

Computação Gráfica (MIEIC)

Trabalho Prático 6


Projeto Final



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

Trabalho prático

Ao longo dos pontos seguintes são descritas várias tarefas a realizar. Algumas delas estão anotadas

com o ícone  (captura de imagem). Nestes pontos deverão, com o programa em execução, capturar uma imagem da execução. Devem nomear as imagens capturadas seguindo o formato "CGFImage-tp6-TtGgg-x.y.png", em que TtGgg referem-se à turma e número de grupo e x e y correspondem ao ponto e subponto correspondentes à tarefa (p.ex. "CGFImage-tp6-T3G10-2.4.png", ou "CGFImage-tp6-T2G08-extra.jpg").

Nas tarefas assinaladas com o ícone  (código), devem criar um ficheiro .zip do vosso projeto, e nomeá-lo como "CGFCode-tp6-TtGgg-x.y.zip", (com TtGgg, x e y identificando a turma, grupo e a tarefa tal como descrito acima). Quando o ícone  surgir, é esperado que executem o programa e observem os resultados. No final, devem submeter todos os ficheiros via Moodle, através do link disponibilizado para o efeito. Devem incluir também um ficheiro **ident.txt** com a lista de elementos do grupo (nome e número). Só um elemento do grupo deverá submeter o trabalho.

Preparação do Ambiente de Trabalho

Este trabalho deve ser baseado nas classes e nos vários elementos básicos criados nas aulas anteriores (**LightingScene.js**, planos, cubos, semi-esferas e cilindros texturados).

1. Modelação do veículo

Pretende-se modelar um veículo automóvel de quatro rodas, que pode ser um veículo desportivo, jipe, buggy, carrinha ou outro, que deve ser composto por vários elementos, e obedecer às seguintes restrições/limites:

- Comprimento: entre 4.0 e 5.0 unidades
- Distância entre eixos: entre 2.0 e 3.5 unidades
- Diâmetro das rodas: entre 0.7 e 1.1 unidades
- Largura: entre 1.8 e 2.5 unidades
- Altura: entre 1.2 e 2.0 unidades

1. Crie uma classe **MyVehicle** que será responsável pelo desenho do veículo, e deve contemplar pelo menos os seguintes componentes:



- Duas rodas traseiras, de orientação fixa (posteriormente serão animadas para rodarem para a frente e para trás)
- Duas rodas dianteiras (posteriormente serão animadas não só para rodar para a frente e para trás, mas também para virarem para a esquerda e para a direita)
- Carroçaria, que deve incluir elementos a simular os vidros, espelhos e faróis (mas sem usar transparências ou reflexões). Pode ser descapotável ou de caixa aberta

Ao nível de geometria no veículo a utilizar, devem ser incluídos pelo menos os seguintes elementos, todos com suporte para texturas:

- Dois ou mais trapézios diferentes
- Semi-esferas
- Cilindros
- Cubos

Podem criar outros objetos que considerem relevantes para a representação do veículo.

2. Ambiente inicial

1. Crie uma classe **MyTerrain** que representará o terreno onde o veículo se deslocará. Esta classe será uma sub-classe de **Plane**, com uma textura associada, que pode ter repetição ativa.
2. Instancie um objeto de **MyTerrain** com 50 x 50 unidades.
3. Altere a cor de fundo da cena para azul céu.
4. Instancie um objeto do tipo **MyVehicle** numa posição visível na cena. (2.4 ) (2.4 )

3. GUI

Neste exercício procura-se criar uma interface gráfica (GUI) com alguns controlos para alterar parâmetros da cena em tempo de execução.

Para esse efeito, iremos acrescentar uma classe de interface que criará uma área de interface gráfica com alguns elementos de interação, e que será também responsável por gerir eventos de teclado. Ao longo do enunciado serão dadas instruções específicas sobre este aspeto.

