Life --

Wyatt Smith, Jerrick Fong, Alex Ottoboni, Hristo Stoytchev, Sonia Tomas

Outline

1. Description
    a. Demo
2. Design
    a. Activity
    b. Sequence
    c. State
    d. Class
3. Testing
    a. Unit Testing

Description

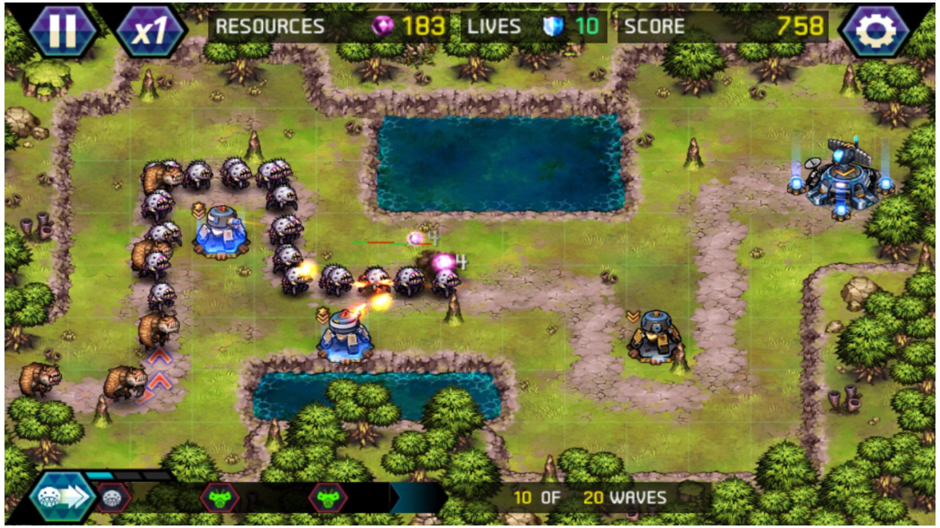
Context - Terminology

Tower Defense

Hack-and-slash

Problem

- Most games focus on defensive strategy or frenzied attacking
- Player idly sits and watches action



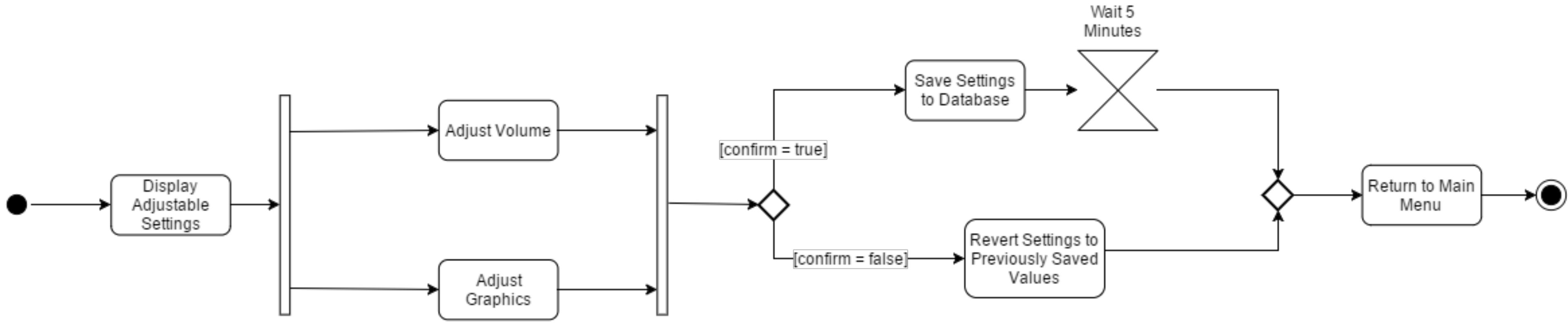
Aim/Goal

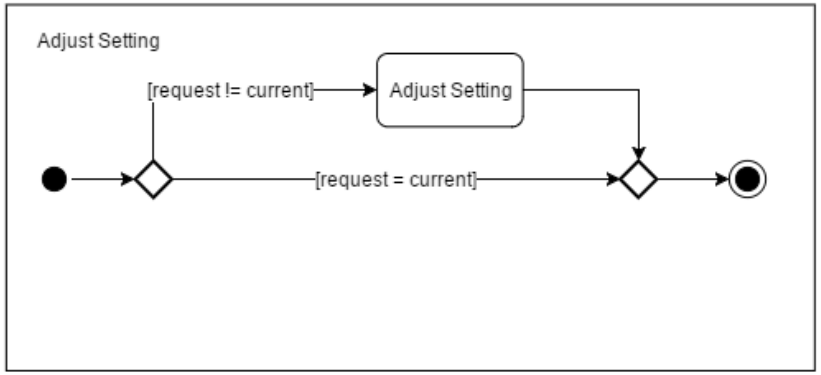
- Combine the two genres
- Actively engaging
- Require some thought

