Sophia's Adventures!

A 2D Maze Game by Mia deLara

Abstract

My project is a two-dimensional visual maze game. There will be a character sprite that can be controlled using the W, S, A, D buttons or the arrow keys.

Introduction

My work was motivated by my previous game design experience. I decided to study computer science after learning how to create my first three-dimensional game in high school. I enjoy both the visual and the technical aspects of gaming, but did not have much knowledge in the programming aspect of it. I also made a text-based game with a few visual aspects in a previous programming class in college.

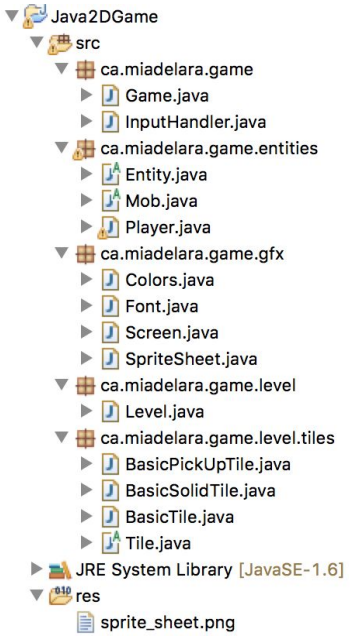
The language I used for my 3D game in high school was javascript and C#. The language I used for my text-based game was Python. I decided to try designing a game in Java to see how that would work out.

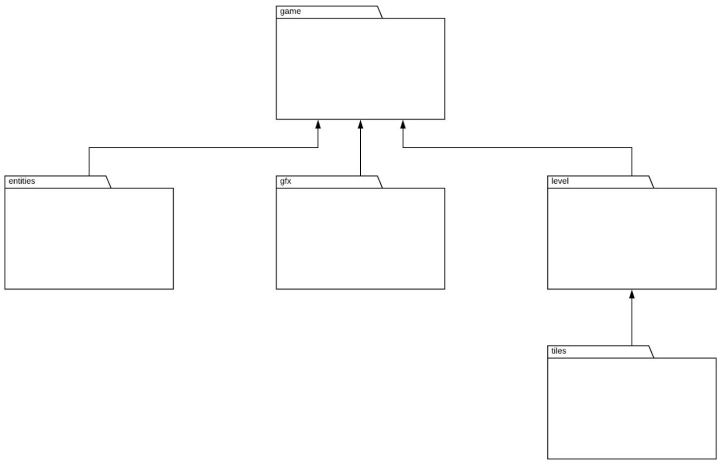
This paper will go over a detailed description of my system (including UML's, user interactions, and what it does), what the system requirement is, how other systems and work have addressed this problem, the user manual, a summary of the goals accomplished, and the references used during this project.

Detailed System Description

The gaming system is used for entertainment and to pass time. Player users interact with the game by controller their character using the W, S, A, D keys or the arrow keys. The system interacts with the player by the InputHandler, which moves the player screen depending on what keys are pressed.

There are fifteen classes among five different packages in the project, and a sprite_sheet that holds the character, font, and tiles in the res folder.





execution environment JRE: JaveSE-1.6

In my main package, the game package, I have the main game class and the basic input handler. The Game class sets up the window in which the game is held, and has the main functions, such as running, rendering, starting, and stopping the game. The Game class refers to all the other classes and is the main class responsible for the game as a whole. The InputHandler class is what handles the user's input (up, down, left, right, W, S, A, D).

The four subpackages include graphics (gfx), entities, level, and tiles (which is a subclass of the level class.

In my graphics package (gfx), I have my Colors, SpriteSheet, Font, and Screen classes. The Color class is self-explanatory and handles how the colors will appear on the screen. The SpriteSheet class is responsible for locating and rendering the sprite_sheet.png file. The font class reads the font in the sprite_sheet.png file and assigns the characters to the font characters. The Screen class sets up the whole screen dimensions and displays the sprites, colors and fonts.

The entities package carries the abstract Entity class, which is extended by the abstract Mob class, which is extended by the Player class. The entity class is self explanatory. It has the x and y coordinates of the entity and the level in which it appears. The Mob class carries the name, speed, number of steps, moving boolean, and scale variable. The Player class holds the color and scale variable.

The level package contains the Level class, which is responsible for generating the level. This is where I created a 2D-array for the maze. The level package is also the parent package of the tiles package, which carries the abstract Tile class, the BasicTile class, the BasicSolidTile class, and the BasicPickUpTile class. The Tile class is referenced by the BasicTile class, and the BasicSolidTile class and BasicPickUpTile class both reference the BasicTile class.

