CSC 406 – Computer Graphics

Programming Assignment 01, Fall 2022

Tuesday, September 21st, 2022

Due date: Thursday, September 29th, 11:55pm

1 What this Assignment is About

1.1 Objectives

This is a simple "get your feet wet" assignment, so the objectives are not too lofty. You just have to do some simple drawing using Processing primitives.
This is an individual assignment. You are of course encouraged to discuss it with other students but I expect you complete it by yourself.

1.2 Handouts

There is no code handout for this assignment.

2 Version 1

2.1 Draw a portrait

Start from a picture of a person. It doesn't matter who that person is. Don't forget to submit the model picture together with your code so that we can evaluate the quality of your drawing.

You are going to use the—limited—set of OpenGL drawing primitives to draw the portrait. To do this, you will probably want to write a few utility functions (e.g. to draw a colored disk or an ellipse)[1].

2.2 Implementation constraints

2.2.1 OOP

This semester, we are going to do a lot of OOP (Object Oriented Programming), and we will start with this assignment. You are going to implement your portrait as a class. At a minimum, your portrait class should store instance variables for the coordinates of the center of the portrait. I don't specify what particular data structures you should be using for your instance data, but I recommend that you use a floating point rather than integral format for coordinates.

[1]What I really mean here is that you absolutely should write such functions, and probably will get penalized for bad coding style if you don't.

2.2.2 Class details

Naturally, your instance variables should be private, and so you will need to provide at least one constructor for your class. As we saw in class, you should also declare as `delete` the standard constructors and operators that you don't plan to implement. You may have as many public and private functions as you want. I only impose one public instance function:

```
void draw(void) const;
```

This is the function that will be called to render an instance of a portrait object.

For the rest of this document, I will use the following reference head:

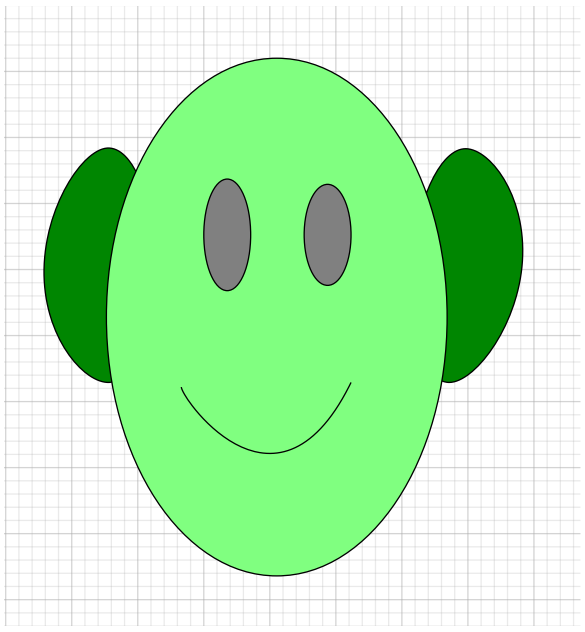

Figure 1: Example of a—poor quality—-portrait.

Hopefully, your portrait will look better than this.

2.3 The complete program

This version of the program (all the source and header files) should be in its own folder named `Version1` (no space). Your program should create a few (say, about half a dozen) instances of your portrait class in the main function, and render them in the display function, by calling the `draw` function of each object.

2.4 Extra credit 1 (5 points)

Store "scale" information in your class, so that you can create an multiple instances of your portrait with different sizes, as shown in Figure 2.

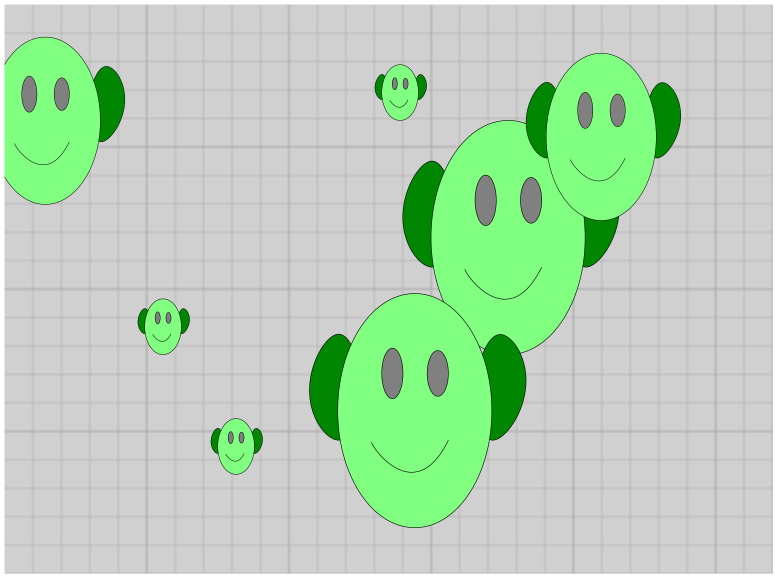

Figure 2: A few random heads.

