Class Name	Lecturer	Date
Object-Oriented Programming	Mr.Nguyen Trung Nghia	25/12/2024

Game project
OF THE COW BO

Table of Contents

01 Introduction

02 Software requirements

03 Design and Implementation

01

Introduction

Developer team

01 Nguyen Huu Hoang Nam

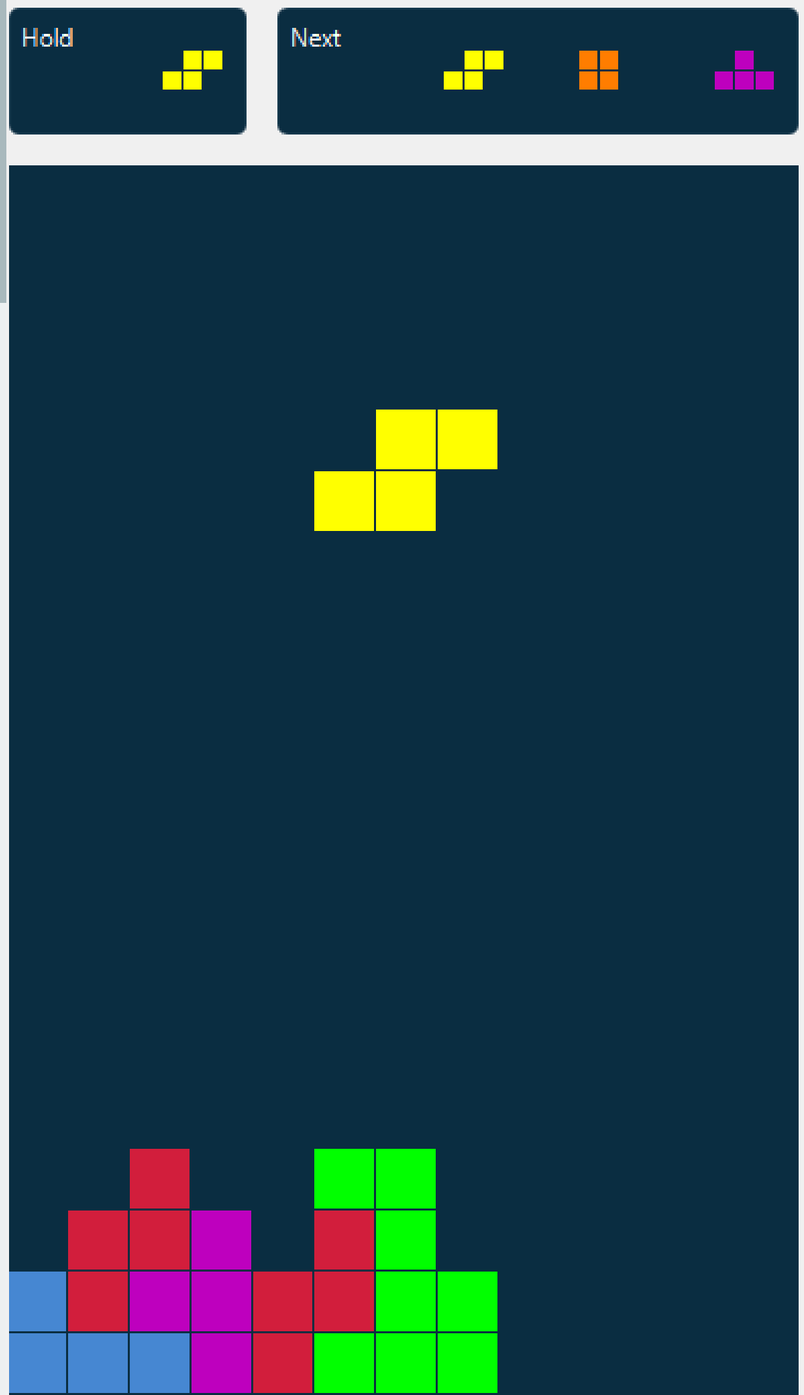
02 Cao Ngoc Anh Tuan

03 Le Pham Thuy An

04 Dao Huu Hoai

Introduction

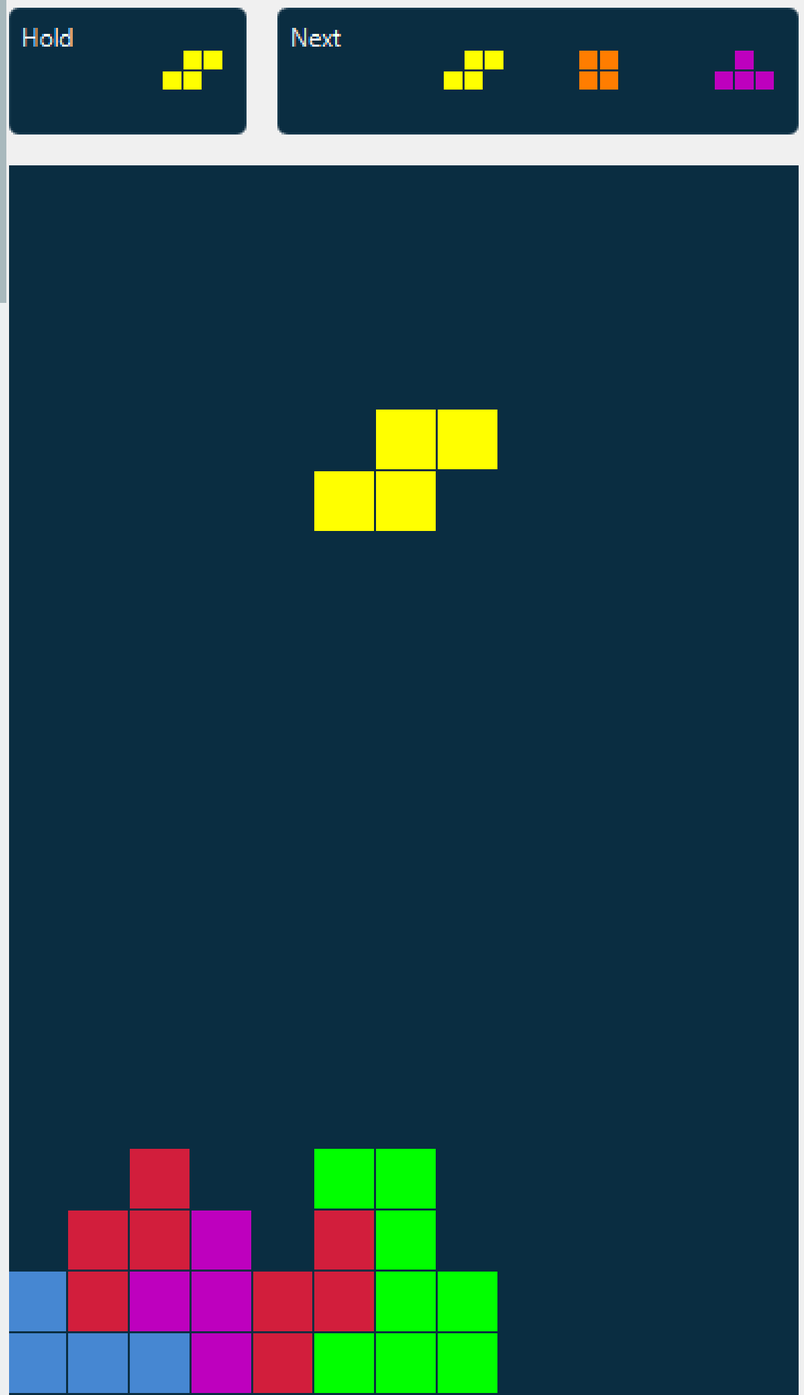
- **Overview of Tetris:**
  - Created by Alexey Pajitnov (1984).
  - Puzzle game with falling tetrominoes to complete lines.
- **Why Tetris?**
  - Iconic and widely loved game.
  - Excellent platform for learning game development.



