Universidade da Beira Interior
Departamento de Informática

Team RODA: *Tetris*

Elaborado por:

Nuno Pedro Nº41117

Guilherme Lacão Nº41787

Diogo Ventura Nº41685

Orientador:

Professor Doutor Abel Gomes

7 de janeiro de 2021

Agradecimentos

A conclusão deste trabalho, não seria possível sem a ajuda do Professor Abel Gomes, e toda a sua disponibilidade para esclarecer dúvidas e ser fundamental na resolução do mesmo.

Conteúdo

Capítulo

1

Introdução

1.1 Enquadramento

Este relatório foi feito no contexto da unidade curricular de Computação Gráfica, sendo efetuado como projeto final desta mesma cadeira e consiste na realização do jogo do Tetris.

1.2 Motivação

A motivação para este projeto foi elevada, pois existem várias versões do Tetris e todas podem ser estudadas e jogadas mas a nossa motivação foi fazer algo diferente e divertido de jogar.

1.3 Objetivos

O objetivo maior deste projeto foi realizar um jogo divertido e fácil de ser jogar, que qualquer jogador consiga interpretar.

1.4 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento.

2. O segundo capítulo – **Tecnologias Utilizadas** – descreve os conceitos mais importantes no âmbito deste projeto, bem como as tecnologias utilizadas durante do desenvolvimento da aplicação.

3. O terceiro capítulo – **Desenvolvimento** – descreve o alguns conceitos importantes do desenvolvimento do projeto e sobre a gestão do mesmo.

4. O quarto capítulo – **Implementação e Testes** – descreve o alguns pontos importantes sobre a implementação do projeto.

5. O quinto capítulo – **Conclusões e Trabalho futuro** – descreve as conclusões finais do trabalho e o trabalho futuro a ser desenvolvido

Capítulo

2

Tecnologias Utilizadas

2.1 Introdução

Este capítulo serve para descrever as tecnologias utilizadas no projeto e os motivos pelos quais foram utilizadas.

2.2 OpenGL

Neste trabalho utilizámos a biblioteca OpenGL (Open Graphics Library), sendo que esta é uma API que serve para renderizar gráficos 2D e 3D, e que serve para o desenvolvimento de aplicações gráficas. Esta também disponibiliza um conjunto de comandos que podem ser utilizados para desenhar polígonos , atribuir cores, aplicar texturas, entre outros.

2.3 GLFW

Também foi utilizada a biblioteca GLFW (Graphics Library Framework), sendo que esta permite a criação de janelas e a interação com o teclado e o rato. Assim, com esta API são lidos os inputs do teclado, para o consequente movimento das peças do Tetris, estando determinadas teclas associadas à rotação e ao movimento para a esquerda, direita e para baixo.

2.4 GLEW

A biblioteca GLEW (OpenGL Extension Wrangler Library), serve para a consultar e carregar outras extensões OpenGL. Assim, fornece mecanismos com

tempo de execução eficientes, para determinar quais extensões OpenGL são suportadas.

2.5 GLUT

A biblioteca GLUT (OpenGL Utility Toolkit), serve para a criação e o controlo de janelas, bem como o tratamento de eventos dos dispositivos de entrada, como o teclado e o rato. Assim, esta biblioteca foi usada para a criação do menu do jogo do Tetris.

2.6 Conclusões

As tecnologias utilizadas permitiram que o fosse desenvolvido o jogo do Tetris em 2D, permitindo a interação do utilizador, bem como o desenvolvimento do ambiente gráfico.

Capítulo

3

Desenvolvimento

3.1 Introdução

Este capítulo serve para explicar a ideia pela qual nos guiamos no desenvolvimento do projeto.

3.2 Pesquisa

Não somos pioneiros no desenvolvimento do jogo do Tetris, então o primeiro passo tomado foi o de pesquisar formas de resolver o problema. Após uma breve pesquisa, encontramos uma ideia baseada em matrizes que se adequou perfeitamente.

3.3 Matrizes

A ideia de usar matrizes para a implementação do jogo do Teris é bastante simples. Começa-se por criar uma matriz de duas dimensões com o tamanho (NxM) do tabuleiro do jogo. Cada elemento da matriz representará um "quadrado"do tabuleiro, se o elemento tiver valor zero, significa que o espaço do tabuleiro está vazio, qualquer valor maior que zero será parte de uma peça. A matriz é iniciada com todos os seus elementos a zero (tabuleiro vazio), e sempre que uma peça nova é criada, é representada na matriz pelo seu id na linha do "topo", movendo-se na vertical, no sentido de cima para baixo, até detetar uma colisão. Quando uma colisão é detetada a peça para de se mover e uma peça nova é criada. Este processo é repetido até que a sobreposição das peças

faça com que uma peça "pare"na linha onde é criada, ou caso o jogador atinja a pontuação 99.

