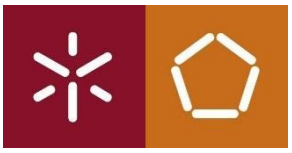


Universidade do Minho
Mestrado Integrado em Engenharia Informática

Relatório Trabalho Prático

Computação Gráfica

Grupo 33

Trabalho realizado por:

António Alexandre Carvalho Lindo A85813

Nuno Azevedo Alves da Cunha A85400

Pedro Dias Parente A85919

Geração de Figuras

A aplicação generator criada pelo nosso grupo é capaz de gerar as coordenadas de pontos que quando fornecidas à nossa engine representam as seguintes figuras:

1. Plano

   O plano é gerado com apenas um argumento, a sua dimensão.

   Os pontos para gerar os 2 triângulos, necessários para a representação do plano são os cantos que são obtidos com a dimensão fornecida e a sua distância à origem no Eixo respetivo.

   A partir desses 4 pontos são gerados os 2 triângulos necessários (2 pontos partilhados).

2. Caixa

   A caixa é gerada com 3 argumentos: o tamanho em x, y e z respetivamente.

   Para gerar os pontos para cada face, o processo é igual ao do plano aplicado a cada face da Caixa.

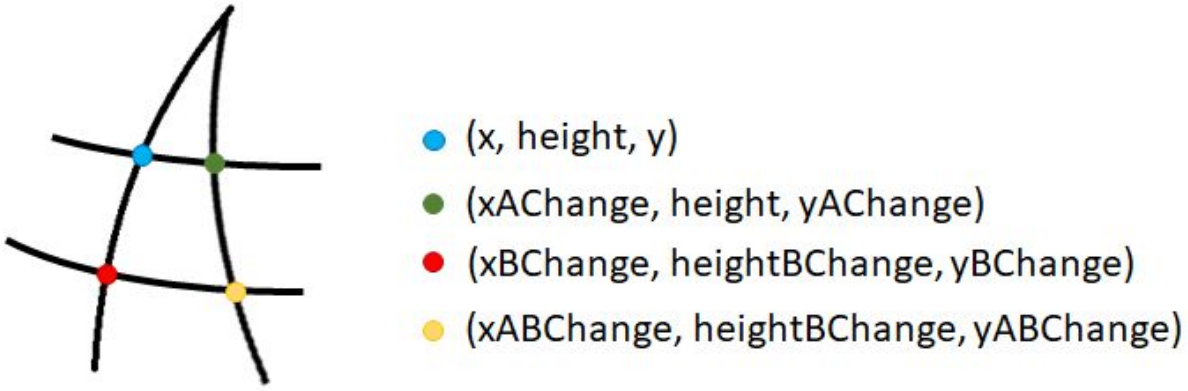
3. Esfera

   A esfera é gerada com 3 argumentos: O raio da esfera, o número de slices e o número de stacks.

   Tendo o número de slices, o ângulo do arco que cada slice deve ter é calculado dividindo 2pi (radianos) sobre o número de slices.