Introduction

Purpose of the Project

- Recreate the Tetris experience.
- Add new features:
  - Hold mechanism, piece preview, customizable themes.
- **Learn advanced programming concepts:**
  - OOP, GUI with JavaFX, algorithms.



02

Software requirements

Software requirements

WHAT WE HAVE

- Friendly, efficient UI/UX design
- Useful members and task manager system.
- Easy to operate.
- Measured coding and professional thinking.

WHAT WE HAVE

- Develop a small, engaging game.
- Apply MVC architecture for scalability.
- Ensure easy maintenance and updates.
- Improve technical document reading skills.
- Enhance coding abilities.
- Gain debugging experience.

Software requirements

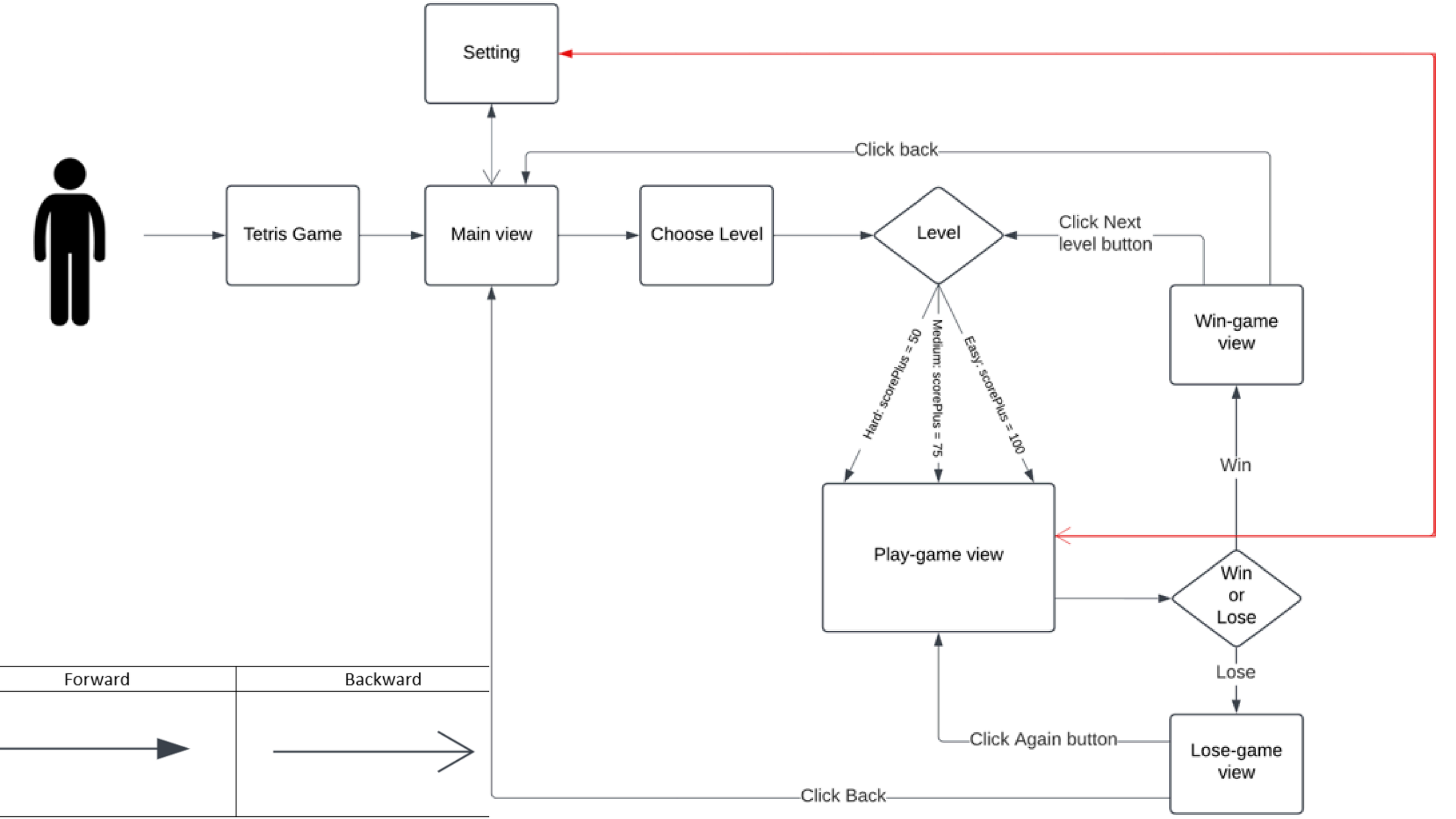
WORKING TOOLS, PLATFORM

- Intellij Idea / Vscode.
- Figma design.
- Github.
- Trello.
- draw.io

Software requirements

USE CASE SCENARIO

- We have created the use cases based on the UI/UX view of the game.


Setting
Click back
Tetris Game
Main view
Choose Level
Level
Click Next
level button
Win-game
view
Hard: scorePlus = 50
Medium: scorePlus = 75
Easy: scorePlus = 100
Win
Play-game view
Win
or
Lose
Lose
Click Again button
Lose-game
view
Click Back
Forward
Backward

03

Design & implementation

Design & implementation

Stucture - MVC Architecture

Model: The Model is the core of the application, responsible for application rules.

View: The View is responsible for the application’s interface, displaying the application's state, as dictated by the Model.

Controller: The controller connects the views and the model, acting as an intermediary, interacts with the view to render output.

Design & implementation

UI Design

- **JavaFX**: For UI and animations.
- **FXML**: Designing layouts.
- **CSS**: Styling UI component.
- **MediaPlayer**: Audio effects.

