

Computação Gráfica (MIEIC)

Projeto Prático

Versão Inicial - v0.1 - 2019/05/03

Projeto B

Objetivos

- Aplicar os conhecimentos e técnicas adquiridas até à data
- Utilizar elementos de interação com a cena, através do teclado e de elementos da interface gráfica
- Utilizar animação de componentes

Descrição

Pretende-se com este projeto a criação de uma cena que combine os diferentes elementos explorados nas aulas anteriores. Para este trabalho deve usar como base o código que é fornecido no Moodle, que corresponde a uma cena com um plano de 60x60 unidades. Terá posteriormente de adicionar alguns dos objetos criados em trabalhos anteriores.

A cena, no final do projeto, deve ser genericamente constituída (pelo menos) por:

- Elementos criados no projeto A:
 - Uma casa;
 - Uma paisagem envolvente (skybox);
- Um terreno com elevações, criadas através de um shader;
- Uma floresta, composta por árvores posicionadas usando geração procedimental;
- Uma ave, animada e controlável pelo utilizador, assim como o seu ninho.

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.

Preparação do Ambiente

Devem descarregar o código disponibilizado para este trabalho do Moodle e colocar a pasta **projB** contida no ficheiro .zip, ao mesmo nível dos trabalhos anteriores. Devem garantir que têm a versão mais recente da WebCGF (2.0.1 ou superior) na pasta **lib**.

O código fornecido para o projeto B tem uma cena constituída apenas pelo eixo de coordenadas, um objeto **Plane**, e uma luz. Deverá incluir neste projeto alguns elementos criados no projeto anterior, nomeadamente o skybox **MyCubeMap**, a casa **MyHouse**.

A casa deverá ter até 3 unidades de lado, colocada a menos de 8 unidades de distância da origem (de forma a ser visível com a inicialização da cena).

Inclua todos os ficheiros necessários para colocar estes elementos em cena.

1. Modelação e Aparência de Ave

Crie uma nova classe **MyBird**, para representar uma ave. Para modelar a ave, poderá utilizar uma combinação dos diferentes objetos criados anteriormente, de forma a que a ave seja constituída por corpo, cabeça com olhos e bico, e asas. As asas deverão ser constituídas por duas partes. A figura 1 ilustra o tipo de estrutura pretendida, mas com o corpo e cabeça simplificados.

A ave a ser desenvolvida deve usar geometria um pouco mais complexa para o corpo e a cabeça (não devendo no entanto ter um número excessivo de triângulos). Sugerimos o uso de cones ou cilindros com um número reduzido de lados (3 a 5). Podem criar outros objetos que considerem relevantes para a representação da ave, como p.ex. uma esfera.

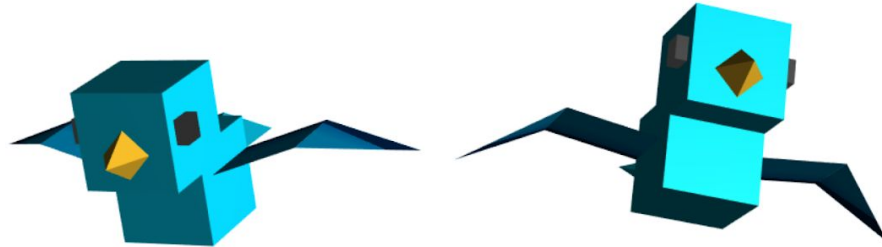



Figura 1: Exemplificação do modelo da ave, simplificada com cubos para o corpo e cabeça.

A ave deverá ser visível quando a cena é iniciada, de forma a facilitar a avaliação da mesma, assim como os outros elementos da cena. Poderá ter dimensões até 2 unidades, sendo que com asas abertas poderá ter comprimento até 3 unidades.

Os diferentes objetos utilizados para criar a ave deverão ter materiais e texturas aplicados aos mesmos, adequados às partes da ave que representam. Mostre uma imagem da aparência da ave

(suficientemente perto para ver em detalhe) .  (1)

2. Animação inicial de Ave

Neste exercício procura-se criar os mecanismos de animação e controlo para um objeto da classe **MyBird**.

A ave deverá ter duas animações aplicadas de forma constante ao longo do tempo, semelhante ao movimento do voo. Coloque a Ave na cena a cerca de 3 unidades acima do chão.

1. O objeto de **MyBird** terá uma animação para oscilar levemente para cima e para baixo, cada oscilação completa (cima e baixo) deverá demorar 1 segundo.
2. Uma segunda animação representará o bater das asas, a velocidade do bater de asas deverá depender da velocidade atual da ave (ver próxima secção).