1. Aceda ao ficheiro **MyInterface.js** disponibilizado no Moodle, e inclua-o no projeto da seguinte forma:
 - a. Coloque o ficheiro na mesma diretoria dos restantes ficheiros Javascript do projeto
 - b. Edite o ficheiro **main.js** e
 - i. adicione '**MyInterface.js**' à lista de ficheiros a incluir
 - ii. substitua no código da função **main** a referência a **CGFInterface** por **MyInterface**
 - c. Edite o ficheiro de cena (**LightingScene.js**) e
 - i. acrescente no método **init**, as seguintes variáveis:

- ```
this.option1=true; this.option2=false; this.speed=3;
```
- ii. acrescente ao ficheiro **LightingScene.js** o seguinte método:
- ```
doSomething()  
{ console.log("Doing something..."); };
```

2. Consulte a classe **MyInterface.js** para ver exemplos de utilização que ajudarão na resolução destes pontos. Estude a biblioteca javascript **dat.GUI** através dos seguintes exemplos: <https://workshop.chromeexperiments.com/examples/gui/#1--Basic-Usage>
3. Adicione à GUI um grupo intitulado "Luzes" (remova/comente/substitua o grupo do exemplo). Acrescente ao novo grupo, por cada fonte de luz utilizada, uma checkbox. Cada checkbox (estado on/off) deve permitir alterar o estado (respetivamente acesa/apagada) da fonte de luz que lhe diz respeito.



4. Adicione um botão que ative/desative o desenho dos eixos da cena

(3.4 ) (3.4 ) 

4. Controlo e Animação do Veículo

Neste exercício procura-se criar um mecanismo de controlo para um objeto da classe **MyVehicle**. Em primeiro lugar será necessário acrescentar código para detetar o pressionar de uma ou mais teclas em simultâneo.


1. Altere a classe **MyInterface.js**, substituindo o método *processKeyboard(event)* pelos seguintes métodos capazes de processar várias teclas ao mesmo tempo:

```
initKeys() {  
    this.scene.gui=this;  
    this.processKeyboard=function(){};  
    this.activeKeys={};  
}  
processKeyDown(event) {  
    this.activeKeys[event.code]=true;  
};  
  
processKeyUp(event) {  
    this.activeKeys[event.code]=false;  
};  
  
isKeyPressed(keyCode) {  
    return this.activeKeys[keyCode] || false;  
}
```

No final da função *init* do **MyInterface** chame a função *initKeys()*.

