# Introduction to OpenGL

Tutorial 1: Create a window and draw a 2D square

**Introduction:**

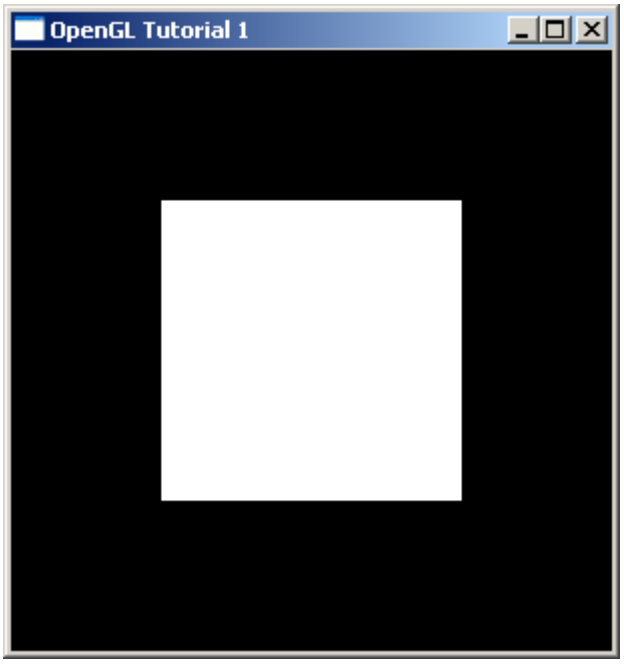
The aim of the first tutorial is to introduce you to the magic world of graphics based on the OpenGL and GLUT APIs. During this laboratory you will learn how to create simple 2D graphics in just a few lines of Microsoft Visual C coding. This tutorial will be used a template so that next programs will be based on. The program consists of only two functions: *draw()* and *main()*. The draw() function constructs a square object in two-dimensions while the main() function displays the generated object on a window. The following program opens a window and draws a white square in a black background. Create a new workspace (call it OpenGLTutorial1) and type the following source code and see what happens. Enjoy!

**Implementation:**

```
// First Tutorial
// Edited by Fotis Liarokapis
// All rights reserved

#include <GL/glut.h> // import GLUT library

void draw(void)
{
	// Clears the color buffer bit
	glClear( GL_COLOR_BUFFER_BIT);

	// Draws a cube by specifying four 2D vertices
	glBegin(GL_POLYGON);
		glVertex2f(-0.5, -0.5);
		glVertex2f(-0.5,  0.5);
		glVertex2f( 0.5,  0.5);
		glVertex2f( 0.5, -0.5);
	glEnd();

	// Forces execution of OpenGL functions in finite time
	glFlush();
}

// Main function
int main(int argc, char **argv)
```

```
{
	// Initialise GLUT library
	glutInit(&argc, argv);

	// Creates a window on the screen
	glutCreateWindow ("OpenGL Tutorial 1");

	// The display function is called each time there is a display callback
	glutDisplayFunc(draw);

	// Causes the program to enter an event-processing loop
	glutMainLoop();

	return 0;
}
```

**Expected Output:**




**Functions Used:**

- void glClear(GLbitfield mask) – The glClear function clears buffers to preset values including: color buffer → GL_COLOR_BUFFER_BIT, depth buffer → GL_DEPTH_BUFFER_BIT, accumulation buffer → GL_ACCUM_BUFFER_BIT, stencil buffer → GL_STENCIL_BUFFER_BIT.
- void glBegin(GLenum mode) – This function specifies the beginning of an object of various type modes including GL_POINTS, GL_LINES, and GL_POLYGON.
- void glVertex2f(GLfloat x, GLfloat y) – Specifies a two-dimensional vector.

- void glEnd() – This function specifies the end of a list and it is always used in conjunction with glBegin().
- void glFlush(void) – Function that forces previously issued OpenGL commands to begin execution.
- void glutInit(int argc, char **argv) – This command takes the arguments from main() in order to initialise GLUT. Note that glutInit() should be called before any OpenGL functions.
- void glutCreateWindow (*char title) – This command creates a window in the screen that contains a title given by the argument.
- void glutDisplayFunc(void (*func)) – This function is called each time there is a display callback.
- void glutMainLoop() – This command causes the program to enter an event-processing loop and must be always used at the end of the main() function.