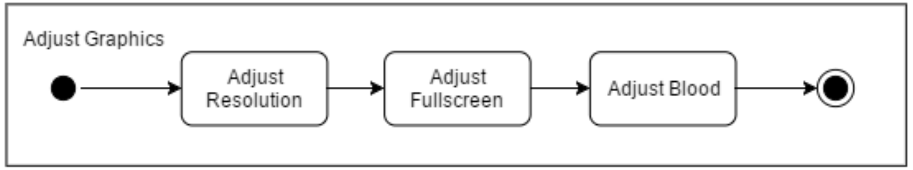
Demo

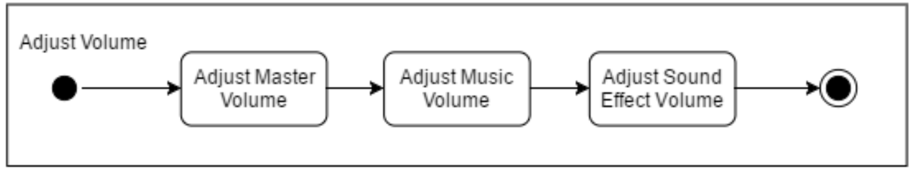
Activity Diagrams

Change Settings

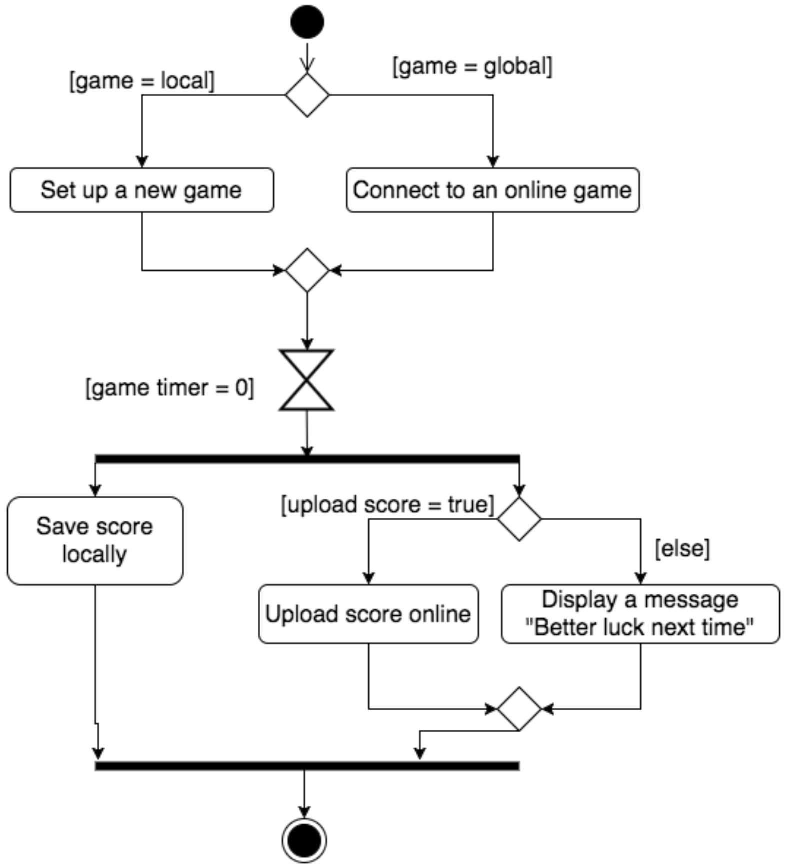

Display Adjustable Settings
Adjust Volume
Adjust Graphics
[confirm = true]
Save Settings to Database
Wait 5 Minutes
[confirm = false]
Revert Settings to Previously Saved Values
Return to Main Menu



