OpenGL Programming

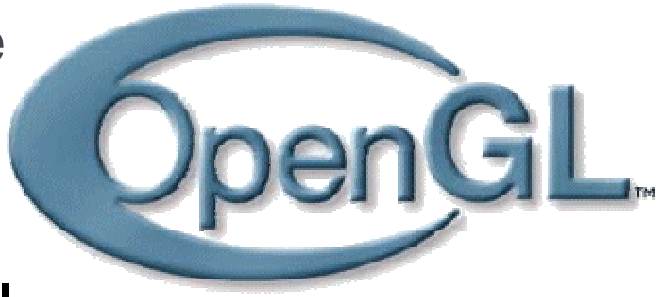
Human-Centered CAD Lab.

Contents

- What is OpenGL? (Review)
- OpenGL Programming with WinAPI
  - Drawing polygon
  - Coloring
  - Rotation
  - 3D object
- OpenGL Programming with GLUT
  - Simple Atom example
  - Push/Pop Matrix
  - Parallel vs. Perspective Projection

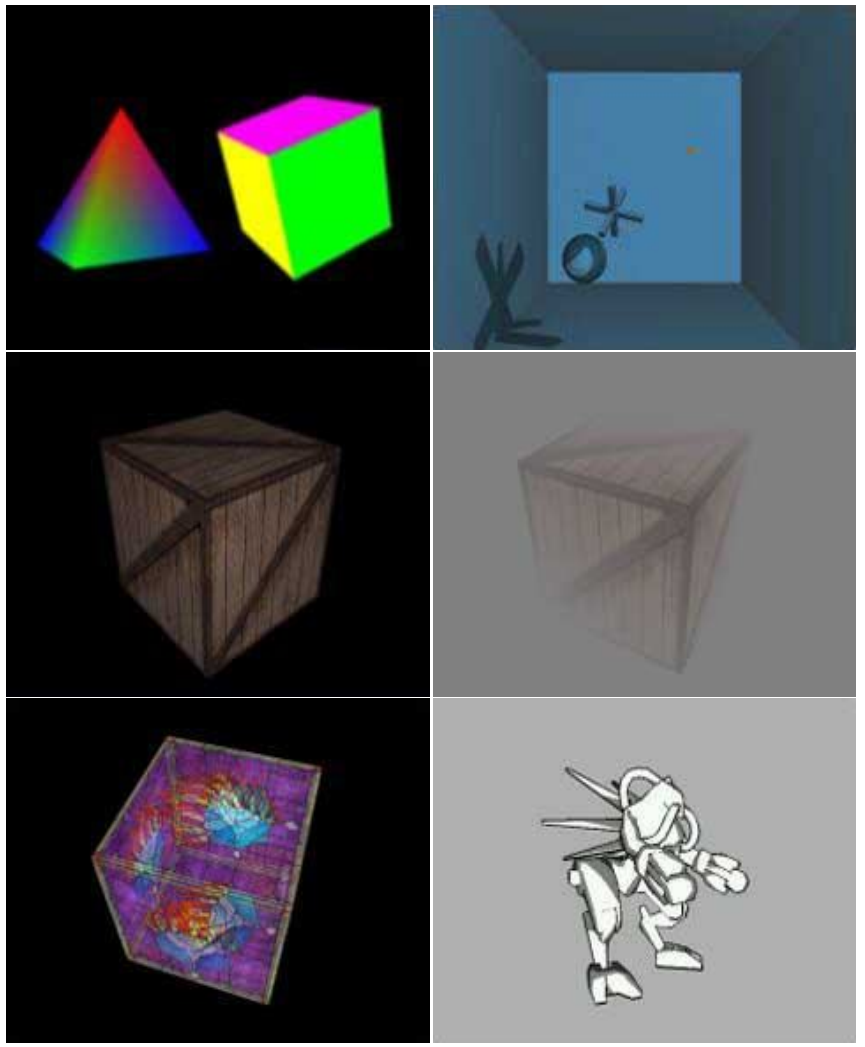
What is OpenGL?

- OpenGL is an acronym for Open Graphics Library
  - OpenGL Architecture Review Board (ARB)
  - Version 2.1 released on August 2, 2006
- Most Widely Adopted Graphics Standard
  - Easy-to-use / Well-documented
  - High Visual Quality and Performance
  - Portable/Reliable/Stable/Scalable
  - Low-level graphics commands
- Platform-independent graphics API.
  - UNIX, Linux, Windows9X/NT/2000/XP, OS/2, MacOS, BeOS, etc.



