Computação Gráfica (MIEIC)

2019/2020

Projeto Final

v1.0 (2020/03/30)

Objetivos

- Aplicar os conhecimentos e técnicas adquiridas até à data
- Criação de uma cena complexa, com diferentes geometrias, materiais, texturas e luzes
- Exploração de shaders, interação, controlo por teclado e animação

Descrição

Pretende-se com este projeto a criação de uma cena que combine os diferentes elementos explorados nas aulas anteriores, acrescentando alguns novos elementos. Para este trabalho deve usar como base o código que é fornecido no Moodle, que corresponde a uma cena vazia. Terá posteriormente de adicionar alguns dos objetos criados anteriormente.

A cena será no final genericamente constituída por:

- Uma paisagem envolvente (*cube map*)
- Um veículo controlado pelo utilizador
- Um terreno
- Outros elementos

Os pontos seguintes descrevem as principais características dos diferentes elementos pretendidos. É dada alguma liberdade quanto à composição dos mesmos na cena, para que cada grupo possa criar a sua própria cena.

O enunciado está dividido em duas partes, com uma proposta de um plano de trabalhos no final de cada uma delas.

Parte A

1. Criação de objetos de base

1.1. MyCylinder - Cilindro (sem topos)

Crie uma nova classe ***MyCylinder*** que aproxime um cilindro de raio de uma unidade, com um número variável de "lados" (**slices**), e altura de uma unidade em Y. A base do cilindro deve ser coincidente com o plano XZ e centrada na origem.

As normais de cada vértice devem ser definidas de forma a aproximar uma superfície curva, de acordo com o método de Smooth Shading de Gouraud, como demonstrado na **Figura 1**.

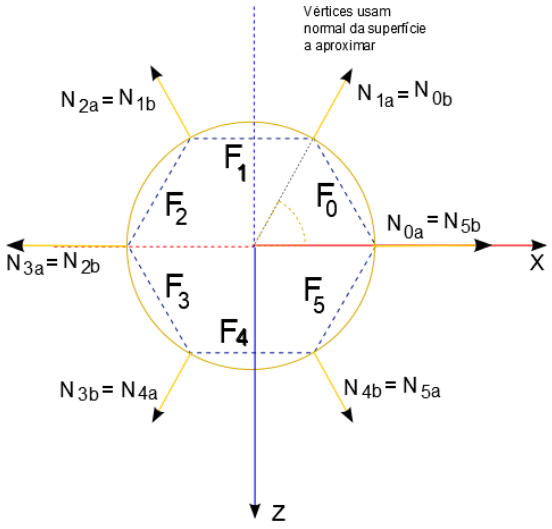

Figura 1: Ilustração das normais a atribuir a cada vértice no caso de um cilindro aproximado por seis lados.

Para cada vértice, defina as coordenadas de textura de forma a mapear uma textura à volta do cilindro. Note que como a textura dá a volta ao cilindro poderá necessitar de repetir a primeira linha vertical de vértices pois estes são, simultâneamente, o início e o fim da textura.

1.2. MySphere - Esfera

No código fornecido encontra uma classe MySphere correspondente a uma esfera com o centro na origem, com eixo central coincidente com o eixo Y e raio unitário.

A esfera tem um número variável de "lados" à volta do eixo Y (slices), e de “pilhas” (stacks), que corresponde ao número de divisões ao longo do eixo Y, desde o centro até aos "pólos" (ou seja, número de "fatias" de cada semi-esfera). A **Figura 2** tem uma representação visual da esfera.

Pretende-se que analise o código deste objeto, e acrescente o código necessário para gerar as coordenadas de textura para aplicar texturas à superfície da esfera, como demonstrado na **Figura 3**.

