( ' Tamanho das Figuras Diferentes' + #13#10 +
                    ' A Primeira Figura Tem de Largura: '     +
                      FloatTostr(FmNovo.LarguraFig1) + #13#10 +
                    ' A Primeira Figura Tem de Altura:  '     +
                      FloatTostr(FmNovo.AlturaFig1)  + #13#10 +
                    ' A Segunda  Figura Tem de Largura: '     +
                      FloatTostr(FmNovo.LarguraFig2) + #13#10 +
                    ' A Segunda Figura Tem de Altura :  '     +
                      FloatTostr(FmNovo.AlturaFig2)  );
      FmNovo.Close;
  end
  else
  begin
     FmNovo.Width := trunc(FmNovo.LarguraFig1) +
                     trunc(FmNovo.LarguraFig2) + 45;
     FmNovo.Height:= trunc(FmNovo.AlturaFig1)  + 45;
     FmNovo.Image1.Picture.LoadFromFile(FmNovo.NomeFigura1);
     FmNovo.Image2.Picture.LoadFromFile(FmNovo.NomeFigura2);
     Fmnovo.Image1.Left:=0;
     FmNovo.Image1.top:=0;
     Fmnovo.Image2.top:=0;
     Fmnovo.Image2.left:= trunc(FmNovo.LarguraFig1) + 10;
     FmNovo.Show;
  end;   {  Para as Figuras Iguais }
```

Quadro 3 - Redimensionar Tela com o **Zoom In**

```
procedure TForm1.ZoomInClick(Sender: TObject);
begin
   ativo := TFmNovo(ActiveMDIChild);
   with ativo do begin
     with image1 do begin
       AutoSize:= False;
       Stretch := True;
       Height  := Height * 2;
       Width   := Width * 2;
       Picture.LoadFromFile(ativo.NomeFigura1);
     end;
     with image2 do begin
       AutoSize:= False;
       Stretch := True;
       Height  := Height * 2;            {Fator de escala}
       Width   := Width * 2;
       Left    := Trunc(image1.Height)+90 ;
       Picture.LoadFromFile(ativo.NomeFigura2);
     end;
     Height :=  Trunc(Image1.Height) + 30;
     Width  :=  Trunc(Image1.Width)  + Trunc(Image2.Width) + 90;
 end;
```

Como pode-se perceber, no caso de redimensionamento da tela com o `Zoom In` e `Zoom Out`, utilizou-se a constante 2 (dois), para a devida multiplicação e divisão dos valores constantes nas propriedades da `TForm`. Para tanto, foi utilizado o NDC (Coordenadas de Dispositivo Normalizado), neste processo ([FOL2000]).

5.2.3 GERANDO A ANIMAÇÃO

Para gerar a animação, foram implementados três métodos distintos, sendo eles: `Interpolação Linear Ponto-a-Ponto`, `Interpolação Linear Ponto-a-Ponto RGB` e `Interpolação Linear de Pontos Médios`.

Em todos os métodos, utilizou-se a equação matemática de Equação Paramétrica da Reta para o processo de interpolação dos nodos ([STE1987], [STE21987]). Esta função

(Equação 1), trabalha com pontos das duas figuras simultaneamente, tendo como resultado o ponto intermediário.

Equação 1 - Equação Paramétrica

$$Qi = A + (B - A) * \mu$$

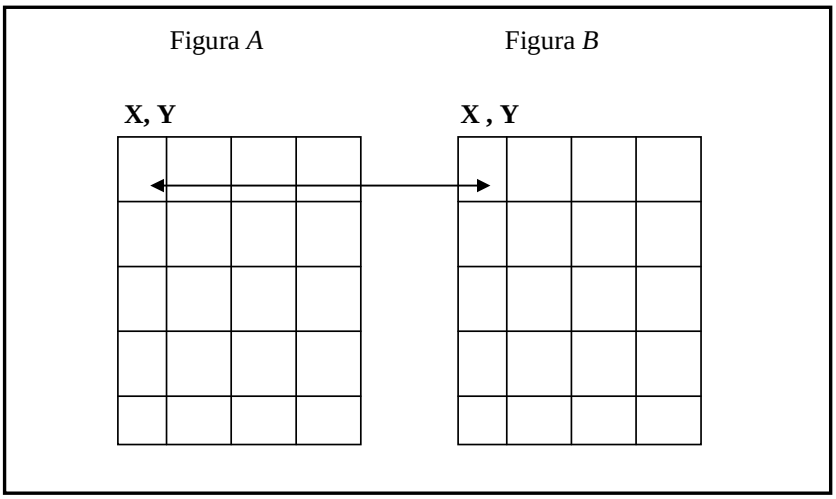
Sendo *A* o ponto referente a primeira figura, o *B* o ponto referente a segunda figura e $\mu \in (0,1)$.

Esta fórmula é utilizada para todos os pontos das duas figuras, calculando os pontos intermediários para cada quadro, e também para todas as cores dos *pixels*, o seu valor no estado intermediário.

5.2.3.1 INTERPOLAÇÃO LINEAR PONTO-A-PONTO

Para gerar a animação, para este método, têm-se a primeira figura cujos pontos nas coordenadas **x** e **y** correspondem aos pontos destas mesmas coordenadas na segunda figura, como mostra a Figura 16 abaixo.

Figura 16 - Coordenadas **x** e **y** Equivalentes para as duas figuras



Para este método, varre-se toda a Figura *A*, como mostrada acima, pegando o valor de cada *pixel* e interpolando-o através da equação paramétrica (Equação 1) com o valor do pixel na posição **x** e **y** correspondentes da Figura *B*. O valor do *pixel* resultante será armazenado para posterior visualização no seu quadro (μ) correspondente.

5.2.3.2 INTERPOLAÇÃO LINEAR PONTO-A-PONTO RGB

Para este método, o processo é praticamente o mesmo, com a diferença que em vez de se pegar o valor do *pixel*, particiona-o em três fragmentos distintos sendo eles R *(red)*, G *(green)* e o B *(blue)*, que contém cada um o valor correspondente das cores vermelho, verde e azul, sendo que o valor resultante dos três serve para referenciar a cor utilizada no *pixel*.

Para tanto, depois de obter o *pixel* da Figura *A* e *B* correspondente, particiona-se nestes três segmentos e faz-se a interpolação pela equação paramétrica (Equação 1) para cada um separadamente, e depois de ter o valor de cada um resultante da equação, monta-se então o *pixel* intermediário unindo novamente os três valores (RGB). Faz-se este processo para μ vezes.

