



Computação Gráfica (MIEIC)

2021/2022

Projeto Final

v1.0 (2022/04/18)

Objetivos

- Aplicar os conhecimentos e técnicas adquiridas até à data
- Criação de uma cena complexa, com diferentes geometrias, materiais, texturas e luzes
- Exploração de técnicas de 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 praticamente vazia, apenas com um plano de base a servir de referência/terreno. Terá posteriormente de adicionar alguns dos objetos criados anteriormente.


A cena será no final genericamente constituída por:

- Uma representação de uma linha de comboio
- Duas ou mais estações
- Uma locomotiva que percorre a pista, parando nas estações
- Um guindaste na locomotiva, controlado pelo teclado, que carrega madeira nas estações
- Uma paisagem envolvente (*cube map*)
- Outros elementos de ambiente
- Alguns controlos na interface 2D

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.



necessário fazer capturas de ecrã em alguns pontos do enunciado, bem como assinalar versões do código no *Git* com *Tags*. Os pontos onde tal deve ser feito estão assinalados ao longo do documento e listados numa check list no final deste enunciado, sempre assinalados com os ícones 

(captura de uma imagem) e  (tags apenas). **Tal como nos trabalhos anteriores, apenas haverá uma submissão de código no Moodle no final.**

Todas as imagens capturadas devem ser incluídas no ficheiro README.md do projeto.

1. *Criação de carris*

O primeiro elemento a ser criado é um conjunto de carris, cujo caminho é definido programaticamente, seguindo a lógica de **polilinhas - conjunto de linhas retas que conectam múltiplos pontos**. Os caminhos criados serão sempre circuitos fechados, e ao longo do caminho deverão existir dois ou mais pontos que funcionarão como estações (ver ponto 1.1).

1.1. *MyTrack - Carris*

Crie uma nova classe **MyTrack**, que permite definir um caminho de carris representado por uma **lista de pontos**, que definem uma **polilinha**.

Estes pontos, para além de definirem uma determinada **posição no plano XZ ($Y = 0$)**, identificam também se esse ponto do caminho deverá ter uma **estação** ou não (a ser modelada em passos seguintes). Para realizar o **circuito fechado**, o último ponto será automaticamente ligado ao primeiro ponto da lista, criando um segmento de recta final. Não devem existir duas estações em pontos consecutivos.

Segue-se um exemplo de uma possível lista de pontos, em que o último segmento de recta, a verde, corresponde à ligação entre o primeiro e último ponto:

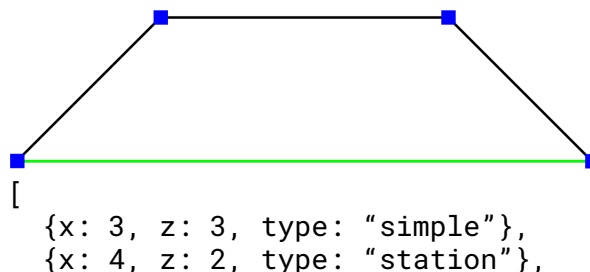




Figura 1: Exemplificação de *polilinha* com estes pontos.

1.2. *MyTrackSegment* - *Modelação de base*

O caminho de carris definido pela lista de pontos fornecida a ***MyTrack*** é representado por um conjunto de quadriláteros, cada um associado a um segmento de recta do caminho.

Sugere-se que crie uma nova classe ***MyTrackSegment*** que facilite a definição de cada segmento. Nesta classe, utilize uma primitiva quadrangular que permita ajustar as coordenadas de textura (e.g., ***MyQuad*** ou ***MyPlane***).

Utilize a **distância e posição relativa** do ponto inicial e final de um segmento para escalar, rodar e posicionar o objeto adequadamente. Para a largura de cada quadrilátero, utilize 4 unidades (pode aplicar como escalamento).



Figura 2: Exemplificação da polilinha com os segmentos de recta representados por planos.

