ANIMAKER - Sistema de Animação Modelada

RODRIGO DE LOSINA SILVA
SILVIA DELGADO OLABARRIAGA

Universidade Federal do Rio Grande do Sul - UFRGS
Instituto de Informática - II
Curso de Pós-Graduação em Ciência da Computação - CPGCC
Caixa Postal 15064 - CEP 91591-970
Porto Alegre, RS, Brasil
rodrigo@inf.ufrgs.br
silvia@inf.ufrgs.br

Abstract. This paper describes Animaker, an animation system that generates wire-frame images, and allows the user to debug his animations in a fast way. Animaker uses LDR, a language to define animation scripts. LDR features and sintax are described here and some examples are presented.

1 Introdução

Uma das áreas de maior avanço dentro da computação gráfica é a animação modelada por computador, onde o computador armazena e exibe objetos que compõem um cenário tridimensional. Já foram desenvolvidos diversos sistemas de animação com base neste conceito, muitos deles com a prioridade de oferecer ao animador recursos sofisticados para geração de imagens e movimentos.

Entendemos, no entanto, que a forma do animador se comunicar com o sistema de animação é tão ou mais importante que os recursos que o sistema oferece para gerar animações, pois, da comunicação com o sistema dependerá a facilidade do animador em fazer um bom uso dos recursos oferecidos.

Esta questão ficou bastante clara durante a definição do sistema de animação ANIMAKER [Silva (1992)], um sistema de animação modelada desenvolvido na UFRGS. Durante a implementação do ANIMAKER optou-se por criar uma linguagem de definição de roteiros, batizada de LDR, visando permitir ao animador descrever suas animações numa linguagem tão compreensível quanto possível. Com isso, espera-se que o sistema possa ser utilizado por animadores não familiarizados com computadores e linguagens de programação, e não apenas por programadores. Foi descartada a hipótese de se desenvolver uma interface interativa para o sistema, devido as restrições de tempo envolvidas no projeto. Tal interface provavelmente fará parte de futuras versões do sistema.

O presente artigo descreve as características principais do sistema de animação ANIMAKER, e detalha a linguagem de descrição de roteiros LDR.

2 O Sistema ANIMAKER

O sistema ANIMAKER foi implementado como trabalho de conclusão de curso na graduação em Ciências de Computação da UFRGS, fazendo uso de conceitos e algoritmos, bem como partes de código fonte, implementados em um trabalho anterior, o sistema de animação PREVIEW [SCHMIDT (1992)]. ANIMAKER foi desenvolvido com a prioridade de ser um sistema tão genérico quanto possível, o que fez com que, em sua implementação, fosse considerada como prioritária a definição de padrões para os formatos de arquivos de entrada e saída.

O sistema foi implementado em estações SUN, em linguagem C, e faz uso do software de câmera sintética PIPE3D [OLABARRIAGA (1989)]. No presente momento, além de seu uso básico como ferramenta para depurar animações, o sistema também vem sendo utilizado para testes de diversos sistemas de animação que estão em desenvolvimento na UFRGS.

2.1 Objetivos do Sistema

O sistema ANIMAKER tem como objetivo básico a depuração de animações. Quando o animador define um roteiro de uma animação, dificilmente o resultado que o sistema de animação produzirá será exatamente o que ele espera e nem mesmo será, na maioria das vezes, um resultado aceitável. Porém, alterar o roteiro de uma animação depois dela já ter sido gerada por um sistema de animação foto-realístico é extremamente caro computacionalmente, já que tais sistemas consomem elevado tempo de máquina. Assim, definiu-se como objetivo do ANIMAKER per-

mitir visualizar de forma aproximada, e num curto espaço de tempo, como será a animação final.

No projeto do sistema ANIMAKER definiu-se que ele deveria permitir comunicação com o animador suficientemente clara para possibilitar seu uso por pessoas não familiarizadas com linguagens de programação.

Outro objetivo definido no projeto foi o de implementá-lo de forma que pudesse funcionar em conjunto com qualquer sistema de geração realística de imagens que viesse a ser desenvolvido na UFRGS. Evita-se, assim, a necessidade de revisar o sistema sempre que um novo sistema de foto-realismo é desenvolvido.

O sistema PREVIEW, no qual se baseou o ANIMAKER, tinha também como objetivo básico a integração de diferentes ferramentas desenvolvidas na UFRGS, para permitir usá-las em conjunto numa mesma animação. Tal objetivo foi incluído na definição do sistema ANIMAKER.

