Pong - WebGL

José Cristiano Santos Moreira Nº79671, Guilherme Martins Henriques Nº80100

***Resumo* - Este relatório tem como objetivo expor o projeto realizado no âmbito da Unidade Curricular de Computação Visual, Pong, que foi feito recorrendo à ferramenta WebGL. Será abordado o jogo em si, não só como o mesmo funciona mas também o que foi necessário utilizar para que o mesmo seja jogável. Serão também abordados outros fatores que apenas afetam indiretamente o jogo em si, mas foram colocados no projeto de forma a demonstrar mais alguns dos conhecimentos adquiridos sobre esta mesma ferramenta.**

***Abstract* - The main goal of this report is to expose the project developed regarding the course Computação Visual, Pong, that was made using the WebGL tool. It will approach the game itself and how it works, as well as everything that was needed for it to be playable. Furthermore, it will talk about other factors that do not affect que game or the gameplay itself, but were used in the project to exhibit some knowledge acquired about this same tool.**

I. Introdução

O projeto realizado no âmbito da unidade curricular Computação Visual é o jogo Pong. É o primeiro jogo de vídeo lucrativo da história e é muito simples, consistindo em apenas um campo, 2 "sticks", que correspondem a cada jogador, e uma bola. O objetivo de cada jogador é evitar que a bola ultrapasse o seu "stick" (horizontalmente) ao mesmo tempo que tenta que o seu adversário não o consiga fazer com sucesso. Para o desenvolvimento do projeto, foi utilizada a ferramenta WebGL, para a criação dos modelos necessários, HTML, para o display do jogo e pontuações, bem como algumas instruções para os utilizadores, e JavaScript, para programar tudo o que estava relacionado com o jogo em si, para além dos modelos.

II. Ferramentas Utilizadas.

Como já foi explicitado, para o projeto foram utilizadas as ferramentas WebGL, JavaScript, e HTML. WebGL, que é uma Api que permite a renderização de modelos através do elemento Canvas de HTML, foi utilizado para a renderização dos modelos desenvolvidos. HTML foi utilizado para a edição da página Web onde é possível dar display do jogo em si, ou seja, demonstrar a renderização, e também o resultado atual, bem como algumas instruções para os utilizadores que serão abordadas mais à frente. JavaScript foi utilizado para tudo o resto, seja a criação dos modelos, interações entre os mesmos, contagem de pontos, e iluminação.

Todos os modelos foram criados com recurso ao ficheiro SceneModels.js, criando assim um array com três cubos, um para o background, e dois para servirem de "sticks", e uma esfera, para servir de bola. Uma vez que o campo é retangular, o tamanho do elemento canvas é 800x400, foi necessário um ajuste em todos os valores, duplicando os valores de x em relação ao valores de y, para não obter modelos deformados.

```
sceneModels.push( new cubeModel( subdivisionDepth: 1 ) );

sceneModels[3].tx = 0.0; sceneModels[3].ty = 0; sceneModels[3].tz = -1.0;

sceneModels[3].sx = 1.4; sceneModels[3].sy = 1.4 ; sceneModels[3].sz = 0.1;

sceneModels[3].kDiff = [ 1.0, 1.0, 0.0 ];

sceneModels[3].kSpec = [ 1.0, 1.0, 0.0 ];
```

Fig. 1 - Exemplo de criação de modelo.

Relativamente à iluminação, apenas existe uma fonte, criada através do ficheiro lightSources,js.

```
var lightSources = [];

// Light source 0 and 1 WHITE

lightSources.push( new LightSource() );

lightSources[0].setPosition( 0.0, 0.0, 1.0, 1.0 );

lightSources[0].setIntensity( r: 1, g: 1, b: 1);
```

Fig. 2 - Instanciação da fonte de luz.

A posição da fonte de luz é móvel em (x, y), sendo instanciada em (0, 0, 1, 1), mas depois ajustável através do rato, e das suas coordenadas atuais.

```
var deltaX = newX - lastMouseX;

lightSources[0].position[0]+=(radians(  degrees: 2.0 * deltaX  ));

var deltaY = newY - lastMouseY;

lightSources[0].position[1]-=(radians(  degrees: 1.0 * deltaY  ));
```

Fig. 3 - Atualização da fonte de luz com coordenadas do rato.

