SPL-1 Project Report, 2019

Thakurmar Jhuli: An Action Adventure Game

SE 306: Software Project Lab I

Submitted by

Syed Ahmedul Kavi

BSSE Roll No. : 1017

BSSE Session: 2017-2018

Supervised by

Dr. Kazi Muhaeymin-Us-Sakib

Designation: Professor

Institute of Information Technology

Institute of Information Technology

University of Dhaka

29-05-2019

Table of Contents

1. Introduction

My project is about making an action adventure game based on Thakurmar Jhuli.Thakurmar Jhuli has been a part of our culture since our childhood. There is almost none who haven't read the marvelous fables based on the ancient mythologies of the subcontinent where virtuous always thrived and evil fell to the clutches of despair in the long run. Thakurmar Jhuli, even being a part and parcel of our culture was never seen in video games. So I decided to relive our childhood fantasy into this 2D action adventure game.

Game is a form of entertainment and action adventure is one of the popular genres. As I could perfectly depict the Thakurmar Jhuli culture in video game, I chose this medium. The game is made using the language Java and for graphics rendering JavaFX library.

1.1. Background Study

The important concepts that my game uses are:

- Main Game Loop
- Game Sprites
- Rendering Animation
- Input Handling
- Seperation of Concern

Main Game Loop

Each game has an infinite loop running until the game is closed or crashed, in this loop all game actions like graphics rendering and Input handling occurs.

Game Sprites

"Sprite is a computer graphics term for a two-dimensional bitmap that is integrated into a larger scene, most often in a 2D video game.

Use of the term *sprite* has expanded to refer to any two-dimensional bitmap used as part of a graphics display, even if drawn into a frame buffer (by either software or a GPU) instead of being composited on-the-fly at display time."

(Source: Sprite (computer graphics), Wikipedia)

Here I used Sprite to define the same 2D character model with attributes like XY coordinate in the scene and height and width, the images for the characters were separate and handled by Graphics Engine.

Rendering Animation

Animation rendering means displaying images and image motions in the graphical display (JavaFX Stage and Scene here). As proper graphic rendering using raw code is beyond the scope of my project, I took aid from the GraphicsContext2D class of JavaFx library to render my images.

But handling the graphics was not totally credit to library. As depending on distance, either the position of hero changes or the hero stands still and the background moves. This is due to the size of the screen being fixed and background transition creates sense of motion and fluidity.
Also from the theory of relativity of Einstein, moving forward and all of background moving backward is same thing. This principle is applied here.

Input Handling

Here the game is keyboard based. The interesting fact about input is that the input is neither from console, nor from JavaFx text field. Rather the inputs are pressed when a scene displays and it is captured from the source of the stage.

Input can be of various types, like key press (tap) or holding. There is no direct library function for single press. It needed to be implemented

Separation Of Concerns

"In computer science, **separation of concerns** (**SoC**) is a design principle for separating a computer program into distinct sections, so that each section addresses a separate concern."

(Source: Separation of Concerns, Wikipedia)

Separation of concerns is used in my project through the independence of the components : Graphics Engine, Sound Engine, Input Engine, Sprites etc. Though Interface is not used here, the philosophy is followed.

1.2. Challenges

Making a game single handedly requires facing lots of challenges as in many cases ad hoc solutions need to be implemented due to lack of proper guidelines for game development using JavaFX in internet. The challenges I faced and how I overcame them are given below:

- As I tried modularizing every components of gaming into independent entity, reducing their dependency to each other to minimum was necessary. This was a huge challenge. I overcame it by making an organizer class 'GameEngine' which will run the main game loop, 'GraphicsEngine' will render the images provided to it in the appropriate XY coordinate provided to it by respective sprite classes which is implemented via a Map data structure, the 'InputEngine' class will solely be responsible for taking inputs and updating Sprites data or signaling GameEngine for any particular instruction. Likewise other components are independent.
- Making .exe or runnable .jar for the first time was quite challenging. It requires th resource path to be given in a specific syntax. I used a static class 'ResourceLoader'

who will search in the resource folder given a string as parameter and return an InputStream.

- ❖ Level transition made me face several challenge like stopping background music and proper respawning. Challenge was overcome with intense debugging.
- ❖ Pausing was a great challenge. As I used 'AnimationTimer' of JavaFX as my main game loop, there was no proper instruction in the Oracle manual for proper pausing and resuming. Ad Hoc approach was taken to make it possible.