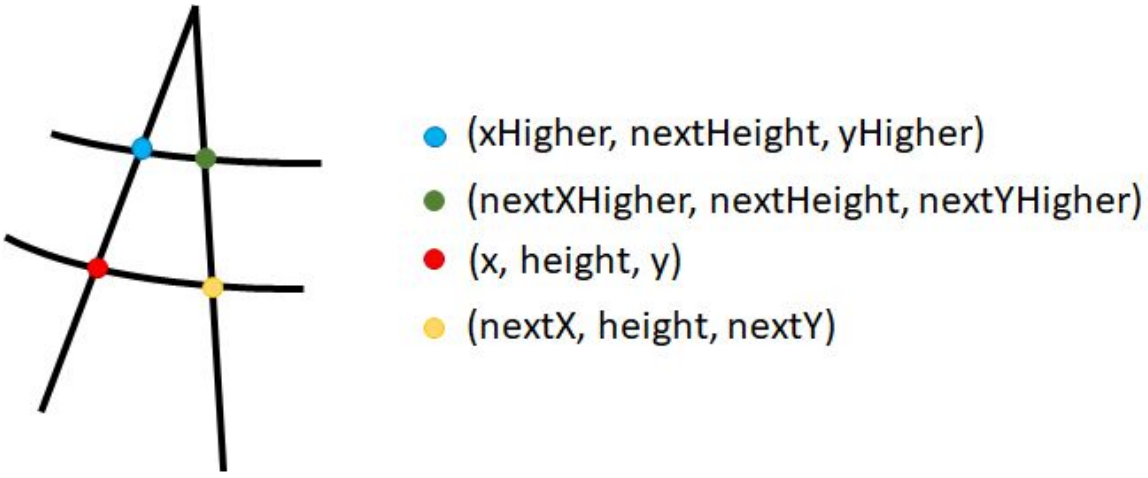
   Tendo o número de stacks, o ângulo do arco que cada stack deve ter é calculado dividindo pi (radianos) sobre o número de stacks.

   Sabendo isto podemos começar o processo de obter os vértices de cada triângulo que constitui a esfera.



Se for a stack superior (ou inferior), apenas temos de ter triângulos completamente à volta do ponto superior (ou inferior) da esfera, ou seja, através de coordenadas esféricas, iremos calcular o x, xAChange, y e yAChange sendo os "next" as coordenadas do começo do próximo slice. Se a stack não for nem a superior nem a inferior (ou seja, uma das do meio) também temos de calcular o xBChange, heightBChange e yBChange (e posteriormente o xABChange e yABChange usando a mesma lógica anteriormente referida) fazendo o mesmo processo mas desta vez obtendo as coordenadas da próxima stack e não do próximo slice. No fundo, o **AChange é a mudança para o slice seguinte** e o **BChange é a mudança para a stack seguinte**. Por último, para desenhar os triângulos destas stacks centrais, visto que no fundo são pequenos planos, usamos a mesma lógica que usamos para o plano.

4. Cone



O cone é gerado com 4 argumentos: O raio do cone, a altura, o número de slices e o número de stacks.

Tal como na esfera a divisão das slices é na mesma feita pela divisão de 2pi radianos pelo número de slices.

Para as stacks, estas são divididas apenas pela altura do cone.

Similar à esfera, na stack superior apenas vamos ter triângulos completamente à volta do ponto superior do cone, apenas tendo de calcular x, y, nextX e nextY.

Para a stack inferior construímos uma base com triângulos em forma de “pizza”, ou seja, a altura vai ser 0, e, como na anterior, apenas temos de calcular x, y, nextX e nextY.

Para as stacks do meio, para além das calculadas nas outras stacks, calculamos também o xHigher, nextHeigth, yHigher (e posteriormente o nextXHigher e nextYHigher) usando um novo raio (pois à medida que a altura do cone aumenta, o raio diminui).

Generator

Para o Generator, usando funções criadas na classe que trata das figuras, somos capazes de obter, fornecendo os vários argumentos necessários, as coordenadas de cada vértice que vai que constituir a figura (em formato de vector), e de seguida usamos a função auxiliar geraPontosFich para escrevermos, com um formato decidido por nós, todos esses vértices num ficheiro .3d, para ser posteriormente lido na a fase do Engine.

Engine

Para o Engine, utilizámos o tinyxml2 como ferramenta de suporte para a leitura dos ficheiros XML. Através desta ferramenta, elaboramos a função: vector<string> readXML(string nome) que percorre um dado ficheiro xml e devolve um vetor com o nome dos ficheiros 3d nele presentes. Depois criamos também um função: void parseFile(string file) que lê os ficheiros 3d e guarda as coordenadas dos vértices numa estrutura do tipo: vector<vector<float>> (variável global).A partir desta estrutura que contém as coordenadas dos vértices de cada figura é depois possível desenhar as figuras. As figuras são desenhadas 1 ponto de cada vez (3 coordenadas) na função void renderScene(void).