2.2 Funcionamento

Como pode ser visto na figura 1, a entrada do sistema é um arquivo contendo o roteiro de uma animação. Esse roteiro encontra-se escrito na linguagem LDR (Linguagem de Descrição de Roteiros), linguagem desenvolvida durante o projeto ANIMAKER, e que é discutida em detalhes na seção 3.

Lido o roteiro, ANIMAKER pode executar duas tarefas distintas, de acordo com as instruções que o usuário venha a fornecer pela interface do sistema. A primeira e essencial é a visualização da animação lida. Essa exibição é feita na forma de aramado ("wire-frame"), devido ao baixo custo de processamento associado. Uma janela de comandos (figura 2), composta de "ícones" e "sliders", permite que o usuário controle a exibição das imagens, exibindo uma de cada vez ou seqüências delas, controlando, neste caso, o número de quadros por segundo.

A segunda função oferecida pelo sistema ANIMAKER é a geração de arquivos contendo a descrição de cada quadro gerado a partir do roteiro. O objetivo desses arquivos é facilitar a tarefa de sistemas de geração de imagens foto-realísticos que funcionem em conjunto com ANIMAKER. Cada arquivo contém a descrição de um quadro (uma cena), e será lido por um sistema de geração de imagens para gerar aquela imagem da animação com realismo. A sintaxe dos arquivos de descrição de cenas, bem como decisões de projeto associadas, encontra-se descrita na seção seguinte.

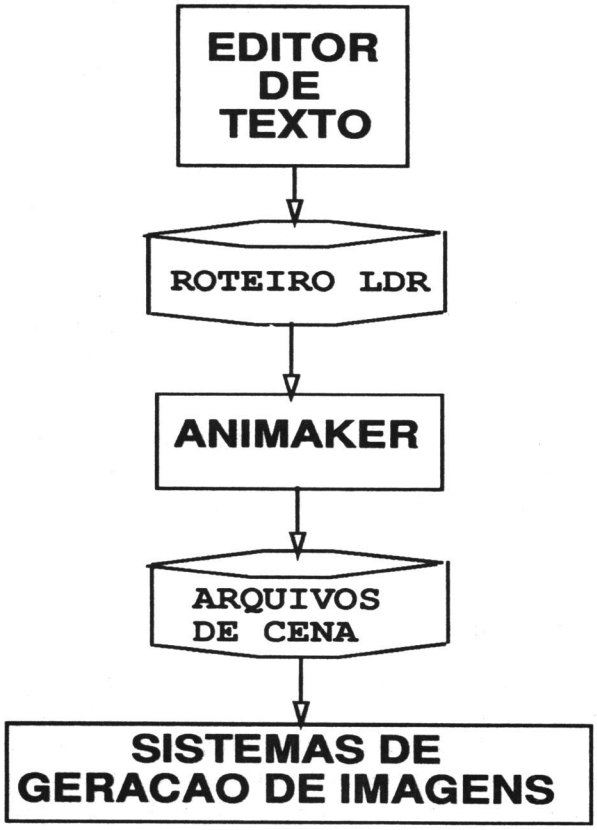

Figure 1: Entradas e saídas do sistema ANIMAKER

2.3 Arquivos de Cena

Na definição do sistema ANIMAKER, estabeleceu-se que ele não seria desenvolvido para funcionar em conjunto com um determinado sistema de geração de imagens realísticas. Definiu-se como um dos objetivos do projeto que os arquivos contendo a descrição das cenas pudessem ser usados por diferentes sistemas. Isto gerou um problema muito complexo, que era o de definir quais informações de realismo deveriam ser armazenadas nos arquivos de cena, já que diferentes sistemas de geração de imagens com rea-

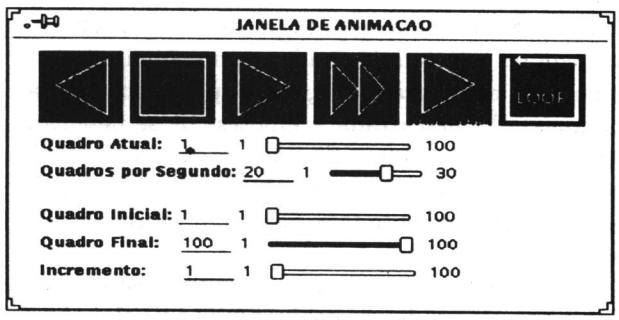

Figure 2: janela de animação

lismo necessitam de diferentes tipos de informações.

