Autor: Anthony dos Santos Pereira, nº 93016

1º Projeto de Introdução à Computação Gráfica

It's Dance of the Druids, Darling!

***ABSTRACT* - This project consisted of the modeling and animation of a vast set of different meshes and geometries, aiming to recreate a scene from the *Outlander (2014)* series, such as BoxGeometry, TorusGeometry, ExtrudeGeometry, CylinderGeometry, PlaneGeometry, among others. Some types of light were also used like AmbientLight, DirectionalLight, PointLight, and HemisphericLight. The use of shadows was also applied. The user can interact with the scene using the keyboard.**

I. INTRODUÇÃO

Este projeto foi desenvolvido aquando da unidade curricular de Introdução à Computação Gráfica da Licenciatura em Engenharia Informática da Universidade de Aveiro. Sendo de tema livre, o objetivo pretendido para este trabalho era demonstrar técnicas de modelação, animação e interação 3D, conjugando com o uso de luzes e sombras, aprendidas ao longo do semestre, recorrendo à biblioteca JavaScript Three.js.

II. CONTEXTO

Uma das cenas mais emblemáticas do primeiro episódio da série de drama-histórico *Outlander (2014)* trata-se da realização de um ritual antigo por um grupo de mulheres à volta de um círculo de pedras de nome *Craigh na Dun*: https://www.youtube.com/watch?v=dz1EW3DKKgo&t=167s

Fig.1 - Cena da "Dança dos Druídas" da série *Outlander (2014)*

Ora, este cenário é o ponto de partida para este projeto, sendo que a sua grande parte tenta recriá-lo. Para o desenho das druídas, no entanto, foi usado como modelo uma personagem do anime *Darling in the Franxx (2018)*, Zero Two, conhecida por usar a palavra "Darling" recorrentemente.

Fig.2 - Representação da Zero Two, personagem do anime *Darling in the Franxx (2018)*

III. CONCEITO GERAL

Como já foi referido anteriormente, dois dos elementos principais do trabalho são o círculo de pedras e as mulheres em formato Zero Two, por isso, seria inevitável elas não terem um grande destaque - estas surgem no centro da cena.

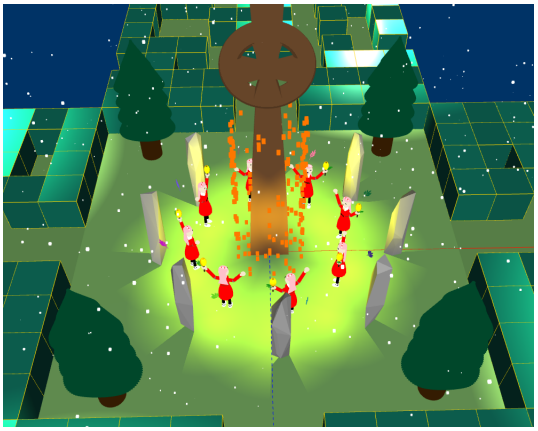


Fig.3 - Centro da cena

Também no centro surgem quatro árvores em forma de pinheiro, além de uma cruz escocesa em chamas bem no meio do círculo de pedras, baseada igualmente na série *Outlander (2014)* - desta vez, no primeiro episódio da quinta temporada.

Fig.4 - "Cruz em chamas" de uma cena do episódio 1 da 5ª temporada da série *Outlander (2014)*

Mas para além disto, à volta do centro da cena existem quatro labirintos - em que o próprio utilizador é convidado a vaguear pelo interior de um deles.

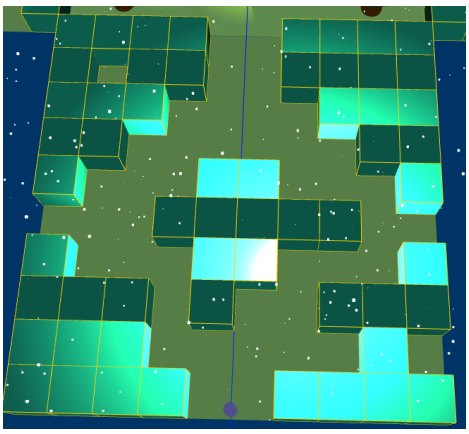


Fig.5 - Exemplo de labirinto presente na cena

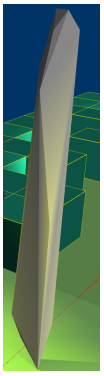
IV. CONTROLOS E INTERAÇÃO

Tendo em conta a existência de interação para com o utilizador ser um dos requisitos para este trabalho, foram usados estes controlos recorrendo ao teclado:

Tab: Mudar posição câmara para início do labirinto
Esc: Mudar para posição default da câmara
Setas: Mover-se no labirinto
Shift: Colocar cada Zero Two à volta de uma pedra
Enter: Colocar todas as Zero Twos em torno do centro (default)
BackSpace: Colocar todas as Zero Twos em torno de uma pedra (aleatória)
Space: Todas as Zero Twos saltam
Alt: Zero Twos giram mais rapidamente e no mesmo sentido que o default (direita)
AltGr: Zero Twos rodam para o sentido oposto (esquerda)
Esc: Mudar para posição default da câmara
+, -: Adicionar/remover Zero Twos (máx. 8)
P: Zero Twos giram de forma ainda mais rápida
X: Mostrar flocos de neve (default)
S: Retirar flocos de neve
W: Ligar luz branca
R: Ligar luz vermelha

Tab.1 - Controlos criados

Existem igualmente outros tipos de interação sem recorrer ao teclado, nomeadamente, através do cursor; este é o caso das pedras, que aumentam de tamanho, e das árvores, que mudam de cor.



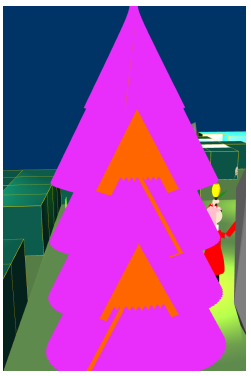


Fig.6 & 7 - Exemplos de interação com uma pedra e árvore, recorrendo ao rato

V. IMPLEMENTAÇÃO

- Círculo de pedras

Para a modelação de cada pedra, foram usadas três funções - duas auxiliares, uma que gera múltiplos pontos aleatórios (com posições não superiores a 25 no eixo das ordenadas e não superiores a 4 nos eixos das abcissas e cotas) e outra que verifica se pelo menos três destes pontos gerados se encontram incluídos no plano y=0 (porque três é o número mínimo de pontos necessário para formar um plano, e para garantir que cada pedra está pousada no chão da cena); a terceira usa estes pontos e cria uma **ConvexGeometry**, que os une. Tendo em conta que estes pontos são criadas de modo completamente aleatório, cada pedra terá um formato diferente, efeito esse pretendido. A cada pedra é igualmente atribuído um **MeshPhongMaterial**, de modo à luz proveniente de cada uma das tochas ser refletida nestas.

Fig.8 - Pedra

- Zero Two

Para cada objeto Zero Two, existem duas componentes principais: a cabeça, e o resto do corpo.

- **Cabeça**: foi usado um **TorusGeometry** com radius igual a 1 para formar o esqueleto da cabeça, tentando imitar o melhor possível uma cabeça humana, que não é uma esfera perfeita; a seguir, para a componente do cabelo, foram criadas duas funções, uma que trata do cabelo que se encontra no topo da cabeça, que deforma (recorrendo a um scale de Y com um valor inferior a 1) a mesh resultante de um **TorusGeometry**, e outra que trata do cabelo na parte das costas, elaborando tanto uma secção retangular graças a uma **BoxGeometry** como uma triangular com um **ExtrudeGeometry** (pontas do cabelo na parte das costas); para a parte dos olhos, foi usada uma função que cria uma forma também através de um **ExtrudeGeometry**, e o mesmo aconteceu para a boca, que contém uma cor aleatória; finalmente, para a faixa para o cabelo com os cornos a vermelho, usou-se uma **BoxGeometry** e ConeGeometry com 3 radialSegments, número mínimo para formar uma mesh a 3 dimensões, conferindo um aspeto de "pirâmide" aos mesmos.

