Pista de carros 3D em WebGL

Márcia Pires, 88747 Tomás Martins, 89286

***Abstract* – This report describes in detail the development of a 3D application using the Javascript API, WebGL. This application allows users to view and manipulate a car movement in a track. The report starts by introducing the idea for the application as well as the objectives set for it. After that, it presents the main points for the development of the project and how it was implemented, and the available features are described aswell. At last, the results obtained are presented along with some observations.**

***Resumo* – Este relatório descreve de forma detalhada o desenvolvimento de uma aplicação 3D utilizando a API em Javascript, WebGL. Esta aplicação permite ao utilizador visualizar e manipular pistas de carros de forma interativa. O relatório começa por introduzir a ideia para a aplicação bem como os objetivos estabelecidos para a mesma. De seguida, são apresentados os principais pontos para o desenvolvimento do projeto e como foi feita a sua implementação, e também são descritas as funcionalidades disponíveis. Finalmente, são apresentados os resultados obtidos juntamente com algumas observações.**

I. INTRODUÇÃO

Desenvolvida no âmbito da cadeira de Computação Visual, a aplicação, tal como o nome sugere, *Pista de carros*, consiste na apresentação de diferentes pistas de carros, em que um carro se movimenta pela mesma, podendo todo o sistema ser manipulado interativamente pelo utilizador, seja variando a velocidade do veículo, escolhendo uma das diferentes pistas, personalizar o carro ou limitar o número de voltas pelo percurso.

Uma vez que a sugestão para o tema nos foi proposta, realizámos uma pesquisa detalhada de maneira a compreender quais as funcionalidades que pretendíamos implementar e que tirassem o melhor partido daquilo que foi aprendido nas aulas práticas ao longo do semestre.

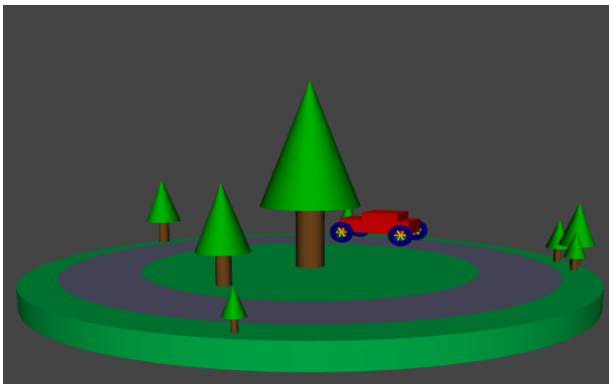

Fig. 1 - Exemplo sobre o qual nos inspiramos para implementar o nosso projeto.

II. IMPLEMENTAÇÃO

A. Módulos

Por forma a ser feita uma boa implementação do projeto, foi necessário estruturar de forma organizada e clara o mesmo, tendo sido o resultado obtido o conjunto de ficheiros em seguida descritos de forma sucinta.

- **models/cars.js** — Conteúdo necessário para a criação do modelo do **carro**.
- **models/cylinder.js** — Conteúdo necessário para a criação do modelo do **cilindro** para o tronco de uma árvore.**
- **models/flagmark.js** — Conteúdo necessário para a criação do modelo da **bandeira de partida**.
- **models/sceneModels.js** — Base para a criação de um modelo.*
- **models/track_x.js** — Quatro ficheiros diferentes, em que x varia entre 1 e 4, sendo que cada um deles fornece o conteúdo necessário para a criação do modelo de uma **pista diferente**.
- **models/treetop_pir.js** — Conteúdo necessário para a criação do modelo do **topo de uma árvore em pirâmide**.
- **models/treetop_quad.js** — Conteúdo necessário para a criação do modelo do **topo de uma árvore em paralelipípedo**.
- **models/world.js** — Conteúdo necessário para a criação do modelo da **base do "mundo"**.
- **utils/initShaders.js** — Inicialização e criação de **shaders**.*
- **utils/lightSources.js** — Definição de **focos de luz** a serem usados.*
- **utils/maths.js** — Funções **matemáticas** auxil-

iares.*
- **utils/webgl-utils.js** — Funções para a **compatibilidade** entre *browsers.**
- **carMovement.js** — Funções para a lógica da **movimentação do carro**.
- **Road.js** — Classe para definir as **pistas**.
- **main.js** — Lógica de alto nível para a **total funcionalidade** da aplicação, executando todo o Javascript.*
- **index.html** — *HTML* necessário para a apresentação e execução da aplicação.
- **getSquareFlag.py** — Função para **gerar os vértices dos quadrados** da bandeira de partida
- **style.css** — Pequeno pormenor de estilo de página *HTML*.

