Game Engine & Space Shooter

Jaime Tijerina

https://github.com/Tijeras94/CMPE_3326_Final_Project

The Plan

- Create one or two games
- Create an abstraction layer for rendering images to the screen and utilizing the mouse and keyboard
- Try to make everything modular
  - This is very important for us because we were able to split the workload for every member of the team
  - It's easier to extend the game to different platforms such as Android
- Maintainable
  - For example, if you build a game and decide to make changes to the render system, you don't have to do anything on your part

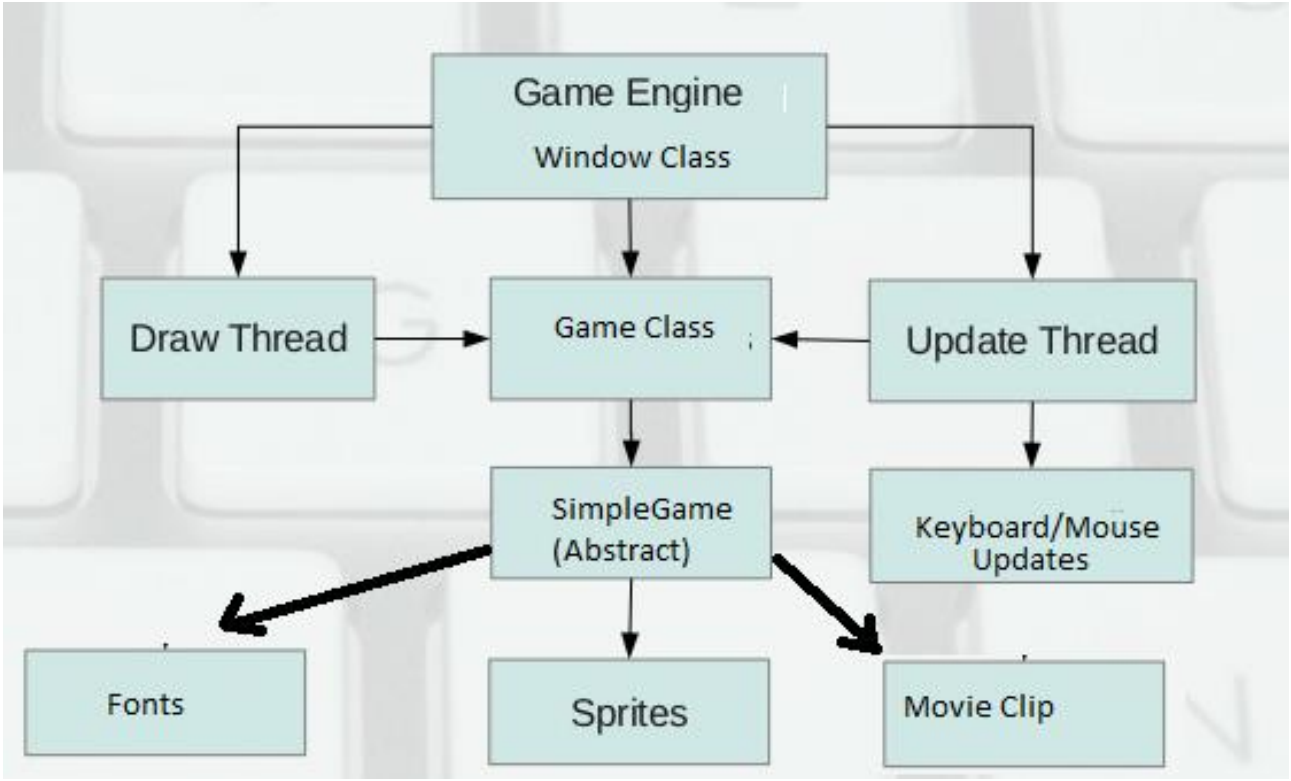
Requirements

- Must be in Java
- No external libraries
- Must be modular
- Must be maintainable

What is a Game Engine?