Nota: Segmentos de reta consecutivos com orientações muito diferentes resultam em “**quebras**” no caminho, como pode visualizar na figura 2. Sugere-se que sejam evitadas variações bruscas de orientação entre segmentos, e que usem um maior número de segmentos para representar curvas mais adequadamente.

1.3. *MyTrack* - Texturas

O caminho deverá ter uma textura aplicada que represente os carris, como a imagem fornecida com o código de base do projeto. Deverá definir as coordenadas de textura dos planos de forma a que a textura seja mapeada de forma semelhante nos diferentes segmentos de recta, considerando a dimensão do segmento. A **figura 3** demonstra um possível caminho de carris.

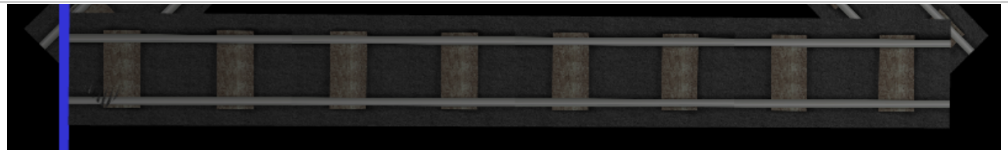


Figura 3: Exemplificação do caminho descrito anteriormente com textura aplicada.
Crie uma imagem e tag no repositório neste

ponto.  (1)  (1)

2. Criação de base da locomotiva

Nesta secção, serão criadas as primitivas necessárias para modelar a locomotiva, e outros elementos da cena a utilizar em secções futuras.

2.4. MyCircle - Círculo

Para ser aplicado em diferentes elementos, pretende-se a criação de uma primitiva **MyCircle**, que aproxima um círculo através de um polígono de N lados (fig. 4a). A primitiva deve receber como parâmetro no seu construtor o nº de lados do polígono que servirá de aproximação, e gerar um conjunto de triângulos de igual número, todos partilhando o vértice central. Devem também ser geradas coordenadas de textura para os diferentes vértices, de forma a ser mapeado na geometria a área de um círculo inscrito na textura (fig. 4b e 4c).

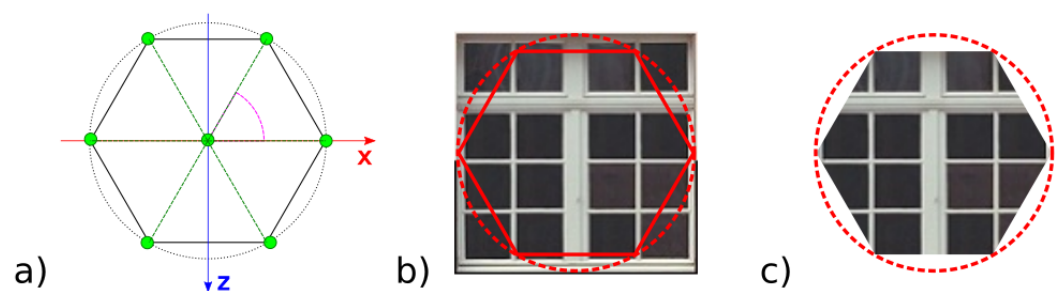



Figura 4: Círculo aproximado com polígono.

a) Vértices/triângulos de exemplo para N=6

b) Mapeamento de textura quadrada

c) Resultado final

Crie uma imagem de um exemplo do círculo.  (2)

2.5. MyCylinder - Cilindro sem topos

Crie uma nova classe **MyCylinder** que aproxime um cilindro de raio de uma unidade,



Shading de Gouraud, como demonstrado na **figura 5**.

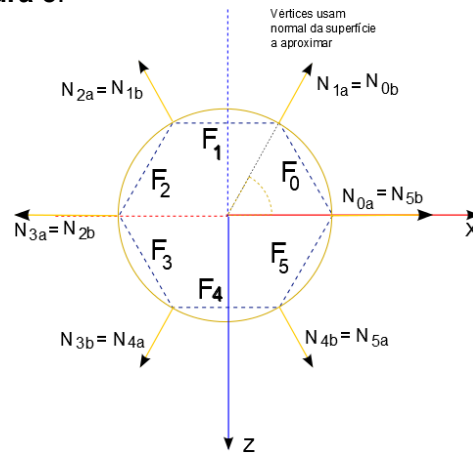



Figura 5: Ilustração das normais a atribuir a cada vértice no caso de um cilindro aproximado por um prisma de seis lados.

