Uma imagem com Tipo de letra, Gráficos, logótipo, texto

Descrição gerada automaticamente

**Trabalho Prático de Programação Avançada**

**Meta 2- P3**

Tiago Rafael Santos Cardoso

2021138999

[a2021138999@isec.pt](mailto:a2021138999@isec.pt)

Índice

Decisões tomadas na implementação - pág. 3

Diagrama da máquina de estados – pág. 4

Classes – pág. 5

Relacionamento entre classes – pág. 8

Implementado/Não Implementado – pág. 10

Decisões tomadas na implementação

Os elementos que compõem este jogo têm por base uma classe que implementa a interface IMazeElements (como pedido no enunciado) chamada Element, nesta classe são guardados as coordenadas e o símbolo do próprio elemento. Isto facilita em várias situações da implementação do jogo nomeadamente, no caso em que os elementos que se movem passam por cima um dos outros.

Foi criada uma classe MazeInfo que tem como objetivo, guardar um objeto Maze (fornecido pelos professores, onde estão todos os elementos que fazem parte do jogo) e facilitar o acesso e a alteração das informações do próprio jogo.

A cada nível é construído um novo MazeInfo, pois desta maneira é conseguida mais facilmente a inicialização do jogo.

Diagrama da máquina de estados

Uma imagem com texto, diagrama, file, Esquema

Descrição gerada automaticamente

O jogo inicia-se no estado INITIAL\_STATE, e mantem se nele até que um dos 4 metodos (Up(), Down(), Left(), Right()) sejam chamados. Ao serem chamados muda-se de estado para o GAME\_STARTED, onde ocorre maior parte do jogo.

No estado GAME\_STARTED, ou no estado VULNERABLE, é possível pausar o jogo chamando o método pause(), passando assim para o estado GAME\_PAUSED, onde é possível retornar para o estado anterior chamando o método resume().

Continuando no estado GAME\_STARTED é feita a transição para o estado VULNERABLE quando no método evolve se verifique que o pacman tem poderes (pamanHasPower()). Quando no método evolve, agora no estado VULNERABLE, se verifique que o pacman já não tem poderes (!pacmanHasPower()), passa se do estado VULNERABLE para o estado GAME\_STARTED.

Tanto no estado GAME\_STARTED como no VULNERABLE caso se verifique que o nível já acabou, tanto por não ter vidas ou por já não haver comida, passa-se do estado em que está para o estado INITIAL\_STATE.

Classes

Element:

Serve para representar todos os elementos que existem no jogo.Contém a sua posição e o seu símbolo.

MoveableElement:

Serve para representar todos os elementos que se movem. Já que todos os elementos se movem dando uma direção, esta classe implementa esse mesmo movimento, e a verificação do que está ao seu redor também é aqui feita, pois todos estes elementos não podem atravessar paredes.

Ghost:

Serve para representar todos os fantasmas e contém tudo o que é comum entre eles como o estado de vulnerabilidade e posição das portas da caverna. Esta classe é importante pois define todas as ações que são comuns entre todos os fantasmas nomeadamente o movimento que fazem quando estão vulneráveis.

PacMan:

Representa o pacman e guarda o valor de poder que possui, verdadeiro com poder, falso sem poder.

Blinky: Representa o fantasma Blinky, e aplica o seu movimento.

Clyde: Representa o fantasma Clyde, e aplica o seu movimento.

Inky: Representa o fantasma Inky.

Pinky: Representa o fantasma Pinky.

MazeInfo:

Contém dados importantes para o jogo. Esta classe facilita imenso a procura de certos elementos e também facilita a realização dos efeitos que no decorrer do jogo acontecem aos mesmos. Aqui também existe uma função que quando chamada chama o evolve dos MoveableElement.

Game:

Inicializa o MazeInfo através da leitura de um ficheiro de texto, realiza a gestão da execução do jogo movendo todos os elementos que são para mover. Aqui trata-se também da troca de nível, atualiza o número de vidas e a quantidade de pontos. Gere também a temporização do poder do pacman.

ScoreEntry: Guarda o nome e a correspondente pontuação de um elemento da lista top5.

Top5Players:

Guarda a lista e ordena pela pontuação dos 5 melhores jogadores. Adiciona ou atualiza a pontuação. Essa lista é constituída por elementos ScoreEntry.

ModelLog:

Implementa um log para armazenar mensagens em forma de string. Utiliza o padrão Singleton, garantindo que apenas exista uma instância.

EGameState: Enumeração com os estados possíveis no decorrer do jogo.