## Tutorial 2: Initialisations in OpenGL

### Introduction:

The second tutorial will teach you how to initialise state variables and GLUT in order to create a more complex graphics application. This tutorial will consist of three functions (*draw()* and *main()* as before) and an initialisation function (called *initialisation()*). The source code for this is provided below. Create a new workspace (call it OpenGLTutorial2) and type the following source code.

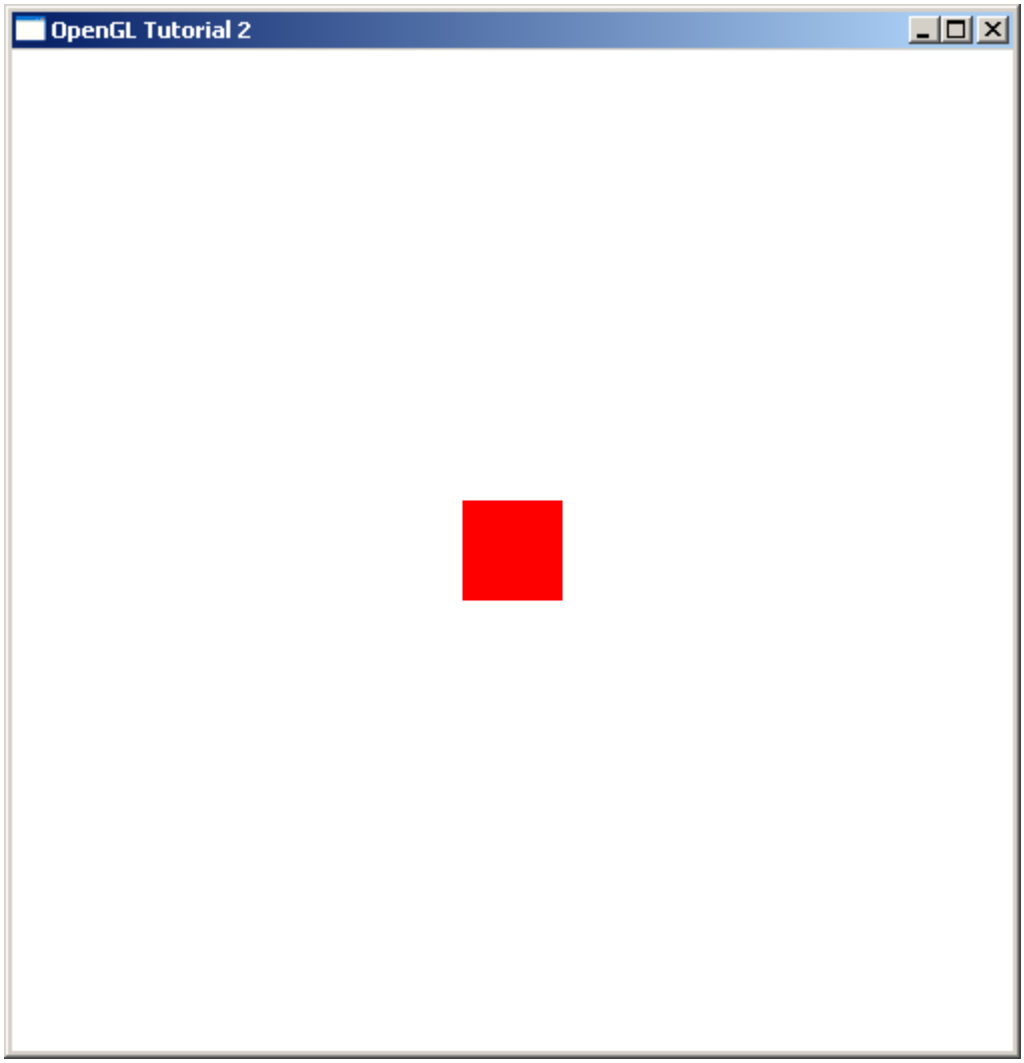
### Implementation:

```
// Second OpenGL Tutorial
// Edited by Fotis Liarokapis
// All rights reserved

#include <GL/glut.h> // import GLUT library


// Draw function
void draw(void)
{
        // Clears the color buffer bit
        glClear( GL_COLOR_BUFFER_BIT);
```

```
	// Set the color to red
	glColor3f(1.0, 0.0, 0.0);

	// Draws a cube by specifying four 3D vertices
	glBegin(GL_POLYGON);
		glVertex3f(-0.5, -0.5, 0.0);
		glVertex3f(-0.5,  0.5, 0.0);
		glVertex3f( 0.5,  0.5, 0.0);
		glVertex3f( 0.5, -0.5, 0.0);
	glEnd();

	// Forces execution of OpenGL functions in finite time
	glFlush();
}


// Initialise function
void initialise()
{
	// Clears the color pixels to white
	glClearColor(1.0, 1.0, 1.0, 0.0);

	// Applies matrix operations to the projection matrix stack
	glMatrixMode(GL_PROJECTION);

	// Replaces the current matrix with the identity matrix
	glLoadIdentity();

	// Multiplies the current matrix by an orthographic matrix
	glOrtho(-5.0, 5.0, -5.0, 5.0, -1.0, 1.0);
}

// Main function
int main(int argc, char **argv)
{
	// Initialise GLUT library
	glutInit(&argc, argv);

	// Initialise the display mode
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

	// Initialise the window position
	glutInitWindowPosition(300, 300);

	// Initialise the window size
	glutInitWindowSize(500, 500);
```

```
        // Creates a window on the screen
        glutCreateWindow ("OpenGL Tutorial 2");

        // The display function is called each time there is a display callback
        glutDisplayFunc(draw);

        // Call the initialise function
        initialise();

        // Causes the program to enter an event-processing loop
        glutMainLoop();

        return 0;
}
```

