Nick Benevento
19207773
Project 1 Report

For this project, I transformed the outline space shooter into a puzzle game. The game follows a seal, Stanley, through different levels as he tries to escape. Each level has a floor of ice that Stanley can slide around on. However, once Stanley has started sliding, he cannot stop until he runs into an obstacle. The game builds on this mechanic to provide fun and engaging gameplay as the player works with Stanley to escape.

Mechanics:

- Ice:

The heart of the game is based on the idea that most of the floor is ice. When the player moves on ice tiles, they will continue to move in that direction until they hit an obstacle or reach the bounds of the game space. During sliding, the player is not able to change the direction of their character. This mechanic is to provide a base for the game, and give it the puzzle aspect. The player has to carefully decide in what direction they are going to slide in to be able to reach the finish tile.

- Ground

These brown tiles are regular ground tiles. Here, the player only moves one tile at a time.

- Boulders

The gray boulders are another core mechanic of the game. These boulders will stop the player if they slide into them, and also prevent the player from moving past them. So, the player has to use them to their advantage in order to slide around the ice in the right sequence and stop at the right times.

- Holes

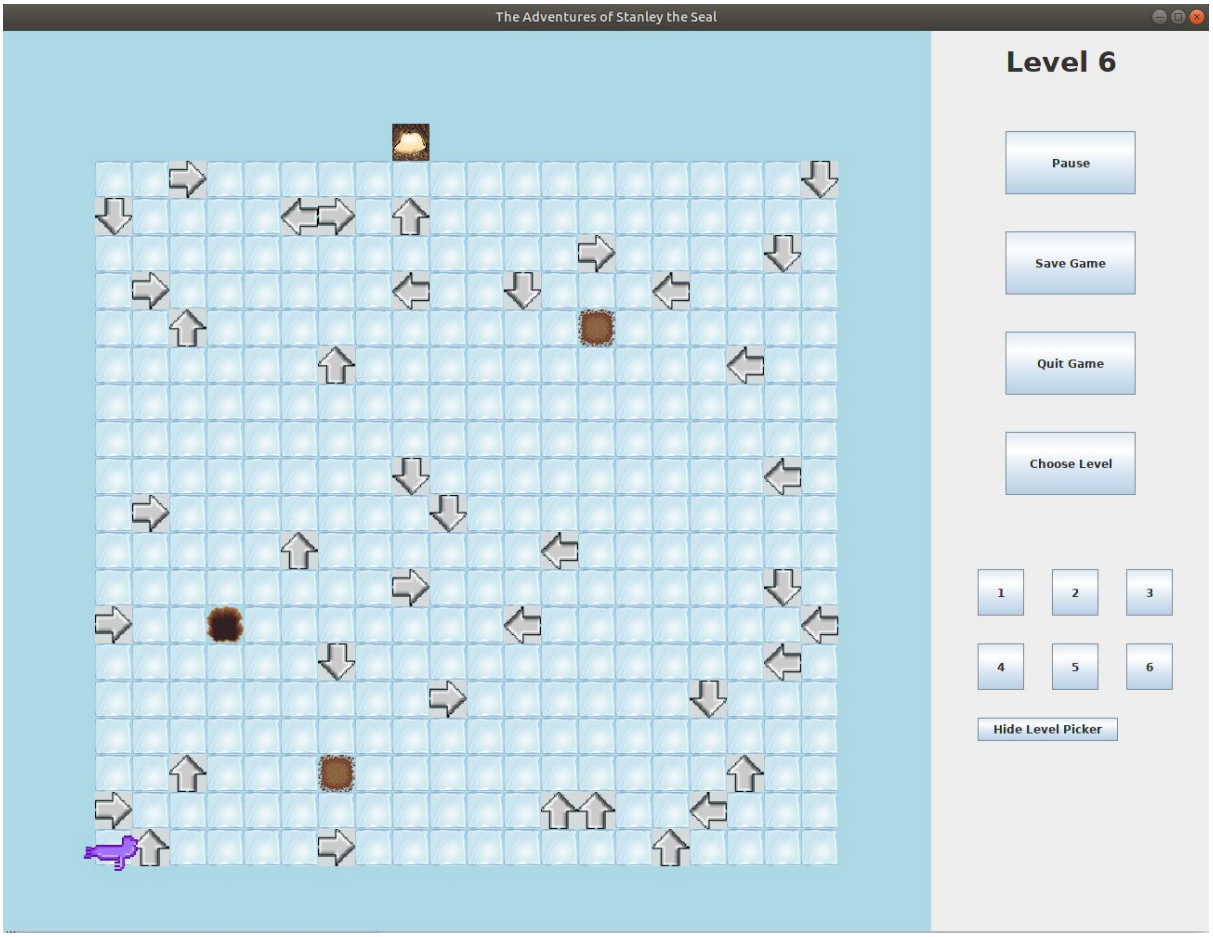
The holes in the ground are obstacles that the player will need to avoid in order to complete the level. If the player slides into them, they will fall through them and be teleported back to the start square.

- Colored directional (arrow) tiles

These are interactive tiles that move the player in the direction that the arrow is pointing. So, if there is an up arrow and the player slides into it while moving right, they will then start to slide up until an obstacle is hit. The player can use the mouse to click on these tiles and rotate the arrows, from right → up → left → down directions.

- Gray directional (arrow) tiles

These tiles have the same functionality as the colored arrow tiles, with the exception that the player cannot change their orientation. They provide a similar but distinct aspect to the game by being detrimental to solving the puzzle if the player is not careful in which of these tiles they slide into. In the following screenshot, moving up from the start will cause the player to slide in a chain-reaction of these tiles, eventually leading them into a hole.



