# CSD2341 Computer Graphics Programming

## Workshop 6: 2D Viewing

**Assessment: 4 Marks. Due in the next workshop.**
**For on-campus students: demonstrate in the next workshop session.**
**For off-campus students only: zip your project directory and submit to blackboard.**

### Related Objectives from the Unit outline:

- understand the programming techniques and algorithms used for developing 2D graphics software.

### Learning Outcomes:

In this workshop, you will learn how to use set up viewing parameters for 2D graphics.

---

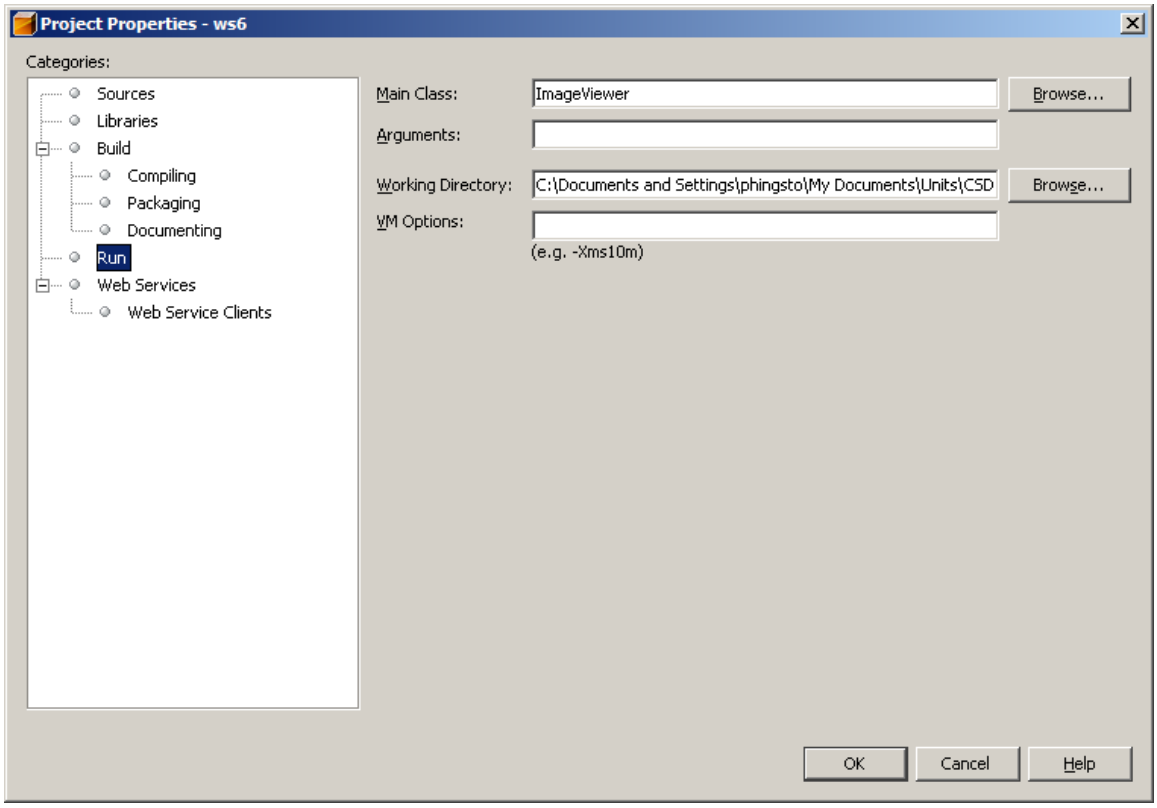
Two dimensional viewing functions can be used to implement zooming and panning functionality similar to that you would see on GoogleEarth (http://earth.google.com). Your task in this workshop is to complete the implementation of a simple application for viewing an image, which allows for zooming and panning, using the OpenGL functions for setting clipping/viewing windows and viewports. Here are your instructions:

1. Download the file `Viewing.zip` from BlackBoard, and save it somewhere.
2. Unzip it – you will get a folder called `Viewing`, with a subfolder called `src`, and an image file named `london-map-tourism.jpg`.
3. Set up a new NetBeans project with `Viewing` as the project folder, and `src` as the source folder. There will be a number of classes:

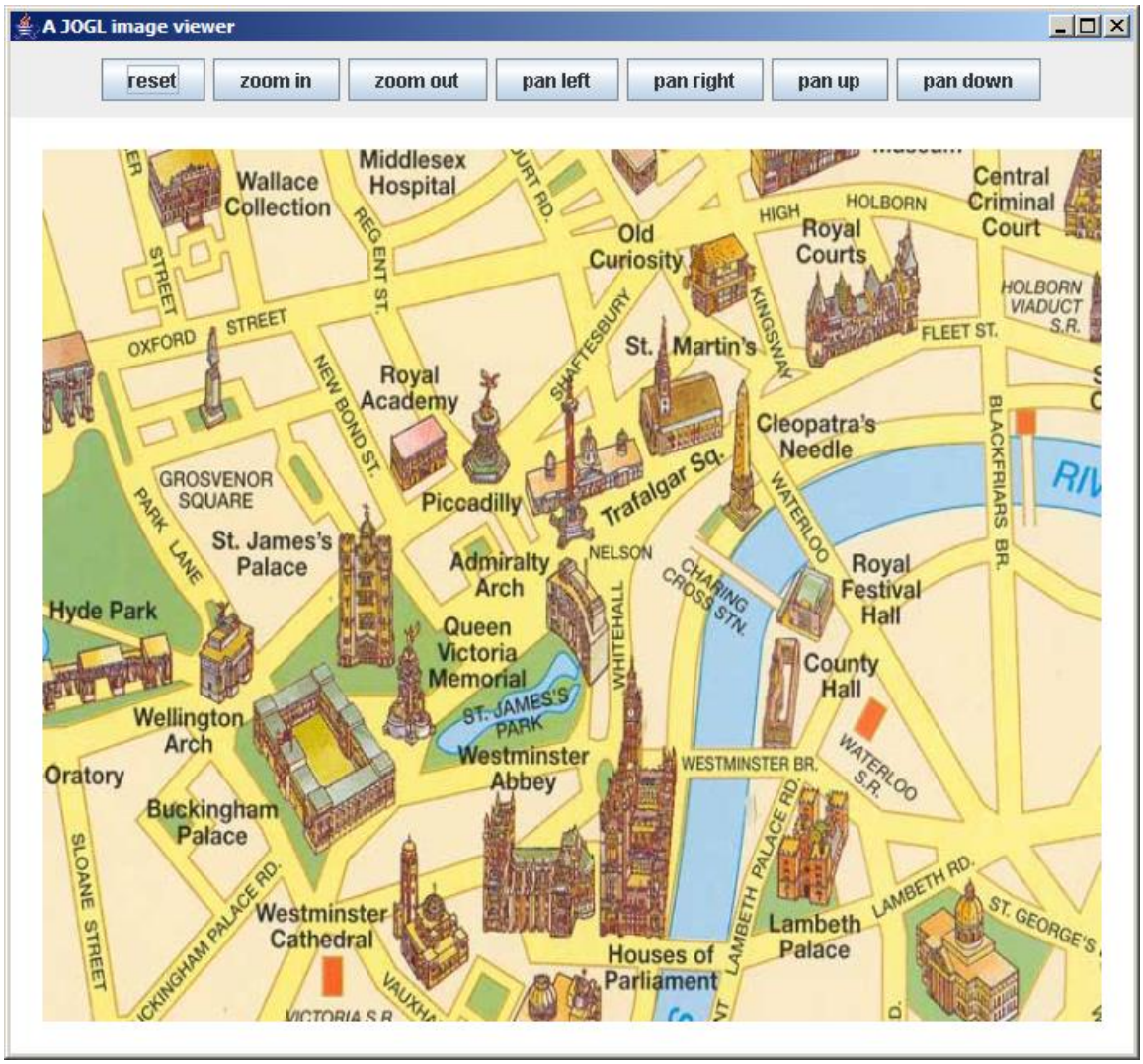
   `ImageViewer`: this is the driver.
   `ImagePanel`: this is the class where all the action happens.
   `JoglImageReader`: a utility class that reads the contents of an image file and changes them to a format that JOGL can use.
   `JoglImageData`: a data structure to hold this information until it is needed.
   `TextureLoader`: a utility class that converts pixel data into a form that can be used for a texture in JOGL.

