

# Computação Gráfica (L.EIC)

## Projeto 2024/2025

Versão v1.0 - 2025/03/30

## 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 *Shaders* na modelação/visualização de objetos
- Utilizar animação de componentes

## Descrição

Pretende-se com este projeto a criação de uma cena 3D que combine a aprendizagem efetuada nas aulas anteriores e que explore algumas novas técnicas de Computação Gráfica. Deve usar como base o código que é fornecido no Moodle, que corresponde a uma cena com um plano de 400x400 unidades. Terá posteriormente de adicionar vários novos objetos.

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

- Uma representação do céu com nuvens em movimento
- Um terreno plano, coberto com relva
- Um edifício de bombeiros dotado de "Heliporto" no teto
- Um helicóptero, animado e controlável pelo utilizador
- Um lago
- Uma floresta com árvores de dimensões, cores, e posições variáveis

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 usar de criatividade e originalidade na obtenção da sua própria cena.

## Preparação do Ambiente

Deve descarregar o código disponibilizado para este trabalho do Moodle (ficheiro ZIP) e colocar na pasta **project**, ao mesmo nível dos trabalhos anteriores.

O código fornecido para o projeto tem uma cena constituída apenas pelos eixos do sistema de coordenadas, um objeto **MyPlane**, e uma fonte de luz.

De momento, aplique uma textura de relva ao plano. No decurso do projeto deverá incluir os restantes objetos, conforme forem sendo definidos.

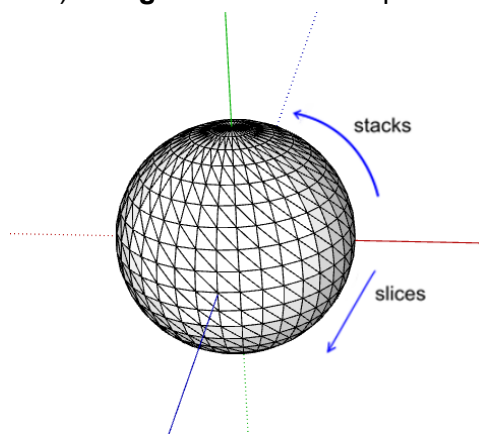
# 1. Sky-Sphere

Uma *sky-sphere* é uma esfera visível pelo seu interior e que simula a visualização de terreno até à linha de horizonte e, acima desta, a visualização do céu. É suposto que a câmara ativa, em dado momento, se encontre no seu interior.

## 1.1 Criação de uma Esfera

Crie uma nova classe **MySphere** que crie uma esfera com o centro na origem, com eixo central coincidente com o eixo Y e raio unitário.

A esfera deve ter um número variável de "lados" em torno do eixo Y (*slices*), e de "camadas" (*stacks*). As camadas devem corresponder a divisões angulares iguais e o seu número é contabilizado desde a "linha de equador" até aos "pólos" (ou seja, é o número de "camadas" de cada semi-esfera). A **Figura 1** tem uma representação visual da esfera.

