CS 3005: Programming in C++

Graphic Interface Parts 1 and 2

Introduction

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

This is a double assignment. It's a lot of work. All documented here in one place.

Assignment

In this assignment, you will create a new application that uses the GLUT and Open GL to build a graphical user interface (GUI) for much of the functionality of the semester's project. To start, you'll download the starter code for the `glut_main` program from the course website, putting the code in the new diretory, `gui-src`, as a sibling of the `src` directory.

The starter code should build using its existing `Makefile`, and linking to the code in `src`.

You will extend this code to add additional functionality.

Programming Requirements

Create/Update `glut_main.cpp`

This file can remain unchanged from the starter code. Maybe you'll want to change the window size or the title.

Create/Update `gl_draw.cpp`

This file can remain unchanged from the starter code.

Create/Update `glut_app.{h,cpp}`

These files can remain unchanged from the starter code.

Create/Update `glut_callback.cpp`

Reconfigure `keyboard_cb`, `special_cb`, and `mouse_cb` to call the `GlutApp` class's member functions that do the actual work.

Create/Update `AppData.{h,cpp}`

These files should exist in the starter code. The `AppData` class provides all of the non-GUI data and functionality of the application.

Public Enumerations:

- `enum InteractionMode { IM_FRACTAL, IM_COLORTABLE };` Used to track whether to display the output image or the color table.
- `enum FractalMode { M_MANDELBROT, M_JULIA, M_COMPLEX };`

Data Members:

- `int mHeight;` The height of the display window, in pixels.
- `int mWidth;` The width of the display window, in pixels.
- `int mMaxNumber;` The maximum number of iterations for the escape counting.
- `InteractionMode mInteractionMode;` The current interaction mode.
- `FractalMode mFractalMode;` The current fractal mode.
- `int mNumColor;` The number of colors in the color table.
- `Color mColor1;` The first color in the color table.
- `Color mColor2;` The second color in the color table.

- `double mMinX;` The minimum X value in the complex plane.
- `double mMaxX;` The maximum X value in the complex plane.
- `double mMinY;` The minimum Y value in the complex plane.
- `double mMaxY;` The maximum Y value in the complex plane.
- `double mA;` The A parameter for the Julia set.
- `double mB;` The B parameter for the Julia set.
- `std::stringstream mInputStream;` The input stream to be used for sending communication to `takeAction()`.
- `std::stringstream mOutputStream;` The output stream to be used for receiving communication from `takeAction()`.
- `ActionData mActionData;` The action data.
- `MenuData mMenuData;` The menu data.
- `int mDebug;` The debug level.

Methods:

Several of these methods will run commands after configuring the input stream to hold the required information for the command to run correctly. The data needed to configure the input stream, will usually come from parameters passed into the method. These methods should first use `clearStreams()` to clear the input and output streams, and then configure the input stream to hold the information needed to run the command, then call `runCommand()` to run the command.

- `AppData(int height, int width);` - Constructor initializes the height and width. Also initializes the rest of the data members such that the default image constructed is an *interesting* Julia set. In the body of the constructor, be sure to `configureMenu()`, call `setGrid(new ComplexFractal)` on the `ActionData` data member, `setColorTable()`, and `createFractal()`.
- `void setSize(int height, int width);` Set the height and width.
- `int getHeight() const;` Returns the height.
- `int getWidth() const;` Returns the width.
- `PPM& getOutputImage();` Returns the output image from the `ActionData` data member.
- `ColorTable& getColorTable();` Returns the color table from the `ActionData` data member.
- `void createJulia1();` Uses the data members method to create a Julia pre-configured set image. Must be *interesting*.
- `void createJulia2();` Uses the data members method to create a Julia pre-configured set image. Must be *interesting*.
- `void createMandelbrot1();` Uses the data members method to create a Mandelbrot pre-configured set image. Must be *interesting*.
- `void createMandelbrot2();` Uses the data members method to create a Mandelbrot pre-configured set image. Must be *interesting*.
- `void createComplexFractal1();` Uses the data members method to create a ComplexFractal pre-configured image.
- `void createComplexFractal2();` Uses the data members method to create a ComplexFractal pre-configured image.
- `void clearStreams();` Clear the input and output streams, by resetting their flags, and setting their contents to the empty string.
- `void runCommand(const std::string& choice);` Call `takeAction()`. If `mDebug` is not 0, then display the `choice` and contents of the input stream to `std::cout` before calling `takeAction()`, and display the contents of the output stream to `std::cout` after calling `takeAction()`.
- `void selectJulia();` Run the “julia” command.
- `void selectMandelbrot();` Run the “mandelbrot” command.
- `void selectComplexFractal();` Run the “complex-fractal” command.
- `void configureGrid(int max);` Run the “grid” command.
- `void juliaParameters(double a, double b);` Run the “julia-parameters” command.
- `void fractalPlaneSize(double x_min, double x_max, double y_min, double y_max);` Run the “fractal-plane-size” command.
- `void fractalCalculate();` Run the “fractal-calculate” command.
- `void gridApplyColorTable();` Run the “grid-apply-color-table” command.