5.2.3.3 INTERPOLAÇÃO LINEAR DE PONTOS MÉDIOS

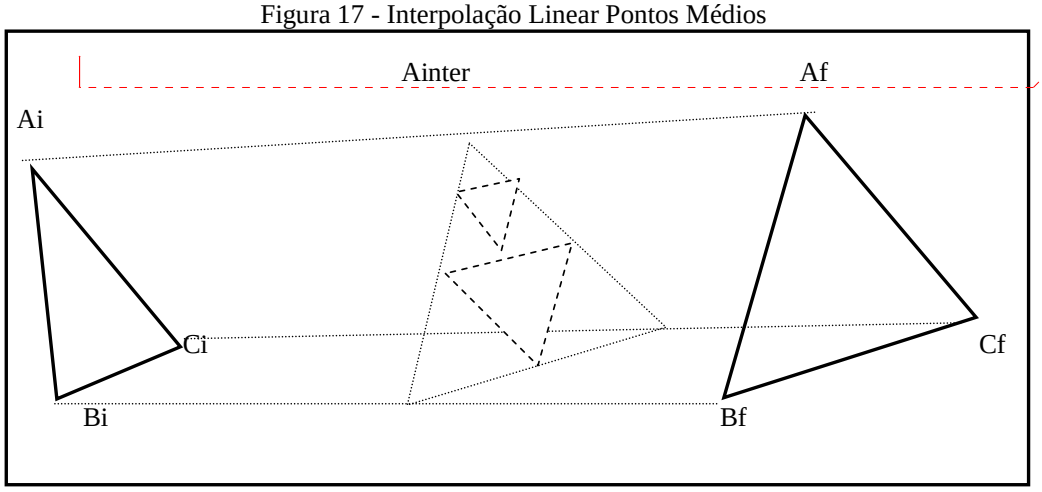
Já para gerar a animação por este método, mantém-se a mesma forma para obter os pontos correspondentes das Figuras *A e B*. No entanto, a forma como são calculados altera-se devido à utilização de outra equação; a de Pontos Médios (Equação 2), e também porque este método só pode ser executado depois que as figuras estiverem totalmente triangularizadas, pois utiliza-se os triângulos para tal.

Equação 2 - Equação de Pontos Médios

$$A\propto = (A1 + A2) / 2$$

Depois que a Figura estiver triangularizada, este método utiliza cada triângulo para efetuar a interpolação como mostra a Figura 17.

Figura 17 - Interpolação Linear Pontos Médios



[a2] Comentário:

Tem-se então:

- Ai,Bi,Ci = Triângulo inicial;
- Ainter, Binter, Cinter = Triângulo intermediário;
- Af,Bf,Cf = Triângulo final.

O primeiro passo então, é descobrir os pontos `Ai`, `Bi`, `Ci` e `Af`, `Bf` e `Cf`, respectivamente, que são os pontos do triângulo da primeira figura e da segunda figura. Desta feita calcula-se então, através da equação paramétrica os `Ainter`, `Binter` e `Cinter` que são os pontos intermediários, de acordo com a variação μ. Calculados os pontos intermediários, pega-se então o valor da cor dos *pixeis* das figuras Inicial e Final, e através da equação paramétrica (Equação 1), calcula-se a cor intermediária para a posição `Ainter`. Este processo é repetido para a coloração das posições de `Binter` e `Cinter`.

Depois de calculadas as posições e a cor para a posição intermediária, chama-se então uma rotina recursiva para dividir este triângulo intermediário em triângulos menores tendo como parâmetro a quantidade de triângulos indicada previamente pelo usuário.

Esta rotina divide então o Triângulo intermediário através da equação de Pontos médios (Equação 2). Ou seja, pega-se os valores de `Ainter` e `Binter`, calcula-se o ponto

médio, e têm-se como resultado o ponto `ABinter`. Depois de obter o resultado, calcula-se o ponto médio da cor dos *pixeis* de `Ainter` e `Binter`, achando a cor do *pixel* na posição `ABinter`. Repete-se este processo para calcular os ponto médios entre todos os demais pontos do triângulo e suas cores.

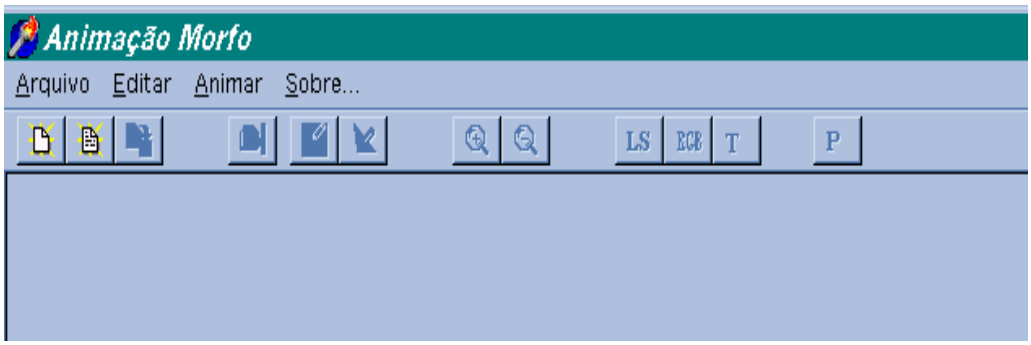
Por fim, tendo-se calculado todos os triângulos menores que pertencem ao triângulo Intermediário, gera-se então outro quadro até completar μ vezes, variando entre 0 e 1.

A próxima seção, apresentará o protótipo detalhadamente, com suas devidas telas e funções, mostrando sua utilização passo a passo desde a escolha dos arquivos ou figuras a serem utilizadas até a geração da animação final.

5.2.4 FUNCIONAMENTO DO PROTÓTIPO

Ao ser inicializado, o *software* apresenta ao usuário sua janela principal, ilustrada na Figura 18, na qual estão disponíveis, inicialmente, a opção `Novo`, a partir da qual o trabalho pode ser iniciado, abrindo imagens do tipo *Bitmap* (.bmp), para os dois `Timages`, e a opção `Abrir` onde pode-se abrir arquivos do tipo *script* (.txt).

Figura 18 – Janela Principal do *Software*

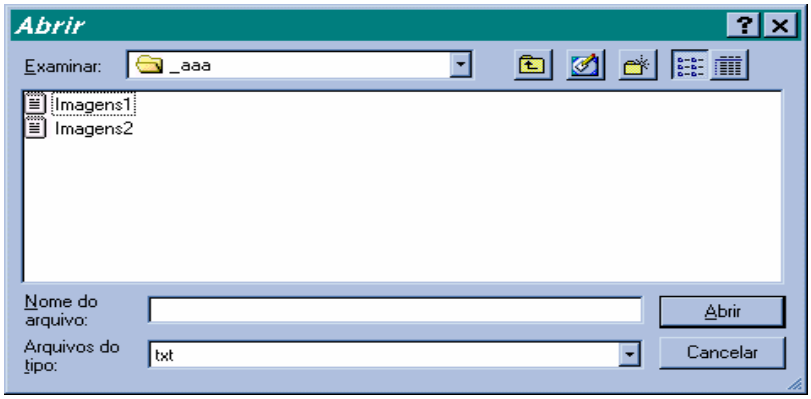


A janela apresentada inicialmente é simples, apresentando apenas os comandos básicos para sua operação, assemelha-se às barras de ferramentas encontradas em alguns *softwares* disponíveis no mercado.

