**Introduction to OpenGL**

The example code in this exercise, act as an initial structure of an OpenGL program under Windows API.

**Setup OpenGL to your project.**

1. OpenGL is included in Windows SDK. Therefore it does not require to setup the directory path.
   a. The OpenGL directory is sited under Windows SDK, inclusion of GL/ is require when including OpenGL header files:
   ```
   #include <gl/GL.h>
   ```

2. Go to your project properties -> Configuration Properties -> Linker -> Input.
   a. Add OpenGL32.lib to Additional Dependencies. This will directs your project to link to the OpenGL static library, which is required for the linker.
   b. Alternately, you can use the `#pragma comment` as in the Lab01 program.
   ```
   #pragma comment (lib, "OpenGL32.lib")
   ```

3. Reminder, please change Configurations setting to "All Configurations" before applying the changes, so that you won't do double works.

From now we will concentrate on the OpenGL commands in the `display()` function.

1. The method `display()` contains all the code for drawing the yellow triangle. In general, we call these commands as fixed function pipeline. As we are calling the OpenGL function to render our graphics.

2. **`glClear(GL_COLOR_BUFFER_BIT);`**
   This performs the screen clear, `glClear()` is a general clear buffer command and the parameter passed tells OpenGL what you want clearing. In this case `GL_COLOR_BUFFER_BIT` informs the system that the colour buffer should be cleared to the background colour, which by default is black.

3. **`glBegin(GL_TRIANGLES)`** … **`glEnd()`**
   This is the simplest way, and the most **inefficient** to render an object in OpenGL. An object is defined by specifying geometry and colour information within the block. The parameter in `glBegin()` specified what type of object you wish to draw.
   You may right click on it and select Go To Definition to view all the available parameters.

4. **`glVertex2f(x, y);`**
   In this example we're defining a 2D triangle, the `glVertex2f()` specified a vertex in 2D space (x and y only). The three calls to glVertex2f(x,y) indicate the three vertices of the triangle.
   By default, OpenGL has a drawing dimension of 2.0x2.0 with the origin in the centre. Take note that this is much difference to what you have learned in Direct3D9.

# Demo

1. Change the background colour to a specific colour other than the default.
    a. Call the `glClearColor()` before the call to `glClear()`.

    For instance, inserting `glClearColor(1.0f, 0.0f, 0.0f, 0.0f);` will set the background colour to red.

    * Remember that you can always right click on a function, select Go To Definition to identify what are the parameters you need to pass into the function.

2. Display a wireframe (outline / un-filled) of a triangle.
    a. Replace `GL_TRIANGLES` with `GL_LINE_LOOP` and run the program. This is a quick way to go from colour triangles to wireframe triangles.

    b. Call the `glLineWidth()` function to set the width of the line being drawn. Try altering the width parameter and see what happens.

3. Display a point at each angle of the triangle.
    a. Replace `GL_POINTS` in the parameter of `GL_BEGIN` and run the program.

    b. Call the `glPointSize()` function to set the size of the point being drawn. Try altering the size parameter and see what happens.

4. Apply different colour for different vertices.
    a. We can call `glColor3f` within the `GL_BEGIN…GL_END` block.

    b. Try specifying a colour prior to calling the `glVertex2f ()` function to set a colour for each vertex.

    c. Try the same modification for your line triangle and point triangle.