- `void setInteractionMode(InteractionMode mode);` Modifies the data member to store the current interaction mode.
- `InteractionMode getInteractionMode() const;` Returns the current interaction mode.
- `void setColorTable();` Uses the "set-color-table-size" and "set-color-gradient" commands to configure the color table. Uses data members to configure the size of the color table and the color gradient. The color gradient spans from the beginning to the end of the color table.
- `void decreaseColorTableSize();` If the number of colors is more than `10`, decrease the number of colors by dividing it by `1.1`. Uses `setColorTable()` and `gridApplyColorTable()` to update the output image.
- `void increaseColorTableSize();` If the number of colors is less than `1024`, increase the number of colors by multiplying it by `1.1`. Uses `setColorTable()` and `gridApplyColorTable()` to update the output image.
- `void randomColor1();` Randomly choose RGB values for color 1. Each RGB value is between 0 and 255. Uses `setColorTable()` and `gridApplyColorTable()` to update the output image.
- `void randomColor2();` Randomly choose RGB values for color 2. Each RGB value is between 0 and 255. Uses `setColorTable()` and `gridApplyColorTable()` to update the output image.
- `void zoomIn();` Decrease the size of the view window to `0.9` the size. Calculate `dx` as `(1.0 - 0.9)* (mMaxX - mMinX) / 2.0`. Add `dx` to `mMinX` and subtract it from `mMaxX`. Do similar for the `y` dimension. Does not recalculate the output image.
- `void zoomOut();` Increase the size of the view window to `1.1` the size. Calculate `dx` as `(1.0 - 0.9)* (mMaxX - mMinX) / 2.0`. Subtract `dx` from `mMinX` and add it to `mMaxX`. Do similar for the `y` dimension. Only do this zoom operation if it will not cause any of the plane values to go past -2.0 or 2.0. Does not recalculate the output image.
- `void moveLeft();` Move the view port to the left by the fraction `0.05`. Calculate `dx` as `(1.0 - 0.9)* (mMaxX-mMinX) / 2.0`. If `mMinX - dx` is at least `-2.0`, then subtract `dx` from `mMinX` and `mMaxX`. Does not recalculate the output image.
- `void moveRight();` Move the view port to the right by the fraction `0.05`, similar to `moveLeft()`, except add to `mMinX` and `mMaxX`.
- `void moveDown();` Like `moveLeft()`, but for the `y` dimension.
- `void moveUp();` Like `moveRight()`, but for the `y` dimension.
- `void setFractalMode(FractalMode mode);` Modifies the data member to store the current fractal mode.
- `FractalMode getFractalMode() const;` Returns the current fractal mode.
- `void increaseMaxNumber();` If the `mMaxNumber` is less than `2048`, increase it by multiplying by `1.1`. Does not recalculate the output image.
- `void decreaseMaxNumber();` If the `mMaxNumber` is greater than `11`, decrease it by dividing by `1.1`. Does not recalculate the output image.
- `void setAB(int x, int y);` If the `mFractalMode` is `M_MANDELBROT`, and the `mActionData` grid is a `ComplexFractal`, then set `mA` to `mMinX + x * delta_x`, and similar for `b` and `y`. `delta_x` is obtained from the dynamically cast `ComplexFractal` pointer with `getDeltaX()`. Does not recalculate the output image.
- `void resetPlane();` Sets `mMinX` and the other three data members to `-2.0` or `2.0`, as appropriate to make the default square. Does not recalculate the output image.
- `void createFractal();` Recomputes the output image. Uses `mFractalMode` to choose whether to `selectMandelbrot()`, `selectJulia()`, or `selectComplexFractal()`. For Julia, also calls `juliaParameters()`. Calls `configureGrid()`, `fractalPlaneSize()`, `fractalCalculate()`, and `gridApplyColorTable()` to calculate the output image. Uses data members for parameters to these functions.

