Tic Tac Toe para Profissionais

João Branquinho Tiago Ramalho

***Abstract* – This article presents the Tic Tac Toe game developed in 3D. This project was developed in scope of the Visual Computation Course Unit. The group aimed to recreate the game in a dynamic and interactive approach. In this article is possible to read about how the problem was both presented and dealt with, the difficulties that were presented and how those difficulties where addressed.**

***Resumo* –O presente artigo apresenta o jogo do galo em 3D, um projeto desenvolvido no âmbito da unidade curricular de Computação Visual que pretende, de forma interativa e dinâmica, recriar o jogo de crianças a 3 dimensões. No decorrer do artigo é possível ler sobre a forma como o problema foi apresentado ao grupo, a forma como foi abordado, as dificuldades encontradas para a sua realização e a forma como essas dificuldades foram contornadas.**

***Keywords* –Visual Computing Tic Tac Toe Three Dimensions Web GL Rendering Textures Picking**

***Palavras chave* –Computação Visual Jogo do Galo 3D Três dimensões Renderização Texturas Seleção de Objetos**

I. Introdução

No contexto da disciplina de Computação Visual, do 4º ano do Mestrado Integrado em Engenharia de Computadores e Telemática foi desenvolvido uma aplicação web que permite que os seus utilizadores joguem o conhecido e tradicional jogo do galo em 3D. A aplicação permite que esta experiência seja feita num campo realista e com altos níveis de interatividade Humano-Computador.

Conforme desenvolvido e praticado nas aulas práticas e teóricas desta UC o esqueleto deste projeto está desenvolvido em *HTML/JS* com recurso à API *Javascript WEB GL*. Todo o trabalho desenvolvido está repartido por 2 ficheiros *.js* por questões de organização. Para esta implementação foram utilizados 2 *canvas* que permitem facilitar o processo de *picking* do qual falaremos em profundidade no decorrer deste relatório.

Ao longo do desenvolvimento deste projeto o grupo deparou-se com várias dificuldades que permitiram obter um maior conhecimento da ferramenta *WEB GL*. As várias dificuldades prenderam-se com:

- A estruturação inicial do projeto
- A interatividade no campo de jogo
- Seleção de um objeto - *picking*

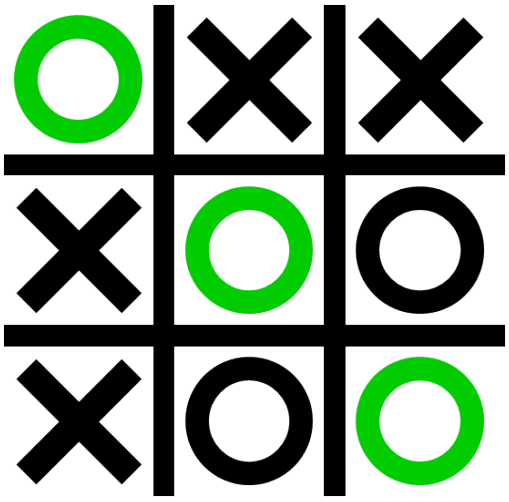


Fig. 1: Tabuleiro jogo tradicional

A informação disponibilizada na *Web*, para além de frequentemente estar noutras linguagens (como c++), maior parte das vezes utilizava frameworks de alto nivél (tree.js). Por este facto a progressão no projeto tornava-se por vezes complicada.

II. Introdução ao jogo

A. Jogo Tradicional

Na versão tradicional do jogo do galo(também conhecido por jogo da velha) o tabuleiro é uma matriz de três linhas por três colunas (ver Figure 1);

- Dois jogadores escolhem uma marcação cada, geralmente um círculo (O) e uma cruz (X);
- Os jogadores jogam alternadamente , uma marcação por vez, numa posição que esteja vazia;
- O objetivo do jogo é conseguir atingir três figuras numa mesma linha, seja ela horizontal, vertical ou diagonal.
- Quando um jogador conquista o objetivo do jogo costumam riscar-se os 3 símbolo numa linha oblíqua.

