The Knight's Tour Problem in WebGL

João Trindade nº mec 89140, Ricardo Carvalho nº mec 89147

***Resumo* - Este relatório tem por objetivo o desenvolvimento do problema matemático "O Percurso do Cavalo" em 3D, através da utilização da ferramenta WebGL. Começando em qualquer casa de um tabuleiro de xadrez, uma peça representativa do cavalo deverá atravessar todas as casas de um tabuleiro, sem repetir nenhuma, e respeitando sempre as regras de movimento do mesmo.**

Primeiramente é estabelecido o problema matemático bem como uma possível solução, seguindo-se das condições tidas em conta para o início do trabalho. É então apresentado o algoritmo recursivo com *backtracking* utilizado, bem como os modelos e visualização 3D desenvolvidos. Passando pelo foco de luz e funcionalidades adicionais implementadas, o artigo retrata finalmente o feedback visual dado, tanto dentro como fora do *canvas*.

***Abstract* - This report aims to develop the mathematical problem "The Knight's Tour" in 3D, through the use of the WebGL tool. Beginning in any square of a chessboard, a piece representing the knight must cross all the squares of the board, without repeating any of them and always respecting the rules of knight's movement.**

First, the mathematical problem is established, as well as a possible solution, followed by the conditions taken into account for the beginning of the work. The recursive algorithm using backtracking is then presented, as well as the models and 3D visualization developed. Going through a point light source and additional functionalities implemented, this article finally portrays the visual feedback given, both inside and outside the canvas.

I. INTRODUÇÃO

O problema matemático "O Percurso do Cavalo" [1], é um desafio no qual a peça do cavalo percorre todas as casas do tabuleiro de xadrez sem repetir nenhuma, sendo um caso particular do conhecido problema do caminho Hamiltoniano, da teoria dos grafos. Neste procura-se descobrir se um caminho hamiltoniano, isto é, um caminho no qual são percorridos todos os vértices uma vez, sem repetir nenhum, é possível num determinado grafo.

A origem deste problema aponta ao século 9 a.C, e ao longo dos anos, vários matemáticos apresentaram contribuições para a resolução deste problema, nomeadamente Leonhard Euler. No entanto, foi o aparecimento dos computadores que facilitou a resolução do problema, sendo este um exercício bastante interessante no âmbito da programação e descrição de algoritmos.

A. Tabuleiro de Xadrez

O tabuleiro de xadrez é um elemento necessário para a prática da modalidade, sendo este normalmente de forma quadrada com alternância de duas cores (uma escura e outra clara) entre as subdivisões, podendo ser representado por uma matriz de 8 (oito) linhas por 8 (oito) colunas, em que cada célula é denominada por casa. Contudo, existem tamanhos e formas diferentes de tabuleiros, sendo que o problema pode abordar também tabuleiros em que o número de linhas ou colunas é diferente de 8 (oito). Convencionalmente é estipulado que deve ser usada a cor clara para a casa situada no canto inferior direito de cada jogador.

Fig. 1 - Tabuleiro de xadrez ocidental, retirado da Wikipedia.

B. Peça do Cavalo

O cavalo é uma das seis peças do jogo de xadrez convencional, sendo possível distinguir esta peça das restantes, não só pelo seu formato único, mas também pelo seu movimento ímpar. O cavalo movimenta-se em "L", podendo ter no máximo 8 (oito) movimentos possíveis, dependendo da casa onde se encontra, sendo estes:

- uma casa para cima e duas casas para a direita;
- duas casas para cima e uma casa para a direita;
- uma casa para baixo e duas casas para a direita;
- duas casas para baixo e uma casa para a direita;
- uma casa para baixo e duas casas para a esquerda;
- duas casas para baixo e uma casa para a esquerda;
- uma casa para cima e duas casas para a esquerda;
- duas casas para cima e uma casa para a esquerda.

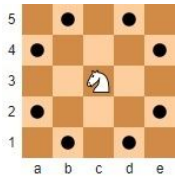

Fig. 2 - Movimentos possíveis do cavalo para um dada casa, retirado da Wikipedia.