Fig. 9 - Cabeça da Zero Two

- **Corpo**: em primeiro lugar, tratou-de de modelar o tronco da personagem e, para isso, recorreu-se a uma **BoxGeometry** para o torso e colarinho, uma **LatheGeometry** para o vestido e um **CylinderGeometry** para o pescoço; de seguida, e para os membros superiores, usou-se **CylinderGeometry** para ambos os braços, antebraços e pulsos, **SphereGeometry** para os cotovelos e **TorusGeometry** para as mãos. Criando a ilusão de estar a ser agarrada pela mão esquerda (como na cena da série), existe uma tocha que consiste na junção de diferentes meshes provenientes de múltiplas geometrias, como **LatheGeometry** para parte superior da tocha, de onde sai a luz, um **CylinderGeometry** para a sua base e vários **ExtrudeGeometry** conjugados para formar uma folha, que se encontra na base da tocha. Também foi criada uma **PointLight** dentro da tocha e a esta foi-lhe designado um **MeshPhongMaterial**, para dar a ideia de brilho devido à presente luz; depois, relativamente aos membros inferiores, foram criados **CylinderGeometry** para as pernas, **LatheGeometry** para a parte superior das botas e **ExtrudeGeometry** para as botas; a sobreposição da parte final da perna/calças com a parte superior das botas cria um padrão de sucessão de várias formas triangulares, devido ao radiusBottom de cada uma da perna, que é igual a 2, ser um pouco superior ao diâmetro do lathe. Ainda existem alguns elementos decorativos, como o cinto com setas na parte da frente, recorrendo a **LatheGeometry** e **ExtrudeGeometry**, respetivamente, uma gravata, recorrendo também a **ExtrudeGeometry**, e uma folha idêntica à reproduzida para a tocha, à qual foi dada uma cor

aleatória e feita uma translação de -10 no eixo YY e -100 no eixo ZZ (no referencial do objeto e não no da cena) que confere a sensação de que paira no ar.

Fig. 10 - Zero Two

- Árvore

Cada árvore é constituída por um tronco, que é formado através de um **CylinderGeometry**, e de uma sequência de formas semelhantes a um triângulo, mas com a aresta da base sendo uma curva, criadas através de um **ExtrudeGeometry**:

Fig.11 - Peça base para a elaboração de cada árvore

Assim sendo, cada árvore é formada por cinco andares, sendo que em quatro destes cada uma de 100 peças é colocada numa posição definida por uma fórmula do tipo *Math.cos(1 + i*0.1), 1, Math.sin(1 + i*0.1)*, sendo *i* o número da peça desse andar, $i \in [0, 99]$; sabendo que qualquer ponto pertencente a uma circunferência de raio 1 tem como coordenadas x,y correspondentes a $cos(\theta), sin(\theta)$, respetivamente, acabou por ser utilizada uma variação desta fórmula para a rotação de cada uma destas peças, fórmula essa que foi obtida por tentativas. A seguir, e à medida que se sobe em andares, esta fórmula sofre pequenas alterações - o raio vai diminuindo sucessivamente, passando, por exemplo, para $1/1.3 \cong 0.77$ no segundo andar (a contar de baixo).

No final, no quinto andar (topo da árvore) são utilizadas 400 peças e cada uma delas sofre uma rotação em torno do seu eixo relativamente a um dado ângulo, diferente, como se se estivesse a recriar um cone peça-a-peça.

Fig.12 - Árvore

- Cruz celta

A cruz celta/escocesa baseada na presente na série *Outlander (2014)* foi obtida através de um grupo de um **TorusGeometry** e sete figuras obtidas através de **ExtrudeGeometry**, tendo sido elaboradas duas variantes: a primeira, que consiste em cada um 4 dos lados da cruz no centro do anel e ainda outra deste mesmo tipo, que foi deformada e colocada no centro da cruz para tapar o orifício que lá se encontrava, e a outra utilizada para a base e topo da cruz, abaixo e acima do anel central. Como material, foi utilizado um **MeshToonMaterial**, o que permitiu igualmente que a luz emitida por cada tocha fosse refletida na sua superfície.