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, simultaneamente, o início e o fim da textura.

Crie uma imagem de um exemplo do cilindro.  (3)

2.6. *MySphere - Esfera*

No código fornecido encontra uma classe **MySphere** correspondente a uma esfera com o centro na origem, 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 6** 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 7**.

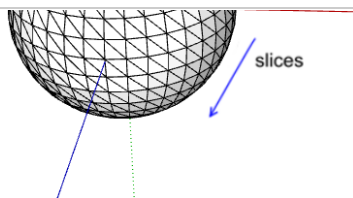


Figura 6: Imagem exemplo de uma esfera centrada na origem.

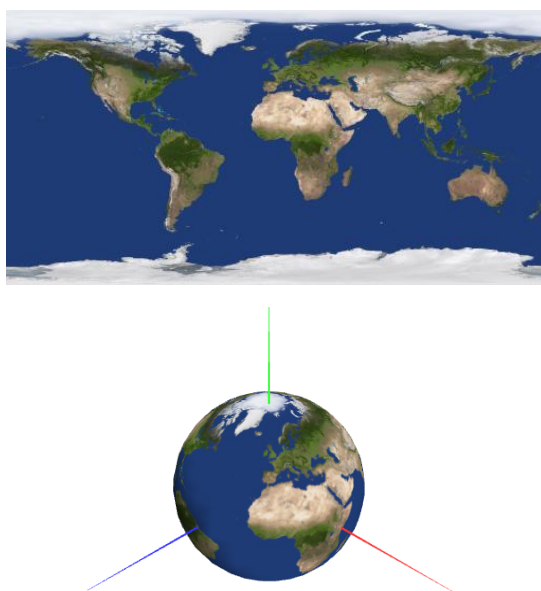


Figura 7: Exemplo de textura (disponível no código base) e sua aplicação numa esfera

Crie uma imagem exemplo da sua esfera.



(4)

2.7. *MyTrainModel* *Locomotiva*

Crie um modelo simples de uma locomotiva, utilizando as geometrias criadas nos pontos anteriores, bem como outras criadas anteriormente, como **MyUnitCubeQuad**. Pode ver um modelo indicativo do pretendido no seguinte link:
<https://www.tinkercad.com/things/5bmY6Jwr5e1-projeto-cgra-20212022>

Este modelo de base será usado nas fases seguintes do projeto. Na secção 5, serão acrescentados mais detalhes a este e outros elementos.



- As rodas devem ser constituídas por um cilindro com um círculo como “tampa”, ter um diâmetro de 1.5 unidades, e espessura de 0.2 unidades
- O paralelepípedo de base deve ter as dimensões de 2.5 x 1 x 7.5 unidades (LxAxC), e estar a 1 unidade do chão
- O corpo cilíndrico deve ter um raio de 0.9 unidades, e comprimento de 3.5 unidades
- A cabine deve ter 2.0 x 2.5 x 1.8 (LxAxC)
- A chaminé deve ser um cilindro sem topo (dimensões à escolha)
- A frente arredondada do corpo cilíndrico deverá ser uma esfera escalada (achatada)

