Puzzle 3D

Computação Visual

Professores: Joaquim Madeira e Paulo Dias

Filipe Tavares 72063 Tiago Henriques 73046

Resumo **- Este projeto foi proposto na disciplina de Computação Visual e tem como objetivo principal aplicar as técnicas de WebGL aprendidas nas aulas. Para isso, foi desenvolvido um puzzle 3D em que as peças do puzzle são um conjunto de poliedros de diferentes tipos que se podem agrupar de modo a formar uma figura; O utilizador pode seleccionar uma peça de cada vez, orientá-la e (tentar) colocá-la na sua posição final. O objectivo é conseguir colocar todos os poliedros na posição correta no menor número de jogadas possíveis.**

Abstract **- This project was proposed in the Visual Computation subject and its main purpose is to apply the multiple techniques learned throught the classes. So, it was developed a 3D puzzle, where the pieces of the puzzle are a set of polyhedra of different types that can be grouped together to form a figure; The user can select one piece at a time, orient it and, try to, put it in its final position. The goal is to get all the polyhedra in the correct position using the minimum attempts as possible.**

I. INTRODUÇÃO

O trabalho proposto para o projeto da unidade curricular de Computação Visual é um Puzzle 3D desenvolvido em WebGl. Para isso, tentou-se desenvolver uma interface atrativa e com bastantes funcionalidades para o utilizador, usando WebGl.

O jogo é composto por 3 niveis, sendo o inicial bastante simples para ajudar o jogador a ambientar-se ao modo de jogo. Cada nível é concluído com sucesso se o jogador conseguir posicionar os poliedros na posição correta, tendo que consegui-lo num número máximo e fixo de jogadas. Se tal não acontecer, tem de reiniciar esse nível. Cada utilizador começa com 4000 pontos e se não conseguir chegar ao fim dos 3 niveis com um valor superior a 0 pontos tem de recomeçar o jogo desde o 1º nível. Por cada jogada errada o jogador perde 10 pontos.

II. CONCEÇÃO

No desenvolvimento deste projeto foi tomado como base o exemplo 28 do guião 9 de programação em WebGL.

Para tornar o programa útil foi, primeiramente, necessário alterar o ficheiro *initShaders.js* (contém os scripts necessários para inicializar cada shader program e para inicializar o shader.), a função initShaders(gl) foi substituída por duas funções , uma inicia as shaders para poliedros sem textura, pelo que a outra inicia as shaders para poliedros com textura, nomeadamente o background que é nada mais que um tampo de uma mesa simulando a interação do utilizador com um plano de jogo.

As funções *initBuffers()* e *drawModel(...)*, também sofreram algumas alterações tendo em conta os diferentes shaders, sendo também preciso realizar diferentes processos para poliedros com e sem textura. As variáveis, também por sua vez, mudavam de um tipo para o outro (como por exemplo o textureCoords, apenas usado nos poliedros com textura e no background). Na função *drawScene()* não foram feitas muitas alterações em relação ao fornecido no exemplo 28 do guião 9 , apenas se introduziu a transformação global para toda a cena.

Depois disto, foram determinados os limites do canvas e introduziu-se então o background e os poliedros do 1º nível.

Foi introduzido também no ficheiro html os botões que manipulam os poliedros de forma a que estes sejam colocados na posição correta. Também foram colocadas imagens que mostram como os poliedros devem estar posicionados na cena, barras de progresso para dar feedback ao utilizador dos movimentos que vai realizando em cada poliedro, um botão de reset e outro de ajuda, mostrando ao utilizador toda a informação que necessita

para o jogo.
Depois de concluída a construção do primeiro nível, teste de bugs e pequenos ajustes, passou-se para a construção de mais dois níveis, em que se introduziram novas figuras e respetivamente novos botões associados a estas figuras.
No ultimo nível optámos por utilizar apenas poliedros com texturas. Sendo que, na maioria dos casos, cada face do poliedro tem uma textura diferente das outras, ajudando deste modo o utilizador a perceber as posições finais com maior facilidade e assim, reduzir o número de jogadas.

III. ORGANIZAÇÃO

O projeto foi organizado em diversas pastas. Na pasta "css", foram incluídos todos os ficheiros de estilo que foram utilizados.
A pasta "dist", contém os ficheiros necessários para o aparecimento de mensagem de alert especiais.
A pasta "imgs" contém as imagens utilizadas no nosso projeto.
A pasta "js" contém os ficheiros javascript que vão ser detalhados pormenorizadamente em baixo.
A pasta "sounds" contém os ficheiros áudio utilizados.
De referir que o nosso projeto contém devidamente referenciados vários links do bootstrap para a apresentação da informação ao utilizador, por exemplo - a cor e estilos dos botões.