A escolha da amostra da imagem a ser processada pode ser feita de duas formas distintas, a partir da opção `Abrir` ou pela opção `Novo.`

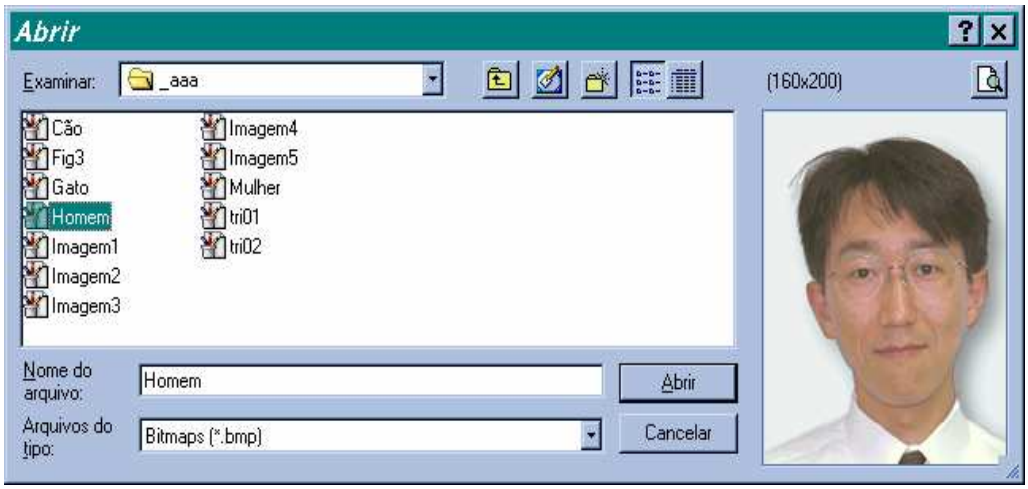
Neste ponto o protótipo de *software* apresenta uma caixa de diálogo para cada opção respectivamente. No entanto, para a opção `Abrir` o usuário informa qual o arquivo *script* (.txt) que deseja (Figura 19).

Figura 19 - Caixa de Diálogo Abrir



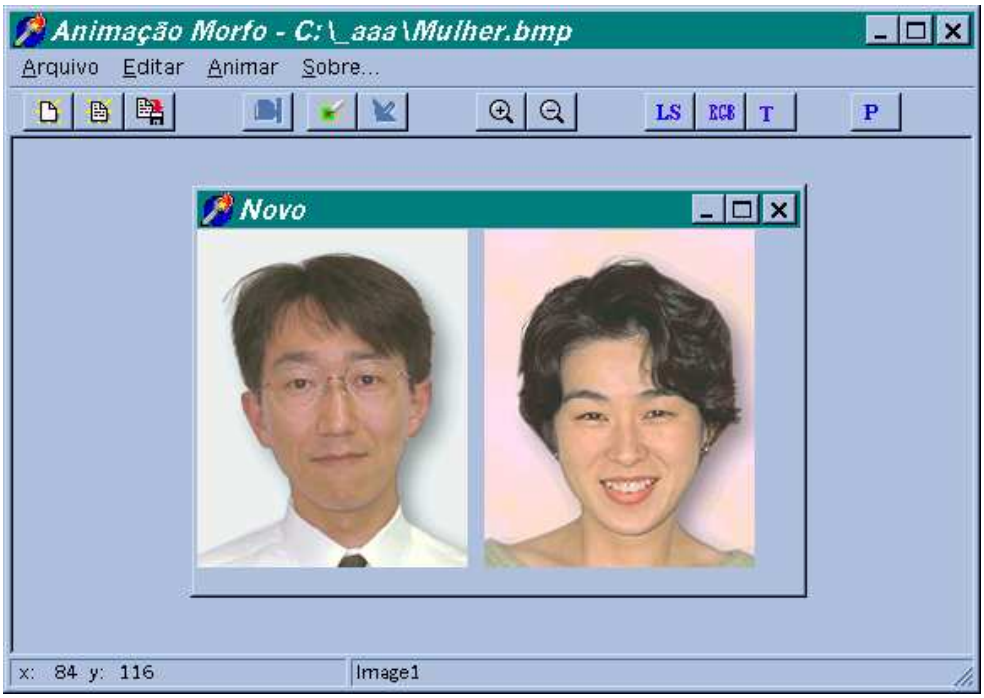
Já na caixa de diálogo para a opção `Novo`, o usuário escolhe quais as imagens do tipo *Bitmap* (.bmp) que deseja trabalhar, podendo visualizá-la na caixa de diálogo, em seu canto direito, como mostra a Figura 20.

Figura 20 - Caixa de Diálogo da Opção Novo



Depois que as figuras estão abertas, o sistema habilitará as seguintes funções da Janela Principal: `Salvar`, `Nodos`, `Zoom In` e `Zoom Out`, `Interpolaçâo Linear Ponto-a-Ponto`, `Interpolação Linear Ponto-a-Ponto RGB`, `Interpolação Linear de Pontos Médio` e `Parâmetros` conforme a Figura 21, abaixo.

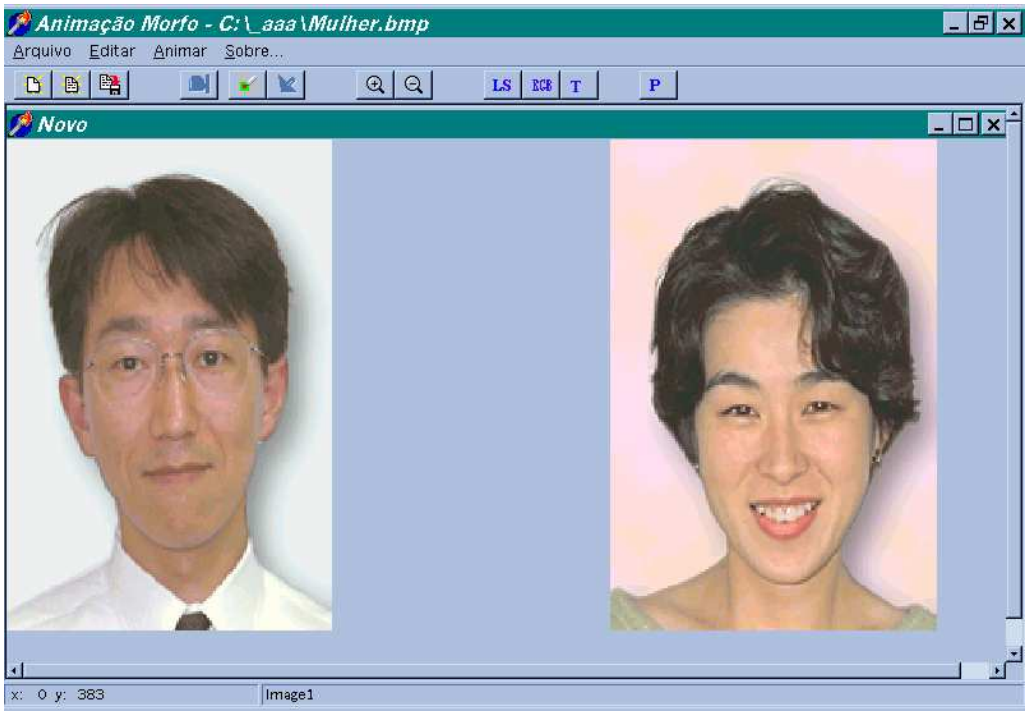
Figura 21 - Tela com as Teclas de Atalho Habilitadas



Como pode-se perceber na figura acima, quando posicionamos o cursor sobre uma das imagens, aparecem as coordenadas **x** e **y** do `Timage`, da respectiva figura, na parte inferior esquerda da Tela Principal. Llogo em seguida o nome do `Timage` na qual está o cursor. Como exemplo, têm-se na Figura 21 o cursor na posição **x** = 84 e **y** = 116 do `TImage` = *Image1*.

