Mestrado em Computação Gráfica e Ambientes Virtuais
H2:Programação e Interacção

Avaliação por trabalho individual

Ano Lectivo 2004/05

Introdução

O trabalho que a seguir se apresenta constitui a avaliação da disciplina H2: Tópicos de Programação e Interacção do Mestrado em Computação Gráfica e Ambientes Virtuais. O objectivo principal do trabalho é consolidar os conhecimentos adquiridos nas aulas e que visaram fornecer mecanismos de homogeneização nas linguagens de programação que vão ser necessárias no decurso do mestrado.

Com vista ao reforço das noções de programação em C++, o presente trabalho opta, a exemplo do ano anterior, por uma problemática retirada da área da computação gráfica, por forma a que esforços efectuados no âmbito da resolução do trabalho possam eventualmente ser reutilizados noutras disciplinas.

O tempo para resolução do trabalho é de 2 semanas, estando a sua apresentação oral marcada para o fim de Novembro de 2004. Os alunos devem enviar email para `anr@di.uminho.pt` por forma a sugerir alternativas para o dia da apresentação.

Durante a apresentação os alunos devem entregar um *pequeno relatório* com a descrição do trabalho realizado, principais decisões tomadas e anexar o código fonte.

Objectivos

O trabalho proposto para esta disciplina tira partido do investimento feito ao longo das aulas em resolver o trabalho do ano anterior, reutilizando inclusivé classes feitas nesse contexto.

É objectivo do trabalho que se possam efectuar as seguintes operações:

1. Criar uma classe que representa pontos no espaço 3D (numa perspectiva vectorial). Estabelecer para esses pontos um conjunto de operações que permitam somar, multiplicar, efectuar deslocamentos sobre vectores 3D, que estão fornecidas no header file (.h) que é disponibilizado.
2. Criar uma classe que permita agregar uma série de esferas representadas por um par $< PontoCentro, Raio >$.
3. Criar uma classe OctTree, que represente a octree que pode ser gerada a partir de uma série de objectos que existam no espaço.

Descrição

Vectores 3D

Considere-se que o seguinte header file representa um vector 3D.

```
class Vect3D
{

public:

        float x,y,z;
```

```
        Vect3D::Vect3D(float x, float y, float z);
        Vect3D::Vect3D(const Vect3D &v);
        Vect3D::Vect3D();
        Vect3D::~Vect3D();

        Vect3D Vect3D::operator +(Vect3D v);
        Vect3D Vect3D::operator -(Vect3D v);
        Vect3D Vect3D::operator *(Vect3D v);
        Vect3D Vect3D::operator *(float t);
        Vect3D Vect3D::operator /(float t);
        Vect3D Vect3D::operator -(void);
        float length();
        void Vect3D::normalize();
        float Vect3D::innerProduct(Vect3D  v);
        void copy(Vect3D v);
        void set(float x,float y, float z);
        Vect3D scalarMult(float a);

        void Vect3D::print();

};
```

No ficheiro `vect3D.cpp`,que também se anexa, estão já definidas grande parte das operações necessárias para trabalhar com vectores 3D.

Sugestão: para determinar se o comportamento pretendido para esta classe está bem implementado crie um programa de teste (com uma função `main()`).

Conjunto de Esferas

Após termos capacidade de criarmos vectores 3D é necessário ter agora a possibilidade de guardarmos uma série de pares $< PontoCentral, Raio >$, que representam esferas em que o PontoCentral é um vector3D e o Raio um float.

Este conjunto de esferas vai ser percorrido para que seja possível criar a octree correspondente.

Como comportamento esta classe deve apresentar os seguintes métodos:

1. Construtor vazio que permita gerar uma série de 1000 esferas, de forma aleatória. Para que essas esferas estejam distribuídas de uma forma consistente, sugere-se a normalização das componentes x, y e z a valores do intervalo $[-10, 10]$. Para o raio sugere-se uma normalização a valores do intervalo $]0, 0.75]$.

2. Construtor parametrizado com o número de objectos a criar.

3. Construtor que permita ler de um ficheiro linhas com a definição das esferas. Essas linhas devem ter a forma `x, y, z, raio`.

4. Método que permita adicionar uma esfera ao conjunto de esferas existentes.

5. outros que julgue necessário implementar.

OctTree

Uma *octree* é uma árvore com 8 ramos que é utilizada para mapear os objectos espacialmente, por forma a criar-se uma noção de particionamento do espaço. A árvore representa a informação que existe em cada particionamento do espaço que se faz, sendo que esta informação é particularmente importante em programas de computação gráfica, na medida em que assim é possível desenhar (ou redesenhar) os objectos que estão a ser vistos. Como consequência da visita aos quadrantes do espaço que se pretendem mapear em termos de objectos que neles constam, o resultado obtido é uma árvore com informação

de quais os objectos que estão em cada quadrante. Como o objectivo é minimizar o esforço quando é necessário redesenhar uma cena, a octree resultante vai ter o maior detalhe possível. O processo de obtenção da octree é recursivo e pode ser definido segundo os seguintes passos:

