Trabalho B

Cena Interactiva com Câmaras Fixas, Câmara Móvel, Instanciação de Primitivas Geométricas, Animações Simples e Colisões

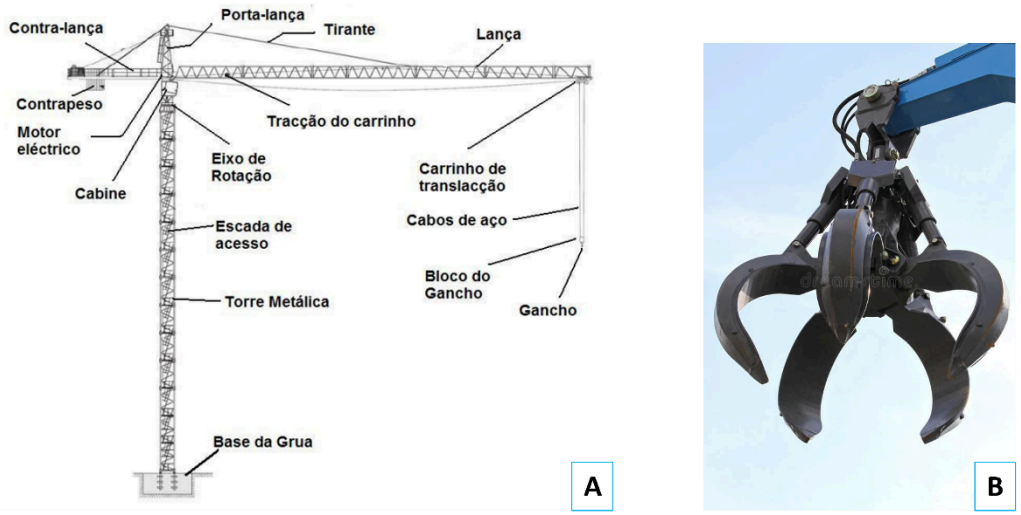

Figura 1 – Diagramas de grua com os seus elementos constituintes (A) e exemplo de garra (B).
Fonte (A): https://scholar.tecnico.ulisboa.pt/records/jIqMiTjmnJHm8RFcSolPxtpybPGYRzmWtQ9K?lang=en
Fonte sobre o funcionamento de uma grua: https://www.bigrentz.com/blog/how-cranes-work

Objetivos

Os objetivos do segundo trabalho de laboratório são: (i) compreender e implementar a arquitectura de uma aplicação gráfica interativa; (ii) explorar os conceitos básicos de modelação geométrica por instanciação de primitivas; (iii) explorar o conceito de câmara virtual; (iv) perceber as diferenças entre projecção ortogonal e projecção perspectiva; (v) aplicar técnicas básicas de animação; e (vi) compreender e implementar técnicas simples de detecção de colisões.

Todos os grupos devem submeter o código até ao dia **10 de Maio às 23:59h** (final da Semana 4). As discussões serão realizadas nos respectivos turnos na **1ª ou 2ª aula** da Semana 5, entre 13 a 17 de Maio. O Trabalho B corresponde a **4 valores** da nota da componente laboratorial. A realização deste trabalho tem um esforço estimado de **14 horas por elemento do grupo**, distribuído por **duas semanas**.

Não esquecer de comunicar ao docente do laboratório as **horas despendidas pelo grupo (média do grupo)** na realização deste trabalho.

