Bilkent University

Department of Computer Engineering

Object Oriented Software Engineering Project

Space Out: Retro sci-fi platformer game

Final Report

Doruk Çakmakcı, Anıl Erken, Umut Berk Bilgiç, Arda Atacan Ersoy

Course Instructor: Bora Güngören

Table of Contents

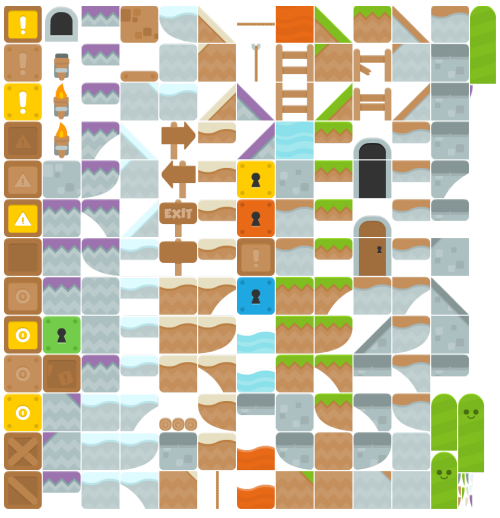
Implementation process

During the first stages of the implementation we were in search for an exact way of doing it whether using default libraries from Java or using external 3rd party libraries of which rely on more efficient technologies. We have looked into Java2D (Swing and AWT), JavaFX, LWJGL and Slick2D. We have decided to go with Slick2D. More will be told about this at the dependencies part.

Slick2D uses state based applications. We used this feature in order to implement several states to our application such as paused, in-game, in-menu, in-settings etc. The states dictate what the user sees and interacts with at that moment. Each state is implemented as separate classes and each of them extends BasicGameState class provided by Slick2D library. The transition between states are made possible with a class(SpaceOut) that extends StateBasedGame, a class which is provided by Slick2D.

This state based application follows class game development techniques such as the game-loop and three basic stages. First being the initialization of the game itself (not levels, players and such but the application as a whole.) Second being the update stage in which the game logic updates itself based on what happened during that frame. The third layer is the render stage which is required to separate game logic from GUI logic. All these stages are implemented as methods in each of the state classes. Within the game loop lies a state manager. This is crucial for the user to be able to move from menu to menu or menu to game and vice versa. When the user moves to the in-game state, a level is loaded from a map file and a player is placed in it.

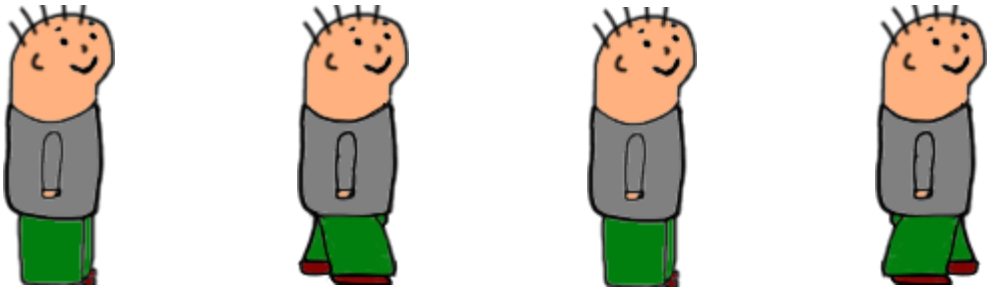
Whilst researching about game mapping practices, we have found out that there is a universal way of saving and interacting with tile based maps. Since our game uses tile based maps we figured the best practice for this would be to create and use these standardized map format rather than creating proprietary map / tile system. We have ended up with using .tmx (Tile Map XML). These maps store the map tile and layer information as well as the useful meta-data for the map such as width, height, background color, version, render order of the tiles etc. These maps use things called tilesets i.e. image files containing every texture being used by the tiled map. Slick2D supports these maps with the TiledMap class. This makes level creation easier while not affecting our design flow.