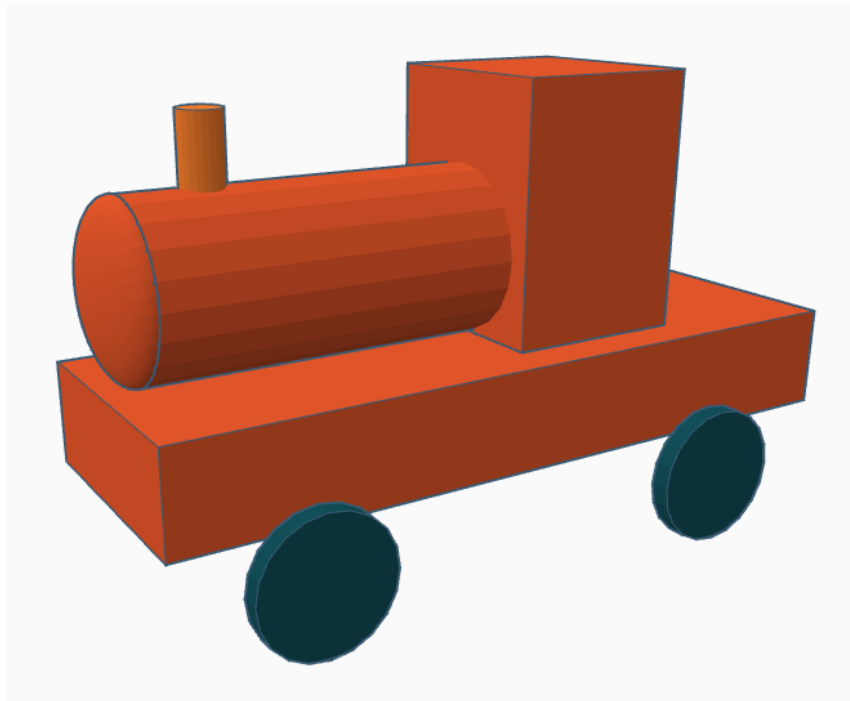


Figura 8: Exemplificação de um modelo básico da locomotiva.

Crie uma imagem da locomotiva e uma tag

no repositório neste ponto.



(5)



(2)

3. Criação de um cubemap para o ambiente

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

- um cubo de grandes dimensões (bastante superiores à da cena visível),
- com material dotado de componentes especular, ambiente e difusa nulas, e

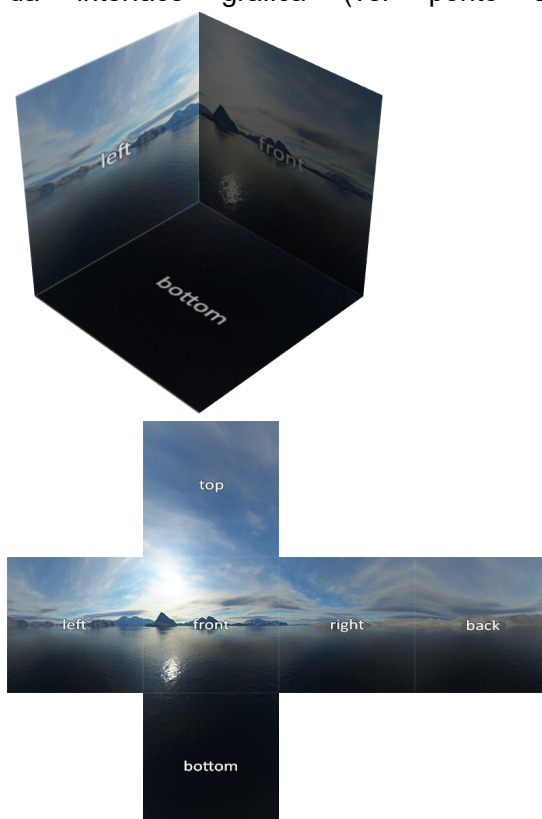


3.1. MyCubeMap

Para este efeito devem usar como base o **MyUnitCubeQuad** e seis texturas diferentes, uma para cada face do cubo como mostra a **Figura 9**. Crie uma cópia do ficheiro da classe **MyUnitCubeQuad**. Modifique essa cópia para criar uma classe **MyCubeMap**, de forma a ser visível por dentro, e com a aplicação da respectiva textura.

O *cube map* deve ser unitário e, ao ser usado na cena, deve ser escalado de forma a medir **50 unidades de lado**. Adicionalmente, deve ser desenhado fazendo coincidir o centro da sua base com a posição da câmara. Pode utilizar para o efeito o membro *position* da câmara guardada na cena: `this.camera.position[0], ...[1], ...[2]`.

