Peter Vieira and Daniel Bezden

Mandelbrothers is a tile-based RPG written in python in which you must fight off waves of enraged creatures in order to save your brother. The game consists of an overworld where the player fights to gather coins, and a shop where the player can purchase upgrades. These areas are connected by a portal, and the player must travel back and forth throughout the course of the game.

The game uses the pygame library to blit all the sprites, play the music and sound effects, and display text. The maps were creating using the program Tiled to create sprite layers and facilitate the drawing process. These maps are read using the pytmx library. All music and sound effects included in the game are original and were produced by us using www.beepbox.co. The sprite art is a mixture of original and online art licensed for personal and commercial use. The art sources are listed below.

https://pipoya.itch.io/free-rpg-monster-pack (golem, pumpkin lantern, and bear)
https://babysamurai.itch.io/super-epic-fantasy-weapons-pack (bow)
https://grandmadebslittlebits.wordpress.com/2015/11/07/themo-monster-sprites/ (old man)
https://kenney.nl/assets/roguelike-caves-dungeons (spritesheet for background)

The Process

When given the opportunity of creative freedom for a final project, we decided to make something both of us have always been eager to create -- a game that actually has some substance to it. Before coming to this conclusion, one suggestion was to create a Python program to compute and plot the Mandelbrot set, allowing the user to zoom into any area and view the endless fractals. After temporarily discarding this idea, we decided to incorporate elements of it into our game and this heavily impacted the design process. For example, the overworld forms the shape of the Mandelbrot set when rotated 90 degrees counterclockwise and the game utilizes the ‘z’ key, in reference to the function $f_c(z) = z^2 + c$ used to generate the set. In addition, the player is stuck in an endless loop between the two areas, forced to travel deeper and deeper into the portals until eventually saving his brother and ending the game.

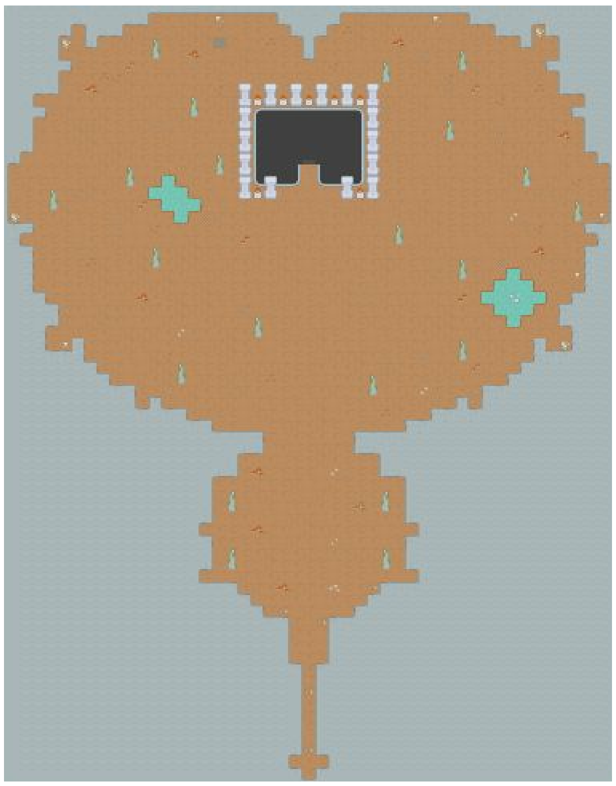
Early on, our goal was to create more areas, potentially with procedurally generated rooms, more items, and more mobs, but this was not possible due to time constraints. In addition, we intended to include a boss fight against an octopus made from a tentacle-like portion of the Mandelbrot set (the image is included in the source code), but this idea was put aside due to time constraints as well. Despite this, we intend to continue developing this game further outside of the course, and hopefully will incorporate these changes in a future update.

Getting everything to work as intended was, as with all coding, a tedious process. Early versions of the game used text files to generate walls and enemies, such as the included overworld.txt file, but this was replaced with Tiled maps. In order to enhance the experience beyond a silent game consisting of colored rectangles, we decided to draw and compose most of our art and music, and this increased the workload significantly. Beyond the superficial aspects of the game, the backend coding consists of over 1000 lines, and required a lot of problem solving to perfect and debug. Some of the difficulties included animations (which we eased with a list of sprites that is cycled through based on the framerate and whether or not the character is sprinting or standing), transporting the player between different maps (which we solved by creating a new game for the new map and transferring information that must be kept using global lists), and the fighting mechanics. However, many hours also went into the extreme attention to detail with the positioning of arrows, enemies, etc.

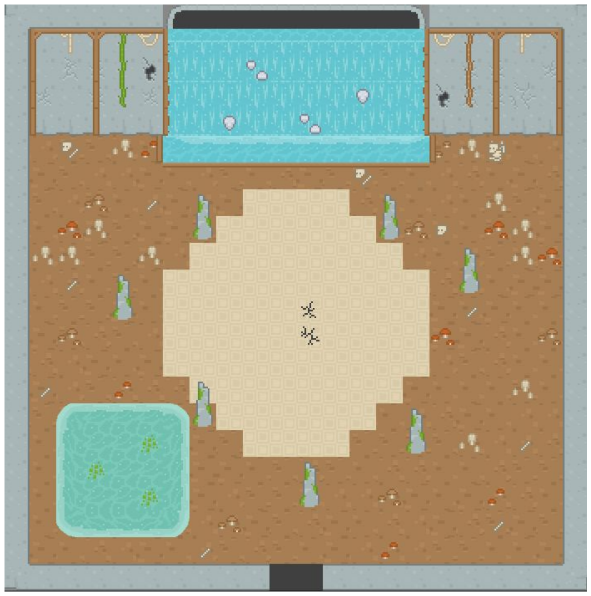
Finally, our project ties in with this course, Principles of Programming Languages, in many ways. We used the imperative and dynamic language Python, as we did at the start of this course, and furthered our understanding of it by using libraries. In addition, we used our knowledge of adapting to unfamiliar languages/concepts, which we learned when we were introduced to functional and dynamic languages in this course. Thus, this project allowed us to utilize what we learned and also increased our experience and coding abilities.

Game Information

Overworld



Shop



Mobs

Snail: Completely useless; gives 0 gold
Electric Snake: Weak; average stats; gives 1 gold
Reaper: Fast; able to walk through walls; gives 2 gold
Flame: Slow; deals massive damage; gives 3 gold
Golem: Tank; deals massive damage; resistant to slows; gives 10 gold
Bear: Tank; deals massive damage; resistant to slows; gives 8 gold
Pumpkin Lantern: Very fast; able to walk through walls; gives 15 gold

Upgrades

Health: Restores health; costs 20
Speed: Doubles fire rate; costs 30
Damage: Doubles damage; costs 40
Armor: Doubles health; costs 50
Ice bow: Slows enemies; costs 50
Triple bow: Shoots three arrows; costs 100
Brother: Ends the game; costs 1000