Target Shooting Project

An experiment in using Processing to create a small simple game

Aidan, ACW, Weller

University of the Fraser Valler, Aidan.Weller@student.ufv.ca

This project is intended to showcase the acquired skills in the Processing platform, including usage of the pushMatrix and popMatrix, as well as a dynamic target generation system. The culmination of this is a small game where the player clicks on targets to receive points. The difficulty comes from the randomized lifespan of each target.

CCS CONCEPTS Software creation and management• Design and analysis of algorithms• Computers in other domains

ACM Reference Format:

1. Introduction

This project has come to be as a result of the education and training provided by Comp 350. This project strives to showcase that education in the form of a short game that loosely simulates a shooting range and requires of the user some skill in reflexes to keep up with the limited lifespan of each target as well as precision in mouse movements to get each target correctly.

1. Body

This project represents a self-imposed challenge to create a game with the Processing platform. There were clear objectives I laid out for myself, some of which were completed, and others were not, or were implemented partially. The objectives are as follows:

1. The popMatrix and pushMatrix functions must be used to add 2d transformations to the matrix stack and remove them from the matrix stack.
2. Separate files with organized classes and functions are to be utilized to avoid a messy main file that could cause confusion to anyone attempting to read it.
3. Implementation of randomized lifespans for each of the targets as well as randomized locations for each target.
4. Targets should be an object organized into an Arraylist that is easy to access and adjust the target on an individual basis.
5. The body or character shall be an easy to call function that creates a usable body without a long cluttered string of 2d shapes in the main file.
   1. Implementation

The objectives above are more or less implemented in a fashion that works, however, there are some bugs and visual glitches worth covering. The objective of using the push and pop matrix functions was easy to implement and works without issue, the same can be said for organization of files, classes, and functions. All of the organization was implemented to a satisfactory degree. Implementation of randomization was incredibly effective and works well, especially the random positioning of each target individually. The body of the character was long to create in code, however with some good organizational implementations it’s practical function call was reduced to a single liner that is very easy to call and use effectively, thus reducing the usual clutter and difficulty in organizing such a complex system of parts.

* 1. Lessons learned

Not every objective was implemented perfectly and there were definitive lessons learned from this process, especially about the nature of Processing itself. Firstly, I created a targeting algorithm that would select the closest target and assign points while reducing the lifespan of the target to zero, thus removing it the next time the program checks to see if each target is still valid and should be displayed. This is buggy in the sense that there is a delay between the click of the user and the update of that specific target being removed from the canvas. This is a visual glitch and something that I believe is something to be learned about erasing specific objects from the canvas. Another objective that was not sufficiently completed as to the original specification was the animation of the arms of the character to follow the cursor, while the static character is in theory fine, it would be much more visually appealing if it moved and tracked the mouse cursor. A version two of this project would include that feature.

* 1. Highlights

There are some features of this project that I am very proud of, specifically the targeting algorithm as mentioned above. I ran into an issue where I would click and receive coordinates for the mouse, which could be used to align with the target and delete it. However, that posed a problem as the only coordinates available for each Target was the center point used to draw it out. Given these problems my solution was to use the coordinates of the mouse at the time of clicking to search through every target on the canvas and find the best match with the least difference between the coordinates of the mouse and the coordinates of the center of the target. Once selected this target was given a life of zero, to remove it from the canvas. I enjoyed this challenge and found it fun to find the solution, even though I realize with hindsight that the solution I created opens up the program to edge case issues. One of the edge cases is that no matter where you click on the screen, the closest target will inevitably be destroyed. This is a simple fix, by creating a maximum difference allowable before not deleting a target would fix this issue completely.

1. HISTORY DATES

REFERENCES