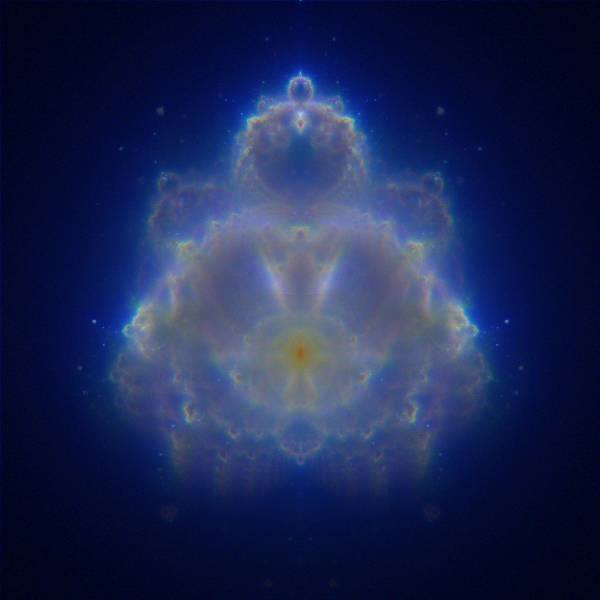
**Buddhabrot**



Royal St. George's College  
2012 ICS3U Final Exam  
Thursday May 31, 2012 • 12:30 pm • The Gym

## The Standard Mandelbrot Algorithm

For the better part of the last month you have developed code for the more popular algorithm for rendering the Mandelbrot Set. That is, points *c*, were selected in an orderly manner starting at the top-left corner (corresponding to *z* = -2.25+1.5*i*), across the domain to the top-right corner (corresponding to *z*= 0.75+1.5*i*), similarly down the range, eventually finishing at the bottom-right corner (corresponding to z = 0.75-1.5*i*). As each *c* was selected, *z* was reset to 0 + 0*i* and the feedback formula *z = z2+c* was iterated until one of two conditions emerged. Either the magnitude of *z* exceeded a value of 2.0 or the maximum number of iterations (the bailout) was reached. The visualization of the image depended solely on the **final** number of iterations for each *c*.

1. If the number of iterations equalled the bailout value, the pixel on the screen corresponding to the *c* value was coloured black, implying that the complex number *c* value was **in** the Mandelbrot Set, or,
2. If the number of iterations was less than the bailout value (implying the orbit *escaped* the Mandelbrot Set) the pixel on the screen was either coloured white or used as a index into a palette which assisted in revealing further clues as the structure of the field around the Set.

## The Buddhabrot Algorithm

A different way of rendering the Mandelbrot Set in attributable to Melinda Green who, in 1993, was not so quick to simply use the **final** number of iterations for each *c*, but rather, recorded information about **all** the *z* values in the orbits of each *c*. Her strategy differed in a number of other ways from the standard algorithm described above. Here it is.