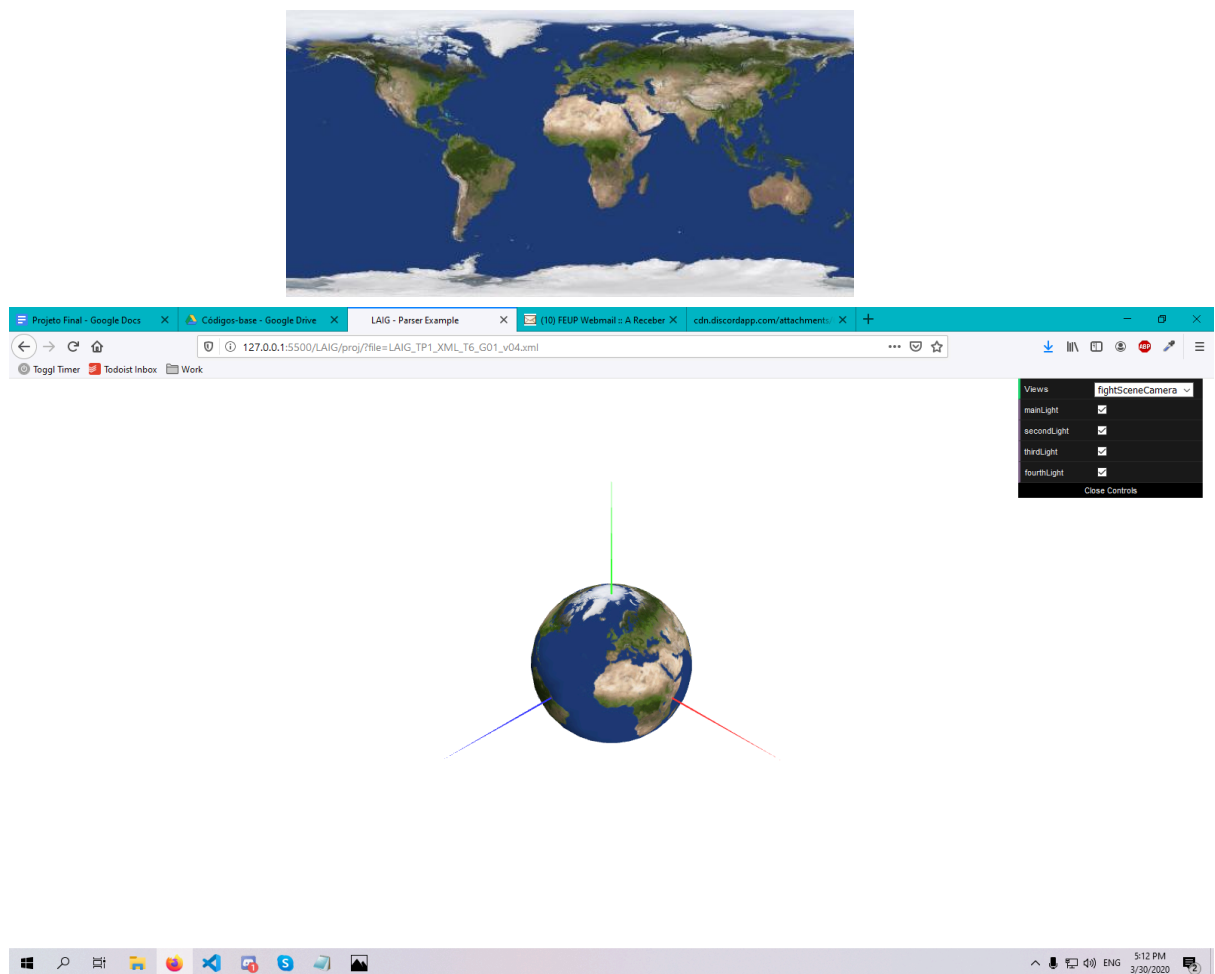


**Figura 1:** Imagem exemplo de uma esfera centrada na origem.

### Notas:

- As camadas (*stacks*) **não devem** ser definidas com base numa divisão em altura segundo espaços regulares  $\Delta y$ ; em alternativa, **devem ser obtidas por divisão angular** do ângulo formado com o eixo dos YY, em incrementos  $\Delta \alpha$ .
- A camada que situa ao redor do pólo Norte deverá ser formada por triângulos em vez de quadriláteros; o mesmo se aplica à camada que se situa em redor do pólo Sul.

O algoritmo utilizado deve ser eficiente e deve gerar as coordenadas de textura para a respetiva aplicação à superfície da esfera, como demonstrado na **Figura 2**.



**Figura 2:** Exemplo de textura (disponível no Moodle) e sua aplicação numa esfera

Coloque uma instância de teste da esfera na cena e crie uma tag no repositório neste ponto.