Relativamente ao jogo em si, a detecção de colisão entre os "sticks" e a bola é feita através de comparações. No caso de uma colisão com a parte lateral do "stick", se a esfera estiver na posição horizontal adjacente ao "stick", verifica-se se a posição vertical da bola coincide com a posição vertical do "stick". Caso esta condição se verifique, a direção da bola é invertida. O processo é semelhante para todas colisões, sejam nos limites do campo, ou na parte superior/inferior do "stick".

```
if((sceneModels[2].tx >=  0.86 && sceneModels[2].tx <=  0.90 )){
    if(sceneModels[1].ty >= (sceneModels[2].ty-0.25) && sceneModels[1].ty <= (sceneModels[2].ty+0.25)){
        if(right){
            right = false;
            count++;
        }
    }
}
```

Fig. 4 - Exemplo de detecção de colisão.

Aproveitando a deteção de colisões, é a partir deste mecanismo que é possível fazer a contagem de pontos de cada jogador. Através da atribuição de um contador a cada jogador, sempre que a esfera colide com o limite direito do campo, o "stick" do lado esquerdo recebe um ponto, sempre que a esfera colide com o limite do lado esquerdo, o "stick" do lado direito recebe um ponto, Pontos estes que são enviados automaticamente para o ficheiro HTML, estando assim a scoreboard sempre atualizada. Sempre que um ponto é marcado, a bola reinicia no centro do campo, ganhando uma direção atribuída aleatoriamente, para o primeiro jogador a reagir não ser sempre o mesmo, nem precisar de se mover semper na mesma direção, e o seu nível de velocidade, que será abordado mais à frente também passa a um. O nível de velocidade, como o nome indica, é a velocidade a que a esfera se movimenta, sendo aumentado automaticamente por cada 4 toques que a esfera dê nos "sticks", ou também pode ser ajustado (aumentado/diminuído) nos botões disponíveis na página Web, ou então através das teclas "+" e "-", se bem que apenas funcionam se o jogo estiver em pausa, não sendo possível fazer qualquer ajuste de velocidade com o jogo a decorrer.

```
if(sceneModels[2].tx >=  0.96){
    count = 0;
    nivel = 1;
    playerOne++;
    sceneModels[2].tx = 0;
    sceneModels[2].ty = 0;
    right = Math.random() >= 0.5;
    up = Math.random() >= 0.5;
}
```

Fig. 5 - Contagem de pontos, reset posição da bola, reset nível de velocidade.

É também possível a qualquer momento colocar o jogo em pausa, mais uma vez, seja através de um botão na página Web, ou através da tecla "P". A qualquer momento, e pensando numa possível troca de adversário, ou simplesmente um recomeço, é também possível dar reset ao jogo em si, colocando a scoreboard a zeros, novamente utilizando o botão disponível na página Web, ou através da tecla "R".

Press P to play/pause and R to Reset Game Play/Pause Reset

Change speed level with + and - keys (Paused Only) + -

Fig. 4 - Menu disponível para o utilizador.

```
case 82 : //Reiniciar o jogo
    right = Math.random() >= 0.5;
    up = Math.random() >= 0.5;
    nivel=1;
    status = 'begin';
    count = 0;
    playerOne = 0;
    playerTwo = 0;
    sceneModels[2].tx = 0;
    sceneModels[2].ty = 0;
break;
```

Fig. 6 - Exemplo do código para dar reset ao jogo.

Por fim, o movimento dos “sticks” é feito apenas com teclas. O jogador do lado esquerdo utiliza as teclas “W” e “S” para mexer o seu “stick” para cima e para baixo respectivamente, já o jogador do lado direito utiliza as teclas “ ↑ ” e “ ↓ ” para mexer o seu “stick” para cima e para baixo respetivamente. Isto é feito através da função handleKeys(), cujo excerto está representado na figura abaixo.

```
function handleKeys() {
    if (status=='play'){
        if (currentlyPressedKeys[87]) {

            // W key
            if (sceneModels[0].ty < 0.75) sceneModels[0].ty += 0.03;
        }
```

Fig. 7 - Excerto da função que permite movimento dos “sticks”.

II. Conclusão

Em suma, alcançamos quase todos os nossos objetivos, faltando apenas uma ligeira correção nas colisões entre as partes de cima e de baixo dos “sticks” com a esfera, e também um ângulo de visão diferente, para diferenciar a única opção, que é uma vista superior.