Visto que já foram habilitadas as opções de `Zoom In` e `Zoom Out`, o usuário tem a liberdade de utilizá-lo conforme necessário, ampliando e reduzindo quantas vezes assim o quiser, como mostra a Figura 22 abaixo.

Figura 22 - Figura Ampliada pelo *Zoom In*



O passo seguinte será a edição dos nodos nas figuras. Desta feita ocorrerá a habilitação das opções `Restaurar` e `Triangularização`, conforme mostrado na Figura 23. Uma vez escolhidas as figuras para trabalhar, deve-se pontilhá-las com os nodos. Este processo é chamado de edição dos nodos, tendo-se ainda, três ítens disponíveis, conforme discriminação abaixo:

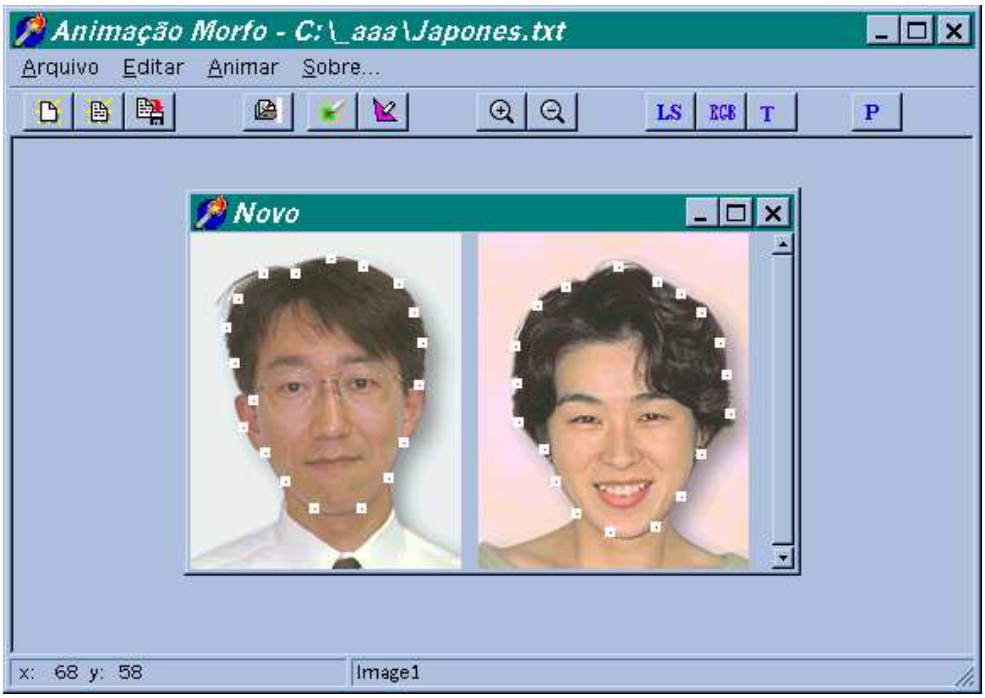
a) **Incluir Pontos**: para a opção `Abrir`, como já mencionado anteriormente, clicando-se sobre o botão `Nodos` a figura será "minada" pelos nodos que estão armazenados no arquivo de *Script.* Já para a opção `Novo` as figuras são abertas sem nodos armazenados, devendo então o usuário se encarregar de "miná-la" com os nodos. Em ambos os casos, pode-se então incluir novos pontos à figura com um *click* no botão esquerdo do *mouse* (limite de l00). Sempre que clicar com o botão

esquerdo do *mouse* em uma figura, corresponderá na mesma posição (**x** e **y**) da outra figura.

b) **Alterar Pontos**: com esta opção, o protótipo permite o livre manuseio destes nodos, podendo mudá-los de posição, atualizando automaticamente as novas coordenada. Para tanto, o usuário precisa clicar o botão direito do mouse, mantendo-o pressionado, arrastando-o para a nova posição desejada, soltando o botão em seguida.

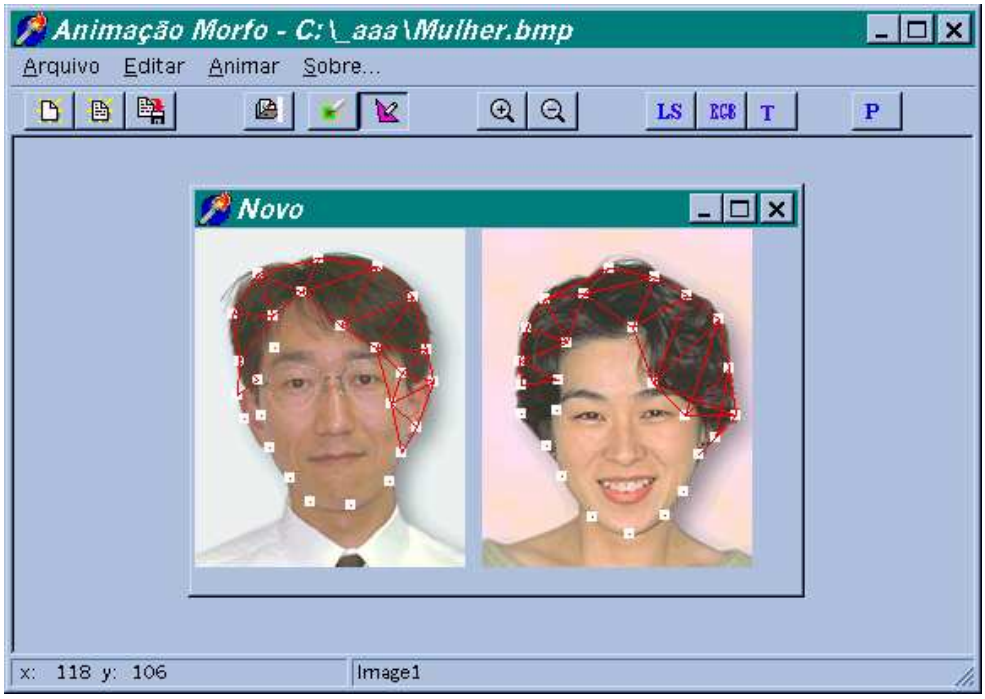
c) **Restaurar Pontos**: esta opção permite que os nodos sejam retirados, limpando as figuras exibidas na tela. Este nodos não são excluídos, ou seja, ficam armazenados em um vetor, e quando clica-se em `Nodos` os pontos são restaurados do vetor pontilhando novamente a figura.