- Engines offer reusable components that can be manipulated to bring a game to life.
    - Ex: Loading, displaying, and animating models, collision detection between objects, physics, input, graphical user interfaces

Engine Architecture



SimpleGame (Abstract)

- This is used as an entry point for our games
- Must Override Methods
  - public boolean update()
  - public void draw(Graphics2D g)

Creating a Game

- Entry Point (Main Function)
  - Window win = new Window("Demo2", 800 ,480, false);
  -
  - win.start(60, Demo2.class);
    - These function initialize the Game Engine (Keyboard, Mouse, Sound)
      - Params
        - Fps => Frames Per Second
        - Class Object(It Extends From SimpleGame)
- There must be only one Window instance in the entire Program

Game Class

- It must extend from SimpleGame
- SimpleGame has two useful functions
  - Draw(Graphics2D g) => Render Images using the Global Graphcs2D
  - Update() => It gets called depending the FPS

Adding Sprites To The Stage

- Sprite:
  - a **sprite** is a two-dimensional bitmap that is integrated into a larger scene. aka Image
- Sprite m1 = new Sprite("/Assets/player.png")
- Display Image to Screen
  - Call => m1.draw(g) in the Update Method
- Delete Sprite
  - m1.destroy(); // Destroys Memory Asset

Moving a Sprite

Move Left/Right:

```
m1.x += speed; or m1.x -= x
```

Move Up/Down:

```
m1.y += speed; or m1.y -= y
```

Checking Collusion:

```
if(m1.rectOverlap(m2))
        //HIT
```

Sprite Animation

- We need a Texture Atlas

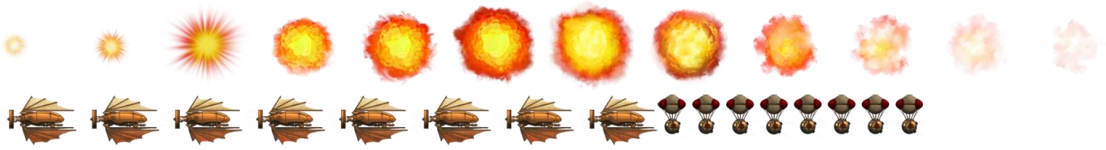
  Texture Atlas: In realtime computer graphics, a texture atlas (also called a tile map, tile engine, or sprite sheet) is a large image containing a collection, or "atlas", of sub-images, each of which is a texture map for some part of a 2D or 3D model.

Init An Atlas

- TextureAtlas attlas = new Texture Atlas("spaceArt/mySpritesheet.xml");
  - We Specify a XML (Config File) mySpritesheet.xml

  Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<TextureAtlas imagePath="sprites.png">
    <SubTexture name="ship_0000" x="1" y="137" width="115" height="69"/>
    <SubTexture name="ship_0001" x="116" y="137" width="115" height="69"/>
    <SubTexture name="ship_0002" x="231" y="137" width="115" height="69"/>
    <SubTexture name="ship_0003" x="346" y="137" width="115" height="69"/>
    <SubTexture name="ship_0004" x="461" y="137" width="115" height="69"/>
    <SubTexture name="ship_0005" x="576" y="137" width="115" height="69"/>
    <SubTexture name="ship_0006" x="691" y="137" width="115" height="69"/>
    <SubTexture name="ship_0007" x="806" y="137" width="115" height="69"/>
</TextureAtlas>
```



Getting Assets From Atlas

- MovieClip enemy =attlas.getMovieClip("fly_");
- Sprite player = attlas.getSprite(“player”);

MovieClip

- MovieClip extends From Sprite
- MovieClip=> A Sequence(SubTexture) of images in order that loop (based on the fps) to create animation
- MovieClip enemy = atlas.getMovieClip("fly_");
- Or
- Sprite enemy = atlas.getMovieClip("fly_");

Audio Class

Init An Audio

```
AudioPlayer musicSound = new
 AudioPlayer("Demo2Assets/Sound/gameMusi
 c.wav");

musicSound.stop(); // Stop Sound
musicSound.play(); // Playing a Sound
```

Font Class

- Init a font

Font text = new Font("spaceArt/GoodDog.ttf");

Params

1=> Font location

- Create a String Text

text.draw(“Score : 10”, 10, 20, 16, g);

Params

1 => Text to Draw

2 -> X position

3 -> Y position

4 -> Graphics2D Object (Found in the update function)

Keyboard And Mouse