O código de base inclui dois exemplos de conjuntos de 6 imagens para mapear nos seis *quads*. O trabalho final deve possuir pelo menos mais uma paisagem envolvente à escolha, para permitir depois ao utilizador escolher entre essa e a de exemplo através da interface gráfica (ver ponto 2.2).







3.2. Controlo do *cube map* na interface

Acrescente os seguintes controlos adicionais na interface gráfica (GUI), para poder parametrizar a aparência do *cube map* e velocidade da locomotiva:

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).

Crie uma imagem da cena e GUI, e uma tag no repositório neste ponto.  (6)  (3)

4. Animação da locomotiva

O comboio deslocar-se-á sobre a “pista”, seguindo a ordem dos pontos que a definem, e repetindo ao chegar ao início, em ciclo. A velocidade a que se desloca depende do tipo de pontos que limitam o segmento em que se encontra. A velocidade máxima - **velocidade de cruzeiro** - deverá ser de 0.01 unidades por segundo.

Identificam-se os seguintes estados possíveis:

- **Parado na estação:** sempre que chega a uma estação, a locomotiva deve parar até a madeira ser carregada ou ser pressionada a tecla **C** (ver ponto 4; temporariamente e para efeitos de teste na fase inicial, pode arrancar imediatamente após parar)
- **Em aceleração:** quando deixa a estação, acelera de forma a atingir a velocidade de cruzeiro no ponto seguinte.
- **Em velocidade de cruzeiro:** mantém a velocidade de cruzeiro.
- **Em desaceleração:** durante um segmento que termina numa estação, desacelera de forma a parar ao chegar à estação.

Relembra-se que não podem existir duas estações seguidas, ou seja, deve haver sempre pelo menos um ponto simples entre elas.

Sugere-se a criação de uma máquina de estados que seja atualizada de cada vez que a locomotiva começa um novo segmento, dependendo dos tipos dos pontos de início e fim.



Pretende-se colocar sobre a cabine da locomotiva um guindaste que permita carregar lenha nas estações enquanto a locomotiva está parada, e que seja controlada pelo teclado. Para esse efeito, deve ser desenvolvida uma primitiva **MyCrane** com as seguintes características (ver Fig. 10):

- **Composição** (cores apenas para identificação na imagem, podem ser outras)
 - Eixo principal, à volta do qual roda todo o guindaste (verde)
 - Braço que se inclina rodando na vertical em torno do ponto de contacto com o eixo principal para levantar/baixar a carga (azul)
 - Cabo com ponta de contacto, que liga a ponta do braço à madeira para a levantar (amarelo/roxo), e está sempre na vertical
- **Controlo**
 - Teclas **A** e **D** rodam o guindaste em torno do eixo vertical
 - Teclas **W** e **S** inclinam o braço para cima e para baixo respetivamente
 - Tecla **P** “agarra” a madeira (castanho) que se encontra na estação, desde que a ponta do cabo esteja próxima da mesma
 - A mesma tecla **P** larga a madeira no depósito na locomotiva, desde que a mesma esteja próxima do mesmo (laranja), permitindo à locomotiva partir
 - A tecla **R** faz “reset” e coloca o guindaste na posição inicial.
 - Para efeitos de teste, a tecla **C** permite a locomotiva partir sem ter feito a carga.

