CSCE 441 - Computer Graphics

Programming Assignment 1

Deadline: Sep. 9th (11:59 pm)

1 Goal

The goal of this assigment is to become familar with OpenGL and event-driven programming and mouse and keyboard interaction using GLFW.

2 Starter Code

The starter code can be downloaded from here.

3 Task 1

Visit the toturial from here to set up OpenGL on your machine. You will need to install some libraries and get your system set up to process OpenGL. It is important to note that you will be using the process outlined there in all future programming assignments; it includes instructions for which directories to submit with your assignments, and how to get each assignment started. You should be sure you can run the test code provided at that link and are clear on the process for starting new programs in the future.

4 Task 2

Download the starter code and go through the process to set it up on your system (you will need to follow the process similar to the test code in the tutorial). Once you set up the code, run the program. You should see blank window. If you click in the window and drag the mouse, you should see the location of the mouse printed. If you press a key, you should see a message printed.

Look through the code to see how it works. Here are a few points regarding the code:

- The `main` function first initializes the window and events. Then it displays the "framebuffer" and checks for the events to see if there is any update from the inputs. This process is repeated until the window is closed.
- Overall, the program keeps track of a "framebuffer" that is an array of pixel colors (the array is defined at the top and is called `frameBuffer`). You will set framebuffer values in the program.
- Calling `ClearFrameBuffer` will set the framebuffer to black.
- Calling `SetFrameBufferPixel` will set a particular (x, y) pixel to a color (R, G, B). The RGB color values should range from 0 to 1. The pixel values should range from 0 to the width/height minus 1.
  - Note that this command uses the upper left corner as (0,0).
  - The framebuffer structure is simple (just a 3D array of floats) so you can access the framebuffer directly if you wish – e.g., to read values. However, realize that the data in the framebuffer is not stored in the order you might expect – if you choose to access it directly, be sure you understand how it is organized!
- Calling `Display` will show the framebuffer on the screen.
- The `MouseCallback` is used to handle mouse clicks. The `CursorPositionCallback`, on the other hand, is used to get the position of the mouse. Moreover, `CharacterCallback` is called when a key is pressed on the keyboard. You should use these functions to handle the input from mouse and keyboard.

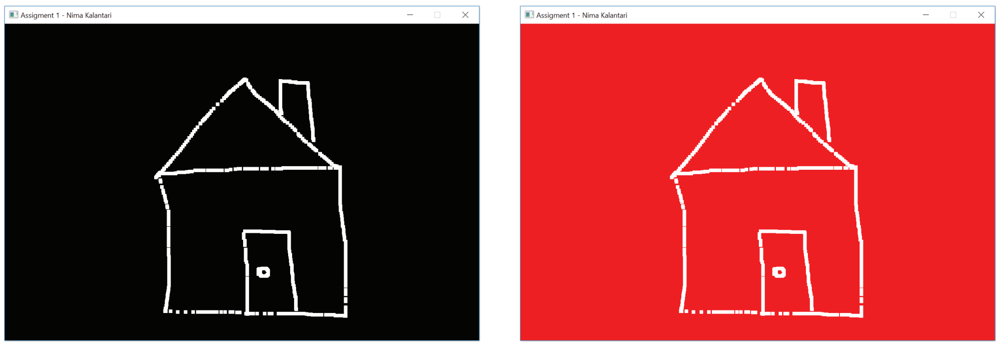

Figure 1: A drawing with the initial background color is shown on the left. On the right, the result of changing the background color to red by pressing Shift + '1' is shown.

You are encouraged to play around with the code (modify it to do different things for different key presses and/or for different mouse clicks) to ensure you understand the basic process of how this code works, before working on the actual assignment.

5 Task 3

You should write a program that allows you to paint on the window. To do so, you have to implement the followings:

- Make the window have a width to height ratio of 3:2 and a width of at least 900 pixels. Also change the title to "Assignment 1 - <Your Name>".
- When the user clicks the left button of the mouse and drags it across the screen, the program should draw a rectangle from $(-s, -s)$ to (s, s) centered at the current mouse position. The variable s controls the size of the brush and user should be able to increase or decrease it by a factor of 2 by pressing '+' and '-', respectively. The size s should be restricted to be between 1 and 128.
- The user should be able to change the brush color by pressing keys 0 - 7. The colors should correspond to the binary version of the numbers, e.g., 5 $\rightarrow$ 101 $\rightarrow$ magenta and 3 $\rightarrow$ 011 $\rightarrow$ cyan.
- The user should be able to change the background color by pressing keys 0 - 7, while holding the shift key. Again the colors should correspond to the binary version of the numbers. Note that, by pressing Shift + (one of the keys 0 - 7) the background color should automatically change, but the drawings should stay unchanged (see Fig. 1).
- Clicking the right button of the mouse should reset the screen to background color by clearing all the drawings.
- **[Extra]** Allow the user to change the brush shape between rectangle and circle by pressing 'b'.
- **[Extra]** Implement the spray paint brush. Instead of painting all the pixels within the brush shape, you have to randomly select a percentage of the pixels and paint them. You can use the `rand()` function to do so.

6 Deliverables

Please follow the instruction below or you may lose some points:

- You should include a README file that includes the parts that you were not able to implement, any extra part that you have implemented, or anything else that is notable.

- Your submission should contain folders “src” as well as “CMakeLists.txt” file. You should not include the “build” folder.
- Zip up the whole package and call it “Firstname_Lastname.zip”. Note that the zip file should extract into a folder named “Firstname_Lastname”. So you should first put your package in a folder called “Firstname_Lastname” and then zip it up.

7 Ruberic

Total credit: [50 points]

[05 points] - Setting the correct window size and title
[10 points] - Implementing the rectangular brush and correct placement of brush at mouse position
[05 points] - Changing the brush size
[05 points] - Changing the brush color
[20 points] - Changing the background color
[05 points] - Clearing the screen and setting it to background color

Extra credit: [10 points]

[05 points] - Implementing a circle brush
[05 points] - Implementing a spray paint brush

8 Acknowlegement

The toturial is provided by John Keyser. Some of the text is based on John Keyser’s assignmnet.