```
Keyboard kb = Keyboard.getInstance();
if (kb.isKeyPress(KeyEvent.VK_LEFT))
  //if LEFT Key IS Press
Mouse ms =MouseManager.getInstance();
if (ms.isKeyPress(MouseManager.M_CLICK_RIGHT))
        //if Mouse Rigth Key IS Press
ms.getMouseX(); // Getting Cordinates
ms.getMouseY();
```

Parallax Scrolling

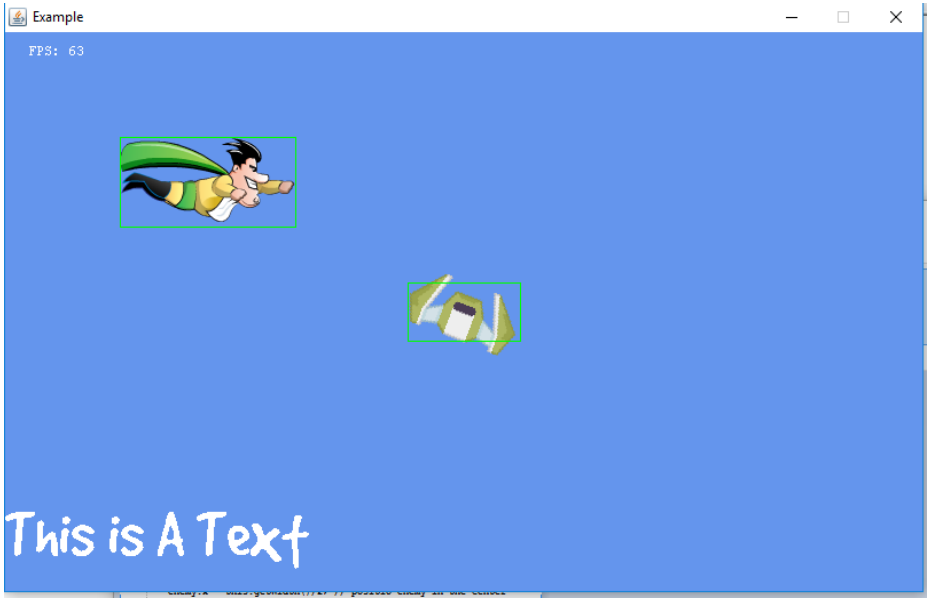
Helper Class ParallaxHSprite Class

Parallax scrolling is a technique in computer graphics and web design, where background images move by the camera slower than foreground images, creating an illusion of depth in a 2D scene and adding to the immersion.

```
ParallaxHSprite bgLayer1 = new ParallaxHSprite("Demo2Assets/bgLayer1.png", 5); // image path , scrolling speed
ParallaxHSprite bgLayer2 = new ParallaxHSprite("Demo2Assets/bgLayer1.png", 10);
```

Example

```
public class Example extends SimpleGame {
    private int speed = 10;
    private Font text;
    private TextureAtlas attlas;
    private AudioPlayer music;
    private Sprite player;
    private Sprite enemy;

    public Example(int width, int height) {

        super(width, height);

        //Debug
        Window.DEBUG = true;

        //Create Music
        music = new AudioPlayer("spaceArt/music.wav");
        //Create A Font
        text = new Font("spaceArt/GoodDog.ttf");
        //Create an Atlas
        attlas = new TextureAtlas("spaceArt/mySpritesheet.xml");

        //Creting the Sprite Images
        player = attlas.getMovieClip("fly_");
        enemy = new Sprite("spaceArt/enemyShip.png");

        enemy.x = this.getWidth()/2; // positio enemy in the center
        enemy.y = this.getHeight()/2;

        music.setVolume(0.1f); // set volume (0.0 to 1.0) (loudest)
        music.play();//start playing
    }

    @Override
    public void draw(Graphics2D g) {
        //draw sprites and text to screen
        player.draw(g);
        enemy.draw(g);
        text.draw("This is A Text", 0, this.getHeight() - 60, 60, g);
    }

    @Override
    public boolean update() {
        this.enemy.rotation += 1;//rotate enemy
        player.update();//update the animation of the sprite
        if (Keyboard.getInstance().isKeyPress(KeyEvent.VK_UP)) {
            player.y -= speed;
        }
        if (Keyboard.getInstance().isKeyPress(KeyEvent.VK_DOWN)) {
            player.y += speed;
        }
        if (Keyboard.getInstance().isKeyPress(KeyEvent.VK_LEFT)) {
            player.x -= speed;
        }
        if (Keyboard.getInstance().isKeyPress(KeyEvent.VK_RIGHT)) {
            player.x += speed;
        }
        //make player to move inside an imaginary box
        player.walkWithinBox(0, 0, this.getWidth(), this.getHeight());
        return true; // must always return true
    }
```




```
package mygame;

import mygame.demos.demo2.Demo2;
import mygame.engine.Window;

public class Main {

    public static void main(String[] args) {

        Window win = new Window("Example", 800, 480, false);
        win.start(60, Example.class);
    }

}
```

Potential Fixes

Fix the rotation functionality for sprites

Adding a 2d Physics Engine

Particle System

Scene Manager

Tween System

Scoreboard System (Using MySQL)

Add a Renderer for Android Devices

Add Touch Support