- Finish tile

This is the end tile that the player is trying to reach; when they reach it, the level will be completed, and the next level will be loaded.

Features:

- Grid system

The base of the game is constructed around a grid system, that is 20x20 tiles. The grid is represented by a 2-D array of chars, with each char denoting a particular tile. This char array is parsed when each level is loaded, and each picture is loaded based on which character is read in. This system makes it very convenient to develop and add more levels to the game.

Since it is fairly expensive to load roughly 400 images every single frame, each level is only loaded once, with each tile drawn onto a BufferedImage. This single image is then

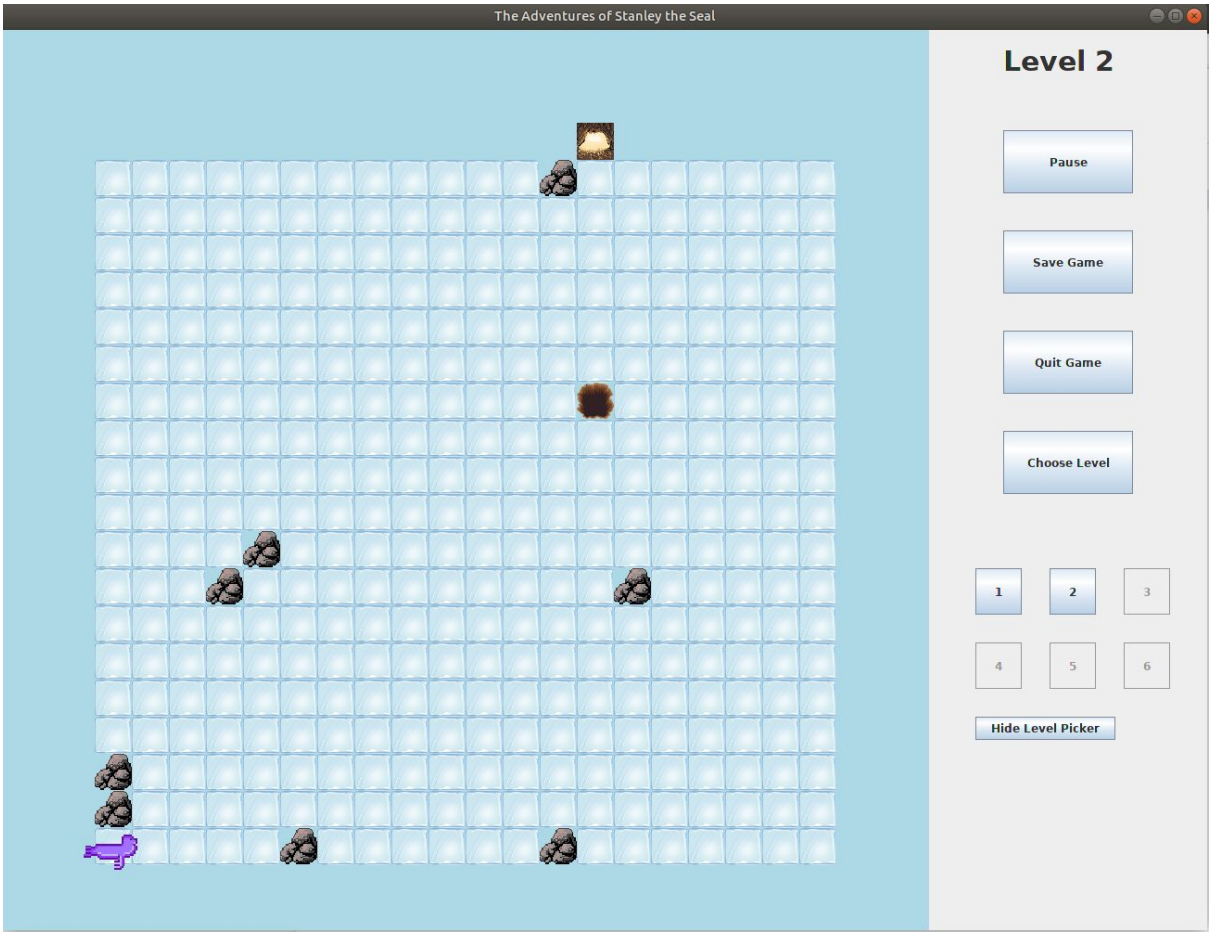
repainted every frame, which is a massive reduction in the amount of time and power needed to draw the level.

- Saving/Loading

The player is able to save the game (as long as they are not currently sliding), and then load that save file in the future. The game saves the player's current level, position, and also the highest level they have reached.

- Level Picker

The player has the option to choose levels if they wish to restart or replay them. However, they are only able to select levels that they have actually reached. Once a player reaches a new level, that level will become clickable on the right side.



The level selector can be seen on the right side of the image. Notice how levels 3 and above are grayed out because the player is only on level 2.

- Pause/Quit game

The player can pause/play the game, or quit the game at any time

- Inputs

The mouse is used in the game to select the directional tiles and change their orientation. In terms of movement, the player can use the ‘WASD’ keys to move around, as well as the 4 arrow keys if they wish. The game will move the player regardless of if the ‘WASD’ keys are capitalized or not. The game detects movement and mouse clicks on the action that they are pressed, not released.

- Collisions

The implementation I went with to detect collisions was to calculate the target position of where the player should end up after they input a move. This makes it easy to check for collisions between obstacles, since it is all done before the player actually starts moving and collision detection does not need to be done on the fly. It also makes it easier with the directional tiles, since the target position is just re-calculated based on the orientation of these tiles when one is hit.