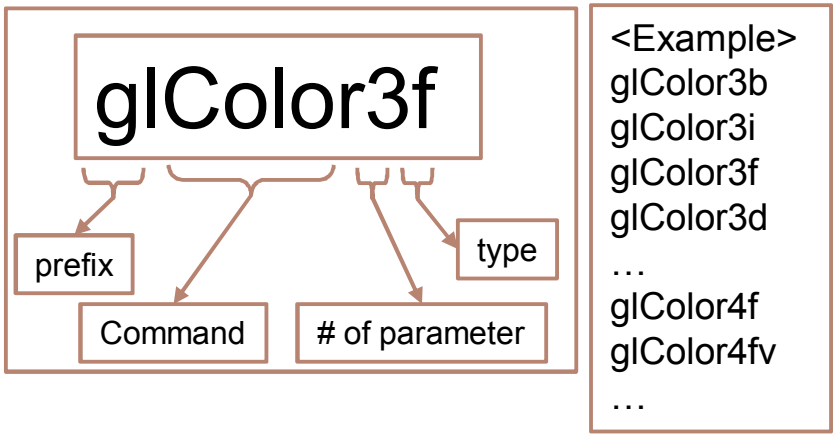
What is OpenGL? – cont'

- What can we do with OpenGL?
  - Modeling Primitives
  - Drawing Curve/Surfaces
  - Colors and Shading
  - Lights and Shadows
  - Texture mapping
  - Fog / Anti-aliasing
  - Blending / Transparency
  - And… So many things.



What is OpenGL? – cont'

- OpenGL Command Syntax
  - About 130 functions.
  - All of functions has a prefix gl
    - Ex) glTranslate3f()
  - All of variables has a prefix GL_
    - Ex) GL_LIGHTING



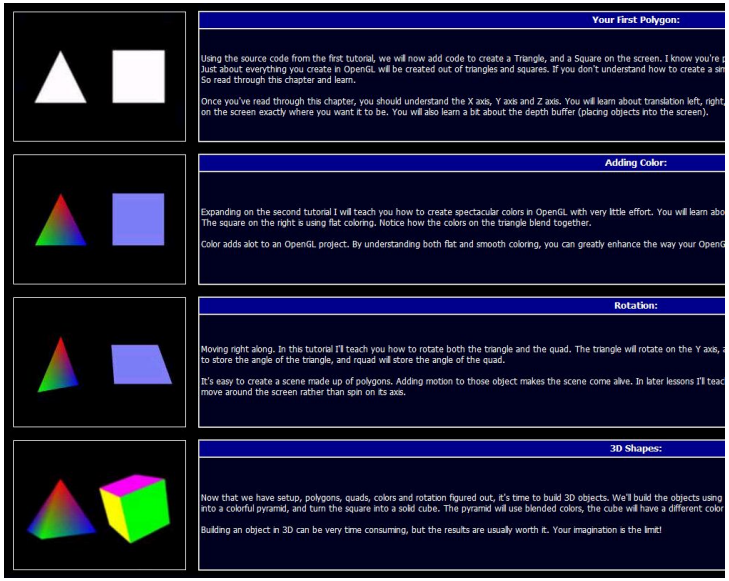
<Example>
glColor3b
glColor3i
glColor3f
glColor3d
…
glColor4f
glColor4fv
…

Suffix	C++ type	OpenGL Type Definition
b	signed char	GLbyte
s	short	GLshort
i	long	GLint, GLsizei
f	float	GLfloat, GLclampf
d	double	GLdouble, GLclampd
ub	unsigned char	GLubyte, GLboolean
us	unsigned short	GLushort
ui	unsigned long	GLuint, GLenum, GLbitfield

