Danh Nguyen

Elton Li

Siwei Quan

Winter’s Tale

I believe we have complete the project according to the specifications set by the requirement document.

Project Iteration 1 ask the team to modify a skeleton graphic program to enable the program which allows a user to draw on a blank canvas using different types of brush predesignated by the requirement document. The predesignated tools are pen, calligraphy pen, highlighter, spray can, eraser and a creative pen design by the team. These brushes need to be able to draw on the canvas with different shades of color between red, green and blue.

To accomplish this task, the team implements the Tool class as a parent class that allows the different subclass of brushes to inherit its objects and function through polymorphism. The design of the Tool class includes the public functions: setColor and draw. It holds the private mask two-dimensional array that contains the predesignate shape and initiate color of the brush.

This design was discussed thoroughly during allotted lab hour and alternative design was proposed. There were only one other design that was being discussed and considered.

The first alternative design also includes a Tool class that has public functions: draw, setColor and a virtual function, setMask. The class will includes a separate Mask class that will initialize the mask size and set the color as the background color for the different tools. The different brushes will inherit the Tool class and can initialize their individual mask shape and size according to their specifications.

The main difference between the two is to have the mask as a separate class or includes it as a private object. The alternative design creates a class that was not needed and will bloat the codes of the program. The original thoughts were the separate Mask class will allow the programmer to set the size of the mask which allows more brushes to be implemented in the future. We conclude that having a separate Mask class is irrelevant since if we include a mask array object in the Tool class, we can declare a mask of any size in the different brush subclasses.

The Mask class was supposed to create an array with background color already filled in and we only need to fill in the color where it is needed but this thought process is also redundant since set the color and exclude parts of the mask that we don't want color in one iteration. We concluded that only the Tool class was needed for the project as it can hold an object that allows its subclasses to initiate the size, set color and overriding the default it sets up.

Another point of contention for the design is if we want the Tool class to draw and set color in the class. The alternative to this would be to have it create a virtual draw function to have its subclasses to initiate it through inheritance.

The alternative will have the Tool class declare a virtual draw and virtual setColor functions to have its subclasses instantiate the require functions. The subclasses will create the mask's size and color the shape specifically to its requirement.

The design was created because of the team's factors with programming which are unfamiliarity with best design practices, and unfamiliar with C code languages. We all have similar background starting in the Intro to Python class instead of C++ and are not too familiar with C++ thus there are some language unfamiliarity and how codes and classes interact with each other. This was not a big problem but it pairs with the fact that we do not fully grasp on to what is best practice to have the best design with classes, inheritance and continuous morphing object, the mask.

We conclude that drawing from point A to B is going to be the same for every brush because getting the pixels, its x and y coordinates, between two points is a repeat action that every brush going to make no matter its type. Initialize and setting color to the tool are one action as well for every tools. The design ends up being the Tool class initiate the draw and setColor functions which allows its subclass to use the function readily without the need to initiate it in its own class. The subclass can also override the function to perform something different if the need is there.