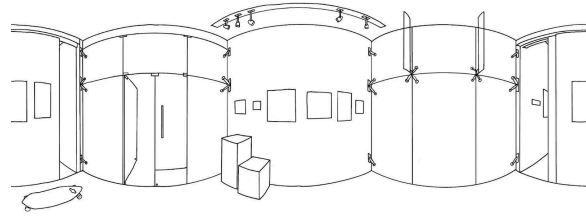


(1)

## 1.2 Adição de Panoramas

Parameterize a classe MySphere para poder inverter as suas faces, de forma a que seja visível por dentro e não por fora (atenção à ordem dos vértices e às respectivas normais). A ideia é usar uma esfera de grandes dimensões à volta da cena com uma imagem panorâmica, para criar a ilusão de uma paisagem, e permitir ao observador deslocar-se **dentro** dessa esfera. Podem encontrar imagens panorâmicas / 360° no formato equirretangular, como as da Figura 3 p.ex. em

<https://www.flickr.com/search/?text=equirectangular%20landscape>



**Figura 3:** Dois exemplos de imagens equirrectangulares

Crie a classe **MyPanorama** que:

- no seu construtor recebe uma **CGFtexture**, e seja responsável por criar uma **MySphere** invertida, dotada de um material apenas com componente emissiva e coberta pela textura mencionada,
- no seu método **display** ative a textura e desenhe a esfera com um raio de 200 unidades.

Inclua na cena um destes panoramas. Sugere-se que experimente diferentes valores para o parâmetro **FoV** (field of view) da câmara da cena, de forma a que a distorção de perspetiva seja satisfatória.

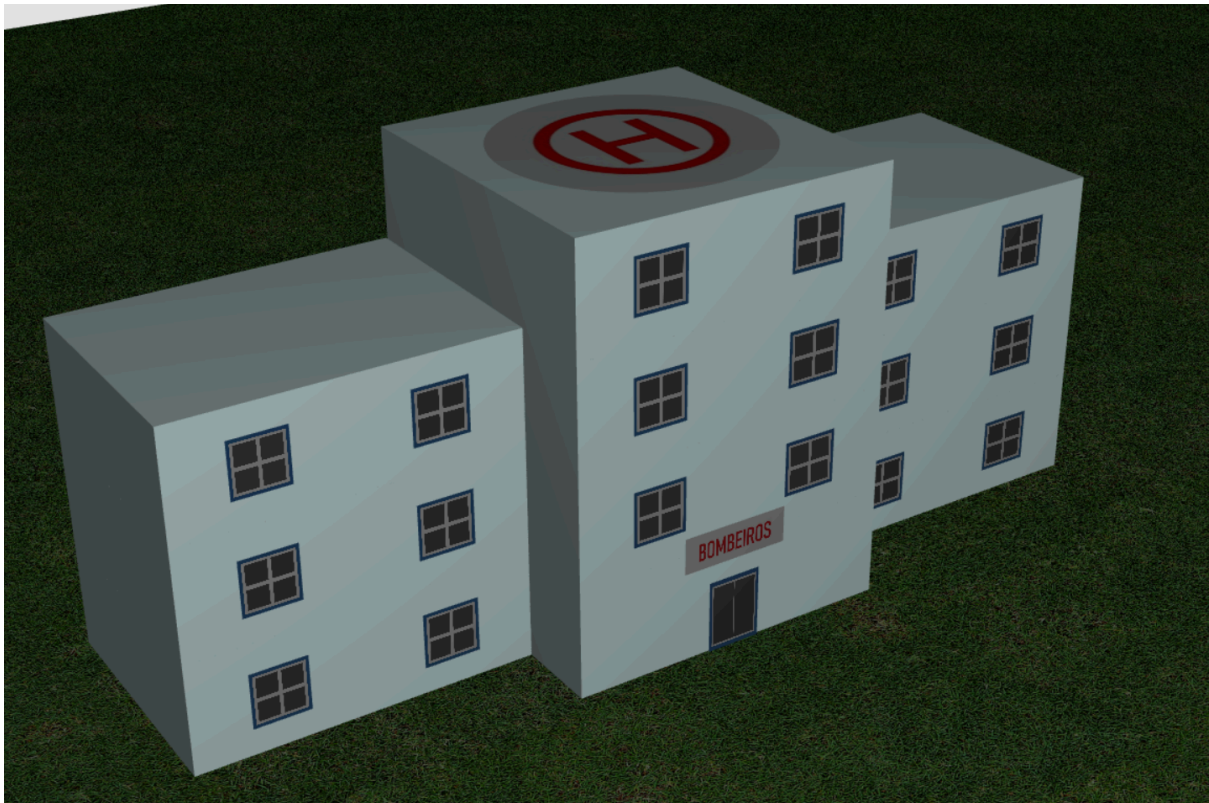
**Situação final:** Altere a classe **MyPanorama** para que fique centrada na posição da câmara, passando a mover-se com a mesma e dando a ilusão de que a superfície esférica se encontra sempre posicionada no infinito. Pode utilizar, para o efeito, o membro **position** da câmara utilizada na **cena** (*this.camera.position*). Trata-se de um **vec3**, pelo que as várias coordenadas podem ser obtidas acedendo às posições **[0]**, **[1]** e **[2]**.