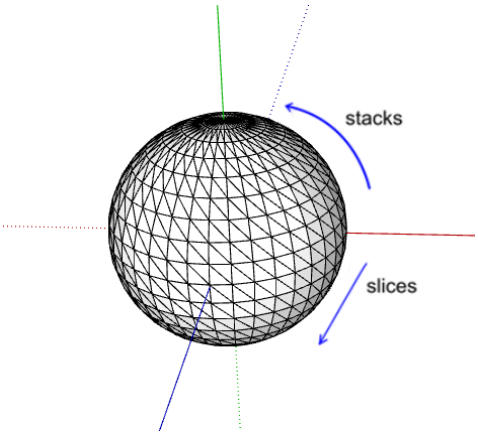

Figura 2: Imagem exemplo de uma esfera centrada na origem.

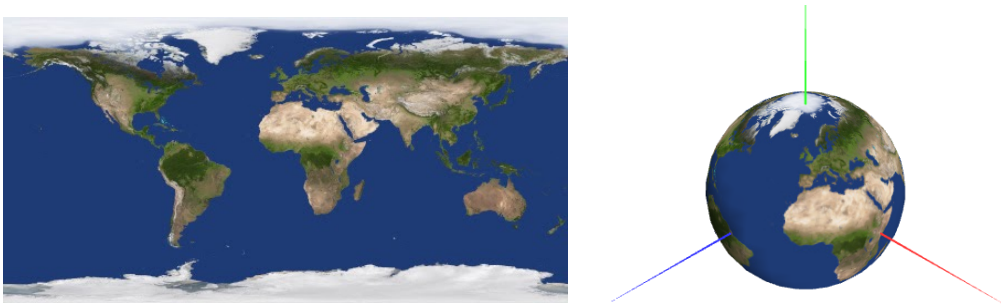

Figura 3: Exemplo de textura (disponível no código base) e sua aplicação numa esfera

1.3. Cube Map - Criação de um cubemap para o ambiente

Com este exercício pretende-se que crie um *cube map* que sirva como ambiente de fundo da cena. Um *cube map* pode ser definido como:

- um cubo de grandes dimensões (bastante superiores à da cena visível),
- com componente especular e difusa nulas, e componente ambiente forte,
- apenas com as faces interiores visíveis,
- e às quais são aplicadas texturas que representam a envolvente da cena (uma paisagem, por exemplo; ver **Figura 4**).

Este objeto pode ser obtido de duas formas diferentes (devem escolher uma):

A. usando por base o ***MyUnitCube*** e uma única textura com a disposição da Figura 4, e mapeando diferentes partes da mesma a cada uma das faces ou

B. usando por base o ***MyUnitCubeQuad*** e seis texturas diferentes, uma para cada face do cubo

Escolha uma dessas duas opções e inclua na pasta do projeto uma cópia do código da classe respetiva - ***MyUnitCube*** ou ***MyUnitCubeQuad***. Modifique essa cópia para criar uma classe ***MyCubeMap***, de forma a ser visível por dentro, e com coordenadas de textura de acordo com essa opção.

O *cube map* deve ser unitário e ao ser usado na cena, deve ser escalado de forma a medir **50 unidades de lado**. Se necessário, poderá ter de alterar a posição da câmara de forma a que fique no interior e centrada dentro do *cube map*.

O código de base inclui um exemplo de imagens para cada um dos dois tipos de *cube map* (uma semelhante à da Figura 4, e um pack de 6 imagens para a opção de seis *quads*).

Devem procurar/criar pelo menos mais uma imagem envolvente à vossa escolha, para permitir depois ao utilizador escolher entre essa e a de exemplo através da interface gráfica (ver ponto 2.2). Como ponto de partida, podem consultar o endereço:
https://www.cleanpng.com/free/skybox.html