3. Controlo de Ave

Para poder controlar o movimento da ave na cena, será necessário acrescentar código para detetar o pressionar de uma ou mais teclas em simultâneo.

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;
}
```

No final da função *init* do **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 **MyBird** para se deslocar:
 - Acrescente no construtor variáveis que definam:
 - a orientação da ave (sugestão: ângulo em torno do eixo YY)
 - a sua velocidade (inicialmente a zero)
 - A sua posição (x, y, z)
 - Altere a função `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 a ave.
 - Crie os métodos **`turn(v)`** e **`accelerate(v)`** para rodar a ave, e para aumentar/diminuir a velocidade (em que **`v`** é um valor positivo ou negativo)
4. Utilize na cena as teclas para invocar os métodos `turn` e `accelerate` da ave de forma a implementar o seguinte comportamento:
 - Acelerar ou travar conforme se pressionar "**W**" ou "**S**", respectivamente.
 - Rodar a ave para a esquerda ou direita se pressionar as teclas "**A**" ou "**D**";
 - Pressionar a tecla "**R**" deverá fazer "reset" à posição e velocidade da ave, isto é, deverá ser colocada na posição inicial, com rotação e velocidade nula.
5. Crie um slider na GUI chamado `speedFactor` (entre 0.1 e 3) que acelere e desacelere a velocidade de deslocamento, rotação e bater de asas da ave.
6. Crie outro slider na GUI chamado `scaleFactor` (entre 0.5 e 3) que permita escalar a ave de forma a que seja mais fácil observar as suas animações.

4. Terreno

Pretende-se melhorar o terreno e a sua altimetria, utilizando shaders. Crie uma nova classe **MyTerrain**, que será constituído por um **Plane** (aula prática de shaders), para representar o terreno com elevações obtidas a partir de uma textura que funciona como mapa de alturas. A criação dos shaders, das texturas associadas e a sua aplicação deve ser encapsulada dentro da classe **MyTerrain**.