Neste ponto, faça capturas de ecrã que permitam ver uma ou duas perspetivas da cena e do panorama. Crie uma tag no repositório neste ponto.



## 2. Edifício de Bombeiros

Será necessário criar, junto a um canto do plano da cena, um edifício de bombeiros composto por três módulos conexos (ver Figura 4). Deve possuir janelas e uma porta principal, sobre a qual existe um letreiro "BOMBEIROS". O número de andares é parametrizável e o módulo central possui mais um andar do que os outros dois; o teto deve ser plano, de forma a incluir um heliporto (circunferência com uma letra "H" circunscrita). Cada andar possui uma linha com um número parametrizável de janelas, sendo que o R/C do módulo central não deve possuir janelas, apenas a porta e o letreiro.



**Figura 4:** Exemplo simples do edifício dos bombeiros

Para criar o edifício devem definir-se pelo menos duas novas classes: ***MyBuilding*** e ***MyWindow***, de acordo com as subsecções seguintes.

## 2.1 Representação de janelas

Implemente uma classe ***MyWindow*** para representar uma janela baseada em texturas. Os módulos do edifício podem possuir janelas com aspetos diferentes baseadas na classe ***MyWindow***.



## 2.2 Representação do edifício

A classe ***MyBuilding***, que representa o edifício na cena, deverá possuir os seguintes parâmetros:

***MyBuilding***:

- Largura total do edifício (conjunto dos três módulos)
- Número de andares dos módulos laterais
- Número de janelas por andar
- Tipo de janela (textura)
- Cor do edifício

Para construir os módulos laterais, deve usar estes parâmetros relativos ao módulo central como base. Por exemplo: a sua largura e/ou profundidade podem ser 75% da largura do módulo central.

Registe este ponto do desenvolvimento:  (2)  (3)

## 3. Árvores e floresta

### 3.1 Modelação de uma árvore

Pretende-se gerar um modelo simplificado de uma árvore, composta apenas por um tronco, representado por um tronco de cone de material castanho, e a copa, formada por um número variável de pirâmides com materiais verdes. A árvore será representada por uma classe **MyTree**.



Figura 5: Exemplos das árvores pretendidas

### 3.2 Parametrização da árvore

O objecto anterior cria árvores com a mesma estrutura básica; no entanto, o aspeto pode ser variado. Adicione ao construtor da classe **MyTree** parâmetros que permitam controlar o seu aspeto:

#### **MyTree**

Inclinação da árvore:

Rotação em graus (0° corresponde à posição vertical)

Eixo de rotação, **X** ou **Z**

Raio da base do tronco

Altura da árvore

A cor da copa em RGB (as árvores podem assim resultar com tons diferentes de verde)

Para construir a copa, deve usar estes parâmetros para gerar outros que possam ser relevantes. Por exemplo:

A altura da copa pode ser 80% da altura da árvore;

O número de pirâmides a usar pode ser definido de acordo com altura da copa;

Os raios das bases das pirâmides podem ser definidos de acordo com o raio do tronco.