Figura 23 - Edição dos Nodos



Uma vez os nodos editados, parte-se para `Triangularização`, ou seja, a unificação dos nodos em forma de triângulos. Para tanto é necessário clicar com o botão esquerdo do mouse na parte central de três nodos subseqüentes, triangularizando todos os nodos existentes na figura, conforme mostrado na Figura 24.

Figura 24 - Figura Triangularizada

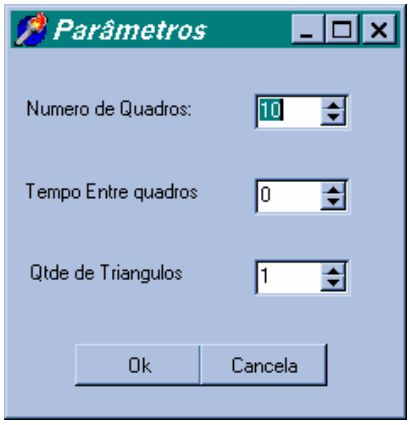


O próximo passo então, é escolher o método de animação. Pode-se escolher três métodos: `Interpolação Ponto-a-Ponto`, `Interpolação Ponto-a-Ponto RGB` e `Interpolação de Pontos Médios`.

Para tanto, deve-se escolher na opção `Parâmetros`, os parâmetros para o protótipo gerar a animação, ou seja, a quantidade de quadros intermediários entre a figura inicial e a figura final, que serão mostrados na tela, o intervalo de tempo entre um quadro e o outro,

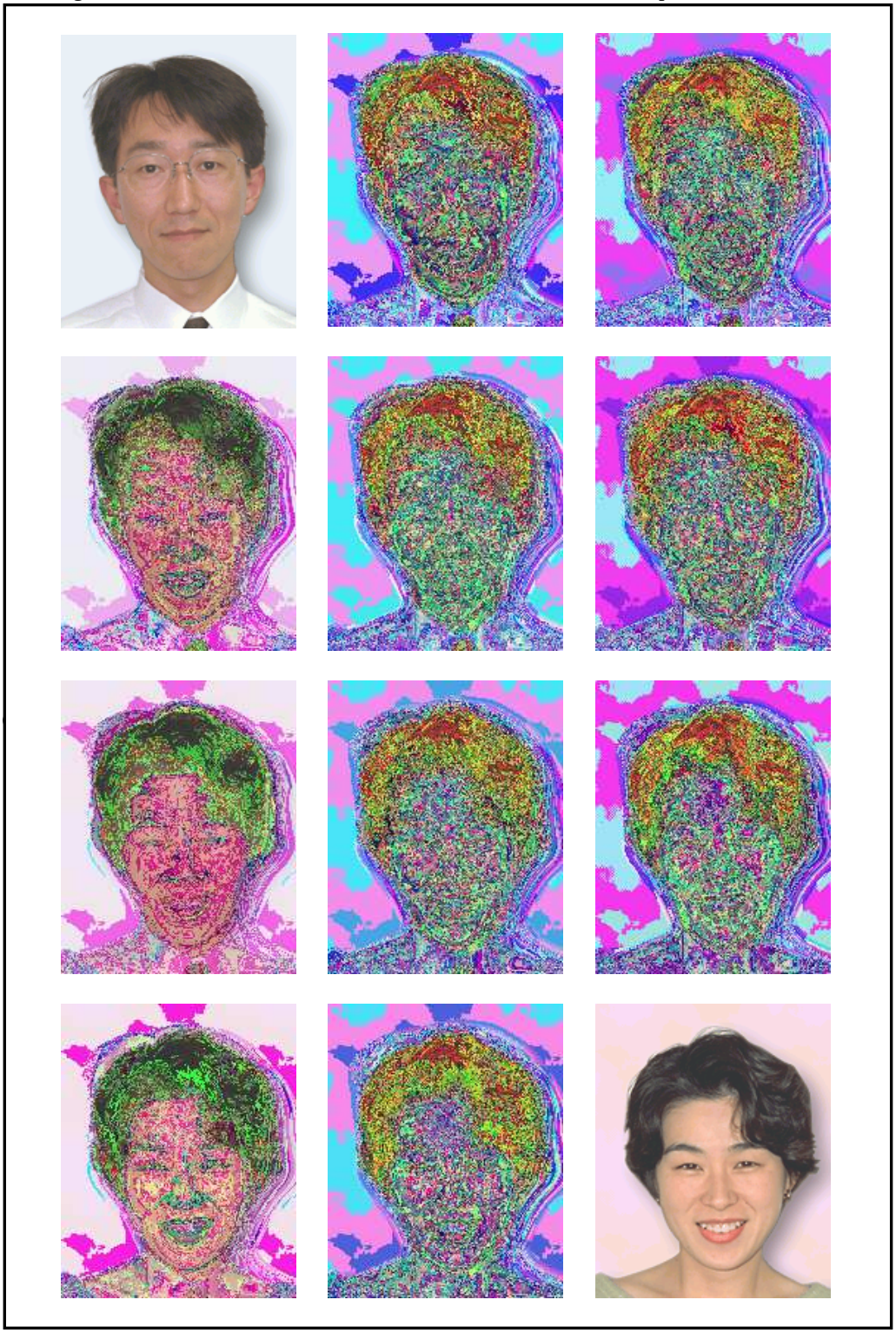
tendo como limite tempo de 5 segundos, e a quantidade de triângulos nos quais serão sub-divididos os triângulos feitos pelo usuário. Esta última opção somente será utilizada para o método de `Interpolação de Pontos Médios.` A tela `Parâmetros` está mostrada na Figura 25.