**Expected Output:**



**Functions Used:**

- void glColor3f(GLfloat red, GLfloat green, GLfloat blue) – This function define the color used by specifying the red, green and blue (RGB) components.
- void glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha) – This function specifies the clear values for the color buffers.
- void glMatrixMode(GLenum mode) – The glMatrixMode function specifies which matrix is the current matrix (GL_MODELVIEW → Applies subsequent matrix operations to the modelview matrix stack, GL_PROJECTION → Applies subsequent matrix operations to the projection matrix stack, GL_TEXTURE → Applies subsequent matrix operations to the texture matrix stack).
- void glLoadIdentity(void) – The glLoadIdentity function replaces the current matrix with the identity matrix.
- void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar) – The glOrtho function multiplies the current matrix by an orthographic matrix.
- void glutInitDisplayMode(unsigned int mode) – Function that requests a display with the properties in mode. The values of mode can be combined using options like the color model (GLUT_RGB, GLUT_INDEX) and buffering of color buffers like (GLUT_SINGLE, GLUT_DOUBLE).
- void glutInitWindowPosition(int x, int y) – Function that specifies the top-left corner of the window (in pixels) starting from the top-left corner of the screen.
- void glutInitWindowSize(int width, int height) – Function that specifies the initial height and width (in pixels) of the window on the screen.

Tutorial 3: Rotation, simple menu and reading from keyboard

**Introduction:**

In the third tutorial you will learn three things: how to rotate a graphical object on a window, how to create a menu in the command window and how to read input values from the keyboard. The rotation operations are provided by OpenGL API. The menu in the command window can be done in standard C while the reading from the keyboard operation is provided by the GLUT API. For these operations three more functions have been added: *spinScene(),Keyboard()* and *menu().* The spinScene() function performs simple mathematical operations to calculate the rotation angle. The Keyboard() function defines the keyboard keys that will be used and the menu() function prints a help menu

on the command window. To see how this works in practice create a new workspace (call it OpenGLTutorial3) and type in the following source code.