Both the BasicSolidTile class and BasicPickUpTile class are basically the exact same thing at the moment, but it is planned to make the tiles that are an instance of the BasicPickUpTile class, these tiles will be named ROCK, change into a tile that is an instance of the BasicTile class, these are called GRASS, to make it look as if the player picked up the rock and added points to the players score.

Requirements

Functional:

Game package - main package in charge of managing all other packages and classes. Ultimately brings them all together to make the game as a whole work. Includes the Game class and the InputHandler class

Graphic effects - display font, screen window, sprites, and colors

Entities - include player display, movement, collision detection.

Game level - generate entire game level with the use of the tile class and screen class.

Tiles - determine if a tile is a solid, basic, pickup, or null tile.

Nonfunctional:

Usability - ensure the user can actually play and use the game with the use of the InputHandler

Speed - adjust the frames per second (fps) the game generates and refreshes with the Game class.

Literature Survey

There are many systems and pieces of work that have addressed the problem of entertainment. Many address it by a social media system or another gaming system in a different genre or type.

Television also addresses entertainment. Though cable is becoming less popular, many companies have streaming and rental systems that allow users to rent or watch a show or movie with a subscription.

Music streaming services are another entertainment service, though many listen to music while playing a game or using social media. Many games, shows, and movies also incorporate music into their mediums.

User Manual

The instructions for using this system is quite simple. The game is to be used for entertainment purposes, and may be played using the controls W, A, S, D or the arrow keys.

Conclusion

The current goals accomplished by this system include the ability to handle input by the user, display text, font, colors, tiles, and the player, creating box colliders and sprites, and running the game in a new window.

A lot has been accomplished for this project, but there is always room for improvement and growth. For example, the pickup tiles do not work correctly as pickup tiles at the moment and

behave primarily as solid tiles rather than tiles that disappear. Player score and health can also be implemented. Points can be added to the score when rocks are picked up, and another tile and tile class could be created to lower the player's health.

Eventually, in time, the game will implement and include these stated features and may have more than just one level.

References

[Ryan van Zeben]. (2012, August 30). *Setting up the JFrame* [Video File]. Retrieved from https://www.youtube.com/watch?v=VE7ezYCTPe4&list=PL8CAB66181A502179&index=1.

[Ryan van Zeben]. (2012, August 31). *Creating a Sprite Sheet* [Video File]. Retrieved from https://www.youtube.com/watch?v=o7pfq0W3e4I&index=2&list=PL8CAB66181A502179.

[Ryan van Zeben]. (2012, September 1). *Displaying the Sprite Sheet* [Video File]. Retrieved from https://www.youtube.com/watch?v=6FMgQNDNMJc&list=PL8CAB66181A502179&index=3.

[Ryan van Zeben]. (2012, September 2). *Basic Input Handling* [Video File]. Retrieved from https://www.youtube.com/watch?v=Vv7G5GMOre8&index=4&list=PL8CAB66181A502179.

[Ryan van Zeben]. (2012, September 3). *Colour & Rendering Optimisation* [Video File]. Retrieved from https://www.youtube.com/watch?v=7eotyB7oNHE&index=5&list=PL8CAB66181A502179.

[Ryan van Zeben]. (2012, September 4). *Mirroring the Sprites* [Video File]. Retrieved from https://www.youtube.com/watch?v=gWSx7S8YiPQ&list=PL8CAB66181A502179&index=5.

[Ryan van Zeben]. (2012, September 6). *Rendering some Fonts* [Video File]. Retrieved from https://www.youtube.com/watch?v=XSBTWhV75hM&list=PL8CAB66181A502179&index=7.