Figura 25 - Tela Parâmetros



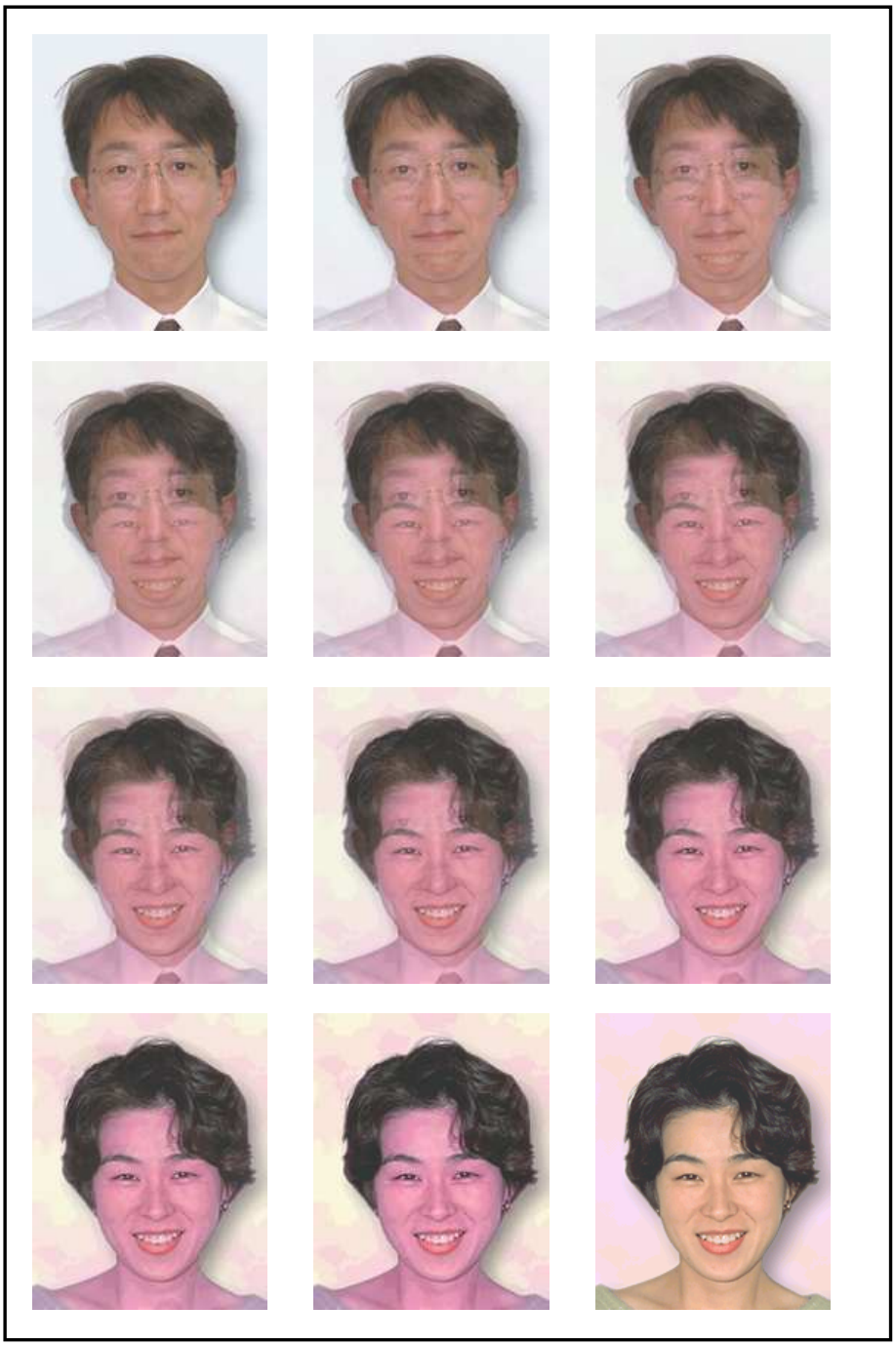
Depois de escolhido os parâmetros, pode-se então escolher o método que irá ser utilizado para a animação. Caso escolha o método pela animação de Interpolação Linear Ponto-a-Ponto, a Figura 26 mostra os quadros intermediários (escolhidos 12 quadros) gerados desde a figura inicial até a figura final.

Figura 26 - Quadros intermediários do Método Inter. Linear Simples Ponto-a-Ponto



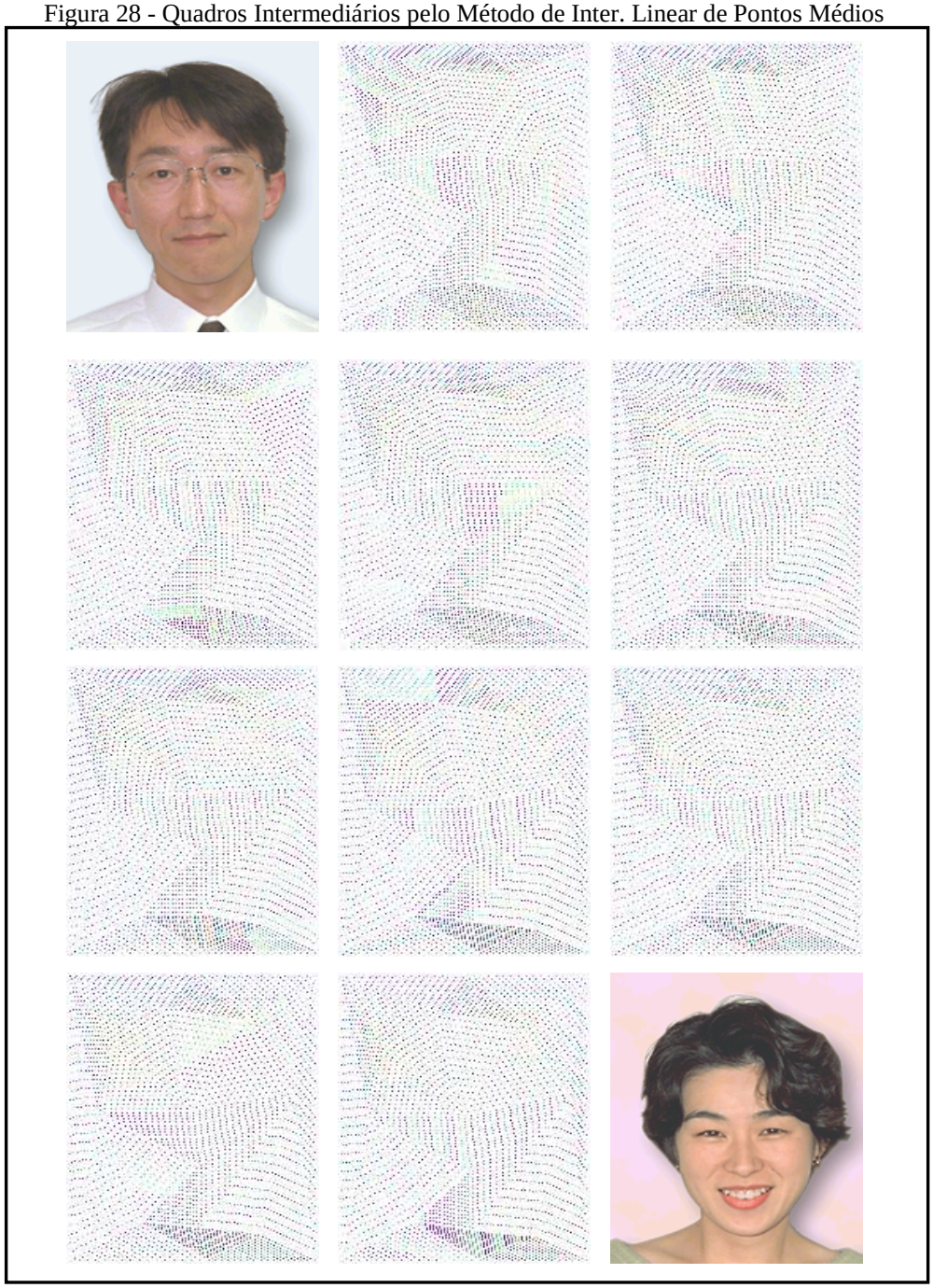
Caso o método escolhido seja o de Interpolação Linear Ponto-a-Ponto RGB, os quadros intermediários gerados podem ser vistos na Figura 27, levando em consideração que a quantidade de quadros geradas seja de 12 quadros.