**Implementation:**

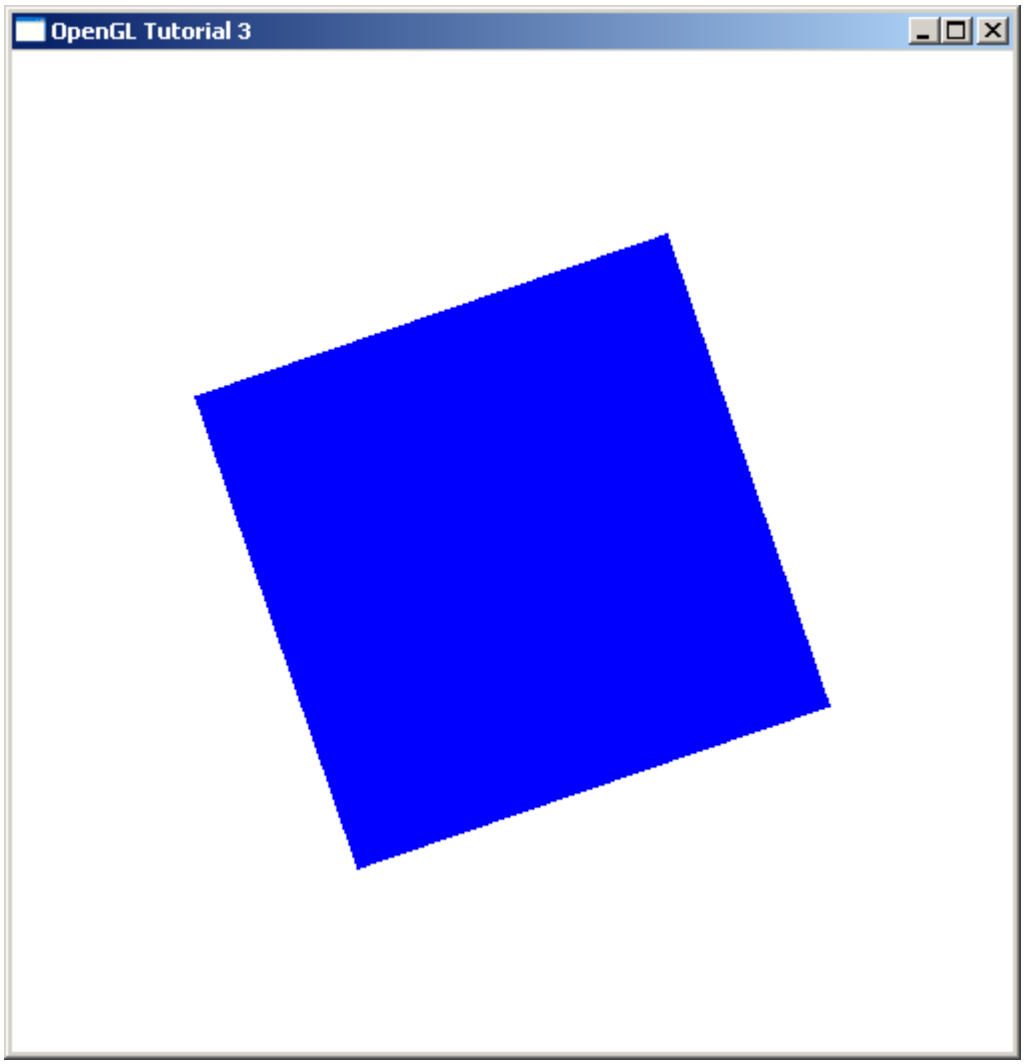
```
// Third OpenGL Tutorial
// Edited by Fotis Liarokapis
// All rights reserved


#include <stdio.h>              // import standard C library
#include <stdlib.h>             // import standard C library
#include <GL/glut.h>            // import GLUT library


static GLfloat spin = 0.0;              // global variable to determine the spin


// Spin the scene
void spinScene(void)
{
        // increment spin by a factor
        spin += 1;

        // Spin for ever
        if (spin > 360.0)
                spin = spin - 360.0;

        // Marks the normal plane of current window as needing to be redisplayed
        glutPostRedisplay();
}


// Draw function
void draw(void)
{
        // Clears the color buffer bit
        glClear( GL_COLOR_BUFFER_BIT);

        // Push the current matrix stack
        glPushMatrix();

        // Set the color to blue
        glColor3f(0.0, 0.0, 1.0);

        // Multiply the current matrix by a rotation matrix
```

```
        glRotatef(spin, 0.0, 0.0, 1.0);

        // Draws a cube by specifying four 3D vertices
        glBegin(GL_POLYGON);
                glVertex3f(-0.5, -0.5, 0.0);
                glVertex3f(-0.5,  0.5, 0.0);
                glVertex3f( 0.5,  0.5, 0.0);
                glVertex3f( 0.5, -0.5, 0.0);
        glEnd();

        // Pops the current matrix stack
        glPopMatrix();

        // Swaps the buffers
        glutSwapBuffers();
}


// Initialise function
void initialise()
{
        // Clears the color pixels to white
        glClearColor(1.0, 1.0, 1.0, 0.0);
}


// Reshapes the window
void reshape(int w, int h)
{
        // Sets the viewport
        glViewport (0, 0, (GLsizei) w, (GLsizei) h);

        // Applies matrix operations to the projection matrix stack
        glMatrixMode(GL_PROJECTION);

        // Replaces the current matrix with the identity matrix