A solução encontrada foi permitir ao usuário incluir qualquer informação de realismo no roteiro de entrada do sistema, e gerar os arquivos de cena incluindo neles essas informações. Assim, cabe ao roteiro, e portanto ao animador, a tarefa de colocar as informações de realismo que são necessárias ao sistema de geração de imagens que se planeja usar. Isso tornou a linguagem LDR consideravelmente complexa e poderosa.

Os arquivos de cena precisariam conter todas as informações de realismo incluídas no roteiro de entrada, e sua sintaxe foi definida considerando-se esta característica.

Cada arquivo de cena inicia pela palavra-chave "cena", seguida pelo nome daquela cena (nome do roteiro de entrada seguido pelo número do quadro descrito na cena). O restante do arquivo de cena é composto pela lista de todas as entidades presentes naquela cena, incluindo "luzes", "superficies" e qualquer outro tipo de entidade que o roteiro defina. A definição de cada entidade inicia pelo seu nome, conforme definido no roteiro, seguido pela lista de todas os parâmetros que o roteiro associa aquele ator, com o valor assumido por eles naquele quadro da animação.

Assim, a sintaxe dos arquivos de cena permite a definição de qualquer tipo de entidade, contendo quaisquer parâmetros, sendo que tanto os nomes das entidades como os dos parâmetros podem ser definidos pelo roteiro de entrada.

A seguir é mostrado um exemplo de um arquivo de cena gerado pelo sistema ANIMAKER, seguido de uma explicação da cena que ele representa. A figura 3 contém a imagem correspondente, conforme exibida pelo sistema ANIMAKER.

```
cena Cubo2.1

camera
  posicao= 0.000000 0.000000 30.000000
  alvo= 0.000000 0.000000 0.000000
  foco= 90.000000

ator cubo.obt
  translacao= 0.000000 0.000000 0.000000
  rotacao= 60.000000 0.000000 0.000000
  escala= 1.000000 1.000000 1.000000
  transparencia= 0

luz
  posicao= 0.000000 0.000000 10.000000
  intensidade= 200
```

A cena tem três elementos, uma camera, um objeto com transparência, cuja forma encontra-se armazenada no arquivo "cubo.obt", e uma luz, definida por uma posição e uma intensidade. Para gerar tal cena, no próprio roteiro de entrada foi definido que os objetos possuem transparência e que existem entidades do tipo "luz", com posições e intensidades a elas associadas. As informações de realismo (iluminação, transparência), não são consideradas na exibição pelo sistema.

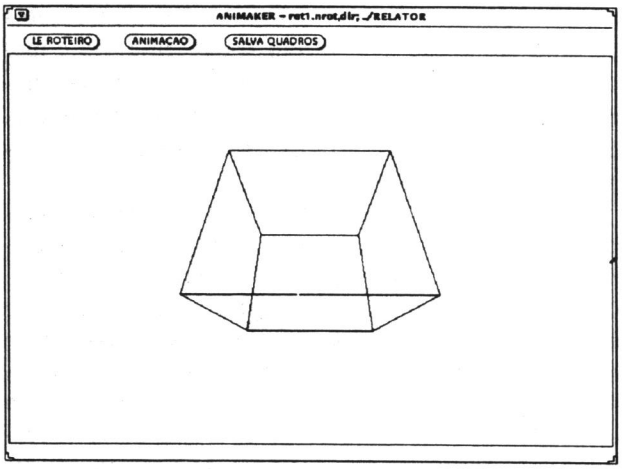

Figure 3: imagem da cena "Cubo2.1"

3 A Linguagem de Definição de Roteiros LDR

Para construir animações com o sistema ANIMAKER, o animador escreve um roteiro em linguagem LDR, armazenando num arquivo que será lido pelo sistema. Considerando-se a fundamental importância da linguagem na utilização do ANIMAKER, e o fato de seu uso se destinar a usuários não familiarizados com linguagens de programação, foram estabelecidas como prioridades em sua definicão a clareza e legibilidade.

Num primeiro momento, pensou-se que a definição do roteiro deveria ser voltada para o sistema ANIMAKER, por exemplo considerando-se apenas animações de imagens na forma de aramado, uma vez que esta é a forma de exibição corrente. Assim não seriam incluídas na linguagem LDR texturas e cores de objetos, grau de transparência e reflexão, iluminação, e outros características que não são representadas pelo mecanismo de exibição de imagens do ANIMAKER.

Percebeu-se, porém, que está visão era equivocada. Embora a visualização em aramado não exigisse informações de realismo, estas informações precisariam ser repassadas aos sistemas de geração de imagens, através dos arquivos de cena.