Design & implementation

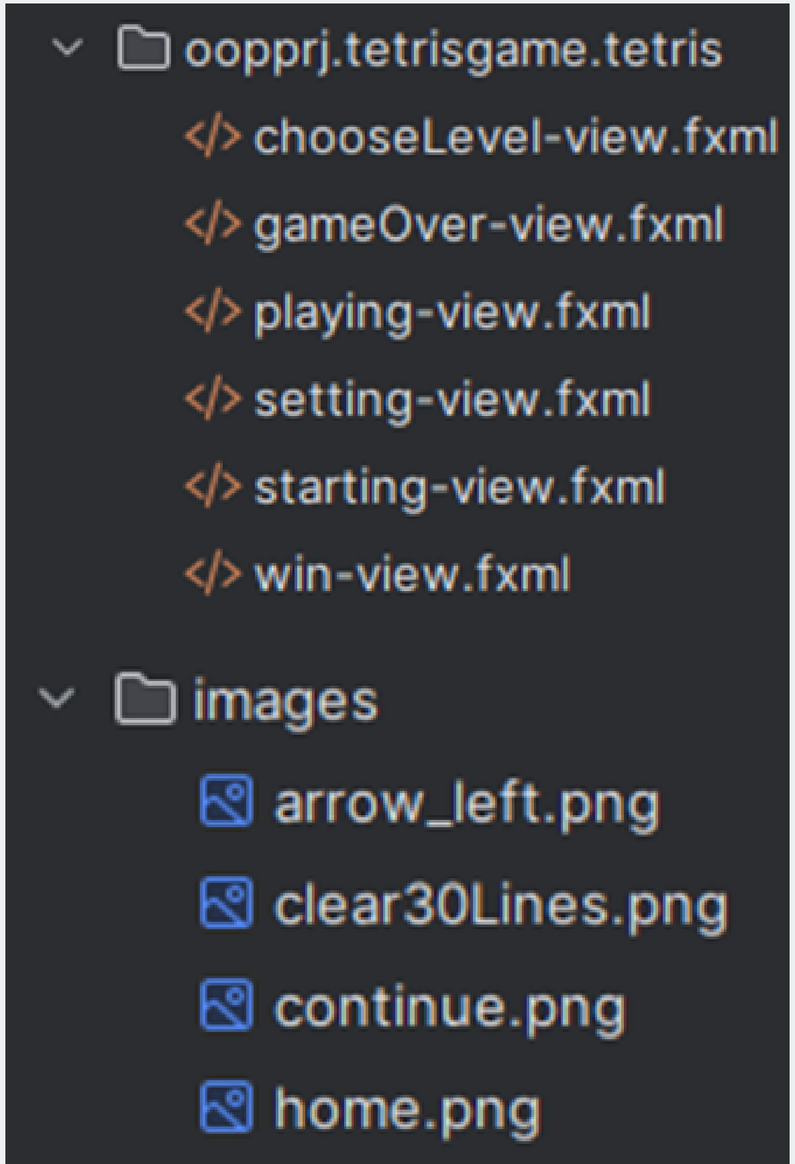
UI Design

- **oopprj.tetrisgame.tetris:** contain fxml file which is used to define the user interface in JavaFX, separating the UI design from the application logic.

Design & implementation

UI Design

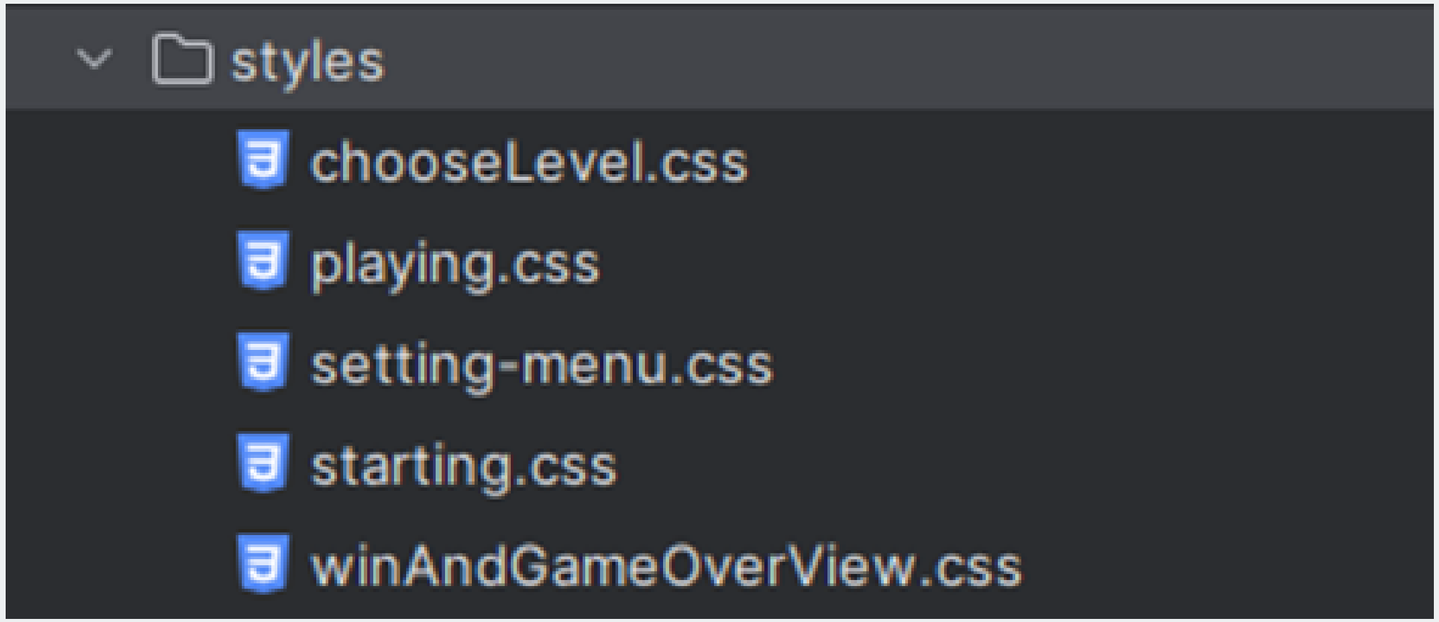
- **Icon:** contain any type of images that represent a symbol
- ·Images: background, button images,.... – most of the images for UI



Design & implementation

UI Design

- **Styles:** css files which is used for styling components



```
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<?import javafx.scene.image.ImageView?>
<?import javafx.scene.text.Font?>
<AnchorPane xmlns="http://javafx.com/javafx"
            xmlns:fx="http://javafx.com/fxml"
            fx:controller="controllers.SettingViewController"
            stylesheets="@../../../assets/styles/setting-menu.css"
focusTraversable="false">
    <Button styleClass="back-button" onAction="#backToPrevScreen" focusTraversable="false">
        <graphic>
            <ImageView fx:id="backBtnIcon" fitWidth="40" fitHeight="40"/>
        </graphic>
    </Button>

        <Label styleClass="tittle" text="Settings" layoutY="50">
            <font>
                <Font name="Arial" size="40"/>
            </font>
        </Label>
        <ImageView fx:id="musicIcon" fitWidth="50" fitHeight="50" layoutX="50" layoutY="200"/>
        <Slider fx:id="musicSlider" min="0" max="100" value="50" blockIncrement="1" layoutX="120" layoutY="230"
        onMouseDragged="#onMusicSliderChange"/>


</AnchorPane>
```

Design & implementation

UI Design

We can call css resource in fxml language by this method:

```
stylesheets="@../../../assets/styles/setting-menu.css">
```

By setting styleClass, For example:

```
<Button text="EASY"
        styleClass="choose_level_button"
        onAction="#clickEasyBtnHandler"
/>
```