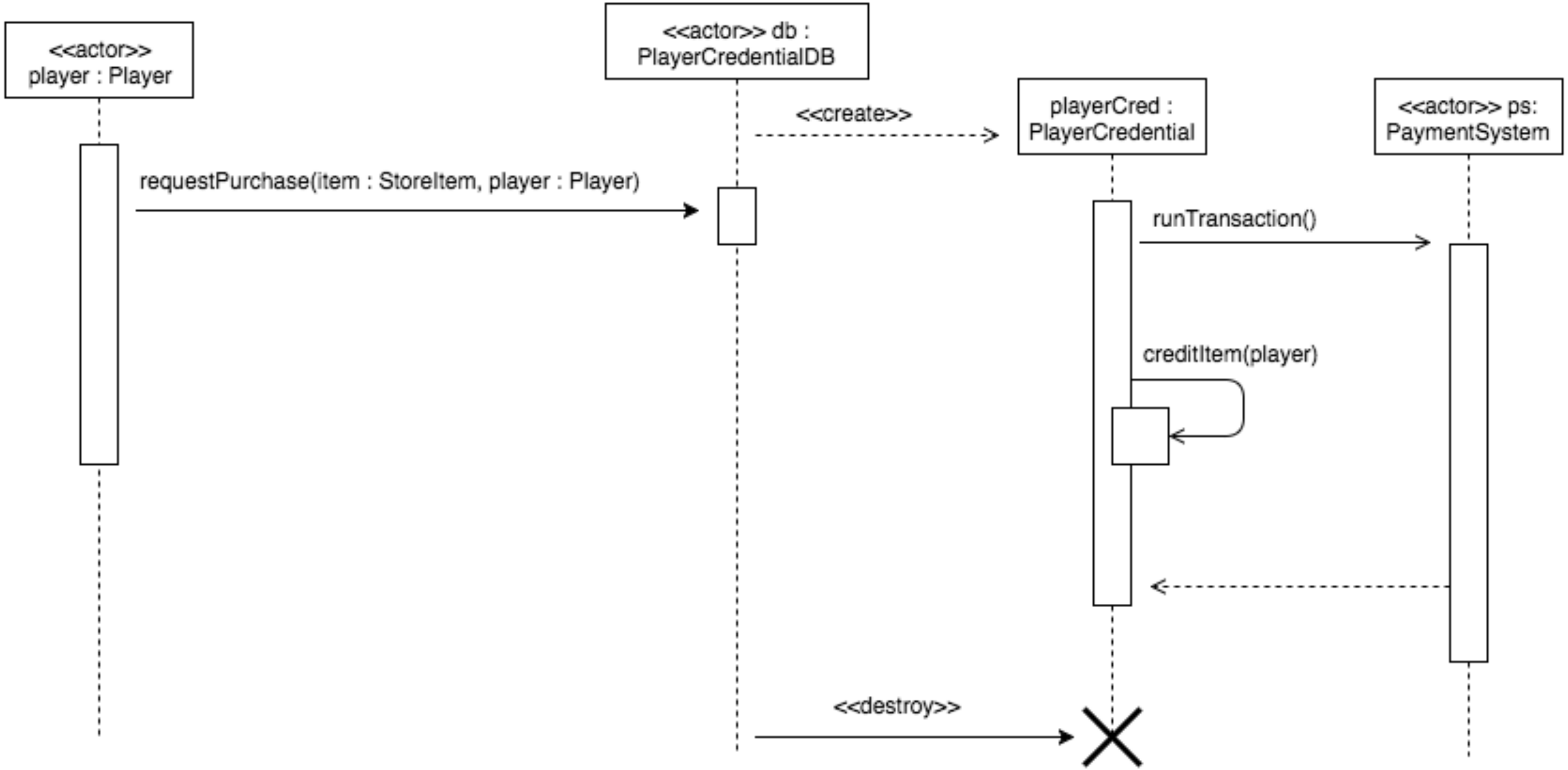




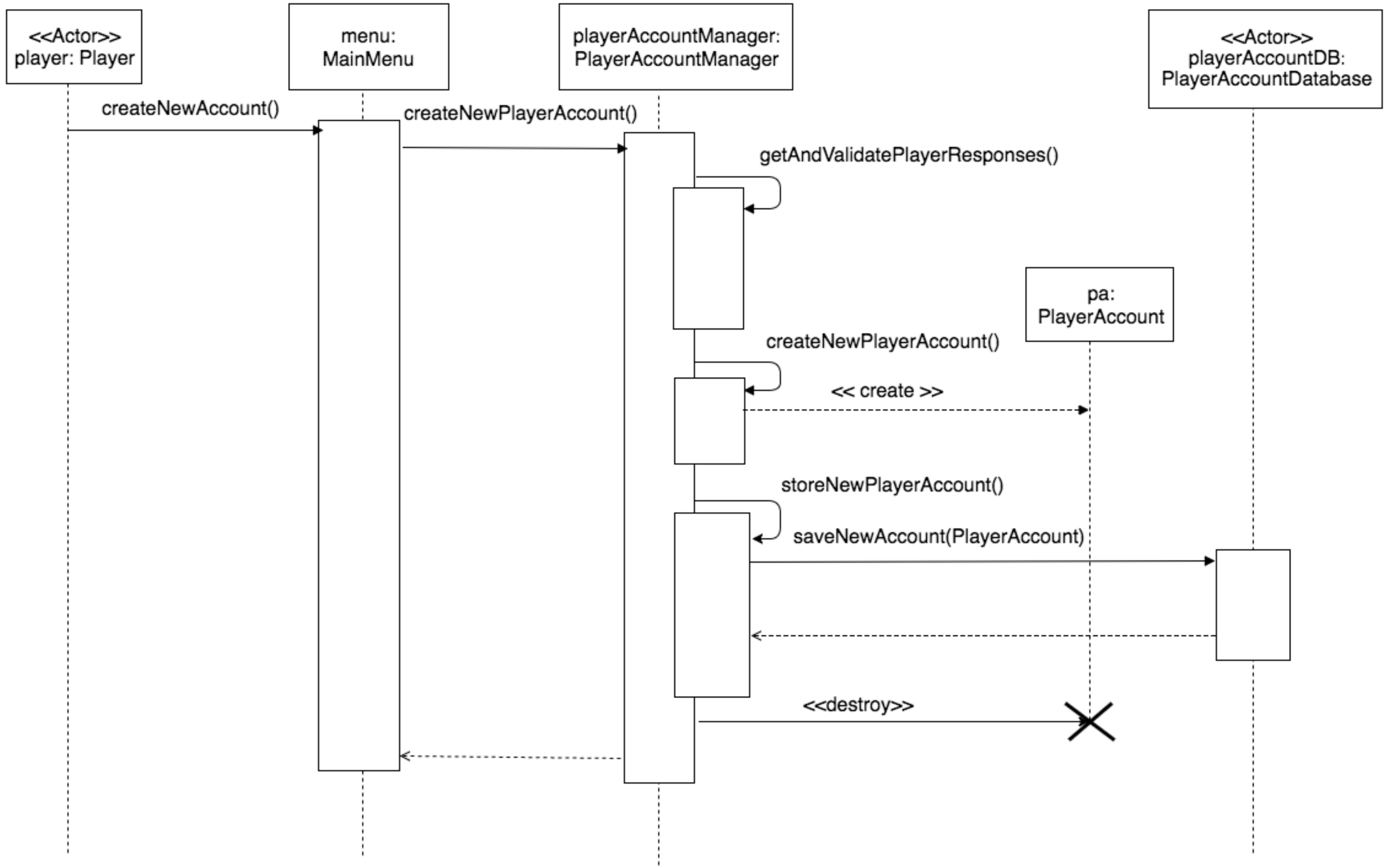
Play Game


[game = local]
[game = global]
Set up a new game
Connect to an online game
[game timer = 0]
Save score locally
[upload score = true]
[else]
Upload score online
Display a message "Better luck next time"