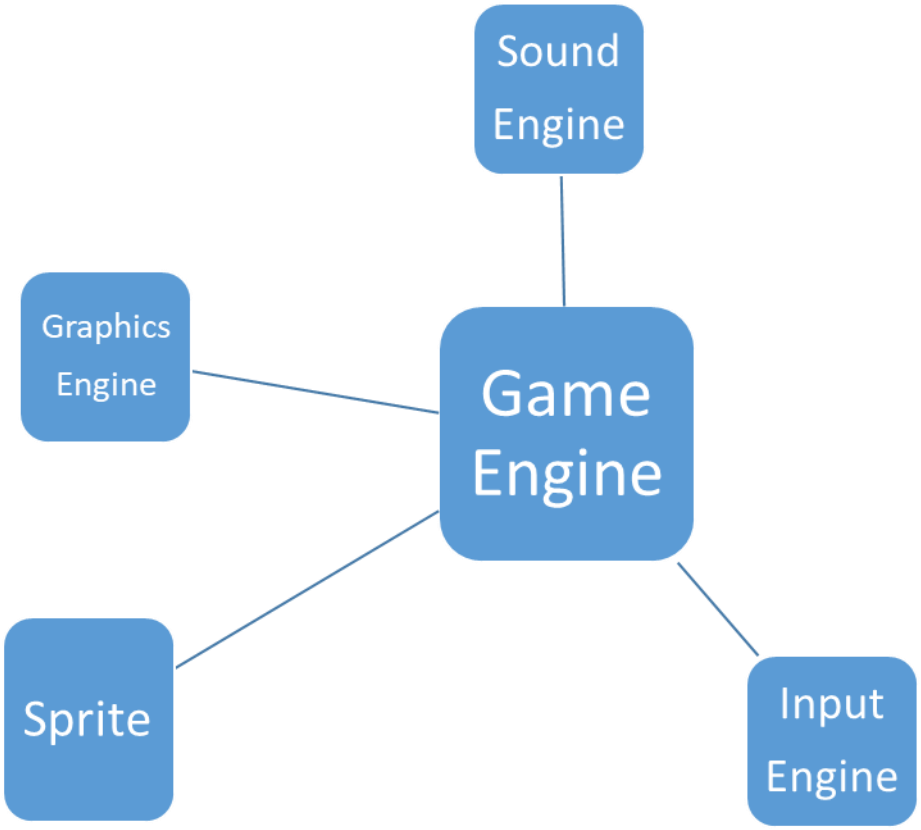
2. Project Overview

My project is dived into several parts. Namely:

- Game Engine
- Sprites
- Graphics Engine
- Input Engine
- Sound Engine
- Story Engine

2.1. Game Engine

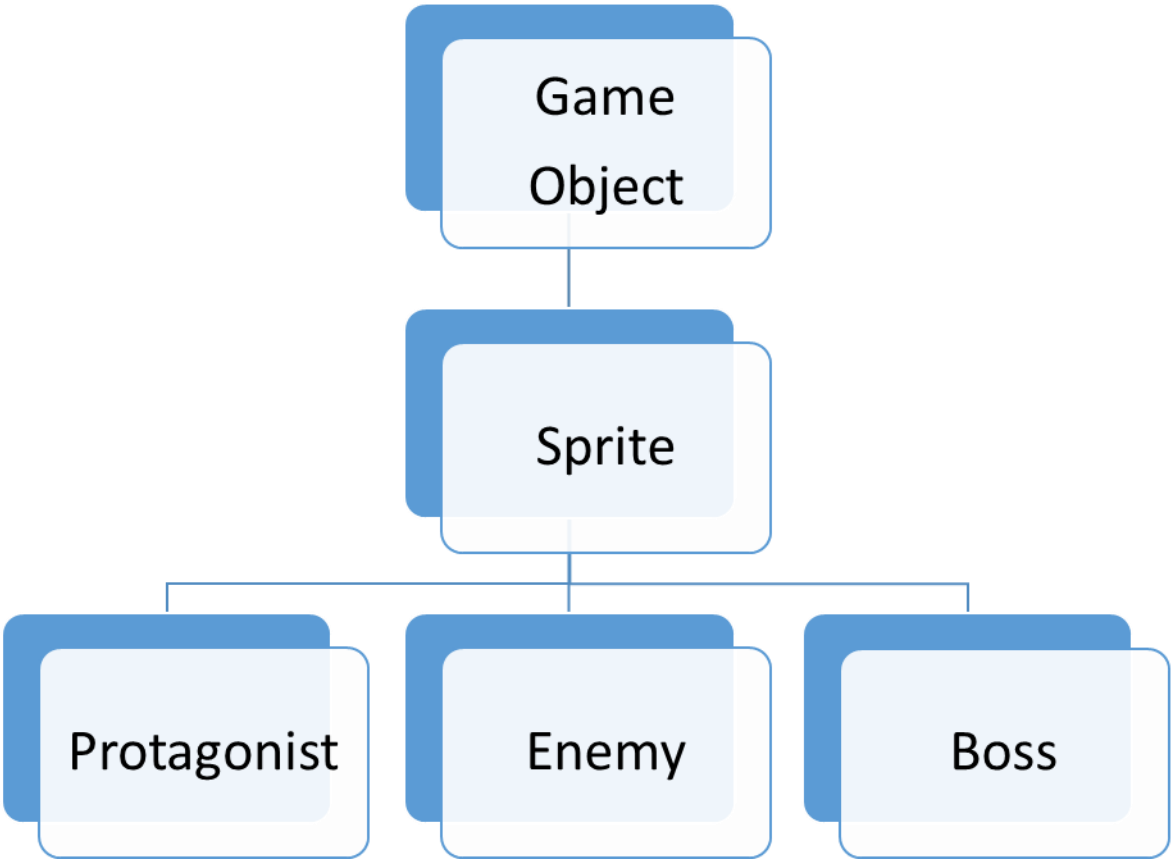
Game Engine contains the main game loop, the objects of protagonist, enemies and bosses. Game Engine contains objects of other components like Graphics Engine, Sound Engine, Input Engine etc. except Story Engine. JavaFX animation timer is used as main game loop where if the game is not paused, graphics is rendered, input is taken and input specific action occurs. However if the game is paused, input is taken to check whereas command resuming is given, but other activities like graphics rendering and enemy movement is paused. The Game Engine literally has the game characters and makes them disappear when they die, reloads the background and everything if the main character dies.