Os ficheiros assinalados com * referem-se àqueles que usam, parcial ou totalmente, código fornecido nas aulas práticas. Já o ficheiro assinalado com ** refere-se a um ficheiro com conteúdo de código externo, referenciado na secção **Referências**.

B. Modelarização

Relativamente aos modelos existentes, optámos por desenvolver os nossos próprios modelos, baseando-nos naqueles que foram desenvolvidos em aula. Cada modelo é constituído por um **nome**, um **conjunto de vértices** que dá a forma ao modelo e um **conjunto de cores e cores iniciais** que coloram o modelo a determinada altura, isto é, o conjunto de cores iniciais serve para guardar as cores originais do modelo, antes do mesmo sofrer alguma alteração (por *input* do utilizador por exemplo ou por iluminação).

O primeiro modelo a ser criado foi a base que sustenta todo o sistema, daí o nome de *"world"*, sendo que o mesmo consiste num paralelipípedo. Em seguida, o modelo principal, o **carro**. Este é constituído por um paralelipípedo principal, um paralelipípedo menor por cima, conferindo-lhe altura e forma, e quatro pequenos paralelipípedos representando as rodas.

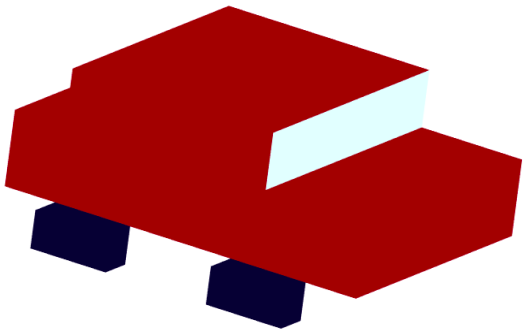

Fig. 2 - Modelo do carro.

Outro elemento bastante importante, são as **pistas de corrida**. Estas pistas, possuem **quatro versões diferentes**, em que a complexidade de percurso aumenta gradualmente, sendo compostas por retângulos. Para essas pistas, foram acrescentados modelos para criar um ambiente mais composto, sendo eles uma **bandeira de partida**, composta por paralelipípedos e retângulos, e **árvores**:

1. composta por um cilindro como tronco e uma pirâmide como copa;
2. composta por um cilindro como tronco e um paralelipípedo como copa;
3. versão igual à anterior mas com tamanho menor.

.

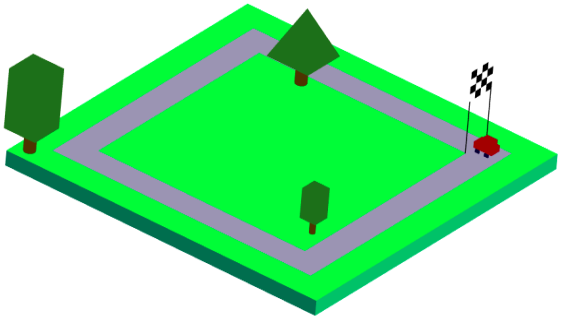

Fig. 3 - Primeira pista, com os elementos complementares de cenário.

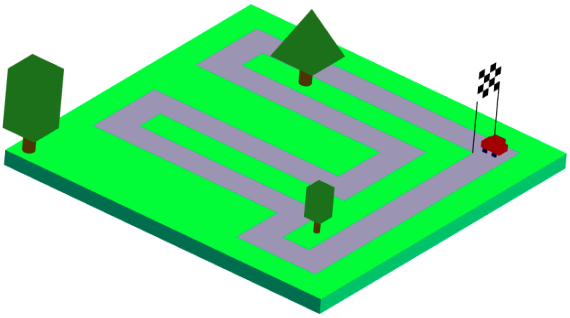

Fig. 4 - Quarta pista, com os elementos complementares de cenário.

