TP1: thisisntsand

Project Description: thisisntsand is a sand art generator, with a couple quirks. Deriving inspiration from thisissand.com, the project features three modes: a sandbox mode, a picture mode, and a game mode. In sandbox mode, the user is free to pick different colors and pile sand to create art, which then can be saved. In picture mode, the user uploads a picture and has the option to watch the computer recreate and sand-ify it, or they can choose to replicate the picture with sand, after the computer will score their recreation. In game mode, the user must direct a flow of sand from a starting point to an end point by drawing lines for the sand to slide on and avoiding the obstacles in the way.

Competitive Analysis:

The two already-existing programs that my project derives inspiration from are thisissand.com and sugar, sugar. My project integrates parts from each--the idea of freeform sand art comes from thisissand, and the idea of a game where users direct a quantity of particles to an end goal by drawing lines is from sugar, sugar. On top of these features, my project also implements a unique "picture mode" not seen in either of the above, which turns the sand art creation process into a game. Additionally, my game mode will implement random level generation of a specified difficulty, unlike sugar, sugar, which has hardcoded levels, although more of them. Finally, my main-post MVP additional feature, which involves sliding the sand based on the angle of the user's computer, is a unique idea not seen in either of these two existing games.

Structural Plan:

The project will make use of modalApp to organize which mode is active out of the splash screen, help screen, sandbox, picture, and game modes. Each mode has its own file that will be referenced and called by the main file. The main component, sand particles, are a class from which sand particle objects are derived so that modifications are easy. Except for the particles, there are no shapes drawn; everything is manipulated in terms of images.

Algorithmic Plan:

The following are the main challenges in this project and how I plan to solve them:

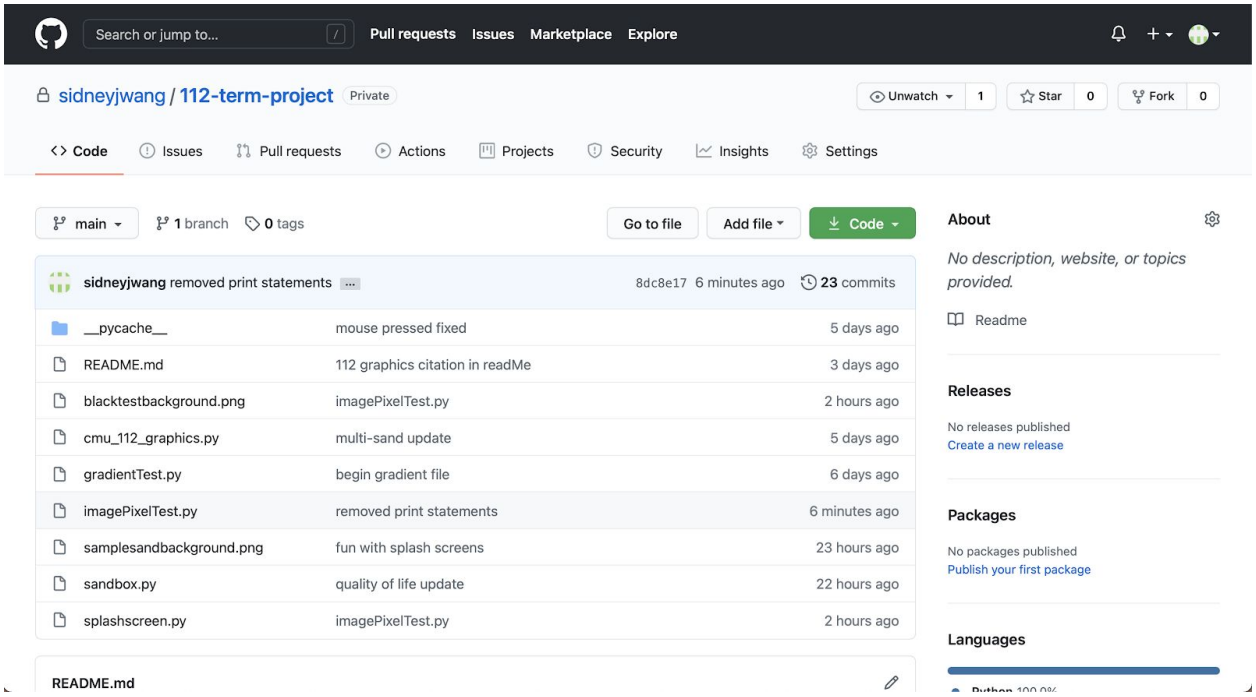
- *Storing large quantities of sand without lagging:* Sandbox mode begins with a blank white image as the background. I will use a method similar to the implementation of Tetris in the homework. Any falling particles are objects of the particle class and are drawn on top of the background and stored in a list that houses all of the particle objects; once a particle lands on other sand particles or on the bottom of the screen, it's popped from the list of particles and the background image is modified to have a sand grain at that pixel. I have previously tried not using the background method and having just sand objects, or implementing the Tetris-framework as-is without involving images, but both are too slow to handle larger quantities of sand. This blank png approach makes it so that every redrawAll, there's only one image plus the moving sand particles (which there is a theoretical cap on the quantity of) that are being updated.
- *Making sand and pile on top of itself:* For each sand particle that is still an object, I will check if the pixel in the background image at its next calculated spot is already occupied

by looking through a list that contains all the smallest y-values for each column. If it is, I will move the sand particle to the next highest unoccupied spot for that column. Then, I will slide the sand particle object by checking the spots to its southeast and southwest to see if they're occupied; if there is one that isn't occupied, I will remove the sand particle object and then color the corresponding background spot to be the same color as the removed sand particle object.

- *Drawing lines that are detectable by sand:* To do this, I will manipulate the background image pixels at the spots that the mouse is dragged. This way, I can use the same sand sliding algorithm described above to govern the interactions between line and sand. I will implement Bresenham's Line Drawing Algorithm to create lines where the user's mouse has been.
- *Animating a computer-generated sand image:* The sand will sweep from left to right, changing colors based on an average of the reference image's pixel colors and be piled such that the end result is very similar to the uploaded image, minus some discrepancies due to sand color randomization (since each sand particle has a unique color!) and sand sliding.
- *Developing a scoring algorithm for user-created images:* I will score the sand art created both on accuracy to the original image, as well as time taken to reconstruct it. The minimum time will be calculated based on the time it takes for sand to fill the entire canvas, plus a bit of extra; any longer time taken will subtract from the score. I'll compare the average value of a pixel cluster in the user's image with an average value in the original image to determine the accuracy; the MSE of all the clusters will be subtracted from the score.
- *Generating semi-random levels of similar difficulty for game mode:* I will generate random obstacles by drawing them first as shapes, then taking a snapshot of the screen and setting that as the background so sand can detect it when falling. This should make it simple to create various semi-random levels.

Timeline Plan:
Working sandbox mode: 11/28
Gradient selection: 11/29
Game mode: 12/2
Picture mode: 12/3

Version Control Plan: I am backing up my code to a private repository on GitHub. I push the latest code every time before I begin making a major structural change; this way, if the change doesn't quite work out, I can recover the previous version to re-brainstorm and start from. Here is a screenshot of my repository:



Module List: I am not using any modules in this project before MVP. After MVP, I plan to incorporate OpenCV to implement the optional feature of having sand slide according to the tilt of the user's computer, using edge detection.

Sources so far: I learned about Bresenham's Line Drawing Algorithm from and based some of my code off of: https://inst.eecs.berkeley.edu/~cs150/fa10/Lab/CP3/LineDrawing.pdf