Title? 19/20

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.

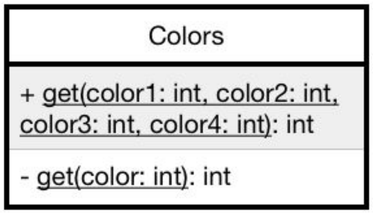
This paper will go over a detailed description of my system (including UML's, user interactions, and what it does), what the system requirement is, how other systems and work have addressed this problem, the user manual, a summary of the goals accomplished, and the references used during this project.

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 currently five classes split among two different packages in the project. In my graphics package, I have my Colors, SpriteSheet, and Screen classes. The Color class is self-explanatory and handles how the colors will appear on the screen. The SpriteSheet class holds a sprite sheet designed in Photoshop and Gimp, and the Screen class sets up the whole screen dimensions and displays the sprites and the colors.

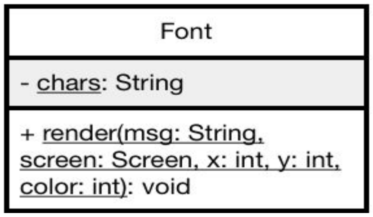
In my main package, the game package, I have the main game class and the basic input handler. The Game class sets up the window in which the game is held, and has the main functions, such as running, rendering, starting, and stopping the game. The Game class refers to all the other classes. The InputHandler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



Colors

+ get(color1: int, color2: int, color3: int, color4: int): int

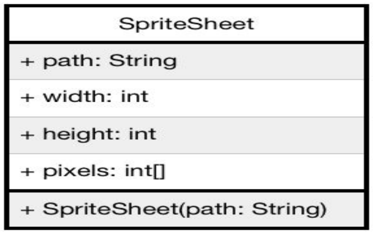
- get(color: int): int



Font

- chars: String

+ render(msg: String, screen: Screen, x: int, y: int, color: int): void



SpriteSheet

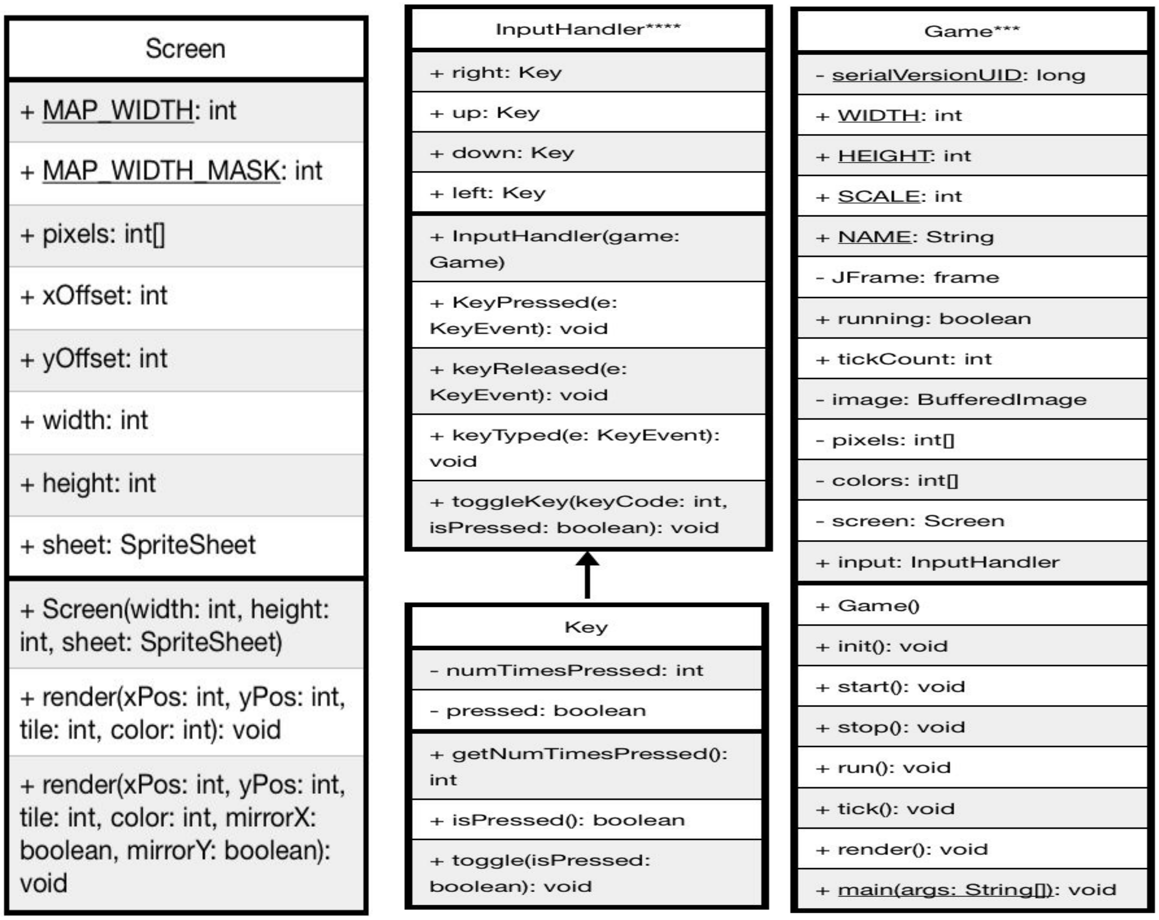
+ path: String

+ width: int

+ height: int

+ pixels: int[]

+ SpriteSheet(path: String)



(*Description is only of what is done up to date. Future diagrams and descriptions of other classes will be presented later.*)(***Game extends Canvas implements Runnable***)(****implements KeyListener****)

Requirements

The purpose of the system is to provide entertainment.

This section describes the requirements you are implementing for this specific game.

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, and colors, and running the game in a new window.

There is still a lot to accomplish. For example, the character sprite still needs to be created, and the maze and box colliders need to be rendered.

Eventually, if there is time, the game may also become a multiplayer platform.

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&index=6&list=PL8CAB66181A502179.

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

(**current references are only those that have been used in the game so far. Further references will follow**).