Sequence Diagram

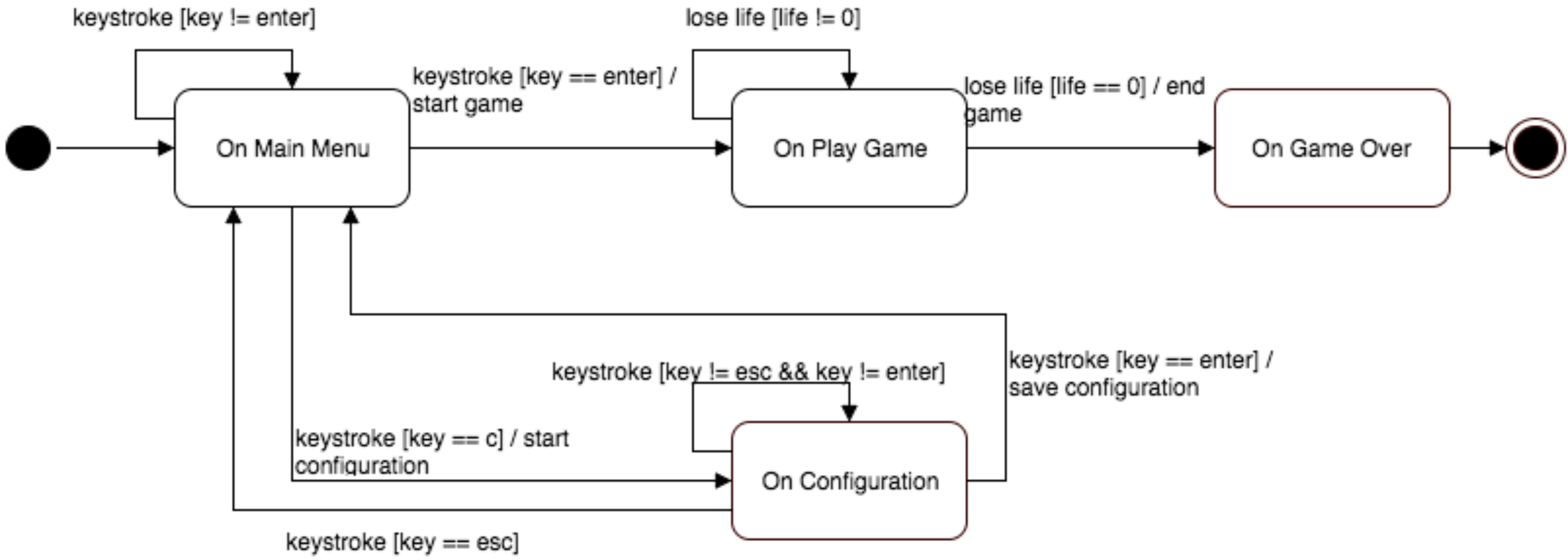

<<actor>> player : Player
<<actor>> db : PlayerCredentialDB
<<create>>
playerCred : PlayerCredential
<<actor>> ps: PaymentSystem
requestPurchase(item : StoreItem, player : Player)
runTransaction()
creditItem(player)
<<destroy>>