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

```
checkKeys()  
{  
    var text="Keys pressed: ";  
    var keysPressed=false;  
  
    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. Crie um mecanismo para controlar o veículo utilizando as teclas da seguinte forma:
- Avançar ou recuar conforme se pressione "W" ou "S", respectivamente. O veículo mover-se-á enquanto a tecla estiver pressionada, e parará imediatamente quando for libertada.
NOTA: ver secção “8. Valorização” para uma versão melhorada
 - As rodas da frente deverão virar em torno do eixo do YY de acordo com as teclas “A” e “D”.
NOTA: ver secção “8. Valorização” para uma versão melhorada
 - Caso as teclas “A” ou “D” estejam pressionadas quando o veículo se estiver a deslocar, o veículo deve avançar (ou recuar) virando para a esquerda ou direita, respectivamente.
 - Por forma a simular o movimento das rodas estas deverão girar com uma velocidade correspondente à velocidade do veículo.

Para este efeito, deve criar as variáveis e código necessários para alterarem a posição e orientação do veículo.

(4.3  )

5. Texturas

Neste exercício procura-se colocar texturas e controlá-las a partir da GUI.

1. Construa uma interface para a seleção das texturas, integrada na GUI da aplicação. Deve para o efeito usar um controlo do tipo “drop-down”. Para implementar este tipo de controlos, sugere-se:
 - a. Declarar na cena um array **vehicleAppearances** que contenha as várias appearances possíveis
 - b. Declarar um dicionário **vehicleAppearanceList** que mapeie as strings identificando cada appearance ao seu índice em **vehicleAppearances**.
 - c. Declarar na cena uma variável **currVehicleAppearance** que identifique o índice da appearance selecionada/atual
 - d. Adicionar um controlo na interface que fique associado a **currVehicleAppearance** e a **vehicleAppearanceList**
(exemplos em <http://workshop.chromeexperiments.com/examples/gui/#2--Constraining-Input>)
 - e. Ajustar o código de desenho da cena ou do veículo para que seja usada a *appearance* correta.
2. Execute e altere as texturas usando a GUI.

(5.2 ) (5.2 ) 

6. Modelação do Terreno

Neste exercício pretende-se modelar um terreno com altimetria, modificando a classe **MyTerrain** para que os vértices do plano tenham uma altura variável.

1. Considere que se pretende modelar um terreno como o da Figura 1, considerando uma grelha com 8x8 divisões (9x9 vértices)

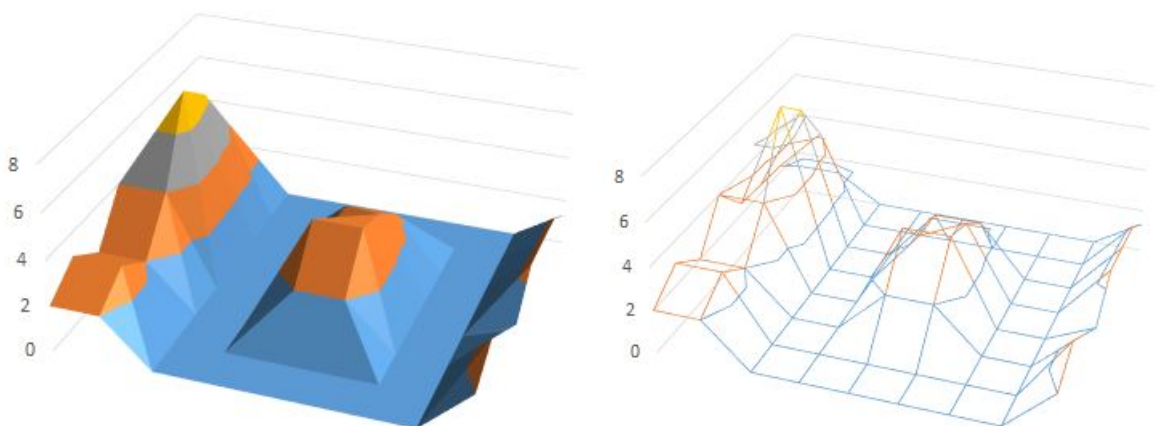


Figura 1: Exemplo de terreno.

A informação de altimetria deste terreno pode ser declarada numa matriz de 9x9 elementos tal como a representada a seguir.

```
//example for nrDivs = 8 -> grid of 9x9 vertices
this.altimetry= [[ 2.0 , 3.0 , 2.0, 4.0, 2.5, 2.4, 2.3, 1.3 ],
                 [ 2.0 , 3.0 , 2.0, 4.0, 7.5, 6.4, 4.3, 1.3 ],
                 [ 0.0 , 0.0 , 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ],
                 [ 0.0 , 0.0 , 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ],
                 [ 0.0 , 0.0 , 2.0, 4.0, 2.5, 2.4, 0.0, 0.0 ],
                 [ 0.0 , 0.0 , 2.0, 4.0, 3.5, 2.4, 0.0, 0.0 ],
                 [ 0.0 , 0.0 , 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ],
                 [ 0.0 , 0.0 , 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ],
                 [ 2.0 , 3.0 , 2.0, 1.0, 2.5, 2.4, 2.3, 1.3 ]
                ];
```

Acrescente esta matriz à inicialização da cena.

2. Altere o construtor da classe **MyTerrain** para aceitar, além do parâmetro *nrDivs* já previsto na classe **Plane**, uma matriz *altimetry* de tamanho $(nrDivs+1) \times (nrDivs+1)$. A criação de **MyTerrain** na inicialização da **LightingScene** passará então a ser algo semelhante a:

