Trabalho Prático
Programação Avançada
2022/2023

Tiny-PAC

Diogo Fernandes Correia - 2018019013

1.Introdução

Este trabalho prático consiste na criação e implementação de uma versão do famoso jogo "Pac-Man",

O trabalho possui uma componente de interface gráfica realizada em Javafx

2.Decisões Tomadas

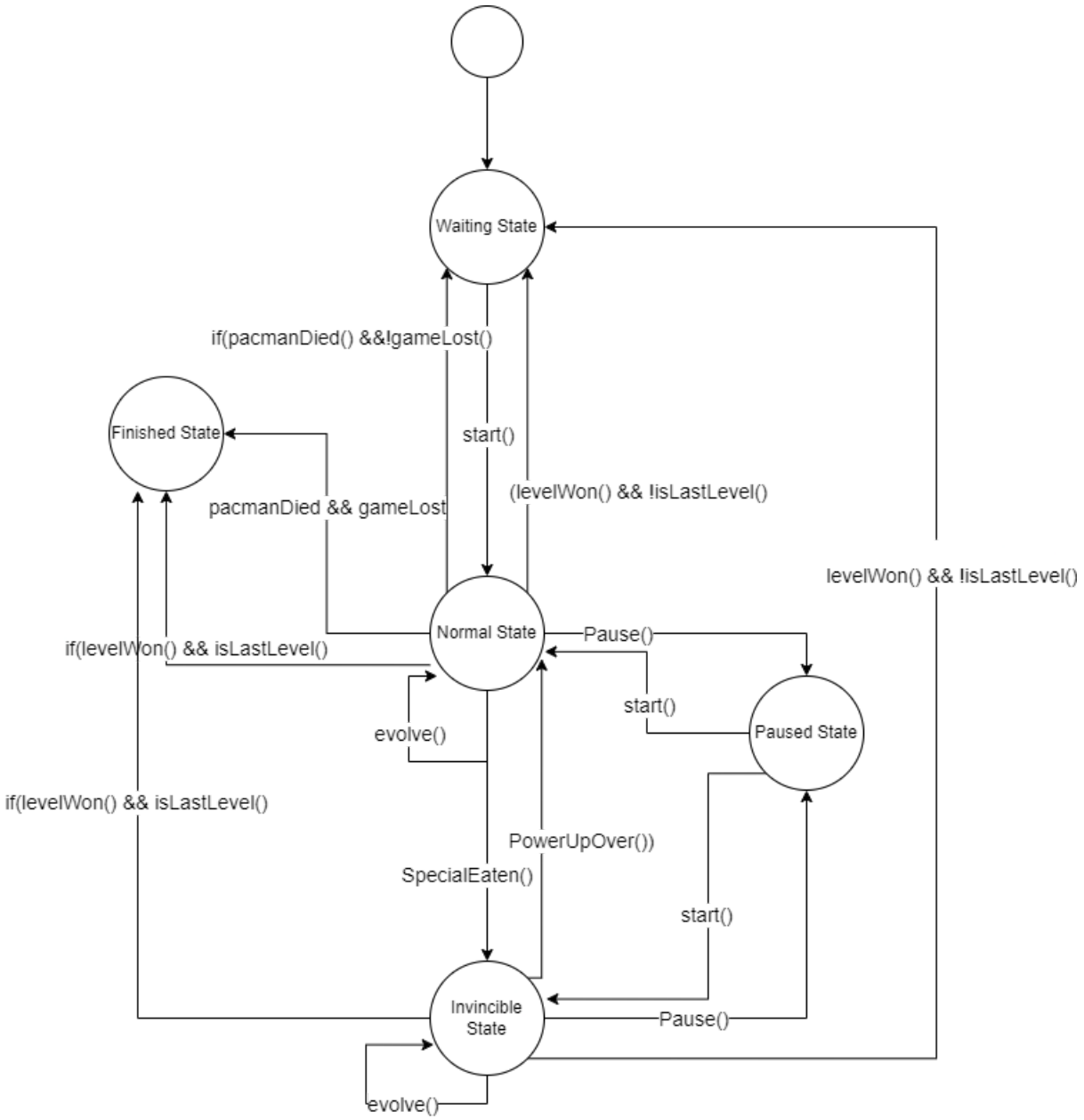
Devido à natureza do enunciado foram tomadas algumas decisões sobre a implementação de certos pontos do trabalho.

Save e Load, optei por adicionar um botão ao menu principal que ira carregar um jogo a partir de um ficheiro .bin, este botão só é mostrado se a existir esse ficheiro para guardar.

Isto não é uma decisão tomada mas sim uma necessidade devido ao tempo, mas nao implementei um movimento diferente para cada fantasma, assim eles partilham todos o mesmo tipo de movimento, o movimento do fantasma Blinky.

