CS 3005: Programming in C++

Mandelbrot Set

Introduction

The Mandelbrot set is a mathematical set defined from a function, similar to and related to the Julia set. However, the Mandelbrot set is more widely know.

Finding Points in the Mandelbrot Set

For our purposes, this is a good enough definition of the Mandelbrot set for a function (x', y') = f(x, y, a, b), where (x, y), (a, b) and (x', y') are the coordinates of points in the 2 dimensional plane.

Take a point (a, b). Using x0 = 0, y0 = 0, a, and b as input to f(x, y, a, b), we receive a new point from the function.

```
x1, y1 = f(x0, y0, a, b)
```

Repeat that process by using the output of the previous call of the function as the input to the current call. We could repeat up to `n` times:

```
x2, y2 = f(x1, y1, a, b)
x3, y3 = f(x2, y2, a, b)
x4, y4 = f(x3, y3, a, b)
...
xn-1, yn-1 = f(xn-2, yn-2, a, b)
xn, yn = f(xn-1, yn-1, a, b)
```

For large enough values of n, the resulting point at xn, yn will fall into one of two categories:

1- The point will still be close to the origin (0,0) of the plane. We define close to mean a distance less than or equal to 2.

2- The point will be far from the origin. We define far to mean a distance greater than 2.

If the point is close to the origin, it is part of the Mandelbrot set for f(x,y,a,b). Otherwise it is not part of the Mandelbrot set. A point that is not part of the Mandelbrot set "escapes" at the first iteration where the new (x,y) point is a distance of more than 2 from the origin. For points that escape, we will want to remember the "escape count", or which iteration it escaped.

From Functions to Images

So, where do the interesting pictures come from?

We choose an actual function for `f(x,y,a,b)`, then color each point in the plane based on how far it is from being in the Mandelbrot set for the selected function. "How far" is the "escape count".

Actually, there are infinitely many points in the plane, so we can't do every point. Instead, we select a regular grid of points each representing the points near it in the plane. For each point in the grid:

- Calculate its `a,b` value, from its position in the grid.
- Use x0 = 0 and y0 = 0.
- Iterate the application of `f(x,y,a,b)` up to `n` times.
- If the result becomes far away, remember the iteration number when it first escaped (became far away from the origin). We call this the "escape value". Points with lower escape values are further from the Mandelbrot set than those with higher escape values.
- If the result doesn't become far away (escape) within `n` iterations, assume it did not escape and remember `n` as the escape value. These points are assumed to be part of the Mandelbrot set for `f(x,y,a,b)`.

Now, we have a set of integers (the escape values) that represent how close each of the points in the grid are to the Mandelbrot set for `f(x,y,a,b)`. Larger numbers mean they are closer to the Mandelbrot set.

Since the points selected are in a regular grid, we can pair each grid point with a pixel in a rectangular image. We assign the color of pixel based on the escape value of the corresponding grid point. All pixels

associated with the same escape value will have the same color.

Defining a Regular Grid

See the description in the Julia set assignment.

Our Function f(x,y,a,b)

For our function `f(x,y,a,b)` we will use this definition:

```
x' = x*x - y*y + a
y' = 2*x*y + b
```

where, `a` and `b` are the coordinates of the original point in the plane. Note this is similar to, but different from the Julia set, where `a` and `b` where fixed values for an entire image calculation. Here, they depend on the point whose escape count you are calculating.

Assignment

In this assignment you will create a class to calculate and store a Mandelbrot set's escape values. Most of the work has already been done in the `NumberGrid` and `ComplexFractal` classes. In this assignment, you will create a new class `MandelbrotSet` class, inheriting from `ComplexFractal` and adding a few methods specific to the calculation of Mandelbrot sets.

You will also extend the `ppm_menu` program to add a few new commands.

The new commands required are:

- `mandelbrot`: Choose to make a Mandelbrot set.

Programming Requirements

The following files must be updated or created and stored in the `src` directory of your repository.

Create `MandelbrotSet.{h,cpp}`

These files will be used to declare and define the `MandelbrotSet` class.

No data members are required. Note that the `NumberGrid`'s maximum number will be used for the `MandelbrotSet`'s maximum escape count. Also note that `MandelbrotSet` inherits from `ComplexFractal`.

Methods:

- `MandelbrotSet( );` The default constructor, chain constructs via the `ComplexFractal` default constructor.
- `MandelbrotSet( const int& height, const int& width, const double& min_x, const double& max_x, const double& min_y, const double& max_y );` Passes arguments on to the `ComplexFractal` constructor.
- `virtual ~MandelbrotSet( );` Does nothing, empty code block.
- `virtual void calculateNextPoint( const double x0, const double y0, const double& a, const double& b, double& x1, double &y1 ) const;` Calculates `x1` and `y1` using the function described above for the Mandelbrot set.
- `int calculatePlaneEscapeCount( const double& a, const double& b ) const;` Repeatedly uses `calculateNextPoint` to find the escape count for the point at `(a,b)`. The first iteration where `x0,y0 = 0,0` causes `x1,y1` to equal `(a,b)` and this does not count as an iteration. (Look carefully at the formula to convince yourself that this is true.) You should still check if the point `(a,b)` has 'escaped' during the first iteration and you should return 0 as the count if this is the case.
- `virtual int calculateNumber( const int& row, const int& column ) const;` Uses other methods to find the plane coordinate of the pixel at `row`, `column` and to calculate the escape count. Returns the count. If `row`, `column` isn't valid, then returns `-1`.

Update/Add Functions in `image_menu.h` and `controllers.cpp`

Add or update the following function declarations to the header file and implementations to the `.cpp` file.

- `void setMandelbrotFractal( ActionData& action_data );` Use `setGrid()` to set `action_data`'s grid to a `MandelbrotSet` object allocated from the heap.
- `void configureMenu( MenuData& menu_data )` add the new actions with the names and descriptions listed below.

Table of New Commands

Command Name	Function Name	Description
mandelbrot	setMandelbrotFractal	Choose to make a Mandelbrot set.

Update `Makefile`

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

Additional Documentation

- C++ Reference
- Examples from class
- Sample Session Input File
- Julia set on Wikipedia
- Mandelbrot set on Wikipedia

Sample PPM Images

- Sample Output1
- Sample Output2
- Sample Output3

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.
- successfully pass all unit tests and acceptance tests

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

Extra Challenges (Not Required)

- Create classes that inherit from `MandelbrotSet` that have different `calculateNumber()` functions for calculating values. Add the ability to use them from the `imageMenu()`.
- Try other ways to modify the plane and parameters that would make it easier to create interesting images.