```
this.terrain = new MyTerrain(this, 8, this.altimetry);
```

NOTA: As dimensões aqui fornecidas são um exemplo, a classe **MyTerrain** deverá ser genérica e funcionar para diferentes valores de *nrDivs*.

3. Crie a sua própria altimetria e adeque a textura do terreno ao relevo. Coloque o veículo numa zona plana com altitude 0. **NOTA: ver secção “8. Valorização” para uma versão melhorada.**

(6.3 ) (6.3 ) 

7. Guindaste Magnético

1. Crie uma classe **MyCrane** que representa um guindaste articulado com um íman como representado na Figura 2. Os eixos dos cilindros representados na base do guindaste e na articulação entre os braços correspondem aos eixos de rotação dos pontos respectivos: a base roda em torno do eixo vertical, a articulação entre os braços roda em torno de um eixo horizontal. O íman e o cabo que o segura mantêm sempre a sua orientação (cabo e eixo do íman sempre verticais). Crie os métodos necessários para definir a rotação de ambas as articulações.

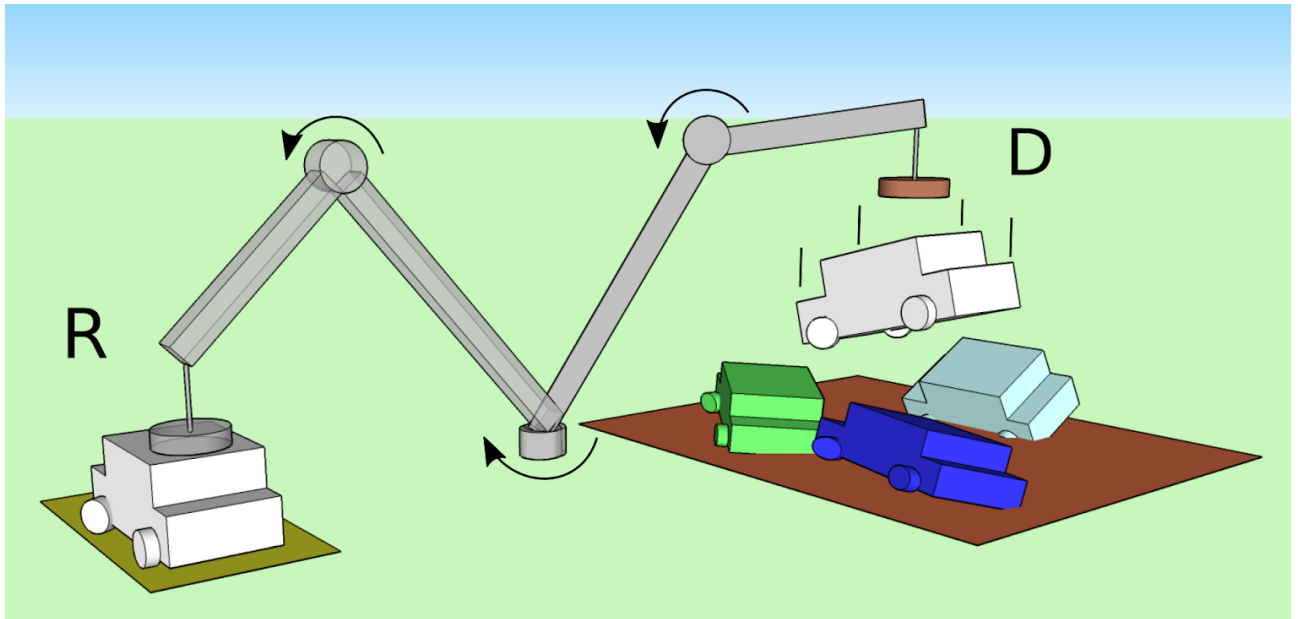


Figura 2: Guindaste magnético em duas posições diferentes.

2. Defina duas posições à volta do guindaste: a posição de depósito (D), onde são colocados os carros, e a posição de recolha, onde ele poderá apanhar o veículo (R). Crie uma animação entre essas duas posições, em ambas as direções (D para R e R para D).
3. O guindaste deve estar normalmente na posição D. Quando o veículo parar na posição R, o guindaste deverá deslocar-se de D para R, “prender” o veículo e deslocar o mesmo para a posição D, largando-o.

Sugestão de implementação: quando o guindaste “prender” o carro, o controlo do desenho do carro deve deixar de feito diretamente no display da cena, e passar a feito pelo display do ímã do guindaste. Desta forma, o carro será automaticamente afetado pelas transformações do ímã.

(7.3 ) (7.3 ) 

8. Valorização (máximo 2 valores)

Este é um desenvolvimento de valorização; possui uma cotação relativamente baixa, devendo ser dada prioridade aos restantes desenvolvimentos solicitados no enunciado. A valorização terá em conta a criatividade, qualidade e coerência da sua implementação.

Pretende-se tornar mais realista a animação do veículo. Para tal sugere-se os seguintes desenvolvimentos de valorização, alternativos para um máximo de 2 valores:

- Quando o utilizador parar de virar, (largar as teclas “A” ou “D”) a direcção deverá estabilizar progressivamente na direcção da frente. (1 valor)
