Prison Break Final Report

**

Figure 1 Title Picture of Prison Break

# Prepared by

# Group 7 - Ho Chon, Adam Vanberkum,

# Simon Rankov

# CS 442

# University of Illinois at Chicago

# May 2020**Table of Contents**

[List of Figures](#_3znysh7) 2

[I Project Description](#_tyjcwt)

1 Project Overview 3

1. General purpose 3
2. How to use the product 3
3. How to build the product from the source code 3
4. Data files needed 3

[II Techniques and Methodologies](#_lnxbz9) 3

2 Agile Practices 3

[III Tools used 4](#_z337ya)

[IV Retrospective Review 4](#_1ci93xb)

[V](#_147n2zr) References / Bibliography 5

### List of Figures

[Figure 1 -](https://docs.google.com/document/d/1RoZaLBTjgmlfChurpV1NHMjkTFn28BEb/edit#heading=h.23ckvvd) Title Picture of Prison Break…………………………………………………………...1

### 

### 

# Project Description

1. Project overview
2. General purpose

The purpose of this game is to test the player’s skills and wit. The main goal is to have fun. The people who would want to play this game are those who like to play games during their free time. Our target audience are highschool and college students. Students become more stressed as they continue their studies, so our solution is to make a game that can relieve their boredom. By playing a challenging game, one can raise their level of concentration.

Other games similar to ours would be escape games and “The Worlds Hardest Game”. These compare to our game because the player has to avoid being caught and find the correct path out of prison to complete the game. However our game differs from others because of the graphics. We have a character sprite sheet animation and dungeon-like feel to our game. Future updates to our game would be to improve our guard’s AI. Improving this would make the game more challenging and makes the player have to focus more to win if they do not want to be caught.

1. How to use the product

To play the game, download the source code from github. Run the code on either Intellij or Eclipse. Run the game from the compiler and the game screen will pop up.

1. How to build the product from the source code

Download the src folder in the group github page. Compile the StartGame.Java class on the compiler. Run the game.

1. Data files needed

The files needed are the Slick2D library, LWJGL library, and the Tiled library. These libraries are free to download and the links are given in the references.

1. [**Techniques and Methodologies**](#_lnxbz9)

2. Agile Practices

For this project we used agile methodologies such as Scrum and XP programming.

For Scrum, we used a site called IceScrum to manage all of our stories we made to create this game. We made user story cards in IceScrum and accepted those stories in the product backlog. We ran several sprints to keep track of our progress with several demos of the game.

For XP programming, the practices we used were sit together, energized work, stories, and weekly cycles. Sit together was used initially when the project started, however with the Covid-19 outbreak, this XP practice could not be used anymore. We used energized work to accomplish most of our coding. Seeing how we had other classes, we only spent the time that was productive for us to finish this game. Stories were made to create the class we need for the game and was organized in IceScrum. Finally the weekly cycles were used for our meetings and to check our progress.

# Tools used

Slick2D is a 2D java game library which was the main library we used for creating our game. It has many useful tools for images, animations, particles, sounds, etc. This library along with LWJGL were used for creating the bulk of the functionality. Sprite movement and user interaction were easy to implement thanks to many of the pre-existing functions in Slick2D. Tiled is an editor which allows you to create 2D maps composed of rectangular tiles. A tile set could be either a single image containing many tiles or a collection of images. When creating the level for our game we used a set of 9 tiled maps such that when the user walks off screen in either direction the next room is already loaded and available. During the beginning of the semester we followed several scrum practices to keep us on track. Once the quarantine started this became an issue, however, we continued meeting via Discord.

# Retrospective Review

Originally, the plan was to use Java Swing and to set up our project in a MVC style hierarchy. We wanted to build the core functionality and then layer it with the GUI. This turned out to be much harder than previously expected. While a lot of the functionality was being implemented, figuring out how to configure our GUI as well as make it visually appealing was presenting many issues. We made the switch to Slick2D, LWJGL, and Tiled because we didn’t want to reinvent the wheel. A lot of the functionality we were struggling to implement, especially with the graphics, was already readily available to us through these libraries. Using Tiled allowed us to quickly design and implement levels and helped us save a lot of time. Slick2D also offered a way of easily tracking the game state and player interactions. After the first demo, we lost one of our group mates and had to double down. This was easier said than done due to the quarantine starting and not allowing us to work together the way we were normally used to. We also decided to focus on smaller and more important features for our second scenario rather than trying to polish what we had already done. This included adding guards, player items, and adjacent rooms. A problem we ran into with the guards was their ability to detect the player. When in proximity of a guard they should spot you and cause you to fail the game. One of the ways we went about this was by checking the squares within a certain perimeter of each guard. While we did implement the guards and their movements the code we had written for the player detection did not work. We believe this was because of the sprites movement being dynamic rather than turn based like we originally planned.

If we were to do a similar project like this one, we would have to make some time to get together and make sure to install the correct libraries. The initial parts of our project were tough since running the program was a problem unless you add the libraries correctly. If the libraries are not in the correct folder, you cannot run it properly. What we would change would be to use pair programming to make sure this is not a problem. Another problem was the IceScrum. We had a lot of problems logging into the new IceScrum server. One change would be to use Jira instead.

# References / Bibliography

Slick2D: <http://slick.ninjacave.com/wiki/index.php?title=Main_Page>

Download: <https://slick.ninjacave.com/>

LWJGL: <https://www.lwjgl.org/>

Download: <https://www.lwjgl.org/download>

Tiled: <https://doc.mapeditor.org/en/stable/reference/support-for-tmx-maps/>