



Relatório Meta 2

Programação Avançada

Instituto Superior de Engenharia Informática
Turma Prática 3

João Carvalho

2019131769

1 de maio de 2023

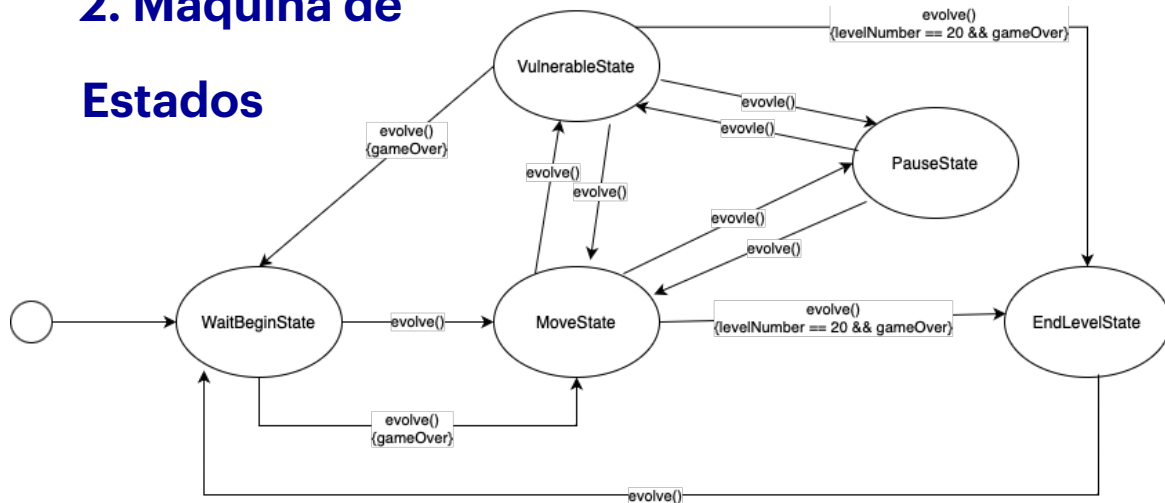
1. Descrição

O objetivo deste trabalho foi a implementação de um jogo tipo PacMan.

Foi implementada uma máquina de estados para gerir os estados do jogo. O jogo também possui elementos divididos em packages, sendo estes correspondentes aos intervenientes no jogo como o PacMan e os Fantasmas, as Frutas, a Comida entre outros.

Nesta meta foi implementada uma interface em JavaFX para o utilizador poder usufruir da aplicação e interagir com a mesma.

2. Máquina de Estados



Esta máquina de estados é constituída por 5 estados:

- WaitBeginState: estado em que o jogo está a espera que seja pressionada uma tecla para começar.
- MoveState: estado em que o jogo decorre e onde os “Mobs” se movimentam.
- VulnerableState: estado do jogo iniciado após o PacMan ingerir uma bola com poderes, tornando os Fantasmas vulneráveis e fazendo com que o PacMan os possa comer.
- PauseState: estado do jogo que é apresentado quando o jogador carrega na tecla Escape.
- EndLevelState: estado de jogo que representa o final do jogo após o jogador ter perdido ou ter ganho.

3. Classes

```
Main.java
1 package pt.isec.pa.tinypac;
2
3 import pt.isec.pa.tinypac.model.GameContextManager;
4 import pt.isec.pa.tinypac.ui.gvi.MainJFX;
5
6 import javafx.application.Application;
7
8 2 usages
9 public class Main {
10     public static GameContextManager gameCManager;
11     static{
12         gameCManager = new GameContextManager();
13     }
14     no usages
15     public static void main(String[] args) { Application.launch(MainJFX.class,args); }
16 }
```

Main:

- Classe onde são inicializados o levelManager, o context, a GameEngine e as UI's.