C. Existência de solução para o problema

Allen J. Schwenk no seu paper *"Which Rectangular Chessboards Have a Knight's Tour?*" [2], provou que para qualquer tabuleiro m*n onde m<=n, um percurso do cavalo é sempre possível, excepto quando:

- m e n são ambos números ímpares;
- m é igual a 1 (um), 2 (dois) ou 4 (quatro);
- m é igual a 3 (três) e n é igual a 4 (quatro), 6 (seis) ou 8 (oito).

Como neste trabalho vamos apenas abordar o caso em que m e n são igual a 8 (oito), é portanto possível concluir que existe sempre, pelo menos, um caminho válido, podendo o cavalo dar início ao movimento em qualquer casa do tabuleiro.

II. REQUISITOS

No desenvolvimento do projeto foram considerados um conjunto de requisitos, de modo a ser produzida uma aplicação interativa que simula o funcionamento do algoritmo:

- Deve ser possível visualizar o tabuleiro de xadrez e a peça de cavalo, sendo esta visualização em 3D;
- O movimento do cavalo deve ser apresentado por uma animação;
- Os movimentos efectuados devem ser apresentados, de modo a indicar as casas já visitadas;
- Deve ser possível ao utilizador selecionar a posição inicial, através da interação com o rato;
- O utilizador deve conseguir rodar e aproximar/afastar o cenário;
- O utilizador deve conseguir resumir/pausar o movimento do cavalo;
- O utilizador deve conseguir escolher se quer observar o movimento progressivo/regressivo do cavalo;
- O cenário deve ser iluminado;
- O utilizador deve conseguir selecionar um movimento específico do cavalo;
- O utilizador deve conseguir alterar o modelo do cavalo;

III. ALGORITMO

A. Algoritmo recursivo com backtracking

A implementação do algoritmo recursivo para a resolução do problema encontra-se presente no módulo knightProblem.js. Através de uma função recursiva, o array bi-demensional que representa o tabuleiro, vai ser preenchido com números entre 1 (um) e 64 (sessenta e quatro), de modo a indicar a ordem das casas por onde o cavalo passa.

O algoritmo faz uso da técnica de backtracking para resolver o problema, uma vez que a solução é construída incrementalmente passo a passo, tendo em conta os próximos movimentos válidos de um cavalo (movimento que respeita as regras de movimento do xadrez e na qual a casa não foi visitada anteriormente). Sempre que não existe uma solução válida, esta é descartada, isto é, as casas por onde o cavalo passou são colocadas novamente a 0 (zero), e o processo é repetido para outros movimentos.

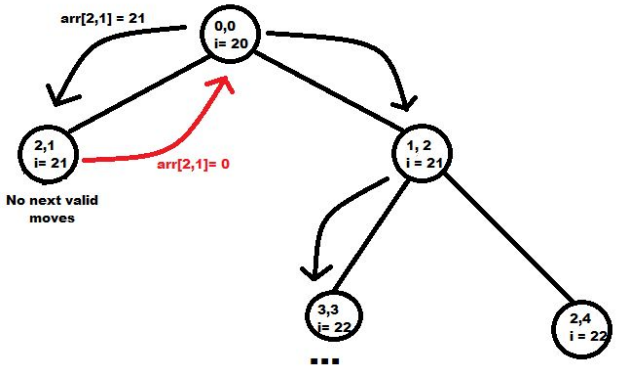

Fig. 3 - Exemplo do algoritmo recursivo com utilização de backtracking.

B. Regra de Warnsdorff

Apesar da implementação anterior ser válida, esta apresenta no pior dos casos uma complexidade temporal na ordem de O(8^(N*N)), o que torna a obtenção de um resultado válido um processo moroso e impraticável num cenário de aplicação Web.

Para tal, foi necessário a utilização de uma heurística na seleção do próximo movimento, em vez da sua seleção "cega". A heurística utilizada foi proposta por H. C. von Warnsdorff em 1823, e é conhecida como regra de Warnsdorff [3], que quando aplicada a este problema, permite a obtenção de uma solução, para muitos grafos, em tempo linear.

A implementação passou por alterar, em cada iteração do algoritmo recursivo, a ordem com que os próximos movimentos são expandidos. Nesta nova abordagem, são primeiros abordados os movimentos com menor grau, isto é, são primeiro abordados os movimentos que possuem um menor número de novos movimentos.