Lista de Tarefas

1. [**0,50 valores**] Primeiro, definir o fundo da cena com uma cor clara. Seguidamente, incluir três câmaras fixas de projecção ortogonal com vistas frontal, lateral e de topo sobre a cena. Estas câmaras devem estar orientadas para o centro da cena. Definir ainda mais duas câmaras fixas colocadas por forma a captar todos os elementos da cena, devendo ser posicionadas num mesmo ponto fora dos eixos principais e orientadas para o centro. Uma das câmaras aplicará a projecção ortogonal e a outra uma projecção perspectiva. Adicionar ainda uma câmara móvel com projecção perspectiva a ser colocada no gancho da grua, alinhada ao longo do comprimento da lança e que aponta directamente sobre o plano xOz da cena (i.e., vista de topo). Deve ser possível alternar entre as seis câmaras utilizando as teclas numéricas '1' (frontal), '2' (lateral), '3' (topo), '4' (câmara fixa com projecção ortogonal), '5' (câmara fixa com projecção perspectiva) e '6' (câmara móvel com projecção perspectiva).

2. [**0,75 valores**] Modelar em Three.js uma versão simplificada de uma grua tomando como referência a Figura 1 (A). Sendo um objeto articulável, é necessário definir uma hierarquia de transformações entre as diferentes peças que compõem o objeto, isto, por forma a movimentar a grua por forma a içar cargas. As peças são as seguintes: (i) base da grua, (ii) torre metálica com porta-lanças, (iii) contra-lança e lança, (iv) contra-peso, (v) tirantes sobre a contra-lança e sobre a lança, (vi) cabine, (vii) carrinho de translação, (viii) cabo de aço conectado a bloco do gancho, e (xix) uma garra articulável de 4 dedos aguçados (em vez do gancho) (Figura 1 (B)). É necessário modelar as peças recorrendo a pelo menos uma das seguintes primitivas geométricas disponíveis na biblioteca Three.js[1,2]: cilindros, cubos e tetraedros.
   Nota: O dimensionamento dos objetos é livre.
   Nota: O resultado deve consistir, pelo menos, numa aproximação 'low-poly' da grua e da garra.

3. [**0,50 valores**] Modelar também em Three.js uma versão simplificada de um contentor aberto deitado sobre o plano xOz e de várias cargas espalhadas à volta da grua. Para tal, basta modelar a base e as 4 paredes do contentor. Os objetos a carregar pela grua devem ser espalhados aleatoriamente pela cena mas sem nunca intersectarem entre si, nem com a própria grua ou o contentor. As cargas deverão ter diferentes dimensões, mas nunca maiores do que a largura e comprimento do contentor. É necessário modelar cada uma das peças recorrendo às primitivas geométricas de cubos, dodecaedros, icosaedros, torus e *torus knot* do Three.js.
   Nota: O dimensionamento do contentor é livre mas em conformidade com o da grua.
   Nota: O resultado deve consistir, pelo menos, numa aproximação 'low-poly' do contentor.

4. **[0,75 valores]** Permitir ao utilizador operar a grua por forma a içar e colocar cargas dentro do contentor recorrendo às seguintes teclas:
   a. 'Q(q)' e 'A(a)' para controlar o ângulo θ_1 que roda o eixo de rotação da secção superior (incluindo a cabine);
   b. 'W(w)' e 'S(s)' para controlar o deslocamento δ_1 que translada o carrinho de translação;
   c. 'E(e)' e 'D(d)' para controlar o deslocamento δ_2 que translada a secção composta pelo bloco do gancho e a garra, ora descendo-o ora subindo-o;

[1] Vários exemplos podem ser encontrados na documentação oficial https://threejs.org/manual/#en/primitives
[2] Ilustração com várias primitivas Three.js https://threejs.org/manual/examples/primitives.html

d. e 'R(r)' e 'F(f)' para controlar o ângulo θ_2 de abertura e fecho da garra.

Nota: Os movimentos do objeto articulado deve apresentar um movimento a velocidade constante, sendo a direcção do movimento dada por um vector tridimensional. O cálculo do movimento deve ter em consideração que o utilizador pode carregar em várias teclas em simultâneo.

Nota: Cada grau de liberdade deve apresentar uma amplitude de movimentos limitada a um intervalo de valores pré-definido. Isto para que não ocorram poses cinematicamente impossíveis (e.g., o volume do carrinho de translação sair fora da lança).