        glLoadIdentity();

        // Multiplies the current matrix by an orthographic matrix
        glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);

        // Applies matrix operations to the modelview matrix stack
        glMatrixMode(GL_MODELVIEW);

        // Replaces the current matrix with the identity matrix
        glLoadIdentity();
```

```
}


// Function that reads input from the keyboard
void Keyboard(unsigned char key, int A, int B)
{
	switch(key)
	{
		// Start rotation
		case '1':
			// Sets the global idle callback
			glutIdleFunc(spinScene);
		break;
		// Stop rotation
		case '2':
			// Sets the global idle callback
			glutIdleFunc(NULL);
		break;
		// Escapes from the program by pressing 'Esc'
		case 27:
			// Terminates the program
			exit(0);
			break;
		default:
			break;
	}
}

// Function that prints a menu on the command window
void menu(void)
{
	printf("------------ OPTIONS -------------\n");
	printf("Press 1 to start rotation\n");
	printf("Press 2 to start rotation\n");
	printf("Press Esc to terminate the program\n");
	printf("---------------------------------\n");
}


// Main function
int main(int argc, char **argv)
{
	menu();

	// Initialise GLUT library
	glutInit(&argc, argv);
```

```
	// Initialise the display mode
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

	// Initialise the window position
	glutInitWindowPosition(300, 300);

	// Initialise the window size
	glutInitWindowSize(500, 500);

	// Creates a window on the screen
	glutCreateWindow ("OpenGL Tutorial 3");

	// Call the initialise function
	initialise();

	// The display function is called each time there is a display callback
	glutDisplayFunc(draw);

	// The reshape function reshapes the scene
	glutReshapeFunc(reshape);

	// Function that reads input from the keyboard
	glutKeyboardFunc(Keyboard);

	// Causes the program to enter an event-processing loop
	glutMainLoop();

	return 0;
}
```