Para a implementação de todos estes modelos, fizémos uso da class **emptyModelFeatures()**.

C. Funcionalidades

O utilizador, na página da aplicação, *index.html*, consegue aceder às seguintes funcionalidades:

1. Escolher a opção **Show world** e ter acesso a:
   - **Escolher a pista** na qual deseja que o carro corra, dentro de uma opção de quatro pistas diferentes.
   - **Definir a quantidade de voltas** que deseja que o carro dê à pista selecionada, ou, em alternativa, não introduzir nenhum valor e **deixar que o carro percorra tantas voltas quantas quiser**. Daí, pode também observar a quantidade de voltas que o carro já fez ao percurso, através do valor das **current laps**.

- **Iniciar uma corrida**, seja com os *inputs* passados anteriormente ou correndo com os default; **Pausar uma corrida** e ainda **Continuar uma corrida**, caso ela tenha sido interrompida anteriormente.
- Além disso, é possível **ajustar a velocidade de movimento do carro**, com ele em movimento ou não, aumentando ou diminuindo a mesma, numa escala de 0 a 4, com incrementações/decrementações de 0.25/clique.
- É possível visualizar a aplicação em 3D em dois tipos de projeção diferentes: **projeção ortogonal** ou **projeção em perspetiva**.
- Através do teclado, também é possível **manipular a rotação dos ângulos de visualização**:
  - Seta para a esquerda/seta para a direita para o ângulo XX;
  - Letra Q/letra A para o ângulo YY;
  - Seta para cima/seta para baixo para o ângulo ZZ.
- É possível dar *reset* da corrida através do botão **Reset race** ou dar reset de todas as definições através do botão **Reset all**.
- Para a **iluminação**, esta pode ser ou não ativada, através dos respetivos botões.

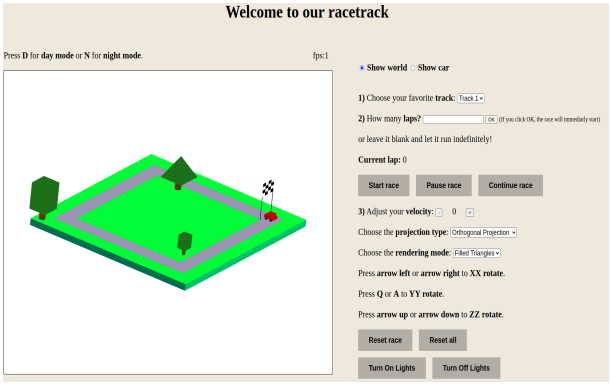

Fig. 5 - Página do **Show world**.

2. Escolher a opção **Show car** e ter acesso a, além da manipulação através do teclado referida anteriormente:
   - **Alterar as cores do carro** para uma de quatro cores disponíveis. Esta alteração é vísivel também quando se volta para a opção **Show world** e assim é possível fazer corridas com o carro na cor desejada.

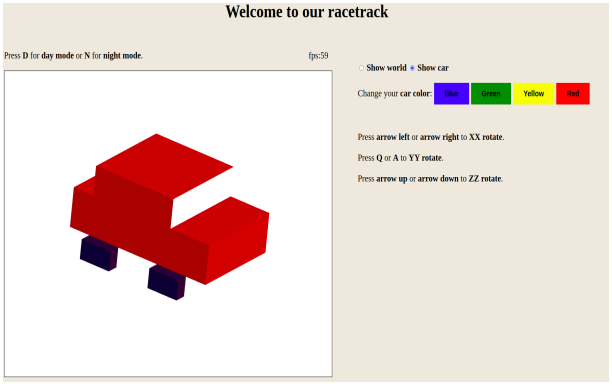

Fig. 6 - Página do **Show car**.