IV. MODELAÇÃO E VISUALIZAÇÃO 3D

A. Modelação do Tabuleiro de Xadrez

A definição dos vértices e cores do tabuleiro encontra-se na função *simpleBoardModel()* do módulo sceneModels.js. O tabuleiro pode ser visto como uma matriz de 10 (dez) linhas por 10 (colunas), pois embora o tabuleiro de xadrez tenha a dimensão de 8 (oito) linhas e 8 (colunas), o modelo também apresenta uma borda. O preenchimento dos *arrays* dos vértices e cores é feito através de um "*Nested For Loop*". Para a borda do tabuleiro é preciso definir vértices para as faces superior e inferior, assim como para as as faces laterais, sendo a cor destes definida como castanho. Para os vértices do tabuleiro em si, apenas é necessário vértices para as faces superior e inferior, sendo a cor alternada entre preto e branco, conforme a casa. No final, são calculadas as normais.

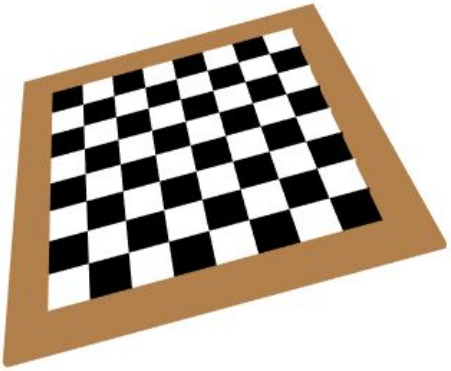

Fig. 4 - Modelo do tabuleiro de xadrez.

B. Modelação do Cavalo

Estabelecido na função simpleKnightModel() do módulo sceneModels.js, a peça cavalo do xadrez foi a utilizada por defeito na representação do percurso. Representado por 495 (quatrocentos e noventa e cinco) vértices, cada um com uma cor associada (uniforme na iteração final), esta peça representativa é então composta por 165 (cento e sessenta e cinco) triângulos.

Foi criada 1 (uma) de 2 (duas) faces principais primeiro, sendo a segunda uma cópia da primeira, transformada de maneira a ser possível visualizar a face correta dependente da localização do ponto de vista do utilizador. Para juntar estas 2 (duas) faces, foram utilizados triângulos com 2 (dois) vértices numa das faces e o terceiro na restante.

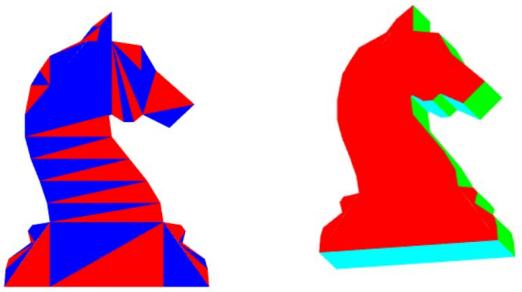

Fig. 5 - Duas iterações da construção da peça representativa por defeito.

C. Modelação das Polylines

Referido no capítulo mais à frente sobre feedback visual, foram modelados segmentos de reta, compostos por 6 (seis) vértices. Este modelo é iniciado através da simpleArrow(startPosition, endPosition, idx), cujos parâmetros definem a posição inicial e final do segmento, com a inicial ser a casa de onde a peça representativa inicia o seu movimento, e a final a casa onde termina o mesmo. O terceiro parâmetro é utilizado para calcular a cor resultante dos 2 (dois) triângulos que constituem o segmento de reta

Fig. 6 - Exemplo de uma das representações do segmento de reta.

D. Transformação sobre os modelos

No projeto são utilizados dois tipos de transformações: globais e locais.

São aplicadas transformações globais de rotação na escolha da posição inicial ou quando se roda o cenário com o rato. Por outro lado, é aplicada uma transformação global de ampliação/redução ao fazer zoom in/out, algo que irá ser abordado no capítulo das funcionalidades.

As transformações locais ocorrem apenas nos movimentos do cavalo, sendo aplicada uma translação da casa onde o cavalo se encontra para a casa de destino.