### 3.3. Criação da floresta

Defina uma classe **MyForest** para gerar uma zona arborizada no seu mundo virtual. Crie uma matriz de árvores com um número de linhas e um número de colunas parametrizável. Por exemplo, uma floresta com 5 por 4 será preenchida com um total de 20 árvores. As dimensões da área arborizada não devem ser alteradas.

Recorrendo à função [random\(\)](#) de Javascript, deve ser adicionada alguma aleatoriedade nos valores dos parâmetros da secção 2.2: as árvores serão construídas com parâmetros aleatórios, dentro de intervalos predefinidos, e deverão ter um pequeno offset aleatório na sua posição dentro da matriz.

### 3.4 Texturas nas árvores

A utilização de texturas permite obter um “look and feel” realista sem aumentar a complexidade computacional.

Para o efeito, selecione e atribua texturas aos troncos e às copas das árvores.

Registe este ponto do desenvolvimento:  (3)  (4)

## 4. Helicóptero

Pretende-se que seja incluído um helicóptero na cena. A sua forma 3D pode ser realista e baseada nos usuais aparelhos de combate a incêndios ou definida ao estilo "filme de animação", modelada de forma simplificada e criativa. Quando em voo, apresenta, pendente, um balde de água para apagar incêndios. Deve ser animado e dotado de movimento controlável a partir do teclado de acordo com as subsecções seguintes.

### 4.1 Modelação do helicóptero



1. Crie uma nova classe **MyHeli**. Para modelar o helicóptero, poderá utilizar uma combinação dos diferentes objetos utilizados nos trabalhos anteriores, de forma a que o objeto seja constituído por cabeça com cabine, cauda, hélice superior e hélice traseira. Deve possuir trem de aterragem e deve poder transportar, pendurado, um balde de água.
2. Sugere-se o uso de esferas, elipsóides, cilindros ou outros objetos que considerem relevantes (não devendo no entanto ter um número excessivo de polígonos de forma a não afetar demasiado o "frame rate" da aplicação).
3. No início da aplicação, o helicóptero deverá estar situado no "heliporto" do edifício. Deve ser bem visível quando a cena é iniciada (recorrendo a uma boa definição dos parâmetros iniciais da câmara) de forma a facilitar a avaliação da sua modelação. O seu comprimento deve ser aproximadamente 10 unidades.
4. Os diferentes objetos utilizados para criar o helicóptero deverão possuir materiais e texturas aplicados aos mesmos, adequados às partes que representam.
5. Mostre uma imagem da aparência do helicóptero, obtida de uma distância suficientemente curta que permita ver o mesmo em detalhe) .



(4)

## 4.2 Animação do helicóptero

Neste ponto devem criar-se os mecanismos de animação e controlo para um objeto da classe **MyHeli**. Seja o helicóptero algures na cena, a pairar no ar. O helicóptero deverá ter o seguinte comportamento:

- É controlável pelo utilizador (secção seguinte), podendo movimentar-se para a frente ou para trás; pode ainda rodar em torno de um eixo vertical (mudança de direção);
- Quando em voo, desloca-se a uma altitude fixa, parametrizável; desce daí para aterrar no aeroporto e, sobre o lago, para carregar o balde com água;
- Inclina-se um pouco para a frente quando se usa a tecla de avançar (secção seguinte);
- Inclina-se um pouco para trás quando se usa a tecla de recuar (secção seguinte);
- Ao levantar do heliporto, sobe à altitude de cruzeiro e, automaticamente, deixa sair o balde de água que fica pendente;
- Ao descer para o heliporto, automaticamente recolhe o balde de água.

## 4.3 Controlo do helicóptero

Para poder controlar o movimento do helicóptero na cena, é necessário prever, na interface, a possibilidade de pressionar uma ou mais teclas em simultâneo.

Analise as funções relacionadas com o controlo por teclado nas classes:

**MyInterface:**

```
initKeys()
processKeyDown() / processKeyUp()
isKeyPressed()
```



## MyScene:

```
update()  
checkKeys()
```

Execute o código e verifique as mensagens na consola quando as teclas “**W**” e “**S**” são pressionadas em simultâneo. Outras teclas poderão vir a ser incluídas de acordo com o texto seguinte.