- As teclas “W” e “S” controlam a velocidade do veículo, em vez da sua posição diretamente. Ou seja, o veículo manterá a sua velocidade se nenhuma daquelas teclas for pressionada, e a velocidade aumentará de cada vez que “W” for pressionada, e diminuirá de cada vez que “S” for pressionada (podendo ser negativa, andando assim para trás). (1 valor)
- O veículo apenas poderá andar nas zonas planas de altitude 0. Crie os métodos e funções necessárias nas classes MyTerrain e MyVehicle para que o veículo não possa deslocar-se para

as zonas com relevo (sugestão: a posição do veículo deve estar mapeada à matriz de altimetria). (2 valores)

- Qualquer outro extra deverá ser discutido com o docente das aulas práticas.

9. Qualidade de Software (1 valor)

Todo o código desenvolvido deverá cumprir com as melhores práticas da programação.

Nomeadamente o trabalho será valorizado de acordo com:

- A boa estruturação do código;
- A boa documentação através de comentários;
- A eficiência das rotinas mais críticas em termos de tempo de cálculo;
- A limpeza e legibilidade do código.

Notas sobre a avaliação do trabalho:

O enunciado incorpora, em cada alínea, a sua classificação máxima, correspondendo esta a um ótimo desenvolvimento, de acordo com os critérios seguintes, e que cumpra com todas as funcionalidades enunciadas.

Desenvolvimentos além dos que são pedidos apenas serão considerados no critério “Valorização” referido abaixo, não compensando nunca a falta ou falha de funcionalidades expressamente referidas no enunciado.

Para efeitos de avaliação do trabalho e tendo em atenção as cotações mencionadas anteriormente, serão considerados os seguintes critérios:

- Modelação do veículo e Ambiente (4 valores)
- GUI (1 valor)
- Controlo e animação do veículo (4 valores)
- Texturas (2 valores)
- Terreno (3 valores)
- Guindaste (3 valores)
- Valorização (2 valores)
- Software (1 valor):

De acordo com a formulação constante na ficha de disciplina, a avaliação deste trabalho conta para a classificação final com um peso de:



$50\% * 40\% = 20\%$ da nota final.

Discussão do trabalho

A avaliação do trabalho decorrerá preferencialmente durante a última aula prática ou em data a indicar pelo docente das práticas. Consistirá numa apresentação/discussão de 10 minutos de cada grupo com o respetivo docente das aulas práticas.

Checklist

Até ao final do trabalho deverá submeter as seguintes imagens e versões do código via Moodle, respeitando estritamente a regra dos nomes, bem como o ficheiro ident.txt com a identificação dos membros do grupo:

-  Imagens (6): 2.4, 3.4, 5.2, 5.5, 6.3, 7.3
(nomes do tipo " CGFImage-tp6-TtGgg-x.y.png")
-  Código em arquivo zip (7): 2.4, 3.4, 4.3, 5.2, 5.5, 6.3, 7.3
(nomes do tipo "CGFCode-tp6-TtGgg-x.y.zip")