Create New Player Account



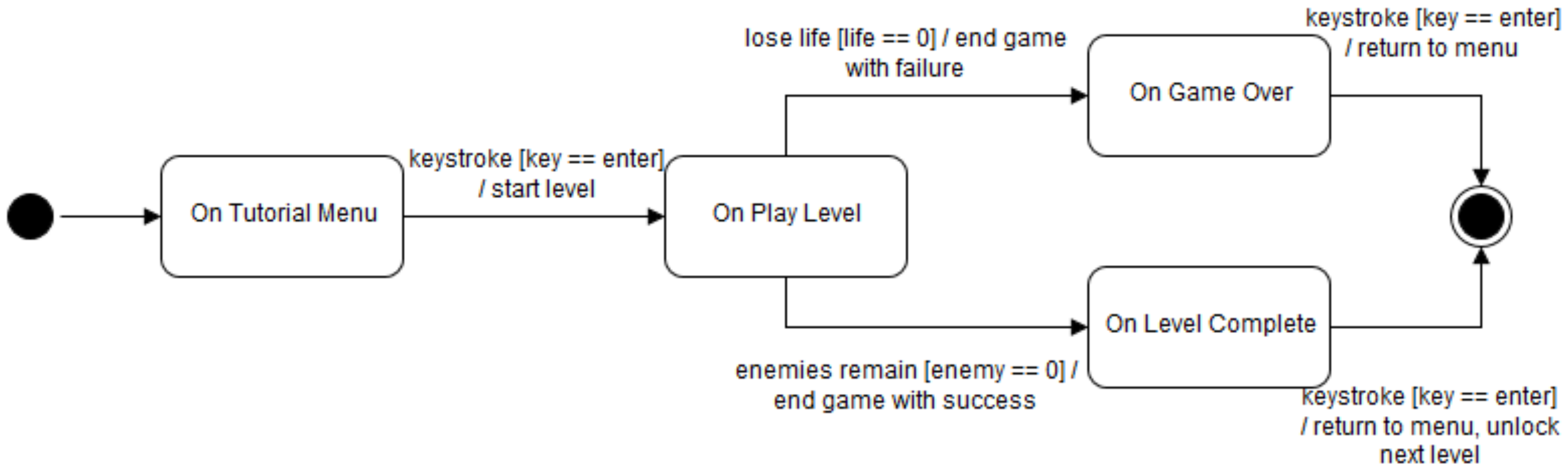
State Diagrams

Play Game

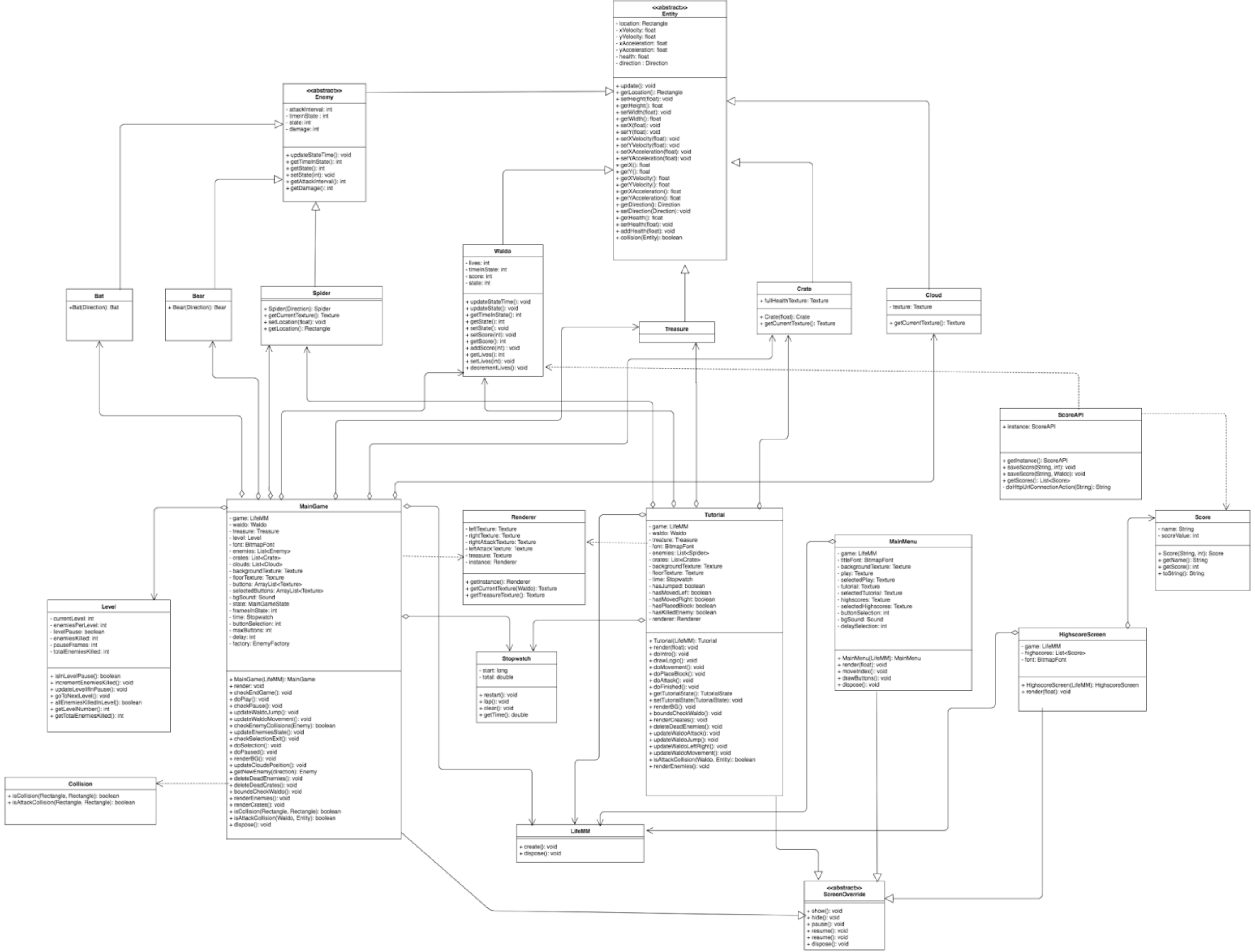


keystroke [key != enter]
lose life [life != 0]
keystroke [key == enter] / start game
lose life [life == 0] / end game
On Main Menu
On Play Game
On Game Over
keystroke [key != esc && key != enter]
keystroke [key == enter] / save configuration
keystroke [key == c] / start configuration
On Configuration
keystroke [key == esc]

Play Tutorial


On Tutorial Menu
keystroke [key == enter] / start level
On Play Level
lose life [life == 0] / end game with failure
On Game Over
keystroke [key == enter] / return to menu
enemies remain [enemy == 0] / end game with success
On Level Complete
keystroke [key == enter] / return to menu, unlock next level

