Programming Assignment 4: A 2D paint and draw program

## DATE DUE: December 11, 2017

## Points possible: 25 points

## CS-116A: Introduction to Computer Graphics

## Instructor: Rob Bruce

# OVERVIEW

For this assignment, you will be creating a simple paint and draw program in C or C++ with the OpenGL Utility Toolkit (GLUT). The specifications are outlined below.

# SPECIFICATIONS

# 1. The program must draw the following shapes:

# Line

# Rectangle

# Ellipse

# Bezier curve

2. Your program must draw rectangles and ellipses in filled or outline (unfilled) form. Bezier curves and lines should NOT be drawn in filled or outline form. Please do NOT include menu options to draw Bezier curves or lines as filled/outline.

3. Users should be able to select from a palette of 8 different colors when drawing a line, rectangle, ellipse, or Bezier curve. You may choose any colors for your palette as long as each color is visible on the paint canvas (the background of your screen). Please do NOT use black as a color choice on a black background canvas or white on a white background canvas.

4. Use the left mouse button for navigation and selection in the paint program drawing menu.

5. Use the right mouse button to select a control point (mouse cursor position) when drawing a line, rectangle, ellipse, or Bezier curve.

6. Bezier curves are drawn through a series of four mouse clicks. Bezier curves should be 1 pixel thick.

7. Lines are drawn through a series of two mouse clicks. Lines should be 1 pixel thick.

8. Please use white or black as the background for your drawing canvas.

9. You may use verbs such as "add" or "draw" in your menu items. For example, you may use "add rectangle" when drawing a shape. Alternately, you may use "draw rectangle" as a menu item instead

10. Your program must allow the user to draw at least twenty shapes (any combination of ellipses, rectangles, lines, or Bezier curves).

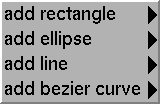
# EXAMPLE 1: DRAW OUTLINE OF RECTANGLE IN ORANGE.

# Here's an example of a mouse-driven menu that conforms to the above specification.

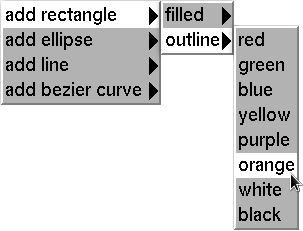
# 1. User clicks the left mouse button of a multi-button mouse:

Computer mouse (user interface) indicating left mouse clicked.


2. The program displays the following four sub-menus: "add rectangle", "add ellipse", "add line", and "add bezier curve".



3. The user wants to draw the outline (i.e. unfilled) of a rectangle in orange. The program should display the following:



4. The user moves the mouse somewhere on the screen and clicks the right mouse button of a multi-button mouse. This represents the starting corner for the rectangle.

Computer mouse (user interface) indicating right mouse button clicked.


# 5. The user moves the mouse somewhere on the screen and clicks the right mouse button again. This represents the ending corner for the rectangle.

# Computer mouse (user interface) indicating right mouse button clicked.

# 6. The program then draws the outline of a rectangle in orange:



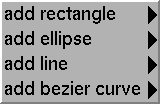
# EXAMPLE 2: DRAW FILLED RECTANGLE IN YELLOW.

Here's a second example of a mouse-driven menu that conforms to the above specification.

1. The user clicks the left mouse button of a multi-button mouse:

Computer mouse (user interface) indicating left mouse clicked.


2. The program displays the following four menu items: "add rectangle", "add ellipse", "add line", and "add bezier curve".



3. The user wants to draw a filled rectangle in yellow. The program should display the following menu:

Screenshot of hierarchical menu with "add rectangle" selected followed by sub-menu with "filled" selected followed by sub-menu with "yellow" selected.


4. The user moves the mouse somewhere on the screen and clicks the right mouse button of a multi-button mouse. This represents the starting corner for the rectangle.

Computer mouse (user interface) indicating right mouse button clicked.


5. The user moves the mouse somewhere on the screen and clicks the right mouse button again. This represents the ending corner for the rectangle.

Computer mouse (user interface) indicating right mouse button clicked.


6. The program then draws a filled rectangle in yellow:



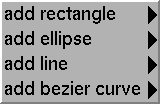
# EXAMPLE 3: DRAW OUTLINE OF AN ELLIPSE IN GREEN.

Here's a third example of a mouse-driven menu that conforms to the above specification.

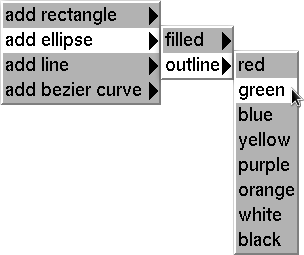
1. The user clicks the left mouse button of a multi-button mouse:

Computer mouse (user interface) indicating left mouse clicked.


2. The program displays the following four menu items: "add rectangle", "add ellipse", "add line", and "add bezier curve".



3. The user wants to draw the outline of an ellipse in green. The program should display the following:



4. The user moves the mouse somewhere on the screen and clicks the right mouse button of a multi-button mouse. This represents the center of the ellipse.

Computer mouse (user interface) indicating right mouse button clicked.


5. The user moves the mouse somewhere on the screen and clicks the right mouse button again. This represents the major radius (radius along major axis).

Computer mouse (user interface) indicating right mouse button clicked.


6. The program then draws the outline of an ellipse in green:



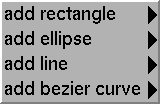
# EXAMPLE 4: DRAW FILLED RED ELLIPSE.

Here's a fourth example of a mouse-driven menu that conforms to the above specification.

