

The University of Hong Kong  
Department of Computer Science and Information Systems  
Programming Methodology and Object Oriented Programming  
for Software Engineering and Double Degree students (CSIS0396A)

### Assignment 3

**Assigned: Wednesday Apr 27, 2001**

**Deadline: Wednesday May 11, 2001, 5:00pm.**

This is a programming assignment. For assignments that are submitted at or before deadline, submit your assignment by using the handin command. For assignments that are late for at most 48 hours, send a mail to the tutor directly. In the latter case, the marks will be multiplied by 0.8. Assignments that are more than 48 hours late are not accepted.

You are allowed to work on assignment in group of at most 4. However, our advice is to do it independently, since it is not easy to split the work for this assignment.

#### 1. The Mandelbrot set

Hidden from its boring and mechanical aspects, there are facets of mathematics that are very beautiful. Fractal is one of these facets, and in this assignment we will try to look into the beauty of a particular kind of fractal, called the Mandelbrot set, with the help of Python and its Tk interface. We will improve the performance of the program by reimplementing specific part of the code with a C++ plugin.

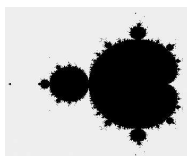
The Mandelbrot set is defined as follows:

Fix  $c$ . Let  $z$  to be 0. Now repeatedly replace  $z$  by  $z^2 + c$ . The complex number  $c$  is within the mandelbrot set if and only if the complex number  $z$  can never grow to the magnitude of 2.

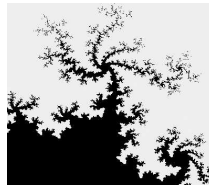
For example, consider  $c = 1$ . Initially  $z = 0$ , after one iteration  $z = 0^2 + 1 = 1$ , after two iterations  $z = 1^2 + 1 = 2$ , so 1 is not within the Mandelbrot set.

On the other hand, consider  $c = 0.2$ . Initially  $z = 0$ , after one iteration  $z = 0^2 + 0.2 = 0.2$ , after two iterations  $z = 0.2^2 + 0.2 = 0.24$ , after three iterations  $z = 0.24^2 + 0.2 = 0.2576$ , ... After a while you'll realize that it never increase over 0.28, or in fact 0.27639320225..., so 0.2 is within the Mandelbrot set.

If you plot the values of  $x, y$  of all the complex numbers  $c = x + yi$  in the mandelbrot set, you get something like this:



If you zoom on somewhere near the edge, you get something like this:



## 2. A Mandelbrot explorer (60%)

Using a computer, we cannot easily know whether a number is within the Mandelbrot set or not, since we cannot apply the rule  $z = z^2 + c$  infinitely. But we can approximate the mandelbrot set by stopping after some large enough number of iterations. The maximum number of iterations determines how good is the approximation, and for our assignment this number must be at least 20.

For the number  $c = 1$ ,  $z$  reaches the magnitude of 2 after two iterations. We say that the escape number of  $c$  is 2. By plotting a different color for a complex number of different escape number, we get a more colorful map of the Mandelbrot set.

For our assignment, write a Python script that plots these colors on a window of the X-window display. Use Tkinter of Python to draw the windows.

Your program should allow at least the following operations. Closing the windows should stop the program, resizing the window should cause the same area to be recalculated with a new resolution. There must be user interface to select the center of the map, and there must be user interface allowing users to zoom into a particular area of the map.

Hint: To show the map, you can use a Tk label containing a PhotoImage object as its image. Use the put method of the PhotoImage object to write pixels into the PhotoImage.

## 3. Improving performance (40%)

You will find that the performance of the Python script is intolerably slow, especially when the resolution is high. There are two problems here: Calculating the color for a point is a slow operation, since it might require up to 100 operations of complex number, and each of these operation create a new complex number object. Second, the script need to plot the points one by one through the Tkinter interface, which is not fast at all.

Implement a C++ plugin that contains a function to compute the color of all the required points, given the boundary of the window (and other parameters). The function should return a Python sequence of sequence, so that the Python script can call the put method of the PhotoImage object with the returned sequence. Update the Python script so that it uses the plugin.

## 4. Hand-in

Turn in a tar-gzipped CVS repository, with the main trunk containing your script and the source of your plugin. There should be a branch, called `pure-python`, that contains a version of the

program written purely in Python. (Note: it is okay to have your whole script in a single Python file and your whole plugin in a single C++ file.) Remember to put the Setup.in file required to compile your program into the repository. The repository should also contain a short README file which lists your group members and describes the capability of your program.

## 5. Hints

You can find a demo program in the course directory `~c0396a/demo/`. To run it, type `~c0396a/demo/mandelbrot`. You need to be using a Unix workstation to run the program. It is written using pure Python, so you can expect that your final program with plugin should be much more efficient than that.

10% of the first 60% will be allocated to the intuitiveness of the interface, and the demo interface would worth around 6%. So be creative.

The documentation of Tkinter is a bit thin. My recommendation is that you visit the following page and fetch the PDF version of the documentation there for reference.

<http://www.pythonware.com/library/an-introduction-to-tkinter.htm>

To write the Python plugin, read the Python documentation in the web page below.

<http://www.python.org/doc/1.5.2p2/ext/intro.html>

If you use Redhat Linux to work on your assignment, you will need to install an RPM for Python development. You can fetch it from the course web page.

To actually build the plugin you write, you need a Makefile. But the makefile can be created automatically from a file called "Setup.in". Read the following page to see how this can be done. Note that the "Makefile.pre.in" file needed resides on `"/usr/lib/python1.5/config/Makefile.pre.in"` and `"/usr/local/lib/python1.5/config/Makefile.pre.in"` respectively for Redhat and for virtue. Also, you should add a line like `"CCC=g++"` into the file "Setup.in" to tell that you want to use g++ as your C++ compiler.

<http://www.python.org/doc/1.5.2p2/ext/building-on-unix.html>

You will also need to create a tuple or list within your plugin code, and assign values to the elements of the tuple or list. Reading the documentation of the Python/C API documentation will enlight you about how to use a Python list within C++. Read it here:

<http://www.python.org/doc/1.5.2p2/api/sequenceObjects.html>

Be very careful to deal with reference counts when you return an object from your plugin. In general, newly created objects has reference count of 1, so you don't need to increment the reference count yourselves. On the other hand, if you get an object from a Python tuple or list and copy it to another list, then you need to increment the reference count.