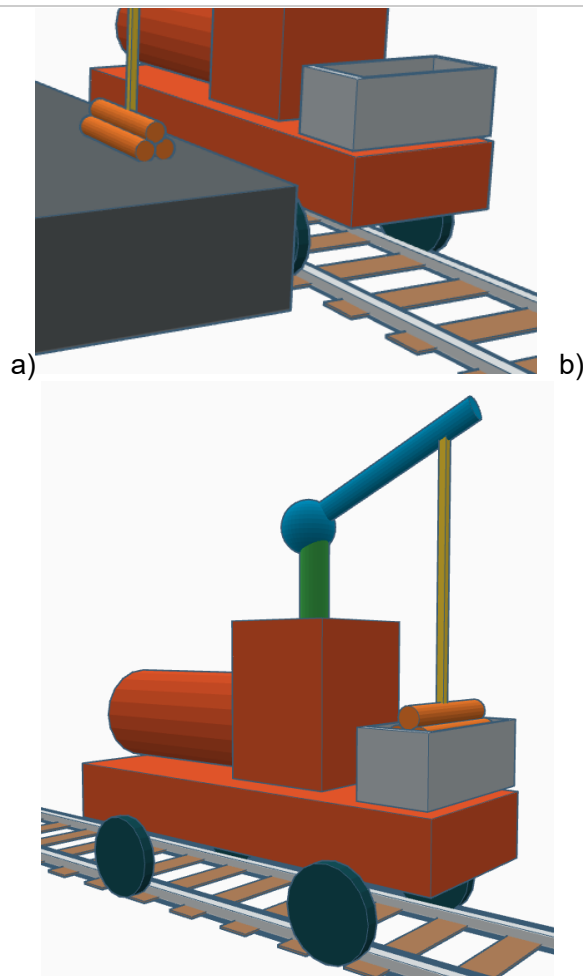


Figura 10: Guindaste e madeira;
a) braço a apanhar madeira fora da locomotiva
b) braço a deixar madeira na zona de carga

5.3. *Criação do guindaste*

O guindaste deve ser criado tendo em consideração as partes que serão movimentadas, expondo por exemplo um parâmetro para o ângulo de rotação do eixo (verde) e outro para o ângulo de rotação do braço (azul). Deve ser possível definir ambos os ângulos através de uma ou duas funções no código, para que depois possam ser controlados pelo teclado (ver ponto seguinte).

5.4. *Controlo do guindaste por teclado*

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

- A **interface**, para detetar as teclas pressionadas
- A **cena**, para invocar funções de controlo no elemento controlável em



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. Analise a classe ***MyInterface***, fornecida no código de exemplo, em particular os métodos ***initKeys***, ***processKeyDown***, ***processKeyUp*** e ***isKeyPressed***.
2. Na classe ***MyScene*** acrescente o seguinte método ***checkKeys()*** e acrescente uma chamada ao mesmo no método ***update()***.

<code>checkKeys() {</code>	
<code> var text="Keys pressed:</code>	
<code> ";</code>	
<code> var keysPressed=false;</code>	
<code></code>	
<code> // Check for key codes</code>	
<code>e.g. in https://keycode.info/</code>	
<code> if</code>	
<code>(this.gui.isKeyPressed("KeyW")) {</code>	
<code> text+=" W ";</code>	
<code> keysPressed=true;</code>	
<code> }</code>	
<code></code>	
<code> if</code>	
<code>(this.gui.isKeyPressed("KeyS"))</code>	<code>{</code>
<code> text+=" S ";</code>	
<code> keysPressed=true;</code>	
<code> }</code>	
<code> if (keysPressed)</code>	
<code> console.log(text);</code>	
<code>}</code>	

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



- O ângulo de rotação do guindaste em torno do eixo principal
 - O ângulo de inclinação do braço, que determinará a elevação do cabo e da carga
- 2.2. Use as variáveis acima na função ***MyCrane.display()*** para orientar e posicionar o eixo, braço e cabo.
- 2.3. Crie os métodos ***turn(val)*** e ***tilt(val)*** para alterar o ângulo de rotação e inclinação respetivamente (em que ***val*** podem ser valores positivos ou negativos).
- 2.4. Crie a função ***reset()*** que deverá recolocar o guindaste na posição inicial.
3. Altere o método ***checkKeys()*** da ***MyScene*** de forma a invocar os métodos ***turn()***, ***tilt()*** ou ***reset()*** do guindaste para implementar o controlo referido acima - ***A/D*** para rodar, ***W/S*** para inclinar, ***P*** para apanhar/largar carga e ***R*** para fazer "reset".

Crie duas imagens (equivalentes à figura 10) e uma tag no repositório neste ponto.