GameContext: Manipula os dados consoante o estado do jogo.

GameStateAdapter:

Implementa a interface IGameState e é a base para as classes que representam cada estado do jogo. Realiza todos os métodos do IGameState, colocando-os vazios e os que têm que retornar alguma coisa retorna false nos booleanos, null no retorno do tipo de estado e zero nos que tem que retornar algum tipo de valor numérico.

IGameState:

Interface do estados o jogo, aqui existem vários métodos a serem executados no correr do jogo que irão ser implementados pelos estados, nomeadamente: getGameState(), Up(), Down(), Left(), Right(), pause(), resume(), evolve(double time).

GamePaused:

Estado pausado do jogo. Contem um evolve vazio, ou seja, não acontece nada no jogo apenas fica tudo na mesma. É possível resumir o jogo voltando para o estado anterior.

GameStarted: Estado do jogo a decorrer. É possível, pausar o jogo, mudar de direção e fazer o evolve do jogo, avança uma iteração.

InitialState: Estado do jogo à espera que seja tomada uma direção. Não acontece nada, não é possível pausar nem resumir, e o evolve é vazio.

VulnerableState:

Estado vulnerável do jogo, quando o pacman tem poder. Este estado apenas é temporário, mas isso é tratado pelo Game. Aqui é possível fazer tudo o que se faz no estado GameStarted.

GameManager:

Gestor do jogo, aplicando o propertyChangeSuport para fins de sincronismo com a interface e o jogo em si. Aqui são guardados o top 5 e o contexto do jogo, toda a execução do jogo é feita com a chamada dos seus métodos que por sua vez chama métodos do game, a lista top 5 é carregada e atualizada aqui no GameManager a partir de um ficheiro .dat .

CSSManager: Aplica os estilos de um ficheiro .css num objeto tipo Parent.

ImageManager: Gere as imagens e guarda-as para não ter que carrega-las cada vez que seja preciso delas.

BeginUI: Ecrã inicial do jogo, com 3 botões, iniciar jogo, ver top 5 e sair.

GameOverUI: Ecrã de fim de jogo, apenas pede o nome do jogador e mostra a pontuação obtida.

GameUI:

Ecrã onde decorre o jogo, é mostrado as vidas, o nível, a pontuação, é possível pausar carregando no botão azul do canto superior direito, ou apenas pressionando na tecla espaço. Ao pausar o jogo mostra o PausePopup.

Top5UI:

Ecrã onde aparece a lista com o nome do jogador e a respetiva pontuação, é possível voltar ao menu principal carregando no botão abaixo da lista que diz “Voltar”.

ExitAlert:

Mostra um alerta para o utilizador escolher se quer mesmo sair ou não, fechando se escolher sim.

InsertNamePopup:

Popup que contém uma zona para o utilizador preencher com o seu nome e ao carregar no ok, guarda o seu nome e a sua pontuação na lista top5.

PausePopup:

Aqui aparece um pop up com dois botões um que diz resumir e outro que diz salvar e sair, ao resumir o jogo continua e fecha o popup, ao salvar e sair mostra o InsertNamePopup.

ListPane:

Mostra a transição de estados numa lista, ou qualquer outra informação que for colocada em formato de string.

MainJFX:

Cria as janelas RootPane e ListPane, passando para as janelas do tipo RootPane o GameManager. Aqui é criado o GameEngine e regista o evolve do GameManager. Ao ser fechado todas as janelas é fechado o gameEngine também.

RootPane:

Coloca o fundo a preto e cria uma pilha de ecrãs a mostrar passando para cada uma delas o GameManager recebido do MainJFX.

Relacionamento entre classes

Dados do jogo:

Uma imagem com diagrama

Descrição gerada automaticamente

Máquina de estados:

Uma imagem com texto, diagrama, file, Esquema

Descrição gerada automaticamente

Interface gráfica:

Uma imagem com texto, diagrama, captura de ecrã, file

Descrição gerada automaticamente

Implementado/Não Implementado

Em caso de faltarem pontos a colocar na tabela tenho a certeza que todos os pontos não implementados ou não implementados na totalidade, foram todos colocados na tabela, ou seja, o resto dos pontos que faltam foi tudo implementado.