2.5 Extra credit 2 (up to 5 points)

Store color information in your class, so that you can create instances of your portrait with different colors for different part of the portrait (face, eyes, hair, etc.).

2.6 Extra credit 3 (3 points)

Add "orientation" data to your class so that you can create instances of your portrait with different orientations.

Note: Remember that in OpenGL, angles sent to the `glRotatef` and `glRotated` functions are expressed in degree, while the functions of the math library, `cos` and `cosf`, `sin` and `sinf`, etc., work with radians.

3 What to do: Part II – Draw the Portraits one at a Time

3.1 Isn't that what I was already doing?

Yes, in the `draw()` function you draw each portrait, one after the other, by calling your portrait drawing function. What I am saying here is that you are now only going to draw one portrait *at all* in your `draw()` function. This section of the assignment, in addition to being the starting point to developing some pretty cool effects in the future, will also give you some ideas about the way we are going to implement animation in Processing.

3.2 Move the call to `background()` out of the draw function

Right now, this call to `background()` at the beginning of the `draw()` function means that you erase the entire display every time the frame gets redrawn (which, as I told you, is typically 60 times per second by default). If you simply move this call to the `setup()` function, right after you have created your window, then the frame will not get erased anymore.

All you have to do now is generate random parameters for a new portrait directly in your `draw()` function (preferably through a call to a custom function) and draw that random portrait. You should see the frame quickly fill with multiple copies of your portrait.

Note: Of course, in this version of the program, you don't need the array of random portrait parameters.

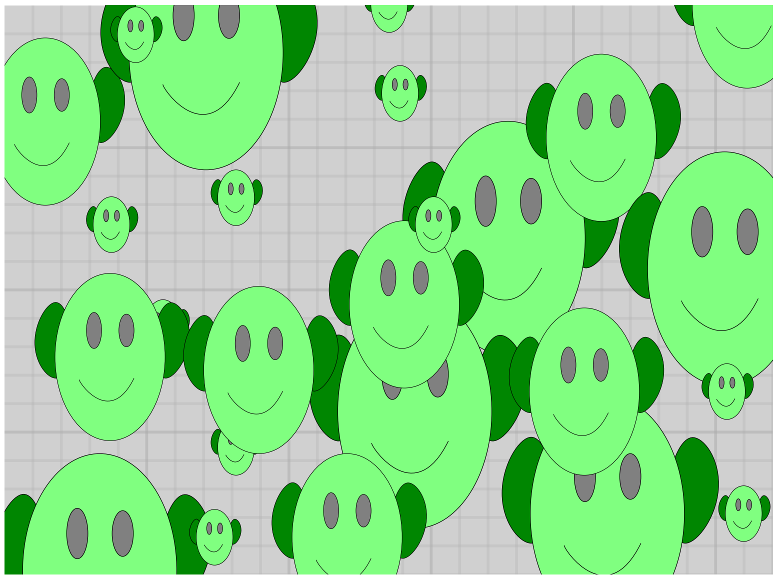

Figure 3: A few random heads.

4 What to Hand In

4.1 Report

You should provide a brief report explaining the design decisions you made for your portrait class.

4.2 Folder organization

Your project folder should be named `Prog01`. In addition to the report, it should contain a sub-folder for each version of the program that you implemented. Each version subfolder should contain all the—well-commented–source and header files required to build the project, as well as a "read me" text file telling how to build and run your program.

4.3 If you have spare time: Doxygen

For the next assignment, I am going to request that all code you submit has complete Doxygn-style documentation. For many years, C++ programmers secretly envied their Java-programming colleagues who could produce good-looking code documentation through the javadoc system.

Javadoc is the documentation generation component of Java. If you have never used Javodoc, or have simply forgotten how to write Javadoc-style comments, you can consult (from most exhaustive to most effective for this course's purpose):

- The Javadoc reference: `http://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html`
- The Wikipedia page: `http://en.wikipedia.org/wiki/Javadoc`
- A simple tutorial: `http://www.mcs.csueastbay.edu/~billard/se/cs3340/ex7/javadoctutorial.html`

To return to C++ programmers, they envied on and on, until a brave soul came up with Doxygen, which interprets javadoc-style comments for a multitude of programming languages (C, C++, Python, Perl, etc.). So if you have a bit of free time right now, you may consider installing Doxygen (and some dependencies such as Graphviz and/or dot) and taking a look at the comment style, so that you don't get overwhelmed when this part of the assignment becomes mandatory.

For this course, you only need to provide the description field, and the list of parameters, the return information, and the list of exceptions when appropriate.

5 Grading

If you submit code that could not possibly compile (contains obvious syntax errors) your assignment will not be graded and you will get a grade of 0. For all others:

- the code does what it is supposed to do: 40%
- good code design: 10%
- comments: 15%
- readability of the code (incl. indentation): 15%
- proper identifier names: 10%
- report: 10%