Class Diagram


<<abstract>>
Entity
- location: Rectangle
- xVelocity: float
- yVelocity: float
- xAcceleration: float
- yAcceleration: float
- health: float
- direction : Direction
+ update(): void
+ getLocation(): Rectangle
+ setHeight(float): void
+ getHeight(): float
+ setWidth(float): void
+ getWidth(): float
+ setX(float): void
+ setY(float): void
+ setXVelocity(float): void
+ setYVelocity(float): void
+ setXAcceleration(float): void
+ setYAcceleration(float): void
+ getX(): float
+ getY(): float
+ getXVelocity(): float
+ getYVelocity(): float
+ getXAcceleration(): float
+ getYAcceleration(): float
+ getDirection(): Direction
+ setDirection(Direction): void
+ getHealth(): float
+ setHealth(float): void
+ addHealth(float): void
+ collision(Entity): boolean
<<abstract>>
Enemy
- attackInterval: int
- timeInState : int
- state: int
- damage: int
+ updateStateTime(): void
+ getTimeInState(): int
+ getState(): int
+ setState(int): void
+ getAttackInterval(): int
+ getDamage(): int
Bat
+Bat(Direction): Bat
Bear
+ Bear(Direction): Bear
Spider
+ Spider(Direction): Spider
+ getCurrentTexture(): Texture
+ setLocation(float): void
+ getLocation(): Rectangle
Waldo
- lives: int
- timeInState: int
- score: int
- state: int
+ updateStateTime(): void
+ updateState(): void
+ getTimeInState(): int
+ getState(): int
+ setState(): void
+ setScore(int): void
+ getScore(): int
+ addScore(int) : void
+ getLives(): int
+ setLives(int): void
+ decrementLives(): void
Treasure
Crate
+ fullHealthTexture: Texture
+ Crate(float): Crate
+ getCurrentTexture(): Texture
Cloud
- texture: Texture
+ getCurrentTexture(): Texture
ScoreAPI
+ instance: ScoreAPI
+ getInstance(): ScoreAPI
+ saveScore(String, int): void
+ saveScore(String, Waldo): void
+ getScores(): List<Score>
- doHttpUrlConnectionAction(String): String
Score
- name: String
- scoreValue: int
+ Score(String, int): Score
+ getName(): String
+ getScore(): int
+ toString(): String
MainGame
- game: LifeMM
- waldo: Waldo
- treasure: Treasure
- level: Level
- font: BitmapFont
- enemies: List<Enemy>
- crates: List<Crate>
- clouds: List<Cloud>
- backgroundTexture: Texture
- floorTexture: Texture
- buttons: ArrayList<Texture>
- selectedButtons: ArrayList<Texture>
- bgSound: Sound
- state: MainGameState
- framesInState: int
- time: Stopwatch
- buttonSelection: int
- maxButtons: int
- delay: int
- factory: EnemyFactory
+ MainGame(LifeMM): MainGame
+ render: void
+ checkEndGame(): void
+ doPlay(): void
+ checkPause(): void
+ updateWaldoJump(): void
+ updateWaldoMovement(): void
+ checkEnemyCollisions(Enemy): boolean
+ updateEnemiesState(): void
+ checkSelectionExit(): void
+ doSelection(): void
+ doPaused(): void
+ renderBG(): void
+ updateCloudsPosition(): void
+ getNewEnemy(direction): Enemy
+ deleteDeadEnemies(): void
+ deleteDeadCrates(): void
+ boundsCheckWaldo(): void
+ renderEnemies(): void
+ renderCrates(): void
+ isCollision(Rectangle, Rectangle): boolean
