Brandon Carlsen, David Lichliter, Michael Zamora

CIS 271

4/23/2018

Project Paper

Our team successfully wrote a game engine that we can use to write a full platformer game in the future. Our engine uses 18 different java classes that consists of 2,331 lines of code and implements every concept that we have learned over two semesters of Java classes and some techniques learned through research. This game engine is designed to allow a developer to create their own platform video game. Below is a brief description of our classes and what they do.

* **Game:** Our main method is in the Game class; it creates a thread that runs the game class in sequential order.
* **Animated Sprite:** This class takes the image of our main character and animates it in place to make it appear as if it is moving. This implements a Game Object which requires a render, update, and mouse listener method.
* **Game Object:** An abstract class that creates an Interface for several different classes. Requires all classes that implement the class to include render, update and mouse listener methods.
* **GUI:** This class takes an array of buttons and provides the user a set of buttons that they can use to edit their platformer.
* **GUIButtons:** This class defines each of the buttons for the GUI class and allows a developer to set their buttons to given tiles.
* **SDKButton:** determines the style of the editing buttons and updates so that they appear remain on the screen for editing.
* **KeyboardListener:** this is a basic keyboard listener that allows the program to recognize user input.
* **MouseEventListener:** this is a basic mouse listener in order for a developer to create the levels by putting tiles into the JFrame.
* **Map:** This class renders the background images and the player avatar. This class also handles collisions by establishing a rectangle around the player and checks when the rectangle intersects with other map tiles like platforms. To do this it utilizes a strut class called block. It also has a MapTile strut class to allow images to layer.
* **Player:** this class constantly updates the position and status of your player. There is a key listener within the class to allow your player to move about the level. The player is performing x and y collision checks to make sure the character doesn’t fall through the floor.
* **Sprite:** an instance of a sprite is created and animated using the sprite sheet associated with it. The sheet is defined in the SpriteSheet class.
* **SpriteSheet:** class pulls the players sprite sheet image which is used to determine the animation needed when the player’s avatar face left, right, or jumps.
* **Sound:** This class pulls audio files from the assets folder to provide background music and sound effects for jumping.
* **Tiles:** Describes how to draw a given tile into the JFrame one line of pixels at a time. Pulls the tile choice from the tiles text document.
* **RenderHandler:** Tells the program how to draw an image pulled from a file and renders the image offscreen into the memory before the image makes it onto the Jframe. This makes the image appear much smoother.
* **Vector2:** This class is named Vector2 because there is already a Vector class in Java with many different methods that are more complex than the vector we needed for this game. The Vector2 class defines our gravity and defines our jump path.
* **Rectangle:** Constructs a rectangle that is used to determine window size, player objects, tiles, and collision boxes around players and tiles used in other classes.
* **StartButton:** extension of the GUIButtons class which defines the start button at the title screen.

The goal of the project was to create a game engine which would give us a firm foundation from which we can build a full, story-driven platform game. David, Michael and I hope to continue working on this project throughout the summer with the goal being to sell the game on the steam platform.

The code underwent various stages and versions; ranging from more of a top down to the finalized side scrolling platformer. Due to people in the group editing and rewriting various sections of code in classes, sometimes in different parts that were written by a different group member, it is difficult to list everything everyone in the group wrote. So, instead of listing every change made to the code by each person over the last four months, listed under are the classes that each person originally wrote.

Michael- Game, Renderhandler, Sprite, SpriteSheet, AnimatedSprite, Map

Brandon- Rectangle, GUI, GUIButtons, SDKButton, StartButton, GameObject

David- Sound, Player, Vector2, KeyboardListener, MouseEventListener, Tiles

Operating the game can be done with the following controls:

* Click the start button to begin the game
* A or the left arrow key to move the character left
* D or the right arrow key to move the character right
* W or the up arrow key to jump
* Left click the GUI to the left to select a tile
  + Left clicking the map will place the tile
    - Some tiles have collision, like the top of the tower tiles
    - Some do not, like the generic stone tile
  + Right clicking a tile removes it from the map
* Control + S allows you to save the map you have edited for future use

*The Window is also resizable if the viewing area is not to your liking. (DO NOT FULLSCREEN)*