EL2310 Project 2 – C
SCR15
Martin Isaksson, misakss@kth.se 10/9/2015

Project 2

Skeleton

In the project a skeleton code was given, that generates a screen and also includes a function so that it's easy to color a pixel in the picture.

Mandelbrot set

The Mandelbrot set is a set of complex numbers and is defined as

$$M = \{c \in \mathbb{C} \mid \lim_{n\to\infty} Z_n \neq \infty\}$$

, where

$$Z_0 = 0, Z_{n+1} = Z_n^2 + c.$$

So the Mandelbrot set is a set of complex numbers where the value of the function Z_n is bounded, i.e. that it's not infinite as n approaches infinity.

When the Mandelbrot set is generated, the function $Z_n = Z_x + i*Z_y$ is complex and $c = c_x + i*c_y$ is complex. Therefore Z_{n+1} can be written as

$$Z_{n+1} = Z_x^2 - Z_y^2 + c_x + i(2Z_xZ_y + c_y)$$

$$Re\{Z_{n+1}\} = Z_x^2 - Z_y^2 + c_x$$
$$Im\{Z_{n+1}\} = 2Z_xZ_y + c_y$$

To generate c_x and c_y, the edges of the picture are defined as c_{xmax}, c_{xmin}, c_{ymax} and c_{ymin}. The x-direction is the width and the y-direction is the height. By interpolating between these four values, c_x and c_y are generated. Every value of c_x corresponds to a position X in width in the picture and every value c_y corresponds to a position Y in height in the picture

$$c_x = c_{xmin} + X\frac{c_{xmax} - c_{xmin}}{totalX - 1}$$

$$c_y = c_{ymin} + Y\frac{c_{ymax} - c_{ymin}}{totalY - 1}$$

, where totalX and totalY are the width and height of the picture, respectively. There is minus one in the denominator because the first pixel represents zero, i.e. it starts from zero.

Generating the Mandelbrot set

The Mandelbrot set is generated by first looping over all X and Y of the picture. While looping over every X and Y there is a loop that will calculate the function Z_{n+1}. For every iteration of the last loop there will be a new value of Z_{n+1}. If the values get bigger and bigger for every iteration, the set is not bounded and it will not be a part of the Mandelbrot set. To check if it's not bounded there is an if-statement in the last loop that says, if the sum of Z_x^2 and Z_y^2 is greater than 4, the set is unbounded. There is also another if-statement that check if the same value of Z_{n+1} ever occur

again in the same set. If it does, the set will be bounded and therefore a part of the Mandelbrot set.

Color the Pixels

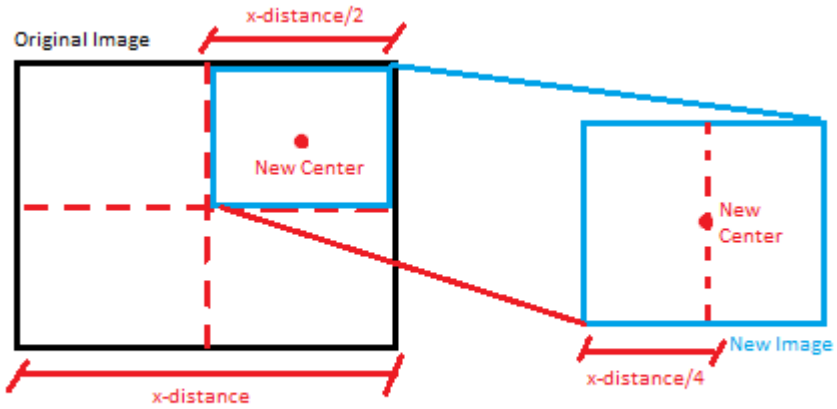
If the set is unbounded the pixel of position X and Y for just that set should be colored. The color is decided with a given function that will give a color in RGB. But the function, called putpixel, that will color the pixel only takes a color in hexadecimal form. Therefore, if the set is bounded and belongs to the Mandelbrot set, it is just to give the pixel the color 0x000000 which corresponds to black.

But in the case of the color, there have to be a converter that take the color from RGB format into hexadecimal format. Here there is a function named createRGB that takes 3 integers as argument (the RGB colors) and as an output it gives a positive (unsigned) hexadecimal number.

In binary numbers the start is from right instead of from left as usual. So here the color representing B will be 0xff, the first part. But then G takes place in front of B with use of a shift operator. Then we have 0xffff where the first "ff" represent G and the second represent B. To convert R works in the same way. Then when it is done, the function return the whole thing.