Altere a classe **MyHeli** de acordo com o seguinte:

- Acrescente no construtor variáveis que definam:
  - A posição do helicóptero (x, y, z);
  - A sua orientação (sugestão: ângulo em torno do eixo YY);
  - O vetor velocidade (inicialmente a zero) atualizado com a respetiva direção;
- Altere a função **update** para atualizar a variável de posição em função de "delta\_t" e do vetor velocidade;
- Use as variáveis de posição e de orientação na função **display()** para orientar e posicionar o helicóptero;
- Crie os métodos **turn(v)** e **accelerate(v)** para rodar e para aumentar/diminuir a velocidade do helicóptero:
  - turn(v): afeta diretamente a variável de orientação (ângulo); deve também originar uma atualização do vetor velocidade (só em direção, mantendo a norma);
  - accelerate(v): afeta diretamente a norma do vetor velocidade, devendo manter-se a sua direção.

Preveja que as teclas possam invocar os métodos *turn* e *accelerate* do helicóptero, de forma a implementar o seguinte comportamento:

- Acelerar ou travar com as teclas "**W**" ou "**S**", respectivamente;
- Rodar o helicóptero para a esquerda ou para a direita respetivamente com as teclas "**A**" ou "**D**";
- Fazer "*reset*" à posição e velocidade do helicóptero através da tecla "**R**": o helicóptero deverá ser colocado no heliporto, em posição de repouso.
- Subir o helicóptero: tecla "**P**"
  - Se estiver em repouso no heliporto, inicia o movimento das hélices e levanta vôo na vertical, até atingir a altitude de cruzeiro; a partir daí fica sob o controlo do utilizador;
  - Se estiver a encher o balde de água no lago, sobe verticalmente até à altitude de cruzeiro; a partir daí fica sob o controlo do utilizador (secção 5).
- Descer o helicóptero: tecla "**L**":
  - Se estiver sobre o lago e com o balde de água vazio, desce até que o balde toque na água (secção 5);
  - Se estiver em qualquer outra posição da cena e com o balde de água vazio, inicia um movimento automático para se colocar sobre o heliporto, em posição de aterragem, e desce até pousar; interrompe o movimento das hélices e fica em posição de repouso.

- Crie um *slider* na GUI chamado **speedFactor** (entre 0.1 e 3) que altere a sensibilidade da aceleração e da rotação anteriores, isto é, "acelere e desacelere" a velocidade de deslocamento e a rotação.

Crie uma tag no repositório neste ponto. Registe este ponto do desenvolvimento:  (5)



## 5. Água e incêndio

Pretende-se adicionar uma nova funcionalidade a **MyHeli**, de forma a que o helicóptero apanhe água do lago e a deixe cair sobre o incêndio que ocorre na floresta. Use a representação que considerar adequada para a água quando cai do balde sobre o incêndio.

Criação dos objetos

1. Crie uma nova classe **MyFire**, que representará um incêndio em curso na floresta. As labaredas podem ser simuladas por triângulos de tamanho variável; aplique materiais e texturas adequados para se assemelhar a um incêndio (a animação das labaredas será efetuada mais tarde, com recurso a shaders);
2. Coloque um lago **MyLake** na cena; pode ser representado por um objeto 2D simples, com material / textura adequados;
3. Garanta que o helicóptero, o edifício, o lago e a floresta (com os objetos de **MyFire**) são visíveis no início da cena, para avaliação.

Adicione a **MyHeli** a funcionalidade de apanhar e largar água da seguinte forma:

1. Ao pressionar a tecla “L”, e na condição de o helicóptero se encontrar *parado* sobre o lago, o helicóptero deve descer na vertical até o balde tocar a água;
2. Pressionando “P”, o helicóptero volta a subir, agora transportando água no balde; ao atingir a altitude de cruzeiro, mantém a sua orientação e velocidade nula, assim como volta a reagir às teclas W, S, A e D;
3. Ao pressionar a tecla “O”, e na condição de o helicóptero se encontrar sobre o incêndio, o helicóptero deve abrir o balde de água; esta cai sobre o incêndio, apagando-o.

