Segundo Trabalho – Cena Simples e Interactiva com Câmara Móvel e Colisões

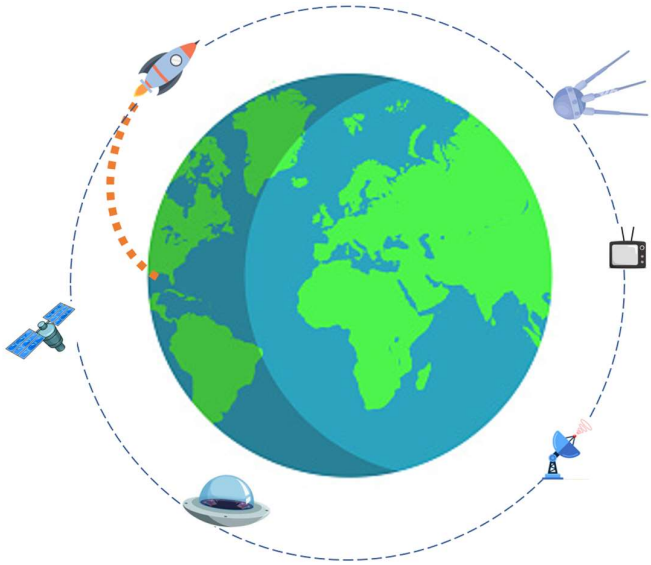

Figura 1 – Ilustração do planeta e os objectos contidos na exosfera que incluem o foguetão e exemplos de lixo espacial.

Objectivos

Os objectivos do segundo trabalho de laboratório são: (i) explorar o conceito de câmara virtual, perceber as diferenças entre (ii) câmara fixa e câmara móvel, e entre (iii) projecção ortogonal e projecção perspectiva; deseja-se ainda (iv) a compreensão das técnicas básicas de animação bem como a detecção de colisões.

Todos os grupos devem submeter o código até ao dia **03 de Junho às 23:59h**. As discussões serão realizadas nos respectivos turnos na semana de 06 a 10 de Junho. O Segundo Trabalho corresponde a **3 valores** da nota do laboratório. 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. Com "papel e caneta"[1] esboçar uma cena similar à que é ilustrada na (Figura 1). O esboço deve apresentar (i) uma figura geral de toda a cena ilustrando a composição pretendida com os vários objectos, bem como (ii) outras pequenas figuras onde são definidas as dimensões que se querem atribuir aos objectos da composição. Podem encontrar mais detalhes e uma melhor descrição dos requisitos de modelação 3D na legenda da Figura 1 e no texto da Tarefa 2, pelo que as devem seguir à risca. [**0,25 valores**]

2. Modelar em Three.js um planeta, um foguetão e lixo espacial como ilustra a Figura 1. Devem ter-se em conta as seguintes características na modelação 3D da cena (nota: o dimensionamento dos objectos é livre mas deverá respeitar sempre as relações abaixo indicadas) [**0,75 valores**]:

    a) o planeta consiste numa esfera centrada na origem da cena com raio R (é o maior objecto dentro da cena);
    b) a nave espacial consiste em, pelo menos, um cilindro para o corpo principal, um outro cilindro para o nariz e 4 cápsulas para os propulsores auxiliares; devem recorrer apenas às primitivas geométricas CylinderGeometry e CapsuleGeometry; note-se que os tampos do cilindro do corpo principal e do cilindro do nariz devem estar alinhados entre si, vértice a vértice; o foguetão deve ser colocado na exosfera do planeta a uma distância 1,20xR desde o centro do planeta; a altura total do foguetão (H) deve ser $R/12 < H < R/10$;
    c) o lixo espacial consiste em primitivas geométricas como cubos, cones ou poliedros regulares; devem ser colocados pelo menos 20 objectos aleatoriamente distribuídos pela exosfera do planeta também a uma distância 1,20xR desde o centro do planeta; a envergadura máxima de qualquer lixo espacial (C) deve ser $R/24 < C < R/20$.

3. Permitir ao utilizador movimentar o foguetão pela exosfera do planeta recorrendo às teclas das setas (longitude/azimute com as teclas esquerda e direita; latitude/zénite com as teclas cima e baixo). O foguetão deve ser inicialmente posicionado aleatoriamente na exosfera do planeta; todo o (re)posicionamento do foguetão deve ser expresso em coordenadas esféricas. O movimento do foguetão deve apresentar um movimento a velocidade angular constante, sempre constrangido à superfície esférica da exosfera. O cálculo do movimento deve ter em consideração que o utilizador pode carregar em várias teclas em simultâneo. [**1,0 valores**]

4. Implementar a detecção de colisões entre o foguetão e o lixo espacial. Recorrer a pares de contacto esfera-esfera para detectar a colisão, isto é, tanto o foguetão como qualquer lixo espacial devem ter acopladas uma geometria de colisão em forma de esfera envolvente. Aquando da colisão, o lixo é simplesmente removido. Em cada instante, os cálculos de detecção de colisões não devem ser realizados entre todos os pares de contacto. Para tal, usar como técnica de aceleração a avaliação dos pares de contacto com base na localização dos objectos no quadrante (i.e., semi-hemisfério) do planeta. [**1,0 valores**]

5. Definir uma câmara fixa com uma vista frontal sobre a cena utilizando uma projecção ortogonal que mostre toda a cena (câmara 1). Definir ainda duas câmaras adicionais: a câmara 2 deve ser

[1] Por "papel" entenda-se optar por um material celulósico (papel analógico) ou por um dispositivo multi-toque como um tablet, smartphone, laptop 2-em-1 (papel digital). Devem ser apresentados desenhos à mão livre pelo que não devem recorrer a templates nem a desenho vectorial de formas idealizadas.

fixa e permitir visualizar toda a cena mas através de uma projeção perspetiva; e a câmara 3 deve igualmente utilizar uma projeção perspectiva mas é móvel pois fica colocada atrás (e um pouco acima) do foguetão acompanhando o seu movimento. Esta câmara deve estar sempre apontada na direcção e sentido do movimento do foguetão. Deve ser possível alternar entre as três câmaras 1, 2 e 3 utilizando as teclas numéricas '1', '2' e '3', respectivamente. [**1,0 valores**]

Notas Importantes:

1. Antes de escrever qualquer linha de código, é necessário esboçar o que se pretende modelar em 3D pois tal actividade ajuda muito a perceber que primitivas e transformações devem ser aplicadas. Não menos importante é o desenho do grafo de cena, enquanto representação abstracta dos objectos, pois consiste num diagrama fundamental para a correcta modelação não só dos objectos mas como de toda a cena.

2. 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 objectos 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.

3. 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.

Sugestões

1. Antes de escrever qualquer linha de código, é necessário esboçar o que se pretende modelar em 3D pois tal actividade ajuda muito a perceber que primitivas e transformações devem ser aplicadas. Não menos importante é o desenho do grafo de cena, enquanto representação abstracta dos objectos, pois consiste num diagrama fundamental para a correcta modelação não só dos objectos mas como de toda a cena.

2. 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 objectos 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.

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 respectivas funções de *callback* de forma independente. **Tenha em atenção que neste**

e no último trabalho é requerida a implementação devida dos acontecimentos de redimensionamento da janela!

4. A posição e direcção inicial do foguetão e do lixo podem ser obtidas recorrendo à função, nativa do JavaScript, Math.random().[2]

5. Por fim, os alunos devem adoptar uma programação orientada a objectos, seguindo sempre boas práticas de programação que permitam a reutilização do código em entregas posteriores e facilitem a escalabilidade.

[2] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random