V. ILUMINAÇÃO

Com a finalidade de iluminar tanto o tabuleiro como a peça representativa, ambos modelos 3D, recorreu-se a um foco de luz pontual, situado na coordenada (0.0, 0.7, 1.5). O modelo utilizado para a implementação foi o modelo de reflexão criado por Bui Tuong Phong. Este consiste na soma de 3 (três) variáveis: iluminação ambiente, reflexão difusa e reflexão especular.

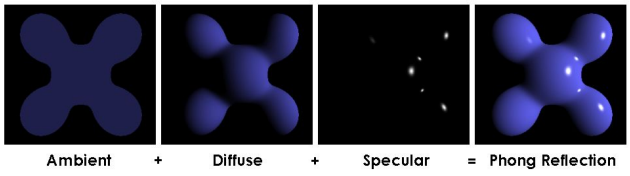

Fig. 7 - Funcionamento do modelo de iluminação de Phong, retirado da Wikipédia.

No entanto, a aplicação deste modelo conforme as linhas do guião [4] realizado nas aulas entra em conflito com a representação das cores originais de cada modelo. De forma a ultrapassar este obstáculo, tanto o foco de luz pontual como o fator de difusão foram iniciados com uma cor branca. A matriz de Phong resultante foi então multiplicada pelas cores originais de cada modelo, conciliando assim iluminação e cores pretendidas aos mesmos.

VI. FUNCIONALIDADES

Como forma de prevenir simulações com valores indesejados, antes do utilizador escolher a casa de partida para a resolução do problema, todas as seguintes funcionalidades estão ocultas da página, tornando-se visíveis assim que o botão de início se encontrar ativo.

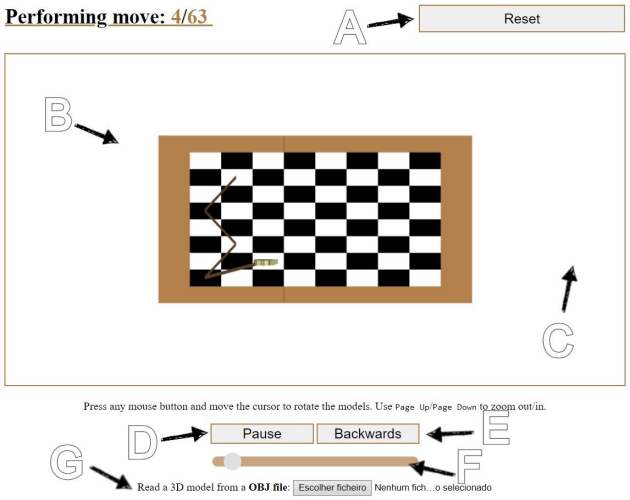

Fig. 8 - Vista por defeito da página após o início do percurso.

A. Reiniciar

O botão de "Reset" permite voltar ao estado inicial da página, restabelecendo os valores originais das rotações, escala, modelo da peça representativa, etc.

B. Rotação

Pode ser efetuada mantendo premido um dos botões do rato, ou outro dispositivo de entrada, dentro do canvas, enquanto arrasta o mesmo, executando assim uma rotação.

C. Zoom In/Out

Ao clicar na tecla "Page Up", os valores globais da escala serão reduzidos, provocando assim um efeito de zoom out. Já a tecla "Page Down" tem um efeito reverso, aumentando a escala e aproximando por isso os modelos do ponto de vista do utilizador.

D. Pausa

Tal como o seu rótulo indica, este botão permite ao utilizador pausar o percurso do cavalo, podendo assim ajustar a perspetiva sem perder nenhum movimento, por exemplo. Após o percurso ser então pausado, o valor do mesmo passa a ser contido por "Resume", sendo que um clique neste irá retomar os movimentos.

E. Orientação do Movimento

Com o valor por defeito a ser o de "Forwards", este botão permite alterar a direção pela qual o algoritmo se rege. Sendo que inicialmente executará os movimentos desde o 1 (um) até ao 63 (sessenta e três), caso o modo "Backwards" seja desencadeado, será apresentado desde o movimento atual, por exemplo 4 (quatro), até ao movimento número 1 (um).