Figura 27 - Quadros Intermediários pelo Método de Inter. Linear Ponto-a-Ponto RGB



Por fim, se o método escolhido for o de `Interpolação Linear de Pontos Médios`, os quadros intermediários gerados, através de triângulos, são mostrados na Figura 28, tendo-se como parâmetros: a quantidade de quadros igual a 12, o tempo entre os quadros igual a 1, e a quantidade de triângulos igual a 3 (nível de recursividade).

Figura 28 - Quadros Intermediários pelo Método de Inter. Linear de Pontos Médios



6 CONSIDERAÇÕES FINAIS

A seguir apresenta-se as considerações finais e sugestões para trabalhos futuros.

6.1 CONCLUSÃO

Pode-se verificar, através do aspecto visual, uma diferença expressiva no processo de interpolação dos quadros de animação, entre os três métodos propostos.

No primeiro método, `Interpolação Linear Ponto-a-Ponto`, pode-se verificar menor qualidade na representação da animação. Apesar de ser possível considerá-lo também o processo mais intuitivo, pois simplesmente utiliza uma interpolação ponto-a-ponto.

No segundo método, `Interpolação Linear Ponto-a-Ponto RGB`, apesar de ser uma extensão do primeiro método, obteve-se um resultado mais próximo de uma animação. Pode-se concluir que não basta simplesmente utilizar uma interpolação linear dos pontos, mas deve-se ter cuidado com a definição da cor dos *pixels* a serem interpolados na implementação.

Já no terceiro método, `Interpolação Linear de Pontos Médios`, o qual utiliza da interpolação linear não a nível de *pixel*, mas sim de triângulos com o uso de pontos médios, observou-se uma sobre carga do processamento da animação. Ao verificar-se a ordem de complexidade teórica e experimental dos métodos, pôde-se constatar que o aumento do nível de processamento é decorrente do uso utilizar de uma função recursiva. Em contrapartida, no que se refere ao aspecto visual, não atingiu-se a qualidade da animação gerada pelo método `Interpolação Linear Ponto-a-Ponto RGB`. Principalmente porque nos teste realizados usou-se um baixo grau de recursividade.

6.2 EXTENSÕES

Como extensão sugere-se para a área de pesquisa de trabalhos futuros, a exploração do uso de outros métodos para gerar a interpolação mencionados neste trabalho, bem como do tratamento de imagens armazenadas em arquivos vetoriais.

Outra extensão possível, seria tratar a diferença entre as *palete´s* das imagens a serem interpoladas, podería-se ainda estender o método proposto, `Interpolação Linear de Pontos Médios`, não utilizando mais a equação de Pontos Médios, mas somente a equação Paramétrica. Outra extensão seria o tratamento de dar cor a nível de interpolação do RGB, como foi utilizado no método `Interpolação Linear Ponto-a-Ponto RGB`, bem como aumentar o grau de recursividade.

Pode-se ainda implementar algoritmos de triangularização de Delaunay, sem a interferência do usuário neste processo, e, por fim, implementar uma janela para acompanhar os valores gerados (coordenadas, cor do *pixel*, etc) para as coordenadas **x** e **y** durante o processo de interpolação.

ANEXO A: INTERPRETAÇÃO DO ARQUIVO *SCRIPT*

```
indice :=1;
Form1.Caption:='Animação Morfo - ' + OpenDialog1.FileName;
Memo.Lines.LoadfromFile(OpenDialog1.FileName);
linha := Memo.Lines.Strings[0];
FmNovo.QtdePontos:= strtoint(copy(linha,1,2));
linha := Memo.Lines.Strings[1];
FmNovo.QtdeTriang:= strtoint(copy(linha,1,2));
linha := Memo.Lines.Strings[2];
FmNovo.NomeFigura1:=linha;
ImagemTemp:= TImage.Create(self);
ImagemTemp.AutoSize:=true;
ImagemTemp.Picture.LoadFromFile(linha);
FmNovo.LarguraFig1 := ImagemTemp.Width;
FmNovo.AlturaFig1  := ImagemTemp.Height;
linha := Memo.Lines.Strings[3];
FmNovo.NomeFigura2:=linha;
ImagemTemp.Picture.LoadFromFile(linha);
FmNovo.LarguraFig2 := ImagemTemp.Width;
FmNovo.AlturaFig2 := Imagemtemp.Height;
ImagemTemp.Free;
if ( FmNovo.LarguraFig1  <> FmNovo.LarguraFig2 ) or
   ( FmNovo.AlturaFig1   <> FmNovo.AlturaFig2 ) then
begin
    ShowMessage ( ' Tamanho das Figuras Diferentes' + #13#10 +
                  ' A Primeira Figura Tem de Largura:  ' +
                    floattostr(FmNovo.LarguraFig1) + #13#10 +
                  ' A Primeira Figura Tem de Altura:   ' +
                    floattostr(FmNovo.AlturaFig1)  + #13#10 +
                  ' A Segunda  Figura Tem de Largura:  ' +
                    floattostr(FmNovo.LarguraFig2) + #13#10 +
                  ' A Segunda Figura Tem de Altura :   ' +
                    floattostr(FmNovo.AlturaFig2)  );
end
else
begin
  FmNovo.Width:=  trunc(FmNovo.LarguraFig1) +
                 trunc(FmNovo.LarguraFig2)+  45;
  FmNovo.Height:= trunc(FmNovo.AlturaFig1)  + 30;
  FmNovo.Image1.Left:=0; FmNovo.Image1.top:=0;
  FmNovo.Image2.top:=0;
  FmNovo.Image2.left:= trunc(FmNovo.LarguraFig1) + 10;
  FmNovo.Image1.Picture.LoadFromFile(FmNovo.NomeFigura1);
  FmNovo.Image2.Picture.LoadFromFile(FmNovo.NomeFigura2);
  FmNovo.Show;
```

```
          for i:=4 to (FmNovo.QtdePontos + 3) do
          begin
             linha:= Memo.Lines.strings[i];
             xx:= strtoint(copy(linha,1,3));
             yy:= strtoint(copy(linha,5,3));

             FmNovo.PtoFig1x.Numeros[indice]:=xx;
             FmNovo.PtoFig1y.Numeros[indice]:=yy;