Registe este ponto do desenvolvimento:  (6)  (6)

## 6. Shaders e animação

Pretende-se nesta secção dar mais realismo à cena, dando-lhe algum movimento, pelo que alguns objetos serão dotados de uma animação elementar:

- Os triângulos que representam as labaredas do incêndio passam a ter geometria animada;

- As manobras de levantar e de aterrar do helicóptero passam a ser assinaladas com animação de texturas na pista do heliporto.

Estas animações serão conseguidas por recurso à programação de shaders.

## 6.1 Ondulação das labaredas

Pretende-se nesta secção animar os polígonos que constituem as labaredas do incêndio. Cada uma deve curvar-se ciclicamente mas com alguma aleatoriedade, dando a ilusão, no conjunto, da ondulação visível num incêndio.

Este efeito pode ser conseguido através da deslocação/oscilação dos vértices dos triângulos que constituem as labaredas. A deslocação deve, para o efeito, ocorrer em todos os triângulos, embora de forma diferente (mais uma vez, alguma aleatoriedade).

Registe este ponto do desenvolvimento:  (7)  (7)

## 6.2 Heliporto em manobras

Pretende-se implementar agora, no heliporto, uma sinalização inovadora, indicadora de ocorrência de manobras do helicóptero: variação da textura do heliporto e utilização de luzes pulsantes:

- Quando em manobra de levantar vôo, a textura utilizada no heliporto deve alternar ("piscar") entre a textura utilizada até aqui (letra H") e uma nova textura, com as letras "UP".).
- Quando em manobra de aterragem, deve proceder-se de forma similar mas com uma textura com as letras "DOWN".
- Devem ser acrescentados quatro pequenos objetos nos quatro cantos do heliporto. Quando em situação de manobra, estes objetos mudam de material, passando a ter propriedades emissivas; a emissividade deve ser variável sinusoidalmente no tempo, de forma a criar a sensação de luzes pulsantes.

## 7. Desenvolvimentos adicionais

Estes desenvolvimentos são para valorização. Por conseguinte, são menos prioritários e possuem um menor peso na avaliação do trabalho.

Das seguintes alternativas, escolha uma (e só uma) para implementação:

- A. Sky-Sphere animada, simulando o céu de uma forma mais realista; para além da textura aplicada anteriormente, deverá ser aplicada uma segunda textura, com nuvens, à esfera criada em **MyPanorama**.

A textura escolhida para as nuvens deverá ser em tons de cinza, com um canal *alpha* de forma a que seja possível "compor" a textura em sobreposição à textura anteriormente aplicada (ver, por exemplo, a textura **waterMap** da TP5).

Para aplicar as duas texturas em simultâneo, será necessário recorrer a *shaders* que manipulem as coordenadas da nova textura de forma a que as nuvens se movimentem no céu. O *fragment shader* deverá determinar a cor do fragmento tendo em consideração as duas cores, da textura base e da nuvem. Poderá ser necessário trabalhar com mais detalhe as coordenadas de textura de forma a obter um resultado mais realista. Por exemplo, deverá evitar-se que as nuvens desçam abaixo de um determinado nível das stacks da esfera para não se sobreporem ao "chão".

- B. Consoante o tipo de manobra, o heliporto deve alternar entre duas texturas com a letra "H" e com as letras "DOWN" ou "UP", mas misturando as texturas nas transições. A mistura deve ser efetuada com base em *shaders*.
- C. Substituir a representação poligonal do lago: fazer uma máscara preto e branco de todo o terreno (400x400); pintar a máscara de preto na área que se pretenda ser o lago (formato livre); com recurso a *shaders*, cada pixel do terreno é "pintado":
  - a. Na área branca da máscara: com uma representação de "relvado" (textura), como anteriormente;
  - b. Na área preta da máscara: com uma representação de "água", representando o lago; a superfície da água deve apresentar ondulações (geometria) e texturas animadas de forma a tornar-se mais realista.

Submeta o código final.