Create/Update `GlutApp.{h,cpp}`

These files should exist in the starter code. The `GlutApp` class provides all of the GUI functionality of the application, using the `AppData` class to store information, and compute the output image.

Data Members:

- `AppData mData;` The data member stores the state of the application.

Methods:

- `GlutApp(int height, int width);` Initializes the data member.
- `void setSize(int height, int width);` Sets the `height` and `width` in the data member. Re-`createFractal()` the currently configured fractal.
- `int getHeight() const;` Returns the height from the data member.
- `int getWidth() const;` Returns the width from the data member.
- `void display();` Depending on the interaction mode in the data member, call one of the following: `displayOutputImage()` or `displayColorTable()`.
- `void displayOutputImage();` Displays the output image from the data member.

- `void displayColorTable();` Displays the color table from the data member. The color table should be displayed with the first color on the left of the window, and the last color on the right of the windows. Here's a rough description of one way to do it. For each row in the display, do the same thing. For each column in the display: calculate the index into the color table using: `i = column * color_table_size / width_of_display`. Use the `i`th color from the color table. Prepare each color channel (red, green, blue) for OpenGL by dividing by `255.0`, then use `glColor3d(red,green,blue);` to set the color. Finally, draw the screen pixel using `glVertex2i(column, row);`. Repeat this process for every pixel in the display.

- `bool keyboard(unsigned char c);` Based on `c`, the keyboard key pressed, call the correct functions in the data member. See the table of required functionality. Returns true if the display should be updated.

- `bool special(unsigned char c);` Based on `c`, the special key pressed, call the correct functions in the data member. See the table of required functionality. Returns true if the display should be updated.

- `bool mouse(int mouse_button, int state, int x, int y);` Based on `mouse_button`, `state`, `x`, and `y`, call the correct functions in the data member. See the table of required functionality. Returns true if the display should be updated.

Expected Functionality

At the end of this assignment, your `glut_main` program should have this functionality:

Key/Action	Method(s) called	Notes
		Initial image
start		Displays a Julia set different from others in methods.
		Pre-built configurations
'J'	`createJulia1()`	
'j'	`createJulia2()`	
'M'	`createMandelbrot1()`	
'm'	`createMandelbrot2()`	
'C'	`createComplexFractal1()`	
'c'	`createComplexFractal2()`	
		Interaction modes
'T'	`setInteractionMode()`	Color table interaction mode
't'	`setInteractionMode()`	Fractal interaction mode
		Fractal Modes
'b'	`setFractalMode()`, `createFractal()`	Mandelbrot mode
'n'	`setFractalMode()`, `createFractal()`	Julia mode
'F'	`setFractalMode()`, `createFractal()`	Complex fractal mode
		Color table operations
'>' or '.'	`increaseColorTableSize()`	The user can use either '>' or '.', your code must support both.
'<' or ','	`decreaseColorTableSize()`	The user can use either '<' or ',', your code must support both.
'r'	`randomColor1()`	Color table display mode only. Otherwise, do not do this action.
'R'	`randomColor2()`	Color table display mode only. Otherwise, do not do this action.
		Plane coordinate operations
'z'	`zoomIn()`, `createFractal()`	Zoom into the plane.

'Z'	`zoomOut()`, `createFractal()`	Zoom out of the plane.
left arrow	`moveLeft()`, `createFractal()`	Move the view port left in the plane.
right arrow	`moveRight()`, `createFractal()`	Move the view port right in the plane.
down arrow	`moveDown()`, `createFractal()`	Move the view port down in the plane.
up arrow	`moveUp()`, `createFractal()`	Move the view port up in the plane.
'R'	`resetPlane()`, `createFractal()`	Fractal display mode only. Otherwise, do not do this action.
left mouse button	`setAB()`, `setFractalMode()`, `createFractal()`	Fractal display mode and Mandelbrot fractal mode only. Otherwise, do not do this action. Set the A/B values, change the fractal mode to Julia, the create the fractal.
		Fractal calculation configuration
'+' or '='	`increaseMaxNumber()`, `createFractal()`	The user can use either '+' or '=', your code must support both.
'-' or '_'	`decreaseMaxNumber()`, `createFractal()`	The user can use either '-' or '_', your code must support both.
		Window configuration
resize		`setSize()` is called. It should recreate the fractal currently configured in the data.

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

- 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.