Zoom

There is a function called zoom in the program that will be zooming into a certain point if it's called. The zoom function will zoom by giving c_{xmax}, c_{xmin}, c_{ymax}, c_{ymin} new values. So if the program is supposed to zoom around a certain point, the point is (c_x,c_y). Which is called "New Center" in *Figure 1*.



$$c_{xmax} = \text{"New point in x-position"+"x-direction/4"}$$

Figure 1 This shows how the new c-value is calculated when zooming into the picture. This is an example for cxmax. It is equally for the other three values except that for cxmin and cymin the second term is subtracted and also there is the new point in y-direction for cymax and cymin.

User Interface

First the user can choose between default settings and to set the size of the image. That is fixed with if-statements that does something if it's true and something else

if it's false. Then the user can choose between to just see the Mandelbrot set and to see the Mandelbrot set with the zoom function. This is fixed with a switch-case. Then with another switch-case the user can close/quit the program by pressing ESC.

The Code in General

When the user has decided which size the image will be and to see the Mandelbrot set with the zoom function, an infinite while-loop starts. It only breaks if ESC is pressed or if the program shuts down in another way. As long as it is running it will keep calling a function named render. And for every new loop the render function will get new input arguments from the zoom function. That is a new c_{xmax}, c_{xmin}, c_{ymax}, c_{ymin}.

In render, the function that generates the Mandelbrot set and put a color to the pixels is called. Also a function that update the screen is called every time render is called.

In the function Mandelbrot there are three for-loops. One that loops over all X, one that loops over all Y and one that iterates and generate the function Z_{n+1}. It is also within these loops that c_x and c_y are generated by interpolation.

Optimization

To make the code faster, some simple changes has been made. That is, e.g. not to use unnecessary mathematic operators. Another thing that makes the program faster is that when it runs the iteration that generates Z_{n+1}, an if-statement checks if the sum of Z_x^2 and Z_y^2 is greater than 4, as mentioned before. If it is greater than 4, the set is unbounded and the pixel should get a color. Then it breaks, that means that it stops the iteration of Z_{n+1}. That saves time.

If this if-statement is false, the pixel will be colored black. No demands, just if the if-statement is false. That saves time.

Then there were a problem when the pictured was zoomed in a bit, because then it is mostly black in the image. That means that the program has to run more iteration to check if the if-statement is true or false. Therefore there is another if-statement (as mentioned before), also within all the loops, that check if an earlier value in the set of Z_{n+1} appears again, then it should break and paint the pixel black. That saves time because it breaks the iteration.

Time

The fastest iteration takes 0.3 seconds and the slowest iteration takes 4.6 seconds. The fastest one is when it is the least black in the image and the slowest is when it is the most black in the image. That is because when the if-statement in the code that checks for repetition of the Z_{n+1} value only compare with 5 old values. The more old values, the faster the code gets.

Other

If the user choose to set the size of the picture, there have to be a relation between the input of the size of the picture and the c_{xmax}, c_{xmin}, c_{ymax}, c_{ymin}. If there isn't, the picture will look like *Figure 2*. And if the relation is

$$c_{ymax} = c_{ymin} + (c_{xmax} - c_{xmin})(\frac{totalY}{totalX})$$

But then the picture will look like *Figure 3*. To make the Mandelbrot set picture centered, this must be fulfilled

$$c_{ymin} = -c_{ymax}$$

Then the picture will look like *Figure 4*.

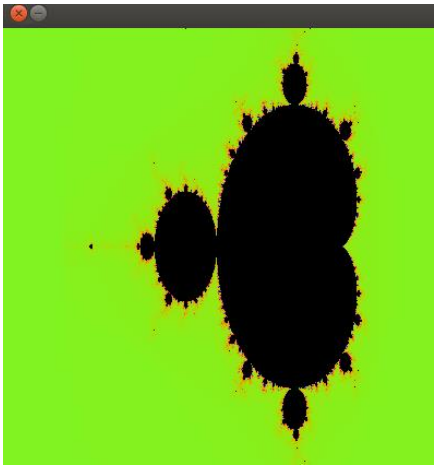


Figure 2 Without the relation between totalX, totalY and cxmax, cxmin, cymax, cymin.

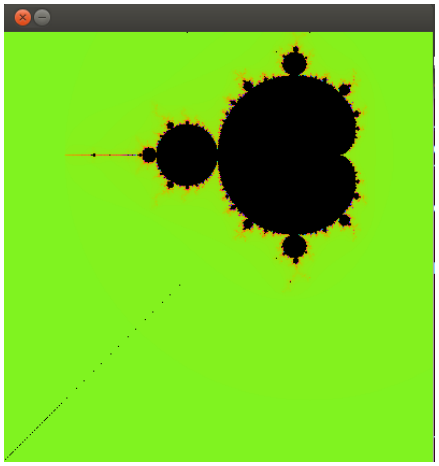


Figure 3 Without the relation between cymax and cymin.

Figure 4 The final picture with the relation.