SpaceTime Invaders

Computação Gráfica

João Oliveira - 2010129856

Joaquim Mendes - 2011133854

FCTUC - Departamento de Engenharia Informática

1 Introdução

O objectivo deste projecto é aprofundar os conhecimentos adquiridos ao longo do semestre na cadeira de Computação Gráfica, englobando num só programa um conjunto de técnicas e tecnologias trabalhadas em separado. O projecto em si possui dois grandes requisitos, usar *openGL* e ter um robô capaz de disparar tiros.

Como linguagem de programação escolhemos o *C++*, para poder usufruir de todos os benefícios de programação orientada a objectos, mas poder na mesma usar a *API* de *openGL* com que nos fomos acostumando a trabalhar ao longo do semestre.

Na sua versão final, o nosso projecto tem as seguintes capacidades:

- Importar modelos a partir de ficheiros *.obj*;
- Colisões entre objectos (apenas *2D*, devido ao jogo implementado, que apesar de *3D* apenas permite movimentações num plano);
- Mapear texturas a modelos a partir de informação também contida em ficheiros *.obj*;
- Interacção de luz com os diversos modelos na cena, cujos coeficientes dos diversos componentes do modelo de *Phong* são lidos a partir de ficheiros *.mtl*;
- Reutilização de modelos e texturas em diversos objectos presentes no ecrã, evitando replicação de informação;
- Objectos construídos como uma malha de polígonos, sendo possível destruir partes desses objectos;
- Ecrã inicialmente informando o utilizador que se está a proceder ao carregamento dos diversos componentes do programa para memória;
- Menú de pausa semi-transparente;
- Nevoeiro.

Foram utilizados diversos modelos e texturas *open source* obtidos do website `http://tf3dm.com/`

2 Detalhes de Implementação

2.1 Importar Modelos e Mapear Texturas

Implementámos uma classe chamada *ModelManager* que se encarrega de carregar todos os modelos e texturas para memória, guardando-os em *hashmaps*. Isto permite que, para diversos objectos que utilizem o mesmo modelo ou textura, não seja necessário carregar os modelos e texturas de novo. Ficam também estes acessíveis a qualquer objecto que os use a partir dos seus nomes.

2.2 Colisões

Visto que o nosso jogo, apesar de *3D*, apenas se passa num plano, as colisões que implementámos foram *2D*. Isto permitiu-nos simplificar bastante o código correspondente, apesar de o termos feito de forma a ser escalável para mais dimensões.

Para cada objecto temos uma *bounding box*, o que transforma o problema das colisões num problema de intersecção de quadrados. Quando é detectada uma colisão, temos apenas de destruir o projéctil e o objecto atingido.

2.3 Malha de Polígonos

Existem no nosso projecto alguns objectos que são constituídos por partes que podem ser destruídas independentemente. Para que tal seja possível, os objectos são divididos em diversos polígonos tanto no eixo das abcissas como no eixo das ordenadas (não no eixo das cotas, visto que o jogo se passar no plano $X0Y$).

Desta forma, as colisões com partes do objecto são idênticas às entre objectos normais, sendo depois removidas as partes que já foram destruídas. Através de uma matriz bidimensional, mantemos a informação de que partes se encontram intactas, permitindo desenhar apenas as partes que ainda não foram destruídas.

Por uma questão de eficiência, quando partes adjacentes do objecto estão intactas, não são desenhadas as faces que seriam invisíveis por estarem adjacentes a outras partes. Estas apenas são desenhadas quando uma face necessita de ser visível para que o objecto esteja bem desenhado.

2.4 Ecrã de *Loading*

Imediatamente após a inicialização das configurações do *openGL*, é desenhado no ecrã um quadrado com as dimensões deste, sendo-lhe aplicada uma textura que indica ao utilizador que o programa está a carregar. Após carregar tudo o que é necessário, procede-se normalmente com o programa.