OpenGL programming with WinAPI

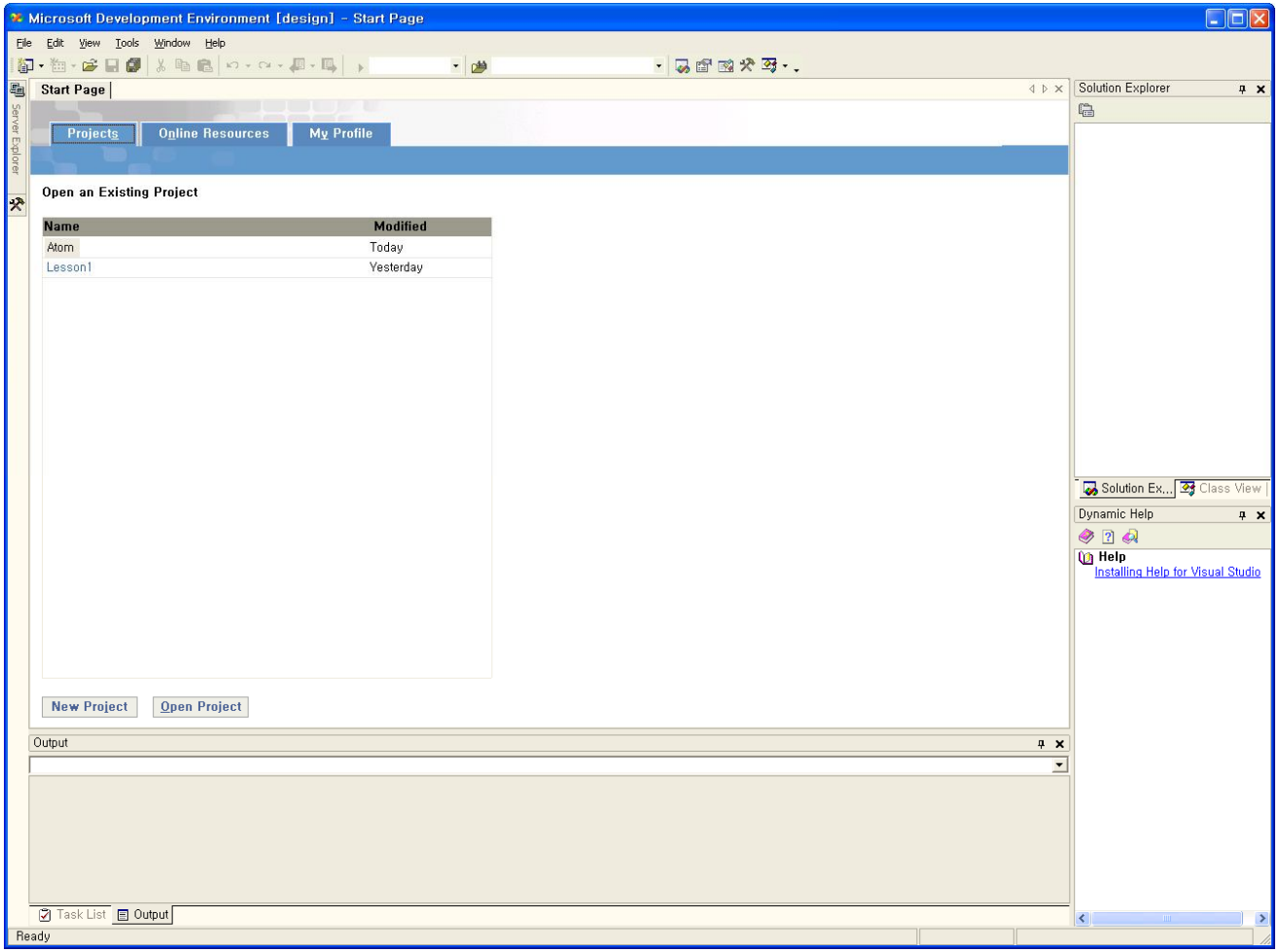
- To do…
  - Draw 2D polygon
  - Coloring the object
  - Rotate the object
  - Draw 3D object