Example TileSet sprite sheet that is used by the tiled map file. The tiled map file should be able to access this image in order to create the map. The maps are not saved as images but rather as references to individual tiles. [1]

Character models are not a part of this tileset and are handled in a different way.

Regarding the Characters within our game, we use simple keyboard listeners to control them and the Slick2D built in moving animations to animate them. This way, we are not reinventing the wheel and can focus on making a better game without worrying too much about losing a lot of development time on developing inefficient versions of things that are already pretty well understood and built. To animate our character, we enumerate (we don't need to this, it is just a readability upgrade) a Facing for characters; left and right. Depending on the last move made by the user the facing variable of the character is changed. We also create HashMaps in order to map different stages of the movement to different character images. We put these images in correct order fast enough so it looks like our character is moving.



These are the four stages of movement currently being used in our test builds. We use built in Java methods to flip these images during the game to produce left orientation.

For the design of game menus, we built our custom Button class. This way, we provided a more flexible user interface by using our own image designs and input listeners. We interpreted each of the menus as “basic game states” and therefore each of them implemented as separate classes. Changes of the game states are done by using the positional coordinate checking method of our custom Button class.

Since our game has features like user signup and leaderboards; we had to store persistent data. We chose Google Cloud Platform for this. User data is stored in the cloud and login and signup processes are managed by it. Also, for each user, game score data are stored in cloud and in each gameplay, data is updated; leaderboards are modified if necessary, with respect to performance of the player. Using Google Cloud also helps us to satisfy the “security” non-functional requirement since Google provides a secure environment for storing data.

Status of the Implementation

At the moment, the main functional requirements of Space Out exists in our implementation. The main functional requirements are: sign up, login, play different levels, display leaderboard. A game has duration and score. Levels have .tmx extension as described above, and they are made with "Tiled" application. Collision detection is present and every GameObject instance have a Hitbox instance correlated with it. Hitbox is an axis aligned(x and y axis) collision rectangle in brief. The player starts from the location of the start block and the purpose of the player is to reach to the finish block in a reasonable time and with smallest number of jumps made. Player jump is made possible with holding the blocks underneath the player and with the game engine. Gravity applies to the player. the game itself has a player centered camera(the view or level moves with respect to the player.) Also, Space Out is a maven project.

For the moment, we are not able to implement the level maker feature of our game. It takes so much time and effort to design a menu to let user drag and drop game elements like enemies, blocks, background images etc. and to convert these images onto a .tmx file, and then upload them to cloud and retrieve them from the cloud. That's why this feature is missing. Also we are not able to implement user achievements and change user settings and in-game power ups because of time constraints.

Space Out is a maven project.

Design Change Choices Based on the Current Implementation

During the first iteration, we didn't have enough information about Slick. After that, when we learnt details about Slick2D, our design really changed. For example, at first iteration, we didn't know about how Slick handles game state changes. Therefore we tried to implement our own state handler, like a Finite State Machine. So we built our diagrams and design choices accordingly. But then, we learnt the technology and noticed that we don't need a Finite State Machine for this. During the second iteration, in design phase, we modified our plan with the knowledge of Slick2D. That latest design is accurate and very similar to our current implementation. Still, we had to do changes on settings menu and level choice menu. In those menus, we were planning to provide different options and change the displayed view with respect to clicked button, in one single state. But then we saw that changes of the view on the same state were a little complex and problematic and we thought that simple would be better. So we decided to provide these features in different game states and therefore implemented more Java classes. Other than that, using this 3rd party technology gave us the opportunity to focus on

actually implementing extra features by getting through the core aspects much faster, these core aspects being things such as maps, levels, game states, characters, level design and such.

User's Guide

First, a user without an account needs to create an account. To create an account and login to an existing account, a proper internet connection is required since, the user data will be provided from cloud. User info consists of a username and password without restriction(However, the usernames are unique). After sign up or login, user can play a level by pressing play game button, change password from settings menu, or check out global or local leaderboard. To play SpaceOut, user needs to select a world(Earth, Moon or Mars) and select a level with buttons. In game key bindings are: SPACE- jump, D- move right, A - move left. The player dies if they fall to void(water in Earth, darkness in Mars and Moon).