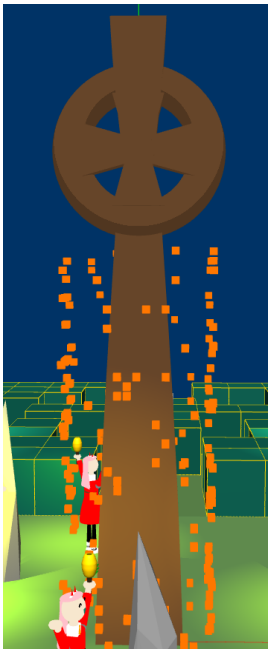


Fig. 13 - Cruz celta/escocesa

- Labirinto

Cada labirinto é gerado recorrendo a um **PlaneGeometry** para o seu chão e a múltiplas **BoxGeometry** (e **EdgesGeometry**) para as suas paredes (i.e., para elaborar o seu percurso). Tomando como referência o eixo ZZ na vertical, foram acrescentados cubos ao labirinto linha-a-linha, sendo que para a primeira e décima linhas foram completamente preenchidas, com exceção do meio destas (ficando espaço livre correspondente a duas caixas). Para as restantes linhas, foi escolhido um número de cubos aleatório (que será o número de cubos existente nessa mesma linha) e estes foram colocados aleatoriamente, variando a coordenada X dos mesmos. Contudo, a geração de cada labirinto não é completamente aleatória - para garantir que o labirinto é válido, i.e., que é possível percorrê-lo do início ao fim, são previamente liberadas algumas caixas para que exista sempre um número mínimo de lugares livres e, sendo assim, o labirinto torna-se nevagável. Por isso, pode-se considerar que a geração destes labirintos é pseudo-aleatória. Relativamente aos materiais utilizados, tentou-se intercalar entre **MeshBasicMaterial** e **MeshPhongMaterial** para cada uma das linhas, de modo a que algumas sejam iluminadas por uma **PointLight**, localizada aproximadamente no seu centro.

Outro componente presente em cada labirinto é uma mesh resultante de um **CylinderGeometry**, objeto esse que, ao pressionar-se nas teclas das setas, se irá movimentar no seu interior. Cada cilindro encontra-se no início do labirinto correspondente e tem uma cor aleatória.

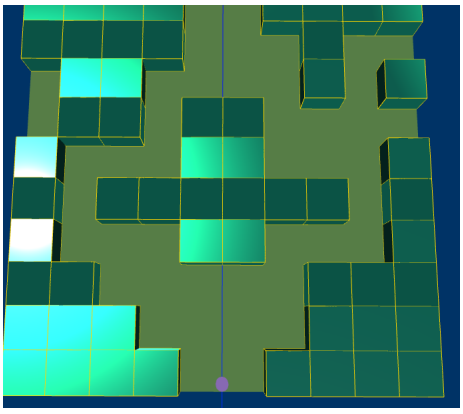


Fig. 14 - Labirinto

VI. LUZES

Já foi anteriormente descrito que foram criadas **PointLight**s que iluminam cada labirinto e as tochas, mas estas não são as únicas fontes de luz que constituem a cena. Nela, também existe uma **AmbientLight** de luz branca, uma **DirectionalLight** que dá mais cor às pedras que estão na primeira fila ao serem observadas pelo utilizador, e ainda duas **HemisphereLight**s, com tons branco e vermelho, que poderão ser utilizadas ao clicar-se nas teclas W e R, respetivamente.

VII. SOM

No HTML foi colocado um áudio embutido com autoplay que reproduz a música *Dance of the Druids*, tema da cena do primeiro episódio de *Outlander (2014)*. Se estiver a usar o Google Chrome no Ubuntu, deve fechar todas as janelas do navegador e no terminal executar **google-chrome --autoplay-policy=no-user-gesture-required**; se usar o Mozilla Firefox, basta permitir a execução do áudio quando aparecer o pop-up para o efeito.

VIII. ANIMAÇÃO

- Neve

Para recriar o efeito da neve a cair sobre a cena, foram criados múltiplos cubos recorrendo a uma **BoxGeometry** de tamanho aleatório, mas nunca superior a 0.5 (metade de 1, valor máximo de Math.random()) e a cada um destes foi-lhe atribuída uma posição inicial igualmente aleatória. Na função animateRain() (“rain” no sentido lato, neste caso trata-se de neve), a posição Y de cada um destes flocos de neve vai diminuindo progressivamente, até chegar a um mínimo de um valor negativo que corresponde a uma posição inferior à onde se encontra a cena; alcançado esse valor, cada floco é reposicionado a uma altura elevada, fazendo com que este ciclo se repita.