- Reference
  - http://nehe.gamedev.net
  - Lesson 01~05



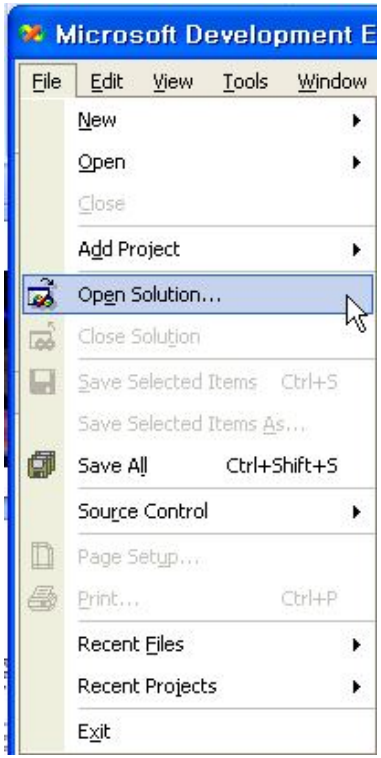
Open Source Code

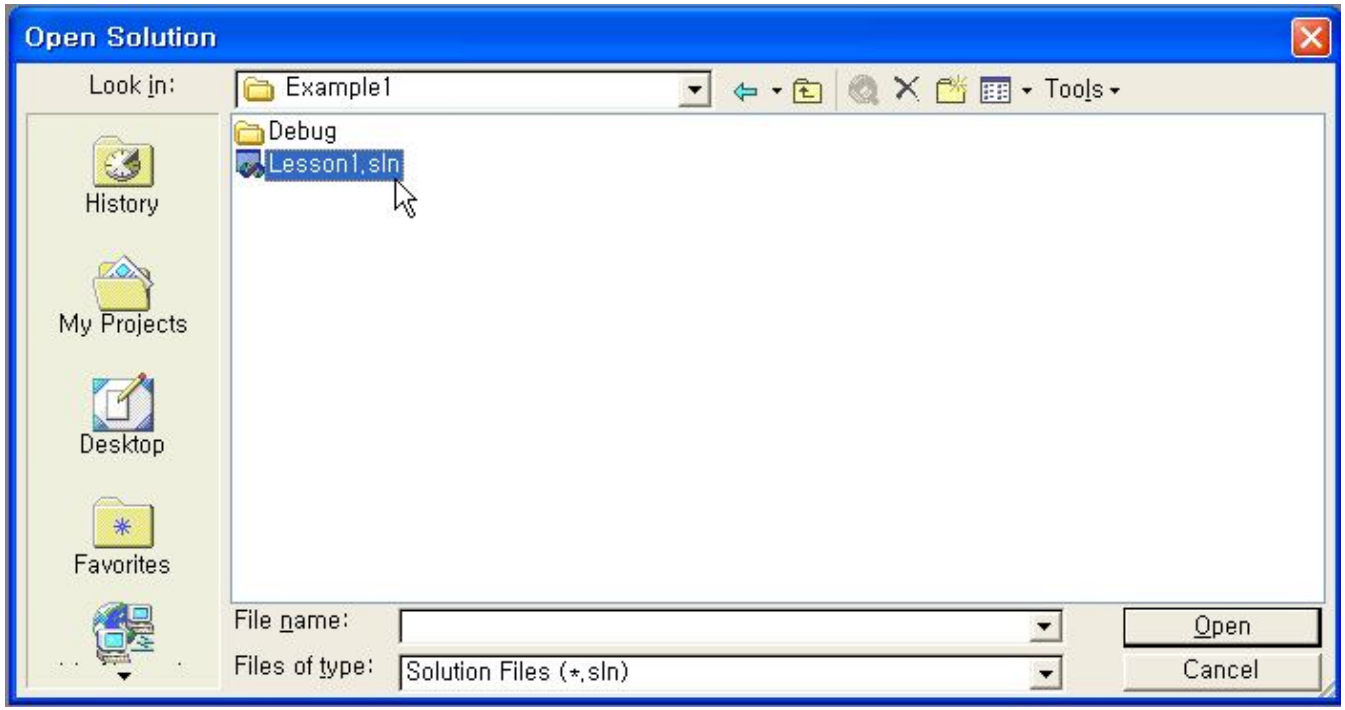
- Open Visual Studio .NET 2003



Open Source Code – cont’

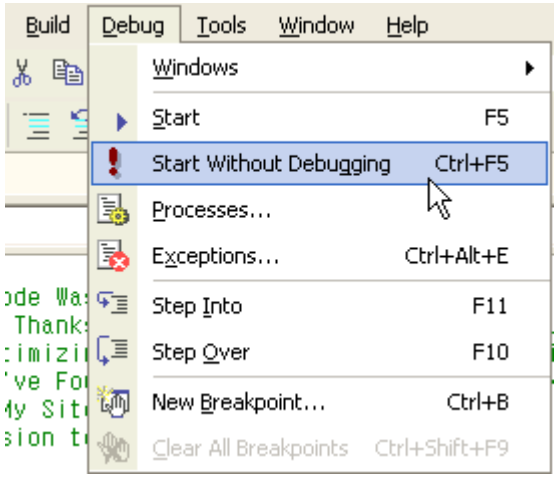
- Open Example1 (Lesson1.sln)
  - File >> Open Solution >> Lesson1.sln

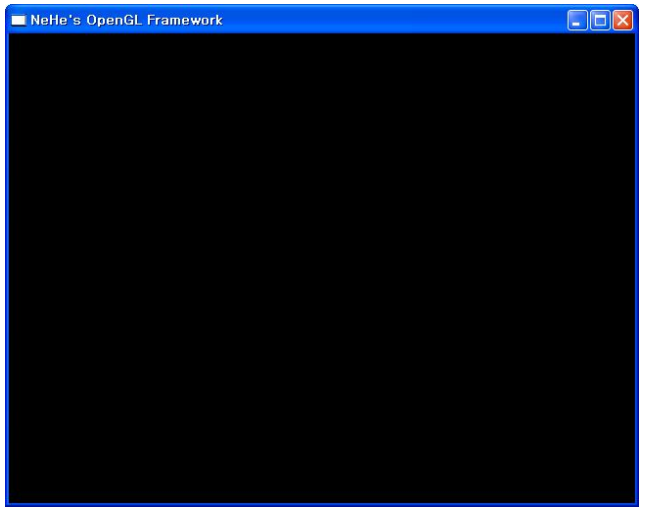




Open Source Code – cont'