   The only class that you will need to change for this workshop is `ImagePanel`. You might find `TextureLoader` useful if you want to try using textures in the programming assignment.

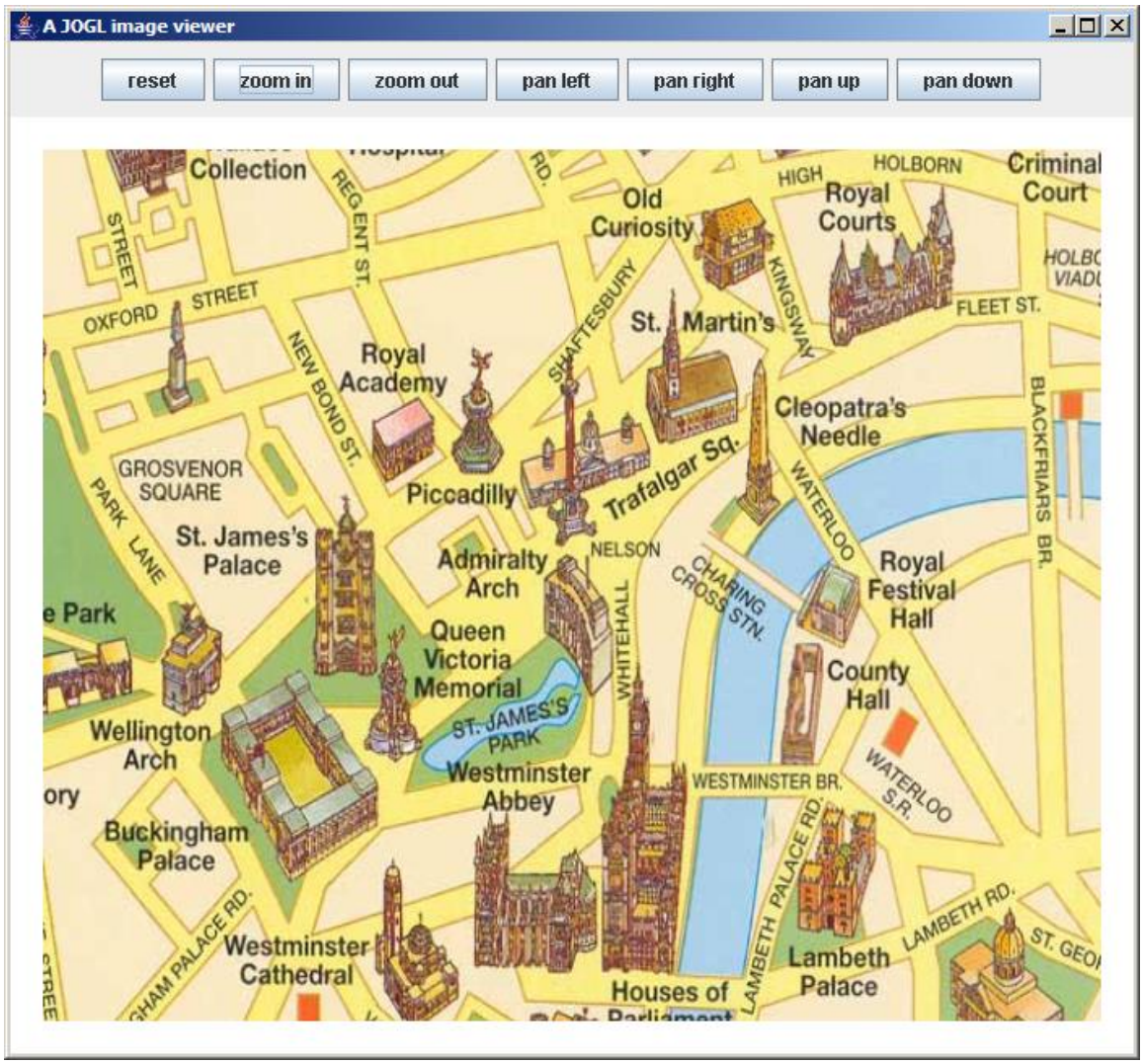
4. Now right-click on the project and go to “Properties”. Click on “Run” and then set “Main Class” to `ImageViewer` and the “Working Directory” to the project folder.