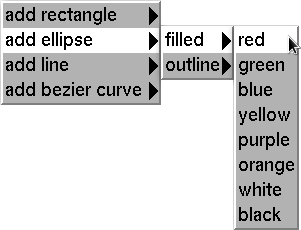
1. The user clicks the left mouse button of a multi-button mouse:

Computer mouse (user interface) indicating left mouse clicked.


2. The program displays the following four menu items: "add rectangle", "add ellipse", "add line", and "add bezier curve".



3. The user wants to draw a filled red ellipse. The program displays the following:



4. The user moves the mouse somewhere on the screen and clicks the right mouse button of a multi-button mouse. This represents the center of the ellipse.

Computer mouse (user interface) indicating right mouse button clicked.


5. The user moves the mouse somewhere on the screen and clicks the right mouse button again. This represents the major radius (radius along major axis).

Computer mouse (user interface) indicating right mouse button clicked.


6. The program then draws a filled red ellipse:



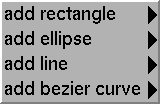
# EXAMPLE 5: DRAW PURPLE LINE.

Here's a fifth example of a mouse-driven menu that conforms to the above specification.

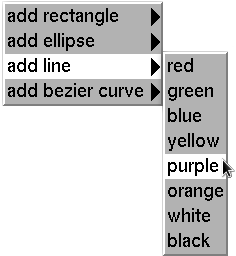
1. The user clicks the left mouse button of a multi-button mouse:

# Computer mouse (user interface) indicating left mouse clicked.

2. The program displays the following four menu items: "add rectangle", "add ellipse", "add line", and "add bezier curve".



3. The user wants to draw a purple line. The program should display the following:



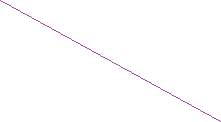
# 4. The user moves the mouse somewhere on the screen and clicks the right mouse button of a multi-button mouse. This represents the starting point for the line.

Computer mouse (user interface) indicating right mouse button clicked.


5. The user moves the mouse somewhere on the screen and clicks the right mouse button again. This represents the ending point for the line.

Computer mouse (user interface) indicating right mouse button clicked.


6. The program then draws a purple line that is 1 pixel thick:



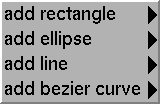
# EXAMPLE 6: DRAW BLUE BEZIER CURVE.

Here's a sixth example of a mouse-driven menu that conforms to the above specification.

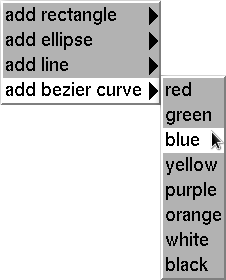
1. The user clicks the left mouse button of a multi-button mouse:

# Computer mouse (user interface) indicating left mouse clicked.

2. The program displays the following four menu items: "add rectangle", "add ellipse", "add line", and "add bezier curve".



3. The user wants to draw a blue Bezier curve. The program should display the following:



4. Upon clicking the left mouse button in step 3 (above), the user has indicated they want to draw a blue Bezier curve one pixel thick. Now we must retrieve four control points from the user: the beginning point where the Bezier curve will start, the second control point, the third control point, and the fourth control point where the curve will end. The first and last control points will be on the curve while the second and third points will not be on the curve. The user moves the mouse somewhere on the screen and clicks the right mouse button of a multi-button mouse. This represents the first point of the Bezier curve.

Computer mouse (user interface) indicating right mouse button clicked.


5. The user moves the mouse somewhere on the screen and clicks the right mouse button again. This represents the second point of the Bezier curve.

Computer mouse (user interface) indicating right mouse button clicked.

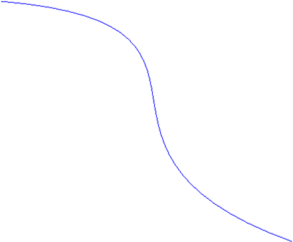

6. The user moves the mouse somewhere on the screen and clicks the right mouse button again. This represents the third control point of the Bezier curve.

Computer mouse (user interface) indicating right mouse button clicked.


7. The user moves the mouse somewhere on the screen and clicks the right mouse button one last time. This represents the fourth control point of the Bezier curve.

Computer mouse (user interface) indicating right mouse button clicked.


8. Immediately after the user clicks the right mouse button a fourth time in step 7 above, a 1-pixel width Bezier curve is drawn in blue:



# PLEASE INCLUDE THE FOLLOWING WITH YOUR PROGRAM

* Please submit your program as a text file labeled “Programming Assignment 4” to Canvas dropbox. I only need the source code. I will build the program myself.
* Please name your programming assignment file with the following naming convention:
  + **lastname\_firstname\_programming\_assignment\_4.c** (for c programs)
  + **lastname\_firstname\_programming\_assignment\_4.cpp** (for c++ programs)

For example, my name is Rob Bruce. If I write my program in the C programming language, my programming assignment should be named: bruce\_rob\_programming\_assignment\_4.c

* Please include your name as a comment at the beginning of your program like this:

**/\* Rob Bruce \*/**

or this:

**// Rob Bruce**

* I usually do not need a Makefile to build your programs; however, in the unlikely event that I cannot build your assignment, I will send an email inquiry as appropriate.
* I expect your program to build without syntax errors.
* I expect your program to execute without run-time errors.

# GRADING

Points will be deducted for any program that does not meet the specification as outlined above.

# LATE ASSIGNMENTS

Assignments submitted after the due date will be worth, at most, 12.5 points. Additional points will be deducted for programs not meeting the specification as outlined above.