(8)



(8)

## Notas sobre a avaliação do trabalho

A classificação máxima a atribuir a cada alínea corresponde a um desenvolvimento ótimo da mesma, no absoluto cumprimento com todas as funcionalidades enunciadas. Sem perda da criatividade desejada num trabalho deste tipo, **não serão contabilizados**, para efeitos de avaliação, **quaisquer outros desenvolvimentos** além dos que são pedidos, **nomeadamente como forma de compensar componentes em falta**.

No item **Aspeto geral e criatividade da cena**, será avaliada a forma como a **utilização de técnicas** definidas **neste trabalho e nos trabalhos anteriores** permitem um **maior grau de inovação, originalidade, assim como a complexidade e o cuidado visual da aplicação**.

No item **Qualidade do software** ter-se-ão em consideração a **eficiência do código, a sua estruturação, legibilidade e a utilização de comentários** para efeitos de documentação.

Os critérios a usar na avaliação e respectivos pesos são os seguintes:

### **Sky-sphere [1.0]**

Criação de esfera [0.5]

Adição de Panoramas [0.5]

### **Edifício de Bombeiros [2.5]**

Representação de janelas [1.0]

Representação do edifício [1.5]

### **Árvores e floresta [3.0]**

Modelação de uma árvore [0.5]

Parametrização da árvore [0.5]

Criação da floresta [1.0]  
Texturas nas árvores [1.0]

**Helicóptero [4.0]**

Modelação do helicóptero [1.0]  
Animação do helicóptero [1.5]  
Controlo do helicóptero [1.5]

**Água e incêndio [2.0]**

**Shaders e animação [1.5]**

Ondulação das labaredas [1.0]  
Heliporto em manobras [0.5]

**Desenvolvimentos adicionais [1.5]**

**Aspeto geral e criatividade da cena [2.0]**

**Qualidade do software [2.5]**

## Proposta de plano de trabalho

Recomenda-se o seguinte plano de trabalho ao longo do tempo restante do semestre (datas de início e término de tarefa):

Pontos do enunciado	Turm. 2a f.	Turm. 3a f.	Turm. 4a f.	Turm. 5a f.
Sky-Sphere + Edifício	31/3 -> 6/4	1/4 -> 7/4	2/4 -> 8/4	3/4 -> 9/4
Edifício (concl.)+Árvore	7/4 -> 13/4	8/4 -> 14/4	9/4 -> 15/4	10/4 -> 16/4
---	14/4 -> 20/4	15/4 -> 21/4	16/4 -> 22/4	17/4 -> 23/4
Floresta + Helicóptero	21/4 -> 27/4	22/4 -> 28/4	23/4 -> 29/4	24/4 -> 30/4
Helicóptero (concl.)	28/4 -> 4/5	29/4 -> 5/5	30/4 -> 6/5	1/5 -> 7/5
---	5/5 -> 11/5	6/5 -> 12/5	7/5 -> 13/5	8/5 -> 14/5
Água e Incêndio	12/5 -> 18/5	13/5 -> 19/5	14/5 -> 20/5	15/5 -> 21/5
Shaders e Animação	19/5 -> 25/5	20/5 -> 26/5	21/5 -> 27/5	22/5 -> 28/5
Desenvolvimentos Adicionais	26/5 -> 29/5	27/5 -> 30/5	28/5 -> 31/5	29/5 -> 1/6

Durante a semana de aulas de 2 a 6 de junho, far-se-á a avaliação final dos trabalhos, em horários especiais a serem anunciados.

## Checklist

Até ao final do trabalho deverá submeter as seguintes imagens e versões do código via Moodle, **respeitando estritamente a regra dos nomes**:



**Imagens:** (nomes do tipo "project-t<turma>g<grupo>-n.png")



**GIT Commits/Tags:** (formato "proj-1", "proj-2", ...)

## Vídeo

Deverá também preparar um **vídeo de 1 minuto em mp4**, a ser submetido em área própria a ser indicada no Moodle. O vídeo deve ser capturado do ecrã demonstrando todas as funcionalidades implementadas (nomes do tipo "project-t<turma>g<grupo>.mp4").