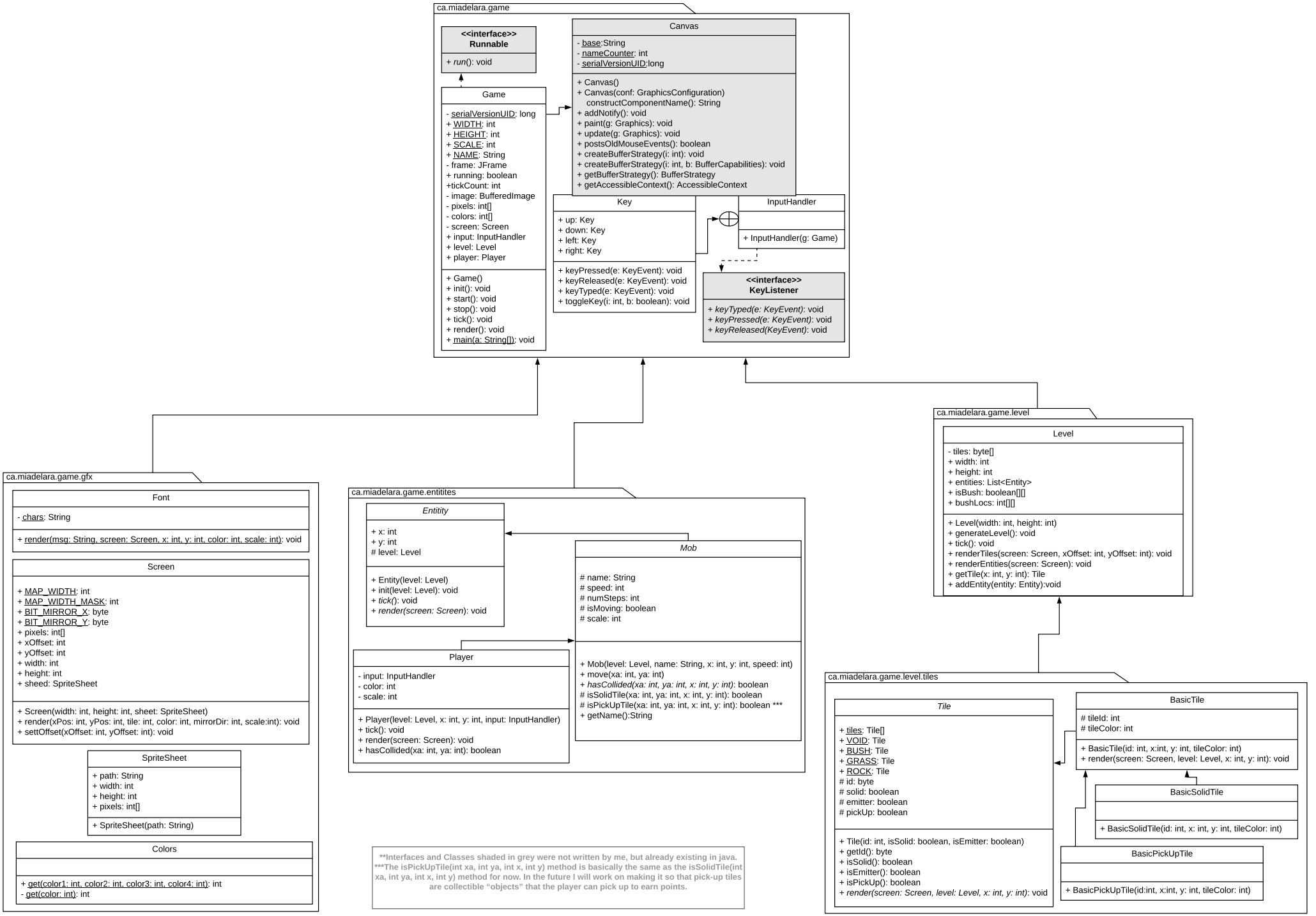
[Ryan van Zeben]. (2012, September 9). *Level/Tile Development* [Video File]. Retrieved from https://www.youtube.com/watch?v=aDZemglb2WI&index=7&list=PL8CAB66181A502179.

[Ryan van Zeben]. (2012, September 25). *Making our first mob, the Player!* [Video File]. Retrieved from https://www.youtube.com/watch?v=dQP7ZmFhqgg&index=8&list=PL8CAB66181A502179.

[Ryan van Zeben]. (2012, October 1). *Scaling Things up!* [Video File]. Retrieved from https://www.youtube.com/watch?v=Xb4NK9B631E&index=9&list=PL8CAB66181A502179.

[Ryan van Zeben]. (2012, October 10). *The Walking Animation!* [Video File]. Retrieved from https://www.youtube.com/watch?v=dnvNcjVNqqs&list=PL8CAB66181A502179&index=10.

[Ryan van Zeben]. (2012, October 11). *The Collision Box* [Video File]. Retrieved from https://www.youtube.com/watch?v=NtXDuJHLxRs&list=PL8CAB66181A502179&index=11.