**Expected Output:**




**Functions Used:**

- void glutPostRedisplay() – Requests the display callback to be executed.
- void glPushMatrix(void) – This function push the current matrix stack.
- void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z) – This function is used for rotation operations. The parameters are the angle which defines the angle of rotation, in degrees and x, y, z which define the coordinates of a vector.
- void glPopMatrix(void) – This function pop the current matrix stack.
- void glutSwapBuffers() – Function that swaps the front and back buffers.
- void glViewport(GLint x, GLint y, GLsizei width, GLsizei height) – The glViewport function sets the viewport. The x, y parameters specify the lower-left corner of the viewport rectangle, in pixels. The default is (0,0). The width, height parameters

spesify the width and height of the viewport. When an OpenGL context is first attached to a window, width and height are set to the dimensions of that window.

- void glutIdleFunc(void (*f) (void)) – The function f() is executed whenever no other events are to be handled. Used when we want to perform animations.
- void glutKeyboardFunc(void *f(unsigned char key, int x, int y)) – Function that identifies the function f() that is called when any key is pressed on the keyboard.

Tutorial 4: Lighting, shading and widget menu

**Introduction:**

Using the following example you will learn how to set a light source and then light a graphical object. Also you will learn hot to perform basic shading to 3D objects. Finally, using GLUT API you will be able to create cool menus in just a few minutes. To see how this works in practice create a new workspace (call it OpenGLTutorial4) and type in the following source code. Enjoy!

**Implementation:**

```
// Fourth OpenGL Tutorial 4
// Edited by Fotis Liarokapis
// All rights reserved


#include <GL/glut.h>            // import GLUT library
#include <stdio.h>                      // import standard C library
#include <stdlib.h>                     // import standard C library

static int spin = 0.0;          // global variable to determine the spin


// Spin the scene
void spinScene(void)
{
        // increment spin by a factor
        spin += 1;

        // Spin for ever
        if (spin > 360.0)
                spin = spin - 360.0;

        // Marks the normal plane of current window as needing to be redisplayed
        glutPostRedisplay();
```

```
}


// Draw function
void draw(void)
{
        // Define the position of the light source
        GLfloat position[] = { 0.0, 0.0, 1.5, 1.0 };

        // Clears the color buffer bit and depth buffer bit
        glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

        // Push the current matrix stack
        glPushMatrix();

                // Defines a viewing transformation
                gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

                // Push the current matrix stack
                glPushMatrix();

                        // Multiply the current matrix by a rotation matrix
                        glRotated((GLdouble) spin, 1.0, 0.0, 0.0);

                        // Set the light source parameters
                        glLightfv(GL_LIGHT0, GL_POSITION, position);

                        // Multiply the current matrix by a translation matrix
                        glTranslated (0.0, 0.0, 1.5);

                        // Disable lighting conditions
                        glDisable(GL_LIGHTING);

                        // Set the color to yellow
                        glColor3f(1.0, 1.0, 0.0);

                        // Draw a sphere
                        glutSolidSphere(0.1, 20, 10);

                        // Enable lighting
                        glEnable(GL_LIGHTING);

                // Pops the current matrix stack
                glPopMatrix();

                // Draw a torus on the screen
```