Optei pelo uso de variáveis para gerir a velocidade do pacman e dos fantasma, garantido que só fazem o movimento a cada certo número que a função evolve() é chamada, no caso dos fantasma cada vez que existe subida de nível o número de evolve() necessário para mover diminui

3.Diagrama da Máquina de Estados



Acima encontra um Diagrama da máquina de estados, este diagrama sofreu algumas reestruturações ao longo da execução do trabalho.

Assim que o jogo inicia encontra-se num estado WaitingState, neste estado o jogo nao sofre qualquer alteraçao, e espera pela transição start, para iniciar sempre para um NormalState,

O NormalState tem várias possiveis transiçoes, specialEaten(), leva a um estado InvincibleState, Pause() inicia um estado PausedState. As

outras transiçoes são um pouco mais complexas, se levelWon() verificar mas lastLevel() nao se verificar entramos num novo WaitingState, se levelWon e lastLevel() se verificarem entramos num estado FinishedState. Da mesma forma se pacmanDied() se verificar mas gameLost() nao se verificar entramos num WaitingState, se ambos se verificarem entramos num FinishedState, por fim a transição evolve leva ao próprio estado NormalState.

InvincibleState, tem menos transições, powerUpOver leva a um NormalState, se levelWon() e lastLevel se verificarem entao entramos num FinishedState, se levelWon() se verificar mas lastLevel() nao entramos num WaitingState, por fim Pause() entra num novo PausedState.

PausedState possui apenas uma transição start(), dependendo do estado anterior será esse o novo estado após a transição.

4.Classes Utilizadas

Dentro do package exception encontramos 3 exceções criadas por mim,

CantSpawnFruitException,

InvalidFileFormatException

InvalidMazeElementException

Foram criadas com o objetivo de serem lançadas para mais facilmente reconhecer o problema que está a acontecer,

Dentro do package MazeCharacter encontram-se as Classes relacionadas com os protagonistas do jogo, aqui podemos encontrar

MazeCharacters, a classe mãe que implementa a interface **IMazeElements**, as classes a seguir estendem todas esta classe, implementa alguns métodos relacionados com o movimento dos fantasmas e do pacman, que vão levar override por essas mesmas classes,

Ghost é uma subclass de **MazeCharacters,** esta classe vai implementar os métodos relativos ao movimento dos fantasmas assim como alguns getters e setters necessários,

Blinky,Clyde,Pinky,Inky, estas classes inicialmente eram suposto implementar os métodos relativos ao movimento de cada tipo de fantasma, no entanto devido a minha de decisão de implementação todos os ghosts têm um tipo de movimento como tal estas classes não têm nada para além de um char Symbol e um int Index;

Pacman esta classe implementa os movimentos relativos ao pacman

De seguida temos várias classes dentro do package **maze_objects**, **Ball, Empty, Fruit, Fruitspot, GhostZone, Portal, Special, Wall, Warp**, estas classes representam os objetos do qual o maze é construído, possuem apenas um symbol e a função get apropriada,

Direction um enumerador das 4 possiveis direções de movimento,

Position serve para guardar posições, possui também um método para retornar uma posição nova dependendo da direção recebida.

Environment classe responsável por gerir os dados do Jogo,

Environment Manager classe responsável por gerir o Environment

As classes no package states, encontram-se as classes **FinishedState, InvincibleState, NormalState, PausedState, WaitingState**, responsáveis por gerir os estados da máquina de estados.

Context classe responsável por redirecionar para cada um dos estados,

IState interface das transições de estado, implementada pela classe **StatesAdapter**

State enumerador dos estados da maquina de estados,

StatesAdapter classe responsável por implementar as transições de estado da interface **IState**,

GameManager classe que interage com a interface gráfica, e com a classe **Contex,** delegando métodos para ela,

GameEndPane classe responsável por tratar do painel de final de jogo

MazePane classe responsável por tratar do painel onde o tabuleiro de jogo é apresentado,

MenuPane classe responsável por tratar do painel do menu principal,

PaneOrganizer classe responsável pelo StackPane que engloba toda a aplicação,

PausePane classe responsável por tratar do menu de pausa

StatsPane classe responsável por tratar da Vbox que mostra as estatísticas do jogo atual

Top5Pane classe responsável por tratar do painel que mostra o TOP 5 jogadores

ImageManager classe responsável por gerir as imagens usadas no trabalho

5.Relações entre Classes

A classe MainJfx procede à criação de um StackPane PaneOrganizer, que engloba a UI este pane organizer depois cria os panes que vai ser apresentados de acordo com estado da aplicaçao,o MainJfx tambem cria um objeto da classe GameManager, a interface gráfica irá interagir com este GameManager,

Este GameManager ira encaminhar para a Classe Context que por si ira encaminhar os metodos para os vários estados da maquina de estados, esses estados vao interagir com o Environment Manager, responsável

por interagir com o Environment, e por fim este interage com as classes dos fantasmas, o pacman e o Maze.

6.Implementações

Interface Gráfica em Javafx	Implementado
Implementação Lógica do Jogo	Parcialmente Implementado
Save/Load	Implementado
Top5	Implementado
Javadocs	Implementado
Testes Unitários	Nao implementado