ca.miadelara.game
<<interface>>
Runnable
+ run(): void
Game
- serialVersionUID: long
+ WIDTH: int
+ HEIGHT: int
+ SCALE: int
+ NAME: String
- frame: JFrame
+ running: boolean
+tickCount: int
- image: BufferedImage
- pixels: int[]
- colors: int[]
- screen: Screen
+ input: InputHandler
+ level: Level
+ player: Player
+ Game()
+ init(): void
+ start(): void
+ stop(): void
+ tick(): void
+ render(): void
+ main(a: String[]): void
Canvas
- base:String
- nameCounter: int
- serialVersionUID:long
+ Canvas()
+ Canvas(conf: GraphicsConfiguration)
constructComponentName(): String
+ addNotify(): void
+ paint(g: Graphics): void
+ update(g: Graphics): void
+ postsOldMouseEvents(): boolean
+ createBufferStrategy(i: int): void
+ createBufferStrategy(i: int, b: BufferCapabilities): void
+ getBufferStrategy(): BufferStrategy
+ getAccessibleContext(): AccessibleContext
Key
+ up: Key
+ down: Key
+ left: Key
+ right: Key
+ keyPressed(e: KeyEvent): void
+ keyReleased(e: KeyEvent): void
+ keyTyped(e: KeyEvent): void
+ toggleKey(i: int, b: boolean): void
InputHandler
+ InputHandler(g: Game)
<<interface>>
KeyListener
+ keyTyped(e: KeyEvent): void
+ keyPressed(e: KeyEvent): void
+ keyReleased(KeyEvent): void
ca.miadelara.game.level
Level
- tiles: byte[]
+ width: int
+ height: int
+ entities: List<Entity>
+ isBush: boolean[][]
+ bushLocs: int[][]
+ Level(width: int, height: int)
+ generateLevel(): void
+ tick(): void
+ renderTiles(screen: Screen, xOffset: int, yOffset: int): void
+ renderEntities(screen: Screen): void
+ getTile(x: int, y: int): Tile
+ addEntity(entity: Entity):void
ca.miadelara.game.gfx
Font
- chars: String
+ render(msg: String, screen: Screen, x: int, y: int, color: int, scale: int): void
Screen
+ MAP_WIDTH: int
+ MAP_WIDTH_MASK: int
+ BIT_MIRROR_X: byte
+ BIT_MIRROR_Y: byte
+ pixels: int[]
+ xOffset: int
+ yOffset: int
+ width: int
+ height: int
+ sheed: SpriteSheet
+ Screen(width: int, height: int, sheet: SpriteSheet)
+ render(xPos: int, yPos: int, tile: int, color: int, mirrorDir: int, scale:int): void
+ settOffset(xOffset: int, yOffset: int): void
SpriteSheet
+ path: String
+ width: int
+ height: int
+ pixels: int[]
+ SpriteSheet(path: String)
Colors
+ get(color1: int, color2: int, color3: int, color4: int): int
- get(color: int): int
ca.miadelara.game.entitites
Entitity
+ x: int
+ y: int
# level: Level
+ Entity(level: Level)
+ init(level: Level): void
+ tick(): void
+ render(screen: Screen): void
Mob
name: String
speed: int
numSteps: int
isMoving: boolean
scale: int
+ Mob(level: Level, name: String, x: int, y: int, speed: int)
+ move(xa: int, ya: int)
+ hasCollided(xa: int, ya: int, x: int, y: int): boolean
# isSolidTile(xa: int, ya: int, x: int, y: int): boolean
# isPickUpTile(xa: int, ya: int, x: int, y: int): boolean ***
+ getName():String
Player
- input: InputHandler
- color: int
- scale: int
+ Player(level: Level, x: int, y: int, input: InputHandler)
+ tick(): void
+ render(screen: Screen): void
+ hasCollided(xa: int, ya: int): boolean
ca.miadelara.game.level.tiles
Tile
+ tiles: Tile[]
+ VOID: Tile
+ BUSH: Tile
+ GRASS: Tile
+ ROCK: Tile
# id: byte
# solid: boolean
# emitter: boolean
# pickUp: boolean
+ Tile(id: int, isSolid: boolean, isEmitter: boolean)
+ getId(): byte
+ isSolid(): boolean
+ isEmitter(): boolean
+ isPickUp(): boolean
+ render(screen: Screen, level: Level, x: int, y: int): void
BasicTile
tileId: int
tileColor: int
+ BasicTile(id: int, x:int, y: int, tileColor: int)
+ render(screen: Screen, level: Level, x: int, y: int): void
BasicSolidTile
+ BasicSolidTile(id: int, x: int, y: int, tileColor: int)
BasicPickUpTile
+ BasicPickUpTile(id:int, x:int, y: int, tileColor: int)
**Interfaces and Classes shaded in grey were not written by me, but already existing in java.
***The isPickUpTile(int xa, int ya, int x, int y) method is basically the same as the isSolidTile(int xa, int ya, int x, int y) method for now. In the future I will work on making it so that pick-up tiles are collectible “objects” that the player can pick up to earn points.