Source code of SpaceOut can be downloaded from https://github.com/umutberkbilgic/3I.Space-Out/tree/beta.

APPENDICES

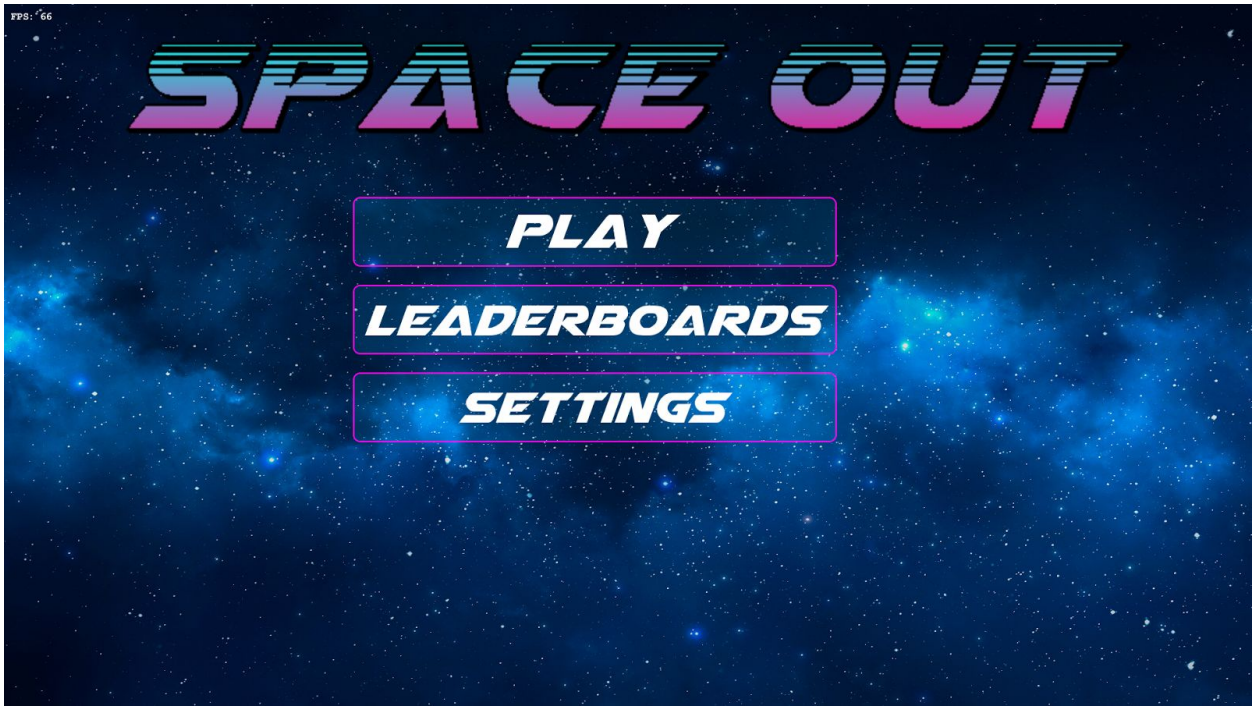
APPENDIX A (from Game class)

APPENDIX B (from LevelState class)

APPENDIX C (from Level class)