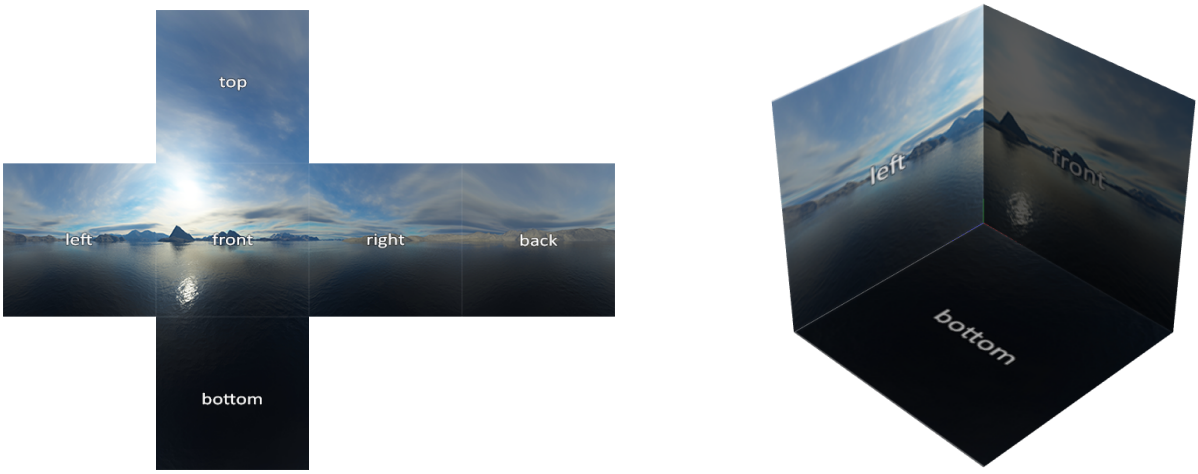

Figura 4: Imagem do tipo skybox, e a sua aplicação num cubemap visualizado de fora

1.4. MyVehicle - Versão preliminar

Crie a classe ***MyVehicle***, que irá representar o veículo na cena. A constituição final do veículo será detalhada na Parte B do enunciado. Nesta fase deve criar-se apenas uma representação básica que permita identificar a sua posição e orientação. Para esse efeito, sugerimos que use uma forma simples, como um triângulo não-equilátero como os ***MyTriangle*** criados para o Tangram, ou a ***MyPyramid*** fornecida no TP3 (ver **Figura 5**).

Independentemente da geometria escolhida, deve ser garantido o seguinte:

- Ser centrada na origem,
- Ter tamanho unitário
- Ter o eixo central alinhado com o eixo positivo dos ZZ, ou seja: ter a frente a apontar na direção de +Z, como demonstrado na **Figura 5.**

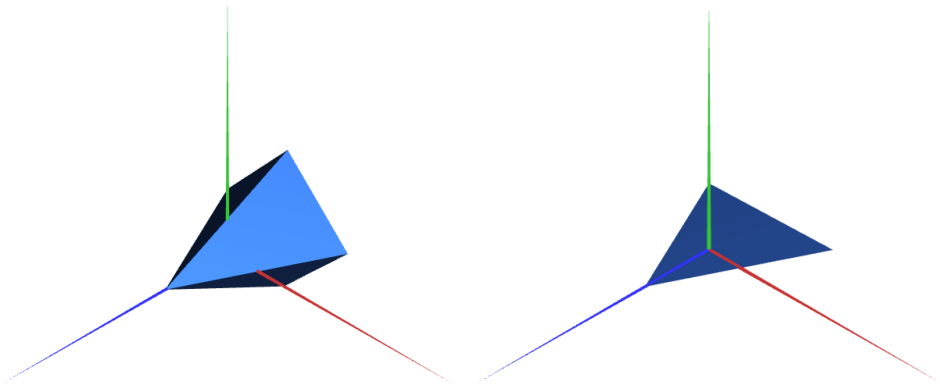

Figura 5: Exemplificação da versão preliminar de MyVehicle
a) Pirâmide: b) Triângulo no plano XZ

2. Interface

2.1. Controlo de veículo por teclado

Para poder controlar o movimento do veículo na cena utilizando teclas, será necessário alterar:

- A interface, para detetar as teclas pressionadas
- A cena, para invocar funções de controlo no veículo em função das teclas detetadas
- O veículo, para contemplar funções de controlo como “rodar” e “avançar” que alteram a posição e orientação do mesmo

Apresentam-se em seguida de forma mais detalhada os passos necessários para o efeito.

NOTA IMPORTANTE: Se usar *copy-paste* com o código seguinte, alguns símbolos podem não ser bem reconhecidos (apesar de parecerem iguais) e causar erros em Javascript. Evite fazê-lo, ou verifique bem o código.

