

Morph

Giovani Facchini
giovanif@turing.unisinos.br

14th June 2005

1 Problema

O problema consiste em criar uma ferramenta capaz de ler arquivos de descrição de objetos (formato obj) e conseguir, através de um algoritmo, transformar um objeto em outro através de *morphing*. Esse morph deve ser feito de maneira com que a *transformação* de um objeto em outro, pareça uma animação do deslocamento das faces.

2 Ferramenta

Para implementação do trabalho utilizou-se uma implementação de OpenGL portada para a linguagem de programação JAVA. Esta chama-se JoGL. Este porte do OpenGL tem vários problemas quanto a atualização da imagem e controle dos eventos. Outro problema se refere a maneira com que exemplos e alguns programas utilizam a biblioteca e tratam eventos, pois é utilizado um 'animador' que fica constantemente atualizando a janela. Isso Demanda um custo altíssimo de processamento e ocupação de uma thread.

Para gerenciamento de janelas e eventos foi utilizada a biblioteca Swing que cuida de eventos como os das janelas, mouse e de teclado. A swing é, hoje, o pacote mais usado em Java para implementação de interfaces por possuir componentes mais leves e flexíveis que a AWT.

3 Implementação

A implementação consiste de, basicamente, duas partes:

- Algoritmo de Morph
- Interface com o usuário

3.1 Morph

Para tratar a parte do algoritmo de morph implementou-se uma classe chamada Morph.java que encontra-se dentro do pacote Morph. Essa classe é responsável por toda a lógica do *morphing*, desde a criação das estruturas auxiliares, associação de faces e vértices até a criação de todos os objetos apresentados na animação final.

O algoritmo implementado associa as faces de maneira seqüencial de leitura do arquivo obj, ou seja, face N do objeto 1 com face N do objeto 2. Se um dos objetos tem maior número de faces, faces do objeto com menor número de faces são duplicadas até que os dois objetos tenham o mesmo número de faces. Análoga a associação de faces está a associação de vértice, pois cada conjunto de vértices na face do objeto um é associado um conjunto de vértices que compõe a face do objeto 2.

Depois de associada as faces e vértices é calculada a distância entre cada vértice do objeto origem até seu vértice associado no objeto destino. Com isso se descobre o incremento necessário em cada vértice em cada passo para que as faces cheguem em suas posições finais ao 'mesmo tempo'.

Com isso, se calcula cada um dos objetos interpolando os pontos. Os objetos calculados são guardados em memória para posterior apresentação. O fato de guardar em memória gera um alto consumo para objetos complexos, mas poupa processamento se desejamos ficar animando a imagem várias vezes, pois não precisamos recalcular o próximo elemento sempre que damos um *next* ou *back* no frame do morph. Para objetos muito grande, teríamos uma diminuição da *interatividade* na amostragem final, pois um cálculo muito demorado iria influenciar na amostragem final.

Não existe restrições no número de vértices ou faces dos objetos que podem ser *morphados*, o limite é a memória da máquina.

Para adicionarmos outro algoritmo, basta modificar/estender a classe modificando o método *associaFaces* de maneira a fazer a nova associação, pois o resto será gerado normalmente.

3.2 Interface

Uma interface amigável foi implementada para este trabalho. Os menus são bem descritivos e se alguma operação considerada 'ilegal' pelo programa tentar ser realizada pelo usuário esse será informado de como proceder.

Para as duas tarefas que podem demandar tempo (processamento do *morph* e gerar o arquivo obj de saída) foram criadas thread que executam essa tarefa em segundo plano enquanto barras de progresso e janelas com informações mostram a situação atual do processo.

Existem teclas de atalho para o acesso interno dos menus de opções.

4 Dificuldades Encontradas

Desta vez as dificuldades se basearam em problemas com a biblioteca JoGL. Esta apresentou-se extremamente instável em sua implementação para o componente de Swing GLJPanel. Com isso tornou-se impossível rodar a aplicação em muitas das máquinas disponíveis na Unisinos. Conseguiu-se em apenas máquinas que tinham placas e drivers da NVIDIA. Em uma máquina com rodando linux com placa NVIDIA teve-se sucesso, mas nos outros momentos foi utilizada uma máquina rodando windows.

Percebe-se que atual implementação da JoGL não é robusta e possui muitas falhas e problemas.

5 Funcionalidades

O trabalho funciona dentro das especificações citadas na página e acrescenta alguns detalhes visuais e de controle de interface com o usuário.