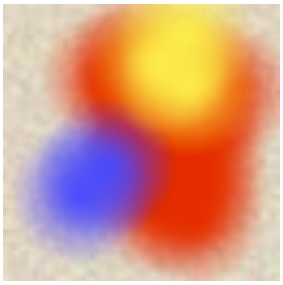
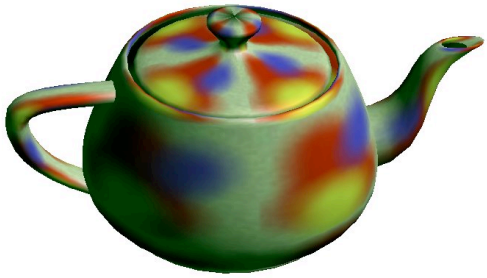
COMP27112 Coursework Assignment 3

This material is examinable (the ideas, not the details)

The purpose of this Assignment is to

- Get some experience with lighting, shading and texturing 3D models using OpenGL.


Task 1: Make a folder **`cwk3`** under your **`COMP27112`** folder and copy the materials for this Assignment into it:

```
shell$ cp /opt/info/courses/COMP27112/Coursework/cwk3/* .
```

You should now have the files **`teapot.c`**, **`bitmap.c`**, **`bitmap.h`**, **`splodge.bmp`** and **`coyote.bmp`**.

Task 2: Use **`cogl`** to compile **`teapot.c`** and run it. Use the mouse (left button) to rotate the teapot about **X** and **Y**. Does the image look right? If not, what do you think is wrong? Try and figure the problem out without looking at the source code.

Task 3: Now edit the source code to fix the problem, and recompile and run. (HINT: if you look in the function **`initialise()`**, you will see an OpenGL call commented out.) Make sure you understand what this call does, and make sure you understand what the z-buffer is for.

Task 4: Change the call to **`glShadeModel()`** to be **`glShadeModel(GL_FLAT)`** and observe the effect. Now alter the **`keyboard()`** callback function so that pressing the "s" key will toggle between smooth and flat shading (remember to call **`glutPostRedisplay()`** at the end of **`keyboard()`**).

Task 5: Experiment with changing the values in these sections of the code:

```
GLfloat white_light[]=     { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_position0[]= { 5.0, 5.0, 5.0, 0.0 };
```

```
GLfloat matSpecular[]=  { 1.0, 1.0, 1.0, 1.0 };
GLfloat matShininess[]= { 50.0 };
GLfloat matSurface[]=   { 0.0, 1.0, 0.0, 0.0 };
```

And observe the effects. Choose your favourite colours for the light and the material of the teapot.

Task 6: Add another light. OpenGL supports up to eight lights, but we would rarely use more than two.

Task 7: Now you'll texture the teapot. Let's do that first, and then discuss what is happening. Edit **`teapot.c`** and in the **`display()`** function, uncomment the whole block of 9 lines of code commented out starting:

```
/* code to set up texturing...
```

and recompile. You should see a splodgy teapot.

Task 8: A 2D texture in OpenGL is a 2D array of texel colours (a texel is a pixel in a texture image), and usually we'll read an external image file into the array. As you know, there are many different file formats and encodings for images, and OpenGL (wisely!) doesn't include any functions at all for decoding and reading image files – the programmer has to take care of all that themselves. Luckily however, people have written portable code to do this, and made it freely available. There are all sorts of image loaders, and here we're going to use a simple one that can load a BMP format image (the code is originally from the modestly-named "OpenGL SuperBible" (www.starstonesoftware.com/OpenGL/).

The code is in the file **`bitmap.c`**, which you can see **`teapot.c`** includes at the top, and you can look at that code if you like, but there's no need to (and it isn't much fun). All we need to know is that it provides a new type **`BITMAPINFO`** and a simple function **`LoadDIBitmap()`** which we call as follows:

```
BITMAPINFO *TexInfo;    // Texture bitmap information
GLubyte *TexBits;       // Texture bitmap pixel bits

TexBits= LoadDIBitmap("splodge.bmp", &TexInfo);
```

This opens and decodes the file **`splodge.bmp`**, and loads its pixels into **`TexBits`**. (**`TexInfo`** contains other information, such as the image size).

Now, try a different texture image: edit **`teapot.c`** and in the call to **`LoadDIBitmap()`**, use the file **`coyote.bmp`** instead.

What do you notice now? Because this image is clearly orientable (unlike the splodge), we can see that it's tiled several times on the teapot, and you can

probably see that it's been mapped separately onto each of several surface "patches" that together make up the teapot. (I added that blue line down one side of the image, so that you could see the edges of the patches clearly). Each of these surface patches is generated using the Bézier formulation (look this up if you don't know what it is).

Task 9: You'll also notice that the texture image has completely replaced the material colour (and the lighting) you specified earlier for the teapot. That's because of this function call:

```
glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_REPLACE);
```

which says **`GL_REPLACE`**. Change this to **`GL_MODULATE`**, and you'll see now that the texel colours are combined with material colours and the lighting. Google for other possible values of this argument, and try them out.

You'll see in the lecture on textures that we need to assign **texture coordinates** to our graphical primitives (using the **`glTexCoord2f()`** function), so that OpenGL can work out a mapping from the texels to the pixels it scan-converts (otherwise it couldn't possibly know how to map a texture image onto an arbitrary polygon). But if you look at the code in **`teapot.c`**, you won't see texture coordinates being specified for the teapot. So how does it work? As you've probably guessed, it's done for us inside **`glutSolidTeapot()`**. As the vertices on the Bézier patches are computed by OpenGL, texture coordinates are also computed, using:

```
glMap2d(GL_MAP2_TEXTURE_COORD_2, ...
```

Task 10: Your final task is to pull together some of the things you've experimented with in the Computer Graphics Coursework Assignments: using the OpenGL "camera", modeling with AC3D, object transformations, lighting and shading, texturing… and anything else that takes your fancy: to create an "interesting" scene of your choice. Your scene **must include at least one teapot**, but otherwise can be absolutely anything you like. Go for it!

Grab an image of your finished scene using **`xwd`**, like you did Assignment 1, and **submit** your image. Here's a reminder of the process:

```
shell$ xwd > cwk3.xwd
shell$ convert cwk3.xwd cwk3.gif
```

then **submit** **`cwk3.gif`**. You're finished.