+ isAttackCollision(Waldo, Entity): boolean
+ dispose(): void
Level
- currentLevel: int
- enemiesPerLevel: int
- levelPause: boolean
- enemiesKilled: int
- pauseFrames: int
- totalEnemiesKilled: int
+ isInLevelPause(): boolean
+ incrementEnemiesKilled(): void
+ updateLevelIfInPause(): void
+ goToNextLevel(): void
+ allEnemiesKilledInLevel(): boolean
+ getLevelNumber(): int
+ getTotalEnemiesKilled(): int
Collision
+ isCollision(Rectangle, Rectangle): boolean
+ isAttackCollision(Rectangle, Rectangle): boolean
Renderer
- leftTexture: Texture
- rightTexture: Texture
- rightAttackTexture: Texture
- leftAttackTexture: Texture
- treasure: Texture
- instance: Renderer
+ getInstance(): Renderer
+ getCurrentTexture(Waldo): Texture
+ getTreasureTexture(): Texture
Stopwatch
- start: long
- total: double
+ restart(): void
+ lap(): void
+ clear(): void
+ getTime(): double
Tutorial
- game: LifeMM
- waldo: Waldo
- treature: Treasure
- font: BitmapFont
- enemies: List<Spider>
- crates: List<Crate>
- backgroundTexture: Texture
- floorTexture: Texture
- time: Stopwatch
- hasJumped: boolean
- hasMovedLeft: boolean
- hasMovedRight: boolean
- hasPlacedBlock: boolean
- hasKilledEnemy: boolean
- renderer: Renderer
+ Tutorial(LifeMM): Tutorial
+ render(float): void
+ doIntro(): void
+ drawLogic(): void
+ doMovement(): void
+ doPlaceBlock(): void
+ doAttack(): void
+ doFinished(): void
+ getTutorialState(): TutorialState
+ setTutorialState(TutorialState): void
+ renderBG(): void
+ boundsCheckWaldo(): void
+ renderCreates(): void
+ deleteDeadEnemies(): void
+ updateWaldoAttack(): void
+ updateWaldoJump(): void
+ updateWaldoLeftRight(): void
+ updateWaldoMovement(): void
+ isAttackCollision(Waldo, Entity): boolean
+ renderEnemies(): void
MainMenu
- game: LifeMM
- titleFont: BitmapFont
- backgroundTexture: Texture
- play: Texture
- selectedPlay: Texture
- tutorial: Texture
- selectedTutorial: Texture
- highscores: Texture
- selectedHighscores: Texture
- buttonSelection: int
- bgSound: Sound
- delaySelection: int
+ MainMenu(LifeMM): MainMenu
+ render(float): void
+ moveIndex(): void
+ drawButtons(): void
+ dispose(): void
HighscoreScreen
- game: LifeMM
- highscores: List<Score>
- font: BitmapFont
+ HighscoreScreen(LifeMM): HighscoreScreen
+ render(float): void
LifeMM
+ create(): void
+ dispose(): void
<<abstract>>
ScreenOverride
+ show(): void
+ hide(): void
+ pause(): void
+ resume(): void
+ resume(): void
+ dispose(): void

SonarQube Demo

Design Patterns

High Score Singleton

Uploads high score to server

Need single instance of uploader

```
public class ScoreAPI {
    private static ScoreAPI instance;
    private ScoreAPI() {

    }
    /**
     * Gets or creates an instance of the ScoreAPI.
     * @return The instance of ScoreAPI.
    */
    public static ScoreAPI getInstance() {
        if (instance == null) {
            instance = new ScoreAPI();
        }
        return instance;
    }
    // Additional methods for instance.
```