3.4 Colisões

Como foi explicado antes, as peças podem mover-se até detetarem uma colisão com a linha abaixo onde esta se encontra, mas também podem mover-se na horizontal. Para que este processo ocorra sem problemas é preciso haver se há colisões caso a peça se mova.

3.4.1 Colisões verticais

As peças são criadas no "topo"do tabuleiro e deslocam-se para baixo a cada ciclo. Então é necessário apenas verificar se existe colisão com alguma peça na linha abaixo. Para isso, verificamos se algum elemento da matriz na linha abaixo de cada uma das "partes"da peça mais inferiores tem valor maior que zero, caso tenha, existe colisão e a peça para, caso contrario, não existe colisão. Esta verificação é feita antes de a peça se mover.

3.4.2 Colisões horizontais

Este processo é feito da mesmo forma que as colisões verticais, sendo que neste caso a verificação é feita para a esquerda e para a direita, e caso exista colisão a peça não para de se mover para baixo, apenas não efetua o movimento horizontal.

3.5 Movimento das Peças

O movimento das peças é feito na matriz na horizontal e vertical com recurso a algumas funções.

3.5.1 Movimento horizontal

O movimento horizontal é feito com recurso a duas funções:

- função moveRight(int tipo, int id, int rot);
- função moveLeft(int tipo, int id, int rot);

3.5.1.1 Função moveRight

A função é a função responsável para realizar movimentos da peça para a direita. Esta função recebe como parametros o tipo da peça, o seu id e a sua orientação inicial. Antes de realizar o movimento na matriz é verificado se a peça que se pode mover a direita, se o movimento não poder ser concretizado então a peça só se irá mover para baixo, mas se não for detetada nenhuma colisão à direita da peça, é efetuado o movimento. Este movimento é feito de forma unica para cada peça conforme a respetiva orientação.

3.5.1.2 Função moveLeft

A função moveLeft é a função responsável por realizar movimentos para a esquerda. Esta função recebe como parâmetros, o tipo da peça, o seu ID e a orientação inicial da peça. Antes de realizar o movimento na matriz é verificado se a peça se pode mover para a esquerda, se o movimento não puder ser concretizado então a peça só se irá mover para baixo, mas se não for detectada nenhuma colisão á esquerda da peça, então é efetuado o movimento para a esquerda. Este movimento é feito de forma única para cada peça conforme a respetiva orientação.

3.5.2 Movimento vertical

O movimento vertical é feito recorrendo á função:

- função moveDown(int tipo, int id, int rot);

3.5.2.1 Função moveDown

A função moveDown é a função responsável por realizar movimentos verticais no sentido de cima para baixo. Esta função recebe como parâmetros, o tipo da peça, o sei ID e a orientação inicial da peça. Antes de realizar o movimento, é verificado se existem colisões para baixo, se o movimento não puder ser concretizado então a peça irá parar e outra peça será criada. Este movimento é feito de forma única para cada peça conforme a respetiva orientação.

3.6 Rotação das peças

A rotação das peças é feita recorrendo á função:

- função RodarPeca(int tipo, int id, int rot);

3.6.1 Função RodarPeca

A função RodarPeca é uma função responsável por realizar a rotação das peças. Esta função recebe como parâmetros, o tipo de peça, o seu ID e a orientação inicial da peça. Antes de ser iniciada a rotação da peça na matriz é verificado se a peça pode efetivamente rodar para aquele local, caso não seja possível a peça vai ficar com o mesmo tipo de rotação.

3.7 Função Jogo

Esta função é responsável por realizar o jogo em si, ou seja esta função é chamada sempre que se quer criar uma peça nova, e o ciclo da peça é todo feito nesta função. Terminando a função quando a respetiva peça atinge o seu ponto mais baixo possível no tabuleiro. No ínicio da função é verificado se existem linhas completas no tabuleiro, caso existam então essas linhas são eliminadas e a pontuação é aumentada. De seguida é feito uma peça random, com a chamada á função makePeca(int tipo, int id, int rot), e vamos iniciar um ciclo que só termina quando a peça colide com outra ou com o tabuleiro, dentro deste ciclo em si, é possível ao utilizador mover a peça tanto na horizontal como na vertical.