MainJFX:

- Classe onde é iniciado o stage e o RootPane.

```
13 2 usages
14 public class WaitBeginUI extends BorderPane {
15     5 usages
16     GameContextManager gameCManager;
17     4 usages
18     Text waitText;
19     1 usage
20     public WaitBeginUI(GameContextManager gameCManager) {
21         this.gameCManager = gameCManager;
22
23         createViews();
24         registerHandlers();
25         update();
26
27         this.requestFocus();
28     }
29
30     1 usage
31     private void createViews() {
32         this.setStyle("-fx-background-color: black;");
33
34         waitText = new Text("@ Pressione uma tecla para começar");
35         waitText.setFont(Font.font("Verdana", FontWeight.BOLD, 24));
36         waitText.setFill(Color.WHITE);
37
38         VBox vbox = new VBox(waitText);
39         vbox.setAlignment(Pos.CENTER);
40         this.setCenter(vbox);
41     }
42
43     1 usage
44     private void registerHandlers() {
45         gameCManager.addPropertyChangeListener(evt -> { update(); });
46         setOnKeyPressed(event -> {
47             gameCManager.evolve();
48         });
49     }
50 }
```

```
MainJFX.java
12 2 usages
13 public class MainJFX extends Application {
14     3 usages
15     GameContextManager gameCManager;
16
17     @Override
18     public void init() throws Exception {
19         super.init();
20
21         gameCManager = Main.gameCManager;
22     }
23
24     @Override
25     public void start(Stage stage) throws Exception {
26         firstStage(stage, "PacMan");
27
28         GameEngine gameEngine = new GameEngine();
29         gameEngine.registerClient((g, t) -> {
30             Platform.runLater(() -> {
31                 gameCManager.evolve(g, t);
32             });
33         });
34         gameEngine.start(Interval(350));
35     }
36
37     1 usage
38     private void firstStage(Stage stage, String title) {
39         RootPane root = new RootPane(gameCManager);
40         Scene scene = new Scene(root, 1920, 1280);
41         stage.setScene(scene);
42         stage.setTitle(title);
43         stage.setWidth(700);
44         stage.setHeight(400);
45         stage.show();
46     }
47 }
```

WaitBeginUI:

- Classe do apresenta no ecrã o que corresponde ao waitBeginState.

```

1 usage
public class VulnerableUI extends BorderPane {
    2 usage
    GameContextManager gameManager;
    3 usage
    private ScrollPane scrollPane;
    4 usage
    private VBox vbox;
    5 usage
    public VulnerableUI(GameContextManager gameManager) {
        this.gameManager = gameManager;
        createViews();
        registerHandlers();
        update();
        this.requestFocus();
    }
    6 usage
    private void createViews() {
        //this.setStyle("-fx-background-color: black;");
        scrollPane = new ScrollPane();
        scrollPane.setStyle("-fx-background-color: black;");
        this.setCenter(scrollPane);
        vbox = new VBox();
        vbox.setPadding(new Insets(10));
        vbox.setSpacing(10);
        vbox.setStyle("-fx-background-color: black; -fx-border-color: orange; -fx-border-width: 4px; -fx-text-fill: white; -fx-font-size: 48; -fx-font-weight: bold; -fx-font-family: monospace;");
        vbox.setPrefWidth(400);
        vbox.setPrefHeight(400);
        vbox.setAlignment(Pos.CENTER);
        Label livesLabel = new Label("@ Lives: ");
        livesLabel.setStyle("-fx-text-fill: white;");
        Label scoreLabel = new Label("@ Score: ");
    }
}

```

VulnerableUI:

- Classe que apresenta no ecrã o que corresponde ao VulnerableState

TOP5UI:

- Classe que apresenta no ecrã o menu do Top5.

```

1 usage
public class TopFiveUI extends BorderPane {
    2 usage
    GameContextManager gameManager;
    3 usage
    Text text;
    4 usage
    Text player1, player2, player3, player4, player5;
    5 usage
    Button btnDoBack;
    6 usage
    public TopFiveUI(GameContextManager gameManager) {
        this.gameManager = gameManager;
        createViews();
        registerHandlers();
        update();
    }
    7 usage
    private void createViews() {
        //Font pixelFont = Font.loadFont(getClass().getResourceAsStream("images/pixel_emulator/Pixel_EmuLator.ttf"), 16);
        this.setStyle("-fx-background-color: black;");
        text = new Text("@ TOP 5");
        text.setFont(Font.font("@ verdana", FontWeight.BOLD, 48));
        text.setFill(Color.WHITE);
        //player1 = new Text("1. " + gameManager.getTopFive().getTop5().get(0));
        player1 = new Text();
        player1.setFont(Font.font("@ verdana", FontWeight.BOLD, 24));
        player1.setFill(Color.WHITE);
        player2 = new Text();
        player2.setFont(Font.font("@ verdana", FontWeight.BOLD, 24));
        player2.setFill(Color.WHITE);
        player3 = new Text();
        player3.setFont(Font.font("@ verdana", FontWeight.BOLD, 24));
        player3.setFill(Color.WHITE);
    }
}

```

```

1 usage
public class ShowConfirmationUI extends BorderPane {
    2 usage
    GameContextManager gameManager;
    3 usage
    Text text;
    4 usage
    Button btnYes, btnNo;
    5 usage
    public ShowConfirmationUI(GameContextManager gameManager) {
        this.gameManager = gameManager;
        createViews();
        registerHandlers();
        update();
    }
    6 usage
    private void createViews() {
        //Font pixelFont = Font.loadFont(getClass().getResourceAsStream("images/pixel_emulator/Pixel_EmuLator.ttf"), 16);
        this.setStyle("-fx-background-color: black;");
        text = new Text("@ Deseja realmente sair?");
        text.setFont(Font.font("@ verdana", FontWeight.BOLD, 24));
        text.setFill(Color.WHITE);
        btnYes = new Button("@ Yes");
        btnYes.setStyle("-fx-background-color: black; -fx-border-color: green; -fx-border-width: 4px; -fx-text-fill: white; -fx-font-size: 48; -fx-font-weight: bold;");
        btnYes.setPrefWidth(200);
        btnYes.setPrefHeight(80);
        btnNo = new Button("@ No");
        btnNo.setStyle("-fx-background-color: black; -fx-border-color: red; -fx-border-width: 4px; -fx-text-fill: white; -fx-font-size: 48; -fx-font-weight: bold;");
        btnNo.setPrefWidth(200);
        btnNo.setPrefHeight(80);
        VBox vbox = new VBox(text, btnYes, btnNo);
        vbox.setAlignment(Pos.CENTER);
    }
}

```

ShowConfirmationUI:

- Classe que mostra uma mensagem de confirmação se quer sair.

```

public class GameContext {
    3 usages
    Game game;

    6 usages
    IMobsState gameState;

    1 usage 1 sudo-john-blossom
    public GameContext(){
        game = new Game("Level101.txt"/ 1);
        this.gameState = new WaitBeginState(context, this, game);
    }

    1 usage 1 sudo-john-blossom
    public IMobsState getState(){return gameState.getState();} //foi dado override no MenuState para ele poder ir buscar o state

    1 usage 1 sudo-john-blossom
    void changeState(IMobsState newState){this.gameState = newState;}

    1 sudo-john-blossom
    public boolean move(){return gameState.move();}

    1 usage 1 sudo-john-blossom
    public boolean vulnerable(){return gameState.vulnerable();}

    3 usages 1 sudo-john-blossom
    public boolean endLevel(){return gameState.endLevel();}

    //getData
    no usages 1 sudo-john-blossom
    public Game getGame(){return game;}
}

```

GameContext:

- Classe responsável por cuidar do jogo e dos estados do jogo.