Design & implementation

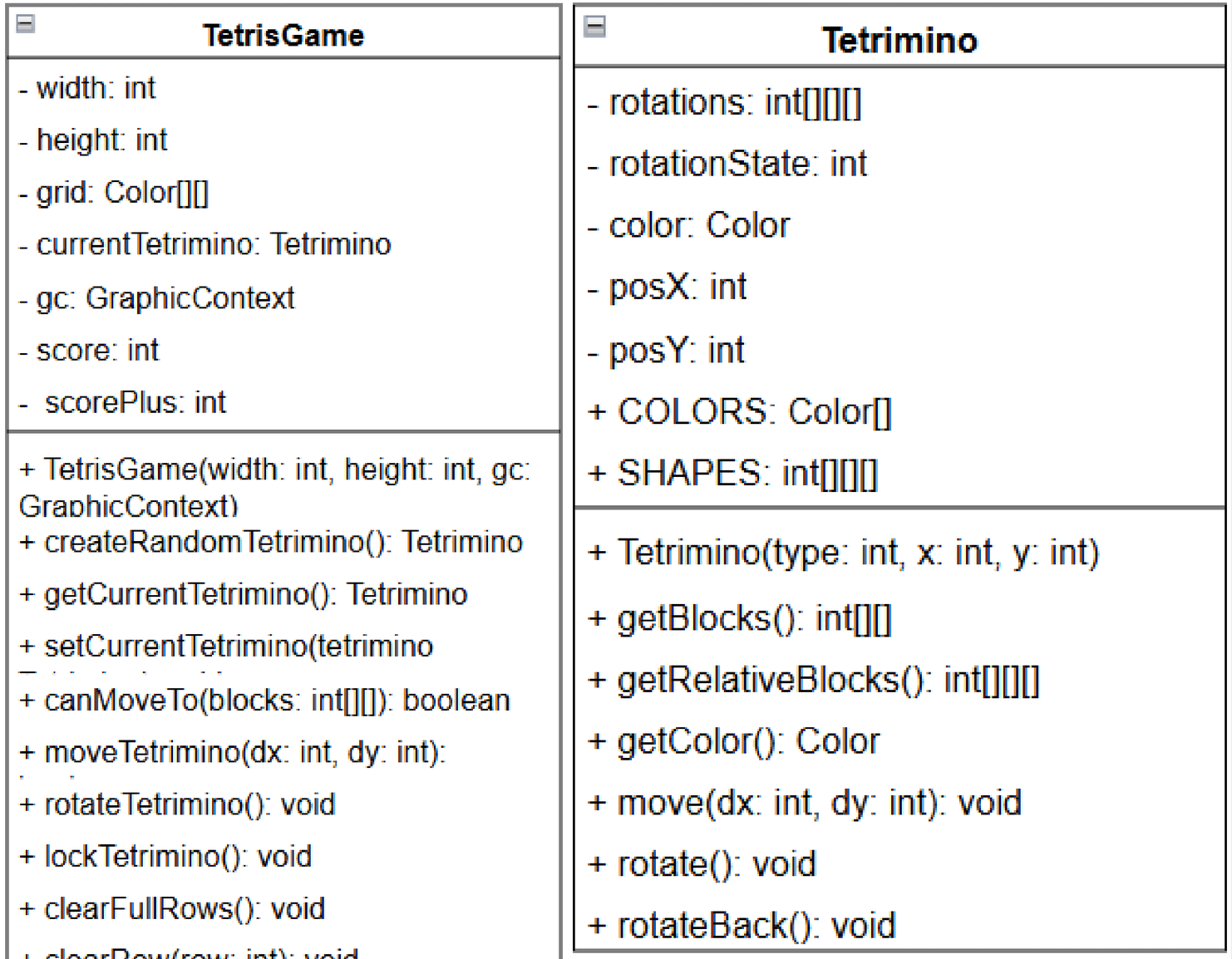
UI Design

We can link it to the class with the same name in css file:

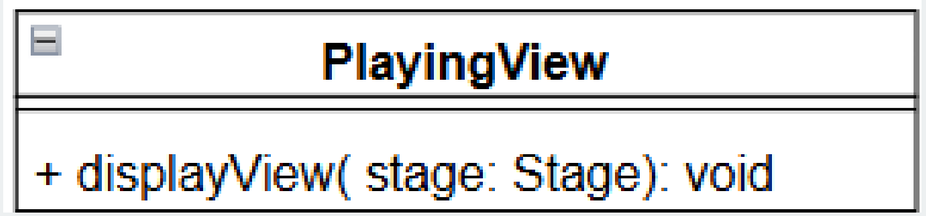
```
.choose_level_button {
    -fx-pref-width: 200px;
    -fx-pref-height: 50px;
    -fx-background-color: #0D2E43;
    -fx-text-fill: white;
    -fx-font-size: 16px;
    -fx-background-radius: 16px;
    -fx-cursor: hand;
}
```

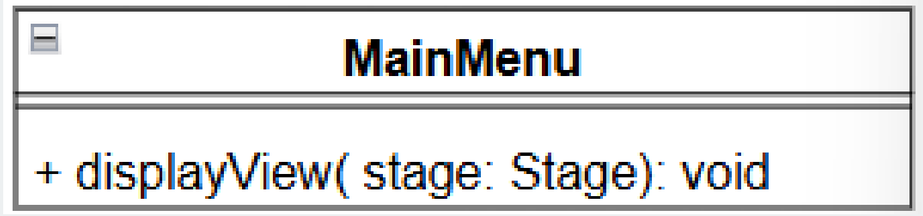
Design & implementation

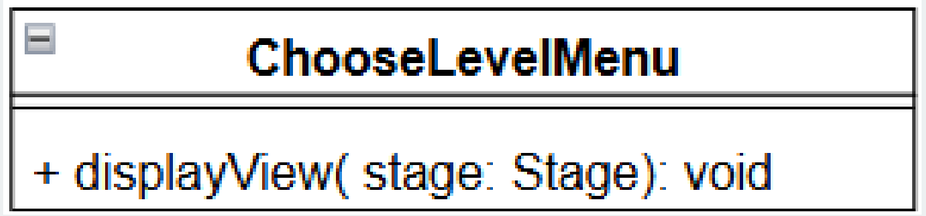
UML design

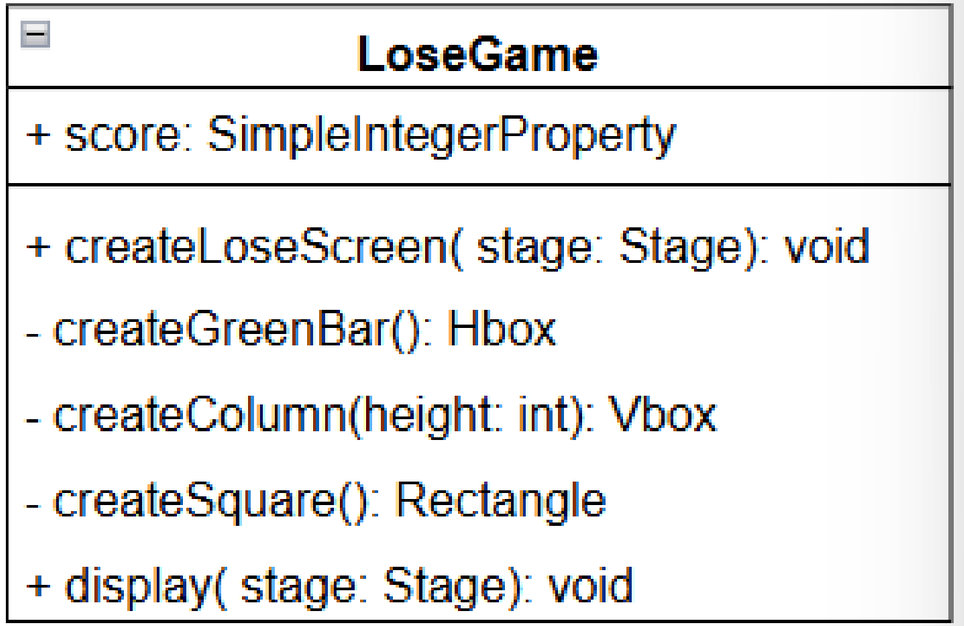

TetrisGame
- width: int
- height: int
- grid: Color[][]
- currentTetrimino: Tetrimino
- gc: GraphicContext
- score: int
- scorePlus: int
+ TetrisGame(width: int, height: int, gc: GraphicContext)
+ createRandomTetrimino(): Tetrimino
+ getCurrentTetrimino(): Tetrimino
+ setCurrentTetrimino(tetrimino
+ canMoveTo(blocks: int[][]): boolean
+ moveTetrimino(dx: int, dy: int):
+ rotateTetrimino(): void
+ lockTetrimino(): void
+ clearFullRows(): void
+ clearRow(row: int): void
+ getScore(): int
+ render(): void
+ getScorePlus(): int
+ setScorePlus(): int
Tetrimino
- rotations: int[][][]
- rotationState: int
- color: Color
- posX: int
- posY: int
+ COLORS: Color[]
+ SHAPES: int[][][]
+ Tetrimino(type: int, x: int, y: int)
+ getBlocks(): int[][]
+ getRelativeBlocks(): int[][][]
+ getColor(): Color
+ move(dx: int, dy: int): void
+ rotate(): void
+ rotateBack(): void

