Group 7

Assignment 2

Authors:
Matthijs Verzijl - 4282604
Emilie de Bree - 4247558
Jim Hommes - 4306090
Toine Hartman - 4305655
Jeffrey Helgers - 4318749

Supervisor:
Alberto Bacchelli

Teaching Assistant:
Aaron Ang

TU Delft Delft University of Technology

October 6, 2015

Contents

1. Resonsibility Driven Design

Here the features of Respsonsibility Driven Design will be explained per feature we added this sprint.

1.1 Jumping on Bubbles

Player

Responsibility:
The player class will handle the movement of the player.
Collaborations:
It will be communicating with SpriteBase for collisions.

SpriteBase

Responsibility:
SpriteBase will be responsible for detecting collisions with BubbleSprites.
Collaborations:
It will be communicating with Player.

1.2 Warping

Level

Responsibility:
When loading the map, it should be clear where the player is able to warp.
Collaborations:
When the player collides with a warpable area, Level should communicate with Player to tell him where he warped to.

Player

Responsibility:
This class will be moving the player and spawning it after the warp.
Collaborations:
When the player warped, it communicates with ScreenController to be redrawn.

1.3 Refactoring Controllers for Testing

LevelController

Responsibility:
The responsibility for this class will *only* include creating levels and all communication between the objects in the game.
Collaborations:

It will communicate with the objects in de game (Model package) and with the MainController. It will also be the bridge to ScreenController for the SpriteBases.

MainController

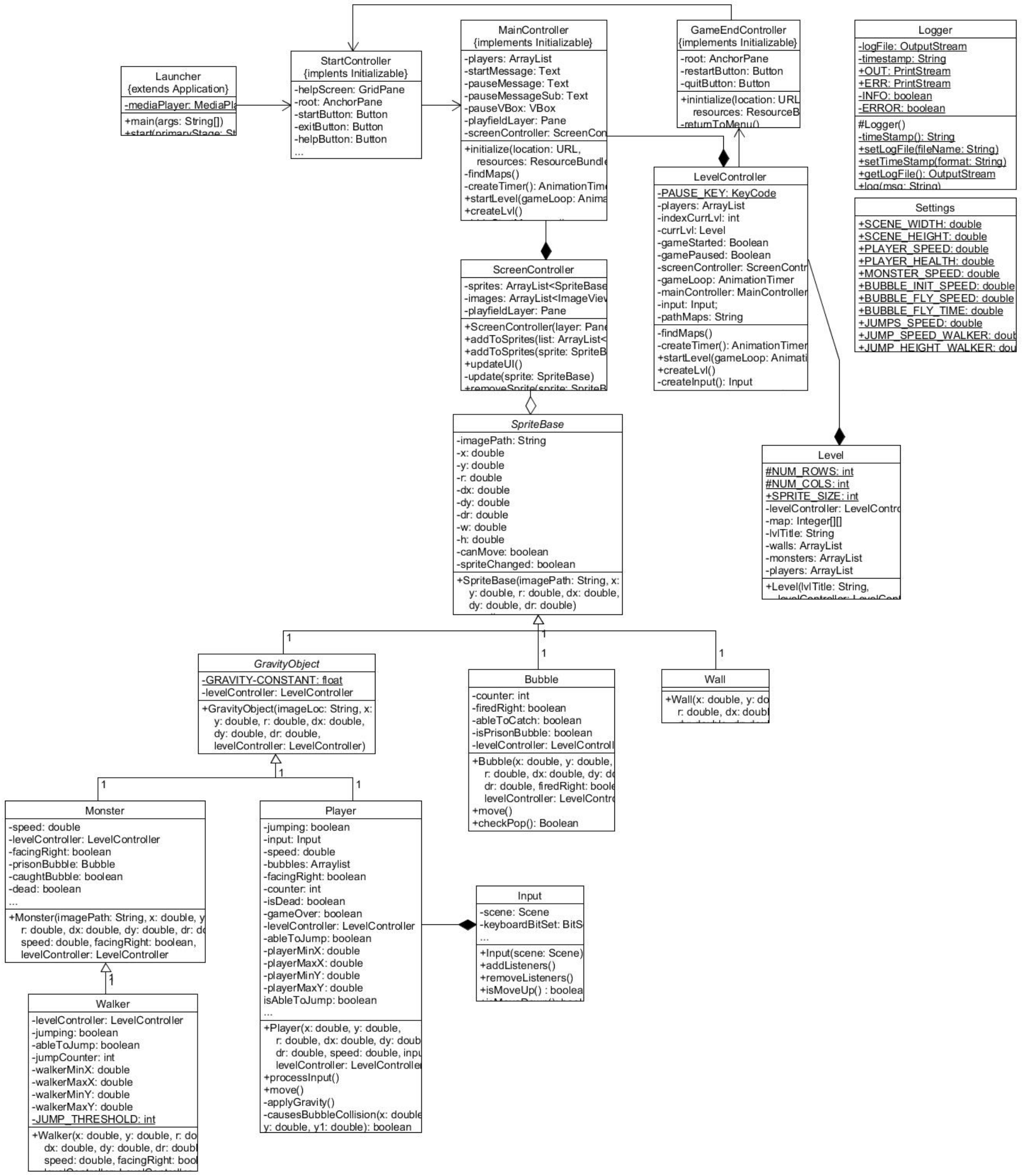
Responsibility:
MainController will be the controller that is loaded by the FXML. It will have all untestable objects and will seperate the GUI from the LevelController.
Collaborations:
LevelController and ScreenController will communicate via MainController.