5. Run `ImageViewer` as an application. You should see a window like this:




Click on the “zoom in” button, and you should see this:




6. The image is slightly magnified. Clicking on “reset” will restore the original view of the image. The other buttons do not function yet – you have to supply the code behind them.

   Here is the code behind the “zoom in” button:

```
public void zoomIn(double factor)
{
      // This one is done for you as an example

      synchronized(this)
      {
         // find centre
         double centrex = (xwmin + xwmax)/2;
         double centrey = (ywmin + ywmax)/2;
```

```
            // calculate new width and height
            double width = (xwmax-xwmin)/factor;
            double height = (ywmax-ywmin)/factor;

            // we want to leave the centre unchanged
            // and change the width and height
            xwmin = centrex - width/2;
            xwmax = xwmin + width;
            ywmin = centrey - height/2;
            ywmax = ywmin + height;          }
    }
```

It works by setting the values of `xwmin`, `xwmax`, `ywmin`, `ywmax`, which give the coordinates of the bottom-left and top-right corners of the clipping/viewing window to be used. Reducing the size of this window means that less of the image will be shown in the viewport, so that what IS shown appears larger.

These variables are used in the `display()` method for the panel like so:

```
        // set clipping/viewing window
        gl.glMatrixMode(GL.GL_PROJECTION);
        gl.glPushMatrix();
        {
            gl.glLoadIdentity();
            synchronized(this)
            {
              glu.gluOrtho2D(xwmin, xwmax, ywmin, ywmax);
            }

            // switch to modelling mode
            gl.glMatrixMode(GL.GL_MODELVIEW);

                  … drawing code goes in here

         }
         gl.glMatrixMode(GL.GL_PROJECTION);
         gl.glPopMatrix();

         gl.glMatrixMode(GL.GL_MODELVIEW);
```

Notice the switching between `MatrixModes`, and the use of `PushMatrix` and `PopMatrix`.

## Exercises

7. Your task is to provide the code for the other buttons:
    a. zoomOut : opposite of zoomIn ! (1 Mark)
    b. panLeft: move the image to the right! (so that the “camera” moves to the left). The parameter **factor** determines what fraction of the currently visible portion of the image we should move. (0.5 Marks)
    c. panRight: opposite of panLeft. (0.5 Marks)
    d. panUp: move the camera up. (0.5 Marks)
    e. panDown: move the camera down. (0.5 Marks)
    f. Add a “rotate” button (you will need to make changes to `ImageViewer` as well as `ImagePanel`). (1 Mark)
8. Answer the Tutorial Questions over the page. (no marks)
9. Demonstrate the assessed parts of the workshop next week, or for off-campus students, Zip up your solution, and submit using BlackBoard, by the next workshop.

Solutions will be posted on BlackBoard for both the programming task and the tutorial questions.

**Tutorial questions on 2D Viewing**

1. Transform the line from (1.0, 3.0) to (15.0, 17.0) from world to screen coordinates where the window is bounded by the rectangle with corners at (0.0, 0.0) and (20.0, 20.0), and the viewport is bounded by the rectangle with corners at (2, 1) and (13, 25). Show all working and draw diagrams showing the window and viewport.
2. The transformation described above would cause the image to be distorted. Why, and how could this be prevented?

1. If you magnify the triangle with vertices A(3,2), B(5,6) and C(8,1) to twice its size while keeping B(5,6) fixed what are the new values for the vertices? Work this out using matrix calculations.