Optou-se, portanto, por permitir que no roteiro fossem incluídas informações de realismo, para que estas informações pudessem ser armazenadas na descrição das cenas, e portanto repassadas aos sistemas de geração de imagens. Naturalmente, apenas as informações da estrutura dos objetos seriam consideradas durante a exibição das cenas no ANIMAKER.

3.1 Atores

A linguagem LDR faz uso do conceito de atores. Atores são tradicionalmente definidos como os objetos de uma animação que sofrem transformações e movimentos ao longo do tempo. Cada ator tem, associadas a ele, informações significativas para a animação, como sua forma, posição, cor, transparência, etc, dependendo do sistema de animação em questão.

No sistema ANIMAKER, no que se refere à exibicão das animações, as únicas informações significativas são a forma, posição, escala e orientação dos objetos. Entretanto, os sistemas de geração de imagens realísticas que fizerem uso das descrições de cenas geradas pelo sistema precisarão de outras informações. A grande dificuldade associada a esta questão foi definir quais seriam as informações de realismo que deveriam ser incluídas no sistema, já que tais informações dependeriam do sistema de geração de imagens que viesse a ser usado para gerar uma animação. A seção 3.3 define melhor esta questão e descreve a solução adotada pela linguagem LDR para incluir informações de realismo nos atores.

Percebeu-se, no entanto, que definir informações de realismo dos objetos (cores, texturas, etc), não seria o suficiente para incorporar na linguagem todas as necessidades de sistemas de geração realística de imagens. Ocorre que determinadas informações da cena essencias a estes sistemas não estão associadas a objetos, em especial informações de iluminação.

A fim de solucionar de forma completa este problema, optou-se por ampliar o conceito de atores. Na linguagem LDR atores são, não apenas os objetos, mas tambem câmeras e quaisquer outras entidades da animação que o animador desejar incluir. O animador determina qual o tipo de cada ator que ele define, como pode ser visto no exemplo de roteiro ao final da seção, sendo a compreensão do significado de qualquer tipo novo de ator tarefa dos sistemas de geração de imagens foto-realísticas, pois o ANIMAKER simplesmente transmitirá essa informação aos arquivos de cena gerados, sem fazer qualquer consistência. Em resumo, o usuário pode criar um tipo de ator e denominá-lo "luz", que o sistema simplesmente transmitirá essa informação para os arquivos de cena.

Os atores definidos como objetos são exibidos no display, e o primeiro objeto definido como câmera é utilizado para as informações de câmera sintética da animação. Todos os outros atores que venham a ser definidos são ignorados durante a exibição no display pelo ANIMAKER, mas considerados na geração dos arquivos de cena para uso por sistemas de geração de imagens.

É importante perceber a metodologia de funcionamento desse sistema: a linguagem LDR aceita qualquer nome para um ator, e o sistema ANIMAKER simplesmente armazena em arquivos de cena cada quadro, sem se preocupar em determinar o significado dos diferentes tipos de atores. O sistema de geração de imagens realísticas que vier a fazer uso dos arquivos gerados é que sabera o que são "luzes", "superficies" ou outros atores que forem definidos no roteiro.

A seguir, encontra-se um exemplo da definição de três tipos de atores. O roteiro indica que há uma camera posicionada em (10,10,10), observando o ponto (0,0,0), com uma lente com ângulo de abertura de 90 graus. Existe ainda um objeto, cuja forma encontra-se armazenada no arquivo "cubo.obt", posicionado em (0,0,0), com escala igual a 1 nos três eixos, e orientação de (0,0,0). A cena é iluminada por uma fonte luminosa situada na posição (10,20,20) que possui uma intensidade definida como "200".

```
 .
 .
camera
 q=1 posicao= 10.0 10.0 10.0
     alvo= 0.0 0.0 0.0
     lente= 90.0
fim
objeto cubo.obt
 q=1 posicao= 0.0 0.0 0.0
     escala=1.0 1.0 1.0
     rotacao= 0.0 0.0 0.0
fim
luz
 q=1 posicao= 10.0 20.0 20.0
     intensidade= 200
fim
 .
 .
```

A exibição dessa imagem no sistema ANIMAKER levará apenas em consideração às informações fornecidas pelos dois primeiros atores, porém um sistema de geração realística de imagens que use a descrição de cena que o ANIMAKER gera consideraria a informação de iluminação.

