ESCOLA POLITÉCNICA

PUCRS

grupo de realidade virtual

Computação Gráfica
Trabalho sobre Métodos de Colisão
2022/2

Introdução

Este trabalho, que deverá ser feito em dupla ou individualmente, consiste em desenvolver programa que avalie algoritmos de inclusão de ponto em triângulos, usando OpenGL.

O programa deverá exibir um conjunto de pontos na tela e sobre este conjunto exibir um triângulo que simule o campo de visão um observador de um cenário 3D, visto de cima.

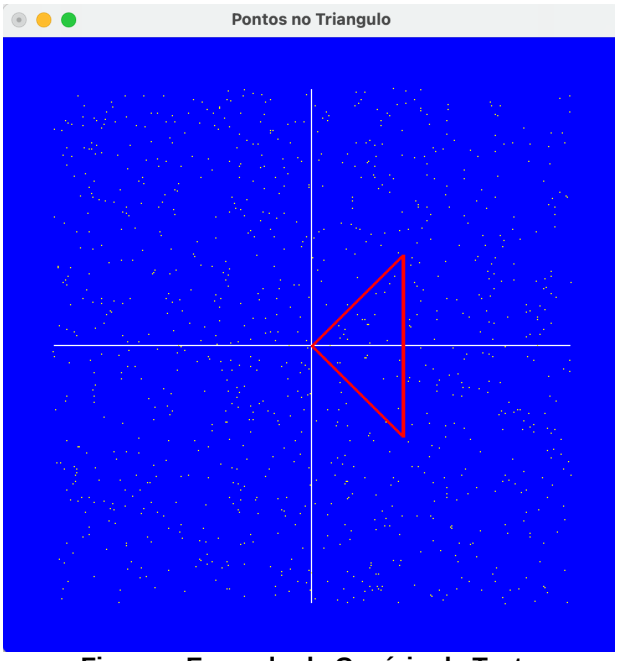

Figura – Exemplo de Cenário de Teste

O conjunto de pontos deve poder ser tanto lido de um arquivo, quanto ser gerado de forma aleatória, dependendo da escolha do usuário. Isto deve ser feito apenas uma vez, no início do programa e não a cada frame.

Independente da forma de obtenção dos pontos, este devem ser armazenados na mesma estrutura, conforme descrição a seguir.

O programa deve permitir que o triângulo navegue sobre a tela, andando para frente e girando ao redor do ponto que representa o observador do cenário. Deve ser possível também aumentar e diminuir o campo de visão do usuário.

Não serão aceitos trabalhos que não sejam genéricos quanto ao número de objetos gráficos manipulados.

Teste da Inclusão dos Pontos no Triângulo

Para determinar se os pontos estão dentro ou fora do triângulo devem ser usados, pelo menos, os seguintes algoritmos:

- Algoritmo de **força bruta** que testa todos os pontos contra o triângulo, usando o algoritmo de detecção de inclusão **em polígono convexo**;
- Algoritmo de **envelope**, "filtra" os pontos que estão fora do envelope do triângulo e chama o algoritmo de força bruta apenas para os pontos que estão dentro do envelope;
- Algoritmo de subdivisão por **Quadtree**, que determina quais as áreas têm colisão com o triângulo e testa, com o algoritmo de força bruta, apenas os pontos que estão nestas áreas. Mais detalhes a seguir.

Estes algoritmos deverão ser ativados ou desativados ao longo da execução, sem a necessidade de sair do programa.

Apresentação de dados de Desempenho

Quando uma tecla específica for pressionada, o programa deve ter a opção de exibir em verde os pontos que estão dentro do triângulo e em vermelho os pontos fora dele. Deve também se impressa a quantidade de pontos de cada categoria.

Quando uma tecla específica for pressionada, o programa deve ter a opção de ativar/desativar um dos algoritmos de otimização. Ao fazer a ativação, o programa deve:

- de exibir em verde os pontos que estão dentro do triângulo;
- exibir em amarelo os pontos que “passaram” nos filtros de envelope ou quadtree, mas que estavam fora do triângulo;
- exibir em vermelho os pontos que **não** “passaram” nos filtros de envelope ou quadtree.

Quando uma tecla específica for pressionada, o programa deve apresentar a contabilização de cada uma das categorias anteriores.

Geração e Utilização da Quadtree

Após a obtenção de todos os pontos, o programa deve gerar uma Quadtree que subdivida o espaço de visualização de forma que um nodo da árvore tenha, no máximo **N** pontos. Este valor de N deve poder ser alterado por teclado, durante a execução do programa, sem a necessidade de sair do programa.

Deve existir uma tecla que permita a exibição do envelope de cada nodo da Quadtree. Em cada nível, a exibição deve ser com uma cor diferente.

