

Trabalho B

Cena Interativa com Câmaras Fixas, Instanciação de Primitivas Geométricas, Animações Simples e Colisões



Figura 1 – Exemplo icónico de robô transformável (A) em ambos os modos de transformação (robô e caminhão), (B) com reboque e (C) a sequência de transformação “caminhão → robô”.

Objetivos

Os objetivos do segundo trabalho de laboratório são: (i) compreender e implementar a arquitetura 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 projeção ortogonal e projeção perspetiva; (v) aplicar técnicas básicas de animação; e (vi) compreender e implementar técnicas simples de deteção de colisões.

Todos os grupos devem submeter o código até ao dia **23 de maio às 23h59** (final da Semana 4). As discussões serão realizadas nos respetivos turnos na **2ª aula** da Semana 5. 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 câmaras fixas de projeção ortogonal com vistas frontal, lateral e de topo sobre a cena. Estas câmaras devem estar orientadas para o centro da cena. Definir ainda mais uma câmara fixa que permita visualizar toda a cena via uma projeção perspetiva. Esta câmara deve ser posicionada sobre a cena e orientada para o centro. Deve ser possível alternar entre as quatro câmaras utilizando as teclas numéricas '1' (frontal), '2' (lateral), '3' (topo) e '4' (perspetiva).
2. **[0,75 valores]** Modelar em Three.js uma versão simplificada do robô transformável apresentado na 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 converter o robô para camião e vice-versa. As peças são as seguintes: cabeça que inclui detalhes para olhos e antenas, braços com dois tubos de escape, antebraços, tronco, abdómen, cintura, coxas, pernas, pés e seis rodas. É necessário modelar as peças recorrendo a pelo menos uma das seguintes primitivas geométricas disponíveis na biblioteca Three.js^{1,2}: cilindros, cubos e cones.
Nota: O dimensionamento dos objetos é livre.
Nota: O resultado deve consistir, pelo menos, numa aproximação 'low-poly' do robô.
3. **[0,50 valores]** Modelar também em Three.js uma versão simplificada do reboque apresentado na Figura 1 (B). Para tal, basta modelar o volume do contentor, as quatro rodas e a peça de ligação. É necessário modelar cada uma das peças recorrendo às primitivas geométricas de cilindros e cubos do Three.js.
Nota: O dimensionamento do reboque é livre, mas conforme o do camião.
Nota: O resultado deve consistir, pelo menos, numa aproximação 'low-poly' do reboque.
4. **[0.75 valores]** Permitir ao utilizador realizar a conversão entre robô-camião, isto é, permitir movimentar os graus de liberdade representados na Figura 1 (C) (vide setas amarelas³) recorrendo às seguintes teclas:
 - a. 'Q(q)' e 'A(a)' para controlar o ângulo θ_1 que roda o eixo de revolução dos pés;
 - b. 'W(w)' e 'S(s)' para controlar o ângulo θ_2 que roda o eixo de revolução da cintura;
 - c. 'E(e)' e 'D(d)' para controlar o deslocamento δ_1 que translaciona os membros superiores medial e lateralmente;
 - d. e 'R(r)' e 'F(f)' para controlar o ângulo θ_3 que roda o eixo de revolução da cabeça.**Nota:** Por forma a simplificar o problema, considerem que os membros superiores não rodam na articulação do ombro, mas que deslizam ora afastando-se ou aproximando-se do tronco.
Nota: O objeto articulado deve apresentar um movimento a velocidade constante, sendo a direção do movimento dada por um vetor tridimensional. O cálculo do movimento deve ter em consideração que o utilizador pode carregar em várias teclas em simultâneo.

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>

³ Adotem a alteração descrita na primeira "Nota: ..." da Tarefa 4.

Nota: Cada grau de liberdade deve apresentar uma amplitude de movimentos limitada a um intervalo de valores pré-definido. Isto para não ocorrerem poses cinematicamente impossíveis (e.g., o volume das pernas não deve intersectar o volume da cabeça).

Nota: Quando em modo camião, todo o objeto deve assentar sobre um plano invisível, isto é, as rodas devem ser coplanares.

5. **[0,50 valores]** Permitir ao utilizador deslocar o reboque pelo plano invisível com o teclado utilizando as teclas das setas para o reposicionar segundo os eixos globais dos X e Y, respetivamente. O referencial do reboque deve estar alinhado com o referencial do robô-camião.

Nota: Os movimentos do reboque devem apresentar uma velocidade constante, sendo a direção do movimento dada por um vetor tridimensional. O cálculo do movimento deve ter em consideração que o utilizador pode carregar em várias teclas em simultâneo.

Nota: O reboque deve ser inicialmente posicionado no lado posterior do robô-camião sem o intersectar.

6. **[0,25 valores]** A representação visual dos objetos da cena deve alternar entre modelo de arames e sólida usando a tecla '7'.
7. **[0,75 valores]** Implementar a deteção de colisões entre o camião e o reboque. Recorrer a pares de contacto AABBs para detetar a colisão, isto é, tanto o camião como o reboque devem ter acopladas geometrias de colisão na forma de paralelepípedos alinhados com o referencial global da cena. Aquando da colisão, ativar uma animação que desloca o reboque para o ponto de ligação com o camião.

Notas Importantes:

1. 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 interativa, separa o desenho da cena no ecrã da atualização 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 atualizados conforme a física inerente. É ainda nesta fase que se processa a deteção de colisões e implementação dos respetivos comportamentos.

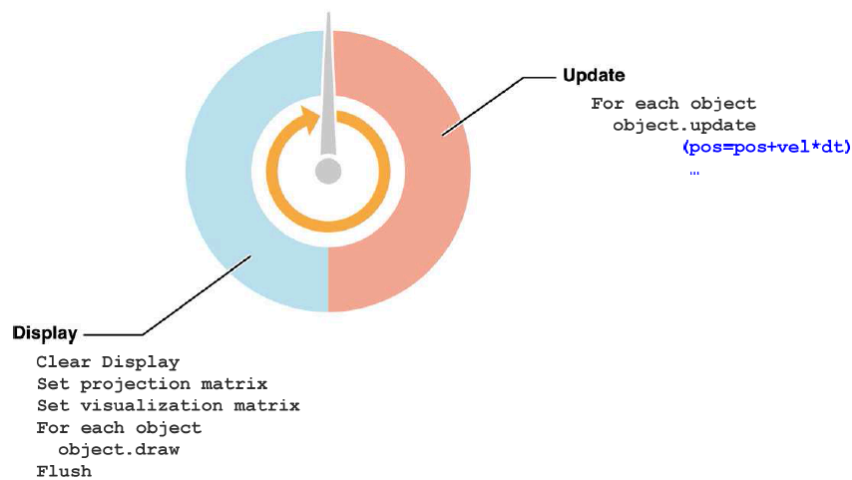


Figura 2 – Ciclo de animação com as fases de *update* e *display*.

2. Não devem utilizar bibliotecas externas nem funções do Three.js para detetar colisões ou implementar a física inerente ao movimento. Esperamos ver o vosso código e não chamadas a funções de bibliotecas.
3. Para além de 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 respetivas funções de *callback* de forma independente. Tenha 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 biblioteca Three.js já contém as classes principais que necessitam para desenvolver os projetos desta cadeira. É por isso que os alunos devem adotar 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.