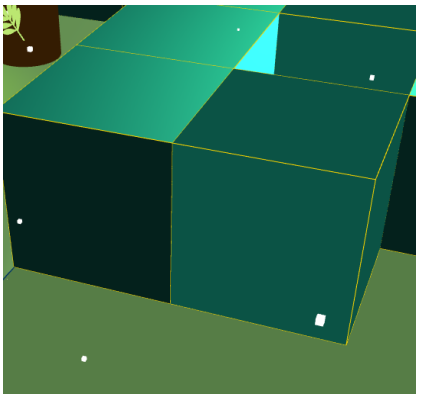


Fig.15 - Flocos de neve

- Fogo

Idêntico ao efeito da neve, foi recriado um efeito de fogo também gerando vários cubos de cor laranja, mas desta vez sendo todos do mesmo tamanho (0.8 de lado); destes, a metade foi atribuída a propriedade visible=false que, como o próprio nome sugere, permite ocultar um elemento da cena. Depois, na função animate(), são obtidos todos estes elementos e, para os que se encontram com visible=false, esta propriedade passa a ser igual a visible=true, e o inverso também acontece. Assim, cria-se

um efeito que tenta recriar uma chama que arde a cruz celta.

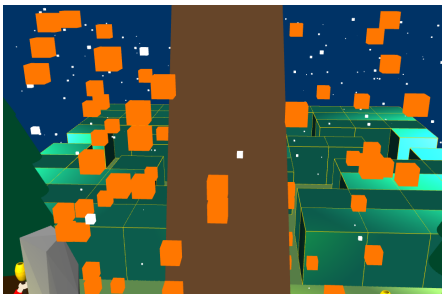


Fig.16 - Fogo

- Raycaster

Recorreu-se ao uso do **Raycaster** para obter as coordenadas do cursor e obter algum tipo de interação aquando da interseção deste para com algum elemento da cena. Um dos casos é colocá-lo sob uma das pedras, em que primeiro se verifica se o cursor intercetou algumas das pedras existentes na cena (o array intersects terá obrigatoriamente de ter tamanho superior a zero), para depois verificar-se se este se encontra dentro do array arrayStone, array este que contém todas as pedras descendentes de um dado object3D (que, neste caso, é apenas uma pedra). Sendo assim, e se o valor de scale Y da pedra for baixo, este é aumentado sucessivamente até chegar a um valor máximo de 2.0; a razão pela qual são feitos vários pequenos incrementos é para tornar o aumento de tamanho da pedra mais natural e menos brusco, o que seria possível dando apenas um único *assignment*. Retirando o cursor de cima da pedra, esta volta ao seu tamanho original.

Também foi utilizado o **Raycaster** para a interação com as folhas das árvores (parte verde) - bastante idêntico ao caso das pedras, a diferença é que agora o array que contém os elementos descendentes do object3D tem tamanho superior a 1 (o que é natural, pois contém todas as peças geradas da árvore, dos cinco andares recriados). Quando o cursor interseta uma das peças, esta ganha uma cor e todas as outras do mesmo andar ganham uma cor diferente da inicial e diferente da atribuída à peça que está *hovered*.

- PointLights

Todas as quatro **PointLight**s que se encontram nos labirintos estão incluídas no mesmo object3D; na função animate(), é feita a rotação sucessiva do mesmo, criando o efeito de rotação de cada uma das luzes à volta do centro da cena.

- Labirinto

Ao clicar-se no Tab, o utilizador é redirecionado para o início de um dos labirintos, tomando o lugar do cilindro que lá se encontra e, a partir desse momento, pode mover-se ao longo do labirinto, até chegar à sua saída. À medida que o cilindro se movimenta, a câmara acompanha-o, uma vez que realiza os mesmos movimentos, olhando sempre em direção do centro da cena.

- OrbitControls

Um **OrbitControls** foi adicionado à cena, permitindo interagir com esta movendo o cursor e, inclusive, permitindo visualizar melhor todos os elementos adicionados à cena e as suas posições aquando do desenvolvimento do projeto.

- Outras animações

Ainda existem vários outros tipos de animações, nomeadamente recorrendo ao teclado, referidas na Tabela 1, como a movimentação das Zero Twos e a mudança de luzes.

IX. TRABALHO FUTURO

- Acrescentar texto, apresentando a letra da música de fundo
- Acrescentar um Dat GUI Controls, para ser mais fácil a interação para com o utilizador
- Utilizar texturas, nomeadamente para a neve, fogo, e labirinto
- Entre outros.