APPENDIX D

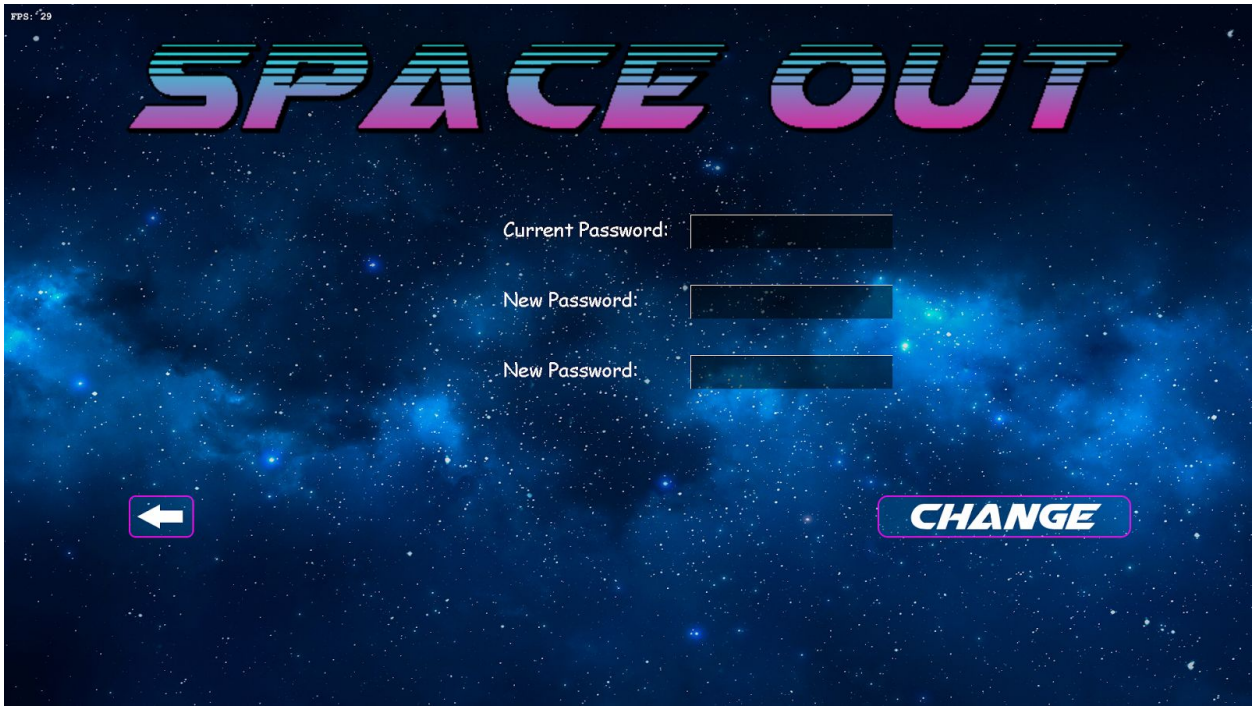
APPENDIX E (Screenshots from SpaceOut)