- Execute the base code
  - Debug >> Start Without Debugging >> (Yes) >> (No)





About Source Code

- GLU, GLAUX libraries are used.
- WinAPI functions are used to generate windows.
- Used Functions
  - ReSizeGLScene()
  - InitGL()
  - KillGLWindow()
  - CreateGLWindow()
  - WinMain()
  - DrawGLScene()
- We will keep modifying DrawGLScene() to create different scenes.

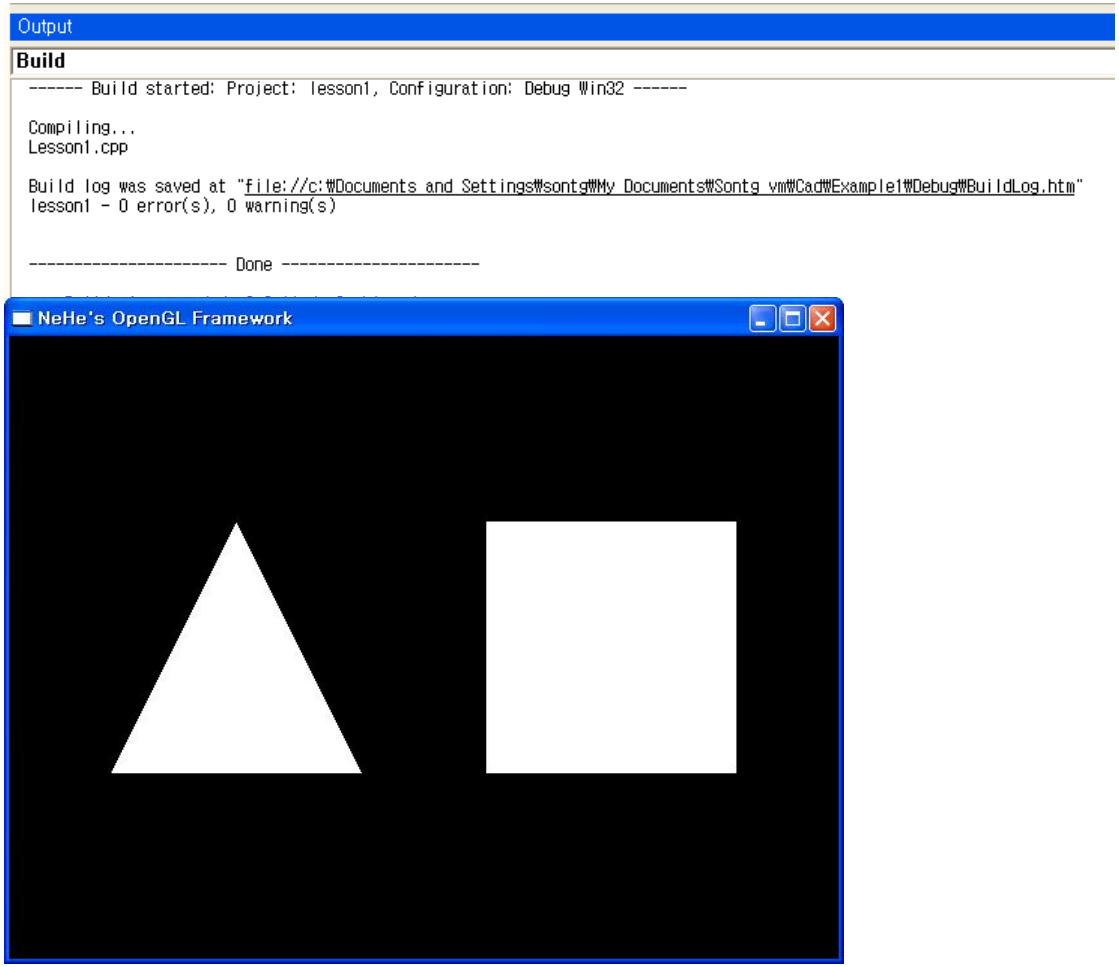
Draw a Triangle & Quad

- In DrawGLScene() function
  - Draw Triangle and Quad using glBegin() and glEnd()

```
int DrawGLScene(GLvoid)                                    // Here's Where We Do All The Drawing
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear Screen And Depth Buffer
    glLoadIdentity();                                      // Reset The Current Modelview Matrix

    glTranslatef(-1.5f, 0.0f, -6.0f); // Move Left 1.5 Units And Into The Screen 6.0
    glBegin(GL_TRIANGLES); // Drawing Using Triangles
        glVertex3f( 0.0f, 1.0f, 0.0f); // Top
        glVertex3f(-1.0f,-1.0f, 0.0f); // Bottom Left
        glVertex3f( 1.0f,-1.0f, 0.0f); // Bottom Right
    glEnd(); // Finished Drawing The Triangle
    glTranslatef(3.0f,0.0f,0.0f); // Move Right 3 Units
    glBegin(GL_QUADS); // Draw A Quad
        glVertex3f(-1.0f, 1.0f, 0.0f); // Top Left
        glVertex3f( 1.0f, 1.0f, 0.0f); // Top Right
        glVertex3f( 1.0f,-1.0f, 0.0f); // Bottom Right
        glVertex3f(-1.0f,-1.0f, 0.0f); // Bottom Left
    glEnd(); // Done Drawing The Quad

    return TRUE;                                           // Everything Went OK
}
```

Execute

- Compile with ‘Ctrl+F7’ and Execute with ‘Ctrl+F5’



Compiling… (Can be skipped)