3.2 Quadros-chaves

As transformações e movimentos dos atores ao longo do tempo são definidas fazendo uso do conceito de quadros-chaves. Quadros-chaves são quadros onde uma determinada característica do ator é definida explicitamente, e em geral representam o inicio ou fim de um determinado movimento ou transformação. A seguir é mostrado um trecho de roteiro onde são definidos quadros-chaves:

```
.
.
objeto cubo.obt
 q=1 posicao= 0.0 0.0 0.0
     escala= 1.0 1.0 1.0
 q=10 posicao= 10.0 10.0 2.0
 q=20 escala= 2.0 1.0 1.0
fim
.
.
```

No segmento de roteiro mostrado são definidos dois quadros-chaves para o parâmetro posição e dois para o parâmetro escala. Isso significa que naqueles quadros (1,10 e 20) os valores dos parâmetros serão os definidos.

Definir os quadros-chaves, porém, não é, por si só, suficiente para gerar animações. É necessário definir também a forma como serão gerados os quadros restantes para cada parâmetro. A forma principal utilizada pela linguagem LDR é o recurso da interpolação, gerando os quadros entre dois quadros-chaves calculando valores intermediários.

O sistema ANIMAKER possui três formas para cálculo de quadros intermediários: as interpolações linear, acelerada (baseada nas leis físicas dos movimentos uniformemente variados) e Bézier (baseada em curvas Bézier)[SCHMIDT (1992)]. A linguagem LDR, em sua presente definição, aceita apenas estas formas de interpolação, mas poderia facilmente ser (e provavelmente será no futuro) expandida para incluir qualquer outra função de interpolação. A seguir está um exemplo em LDR de interpolação, seguido de sua explicação. Na figura 4 encontra-se o resultado, conforme exibido no ANIMAKER.

```
.
.
objeto cubo.obt
/* definicao dos quadros-chaves*/
 q=1 posicao=0.0 0.0 0.0
     escala=1.0 1.0 1.0
     rotacao=0.0 0.0 0.0
 q=10 posicao= 0.0 200.0 0.0
 q=20 escala=2.0 1.0 1.0
     rotacao=90.0 0.0 0.0
/*definicao das interpolacoes*/
 interpola=1,10 bezier(-100.0 80.0 0.0 ,
                       100.0 120.0 0.0) posicao
 interpola=1,20 acelerado (1.0 ,
                           100.0) escala
 interpola=1,20 linear rotacao
fim
.
.
```

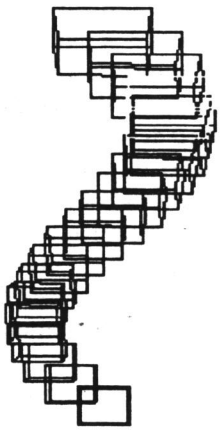

Figure 4: sobreposição das 20 imagens mostradas

O parametro "posicao" é interpolado entre os quadros 1 e 20 por uma curva Bézier, onde os seis parâmetros associados à função representam os dois pontos auxiliares necessários à definição da curva. O resultado final da função é um movimento curvilíneo do objeto entre os quadros 1 e 20.

O comando seguinte indica uma interpolação dos quadros 1 a 20 do parâmetro "escala", onde os dois valores associados à função indicam a velocidade inicial e final da interpolação, resultando numa variação de escala que começa lenta e vai acelerando. O último comando representa a interpolação linear do parâmetro "rotação", resultando numa rotação de 90 graus no eixo "X" com velocidade constante.

3.3 Variáveis Definidas no Roteiro

Conforme citado anteriormente, uma das questões mais complexas na especificação do roteiro foi definir como seriam implementadas as informações de realismo, uma vez que cada sistema de geração de imagens faz uso de informações de tipos diferentes. A solução encontrada foi permitir que o usuário indicasse no roteiro que informações de realismo são necessárias.

Para isso, o usuário conta com a possibilidade de definir variáveis. Uma variável é uma característica de realismo ligada a um ator, que possui um valor associado. Na prática, uma variável se comporta da mesma forma que os parâmetros pré-definidos dos objetos, como a posição e escala, possuindo também quadros-chaves que serão interpolados pelo sistema para gerar os quadros intermediários.

Por exemplo, supondo uma animação que será utilizada por um sistema de geração de imagens que trabalhe com objetos com cores definidas em RGB. O animador definirá três variáveis livres, por exemplo "R", "G" e "B", e indicará, nos atores do tipo objeto, quais os valores dos parâmetros nos quadros-chaves e quais as formas de interpolação que serão utilizadas. O sistema ANIMAKER não conhece o significado semântico destes parâmetros, e não os levará em consideração durante a exibição das imagens, porém repassará os valores correspondentes a eles nas descrições de quadros que forem armazenadas em arquivos (seção 2.3). Apenas o sistema de geração de imagens que fizer uso dos arquivos terá "consciência" do significado semântico de "R", "G" e "B".