1. Altere a classe ***MyInterface***, adicionando os seguintes métodos para processar várias teclas ao mesmo tempo:

```
initKeys() {
  // create reference from the scene to the GUI
  this.scene.gui=this;

    // disable the processKeyboard function
  this.processKeyboard=function(){};

  // create a named array to store which keys are being pressed
  this.activeKeys={};
}
```

```
  processKeyDown(event) {
    // called when a key is pressed down
    // mark it as active in the array
    this.activeKeys[event.code]=true;
  };

  processKeyUp(event) {
    // called when a key is released, mark it as inactive in the array
    this.activeKeys[event.code]=false;
  };

  isKeyPressed(keyCode) {
    // returns true if a key is marked as pressed, false otherwise
    return this.activeKeys[keyCode] || false;
  }
```

NOTA: No final da função *init()* da ***MyInterface***, chame a função *initKeys().*

2. Na classe ***MyScene*** acrescente o seguinte método *checkKeys()* e acrescente uma chamada ao mesmo no método *update().*

```
  checkKeys()  {
     var text="Keys pressed: ";
     var keysPressed=false;

     // Check for key codes e.g. in https://keycode.info/
     if (this.gui.isKeyPressed("KeyW")) {
           text+=" W ";
           keysPressed=true;
     }

     if (this.gui.isKeyPressed("KeyS"))     {
           text+=" S ";
           keysPressed=true;
     }
     if (keysPressed)
           console.log(text);
  }
```

Execute o código e verifique as mensagens na consola quando “W” e “S” são pressionadas em simultâneo.

3. Prepare a classe ***MyVehicle*** para se deslocar:
    - Acrescente no construtor variáveis que definam:
        - a orientação do veículo no plano horizontal (ângulo em torno do eixo YY)
        - a sua velocidade (inicialmente a zero)
        - a sua posição (x, y, z)
    - Crie a função ***MyVehicle.update()*** para atualizar a variável de posição em função dos valores de orientação e velocidade.
    - Use as variáveis de posição e orientação na função ***display()*** para orientar e posicionar o veículo.
    - Crie os métodos ***turn(val)*** e ***accelerate(val)*** para alterar o ângulo de orientação, e para aumentar/diminuir o valor da velocidade (em que ***val*** podem ser valores positivos ou negativos).
    - Crie a função *reset()* que deverá recolocar o veículo na posição inicial, com rotação e velocidade nula.

4. Altere o método *checkKeys()* da **MyScene** de forma a invocar os métodos *turn(), accelerate()* ou *reset()* do veículo para implementar o seguinte comportamento em função das teclas referidas:
    - **Aumentar ou diminuir a velocidade** conforme se pressionar **"W"** ou **"S"**, respectivamente.
    - **Rodar o veículo** sobre si mesmo para a esquerda ou direita se pressionar as teclas **"A"** ou **"D"** respetivamente.
    - Pressionar a tecla "**R**" deverá invocar a função *reset()* do veículo.

2.2. Controlos adicionais na interface

Acrescente os seguintes controlos adicionais na interface gráfica (GUI), para poder parametrizar o movimento do veículo e a aparência do *cube map*:

1. Crie uma *list box* onde apareçam as **diferentes texturas de paisagens envolventes** disponíveis, para o utilizador poder alterar a textura do *cube map* em tempo de execução (sugestão: siga o exemplo das texturas no TP4).

2. Crie um *slider* na GUI chamado **speedFactor** (entre 0.1 e 3) que multiplique o valor da velocidade de deslocamento do veículo, a cada vez que se pressiona uma tecla "W" ou "S".

3. Crie outro *slider* na GUI chamado **scaleFactor** (entre 0.5 e 3) que permita escalar o veículo de forma a que seja mais fácil observar as suas animações.

Proposta de plano de trabalho

- Semana de 30 de março: Início do ponto 1
- Semana de 6 de abril: Continuação do ponto 1; Ponto 2.1
- Semana de 13 de abril: Ponto 2.2 (+ início da Parte B, a ser publicada até lá)

Parte B

(a publicar brevemente)