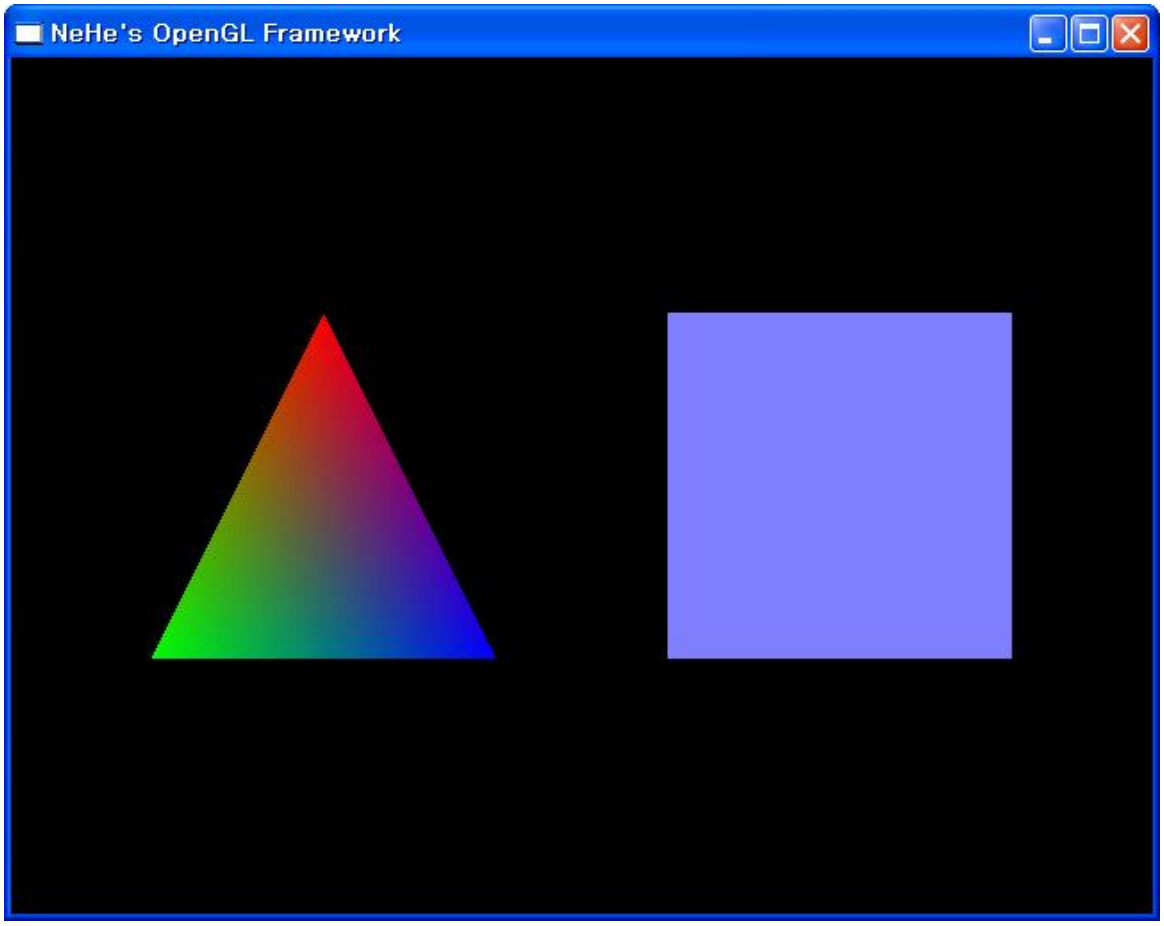
Coloring

- Modify DrawGLScene() using glColor3f()

```
int DrawGLScene(GLvoid)                                     // Here's Where We Do All The Drawing
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear Screen And Depth Buffer
    glLoadIdentity();                                       // Reset The Current Modelview Matrix

    glTranslatef(-1.5f, 0.0f, -6.0f); // Move Left 1.5 Units And Into The Screen 6.0
    glBegin(GL_TRIANGLES); // Drawing Using Triangles
        glColor3f(1.0f, 0.0f, 0.0f); // Set the color to red
        glVertex3f( 0.0f, 1.0f, 0.0f); // Top
        glColor3f(0.0f, 1.0f, 0.0f); // change the color of second vertex to green
        glVertex3f(-1.0f,-1.0f, 0.0f); // Bottom Left
        glColor3f(0.0f, 0.0f, 1.0f); // change the color of third vertex to blue
        glVertex3f( 1.0f,-1.0f, 0.0f); // Bottom Right
    glEnd(); // Finished Drawing The Triangle