O exemplo a seguir contém a definição de variáveis e exemplos de seu uso.

```
  .
  .
define transparencia real
/* 0=opaco, 100= transparente */
define R inteiro /* 0 a 255 */
define G inteiro
define B inteiro

objeto piramide.obt
 q=1 posicao= 0.0 0.0 0.0
     transparencia= 0.0
     R= 16
     G=100
     B=200
 q=10 transparencia= 99.5
     R= 20
     G=100
     B=0

 interpola=1,10 acelerado( 200, 1)
                transparencia
 interpola=1,10 linear R G B
fim
  .
  .
```

No exemplo, são incluídas no objeto informações de transparência e cor, que são interpoladas entre os quadros 1 e 10 da mesma forma que seriam quaisquer outros parâmetros. O resultado da animação resultante não é visível pelo sistema ANIMAKER, já que ele exibe apenas em aramado, porém a animação será perfeitamente gerada por sistemas capazes de considerar informações de transparência e cor pelos componentes RGB. O resultado é um objeto imóvel que torna-se transparente até quase desaparecer (99.5 % transparente), e que muda de cor diminuindo fortemente a componente azul e aumentando ligeiramente a componente vermelho.

3.4 Animações

Uma importante característica da linguagem LDR, e que amplia consideravelmente as potencialidades do sistema, é a possibilidade de definir animações. Animações se baseiam no conceito de funções de linguagens de programação.

Numa animação o animador define um ou mais movimentos ou transformações, da mesma forma que nos atores. O conjunto de operações definidos é aplicado a qualquer ator a partir do quadro em que este "chama" a animação.

Com isto, uma mesma operação (ou conjunto de operações) pode ser aplicada a diferentes atores, permitindo, por exemplo, que vários atores executem o mesmo movimento, ou que uma câmera siga todos os movimentos de um determinado ator. Este último caso é mostrado no exemplo a seguir:

```
  .
  .
/* definicao da animacao movimento*/
animacao movimento
 q=0 posicao= 0.0 0.0 0.0
    escala= 1.0 1.0 1.0
 q=10 posicao= 0.0 100.0 0.0
    escala= 2.0 1.0 1.0
 q=20 posicao= 0.0 100.0 50.0
    escala= 2.0 2.0 1.0
 interpola=1,10 linear posicao escala
 interpola=10,20 linear posicao escala
fim

camera
 q=1 posicao=0.0 0.0 0.0
     alvo= 0.0 200.0 25.0
 animacao=10,30 movimento
/* chama movimento a partir do quadro 10*/
fim
objeto cubo.obt
 animacao=1,21 movimento
/* chama movimento a partir do quadro 1*/
fim
```

No exemplo há a definição de uma animação, denominada "movimento", que define 21 quadros para os parâmetros "escala" e "posicao". A animação é chamada pelo objeto câmera a partir do quadro 10, significando que o parâmetro "posicao" da camera seguirá o movimento especificado até o quadro 30. O parâmetro "escala" é ignorado, pois não se aplica a atores do tipo câmera. Já o objeto executa o mesmo movimento a partir do quadro 1. Logo, a câmera repete, 9 quadros atrasada, os movimentos do objeto.

O exemplo salienta o fato de que as animações são independentes do tipo dos atores que fazem uso delas. No caso em que um determinado tipo de ator possui parâmetros diferentes dos definidos na animação, apenas aqueles parâmetros que têm correspondência na animação são considerados no processamento da mesma.

Embora seja um recurso poderoso, a animação poderia ser consideravelmente ampliada para permitir uma série de outros efeitos. Está prevista, para as próximas versões do sistema ANIMAKER e da linguagem LDR, a definição de animações relativas ao estado dos atores nos quadros em que as chamam. Atualmente não se pode definir animações que, por exemplo, executem movimentos dos atores a partir de sua posição atual, mas sim sempre de uma posição absoluta.

3.5 Loop

Outra importante característica da linguagem LDR é a possibilidade de definir "loops", que nada mais são do que comandos de iteração similares aos das linguagens de programação, embora naturalmente mais simples.

A idéia do "loop" é permitir a repetição por um número determinado de vezes de uma mesma operação. Assim, pode-se definir um objeto que ande em circulos do início ao fim da animação, que se dilate e contraia, mudando repetidas vezes sua escala, ou que mude qualquer outra de suas características por várias vezes. O mesmo se aplica à câmera, que pode, por exemplo, circular diversas vezes em torno do cenário, ou se aproximar e se afastar de um objeto a intervalos regulares.