Descrição do Programa Base

Como base para implementar seu trabalho, utilize o código disponível na página de OpenGL da disciplina: https://www.inf.pucrs.br/pinho/CG/Aulas/OpenGL/OpenGL.html.

Ajuste o projeto para compilar o fonte **PontosNoTriangulo**.

Neste fonte, a função **init()**, chamada no início da execução do código, gera os pontos do cenário. Este processo pode tanto ser feito com dados lidos de arquivo, quanto com pontos gerados aleatoriamente, conforme a seguir.

```
// Geração de pontos a partir de um arquivo
PontosDoCenario.LePoligono("PontosDenteDeSerra.txt");

// Geração de pontos de forma aleatória no intervalo (0,0)-(500,500)
GeraPontos(1000, Ponto(0,0), Ponto(500,500));
```

Estes pontos são armazenados em uma estrutura de dados baseada na classe **Poligono**, conforme o código a seguir, apresentado em C++. Os códigos em Python e Java são muito semelhantes.

```
class Poligono
{
    vector <Ponto> Vertices;
public:
    Poligono();
    Ponto getVertice(int);
    unsigned long getNVertices();
    void insereVertice(Ponto);
    void desenhaPoligono();
    void desenhaVertices();
    void imprime();
};
```

Após a leitura, são ajustados os limites da área de trabalho do OpenGL, a fim de que os pontos possam ser exibidos na tela. As variáveis que armazenam estes limites chamam-se **Min** e **Max** e deverão ser usadas na geração da **Quadtree**.

Após a carga, a função **display()** é executada automaticamente.
Para exibir os pontos é chamado o método **desenhaVertices()**, conforme o trecho de código a seguir.

```
    glPointSize(5);
    glColor3f(1,1,0); // R, G, B  [0..1]
    PontosDoCenario.desenhaVertices();
```

Criação de um envelope

Um envelope pode ser definido como sendo um **polígono de 4 vértices**. Sua *declaração*, *inicialização*, *atualização* e *exibição* deve ser feita nos mesmos pontos do código em que o objeto **CampoDeVisao** é criado, atualizado e utilizado.

Para declará-lo, crie um objeto da classe polígono no mesmo local onde é criado o objeto **CampoDeVisao**.

Para inicializá-lo insira, no final função **init()**, um código que insira os vértices do envelope no polígono do envelope.

Para atualizá-lo, coloque, no final função **void PosicionaTrianguloDoCampoDeVisao()** um código que altere as coordenadas do polígono do envelope. Para alterar as coordenadas de um vértice, a classe Poligono possui um método chamado **alteraVertice(int i, Ponto P)**, no qual o primeiro parâmetro é um índice do vértice e o segundo, um ponto que contém a nova coordenada.

Para exibi-lo, coloque, na função **void display()**, uma chamada do método **desenhaPoligono()**, logo depois do trecho onde é exibido o polígono do campo de visão.

Acesso aos pontos do polígono

Para acessar um ponto de um polígono e manipulá-lo individualmente, para, por exemplo, exibi-lo com uma determinada cor, pode-se rodar um código como que segue.

```
#include "ListaDeCoresRGB.h"
void DesenhaVerticesColoridos (Poligono Poly)
{
    for (int i=0; i<Poly.getNVertices(); i++)
    {
        Ponto P;
        P = Poly.getVertice(i); // obtem a coordenada
        glBegin(GL_POINTS);
            if (P.x> P.y == 0) // critério para escolher a cor
                defineCor(NeonPink);
            else defineCor(GreenYellow);
            glVertex3f(P.x,P.y,P.z);
        glEnd();
    }

}
```

A função **DesenhaVerticesColoridos** deve ser chamada na função **display**.

Entrega

- Data de entrega no *Moodle* e apresentação: **08/09/2022** até o horário da aula**.**
- Os trabalhos podem ser desenvolvidos em duplas. Os arquivos, contendo os fontes do programa, devem ser compactados e submetidos pelo *Moodle* até a data e hora especificadas. **ENVIE APENAS ARQUIVOS .ZIP, ou seja, não envie 7z, rar, tar.gz, tgz, tar.bz2, etc.**
- A nota do trabalho depende da apresentação deste no laboratório, na data marcada. Trabalhos entregues, mas não apresentados, terão sua nota anulada automaticamente. Durante a apresentação será avaliado o domínio da resolução do problema, podendo inclusive ser possível invalidar o trabalho quando constatada a falta de conhecimento sobre o código implementado.
- **A cópia parcial ou completa do trabalho terá como consequência a atribuição de nota ZERO ao trabalho dos alunos envolvidos.**

FIM.