    glTranslatef(3.0f,0.0f,0.0f); // Move Right 3 Units
    glColor3f(0.5f,0.5f,1.0f); //Set The Color To Blue One Time Only
    glBegin(GL_QUADS); // Draw A Quad
        glVertex3f(-1.0f, 1.0f, 0.0f); // Top Left
        glVertex3f( 1.0f, 1.0f, 0.0f); // Top Right
        glVertex3f( 1.0f,-1.0f, 0.0f); // Bottom Right
        glVertex3f(-1.0f,-1.0f, 0.0f); // Bottom Left
    glEnd(); // Done Drawing The Quad

    return TRUE;                                            // Everything Went OK
}
```

Execute

- ‘Ctrl+F5’



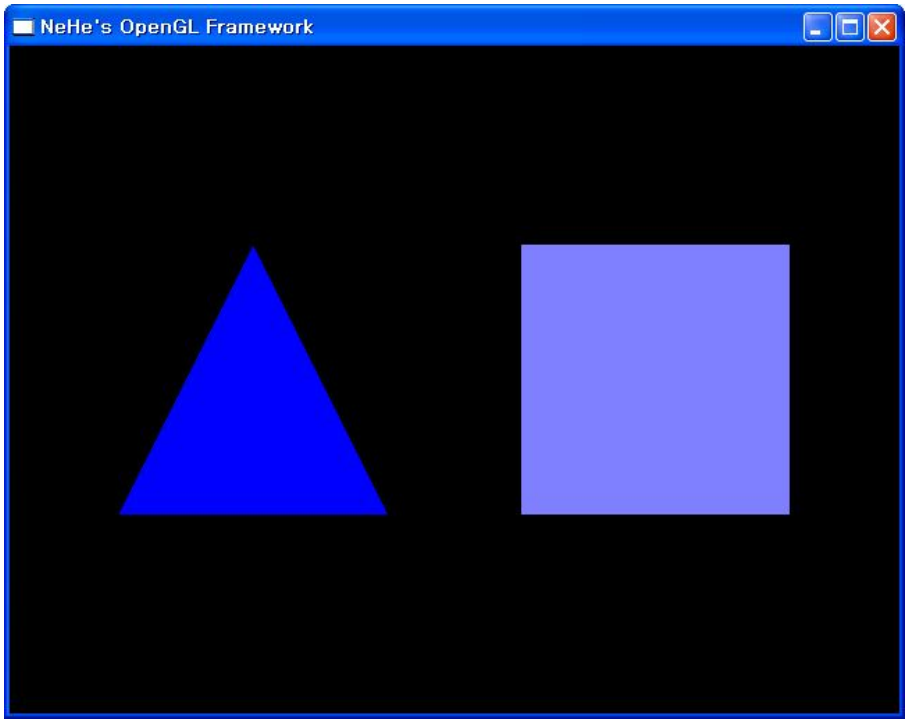
Shade Mode

- In InitGL() function
  - GL_SMOOTH → GL_FLAT

```
int InitGL(GLvoid)
{
    glShadeModel(GL_SMOOTH);

    glClearColor(0.0f, 0.0f, 0.0f, 0.5f);
    glClearDepth(1.0f);
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LEQUAL);
    glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
    return TRUE;
}
```

```
glShadeModel(GL_FLAT);
```

```
// All Setup For OpenGL Goes Here
```



