Project Description

Title: Addventure

An interactive game that requires users to use their math skills to navigate through terrains. A linear function to complete a break? A piecewise function to create a “bridge”? Using parabolas to help collect a star? Complete your path using a range of functions, keep in line with physics, and run your avatar to maximise your reward and stay from dangerous items while reaching your target. This version consists of 9 levels where you need to move your avatar from the entrance to the target but need to fix the breaks and collect awards/dodge obstacles using functions.

Competitive Analysis

After thorough research into both web online games and app-based math games, there are a variety of function-learning games. However, most of them are quiz-style games, art with function apps, modeling platforms to visualise custom functions, and trajectory-function games. This game adds a bigger gamification element to augment the user experience while creating an exploratory environment that allows users to play with multiple functions per level to enable faster learning experiences. Navigation style games are already really popular and integrating this fun “math element” adds an educational twist to this mainstream style. Gamification in education is shown to have numerous benefits including higher engagement and better recollection. Unlike other math-based games that rely on you learning the particular material before playing, we give simple explanations and guide you with the parameters needed to design the functions used.

Structural Plan

Envisioning to use the CMU Graphics App/ModalApp class to implement the graphics and interface. I am using the keyPressed, timerFired, mousePressed, redrawAll, and other methods to allow for interactivity and build a rich user experience. For the game aspect itself, I have divided tasks into multiple files: terrain generation, character animation, function-path integration, and navigation tasks. The levels, player names, scores, etc will be stored in local files (possibly include a database after MVP). The MVC principle will be leveraged and functions will be made accordingly: model building in appStarted, drawing helper functions called in redrawAll, and many checkPressed style functions to help with user interactivity. Here, the use of modal app to seamlessly integrate different modes of the game is critical. I am also using a multitude of image files for graphics and character animation purposes. Post-MVP, the function creator could be implemented in a separate window using TopLevel in CMU Graphics with tkinter.

Algorithmic Plan

One of the complicated parts of the project is to manage multiple frames, the game character, and the terrain. Using ModalApp is a good option but, currently, I have implemented a boolean flag system to help the program recognise the stage it is in. The terrain generation, however, is very algorithmically complex. Not only am I planning to use a noise algorithm or diamond-square algorithm to devise a realistic 2D terrain, I also have to introduce breaks in the terrain, find locations to add stars, TNTs, and spikes. I have created a function to figure out how close a point is to the

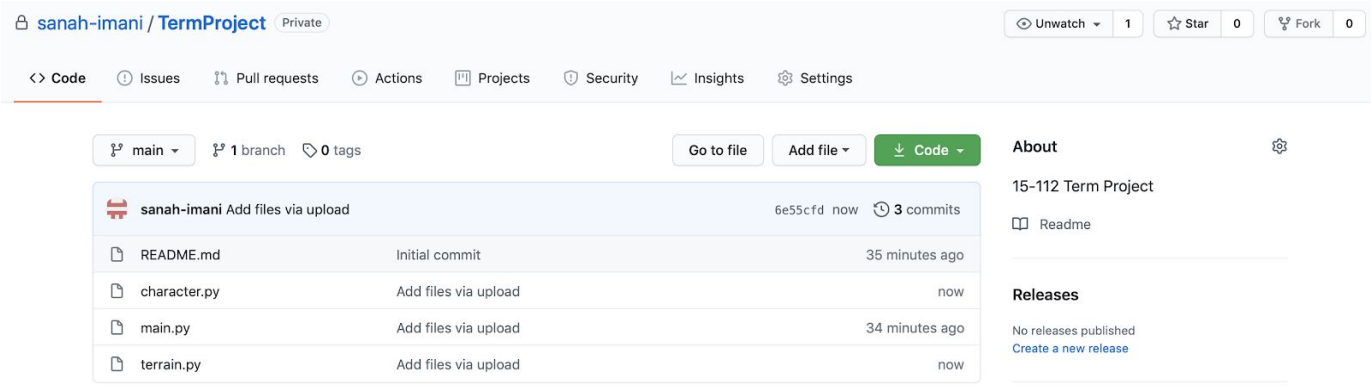
ground which is really useful in placing obstacles. The collision detection will also be a challenge and I will have to iterate over the 2D array representation of the terrain to figure out empty vs occupied spaces. I also plan to work on sufficient realistic character animation that requires derivatives and physics-intuitions so that the figure moves orthogonally to the slope of the terrain.

Timeline Plan

I intend to finish the function drawing interface, most of the UI elements, and the game template (a preset level with obstacles), and partial character navigation (only linear) methods by the 30th of Nov. Some chunk of the terrain generation will also be covered by this date. Terrain generation for all 9 levels and character navigation (moving realistically on curves - physics inspired) by the 3rd December. The user interface and possibly more complex terrain elements/functions will be worked upon and refined till the 7th December. Finally, I will perform a series of tests to enable me to figure out any flaws and research into the use of some external modules to help better the game levels. Another post MVP possible improvement includes user-controlled navigation instead of automated navigation.

Version Control Plan

I am using a GitHub repository (not local) in order to maintain version control. Here is an image of the organisation:



Module List

- Numpy (after MVP)
- Matplotlib (after MVP)
- CMU graphics
- Tkinter (after MVP)

TP2 Update

Worked towards the core components required for my MVP. Spent hours on research and used my own knowledge about calculus to derive a terrain-generation algorithm. This was difficult because the terrain needed to be gradual and natural. I also wanted the power to modulate the steepness of the terrain to create varied levels. Another difficult part was finding potential coordinates to flag as breaks and also ensuring that there were some breaks at an angle and breaks at a flat surface. I also implemented a way to make frames of the terrain as otherwise the terrains would be too short. I used different frames of a stickman sketch and constructed an algorithm to easily shift between the different states. The difficult part was making a time delay system to make sure that each of the frames were visible during the journey. Since the terrain is not necessarily flat, I used some basic principles to move in the direction of the slope. This involved the use of trigonometry, gradients, and vectors to effectively accomplish. Finally, I worked on building most of the function drawing interface.

The changes I made: completely changed the terrain generation algorithm, I made an algorithm for better selection of breaks in the terrain instead of selecting them purely randomly, changed the framework to a ModalApp one.

TP3 Update

Instead of incorporating multiple classes of functions, I stuck to just one class: polynomials (square root, linear, quadratic, cubic, quartic). Added a rotation and reflection feature for functions which can be performed by pressing certain keys. Finally, instead laying spikes on the terrain, I displayed flying spikes that are distributed just like the stars.