IV. FICHEIROS JS

Os ficheiros javascript onde está a maior parte da implementação, desde "listeners" a código em WebGL estão localizados na pasta de nome "js".

- *initShaders.js*: Contém os scripts necessários para inicializar cada shader program e para inicializar os shaders. Neste ficheiro, está contida a inicialização de shaders sem background, *initShaders(gl),* e a inicialização de shaders com background, como é o caso da textura usada como imagem de fundo, *initShaders_back(gl).*

- *maths.js*: Contém os scripts auxiliares usados nas aulas práticas.

- models.js: Não tem muita relevância no nosso projeto mas foi incluído na mesma. Contém as funções necessárias para processar as "triangle mesh models".

- *webgl-utils.js*: Copyright 2010, Google Inc, All rights reserved.

- *background.js*: Inclui as variáveis e os shader programs necessários bem como as funções responsáveis pela inicialização e carregamento de texturas, usado inicialmente apenas para o background, razão pela qual foi dada este nome.

- *initial.js*: Ficheiro responsável pela população de cada nível. É também encarregue de carregar as imagens, poliedros, botões, etc… específicas para cada nível.

- *progresso_bar.js*: Contém o código para inicialização e o adequado funcionamento da barras de progresso.

- *puzzle.js*: Contém todas as variáveis globais necessárias, contém também as funções *drawModel*, *drawScene*, *handleKeys*, *handleMouseEvents*, *initWebGL*, *runWebGL* e ainda outras funções não tão importantes de referir aqui.

- *vértices.js*: Contém todos os vértices e cores dos diferentes poliedros que vão ser usados em todos os níveis.

- *setEventListeners.js*: Contém os event listeners necessários para os controlos do puzzle e da página.

- *points.js*: Contém as posições corretas de cada uma das variáveis dos diferentes poliedros e as funções responsáveis por verificar a posição das variáveis de cada um dos poliedros.

- *nivel3.js*: Foi criado um ficheiro específico para o nível 3 pois é o nível mais complexo porque contém texturas em cada uma das faces de qualquer poliedro. Contém as inicializações e carregamentos necessários para o bom funcionamento deste nível.

V. BACKGROUND

O background usa uma textura simulando um tampo de mesa de madeira. Pretendemos, em 2D, dar a ideia ao utilizador da colocação sobre a hipotética mesa das diferentes peças de jogo, que variam de nível para nível.

VI. MODELOS (POLIEDROS)

Foram criados modelos para cada poliedro, no nível 1 existem 2 poliedros, um cubo e um pirâmide quadrangular, cada um com cores diferentes em cada face. Foi pretendido a simulação de uma casa.
No nível 2 existem 3 poliedros, uma estrela com 4 pontas, uma pirâmide quadrangular verde e um paralelipipedo castanho. Foi pretendido a simulação de uma árvore de Natal, já que a altura é propícia.
No nível 3 existem também três poliedros, um cubo com uma textura de letra do abecedário em cada face , uma pirâmide com uma textura fixa em todas as faces menos na inferior, esta face contêm a letra V, um paralelipipedo com uma face que contem o número 16 (2016) , outras duas

que contêm uma foto de cada um dos autores do trabalho e uma que contém o icone da Universidade de Aveiro. Foi pretendida a simulação da frase "CV 2016", referindo a cadeira referente ao projeto em questão.

VII. VARIÁVEIS DOS POLIEDROS

Os poliedros são colocados no canvas com um tx e ty *random* dentro de um range pré-definido, as outras variáveis têm um valor definido à priori.

As posições dos poliedros vão afetar o número máximo de jogadas possíveis em cada nível, pois quanto mais afastados os poliedros estão da posição pretendida mais jogadas é preciso realizar para os colocar na posição certa, sendo portanto, necessário ter mais jogadas disponíveis.

Os ângulo Z de rotação dos poliedros dependem do ângulo X e Y destes, o ângulo X de rotação depende do ângulo Y.

Isto acontece devido ao seguinte código colocado na seguinte ordem no *drawModel(…)*, pois a ordem é relevante:

```
mvMatrix = mult(mvMatrix, rotationYYMatrix(angleYY));
mvMatrix = mult(mvMatrix, rotationXXMatrix(angleXX));
mvMatrix  = mult(mvMatrix, rotationZZMatrix(angleZZ));
```

Os jogadores não estão autorizados a efetuar translações das peças para fora do puzzle, ou seja, existe um limite quer para tx, ty e tz, de modo a que o utilizador não movimente a peça para fora do canvas ou para baixo da textura de background (tampo da mesa) e deste modo a tirá-la do campo de visão.

