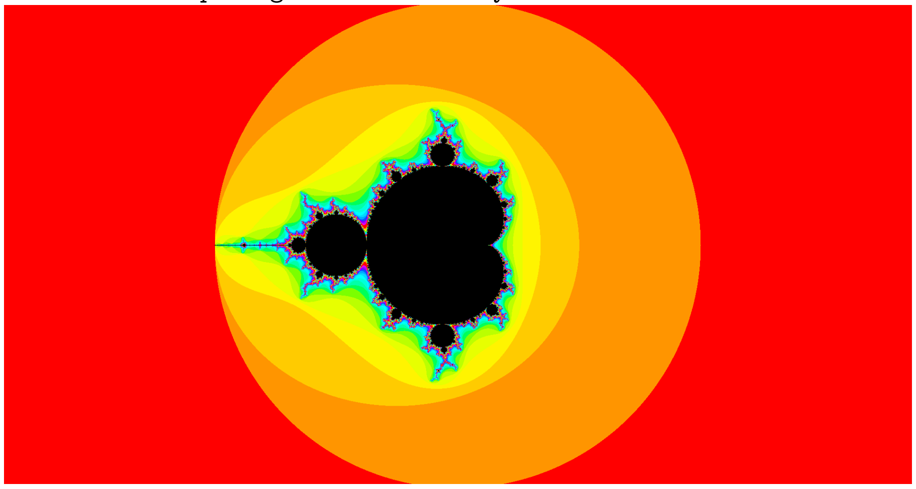
CS175 — Final Project

Max Snyder

10 May 2023

Figure 1: The Mandelbrot Set
`https://github.com/maxsnyder2000/mandelbrot`



1 What I Did

This write-up describes an interactive OpenGL graph of the Mandelbrot Set (Resource 3). The Mandelbrot Set is the set of complex numbers c such that repeated iterations of $z_n = z_{n-1}^2 + c$ (with $z_0 = 0$) does not cause $|z|$ to exceed a certain threshold. In this case, at most 200 iterations and threshold 2 are used.

1.1 Features

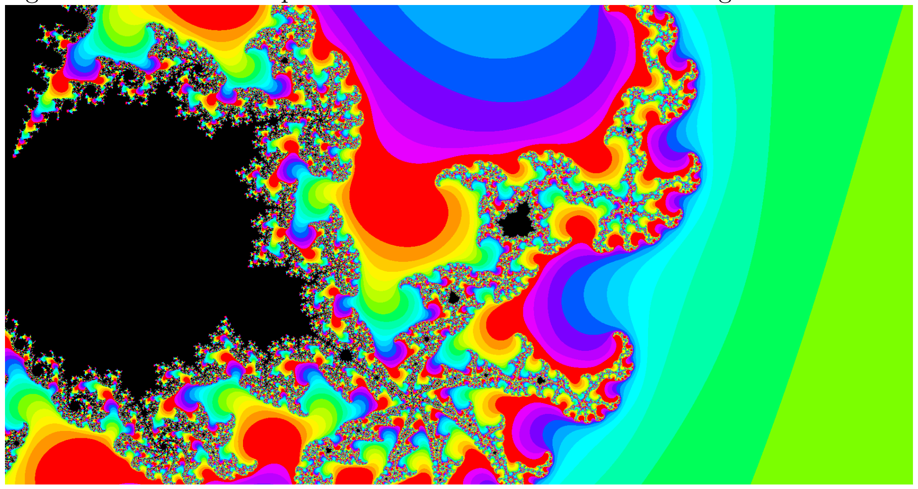
1.1.1 Navigation: Left / Right / Down / Up

The graphical display begins by plotting the Mandelbrot Set on a complex plane from -2 to 2 on the real (X) and imaginary (Y) axes, centered at $(0, 0)$. The user can shift the center left/right/down/up using the controls described below.

1.1.2 Animation and Zoom In / Out

In addition to translation-based navigation, the user can zoom in and out of the graph using the controls described below. Unlike the former type of navigation, the level of zoom is animated using physics-based acceleration (similar to bunny hairs). The zoom-in control can be thought of as the accelerator ("gas pedal"), and the zoom-out control as the decelerator ("gas pedal when in reverse gear").