A figura 2 contém um exemplo possível de **MyTerrain**. Deve substituir o **Plane** inicial de 60x60 unidades por um **MyTerrain** com a mesma dimensão.

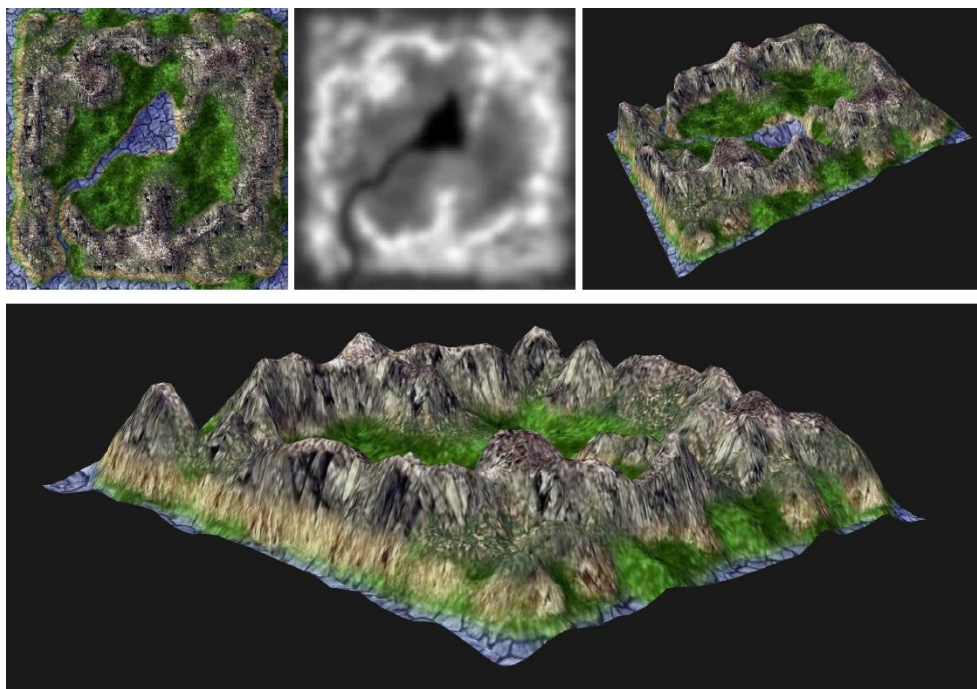



Figura 2: Textura de cor, mapa de alturas e geometria gerada.

(origem: Outside of Society http://oos.moxiecode.com/js_webgl/terrain/index.html)

1. Teste uma versão inicial com as texturas de cor e altura fornecidas.
2. Modifique a textura de alturas (usando um editor de imagem) de forma a que haja uma zona central **plana** de dimensões aproximadas de 20x20 unidades ($\frac{1}{3}$ da largura/comprimento do terreno). Em alternativa, poderá utilizar outro par de texturas de cor e altura para o efeito - **mas que terá de ter a zona central plana como descrito**
3. Altere o **shader** de forma a que:
 - a. receba uma terceira textura representando um gradiente de cor (ver figura 3a, imagem fornecida com o código)
 - b. Em cada fragmento, em vez de aplicar apenas a cor da textura original do terreno, combine essa cor com o valor da cor do gradiente correspondente à altura: nas altitudes mais elevadas a cor original deverá ser mais “branca” e nas inferiores será mais “azul” (ver exemplo na figura 3c e 3d). (Nota: poderá ser feito algum ajuste a escalas ou offsets para adaptar o gradiente à gama de alturas).
4. Capture uma perspetiva da cena contendo o terreno, a casa e a ave (pode exagerar a escala desta última para ser mais visível)  (2)

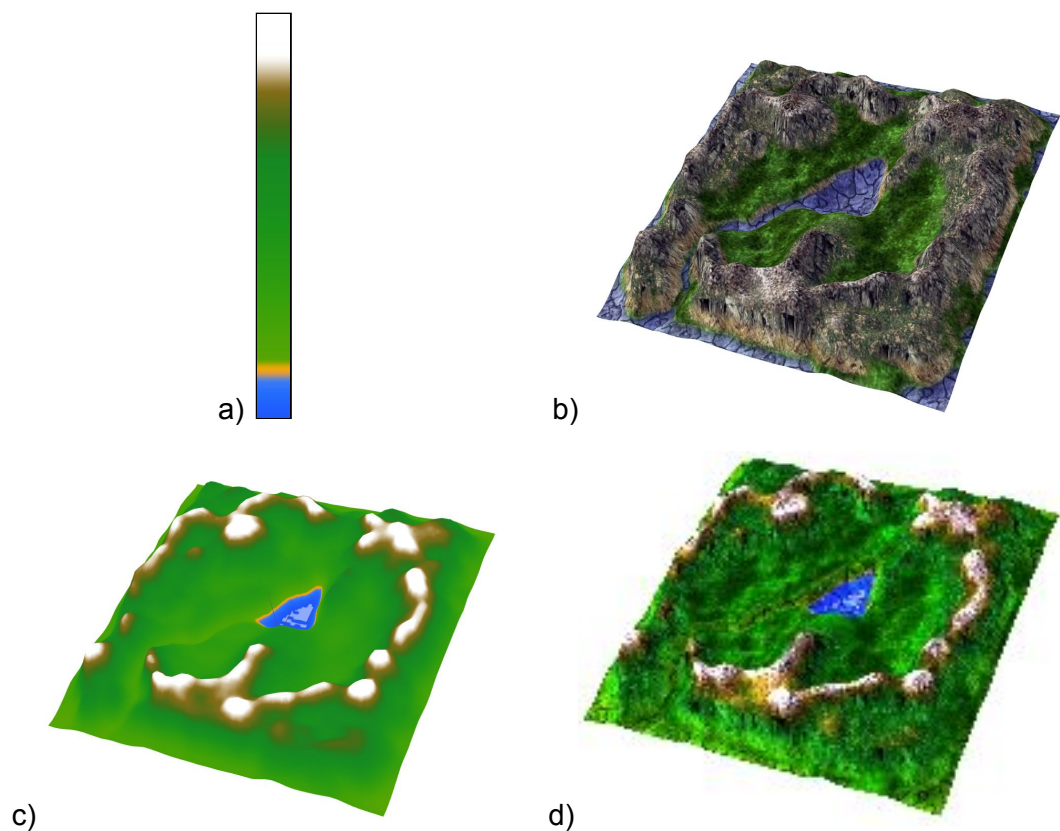


Figura 3: Utilização de uma textura para colorir zonas do terreno em função da altura

a) gradiente;

b) terreno só com textura de cor original

c) terreno só com cor do gradiente;

d) combinação gradiente e cor original

(2ª parte a ser publicada posteriormente)