2.2 Sprites

Sprite means in game characters (Both main character and enemies). Sprite class contains XY coordinates in the FX scene, height and width of the object. Sprite class inherits into Protagonist, Enemy and Boss.
All sprites have a health bar. If the health bar reaches 0, the character dies.
Protagonist is the character we control, thus it has some other attributes like distance which indicates the position of hero in the game background, hero has stamina which depletes during attacking and blocking and regains during inactivity. Hero also has Focus which is gained by attacking enemies and a full focus bar can be consumed to regain health.

Enemies and Bosses patrol in a provided radius and will attack hero if he is within range.



2.3. Graphics Engine

Graphics Engine is responsible for all graphics rendering.
Graphics Engine mainly contains a map between Sprite and Views. Sprite contains the XY coordinate in the scene as mentioned before and Views is a class which contains all resource images in various status (STATUS is a declared enum for various status of the sprites, like right facing, attacking etc.)

Graphics Engine takes appropriate coordinate from Sprite class and renders the graphics using draw method of GraphicsContext2D.

2.4. Input Engine

Input Engine has two main functions: taking input and perform required action.
A map is kept for string key code as key and a Boolean as value. The Boolean indicates whether the key is pressed or not.
There are certain scenarios like movement where holding right key moves character but also some situation where holding a button does no good but instead tapping multiple times does (e.g. tapping control for attacking). These cases were handled using map and the function removeActiveKey for tapping where it checks if the Boolean value is false only then it registers an action and makes the value false again where in case of holding the key, simple map.contains() is enough to check.

2.5. Sound Engine

Sound Engine is responsible for playing all background music and transitioning music in appropriate scenario stopping the previous playing.
The music is played using JavaFX media player.

2.6. Story Engine

This component reads from a .txt file the background story of the game and displays in proper context and also the appropriate images with it, this component calls the play method of game engine which starts the main game loop.

3. User Manual

Minimum requirements:

Java 8 or above

Operating System: Windows

The user have to run the Nilkomolv_1.7.exe to run the latest version of the game.

At first the story will play and user have to press Enter to know more.

After the game starts the user have several keys mapped for several actions.

The control scheme is given below:

1. Directional Left and Right – Move left and Right,
2. Directional Up – Jump
3. Control – Attack
4. Alt – Block
5. H – Heal
6. P – Pause the game
7. G – God mode cheat code, makes health infinity and damage to infinity too (For debugging purpose)

4.Conclusion

Completing this project let me learn many things like:

- Handling large volume of code
- Using Separation of Concerns architecture
- Handling different game logics
- Learning to make executables

I hope I succeeded in making a game which allowed me to manage such large amount of code and also cherishing my childhood dreams of making video game about thakurmar jhuli

5. Appendix

I further want to improve my game in future by adding more level, features etc.

References

https://gamedevelopment.tutsplus.com/tutorials/introduction-to-javafx-for-game-development--cms-23835
https://en.wikipedia.org/wiki/Separation_of_concerns

YouTube links for the music used:
https://www.youtube.com/watch?v=MzLnzTDqQ30&t=3s
https://www.youtube.com/watch?v=u--BHiiW9OI
https://www.youtube.com/watch?v=XyvspF1PXNA
https://www.youtube.com/watch?v=fhx4he8mfqI
https://www.youtube.com/watch?v=uOIHHMnI_Ig

The images and animations were made by my father, Syed Kamal Hossain , Chief Of Graphics, Channel I