Através do uso em conjunto com animações foi ampliado consideravelmente o conceito de "loops": um "loop" de uma animação resulta na repetição de um conjunto de operações. Assim, a iteração envolve diferentes operações sobre diferentes parâmetros, podendo inclusive conter novos "loops".

Portanto, o comando iterativo "loop" foi definido como a repetição de qualquer transformação ou conjunto de transformações de um ator ou da câmera, entendendo-se por transformações não apenas as interpolações, mas também outros comandos de "loop", conforme mostrado no exemplo abaixo.

```
 .
 .
animacao gira
 q=0 rotacao= 0.0 0.0 0.0
 q=10 rotacao= 360.0 0.0 0.0
 interpola=0,10 linear rotacao
fim

animacao move_e_gira
 q=0 posicao= 0.0 0.0 0.0
 q=100 posicao= 100.0 0.0 0.0
 interpola=0,100 linear posicao
 animacao=0,10 gira loop=10
 /* repete 10 vezes a animacao gira */
fim
 .
 .
objeto cubo.obt
 animacao=1,100 move_e_gira loop=5
 /* repete 5 vezes a animacao move_e_gira */
fim
 .
 .
```

No exemplo, a animacao "move_e_gira" define um movimento e chama a animacao "gira" com um valor de iteracao definido como 10, significando que a rotação definida em "gira" será repetida por 10 vezes, até o quadro 100 (10*10). Da mesma forma, o ator repete a animação "move_e_gira" por 5 vezes.

Não passou despercebido o fato de que o conceito ainda poderia ser mais expandido, incluindo, por exemplo, loops relativos às posições em que câmera e atores se encontrassem em cada iteração e mesmo loops que envolvessem mais de um ator. Futuras ampliações da linguagem oferecerão novos recursos nesse sentido.

3.6 Leitura de Valores

O sistema PREVIEW[SCHMIDT (1992)], no qual se baseou ANIMAKER, possuía como um de seus objetivos a integração de diversas ferramentas desenvolvidas na UFRGS, ou seja, a utilização de sistemas que geram movimentos baseados em leis físicas [LEMOS (1992)], sistemas de geração de movimentos de objetos articulados [MUSSE (1992)], entre outros, para a definição das animações. Esse objetivo foi mantido no projeto ANIMAKER, e, consequentemente, na linguagem LDR.

A forma como LDR se comunica com tais sistemas é através de arquivos. Os sistemas citados armazenam em arquivos os resultados que geraram.

Em LDR o animador pode definir que a informação de um parâmetro qualquer (embora até o momento, apenas o parâmetro posição tenha sido utilizado na prática) em um ou mais quadros é lida de um determinado arquivo. O exemplo a seguir demonstra a forma como isso é definido em LDR.

```
  .
  .
objeto cubo.obt
 leitura=1,20 posicao arq_pos.fis
 leitura=1,10 escala arq_esc.prt
fim
  .
  .
```

No exemplo, os valores dos quadros 1 a 10 do parâmetro "escala" são os valores armazenados nas dez primeiras linhas do arquivo "arq_esc.prt", enquanto que o parâmetro "posicao" recebe os valores armazenados nas vinte primeiras linhas de "arq_pos.fis", que poderia, por exemplo, ter sido gerado por um sistema de simulação de movimentos físicos.

3.7 Mudanças de Forma

Em sua presente versão, LDR não permite que se defina diretamente as características estruturas dos objetos, mas apenas que se indique o arquivo que contém a forma dos mesmos. Porém, para permitir a comunicação com sistemas de interpolação de formas de objetos [OLIVEIRA (1992)], LDR permite que se faça uso de um recurso já definido no sistema PREVIEW: a troca, a cada quadro, do arquivo de descrição da forma do objeto.

Indica-se, no roteiro, um arquivo que contém uma lista de arquivos de formas de objetos, e cada um dos arquivos será considerado na exibição de cada quadro da animação. A figura 5 contém uma imagem de mudança de forma. Segue um exemplo em LDR, e a listagem do arquivo utilizado.

```
  .
  .
objeto cubo.obt
 q=1 posicao= 0.0 0.0 0.0
 q=4 posicao= 50.0 0.0 0.0
 interpola=1,4 linear posicao
 forma=1,4 lista.frm
fim
  .
  .
```

Arquivo "lista.frm":

```
cubo.obt
cubo1.obt
cubo2.obt
octaedro.obt
```

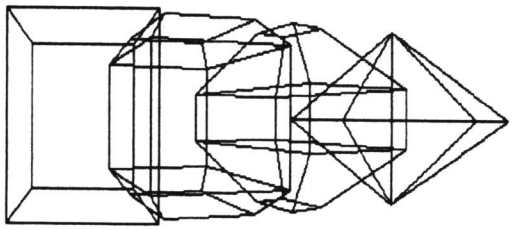

Figure 5: sobreposição de imagens de mudança de forma

No exemplo, para cada um dos quatro primeiros quadros, o objeto possuirá uma estrutura definida por um dos quatro arquivos da lista armazenada em "lista.frm".