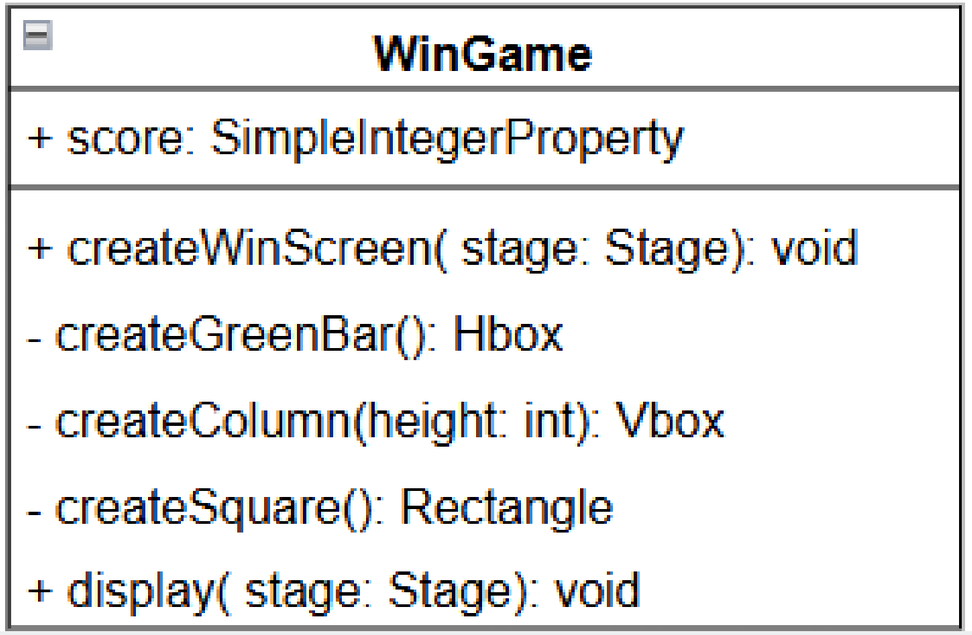

SettingMenu
+ displayView(stage: Stage): void

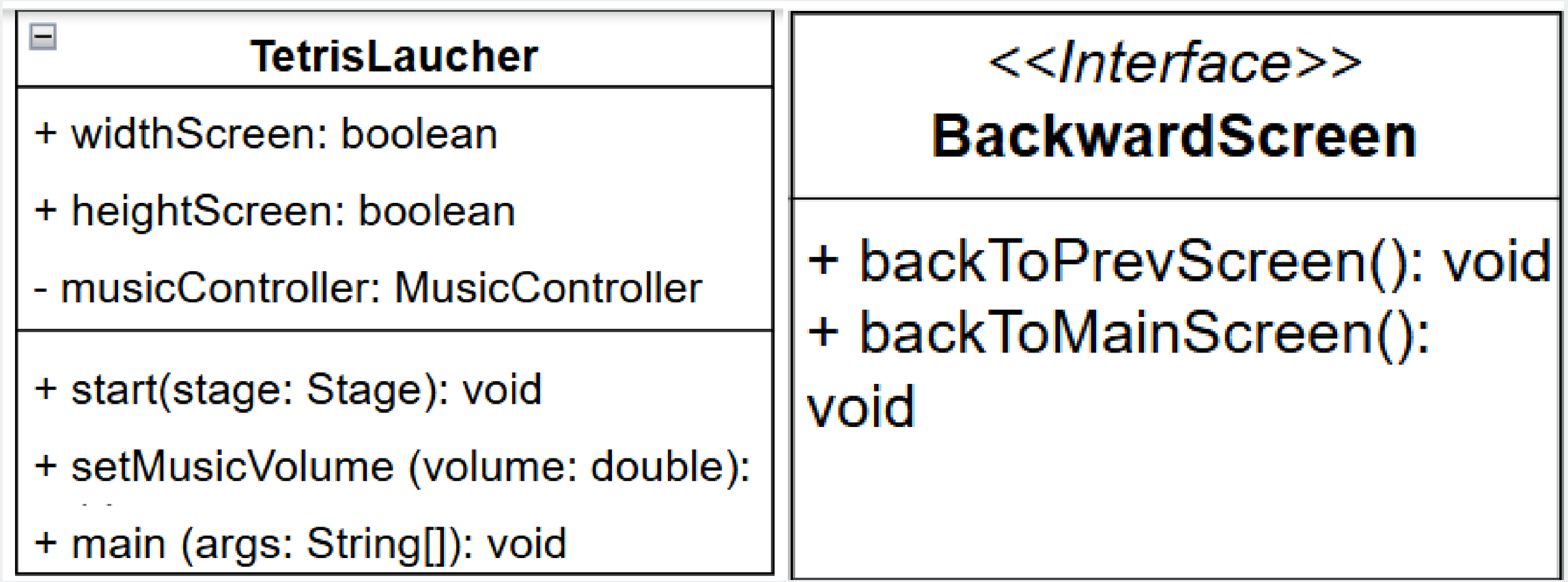

PlayingView
+ displayView(stage: Stage): void

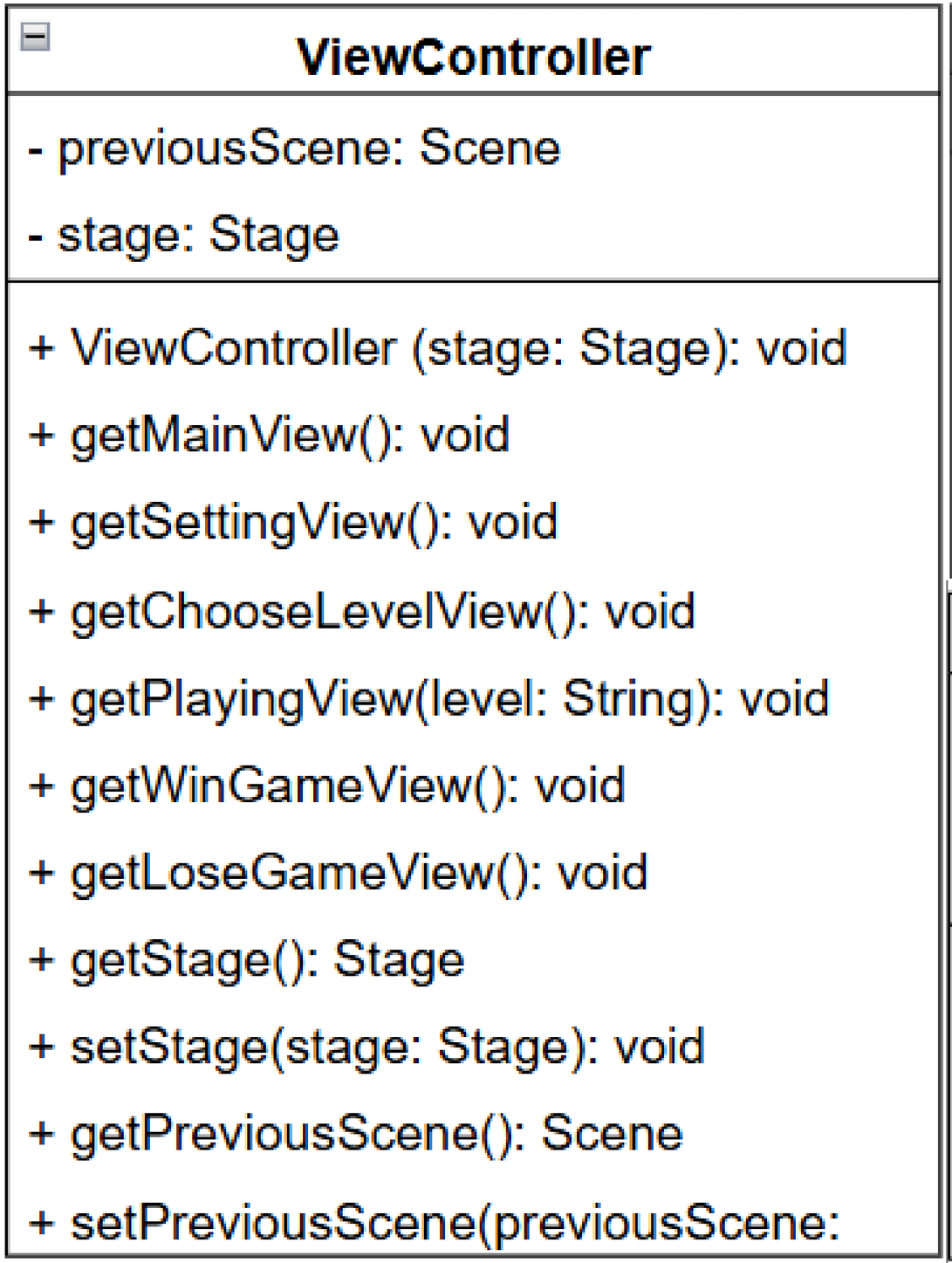

MainMenu
+ displayView(stage: Stage): void

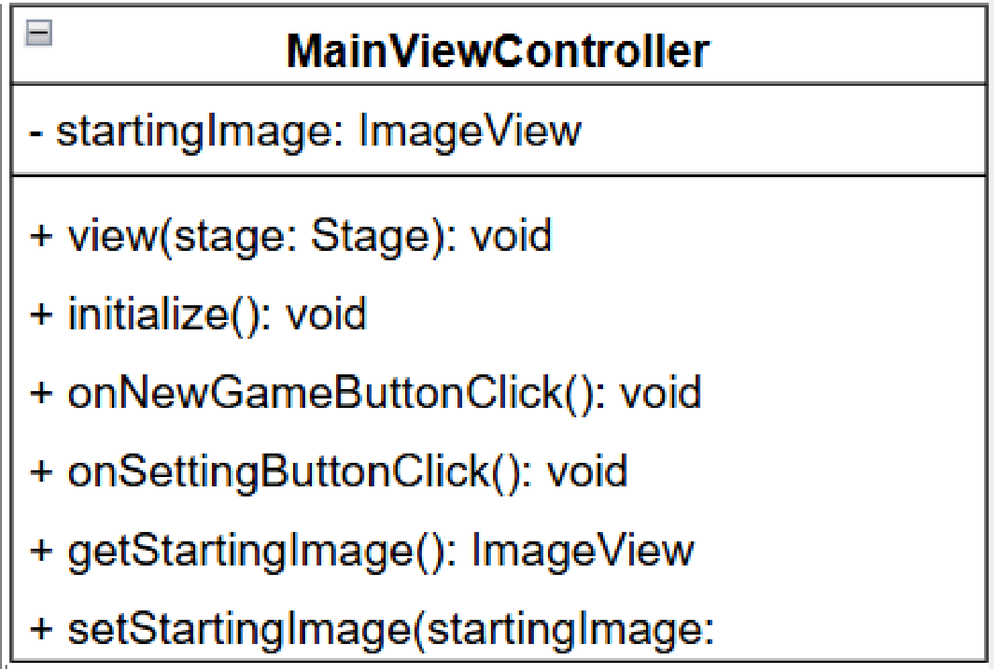