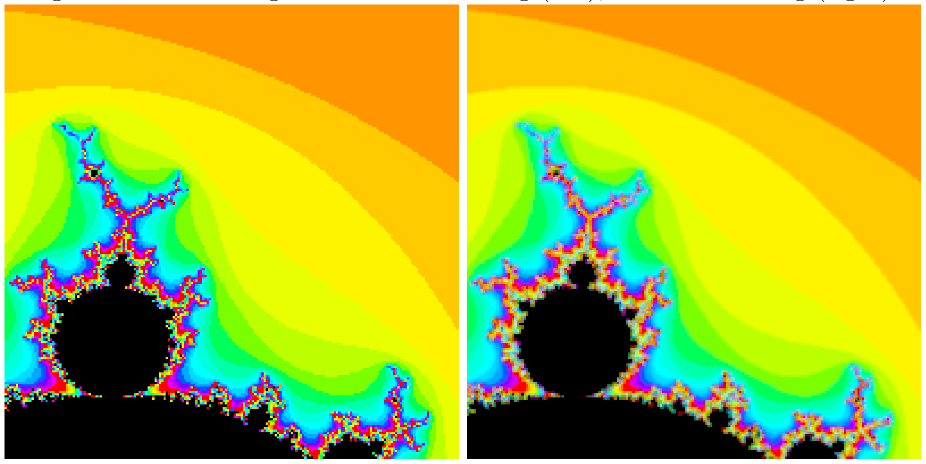
Figure 2: Animation: capture of the Mandelbrot Set after navigation and zoom.



1.1.3 Antialiasing

Antialiasing can be toggled on and off using the controls described below.

Figure 3: Antialiasing: without antialiasing (left); with antialiasing (right).



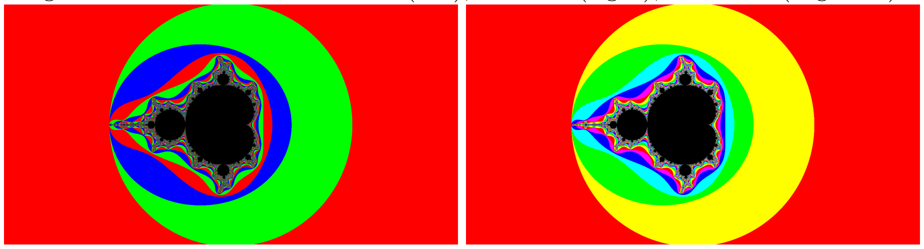
1.1.4 Color Palette

Every coordinate in the Mandelbrot Set is colored black. The choice of color for other coordinates is arbitrary, but it should be based on the number of iterations until $|z|$ diverges beyond the chosen threshold. The user can choose the number of colors N in the palette using the controls described below. The colors are determined by interspersing N points on the number line from 0 to < 6, where:

- 0 represents Red (red: 1, green: 0, blue: 0)
- 1 represents Yellow (red: 1, green: 1, blue: 0)
- 2 represents Green (red: 0, green: 1, blue: 0)
- 3 represents Aquamarine (red: 0, green: 1, blue: 1)
- 4 represents Blue (red: 0, green: 0, blue: 1)
- 5 represents Purple (red: 1, green: 0, blue: 1)
- 6 represents Red (red: 1, green: 0, blue: 0)

With 10 colors, for instance, the first colors are (1, 0, 0), (1, 0.6, 0), (0.8, 1, 0).

Figure 4: Color Palette: 3 colors (left); 6 colors (right); 20 colors (Figure 1).



1.2 Global Variables

- `static bool g_antialiasing = false;`

  If `false`, antialiasing is off. If `true`, antialiasing is on.

- `static float g_centerDelta = 0.1;`

  This represents the left/right/down/up stride during navigation.

- `static float g_centerX = 0.0;`

  This is the x-coordinate (real) of the center of the screen.

- `static float g_centerY = 0.0;`

  This is the y-coordinate (imaginary) of the center of the screen.

- `static int g_colors = 20;`

  This is the total number of colors in the palette.

- `static int g_framesPerSecond = 30;`

  This is the number of frames per second during zoom animation.

- `static double g_lastFrameClock = 0.0;`

  This is the last frame's clock time.

- `static int g_steps = 100;`

  This is the total number of steps in the Mandelbrot computation.

- `static int g_threshold = 2;`

  This is the threshold of the norm in the Mandelbrot computation.

- `static int g_windowWidth = 512;`

  This is the window width, in pixels.

- `static int g_windowHeight = 512;`

  This is the window height, in pixels.

- `static double g_wScale = 1;`

  This is the window scale, in case of Retina display.

- `static double g_hScale = 1;`

  This is the height scale, in case of Retina display.

- `static float g_zoom = -1.0;`

  This is the logarithm of the zoom scale.

- `static float g_zoomDamping = 0.99;`

  Every frame, `g_zoomDampening` is multiplied to `g_zoomVelocity`.

