Greg Rapp

**Apple Shooter Report**

**Overview:**

Fractals are geometric figures that use the pattern of a simple shape to grow into a complex and chaotic whole. Fractals can often be used to model nature. For example, a tree starts with its trunk which splits into branches which split into more branches and so on until it reaches the end branch. Originally, I wanted to build a program that created fractals to model certain aspects of nature, however the math required to be able to create such a program was beyond what I am capable of. This being so, I still wanted to implement a fractal design into my project so I found a program online off of the Mathworks website which created two-dimensional fractal trees. I decided to make a game where the objective is to shoot all apples that fall from the ends of the branches on the fractal tree as fast as possible using a gun controlled by the mouse. The apples start at a certain size and shrink when hit by a bullet from the gun until they are eliminated completely. If the apple falls off the screen, the apple would grow in size and put back to its original spot on the tree.

The fractal tree program originally made a figure of a tree with variable aspects as input arguments such as how many branches the tree made and the angles of which those branches shoot out from. I then created a graphical user interface that was able to call this function and plot it onto the axes for the GUI. I did this by adding an axes input to the function and adding the input to the line functions within the program. Next, I plotted the apples at the endpoints on the tree by finding the variable that held the positions of the branches, making it a global variable, and plotting the apples at the end of that global variable.

Once I had the apples on the tree, I then used much of what we learned in asteroids to make them fall off the tree using a timer in execution mode. When I got the apples to fall off the tree, I learned how MatLab can detect cursor position using the window button motion function. This function keeps track of the motion of the mouse over the GUI. Using this function, I plotted a “gun” that would update its position according to the x-position of the mouse with a constant y-position. In order to get the gun to shoot projectiles, I used the window button down and window button up functions in MatLab. These functions activate when the mouse is pressed and released. When the mouse is pressed, the game will create a new bullet every ten frames with the same coordinates as the gun and stop when released. Now that the gun could shoot bullets I could calculate the distance between the bullets and the apples, making the apples shrink or disappear when the bullets hit them.

In order to make the game a bit more challenging, I added growth to the apples. If the apples make it to the bottom of the screen, the apples grow and are placed in their original location. Only after the smallest form of the apple is hit by a bullet will the apple be completely eliminated and not grown back onto the tree. This is also how the difficulties differ within the game. If the game is on easy difficulty, the apples can only grow to be one size larger than its original size. Medium difficulty will allow the apples to grow two sizes larger and hard difficulty will allow them to grow to three sizes larger. Once all the apples are completely eliminated, the end game function is called which stops the timer and computes the final time to eliminate all the apples. This computed time is then displayed in a static text box on the GUI. Once the apples are eliminated, the user may press reset which resets all variable aspects of the game and stops the timer.

This project helped me better understand most of the topics covered in class as well as introduced new tools which all helped to create a game that runs smoothly and intuitively. I feel as though I am much more capable in MatLab to create unique programs on my own. Overall, I improved my ability to read complex code and manipulate it, I learned about many functions within a graphical user interface, and I understand how to better utilize many of the aspects of MatLab covered in class. I now feel more comfortable with MatLab than ever before and will be able to use it for future endeavors.

**Example:**

<https://drive.google.com/a/appstate.edu/file/d/1xeskMUw_-2FXP3A_P3bih4J_8LsTSPU0/view?usp=sharing> (video)

**User Guide:**

1. Click Start to begin playing Apple Shooter on the default setting (easy mode).
2. Apples will start to fall, press stop to pause the game.
3. Move mouse over figure to control gun.
4. Hold the mouse button down to fire the gun.
5. Shoot the apples on the screen, they are only eliminated when the smallest version of them are hit.
6. The current time since the start of the game is displayed in the bottom left text box.
7. When all of the apples are eliminated, the game ends and the final time is displayed in the text box.
8. Press reset to replay the game.
9. To make the game harder, increase the difficulty which will allow the apples to grow larger.

**Tutorial:**

Radio buttons:

Radio buttons were used in order to set the difficulty of the game. Each radio button when pressed first checks the other radio buttons’ values and turns them off if they are on. They then stop the game timer and call a reset function which resets the game back to its initial state.

Growth of Apples:

When apples fall to the bottom of the axis in apple shooter, they reappear in their initial location on the tree one size larger than what they were before they fell. This was done by giving all apples a growth counter and having the update positions function constantly check each apple’s growth counter. When an apple falls below the axis, the growth counter for that apple is increased by one and when an apple is hit, it’s growth counter is decreased by one. The apple is only eliminated when it’s growth counter is less than zero and that apple’s positions is given the value of NaN.

Score and High Score:

The “score” in this game is based on the time it takes the user to eliminate all the apples from the tree. It is calculated by the amount of tasks the timer has executed since it was started multiplied by the time it takes the timer to call one task. The score is then displayed at all times in a static text box on the GUI and saved in an excel file which stores all scores ever received. This excel file is then read and the lowest time in the file is displayed as the high score in the textbox.

Window Button Functions:

The window button motion function allows the program to detect the mouse cursor position over the GUI. This data is saved in the handles structure and the x data of the mouse cursor’s position was used to plot the gun so the gun would follow the mouse as it went across the screen. The window button down and up functions activate when the mouse is pressed down or released. When the window button down function activates, a variable called mouseDown is set to true and vis versa for the window button up function. This allowed the gun to only shoot when the mouseDown variable is “true”.

**Resources:**

Mitry, D. “MathWorks File Exchange”, *Generation of 2D Fractal Trees*, The MathWorks Inc, 25 Nov. 2010, [www.mathworks.com/matlabcentral/fileexchange/29536-generation-of-2d-fractal-trees](http://www.mathworks.com/matlabcentral/fileexchange/29536-generation-of-2d-fractal-trees).

Buckingham, David. “MathWorks File Exchange”, *Dave's MATLAB Shooter*, The MathWorks Inc, 8 May 2011, [www.mathworks.com/matlabcentral/fileexchange/31330-dave-s-matlab-shooter](http://www.mathworks.com/matlabcentral/fileexchange/31330-dave-s-matlab-shooter).