Introduction

In this assignment, you will start to introduce a graphic interface to the semester's project, using GLUT and OpenGL. The next assignment will complete the process.

Assignment

Download the example code and put it into a new directory gui-src that will be a sibling to the src folder. The Makefile should link code from the src folder into the glut_main program in the gui-src. This starter code will draw a Julia set image if the user enters the J key.

You will extend this code to add a few more actions.

Programming Requirements

GlutApp.{h,cpp}

Add the following methods to the class. Each of these is copies code that was in createJulia() and moving it to a method to make it reusable, or it does actions that are similar to those.

- void selectJulia(); Sets up the input and output streams, and calls takeAction with the “julia” choice.
- void selectMandelbrot(); Sets up the input and output streams, and calls takeAction with the “mandelbrot” choice.
- void configureComplexFractal(); Sets up the input and output streams, and calls takeAction with the “complex-fractal” choice.
- void configureGrid(int max); Sets up the input and output streams, and calls takeAction with the “grid” choice. Uses mHeight and mWidth for the height and width, and max for the maximum number.
- void juliaParameters(double a, double b); Sets up the input and output streams, and calls takeAction with the “julia-parameters” choice.
- void fractalPlaneSize(double x_min, double x_max, double y_min, double y_max); Sets up the input and output streams, and calls takeAction with the “fractal-plane-size” choice.
- void fractalCalculate(); Sets up the input and output streams, and calls takeAction with the “fractal-calculate” choice.
- void gridApplyColorTable(); Sets up the input and output streams, and calls takeAction with the “grid-apply-color-table” choice.

Update the following method:

- void createJulia(); Rewrite this method to use the methods created above to take the same actions.

Create the following methods:

- void createJulia2(); Creates a different Julia set that createJulia. Must use different Julia parameters and fractal plane size. Must also look interesting. Use the methods defined above, as appropriate.
- void createMandelbrot(); Creates a Mandelbrot set. Must look interesting. Use the methods defined above, as appropriate.
- void createMandelbrot2(); Creates a Mandelbrot set. Must be different than the other one. Must look interesting. Use the methods defined above, as appropriate.
- void createComplexFractal(); Creates a Complex Fractal. Use the methods defined above, as appropriate.
- void createComplexFractal2(); Creates a Complex Fractal. Must be different than the other one. Use the methods defined above, as appropriate.

glut_callback.cpp

Update the following functions.

- void keyboard_cb(unsigned char c, int x, int y); Add options for J, M, C, and C. Each of these will call a different method of the GlutApp class, similar to the J option causing createJulia() to be called. J calls createJulia2(), M calls createMandelbrot(), C calls createMandelbrot2(), C calls createComplexFractal(), C calls createComplexFractal2()
Update **src/Makefile**

No changes here: The following commands should work correctly.

- `make_hello` - builds the hello program
- `make_questions_3` - builds the questions_3 program
- `make_ascii_image` - builds the ascii_image program
- `make_image_file` - builds the image_file program
- `make_ppm_menu` - builds the image_file program
- `make_all` - builds all programs
- `make` - builds all programs (same as `make all`)
- `make_clean` - removes all .o files, and all executable programs

Update **gui-src/Makefile**

Should be able to use the file as is.

- `make_glut_main` - builds the application.
- `make_clean` - removes all .o files, and all executable programs

**Additional Documentation**

- [C++ Reference](#)
- [Examples from class](#)

**Show Off Your Work**

To receive credit for this assignment, you must

- complete the unit tests available in CodeGrinder (if available)
- use git to add, commit and push your solution to your repository for this class.

Additionally, the program must build, run and give correct output.

**Extra Challenges (Not Required)**

- Allow the user to design their own color tables.
- Allow the user to change the maximum number (max escape count).
- Allow the user to move the center point with the arrow keys.
- Other interesting user interactions.