- `static float g_zoomDelta = 0.01;`

  This is the in/out stride during zoom animation.

- `static float g_zoomVelocity = 0.0;`

  Every frame, `g_zoomVelocity` is added to `g_zoom`.

1.3 Controls

- `GLFW_KEY_A`
  Toggle antialiasing on/off.
- `GLFW_KEY_LEFT`
  ("Navigate Left")
  Subtract `g_centerDelta / pow(2, g_zoom)` from `g_centerX`.
- `GLFW_KEY_RIGHT`
  ("Navigate Right")
  Add `g_centerDelta / pow(2, g_zoom)` to `g_centerX`.
- `GLFW_KEY_DOWN`
  ("Navigate Down")
  Subtract `g_centerDelta / pow(2, g_zoom)` from `g_centerY`.
- `GLFW_KEY_UP`
  ("Navigate Up")
  Add `g_centerDelta / pow(2, g_zoom)` to `g_centerY`.
- `GLFW_KEY_MINUS`
  Decrement `g_colors`.
- `GLFW_KEY_EQUAL`
  Increment `g_colors`.
- `GLFW_KEY_BACKSPACE`
  ("Zoom Out")
  Subtract `g_zoomDelta` from `g_zoomVelocity`.
- `GLFW_KEY_SPACE`
  ("Zoom In")
  Add `g_zoomDelta` to `g_zoomVelocity`.
- `GLFW_KEY_Q`
  Quit.

2 What I Tried

2.1 Features

2.1.1 Navigation: Left / Right / Down / Up

Initially, navigation used a fixed constant for stride, but this led to large strides when zoomed in, necessitating that the constant be divided by the zoom factor.

2.1.2 Animation and Zoom In / Out

Initially, the zoom value was the zoom scale exactly, rather than its logarithm, but this led to small zooms when zoomed in, necessitating use of the logarithm.

2.1.3 Antialiasing

Initially, antialiasing used the average of the pixel value with neighboring pixel values, but better quality was achieved using diagonal neighboring pixel values.

2.1.4 Color Palette

Initially, the number of colors was fixed at 6, but applying the concept of linear interpolation to color arithmetic enabled the colors to be modified on demand.

3 What I Learned

3.1 Antialiasing

Antialiasing does not really increase the visual quality of the experience.

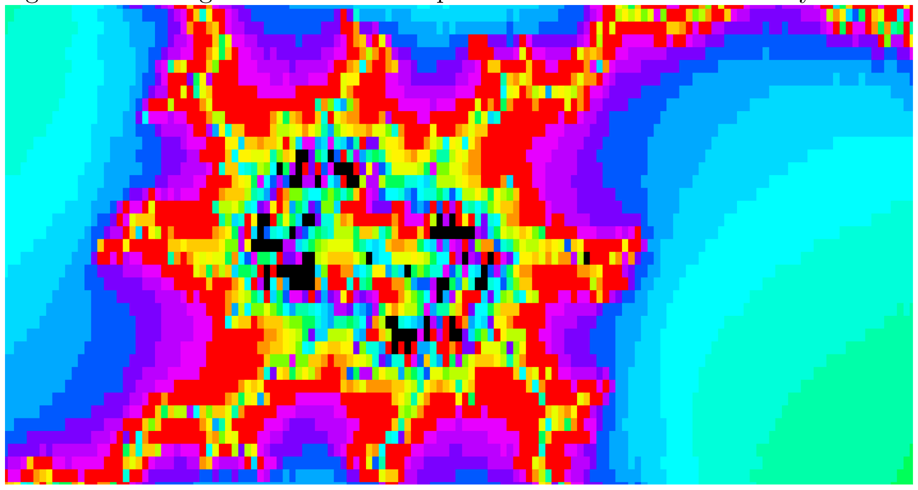
3.2 Framework Overhead

Using OpenGL directly is much faster than a game framework (Resource 2).

3.3 Floating-Point Arithmetic

Fragment shader computation leads to precision errors when sufficiently zoomed.

Figure 5: Floating-Point Arithmetic: precision errors when sufficiently zoomed.



4 What My Resources Were

1. `https://github.com/cs175/cs175-assignment1-maxsnyder2000`
2. `https://github.com/maxsnyder2000/SimpleGUI/blob/master/Mandelbrot%20Set/game.py`
3. `https://en.wikipedia.org/wiki/Mandelbrot_set`
4. `https://www.youtube.com/watch?v=pCpLWbHVNhk`
5. My roommate Liam, who inspired the idea of logarithmic zoom scale.