Anishka Bompelli
April 16th, 2022
15112: Fundamentals of Programming and Computer Science

Term Project Design Proposal

Project Description

Name of Project: The Dinosaur Game 2.0

The Dinosaur Game 2.0 is a rendition of the original Chrome browser game, the Dinosaur Game. This project is a side-scrolling game where the user guides a Tyrannosaurus Rex through several obstacles. The T-Rex must avoid falling into holes, jump over cacti, and evade flying birds. The longer the T-Rex survives, the greater the score count. If it collides with an enemy or falls down a hole, it's game over.

Competitive Analysis

As mentioned before, the Dinosaur Game 2.0 is based on the original Dinosaur Game. It utilizes a similar concept of a never-ending game where the main objective is to avoid obstacles to survive and increase your score. What makes 2.0 different is the rocky terrain with added holes: an extra obstacle for the T-Rex to avoid, which can be done by jumping over it. In addition, the enemy birds will divebomb the T-Rex once they are in close range of it.; the T-Rex can avoid this attack by jumping over the birds once the birds are low enough.

Structural Plan

The project will be organized as follows:

- A main.py file that holds the main functions from cmu_112_graphics.py, i.e. appStarted, timerFired, keyPressed, and redrawAll
- The following files will be imported to main.py
    - The terrain.py file will hold the functions for the midpoint displacement algorithm and drawing terrain.
    - The dinosaurSprite.py file will hold the functions for drawing the dinosaur sprite, and for making it walk on the terrain
    - The birds.py file will hold a class for the enemy birds that include functions for drawing the bird sprite, randomly spawning them, and their dive-bomb attacks
    - The cacti.py file will hold the function for drawing the cacti
    - The score.py file will hold functions for calculating and drawing the score in the top right of the screen

Algorithmic Plan

Terrain Generation

The terrain generation makes up a significant amount of the complexity of this project. For the terrain, I am using the midpoint displacement algorithm which I programmed into a recursive function. I am passing into the function a 2D list holding two points on the canvas (start and the end), and I am finding the midpoint between these two points. Adding a random offset to it creates "jagged" terrain. After inserting this calculates midpoint in the middle of the 2D list, I call the

function recursively on the left line segment and right line segment. This way, the function is continuously finding the midpoint of every new line segment created, until it reaches the base case. The base case is when the number of times I want a midpoint to be calculated reaches 0. Every time a recursive call is made, I subtract the number of iterations by one. The purpose of having a set number of iterations is to generate a set number of points to be drawn as the terrain. The more iterations, the more points there are in the terrain and thereby the more detailed. Having too many iterations can cause the game to run very slowly, so having less than 10 iterations would be optimal. Something to keep in mind is that storing every point of the terrain, including the points that have already been drawn and have moved off the canvas, would result in the terrain gradually generating slower. To avoid this, I would have to check which points are not on the screen and slice them off the 2D list.

Enemies

The moving enemies in this game would be the flying birds. In the original Dinosaur Game, the birds are spawned at a certain height on the screen and fly towards the dinosaur in a straight line. In this project, the birds will dive down to collide with the dinosaur once it has reached a certain proximity to it. I will check for the number of points on the terrain between the bird's current position and the dinosaur's position, and if it is within 30 points, the bird sprite will be switched to the sprite of the birds angled downwards. The bird's 'x' coordinate will decrease its and 'y' coordinate will increase every timerFired interval; this way the bird will move towards the dinosaur.

As a part of the complexity of making enemies in this game, I have to check for collisions between the dinosaur and the enemy. This can be done by checking if the 'x', 'y' coordinates of the dinosaur's position is the same as the 'x', 'y' coordinates of the bird's/cactus's position. If they do, then the game will be over.

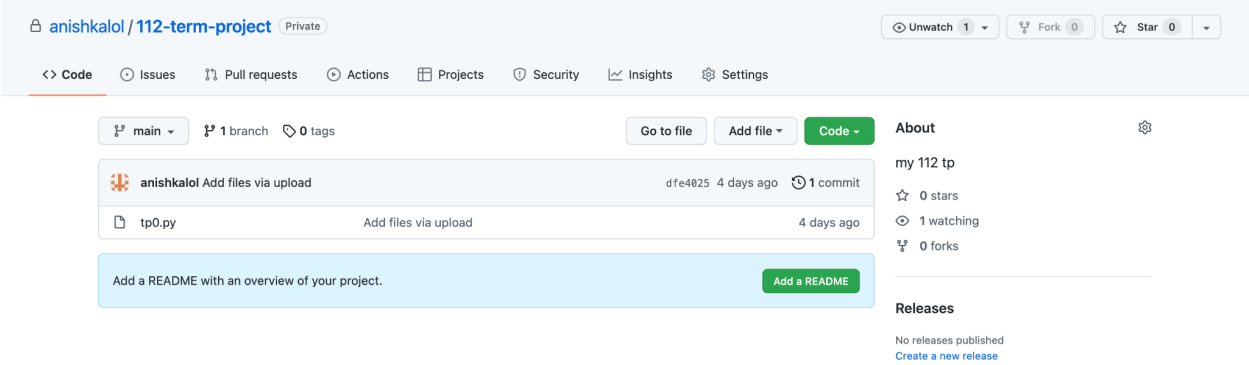
Timeline

- TP0
    - Prototype code
        - Have an algorithm ready for the terrain generation
        - Have code that captures the general idea for drawing and spawning the birds, cacti, and dinosaur
- TP1
    - Finish terrain generation with holes in the terrain
    - Have a working dinosaur sprite that can jump/fall
    - Have the dinosaur be able to walk on the terrain (be orthogonal to the terrain)
- TP2
    - Have randomly generated cacti
        - Detect collisions
    - Have randomly generated birds
        - Have birds divebomb once dinosaur is in close proximity
        - Detect collisions
    - Detect when dinosaur has fallen into hole

- TP3
    - Calculate and draw score
    - Draw game over

Version Control Plan

All iterations of my term project will be uploaded to a private repository on GitHub. I have uploaded TP0 so far, and will upload every iteration after each deadline.



Modules

I will not be using any external modules for my term project.

TP2 Updates

None

TP3 Updates

None