1. Dividir o espaço em quadrantes. Inicialmente os quadrantes são obtidos sendo $x = 0$, $y = 0$ e $z = 0$, o que corresponde a 8 sub-divisões;
2. Para cada quadrante determinar os objectos que nele estão contidos. Se um determinado quadrante não contiver nenhum objecto a procura nesse quadrante pára e a sub-árvore correspondente é `NULL`.
3. Se num determinado quadrante os objectos não estiverem completamente contidos dentro do quadrante, o algoritmo pode contemplar 3 alternativas para continuar o processo de criação da árvore:
   (a) Deixar o objecto associado à raíz da sub-árvore, isto é, permite prosseguir a sub-divisão e implicitamente afirma que sempre que aquele quadrante tiver que ser desenhado, o objecto será desenhado;
   (b) Copiar o objecto para os quadrantes que ele atravessa, fazendo corresponder o objecto a cada um dos quadrantes que é por ele intersectado, e
   (c) partir o objecto por forma a ter em cada um dos quadrantes um único objecto. Esta alternativa é muito custosa, logo não deverá ser implementada.
4. Para cada quadrante, efectuar a repartição do espaço, ou seja dividi-lo em 8, e efectuar a mesma lógica de atribuição de objectos aos quadrantes;
5. A condição de paragem depende do grau de precisão que queiramos. Podemos parar o processo quando tivermos apenas um objecto em cada quadrante, ou então podemos ter pré-definido um número máximo de níveis de recursividade.

Para mais informação sobre Octrees, sugere-se a leitura do tutorial online que se pode obter em `http://www.gametutorials.com/Tutorials/OpenGL/Octree.htm`.

Exemplo de criação de uma árvore

Para melhor perceber a construção de uma árvore com a representação dos objectos, considere o exemplo da imagem da Figura 1.

A árvore que se encontra na Figura 2 é uma quadtree (neste caso apenas estamos a trabalhar as componentes x e y) que permite associar a cada quadrante a identificação dos objectos lá existentes. Repare que à medida que se desce na árvore vai-se refinando o nível de detalhe e é possível identificar com maior precisão o posicionamento dos objectos. Para a construção da árvore apresentada utilizou-se a estratégia que em caso de um objecto estar em mais do que um quadrante fica na raíz da sub-árvore associada a esse quadrante.

Entrega do trabalho

Por forma a entregar o trabalho deve criar um programa (um ficheiro.cpp) que contenha um método `main()` no qual declara as variáveis que forem necessárias. Na entrega do trabalho poderemos alterar o programa para acrescentar mais informação (por exemplo: mais esferas)

Anexos

Encontram em anexo, no website da disciplina a classe `Vect3D` quase completamente implementada. Encontra na referência dada acima do site `www.gametutorials.com` informação sobre o que é e para que serve uma octree.

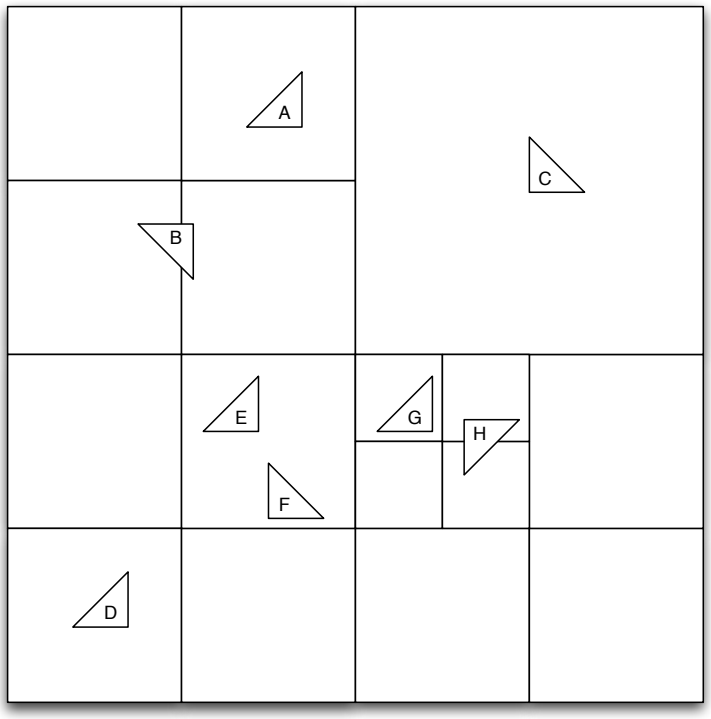

Figura 1: Quadrantes e objectos distribuídos espacialmente.

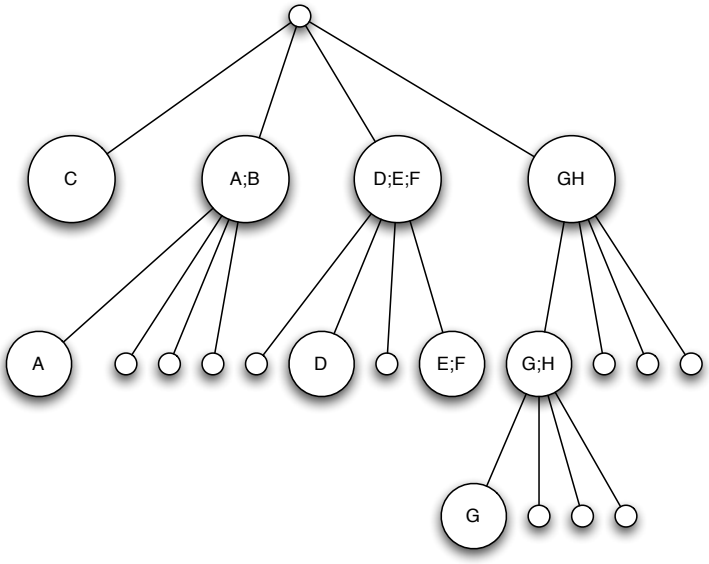

Figura 2: Árvore gerada a partir da Figura 1.