F. Escolha do Movimento

Esta funcionalidade está dependente da indicada na alínea D, sendo por isso necessário que o utilizador pause a apresentação de modo a poder utilizar o controle deslizante. Posto isto, o número do movimento atual variará de forma a corresponder ao valor definido pelo *slider*, ao mesmo tempo que a peça representativa sofre uma translação imediata para a nova posição atual (definida pelo número do movimento).

De notar que esta funcionalidade apaga por completo todas as representações de *polylines* (descritas no próximo capítulo) apresentadas até ao momento.

G. Alteração da Peça Representativa

O último botão concede liberdade ao utilizador para modificar a peça representativa. Repare-se que o ficheiro de entrada deve restringir-se pela extensão .obj, de modo a alterar o modelo atual. Tal como referido anteriormente, a ação de reinício irá limpar o ficheiro de entrada, sendo por isso, novamente utilizada a peça do cavalo inicial.

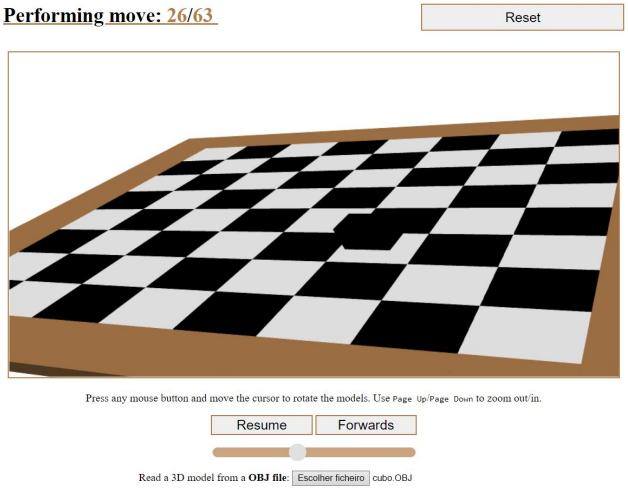

Fig. 9 - Vista alterada pelas funcionalidades permitidas na página.

VII. FEEDBACK VISUAL

A. Fora do Canvas

Após carregar a página desenvolvida, o utilizador pode, através do botão de início, perceber quando é que se encontra pronto para visualizar a resolução do problema relativo ao percurso do cavalo.

Fig. 10 - Antes e depois da escolha da casa de partida para o percurso do cavalo.

Já com o algoritmo em curso, é possível constatar o estado atual do mesmo. Uma variável indicará em que movimento se encontra o algoritmo, existindo um total de 63 (sessenta e três) movimentos.

Fig. 11 - Exposição do movimento atual do percurso do cavalo.

Também foi incluído na página web um controlo deslizante que representa, à escala, o progresso do algoritmo em tempo real

Fig. 12 - Exposição do progresso em tempo real para a conclusão do percurso do cavalo.

B. Dentro do Canvas

Sob a perspetiva do canvas, antes de proceder ao início do percurso e respetivo algoritmo, o utilizador deverá selecionar qual a casa que pretende definir como ponto de partida para a execução do conjunto de operações definidas. Não são permitidas quaisquer outras funcionalidades nesta fase, estando por isso o tabuleiro num ângulo e posição fixos. No momento de desencadeamento do algoritmo, a peça representativa sofre uma translação para o centro da casa escolhida, dando início à apresentação dos movimentos.

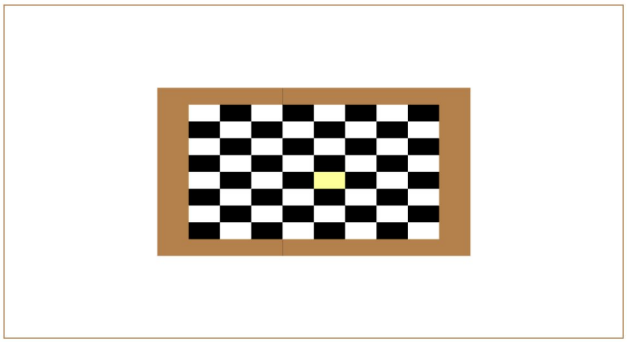

Fig. 13 - Escolha da casa de partida para o percurso do cavalo.

