CP2 Exercise 14 **12/02/2007**

1. Using last week's example program as a reference point:

a) Change the displayed object into a triangle. Each of the triangle's end points should have a different colour.

hint: *A RGB drawing colour in OpenGL is set using the function* ***glColor3f(red,green,blue);*** *Once selected a colour will be used for all vertices that are drawn after the selection..*

b) Create an outline (*made up from line segments*) for the displayed object. To be visible the line segments should have a different colour than the object itself.

c) Create a second object (*triangle or rectangle*) next to the original one. You might need to resize the original object to achieve this.

hint: *A separate object within its own* ***glBegin()-glEnd()*** *block may have to be used to achieve the outline effect. Line segments are drawn using the* ***GL_LINE_LOOP*** *constant as parameter for the* ***glBegin()*** *statement. You may also need to initialize and use a depth buffer to prevent rendering errors.*

2. Using the previous exercise as a reference**,**

a) Experiment with different projection matrices (*orthographic & perspective*).

b) Change the program code so that a second viewport is added (*the window size might need to be changed, and the object will have to be rendered once for each viewport*) and that the two viewports have separate projection matrices:
One of the 2 viewports should be set to a perspective projection while the other one should be set to an orthographic projection. Try out different values for the perspective projection to see how the image is affected.

hint: *For this you might want to use different Z-coordinates for the two displayed objects.*

3. Write a minimalist GLUT program that draws three partially overlapping triangles that use flat shading.

a) Blend the colours of the triangles. Experiment with different alpha values, blending factors and with the drawing order of the triangles.

hint: *Flat shading is selected by setting the flat shade model – for this call* ***glShadeModel(GL_FLAT);*** *You might want to try out what happens if the corner vertices are set to different colours and if the shade model is set to* ***GL_SMOOTH****.*

b) Display circles (*created using a* ***GL_TRIANGLE_FAN*** *with 10 edges each*) instead of triangles.

c) Then change the program to use polygons instead of triangle fans as the drawing primitive.

hint: *To calculate the X & Y co-ordinates for each corner of the circular polygon use the functions cos(2*corner/PI) & sin(2*corner/PI) and off-set them to their correct positions.*

4. Using the previous exercise as a reference**,**

a) draw the polygons using vertex and colour arrays.

b) change the program code so that only one vertex array is used (*using different colour arrays for each differently coloured circle and only one vertex array with the centre of the circular polygon at the origin*). The polygons should be moved to their different positions using the **`glTranslatef(x,y,z)`** function.

hint: *You may want to use the modelview matrix stacks.*