4 Conclusões

Um dos objetivos do sistema ANIMAKER era possibilitar o uso do sistema por pessoas que não estivessem familiarizadas com linguagens de programação. Entendemos que a criação da linguagem LDR é um importante passo neste sentido, oferecendo ao usuário recursos poderosos de forma relativamente simples e legível.

Cabe ressaltar, no entanto, que tal objetivo só poderá ser plenamente atingido quando do desenvolvimento de uma interface interativa para a definição das animações, pois, ainda que não se compare, em termos de complexidade, as linguagens de programação de propósitos gerais, LDR não é uma forma de definir animações muito natural para pessoas não familiarizadas com programação.

Através de uma interface interativa, o usuário poderá definir quadros-chaves, interpolações, transformações de forma de objetos e demais recursos que o sistema oferece indicando por "mouse", na própria tela, cada ação a ser introduzida no roteiro, e observando o resultado imediatamente. Com isso obtem-se uma comunicação homem-máquina mais amigável ao usuário.

Apesar das óbvias vantagens de uma interface interativa em relação a um arquivo contendo o roteiro, optou-se, na definição da primeira versão do

sistema ANIMAKER, pelo definição da linguagem LDR. O principal argumento para esta decisão foi a restrição de tempo associada ao projeto (cerca de quatro meses). O desenvolvimento de uma interface interativa foi deixado para uma segunda versão do sistema.

Tanto a linguagem LDR quanto o sistema ANIMAKER ainda serão ampliados no futuro, pois existem diversos recursos importantes para a geração de animações que ainda não foram incluídos na presente versão. Uma das próximas ampliações será a inclusão de hierarquia de objetos, permitindo a definição de animações de corpos articulados.

Outra ampliação do sistema a ser efetuada é a inclusão de novos recursos de interpolação. Tanto a sintaxe da linguagem LDR quanto a estrutura interna do sistema ANIMAKER permitem que novas funções sejam facilmente incluídas.

Outro objetivo considerado como prioritário na definição do sistema foi possibilitar seu uso em conjunto com qualquer sistema de geração de imagens realistícas. Várias seções do artigo descrevem recursos da linguagem LDR que permitiram que esse objetivo fosse plenamente alcançado. A linguagem LDR é perfeitamente capaz de definir quase qualquer tipo de informação que venha a ser necessária para qualquer sistema de geração de imagens.

Considerando que os objetivos supra-citados foram atingidos e que o sistema já vem sendo usado pelo grupo de computação gráfica da UFRGS, entendemos que o sistema ANIMAKER atinge o objetivo básico de ser um sistema eficiente de depuração de animações. Não é, entretanto, um sistema fechado e completo, existindo muitas ampliações, algumas citadas acima, que ainda precisam ser implementadas.

5 Referências

LEMOS, Robson Rodrigues. **Controle de movimentos em animação: teoria e aplicação experimental**. Porto Alegre: CPGCC da UFRGS, 1992. Dissertação de mestrado em andamento.

MUSSE, Soraia Raupp. **Um estudo sobre animação computadorizada de objetos rígidos articulados**. Porto Alegre: CPGCC da UFRGS, 1992. Dissertação de mestrado em andamento.

OLIVEIRA, Manuel Menezes. **Algoritmo interpolador de formas entre objetos modelados por superfícies spline**. Porto Alegre: CPGCC da UFRGS, 1992. 136p. Dissertação de mestrado.

OLABARRIAGA, Sílvia Delgado **Relatório sobre o PIPE3D:** Pipeline para Visualização Tri-dimensional. Lisboa: INESC, 1989.

SCHMIDT, Ana Elisa; MUSSE, Soraia Raupp. **PREVIEW:** Sistema de animação. Porto Alegre: CPGCC da UFRGS, 1992.

SILVA, Rodrigo de Losina. **ANIMAKER:** Sistema de animação. Porto Alegre: UFRGS, 1992.