1.4 Better Physics

Player

Responsibility:
Player is responsible for the movement of the player itself.
Collaborations:
Player will be communicating with LevelController for the set of walls it could collide with. This all affects the physics.

2. Updated UML


Launcher
{extends Application}
-mediaPlayer: MediaPla
+main(args: String[])
StartController
{implents Initializable}
-helpScreen: GridPane
-root: AnchorPane
-startButton: Button
-exitButton: Button
-helpButton: Button
...
MainController
{implements Initializable}
-players: ArrayList
-startMessage: Text
-pauseMessage: Text
-pauseMessageSub: Text
-pauseVBox: VBox
-playfieldLayer: Pane
-screenController: ScreenCon
+initialize(location: URL,
resources: ResourceBundl
-findMaps()
-createTimer(): AnimationTim
+startLevel(gameLoop: Anima
+createLvl()
GameEndController
{implements Initializable}
-root: AnchorPane
-restartButton: Button
-quitButton: Button
+inintialize(location: URL
resources: ResourceB
Logger
-logFile: OutputStream
-timestamp: String
+OUT: PrintStream
+ERR: PrintStream
-INFO: boolean
-ERROR: boolean
#Logger()
-timeStamp(): String
+setLogFile(fileName: String)
+setTimeStamp(format: String)
+getLogFile(): OutputStream
Settings
+SCENE_WIDTH: double
+SCENE_HEIGHT: double
+PLAYER_SPEED: double
+PLAYER_HEALTH: double
+MONSTER_SPEED: double
+BUBBLE_INIT_SPEED: double
+BUBBLE_FLY_SPEED: double
+BUBBLE_FLY_TIME: double
+JUMPS_SPEED: double
+JUMP_SPEED_WALKER: doub
+JUMP_HEIGHT_WALKER: dou
LevelController
-PAUSE_KEY: KeyCode
-players: ArrayList
-indexCurrLvl: int
-currLvl: Level
-gameStarted: Boolean
-gamePaused: Boolean
-screenController: ScreenContr
-gameLoop: AnimationTimer
-mainController: MainController
-input: Input;
-pathMaps: String
-findMaps()
-createTimer(): AnimationTimer
+startLevel(gameLoop: Animati
+createLvl()
-createInput(): Input
ScreenController
-sprites: ArrayList<SpriteBase
-images: ArrayList<ImageView
-playfieldLayer: Pane
+ScreenController(layer: Pane
+addToSprites(list: ArrayList&lt;
+addToSprites(sprite: SpriteB
+updateUI()
-update(sprite: SpriteBase)
SpriteBase
-imagePath: String
-x: double
-y: double
-r: double
-dx: double
-dy: double
-dr: double
-w: double
-h: double
-canMove: boolean
-spriteChanged: boolean
+SpriteBase(imagePath: String, x:
y: double, r: double, dx: double,
dy: double, dr: double)
Level
#NUM_ROWS: int
#NUM_COLS: int
+SPRITE_SIZE: int
-levelController: LevelContro
-map: Integer[][]
-lvlTitle: String
-walls: ArrayList
-monsters: ArrayList
-players: ArrayList
+Level(lvlTitle: String,
1
1
1
1
GravityObject
-GRAVITY-CONSTANT: float
-levelController: LevelController
+GravityObject(imageLoc: String, x:
y: double, r: double, dx: double,
dy: double, dr: double,
levelController: LevelController)
Bubble
-counter: int
-firedRight: boolean
-ableToCatch: boolean
-isPrisonBubble: boolean
-levelController: LevelControll
+Bubble(x: double, y: double,
r: double, dx: double, dy: d
dr: double, firedRight: boole
levelController: LevelContro
+move()
+checkPop(): Boolean
Wall
+Wall(x: double, y: do
r: double, dx: doub
1
1
1
Monster
-speed: double
-levelController: LevelController
-facingRight: boolean
-prisonBubble: Bubble
-caughtBubble: boolean
-dead: boolean
...
+Monster(imagePath: String, x: double, y
r: double, dx: double, dy: double, dr: d
speed: double, facingRight: boolean,
levelController: LevelController
Player
-jumping: boolean
-input: Input
-speed: double
-bubbles: Arraylist
-facingRight: boolean
-counter: int
-isDead: boolean
-gameOver: boolean
-levelController: LevelController
-ableToJump: boolean
-playerMinX: double
-playerMaxX: double
-playerMinY: double
-playerMaxY: double
isAbleToJump: boolean
...
+Player(x: double, y: double,
r: double, dx: double, dy: doub
dr: double, speed: double, inpu
levelController: LevelControlle
+processInput()
+move()
-applyGravity()
-causesBubbleCollision(x: double
y: double, y1: double): boolean
Input
-scene: Scene
-keyboardBitSet: BitS
...
+Input(scene: Scene)
+addListeners()
+removeListeners()
+isMoveUp() : boolea
1
Walker
-levelController: LevelController
-jumping: boolean
-ableToJump: boolean
-jumpCounter: int
-walkerMinX: double
-walkerMaxX: double
-walkerMinY: double
-walkerMaxY: double
-JUMP_THRESHOLD: int
+Walker(x: double, y: double, r: do
dx: double, dy: double, dr: doub
speed: double, facingRight: boo