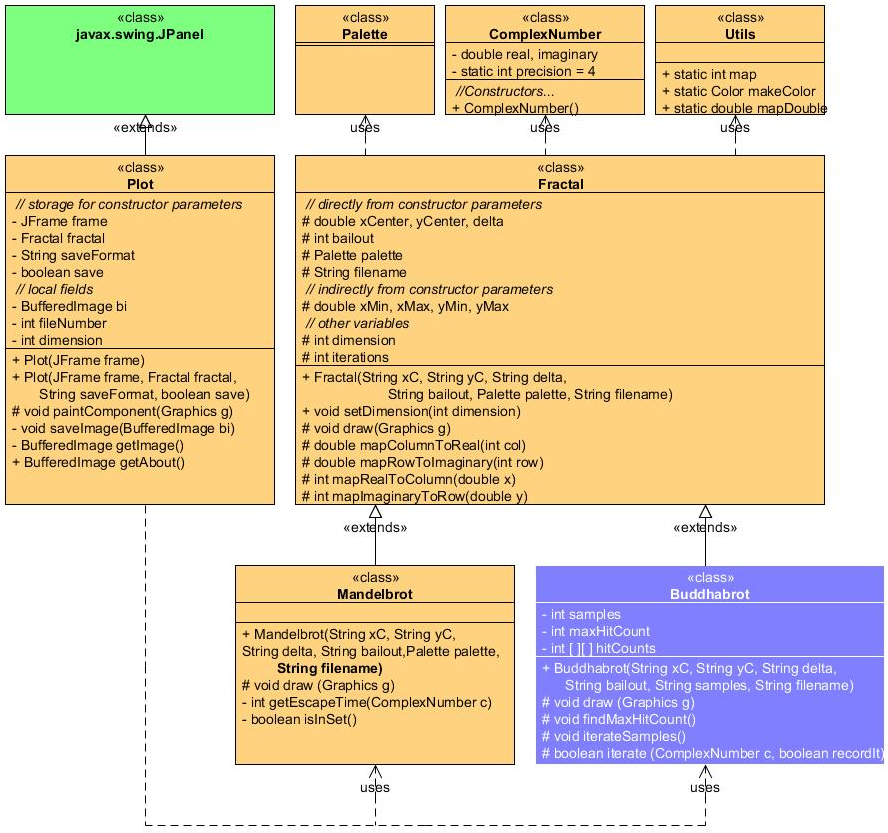
1. To start, a two-dimensional integer array corresponding to the dimensions of the panel is established. For example, since we've been using a 600×600 panel, a 600×600 integer array (matrix) is required. This matrix will be used to record the hit counts for each *z* . That is, the number of times the z-value in **any** orbit is encountered must be maintained. For each *z* in the orbit for a given *c*, the corresponding cell in the hit count array is incremented.
2. The domain and range are not traversed in an orderly top-down, left-right. Rather, random *c*  values are selected within the region to be tested.
3. Furthermore, for each *c,* the algorithm takes ***two*** passes through the feedback formula. Since hit counts are only recorded for those *c* values that manage to escape the Mandelbrot Set (*z* magnitude greater than 2.0), a first pass is needed to determine this (hit counts are not recorded for the first pass). If the orbit *did* escape (iterations less than the bailout) a second iteration of the feedback formula (with the same *c* and *z* starting again at *0+0i*) is undertaken. This time the hit counts **are** recorded.
4. After a few thousand *c* values (samples) are treated in the manner described above, the hit count array can then be used to provide a source for a Grayscale rendering strategy. First, the maximum hit count must be determined (*traverse the rows and columns of the hit count array to discover this value*). Let's say it was 37. Since all other counts are no greater than this maximum, each can be divided by the maximum to produce a value less than or equal to 1. This ratio can be cast to a float value and used as the red, green and blue for use in the Color constructor. This will produce a shade of gray somewhere from black to white.

## Examples

The table below includes Buddhabrot renderings on a 300×300 panel using my Desktop PC running at 3.2GHz with 8 GB RAM. The rendering times range from 1 second for Figure 1. up to 5 minutes for Figure 4. You would be wise to restrict yourself to the parameters of Figure 1 until your code is working.

|  |  |
| --- | --- |
| **Figure. 1. Bailout: 200; Samples: 50 000;** | **Figure 2. Bailout: 200; Samples: 500 000** |
| Buddhabrot3C2C50K.gif | Buddhabrot3C2C500K.gif |
| **Figure 3. Bailout: 2000; Samples 500 000;** | **Figure 4.Bailout: 2000; Samples: 50 000 000** |
| Buddhabrot3C2K500K.gif | Buddhabrot3C2K50M.gif |

## Task.

1. You are to implement the Buddhabrot class defined by the UML below and incorporate it into your Explorer Project. Review the documentation thoroughly at  
   <http://darcy.rsgc.on.ca/ACES/ICS3U/FractalFramework/Buddhabrot/doc/index.html>  
     
   
2. Create the script file Buddhabrot.exp with these contents to produce an image resembling Figure 1 on the previous page (*be sure to insert tabs between fields*). Adapt your Explorer driver to accommodate instantiation of this new fractal type.   
     
    Frame 300 The Buddhabrot gif true  
    # Fractal type xCenter yCenter delta bailout samples  
    Fractal Buddhabrot -0.4 0 1.5 200 50000
3. As for rendering you have already probably guessed that the row and column coordinates have been interchanged so that the Buddhabrot appears in oriented roughly as if the Mandelbrot Set was rotated 90° clockwise.
4. Feel free to explore the internet for alternative explanations of the algorithm if you feel mine is lacking. You will find a number of them, some even with source code provided if that helps. You will not be penalized for adapting code from the internet but I will caution you from doing so as what may seems like a straightforward conversion can quickly turn into a nightmare. Trust me on this point.
5. Document your code fully and submit all project files to handin by the end of the examination.

## Evaluation

Your Explorer application is a tremendous achievement resulting from hours and hours of skilful design and coding . The evaluation of this submission will be in direct correlation with the smooth integration of the Buddhabrot algorithm, both visually and mechanically. Specific areas include,

* **(5) Design**. Reflects sound, well-thought-out *stepwise refinement* strategies
* **(5) Documentation**. superior documentation styles and content .
* **(5) Efficiency**. No unnecessary statements.
* **(15) Accuracy**. Functions exactly as required by the examination specification.

**Note**. The cover image of this examination is of the *Nebulabrot*, a coloured extension of the *Buddhabrot* algorithm you may wish to explore in the future.

Work hard. It will serve your reputation well.