3.7.1 Criar peça

A criação das peças é feita recorrendo à função:

- makePeca(int tipo, int id, int rot);

3.7.1.1 Função makePeca

A função makePeca é responsável por criar uma peça no inicio do tabuleiro, esta função cria a peça de acordo com os dados que lhe são passados, ou seja de acordo com o tipo da peça e a sua orientação, esta função vai mudar na matriz do jogo as posições respetivas, ficando assim essas posições associadas ao id da respetiva peça, o que depois vai permitir realizar todos os movimentos das peças.

3.8 Ambiente gráfico

Com recurso a função draw(), vai ser desenhado no ecrã o tabuleiro do jogo e alguns extras que envolvem o tabuleiro. Sendo que o desenho das peças no ecrã é efetuado por 3 funções, este desenho é efetuado individualmente para

cada peça, e cada cor de uma peça esta associado ao seu ID, sendo chamado umas das funções seguintes, de acordo com a cor da peça.

- drawBlock(int i, int j);
- drawBlockAzul(int i, int j);
- drawBlockVerde(int i, int j);

A função drawBlock responsável por desenhar no ecrã todos os quadrados de cor vermelha, enquanto que as outras duas funções são responsáveis pelas outras duas cores.

3.9 Conclusões

Como conclusão deste capitulo, queremos reforçar que com certeza haverá mais formas de implementar o jogo do Tetris, mas esta foi a que melhor se adequou aos nossos conhecimentos.

Capítulo

4

Implementação e Testes

4.1 Introdução

Este capitulo serve para descrever de uma forma sucinta a implementação e os testes realizados ao projeto e o seu contexto.

4.2 Tetris

Neste Projeto foi implementado uma versão simples do jogo do Tetris, onde é possível deslocar as peças na horizontal e alterar a orientação das mesmas. A pontuação máxima que um jogador pode atingir é de 99.

4.2.1 Tipos de Peças

No total existem 7 peças no jogo, sendo que 3 delas podem surgir em 4 orientações diferentes, outras 3 contêm duas orientações diferentes.E por ultimo existe uma peça só com uma orientação. Estas podem surgir no jogo em qualquer orientação e em ordem aleatória também.

4.2.2 Mover e Rodar as Peças

Partindo do momento que as peças surgem no ecrã e até ao momento em que estas são pousadas é possível mover cada uma destas. Usando as teclas 'a' e 'd' para mover para a esquerda e direita, respectivamente. Ou usando a tecla 's' para as peças descerem totalmente. Também é possível rodar as peças usando a tecla 'r', assim vamos mudar a orientação de cada peça.

4.2.3 Pontuação

A pontuação do jogo é simples, sempre que houver uma ou mais linhas completamente cheias, então essas são removidas e a pontuação é aumentada de acordo com o número de linhas removidas.

4.2.4 Dificuldade do jogo

É possível jogar o jogo em 3 dificuldades diferentes, sendo que nas duas primeiras dificuldades a diferença é apenas o tempo que as peças demoram do topo do tabuleiro até ao fundo do mesmo. Na dificuldade dificil, o tempo será menor e também deixa de ser possível visualizar a peça seguinte.

4.3 Visual Studio

Para realizar todos os testes do projeto, foi utilizado o programa Visual Studio 2019, com as opções gráficas que este mesmo programa permite. Foi testado em versões diferentes como Visual Studio 2019 e 2019.

4.4 Conclusões

Como conclusão deste capítulo é importante reforçar que todos testes e implementação realizada serve para proporcionar a melhor experiência ao utilizador e verificar que tudo está a funcionar nos melhores termos possíveis.

Capítulo

5

Conclusões e Trabalho Futuro

5.1 Conclusões Principais

Neste trabalho conseguimos criar um simples jogo utilizando várias tecnologias aprendidas ao longo da unidade e utilizando vários recursos diferentes, o que no fim deu um resultado final interssante e com uma experiência interessante para o utilizador.

5.2 Trabalho Futuro

Um dos pontos que deve ser mencionado aqui é sem dúvida a texturização das peças do jogo, apesar de existir modelação e as peças corresponderem ás peças reais, no futuro o trabalho a fazer é inserir no contexto do trabalho texturização nas peças do jogo, que permitam uma melhor experiência ao utilizador.

Capítulo

6

Bibliografia

- https://github.com/andykhv/Tetris3D
- https://gamedevelopment.tutsplus.com/tutorials/implementing-tetris-collision-detection–gamedev-852