|  |  |  |
| --- | --- | --- |
| ID | Descrição | Implementado/Não implementado |
| 1 | O jogo deverá possuir um ecrã inicial no qual deverão ser disponibilizadas ao utilizador as opções para “Iniciar Jogo”, “Consultar Top 5” e “Sair”. | Implementado |
| 2 | Deve surgir obrigatoriamente uma menção ao DEIS-ISEC-IPC (na segunda meta deverá ser incluído o logotipo do ISEC), ao curso (LEI, LEI-PL ou LEI-CE), à unidade curricular, ao ano letivo e ao nome e número do aluno, bem como a indicação de que é um trabalho académico. | Implementado |
| 3 | Caso o utilizador escolha a opção sair, deverá surgir no ecrã uma janela de diálogo para confirmar se pretende mesmo sair. | Implementado |
| 4 | O pac-man deverá ser controlado através de teclas de mudança de direção. O movimento será automático ao longo da direção correspondente à tecla, não sendo necessário manter a tecla premida, e continuará nessa direção até encontrar uma parede, onde deverá parar. A pressão repetida da tecla de direção correspondente à direção seguida pelo pac-man não deverá fazer com que este ande mais depressa. | Implementado |
| 5 | Quando passa por uma bola comestível normal, bola comestível com poderes ou fruta, a mesma deverá desaparecer e o jogador acumulará os pontos. Para cada bola com poderes recolhida, esta sequência de atribuição de pontos volta a ser reiniciada em 50 pontos. | Implementado |
| 6 | Os fantasmas podem ser comidos quando estão vulneráveis (após o pac-man comer a bola com poderes), recebendo o jogador 50 pontos pelo primeiro, 100 pontos pelo segundo, 150 pelo terceiro e 200 pelo quarto | Implementado |
| 7 | O nível termina quando todas as bolas do labirinto são comidas (incluindo as bolas com poderes), passando o jogo para o nível e labirinto seguinte. | Implementado |
| 8 | Caso o pac-man seja apanhado por um fantasma (encontram-se na mesma posição ao mesmo tempo) o jogador perde uma vida, reiniciando-se o nível | Implementado |
| 9 | O jogo termina ao chegar ao vigésimo nível ou quando o jogador pac-man perde todas as, sendo a pontuação incluída no Top 5. | Implementado |
| 10 | O Top 5 não deverá ser perdido quando o jogo é encerrado, devendo ser guardado em ficheiro através do processo de “serialização” para permitir a sua reposição numa execução posterior. | Implementado |
| 11 | Na situação inicial, o labirinto e respetivo conteúdo são apresentados no ecrã, não existindo qualquer ação nem movimento | Implementado |
| 12 | Ao ser pressionada uma tecla de direção, a ação tem início: o pac-man movimenta-se segundo a direção escolhida. | Implementado |
| 13 | Após 5 segundos (o aluno pode definir outro valor) os fantasmas entram no labirinto e começam a deslocar-se. | Implementado |
| 14 | O jogo desenrola-se a partir daí até o pac-man ser apanhado por um fantasma ou uma bola com poderes ser comida. Neste caso, o jogo entrará num novo estado, de tempo limitado (definido pelo aluno), correspondente à fase em que os fantasmas ficam vulneráveis. Este estado termina quando se esgota o tempo especificado ou quando todos os fantasmas forem comidos e reverterem ao comportamento normal. | Implementado |
| 15 | Durante o jogo o mesmo poderá ser colocado em pausa. Quando estiver em pausa deverá ser possível sair do jogo ou salvaguardar o jogo para o poder restaurar numa execução futura da aplicação. | Implementado |
| 16 | Quando for escolhida a opção do menu principal para iniciar um jogo, deve ser detetado se existe um jogo gravado e deve perguntar se pretende continuar o jogo anterior. Caso o jogador responda que não, essa possibilidade não dever ser facultada novamente para esse jogo guardado. | Implementado |
| 17 | A máquina de estados apenas se aplica ao momento de jogo, não devendo ser usada para gerir as opções no ecrã inicial da aplicação ou, por exemplo, a consulta do Top 5. | Implementado |
| 18 | A atualização desta interface, bem como outras atualizações de informação, deverá estar de acordo com o padrão de notificações assíncronas estudado nas aulas | Implementado |
| 19 | Incorporação de testes unitários nas classes relativas à máquina de estados | Não implementado |
| 20 | Para cada nível, a velocidade dos fantasmas deve ir aumentando e o tempo de vulnerabilidade dos fantasmas deverá ir diminuindo (com valores à escolha do aluno). | Não implementado |
| 21 | Movimento dos fantasmas | Implementado parcialmente apenas foi implementado como pretendido o movimento dos fantasmas Blinky e Clyde, os outros fantasmas como não consegui implementar o movimento apenas movimentam-se como o Blinky. |