Renderer Singleton

Stores textures to draw on screen

Separate GUI and logic

```
public class Renderer {

          private Renderer() {
                    this.leftTexture = new Texture(“playersmall.png”);
                    this.rightTexture = new Texture(“playersmallr.png”);
                    this.treasure = new Texture(“heart.png”);
          }

          public static Renderer getInstance() {
                    if (instance == null) {
                              instance = new Renderer();
                    }
                    return instance;
          }
}
```

Enemy Factory

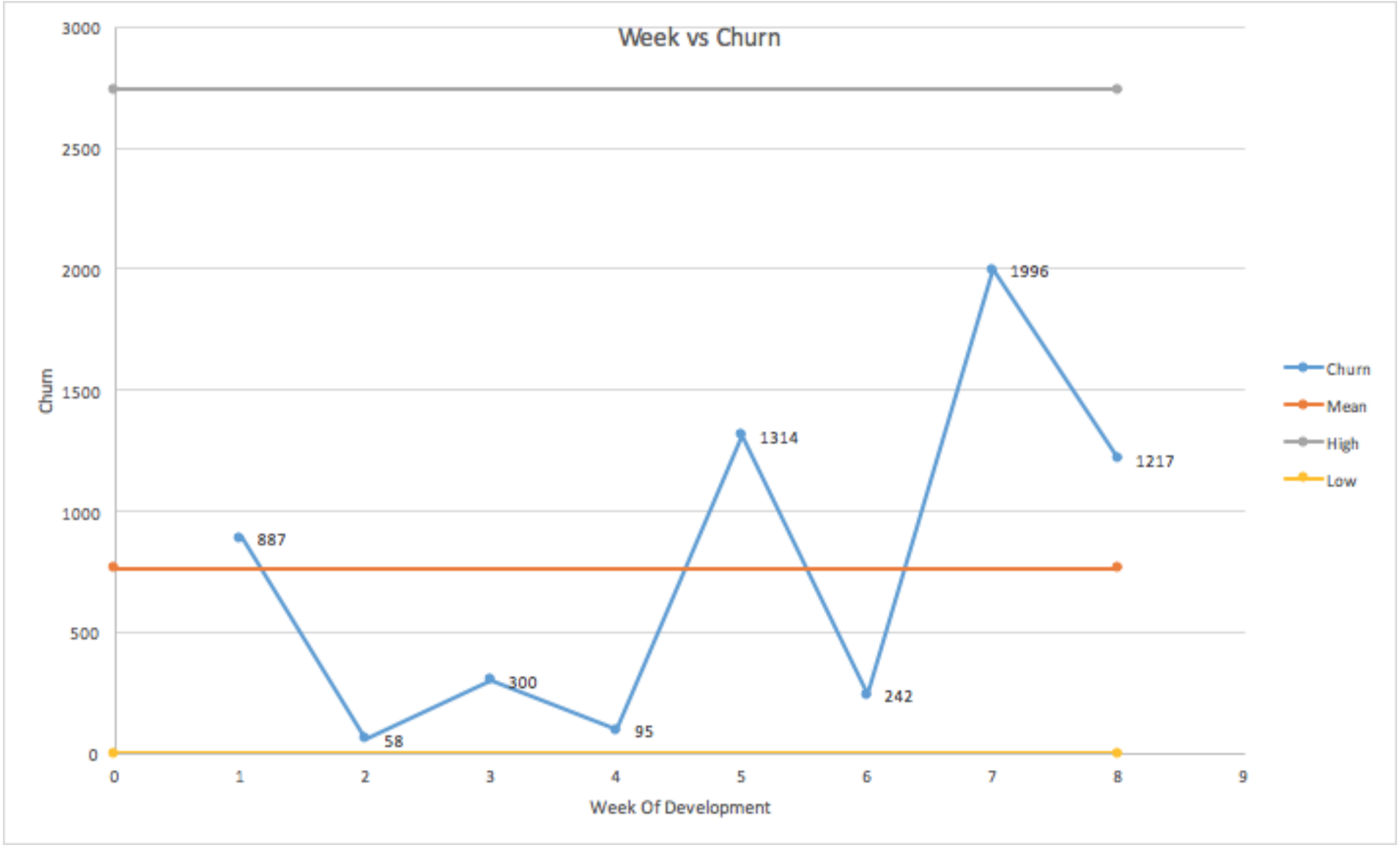
Enemies are created randomly

Factory used to generate specific enemy

```
if (enemyType.equalsIgnoreCase("SPIDER")) {
         return new Spider();
      } else if (enemyType.equalsIgnoreCase("BEAR")) {
         return new Bear();
      } else if (enemyType.equalsIgnoreCase("BAT")) {
         return new Bat();
      }
}
```

Tests (Demo)

Statistical Process Control Chart



Conclusions

Game met all functional requirements

- Basic movement
- Attacking
- Scores
- Dynamic enemies

Future

- Implement more non-gameplay features (Transactions)
- Add animations
- Add interactions with players/enemies and blocks

Questions?