GameContextManager:

- Classe que serve como proxy entre o gameContext e o resto do programa.

```

public class GameContextManager implements IGameEngineEvolve {
    2 usages
    private static final String SAVE_FILE = "files/save.dat";
    2 usages
    private static final String TOP5_FILE = "files/topFive.dat";
    23 usages
    private GameContext fsm;
    12 usages
    PropertyChangeSupport pcs;

    public GameContextManager(){pcs = new PropertyChangeSupport(sourceBean, this);}

    public GameContext getFsm(){return fsm;}

    1 usage
    public void setFsmNull(){fsm = null;}

    public void addPropertyChangeListener(PropertyChangeListener listener){pcs.addPropertyChangeListener(listener);}

    public void start(){
        fsm = new GameContext();
        pcs.firePropertyChange(propertyName: null, oldValue: null, newValue: null);
    }

    public IMobsState getState(){return fsm.getState();}

    public boolean evolve(){
        //if(fsm == null) return false;

        var ret = fsm.evolve();
        pcs.firePropertyChange(propertyName: null, oldValue: null, newValue: null);
        return ret;
    } //evolve de mudanca de estado

    public boolean pause(){
        var ret = fsm.pause();
        pcs.firePropertyChange(propertyName: null, oldValue: null, newValue: null);
    }
}

```

```

public class EndLevelState extends MobsStateAdapter {
    1 usage 1 sudo-john-blossom
    public EndLevelState(GameContext context, Game game){
        super(context, game);
    }

    //SETTERS

    1 usage 1 sudo-john-blossom
    @Override
    public boolean endLevel(){
        //aqui vai mudar o level
        //game.setLevel(new Level(game.getLevel().getLevelNumber() + 1)); // demonstracao que vai para o nivel seguinte
        changeState(IMobsState.WAIT_BEGIN);
        return true;
    }

    1 usage 1 sudo-john-blossom
    @Override
    public IMobsState getState(){return IMobsState.END_LEVEL;}
}

```

States:

- Classes responsáveis pela lógica do jogo naquele estado.
- Também são responsáveis por encaminhar o jogo para outros estados.

```

22 usages 14 inheritors sudo-john-blossom
public abstract class Element implements IMazeElement {

    protected Level level;

    6 usages sudo-john-blossom
    protected Element(Level level) {
        this.level = level;
    } // o facto de ter aqui o construtor eu depois n tenho que repetir nas classes derivadas

    //abstract public void evolve();

    1 override sudo-john-blossom
    public boolean move(){return false;}
    no usages 1 override sudo-john-blossom
    public boolean eat(){return false;}
}

```

Element:

- Classe abstrata que funciona como classe base para todos os elementos do maze.
- Contém implementações por omissão das funções que os elementos vão implementar.

Mobs:

- Classes que definem os mobs e o símbolo que os representa.
- Contém as funções de movimento, etc.

```

public class TinyPac extends Element {
    public static final char SYMBOL = 'M';

    2 usages sudo-john-blossom
    public TinyPac(Level level) { super(level); }

    sudo-john-blossom
    @Override
    public char getSymbol() { return SYMBOL; }

    sudo-john-blossom
    @Override
    public boolean move(/*KeyEvent e*/){
        Level.Position myPos = level.getPositionOf( element: this);
        level.addElement(new EmptyCell(level), myPos.y(), myPos.x());
        level.addElement(new TinyPac(level), myPos.y(), (x: myPos.x() + 1);
    }
}

```

```

public class Cell extends Element {

    1 usage
    private static final char SYMBOL = 'x';

    8 usages sudo-john-blossom
    public Cell(Level level) { super(level); // depois na herança cada célula vai ter um type diferente }

    8 overrides sudo-john-blossom
    @Override
    public char getSymbol() { return SYMBOL; }

    /*@Override
    public void evolve(){
    }*/
}

```

Cells:

- Classes que seguem exatamente o mesmo princípio das classes dos Mobs

JavaFX