VIII. MOVIMENTO DO RATO

Foi implementado uma translação global em Y e X controlada pelo evento do movimento do rato enquanto o jogador "arrasta" o canvas, para qualquer posição. Desta forma, pretendemos dar uma melhor perceção ao jogador de como a figura está posicionada na cena.

IX. INTERFACE DO UTILIZADOR

Figura 1 – Interface do Utilizador

O interface do utilizador está implementado de modo a disponibilizar ao utilizador botões para controlar translações em X, Y e Z bem como rotações nesses sentidos também. As posições da peça vão ser representadas através de barras de progresso. O utilizador tem também acesso ao nível em que se encontra, o número de jogadas disponíveis para esse nível e a pontuação geral. Estes aspetos que foram referidos vão ser falados com maior detalhe mais em baixo no relatório.

X. CONTROLOS DO UTILIZADOR

A interface de utilizador dispõe de comandos/botões que permitem ao jogador controlar os movimentos dos poliedros. Para realizar translações num dos poliedros é necessário inicialmente escolher sobre qual dos poliedros o utilizador pretende realizar ações.

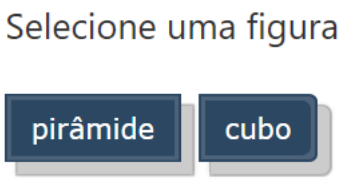

Figura 2 – Seleção da Peça

Depois de selecionado o poliedro, o utilizador deve pressionar qualquer um dos seguintes botões:

Figura 3 - Deslocações em tx, ty e tz

O jogador pode também efetuar rotações em X, Y e/ou Z no poliedro selecionado.Para isso, basta pressionar para qualquer um dos seguintes botões:

Figura 4 - Rotações em X, Y e Z

Para ajudar o utilizador a ter uma noção mais clara do valor real de qualquer uma destas variáveis, foi implementado um sistema baseado em barras de progresso com o propósito de dar um feedback imediato ao utilizador da variável em questão, ou seja, se o valor do mesmo se encontra muito longe ou perto do valor correto para essa variável.

Foi colocada uma barra colorida a vermelho mostrando a percentagem de proximidade do valor atual da variável com o valor correto dessa variável. Valores abaixo de 20% constituem este tipo de barra.

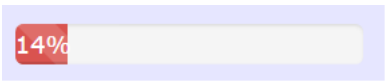

Figura 5 – Barra de Progresso Vermelha

Por outro lado, para valores situados entre 21% e 75%, a barra é colorida a azul, pretendendo mostrar ao utilizador que este se encontra mais próximo do objetivo.

Figura 6 – Barra de Progresso Azul

Para valores situados entre 76% e 99%, a barra é colorida a amarelo e desta forma, indica ao utilizador que o objetivo está bastante próximo de ser obtido.

Figura 7 – Barra colorida a amarelo

Finalmente, para o valor correto dessa variável e corresponde 100%, a cor escolhida para colorir a barra é verde, como não podia deixar de ser.

Figura 8 – Barra colorida a verde

Cada vez que o jogador acerta em qualquer uma destas variáveis do poliedro, aparece uma mensagem a alertar o jogador que a posição da variável em questão foi atingida. Após o aparecimento de tal mensagem, o jogador deve-se então focar em completar as variáveis que faltam.

Figura 9 - Posição Translação X certa

No final do nível aparece também uma mensagem de *Alert* semelhante mas para mostrar que o jogador acertou em todas as variáveis de todos os poliedros desse nível e portanto, encontra-se válido de avançar de nível. Juntamente com essa mensagem de *Alert, com o intuito de* melhorar a experiência do jogador e a interação ser mais memorável para o utilizador, foi também introduzido um som de "resposta correta". Os ficheiros áudio encontram-se na pasta "sounds".

A pontuação é disponibilizada ao utilizador num círculo verde no canto superior direito. Cada jogador começa com uma pontuação fixa de 4000, tendo como objectivo chegar ao fim dos 3 níveis com uma pontuação superior a 0. Sempre que o jogador completa um nível pode voltar ao mesmo para melhorar a sua pontuação nesse nível e por conseguinte a pontuação geral. Para cada nível, existe um número máximo fixo de jogadas possíveis. Se o número de jogadas possíveis desse nível chegar a 0, o jogador terá de recomeçar esse nível do início, mas agora, com novas posições dos poliedros para tornar possíveis estratégias de memorização de jogadas inútil para o jogador. Se por outro lado, o jogador ficar com 0 pontos antes do fim do 3º nível, isso significa que terá de começar o puzzle do início (nível 1). Quando isto acontece aparece uma mensagem de *Alert* ao jogador juntamente com um som de "game over" mostrando ao jogador que para a próxima terá de utilizar outra estratégia.

Cada nível também possui um botão de reset e um botão de ajuda.