(7, 8)



(5)

6. Refinamento da modelação dos elementos em cena

Neste ponto pretende-se o melhoramento visual dos elementos em cena com materiais e texturas. Para esse efeito, espera-se também a criação de luzes adequadas para realçar os mesmos. Tenha em atenção o tipo de materiais reais usados nos diferentes elementos ao escolher os coeficientes de reflexão especular, difusa, etc. (p.ex. a locomotiva é metálica, a carga é de madeira, as paredes são pintadas, as janelas são de vidro, etc.)



Sugere-se como dimensões para o contentor as seguintes: 2.25 x 1 x 1.25 unidades (LxAxC).

Aplique materiais a simular metal e texturas ao modelo da locomotiva de forma a ter pelo menos dois tipos diferentes de materiais/texturas aplicadas. Devem ser adicionadas texturas a todos os elementos da locomotiva.

Podem adicionar mais geometria para criar mais detalhe a este modelo, sem alterar as dimensões/posições especificadas para os elementos do modelo principal.

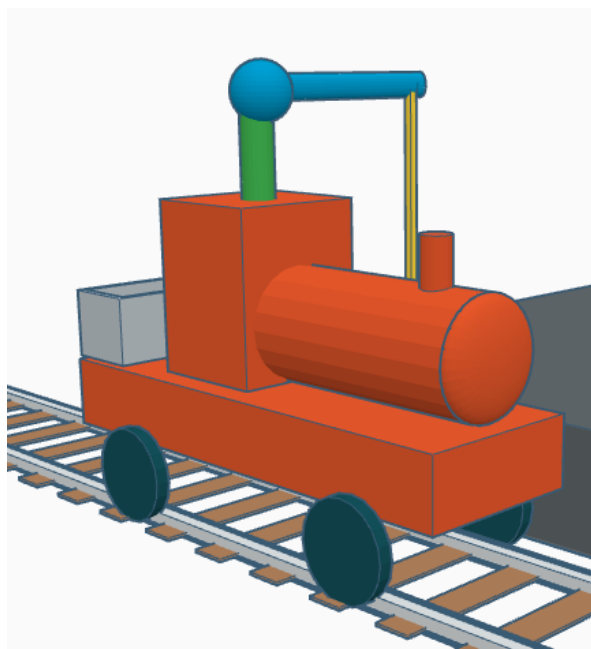


Figura 11: Exemplo do modelo da locomotiva (cores aplicadas de forma ilustrativa apenas; devem ser adicionadas texturas).

Crie uma imagem da locomotiva

atualizada.  (9)

6.6. *MyStationModel* - *Estação de comboio*

Adicione uma representação visual para as estações, definidas como pontos pelo **MyTrack**. Nesses pontos, deve ser apresentado um modelo de estação semelhante ao apresentado na figura 12. Este modelo deve também ter pelo menos dois materiais diferentes aplicados, assim como texturas. Especificamente, utilize texturas para desenhar as portas (e janelas opcionalmente) da estação.

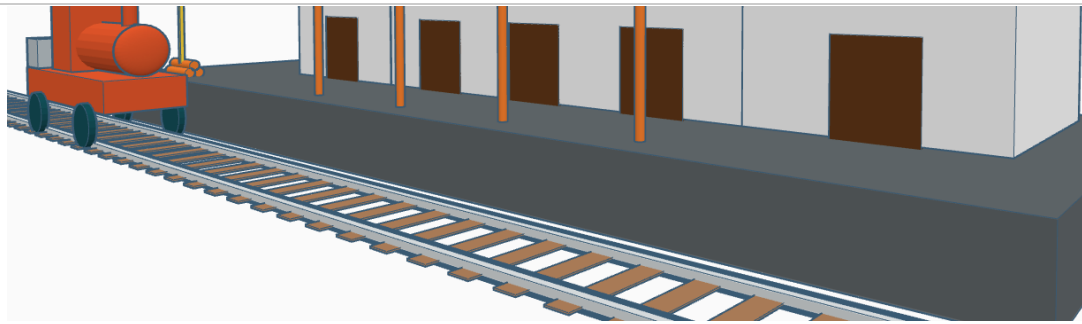


Figura 12: Exemplificação do modelo da estação de comboio.

