Mandlebrot Set

Ashvin

December 17, 2020

Contents

1 Introduction

Mandlebrot set is the set of complex numbers c such that starting from $z_0 = 0$ and applying:

$$z_{k+1} = z_k^2 + c$$

repeateadly,$\forall k > 0$, $|z_k| \leq 2$

2 Implementation Details

Drawing mandlebrot set is quite easy. All we need to do is map each pixel with a complex number(c) and repeat the iteration to a "Max" number. If $|z_k| \leq 2$ upto max iteration then we color the pixel black else we color it white, or for more fun we might use the iteration count itself for coloring. Real part of complex number corresponds to x-coordinate, and imaginary part corresponds to y-coordinate. x-coodinate is columns number of screen, and y is the row number, when we try to map (x,y) of a grid to rows and columns of a matrix. The screen in which we plot is a matrix, so we draw such that the center is (0,0) and also:

$$|c| > 2 \implies |z_1| > 2$$

$$\implies |c| \leq 2$$

for any chance of convergence. So, we have to scale our screen or else only very small portion of the screen will have any drawing and that is no fun. So apply two transformation to (x,y) of each pixel. We first shift the origin to center, then we scale it such that $\forall c \in$ transformed set, $|c| \leq 2$ i.e the full screen has the width of 4 i.e radius of 2.

2.1 Transformation

X = $(x - \frac{width}{2}) \times \frac{4}{width}$
Y = $(y - \frac{height}{2}) \times \frac{4}{width}$
Note: I have scalled equally in both directions to prevent distortion of image, and y axis is inverted, but this doesn't much affect our shape.

2.2 A bit of Complex algebra

Lets say $z_k = z_{k_r} + z_{k_i} i$ and $c = c_r + c_i i$ then

$$z_{k+1} = (z_{k_r}^2 - z_{k_i}^2 + c_r) + (2 z_{k_r} z_{k_i} + c_i) i$$

where $c_r = X$ and $c_i = Y$.

3 Setting up SDL2

To setup SDL2 in windows follow these instructions. Setting up SDL2 in linux follow this site.

4 Building

Simply:

```
1 make mandle.exe
```

5 Code

5.1 Headers and function initialization

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <SDL2/SDL2_gfxPrimitives.h>
#include <SDL2/SDL.h>


#define WIDTH 1200
#define HEIGHT 650
#define MAX_ITER 50


void draw(SDL_Renderer **,int);

int main(int argc, char *argv[])
{
  if (SDL_Init(SDL_INIT_VIDEO))
    {
        printf ("SDL_Init Error: %s", SDL_GetError());
        return 1;
    }
    SDL_Window *window = NULL;
    SDL_Renderer *renderer = NULL;

    window = SDL_CreateWindow("Mandelbrot Set", SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERE
    if (window == NULL)
    {
        printf ("SDL_CreateWindow Error: %s", SDL_GetError());
        SDL_Quit();
        return 2;
    }

    renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED | SDL_RENDERER_PRESENT
    if (renderer == NULL)
    {
        SDL_DestroyWindow(window);
        printf ("SDL_CreateRenderer Error: %s", SDL_GetError());
        SDL_Quit();
        return 3;
    }

    SDL_Event event;
    int quit = 0;

    //Factor is a random number that will spice things up for the image.
    int factor = 1;

```

```
    while (!quit){
        while (SDL_PollEvent(&event))
        {
            if (event.type == SDL_QUIT)
                quit = 1;
        }

        //Clear using white color
        SDL_SetRenderDrawColor(renderer, 255, 255, 255, SDL_ALPHA_OPAQUE);
        SDL_RenderClear(renderer);

        //Draw pixels on the renderer
        draw(&renderer,factor);
        SDL_RenderPresent(renderer);

        //Increaseing Factor by 10 each time.
        factor+=1;

    }

    //free resources
    if (renderer){
      SDL_DestroyRenderer(renderer);
    }
    if (window) {
      SDL_DestroyWindow(window);
    }

    SDL_Quit();
    return 0;
}
```

5.2 Main logic

```
void draw(SDL_Renderer **renderer,int factor){
   for (int x = 0; x < WIDTH; x++) {
      for (int y =0;  y < HEIGHT; y++) {
      //Transforming and scaling such that origin is center and  radius of 2 around it. Scali
        float c_real = (x - WIDTH/2.0) * (4.0/WIDTH);
        float c_img = (y-HEIGHT/2.0) * (4.0/WIDTH);

        float z_real = 0;
        float z_img = 0;
        int iter_count = 0;
        while (pow(z_real,2)+pow(z_img,2) <= 4 && iter_count < MAX_ITER) {
          float temp_real = pow(z_real,2)-pow(z_img,2)+c_real;
          float temp_img = 2*z_real*z_img + c_img;
          z_real = temp_real;
          z_img = temp_img;
          iter_count++;
```

```
17          }
18
19          //If any number exits before reaching MAX_ITER then, it is not in the set. So colour
20          if (iter_count == MAX_ITER) {
21            //printf("SELECT %.2f %.2f %d %d\n",c_real,c_img,x,y);
22            //Draw with black
23            SDL_SetRenderDrawColor(*renderer, 0,0, 0, SDL_ALPHA_OPAQUE);
24            SDL_RenderDrawPoint(*renderer,x,y);
25          }else{
26             //Draw with custom shade
27            SDL_SetRenderDrawColor(*renderer, iter_count*factor*3,iter_count*factor, iter_count
28            SDL_RenderDrawPoint(*renderer,x,y);
29          }
30        }
31    }
32  }
```