Figura 10 – Botões de reset e ajuda

Quando o botão de ajuda for pressionado, aparece a seguinte informação:

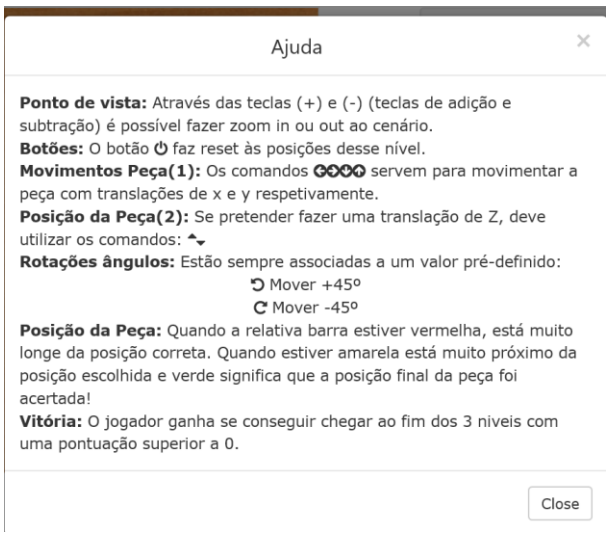

Figura 11 – Ajuda requerida

Possui também tooltips que surgem com o passar por cima do rato de determinados botões, de modo a ajudar o jogador a perceber a função desse botão, como por exemplo:

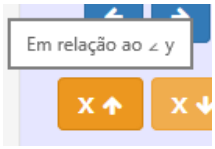

Figura 12 – Exemplo de Tooltip

Por fim, existem também 2 imagens por cada nível que mostram a posição correta dos poliedros, cada uma com uma perspetiva diferente.

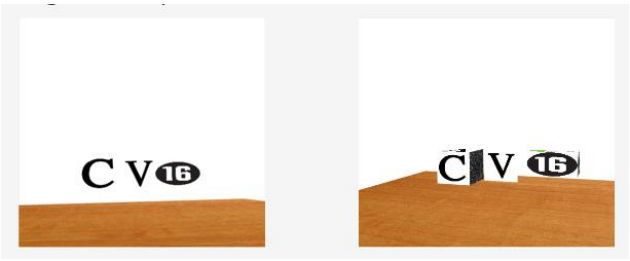

Figura 13 – Imagens Objetivo do nível 3

XI. USO DO TECLADO

Foi implementado e adaptado de aulas práticas da cadeira, a geração de eventos através do uso de teclas, no nosso caso, usámos a tecla (+) – Adição e (-) – Subtração para fazer Zoom In ou Zoom Out, respetivamente, ao cenário de fundo pretendendo desta forma ajudar o utilizador a ter uma visão e perceção mais acentuada da verdadeira distância entre poliedros.

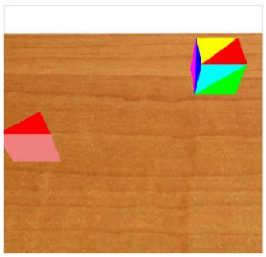

Figura 14 – Zoom In (+)

Figura 15 – Zoom Out (-)

XII. INFORMAÇÕES ADICIONAIS

O jogo foi testado em 2 diferentes browsers: *Firefox* e *Edge*, sendo que não houve qualquer problema nestes browsers.

Devido ao uso de texturas locais não é possível executar o jogo no *Google Chrome* pois este não suporta as texturas importadas localmente.

XIII. CONCLUSÃO

O principal objetivo do projeto foi atingido pois tendo por base um conjunto de poliedros de diferentes tipos que se podem agrupar de modo a formar um prisma, foi possível fazer com que o utilizador pudesse selecionar uma peça de cada vez, orientá-la e colocá-la na sua posição final. Optou-se por usar algum do código usado nas aulas práticas e fazer apenas uma reformulação tendo como objetivo instanciar objetos independentes, com atributos diferentes, translações e rotações. Deste modo, foi possível desenvolver um puzzle 3D com determinadas peças por nível de uma forma relativamente simples. Os principais problemas prenderam-se em desenvolver esta aproximação em modelos independentes e na criação de shaders de poliedros e shaders de um modelo com textura como imagem de fundo a serem processados em simultâneo. No entanto, o projeto acabou por ser realizado com todas as funcionalidades pedidas e algumas que foram adicionadas no decorrer do desenvolvimento do projeto de forma a facilitar a interação do utilizador com o jogo.

XIV. REFERÊNCIAS

http://sweet.ua.pt/jmadeira/CV/index.html

http://stackoverflow.com/

http://learningwebgl.com/blog/?page_id=1217

http://webglfundamentals.org/

http://www.w3schools.com/js/default.asp

http://getbootstrap.com/