             inc(indice); end; { for 1 }
             indice := 1;
             for i:= Fmnovo.Qtdepontos+3 to (FmNovo.QtdePontos+
                                             Fmnovo.QtdePontos) do
             begin
              linha:= Memo.Lines.Strings[i];
              xx:= strtoint(copy(linha,1,3));
              yy:= strtoint(copy(linha,5,3));

              FmNovo.PtoFig2x.Numeros[indice]:=xx;
              FmNovo.PtoFig2y.Numeros[indice]:=yy;
              inc(indice);
             end;
           end;  { for  2 }
        end; { else  figuras iguais }
```

ANEXO B: SALVANDO UM ARQUIVO *SCRIPT*

```
procedure TForm1.SalvarClick(Sender: TObject);
var
indexador : integer;
begin

ativo := TFmNovo(ActiveMDIChild);
 if ativo.caixa then
 begin
  Statusbar1.SimpleText:= 'Salvando um Arquivo...';
  if Savedialog1.execute then
      begin
       Memo.lines.Clear;
       Memo.Lines.Add(Format('%3d',[FmNovo.QtdePontos]));
       Memo.Lines.Add(inttostr(fmNovo.QtdeTriang));
       Memo.Lines.Add(FmNovo.NomeFigura1);
       Memo.Lines.Add(FmNovo.NomeFigura2);
       for indexador := 1 to ativo.QtdePontos do
         Memo.Lines.Add(Format('%3d',[FmNovo.PtoFig1x.Numeros[indexador]])
         + ' ' + Format('%3d',[ativo.PtoFig1y.Numeros[indexador]]));
       for indexador := 1 to ativo.QtdePontos do
         Memo.Lines.Add(Format('%3d',[FmNovo.PtoFig2x.Numeros[indexador]])
         + ' ' +Format('%3d',[ativo.PtoFig2y.Numeros[indexador]]));
       for indexador := 1 to ativo.QtdeTriang do
       begin
         Memo.Lines.Add(Format('%3d',[FmNovo.TriFig1[1,indexador]]) +
         Format('%3d', [FmNovo.TriFig1[2,indexador]]) + Format('%3d',
         [FmNovo.TriFig1[3,indexador]]) +
         ' ' +(Format('%3d',[FmNovo.TriFig2[1,indexador]]) +
         Format('%3d', [FmNovo.TriFig2[2,indexador]]) + Format('%3d',
         [FmNovo.TriFig2[3,indexador]])));
       end;

       Memo.Lines.SaveToFile(SaveDialog1.FileName);
       ativo.caixa:= false;
     end;
 end
```

```
else
 begin
       Memo.lines.Clear;
       Memo.Lines.Add(Format('%3d',[FmNovo.QtdePontos]));
       Memo.Lines.Add(inttostr(fmNovo.QtdeTriang));
       Memo.Lines.Add(FmNovo.NomeFigura1);
       Memo.Lines.Add(FmNovo.NomeFigura2);
       for indexador := 1 to ativo.QtdePontos do
         Memo.Lines.Add(Format('%3d',[FmNovo.PtoFig1x.Numeros[indexador]])
         + ' ' + Format('%3d',[ativo.PtoFig1y.Numeros[indexador]]));
       for indexador := 1 to ativo.QtdePontos do
          Memo.Lines.Add(Format('%3d',[FmNovo.PtoFig2x.Numeros[indexador]])
         + ' ' +Format('%3d',[ativo.PtoFig2y.Numeros[indexador]]));

       for indexador := 1 to ativo.QtdeTriang do
       begin
         Memo.Lines.Add(Format('%3d',[FmNovo.TriFig1[1,indexador]]) +
         Format('%3d', [FmNovo.TriFig1[2,indexador]]) + Format('%3d',
         [FmNovo.TriFig1[3,indexador]]) +
         ' ' +(Format('%3d',[FmNovo.TriFig2[1,indexador]]) + Format('%3d',
         [FmNovo.TriFig2[2,indexador]]) + Format('%3d',
         [FmNovo.TriFig2[3,indexador]])));
       end;

       Memo.Lines.SaveToFile(SaveDialog1.FileName);
       ativo.caixa := false;
 end;
 Statusbar1.SimpleText:= ' ';
end;
```

REFERÊNCIAS BIBLIOGRÁFICAS

[BIN1994] BINDER, Fábio Vinícius. **Multimídia Animação Gráfica e Sons Utilizando Linguagem C.** São Paulo: Editora Érika Ltda,1994.

[CAN1996] CANTÚ, Marco. **Dominando o Delphi.** São Paulo: Makron Books do brasil Editora Ltda, 1996.

[CAN1998] CANTÚ, Marco. **Dominando o Delphi 3.** São Paulo: Makron Books do Brasil Editora Ltda, 1998.

[FIG1987] FIGUEIRAS, Lucia Vilela Leite; TORI, Romero; MASSOLA, Antonio marcos Aguirra; ARAKAKI, Reginaldo. **Fundamentos da Computação Gráfica**. São Paulo: LTC- Livros Técnicos e Científicos Editora S/A.,1987.

[PAU2000] PAULA FILHO, Wilson de Pádua. **Multimídia – Conceitos e Aplicações.** Rio de Janeiro : Editora LTC- Livros Técnicos e Científicos Editora S/A.,2000.

[FOL1993] FOLEY, James D. ***Computer Graphics: Principles and Pratice*.** Addison-Wesley, 1993.

[LOP1990] LOPES, Pedro Faria; GOMES, Mario Rui. ***Computer Animation'90*.** Tokyo: Springer-Verlag Tokyo, 1990.

[MAI1990] MAIOCCHI, Roberto; PERNICI, Barbara. ***Computer Animation'90*.** Tokyo: Springer-Verlag Tokyo, 1990.

[MEL1990] MELENDEZ, Rubem Filho. **Prototipação de Sistemas de Informações.** Rio de Janeiro : LTC – Livros Técnicos e Científicos Ed, 1990.

[MIL2000] MILBRATZ, Marlise Frotscher. **Protótipo para a Análise da Percepção do Movimento Aparente em Computação Gráfica**. Trabalho de Conclusão

do Curso de Bacharel em Ciências da Computação da Universidade Regional de Blumenau, 2000.

[MOR1995] MORRISON, Mike. **Mágicas da Computação Gráfica.** São Paulo: Berkeley,1995.

[PEP1995] PEPKE, Eric. ***Computer Visualization Graphics Thecniques for Scientifc and Engineering Analysis.*** Florida: CRC Press, 1995.

[SEI1987] STEINBRUCH, Alfredo; WINTERLE, Paulo. **Geometria Analítica**. São Paulo : McGraw-Hill, 1987.

[SEI1987] STEINBRUCH, Alfredo; WINTERLE, Paulo. **Álgebra Linear**. São Paulo : McGraw-Hill, 1987.

[WYV1989] WYVILL, Brian; WYVILL, Geof. **Computers in Art, Design and Animation.** New York: Springer-Verlag New York Inc., 1989.