Crie uma imagem da estação.  (10)

6.7. *Terreno - Texturas*


Aplique uma textura à escolha do grupo ao terreno fornecido no código base. Crie um material adequado para o terreno, e associe a textura escolhida ao mesmo.

Reveja a criação do objeto **MyPlane** usado para o terreno, especificamente as coordenadas de textura fornecidas. Ajuste estas coordenadas se necessário para a textura escolhida.

6.8. *Iluminação*

Crie as luzes necessárias para realçar os materiais e texturas usados nos elementos anteriores.

Crie uma imagem da cena total e uma tag

no repositório neste ponto.  (11)  (6)

7. *Funcionalidades de valorização (até 2 valores)*

Nesta secção é apresentado um conjunto de funcionalidades variadas, de dificuldade acrescida. Devem escolher um subconjunto dessas funcionalidades que totalize a cotação máxima deste ponto (2 valores). Por exemplo, poderão optar por implementar as funcionalidades 1, 2 e 3, ou as funcionalidades 3 e 4.

A cotação apresentada para cada funcionalidade representa o seu valor máximo, caso a sua implementação não tenha falhas. A implementação de mais do que 2 valores não permite compensar falhas noutros pontos. No ficheiro README.md do projeto devem estar identificadas as funcionalidades escolhidas para ser avaliadas, e brevemente



Projeto Final 2021_2022.docx

Atualização automática a cada 5 minutos

movimento da locomotiva. Deve estar dependente da velocidade da locomotiva.

2. **Fumo da chaminé da locomotiva (0.5 valores)**

Criar várias esferas que, ao saírem da chaminé, vão aumentando de tamanho e desaparecendo posteriormente.

3. **Correção de ligações da polilinha (1 valor)**

Alterar a classe que desenha os segmentos da pista de forma a que não sejam visíveis as falhas no caminho referidas na secção 1.2

4. **Suavização da animação da locomotiva (1 valor)**

Aplicar rotação à locomotiva durante a transição entre dois segmentos, de forma a que a sua orientação se altere de forma gradual. Sugere-se que a rotação da locomotiva se inicie antes de ser atingido o ponto de transição, e termine um pouco depois.

Crie uma imagem e tag por cada

funcionalidade implementada.  (12-14)



(7-9)

Notas sobre a avaliação do trabalho

A classificação máxima a atribuir a cada alínea corresponde a um desenvolvimento ótimo e eficiente 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 desenvolvimentos além dos que são pedidos, como forma de compensar outros componentes em falta.

1. Criação de carris (2.5 valores)
2. Criação da Locomotiva (3 valores)
3. Cubemap e interface (2 valores)
4. Animação da locomotiva (3 valores)
5. Guindaste controlável por teclado (3 valores)
6. Refinamento da modelação dos elementos em cena (2.5 valores)



Projeto Final 2021_2022.docx



Atualização automática a cada 5 minutos

Proposta de plano de trabalho

- Semana de 18 de abril: Apresentação do projeto; Ponto 1
- Semana de 25 de abril: Ponto 2 e início do Ponto 3
- Semana de 2 de maio (*sem aulas*): Conclusão do ponto 3
- Semana de 9 de maio: Ponto 4
- Semana de 16 de maio: Ponto 5
- Semana de 23 de maio: Ponto 6
- Semana de 30 de maio: Ponto 7 e refinamentos
- Semana de 6 de junho: Entrega e avaliações

Checklist

Durante a execução da parte A deverá identificar os commits com as tags assinaladas acima, **respeitando estritamente a regra dos nomes**:

-  **Imagens (13+):** (nomes do tipo "cgra-t<turma>g<grupo>-proj-n.png")
-  **GIT Commits/Tags (8+):** (formato "proj-1", "proj-2", ...)