5. **[0,25 valores]** A representação visual dos objetos da cena deve alternar entre modelo de arames e sólida usando a tecla '1'.

6. **[0,75 valores]** Implementar a detecção de colisões entre a garra e a carga. Recorrer a pares de contacto esfera-esfera para detectar a colisão, isto é, tanto a garra como as cargas devem ter acopladas geometrias de colisão na forma de esferas alinhadas com o referencial local. Aquando da colisão, activar uma animação que movimenta a grua e desloca a carga, desde o seu local inicial até ao interior do contentor. Enquanto a animação se processa, nenhuma tecla deve ser processada.

7. **[0,50 valores]** Incluir um Heads-Up Display (HUD) com o mapa das teclas. O HUD deve assinalar qual(ais) a(s) tecla(s) activa(s).

Notas Importantes:

1. A biblioteca Three.js já contém as classes principais que necessitam para desenvolver os projectos desta cadeira. É por isso aconselhável que os alunos devam adoptar uma programação orientada a objetos recorrendo às classes desta biblioteca, devendo sempre seguir boas práticas de programação que permitam a reutilização do código em entregas posteriores e facilitem a escalabilidade.

2. Não devem utilizar bibliotecas externas nem funções do Three.js para detectar colisões ou implementar a física inerente ao movimento. Esperamos ver o vosso código e não chamadas a funções de bibliotecas.

3. Para além dos acontecimentos de *update* e *display* existem mais um conjunto de acontecimentos, tais como teclas pressionadas ou soltas, temporizadores e redimensionamento da janela. Sugerimos vivamente que tais acontecimentos sejam tratados pelas respectivas funções de *callback* de forma independente. Tenham em atenção que neste trabalho **não** é requerida a implementação devida dos acontecimentos de redimensionamento da janela, mas tal vai ser pedido no Trabalho C.

4. A implementação de todos os trabalhos desenvolvidos nos laboratórios de Computação Gráfica deve usar o ciclo de animação (update/display cycle). Este padrão de desenho, usado nas aplicações de computação gráfica interactiva, separa o desenho da cena no ecrã da actualização do estado do jogo em duas fases distintas. Na fase de display são cumpridos três passos base: limpar o buffer; desenhar a cena e forçar o processamento dos comandos. Na fase de update todos os objetos do jogo são actualizados de acordo com a física inerente. É

ainda nesta fase que se processa a detecção de colisões e implementação dos respectivos comportamentos.

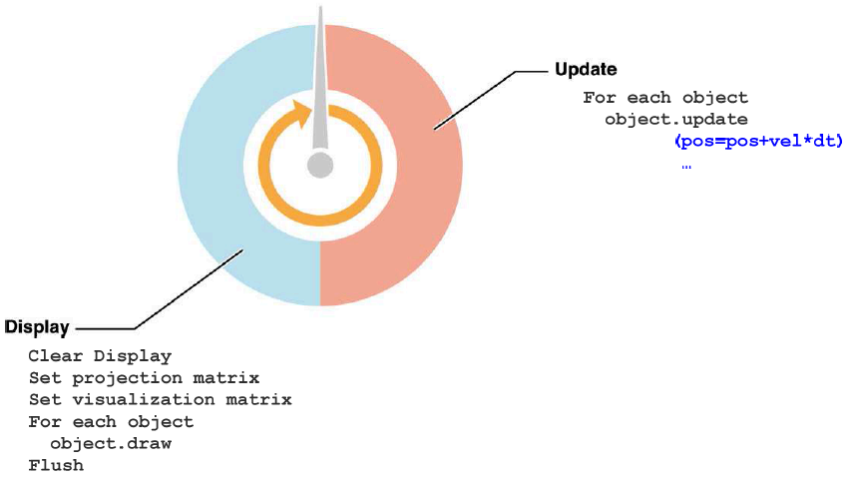

Figura 2 – Ciclo de animação com as fases de *update* e *display*.