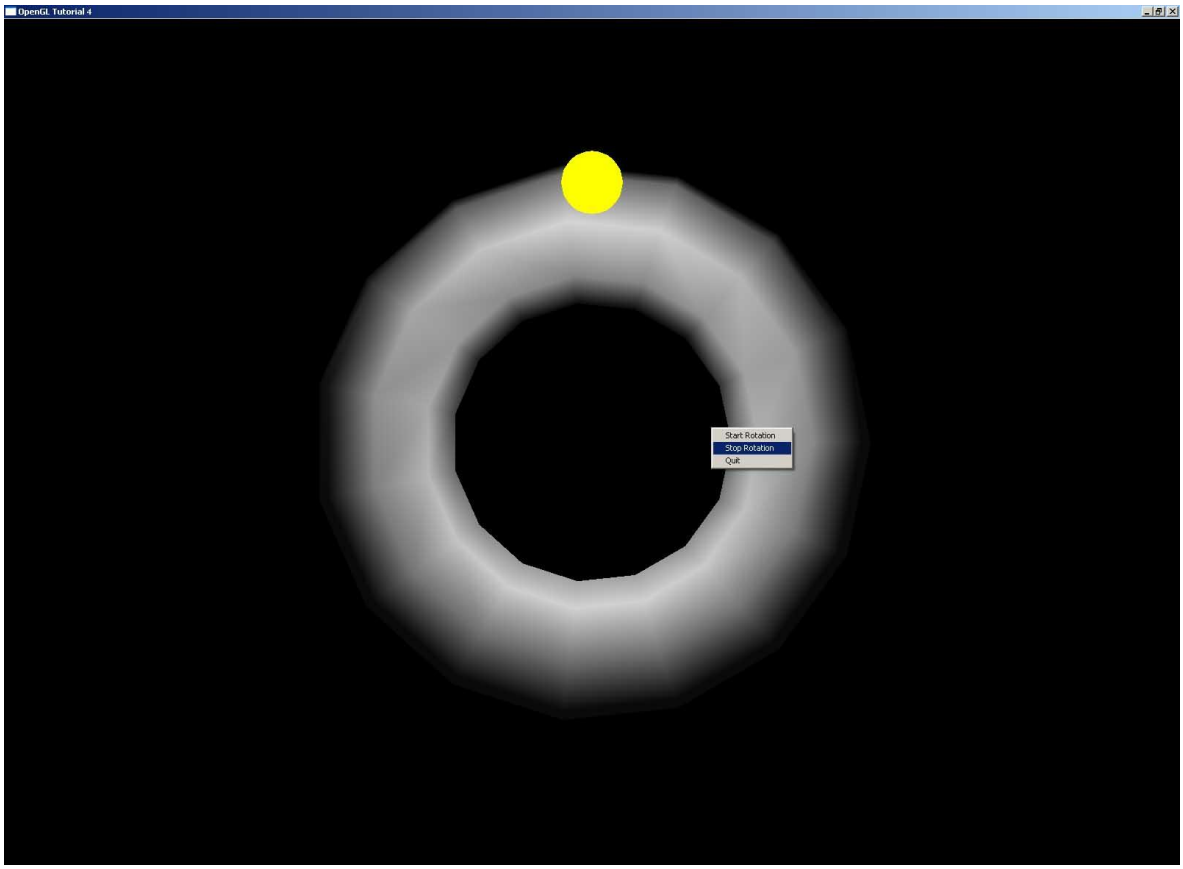
```
                glutSolidTorus(0.3, 0.9, 8, 15);

                // Pops the current matrix stack
                glPopMatrix();

        // Swaps the buffers
        glutSwapBuffers();
}


// Initialise function
void initialise()
{
        // Clears the color pixels to white
        glClearColor(0.0, 0.0, 0.0, 0.0);

        // Selects smooth shading
        glShadeModel(GL_SMOOTH);

        // Enable lighting operations
        glEnable(GL_LIGHTING);

        // Enable an OpenGL light
        glEnable(GL_LIGHT0);

        // Enable depth test
        glEnable(GL_DEPTH_TEST);
}


// Reshapes the window
void reshape(int w, int h)
{
        // Sets the viewport
        glViewport (0, 0, (GLsizei) w, (GLsizei) h);

        // Applies matrix operations to the projection matrix stack
        glMatrixMode(GL_PROJECTION);

        // Replaces the current matrix with the identity matrix
        glLoadIdentity();

        // Sets up a perspective projection matrix
        gluPerspective(40.0, (GLfloat) w/(GLfloat) h, 1.0, 20.0);

        // Applies matrix operations to the modelview matrix stack
```

```
	glMatrixMode(GL_MODELVIEW);

	// Replaces the current matrix with the identity matrix
	glLoadIdentity();
}


// Function that reads input from the keyboard
void Keyboard(unsigned char key, int A, int B)
{
	switch(key)
	{
		// Start rotation
		case '1': glutIdleFunc(spinScene); // Sets the global idle callback
			break;
		// Stop rotation
		case '2': glutIdleFunc(NULL); // Sets the global idle callback
			break;
		// Escapes from the program by pressing 'Esc'
		case 27: exit(0); // Terminates the program
			break;
		default:
			break;
	}
}

// Function that creates a menu used from a mouse
void mouseMenu(int value)
{
	if(value == 1)
		glutIdleFunc(spinScene);
	if(value == 2)
		glutIdleFunc(NULL);
	if(value == 3)
		exit(0);
}

// Function that prints a menu on the command window
void menu(void)
{
	printf("------------ OPTIONS -------------\n");
	printf("Press 1 to start rotation\n");
	printf("Press 2 to start rotation\n");
	printf("Press Esc to terminate the program\n");
	printf("----------------------------------\n");
}
```

```
// Main function
int main(int argc, char **argv)
{
	menu();

	// Initialise GLUT library
	glutInit(&argc, argv);

	// Initialise the display mode
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
	//glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);

	// Initialise the window position
	glutInitWindowPosition(300, 300);

	// Initialise the window size
	glutInitWindowSize(500, 500);

	// Creates a window on the screen
	glutCreateWindow ("OpenGL Tutorial 4");

	// Create a widget menu
	glutCreateMenu(mouseMenu);

	// Add a menu entry
	glutAddMenuEntry("Start Rotation", 1);

	// Add a menu entry
	glutAddMenuEntry("Stop Rotation", 2);

	// Add a menu entry
	glutAddMenuEntry("Quit", 3);

	// Attach menu to mouse right button
	glutAttachMenu(GLUT_RIGHT_BUTTON);

	// Call the initialise function
	initialise();

	// The display function is called each time there is a display callback
	glutDisplayFunc(draw);

	// The reshape function reshapes the scene
	glutReshapeFunc(reshape);
```

```
        // Function that reads input from the keyboard
        glutKeyboardFunc(Keyboard);

        // Causes the program to enter an event-processing loop
        glutMainLoop();

        return 0;
}
```