ChooseLevelMenu
+ displayView(stage: Stage): void

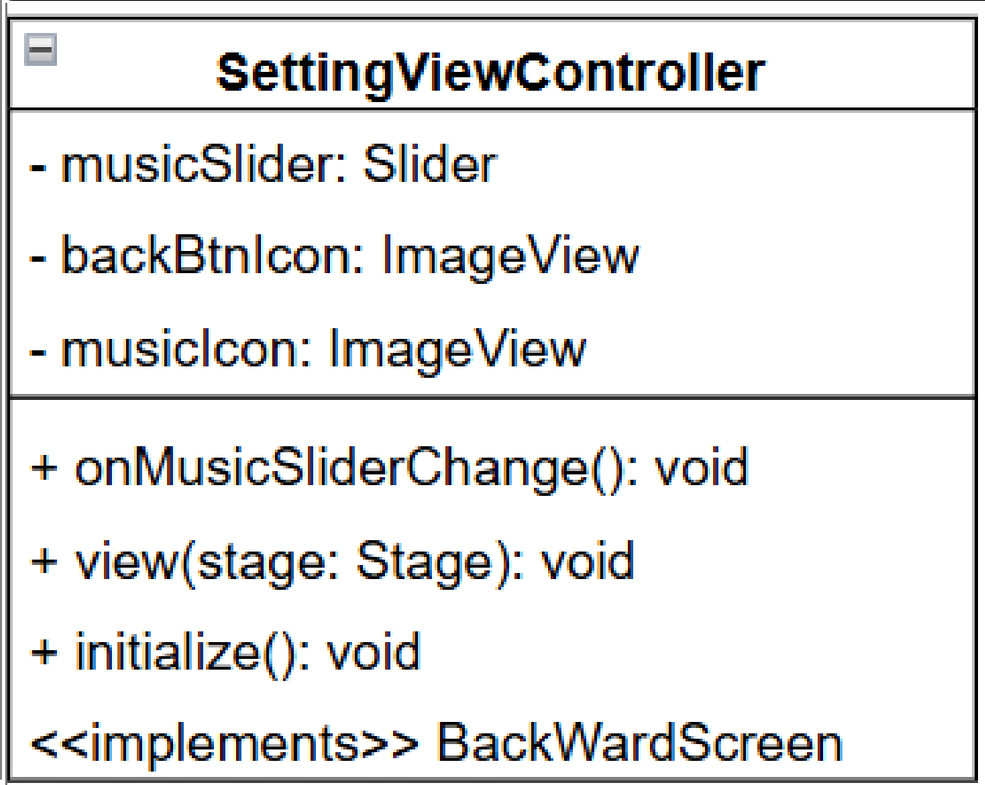

LoseGame
+ score: SimpleIntegerProperty
+ createLoseScreen(stage: Stage): void
- createGreenBar(): Hbox
- createColumn(height: int): Vbox
- createSquare(): Rectangle
+ display(stage: Stage): void