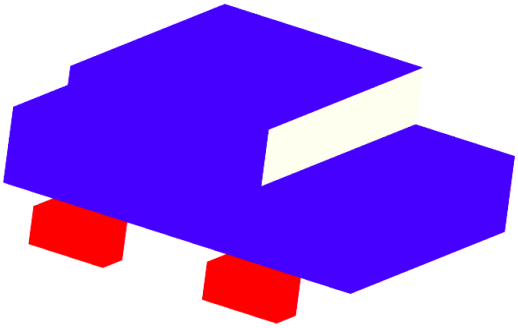

Fig. 7 - Carro com a cor alterada.

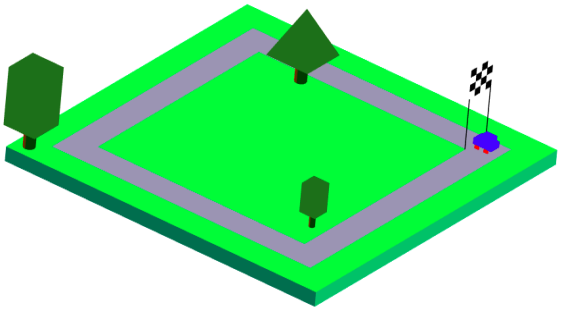

Fig. 8 - Carro com a cor alterada, a correr na pista.

Existe ainda a funcionalidade comum em ambas as opções que permite, através do teclado, selecionar a **tecla D** para o day mode ou a **tecla N** para o night mode.

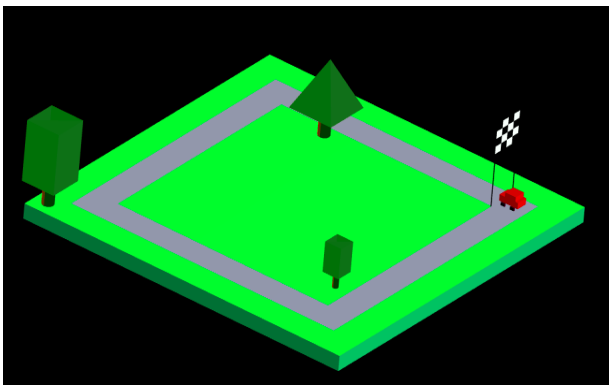

Fig. 9 - Opção **night mode** ativada.

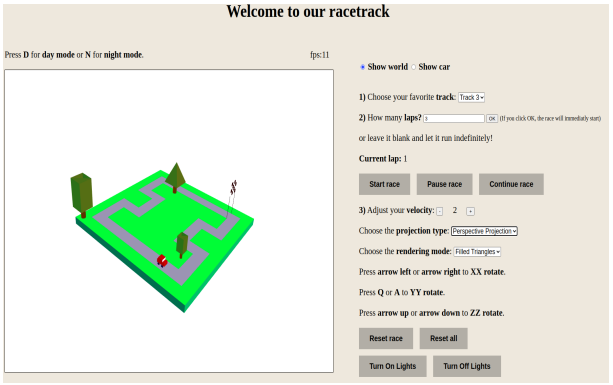

Fig. 10 - Funcionalidades com vários inputs a correr.

D. Implementação das funcionalidades

D.1 Voltas e velocidade do carro

O veículo pode-se movimentar em **2 modos distintos**:

- **Indefinidamente -** O movimento só para quando requisitado pelo utilizador.
- **Pré-definido -** O carro irá suspender o movimento, assim que atingir o número de voltas especificado previamente pelo utilizador.

A movimentação e velocidade do carro vão ser definidas alterando a **matriz de translação** do carro através das seguintes expressões, dependendo da direção:

- **"X positivo" -** *tx += 0.01 * **car_speed***
- **"X negativo" -** *tx -= 0.01 * **car_speed***
- **"Z positivo" -** *tx += 0.01 * **car_speed***
- **"Z negativo" -** *tx -= 0.01 * **car_speed***

A variável *"car_speed"* pode ser alterada através do input do utilizador. Este valor nunca pode ser maior que 4, nem menor que 0 e tem como valor padrão em movimento, de 1. É também este valor que controla o estado do carro, ou seja, se está parado ($car\_speed = 0$) ou se está em andamento ($car\_speed > 0$). É possivel também reiniciar o estado do carro, tanto em posição como em velocidade, através do método *reset_car()* e iniciar a trajetória através de outro método *start_car()*.