**Expected Output:**




**Functions Used:**

- void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz) – The gluLookAt function defines a viewing transformation. The parameters eyex, eyey, eyez define the position of the eye point. The centerx, centery, centerz define the position of the reference point. The upx, upy, upz define the direction of the up vector.
- void glLightfv(GLenum light, GLenum pname, const GLfloat *params) – Specifies the light-source parameters. The parameters include the light which is an OpenGL defined light. The pname which is a light source parameter for light. The following

values are accepted: GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_POSITION, GL_SPOT_DIRECTION, GL_SPOT_EXPONENT, GL_SPOT_CUTOFF, GL_CONSTANT_ATTENUATION, GL_LINEAR_ATTENUATION, GL_QUADRATIC_ATTENUATION. And finally the params which is a pointer to the value or values to which parameter pname of light source light will be set.

- void glTranslated(GLdouble x, GLdouble y, GLdouble z) – This function multiplies the current matrix by a translation matrix.
- void glEnable(GLenum cap) – Enables OpenGL capabilities. The parameter cap refers to a symbolic constant indicating an OpenGL capability (i.e. GL_LIGHTING, GL_DEPTH_TEST).
- void glDisable(GLenum cap) – Disables OpenGL capabilities.
- void glutSolidSphere(GLdouble radious, Glint slices, Glint stacks) – Creates polygonal approximations of a sphere, centred at the origin with the specified radius, through stacks lines of latitude and slices lines of longitude.
- void glutSolidTorus(GLdouble inner, GLdouble outer, Glint sides, Glint slices) – Function that defines a torus aligned with the z-axis by its inner and outer radius, sides divisions for radial sections, and slices around the torus.
- void glShadeModel(GLenum mode) – The glShadeModel function selects flat or smooth shading. The parameter mode is a symbolic value representing a shading technique. Accepted values are GL_FLAT and GL_SMOOTH. The default is GL_SMOOTH.
- void gluPerspective(GLdouble fovy,GLdouble aspect, GLdouble zNear, GLdouble zFar) – The gluPerspective function sets up a perspective projection matrix. The parameters include the fovy (the field of view angle, in degrees, in the y-direction), the aspect (the aspect ratio that determines the field of view in the x-direction), the zNear (the distance from the viewer to the near clipping plane) and the zFar (the distance from the viewer to the far clipping plane).
- void glutCreateMenu(void (*f) (int value)) – Command that creates a top-level menu that uses the callback f(), which is passed an integer value for the menu entry. A unique identifier is returned for the menu created.
- void glutAddMenuEntry(char *name, int value) – Function that adds an entry with name displayed to the current menu. The value is returned to the menu callback.
- void glutAttachMenu(int button) – This command attaches the current menu to a specified mouse button. This could be either GLUT_RIGHT_BUTTON, GLUT_MIDDLE_BUTTON, or GLUT_LEFT_BUTTON.

Task: Add another light and add more user interaction

Add another light in the scene and set the color to yellow again. Place the new light in a different position and perform another rotation using different keys. Finally add the interactions to the widget menu.