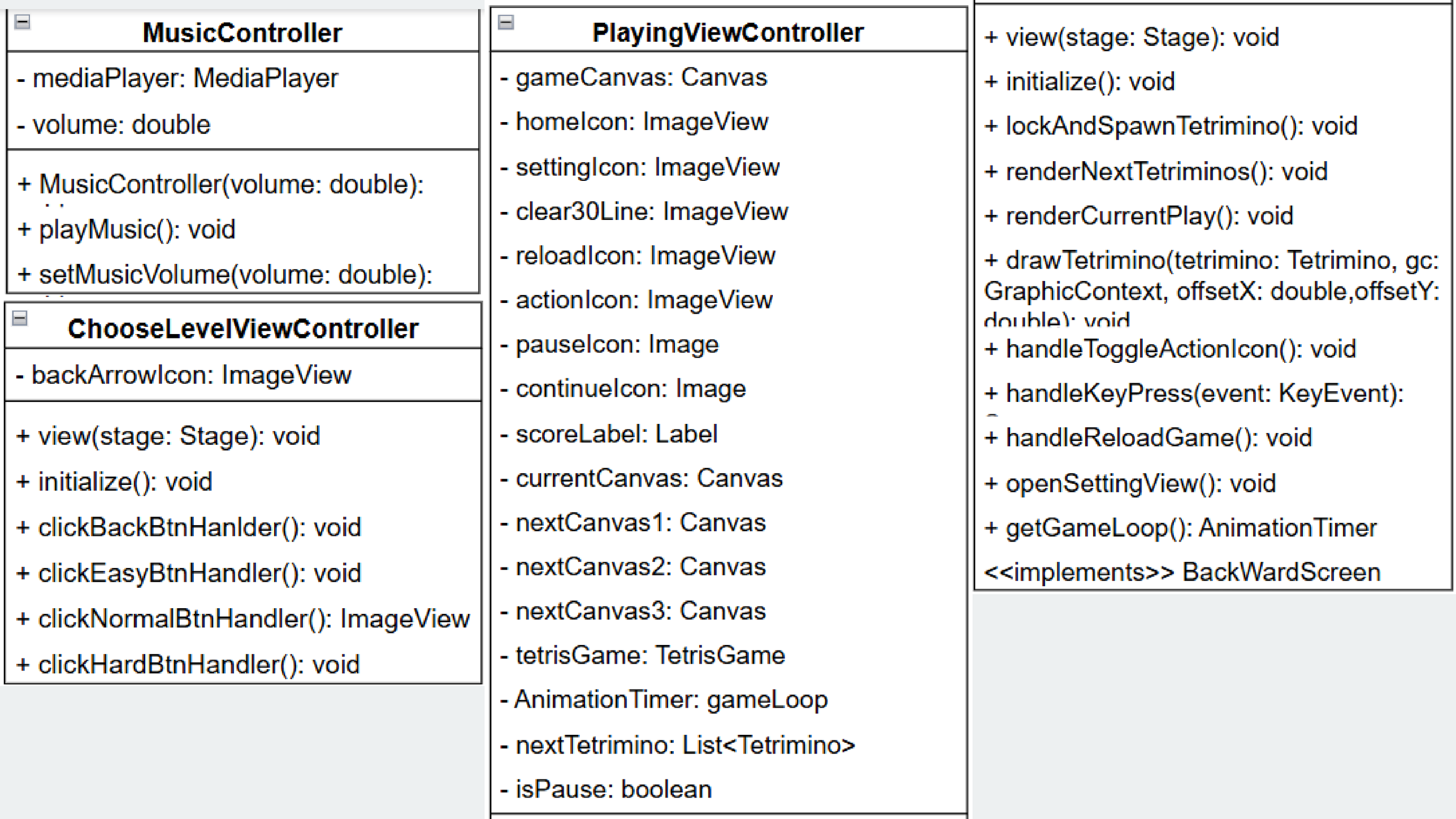

WinGame
+ score: SimpleIntegerProperty
+ createWinScreen(stage: Stage): void
- createGreenBar(): Hbox
- createColumn(height: int): Vbox
- createSquare(): Rectangle
+ display(stage: Stage): void

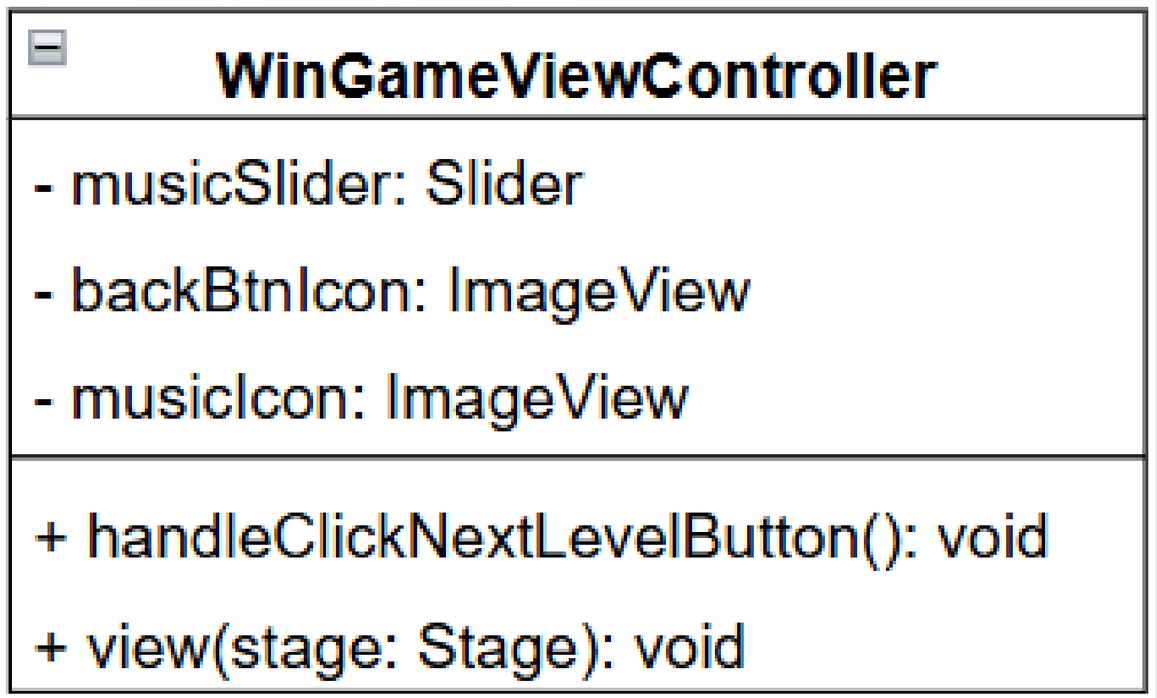

TetrisLaucher
+ widthScreen: boolean
+ heightScreen: boolean
- musicController: MusicController
+ start(stage: Stage): void
+ setMusicVolume (volume: double):
+ main (args: String[]): void
<<Interface>>
BackwardScreen
+ backToPrevScreen(): void
+ backToMainScreen():
void