D.2 Pistas

Para melhor organização, o sistema de pistas é controlada através de uma classe *Road*, que define os movimentos que a viatura tem de executar para cada circuito da aplicação.

Esta classe, contém os arrays *"directions"* e *"positions"*. O primeiro, é composto por as instruções que o carro deve executar por ordem: *"X_Pos"*, *"X_Neg"*, *"Z_Pos"* ou *"Z_Neg"*.

O segundo, é composto por tuplos que simbolizam o ponto limite que o carro deve atingir, até processeguir para a próxima direção.

Estes arrays são controlados por um index comum a ambos, que vai incrementando à medida que o veiculo progride na pista .

D.3 Movimento do carro

A lógica da movimentação do carro é implementada no ficheiro *carMovement.js*. A mesma é conseguida **alterando a matriz de transformação** local do modelo do veiculo, em função de uma variável de velocidade incrementada pelo utilizador, tal como descrita anteriormente, de acordo com o percurso atualmente em utilização.

Este percurso é escolhido através de um array com todos os objetos do tipo *Road* mencionado anteriormente. Dependendo da pista em questão, o carro pode-se movimentar em quatro direções distintas: em direção ao eixo positivo ou negativo dos *x's* e em direção ao eixo negativo ou positivo dos *z's*. Assim que a viatura atingir o ponto limite definido pela *Road*, é incrementado o index da mesma, passando assim à próxima posição e direção. Quando o veiculo chegar ao final da pista, este index vai ser reiniciado. Durante a mudança de direção, o veiculo compara a atual direção e a seguinte, de modo a saber que rotação deve aplicar em si mesmo, de modo a processeguir para o próximo caminho. Esta rotação é aplicada localmente no eixo dos *y's* do modelo do carro.

D.4 Iluminação

A iluminação funciona através do ficheiro lightSource.js, cujo propósito serve para criar a nossa única fonte de luz e os seus métodos auxiliares.

A sua utilização é feita na *main.js*, na função computeIllumination, que dado um modelo, vai alterar o valor das cores desse objeto, com os atributos da própria fonte de luz. A luz vai gradualmente aumentando e através de um contador, conseguimos restringir a mesma a um valor pretendido. Se a luz for desligada pelo utilizador, ou se ocorreu uma mudança de luz natural (dia para noite ou vice-versa), então a luz vai reiniciar para restituir ao modelo, as suas cores originais.

Durante a funcionalidade do "dia", a intensidade e a ambientação da fonte de luz vão ser consideravelmente maiores em comparação com a luz da funcionalidade "Noite" de maneira a recriar um realismo entre as diferentes alturas do dia.

III. CONCLUSÃO

Tal como esperado, a realização desta aplicação permitiu-nos aprofundar e aplicar os conhecimentos aprendidos nas aulas práticas de Computação Visual, desafiando as nossas capacidades de implementação dos conceitos para obter as funcionalidades pretendidas.

Apesar de muito satisfeitos com o resultado final, a implementação de algumas funcionalidades não correu da maneira que pretendíamos, como por exemplo as luzes, pois por vezes surge com alguns erros e comportamentos errados e inesperados. Esta aplicação obtém uma performance ligeiramente melhor no navegador Opera , relativamente ao navegador Firefox. Ainda assim é possivel usufruir de todas as funcionalidades em ambos os navegadores.

Além disso, por vezes pode-se deparar com uma queda frame rate, maioritariamente devido à iluminação, sendo que o problema é facilmente resolvido assim que a iluminação for desligada.

Alcançámos os objetivos que nos propusémos a realizar, sendo que algumas funcionalidades foram ultrapassadas com sucesso, enquanto outras acabámos por ter mais dificuldade, mas concordamos que o balanço final foi bastante positivo. A percentagem de divisão de trabalho é de 50/50.

IV. REFERÊNCIAS

- Joaquim Madeira - Diretor do MIECT do DETI - Código adaptado dos guiões das aulas

References

https://cse.taylor.edu/~jdenning/classes/cos350/slides/08_Cylinders.html