Acompanhando os movimentos efetuados pela peça representativa, são desenhadas representações de *polylines*. É acrescentado um novo segmento de reta sempre que o modelo chega a uma casa, com origem na posição atual da peça, e término na posição seguinte, para a qual se deslocará a mesma.

É assim também possível, visualmente, perceber se o modelo já passou por uma casa em específico. A cor dos segmentos de reta vai sofrendo alterações, simulando assim um gradiente no percurso final, com o intuito de tornar mais percetível a ordem do caminho tomado.

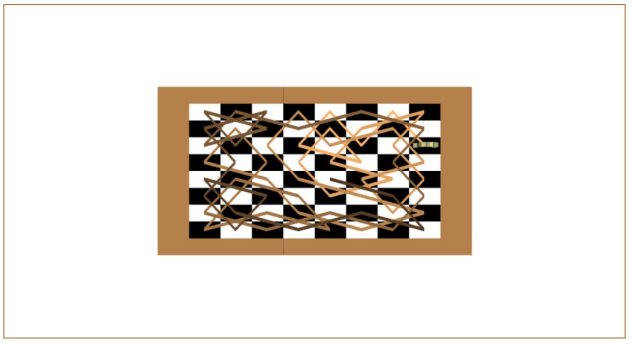

Fig. 14 - Exposição do progresso em tempo real para a conclusão do percurso do cavalo.

De forma a tornar mais subtil a deslocação entre casas do tabuleiro e a simular um movimento mais realista, foi realizada uma animação em torno dos 3 (três) eixos: x, y e z. Ao estar sob a posição de origem do movimento, a peça começa por subir momentaneamente, descendo gradualmente à medida que se desloca até à posição final do movimento.

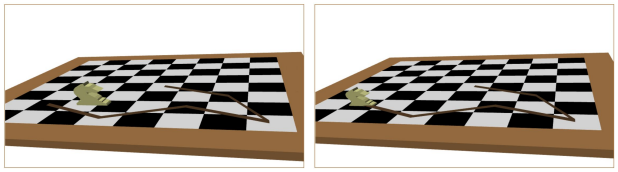

Fig. 15 - Modelo do cavalo a efetuar um salto entre duas casas, simulando assim o movimento.

VI. CONCLUSÃO

Apesar de ser um problema milenar, o percurso do cavalo é um problema matemático apelativo, conseguindo manter a sua essência e cativação em épocas distintas. Apesar de possíveis soluções serem já conhecidas há diversos anos, a sua implementação foi bastante desafiante.

Desde a modelação da peça representativa e do tabuleiro de xadrez, passando pela versatilidade do algoritmo para conceder que a posição inicial fosse introduzida pelo utilizador, a realização deste trabalho permitiu um melhor desenrolar acerca do funcionamento da perspetiva 3D e das suas dificuldades. O obstáculo da iluminação teve como efeito a consolidação do modelo de Phong e uma nova variante relativamente à cor.

Tanto as funcionalidades como o feedback visual, referidos nos capítulos relativos aos mesmos, introduziram nuances originais ao trabalho. Esta diversidade promoveu um maior uso da ferramenta WebGL, exibindo assim o poder e vastidão da mesma.

AGRADECIMENTOS

Os alunos agradecem ao professor Joaquim Madeira e ao professor Paulo Dias, pelo apoio prestado durante a realização do trabalho, através do esclarecimento de dúvidas e feedback dado sobre as funcionalidades da aplicação, de forma a melhorar a mesma.

REFERÊNCIAS

[1] “Knight’s tour”, Wikipedia, Wikipedia.org, https://en.wikipedia.org/wiki/Knight%27s_tour. Acedido a 16 de dezembro de 2020.

[2] Allen J. Schwenk, "Which Rectangular Chessboards Have a Knight's Tour?", Mathematics Magazine: 325–332, 1991.

[3] Squirrel, Douglas; Cull, P., "A Warnsdorff-Rule Algorithm for Knight's Tours on Square Boards", 1996.

[4] Joaquim Madeira, Material das aulas práticas no âmbito da Unidade Curricular de Computação Visual, http://sweet.ua.pt/jmadeira/WebGL/, 7 de novembro de 2019.