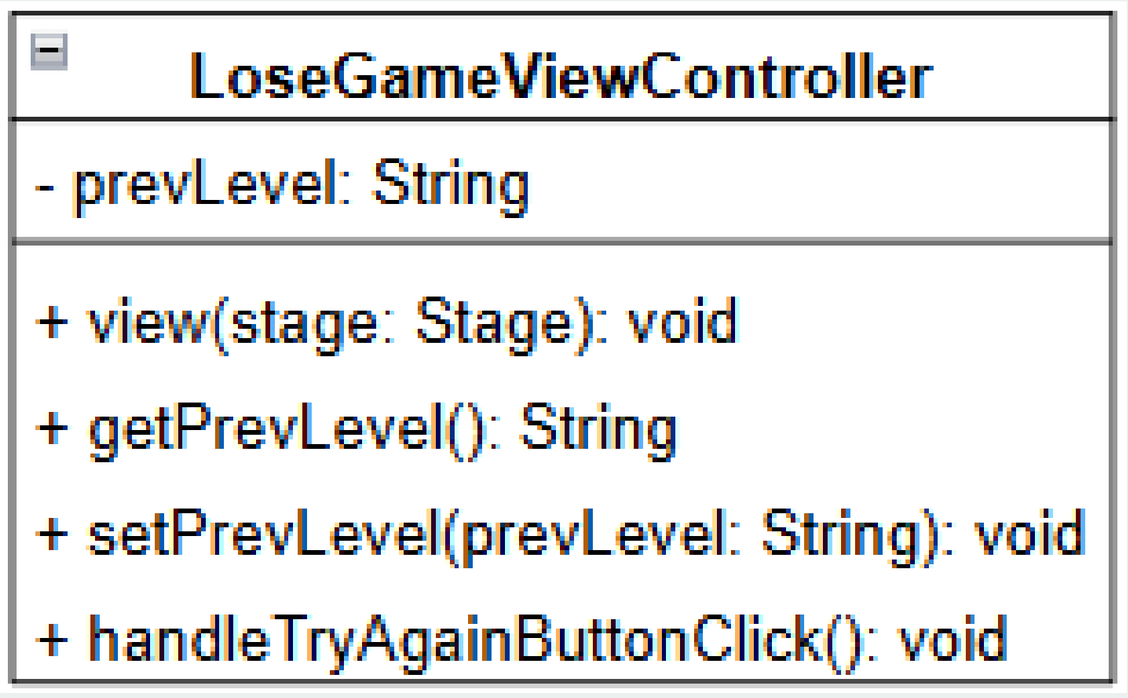

ViewController
- previousScene: Scene
- stage: Stage
+ ViewController (stage: Stage): void
+ getMainView(): void
+ getSettingView(): void
+ getChooseLevelView(): void
+ getPlayingView(level: String): void
+ getWinGameView(): void
+ getLoseGameView(): void
+ getStage(): Stage
+ setStage(stage: Stage): void
+ getPreviousScene(): Scene
+ setPreviousScene(previousScene:


MainViewController
- startingImage: ImageView
+ view(stage: Stage): void
+ initialize(): void
+ onNewGameButtonClick(): void
+ onSettingButtonClick(): void
+ getStartingImage(): ImageView
+ setStartingImage(startingImage:


SettingViewController
- musicSlider: Slider
- backBtnIcon: ImageView
- musicIcon: ImageView
+ onMusicSliderChange(): void
+ view(stage: Stage): void
+ initialize(): void
<<implements>> BackWardScreen


MusicController
- mediaPlayer: MediaPlayer
- volume: double
+ MusicController(volume: double):
+ playMusic(): void
+ setMusicVolume(volume: double):
ChooseLevelViewController
- backArrowIcon: ImageView
+ view(stage: Stage): void
+ initialize(): void
+ clickBackBtnHanlder(): void
+ clickEasyBtnHandler(): void
+ clickNormalBtnHandler(): ImageView
+ clickHardBtnHandler(): void
PlayingViewController
- gameCanvas: Canvas
- homeIcon: ImageView
- settingIcon: ImageView
- clear30Line: ImageView
- reloadIcon: ImageView
- actionIcon: ImageView
- pauseIcon: Image
- continueIcon: Image
- scoreLabel: Label
- currentCanvas: Canvas
- nextCanvas1: Canvas
- nextCanvas2: Canvas
- nextCanvas3: Canvas
- tetrisGame: TetrisGame
- AnimationTimer: gameLoop
- nextTetrimino: List<Tetrimino>
- isPause: boolean
+ view(stage: Stage): void
+ initialize(): void
+ lockAndSpawnTetrimino(): void
+ renderNextTetriminos(): void
+ renderCurrentPlay(): void
+ drawTetrimino(tetrimino: Tetrimino, gc: GraphicContext, offsetX: double,offsetY: double): void
+ handleToggleActionIcon(): void
+ handleKeyPress(event: KeyEvent):
+ handleReloadGame(): void
+ openSettingView(): void
+ getGameLoop(): AnimationTimer
<<implements>> BackWardScreen

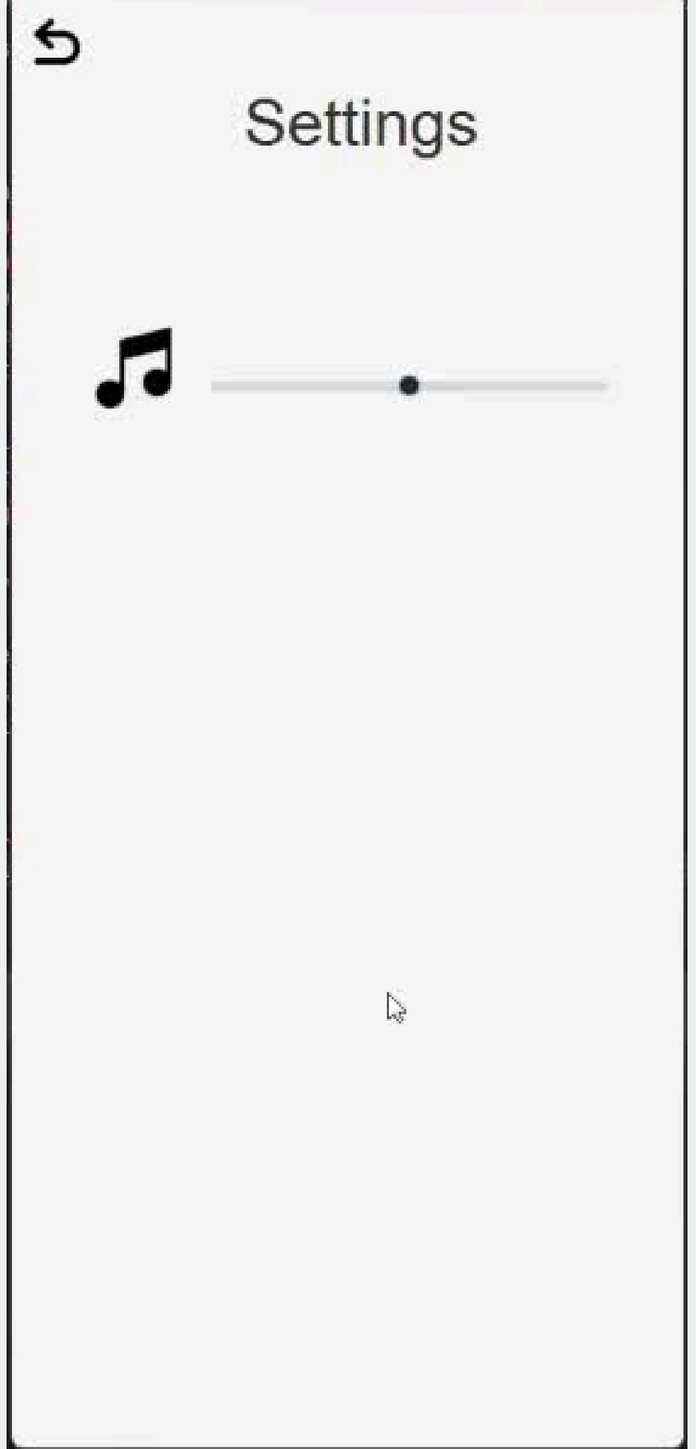

WinGameViewController
- musicSlider: Slider
- backBtnIcon: ImageView
- musicIcon: ImageView
+ handleClickNextLevelButton(): void
+ view(stage: Stage): void


LoseGameViewController
- prevLevel: String
+ view(stage: Stage): void
+ getPrevLevel(): String
+ setPrevLevel(prevLevel: String): void
+ handleTryAgainButtonClick(): void

How to play

Left Arrow (←): Move the tetromino left.
Right Arrow (→): Move the tetromino right.
Up Arrow (↑): Rotate the tetromino 90 degrees clockwise.
Down Arrow (↓): Speed up the falling of the tetromino (double the falling speed).

Demo



Thank You