B. Versão modificada

A proposta feita pelos professores da cadeira é um pouco diferente. O campo de jogo deve ter mais uma dimensão (profundidade), dado que o numero do posições a jogar deve ser multiplicado por três. As regras do jogo não variam pelo que continua a ser possível ganhar vertical, horizontal ou diagonalmente agora em mais uma dimensão.

III. Ideia inicial

Para realizar este projeto o primeiro passo foi a análise das várias abordagens possíveis e consequente decisão das mesmas. Quando o grupo se sentou para realizar um *brainstorming* de realização do projeto estas foram algumas das ideias que se apresentaram:

- Desenho de um cenário de jogo que contem um total de vinte sete cubos (3 x 3 x 3)
- Recurso a texturas para representação de jogadores.
- Visualização do campo de jogo sempre à mesma distância. Câmara circula sobre uma superfície esférica.
- Projeção perspetiva para permitir sensação de profundidade.
- Boa *IHC* (Interação Humano-Computador) - com recurso ao rato para uma movimentação espacial e ao *click* para seleção da jogada.

Esta foi a ideia inicial do projeto. Como veremos nas secções seguintes apresentaram-se vários problemas que obrigaram à realização de outras soluções.

IV. Implementação

No desenvolvimento do jogo do galo 3D o grupo deparou-se com algumas dificuldades, tendo conseguido contorná-las. De seguida vamos apresentar vários sub-capítulos onde falaremos destes problemas e respetivas soluções.

A. Interatividade no Campo de Jogo

Para obter uma melhor interação humano-computador e uma maior dinamicidade o grupo decidiu que seria possível rodar o campo de jogo livremente. Ao clicar e arrastar o rato é possível rodar o cenário sobre o seu centro permitindo ao jogador visualizar todos os cubos e jogadas possíveis. A dificuldade nesta rotação provêm do calculo da matriz, pois para isto acontecer é necessário calcular uma matriz temporária, que é o produto da matriz de rotação segundo o eixo do **X** com a matriz de rotação segundo o eixo do **Y**, e a partir desta é que se calcula a *mvMatrix*. Se multiplicássemos separadamente as matrizes de rotação o cenário só rodaria segundo um eixo.

B. Seleção do Cubo

A funcionalidade em que o grupo sentiu maior dificuldade foi a seleção do cubo que ditaria a jogada do jogador. Inicialmente a abordagem foi matemática, isto é, através das matrizes de rotação globais e do vetor proveniente das coordenadas do *click* do rato e da profundidade do cenário, calcular-se-ia a interseção do vetor com um dos planos do cubo. Esta abordagem embora interessante e aparentemente fácil, tornou-se bastante complexa devido ao elevado número de cubos e consequentemente grande número de planos. Inicialmente pensou-se em intersetar o vetor calculado com os cubos depois de aplicar as transformações da *mvMatrix* a estes últimos, esta solução embora trabalhada não foi possível realizar devido ao grande número de planos e de interseções que existiriam com os planos dos diversos cubos tendo depois de ser preciso selecionar o cubo mais próximo se caso houve-se interceção com mais de que um cubo. Computacionalmente seria uma solução mais exigente do que a apresentada posteriormente neste relatório. Matematicamente o grupo tentou fazer algo similar aplicando as transformações da *mvMatrix* ao vetor e não aos cubos contudo esta solução embora mais elegante do ponto de vista do grupo, levantaria novamente alguns dos problemas descritos anteriormente e que não foi encontrada solução a nível matemático.

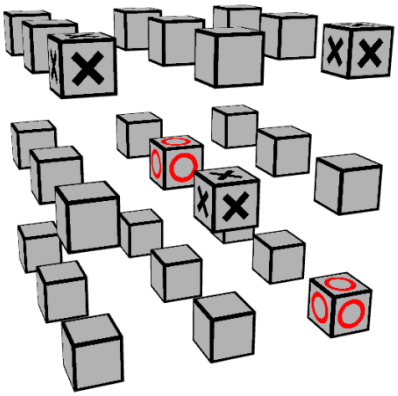
Ao longo de alguma pesquisa de como fazer o *picking* (seleção do cubo) foi encontrada a abordagem, que foi utilizada no projeto. Esta aproximação exige um esforço computacional menor mas mais memória. Esta solução é bastante simples: utilizando um novo *canvas* e com recurso a vinte sete *color buffer* cores *random* uma para cada cubo. O calculo do cubo clicado é feito da seguinte forma:

1. Através do *click* no *canvas 1* (Figure 2a) é calculada a posição relativa do *click* relativamente às coordenadas base do canvas;
2. Após serem obtidas essas coordenadas, e dado que o *canvas 2* (visível na Figure 2b) tem as mesmas dimensões que o *canvas 1*, são posicionadas sobre o *canvas 2* essas mesmas coordenadas e é lida a cor do pixel que essa coordenada representa;
3. Cada cor tem associada a si um cubo e dado que essa cor é única é possível deduzir em que cubo foi realizado o *click* e consequentemente proceder à jogada.
4. É alterada a textura no *canvas 1* caso a jogada seja válida e aquela posição no campo de jogo fica associada a um jogador.

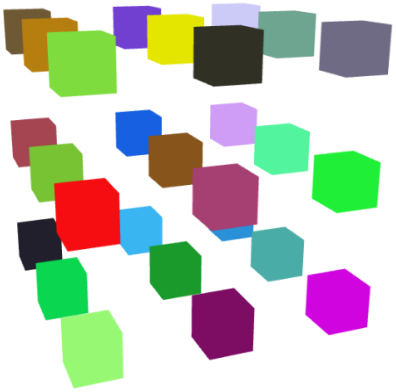
Para esta solução o *canvas 2* só tem de ser desenhado (função *DrawScene2()*) quando se deteta um *click* sobre um cubo. Pois só é necessário realizar a operação a cima descrita quando se deteta uma tentativa de jogada, e é por isso que esta solução é computacionalmente eficiente para o objetivo deste projeto.

B.1 Limpeza de variáveis locais

Ao tentar concretizar esta segunda solução também surgiu um problema relacionado com a limpeza de variáveis locais nomeadamente dentro da função *DrawScene2()*. Para realizar o *picking* é necessário fazer a leitura da cor do pixel que estava a ser clicado. Para isso, o grupo, tentou por várias formas executar a função *readPixels()* (documentada aqui) que permite a extração dessa informação. Esta função tem um problema para o qual a informação na *web* é escassa: dado que o cenário é desenhado na função *DrawScene2()* o *WEB GL*, para poupar espaço em memória, apaga as variáveis locais associadas ao desenho do cenário, pelo



(a) Cenário Real de Jogo (Canvas 1)



(b) Cenário com cubos de diferentes cores (Canvas 2)

que a execução da função *readPixels()* deve ser feita na mesma função em que este é desenhado. Este *sincronismo* não é referido na documentação.

C. Matriz de Jogo

Durante o jogo é necessário manter um registo das jogadas de cada jogador no decorrer do tempo de jogo, avaliando as jogadas e testando a vitória de qualquer um dos jogadores. Para isso, o grupo decidiu criar uma matriz 3 x 3 x 3 interna onde é armazenado o estado do jogo. Assim sempre que se realiza uma jogada é atualizada nesta matriz a posição correspondente ao cenário do jogo em que foi feita uma jogada, verificando as colunas, linhas e diagonais em todas as jogadas até um jogador ganhar. Quando um jogo acaba ou quando é reiniciado a matriz volta também ela ao estado inicial.

V. Conclusão

O grupo de trabalho reconhece que com a realização deste projeto foi possível a utilização de conhecimentos e capacidades adquiridos ao longo das aulas promovendo por isso uma consolidação destes conteúdos.

O recurso a translações e rotações foi essencial para o desenvolvimento do trabalho bem como a utilização de texturas e *buffers* de cores. A interligação dos vários módulos também se revelou um desafio contornado muitas das vezes com recurso ao código fornecido das aulas.

Tendo em vista uma operacionalização deste projeto seria desejável uma melhoria do design da página e um possível sistema de *login* que permitisse a contabilização dos pontos para cada utilizador.

Consideramos que o produto final apresentado cumpre os requisitos enunciados e faz uso das ferramentas indicadas para a realização do mesmo. Acreditamos ter atingido um patamar de qualidade do qual nos orgulhamos.