Raphael Astrow

Term Project Design Document

Overview

My program, Eco City, is a game based off popular computer games SimCity, Roller Coaster Tycoon, and Age of Empires: Age of Kings. All of these games involve making calculated decision on how to develop a city, theme park, or army. In my game, I wanted to make something similar (I love those games), but relate it to my work designing an Ecological Footprint Calculator. My game is about developing a city in an ecologically sustainable way while satisfying the needs of a rapidly growing population. The two most important metrics of my game are "EcoCredits" and "Satisfaction Score." Using my ecological footprint calculator in an extremely loose manner, I self-defined the ecological cost of building and maintaining various improvements, such as a vegetable farm and airport, in terms of EcoCredits. SatisfactionScore decreases every time the city's population increments by 1,000. Since EcoCredits never replenish, there is no way to "win" the game, per se. The goal of the game is to achieve as high a score as possible by building improvements that satisfy the population.

What I Had to Do

- Make a scrolling map in which many different images can be dragged from a scrolling sidebar and drawn on the map so they do not overlap and are easily locatable on a mini map. The map scrolls according to arrow key presses. The mini map tracks the current map location. Image selections and placements are tracked between four different lists: newImages,oldImages,occupied, and oldOccupied. Image dragging is controlled by a number of variables, including

drOne, drTwo, drThree, display, and imageScroll. Almost every segment of my project is list-based or tuple-based.

- Learn to use Pygame. From the online tutorial I studied, I began writing code within one main program loop. In that program loop, many different functions are called in a well thought order. Pygame's clock.tick function was used to limit the frame rate of the program loop. Since pygame works differently than Tkinter, it was not feasible to use the model-view-controller template discussed in class.
- Develop a complex sidebar. Improvements are purchased from the "improvements" sidebar. Since there are many improvements, it was necessary to make the sidebar scroll when the user clicks on photoshopped arrow buttons on the sidebar. Since only three images can be displayed on the sidebar at a time, but all images can be dragged and placed on the map, controlling the images that are placed on the sidebar was a challenge. Also, I wanted each improvement's caption to be centered, and since each caption has a different length, each caption was assigned different blit positions (in the imageNamesPositionsXList list). imageScroll is an especially important variable because it controls sidebar scrolling. Every time the "up arrow" is clicked, imageScroll increases and every time the "down arrow" is clicked, imageScroll decreases. imageScroll, imageScroll+1, and imageScroll +2 are the indices of the three images displayed on the sidebar. Everything is drawn using pygame's "blit" function. In the bottom of the sidebar, there is a box that alternates between displaying improvement descriptions and displaying a constantly updated mini map. Printing the appropriate description was sometimes a challenge and drawing an

accurate mini map required a lot of mathematical calculations. The list "printed" is used to ensure that lines were not printed multiple times. This list is necessary because the description being displayed is constantly updated from the main loop.

- Develop a simple game concept, where the user runs against time constraints to discover which improvements are most worthwhile to build. Users who make the right decisions on which improvements to build can achieve very high scores. Wrong decisions are costly, though, because the user has only 150 credits to purchase improvements and these credits never replenish. Also, buildings can never be removed.
- Design a simple, yet self-evident user interface with a name input and a high score list that gets saved in an external text file. The game also features a pause screen, splash screen, game over screen, high score screen, instructions screen, name screen, and game screen. The pause screen may be accessed anytime during the game by pressing "p". From the pause screen, users have the option of pressing "i" to view the game instructions. The game can be restarted during game play by pressing "r." The splash screen signifies the start of a new game. The high score list was particularly difficult to create because everything saved to an external file becomes a string. Since I only used one save file and that file contains both letters, numbers, and at one time lists, manipulating and sorting the string upon loading the file became complicated.