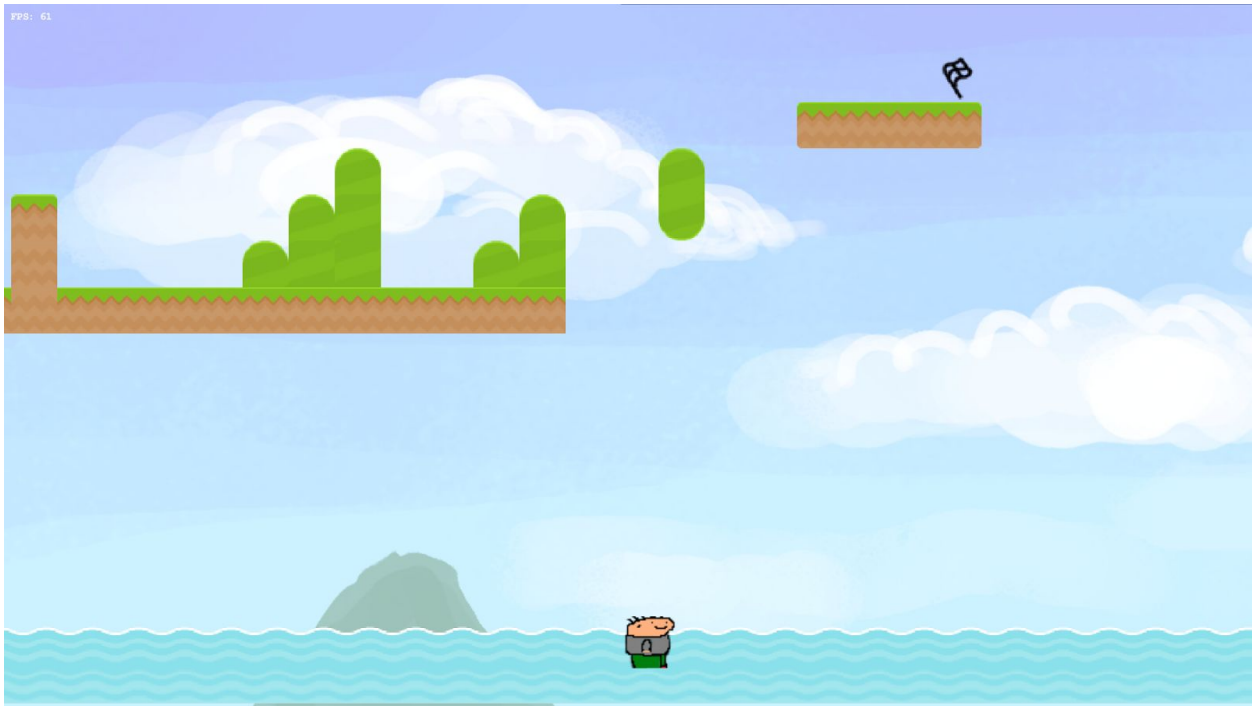

FPS: 66
SPACE OUT
PLAY
LEADERBOARDS
SETTINGS

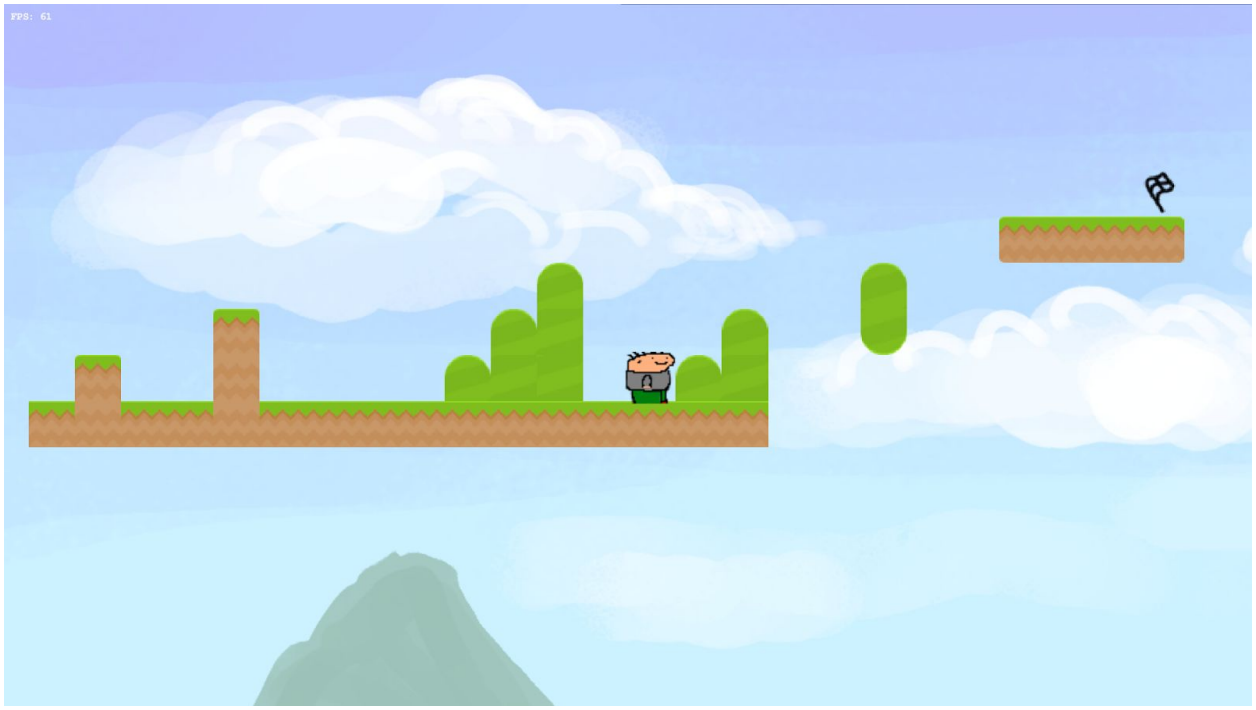

FPS: 61
SELECT WORLD
PLAY

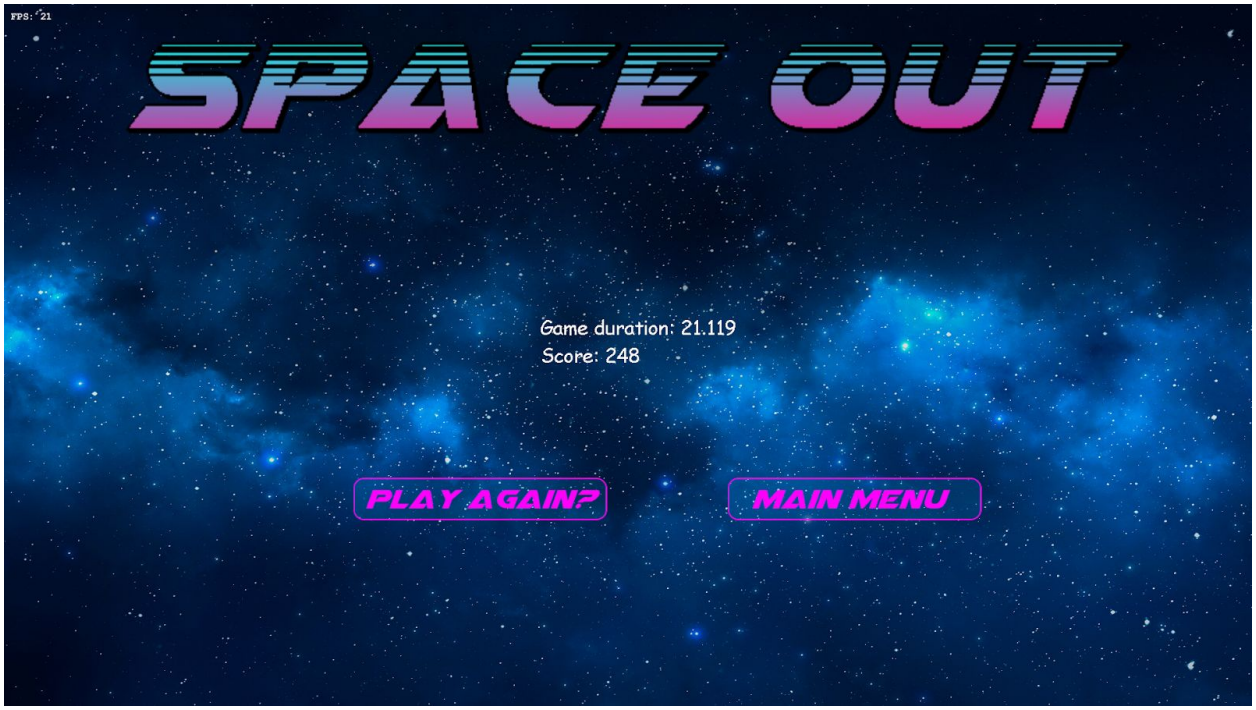

FPS: 61
SELECT LEVEL
LEVEL 1
LEVEL 2
PLAY


FPS: 61
SPACE OUT
CHANGE PASSWORD


FPS: 29
SPACE OUT
Current Password:
New Password:
New Password:
CHANGE


FPS: 61





From up to down:

Figure 1: Sign up view

Figure 2: Login view

Figure 3: Main menu view