Result of Flat Shading Mode

Rotate the objects

- Declare variables for Rotation Angle

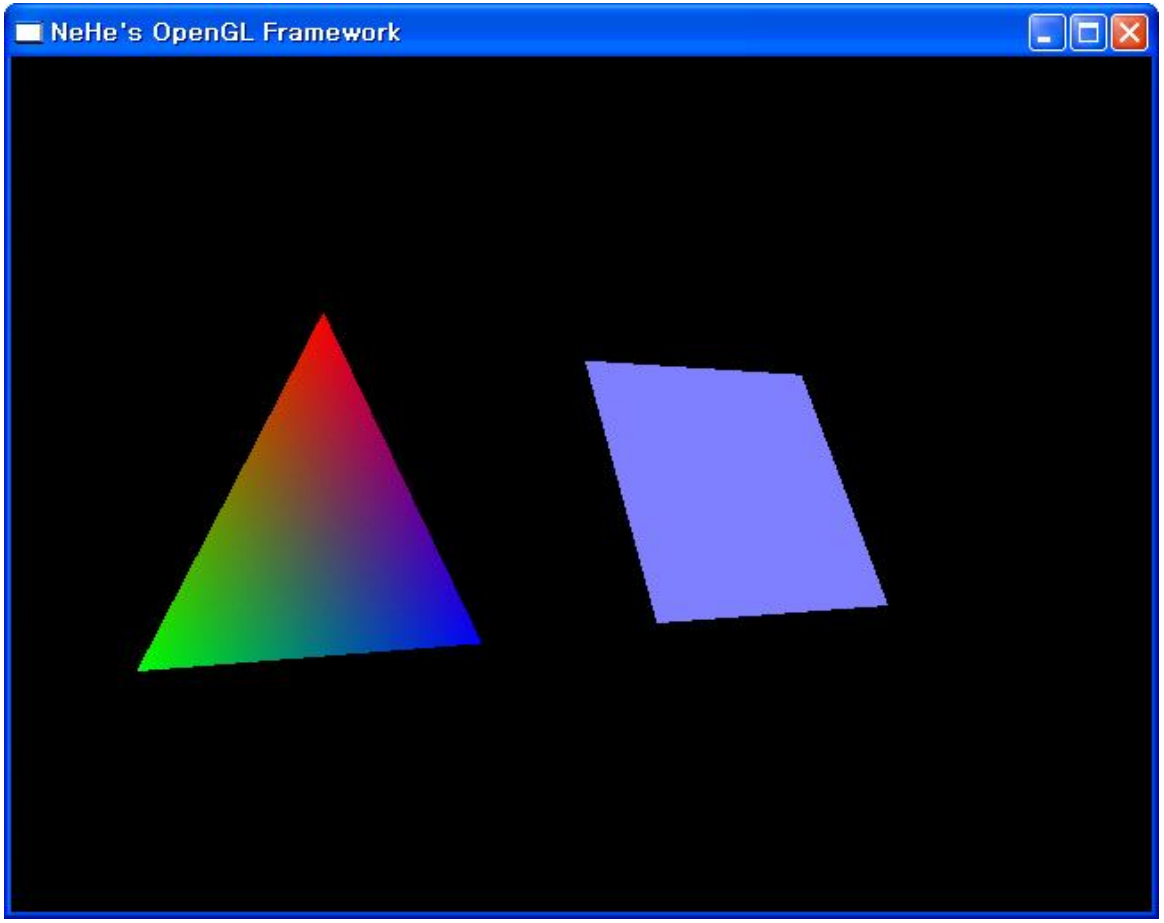
```
GLfloat rtri;                // Rotation Angle for the Triangle
GLfloat rquad;               // Rotation Angle for the Quad
```

- Rotate the objects

```
int DrawGLScene(GLvoid)                                    // Here's Where We Do All The Drawing
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear Screen And Depth Buffer
    glLoadIdentity();                                      // Reset The Current Modelview Matrix
    glTranslatef(-1.5f, 0.0f, -6.0f); // Move Left 1.5 Units And Into The Screen 6.0
    glRotatef(rtri, 0.0f, 1.0f, 0.0f);  // Rotate The Triangle On The Y-axis
    glBegin(GL_TRIANGLES); // Drawing Using Triangles
        glColor3f(1.0f, 0.0f, 0.0f); // Set the color to red
        glVertex3f( 0.0f, 1.0f, 0.0f); // Top
        glColor3f(0.0f, 1.0f, 0.0f); // change the color of second vertex to green
        glVertex3f(-1.0f,-1.0f, 0.0f); // Bottom Left
        glColor3f(0.0f, 0.0f, 1.0f); // change the color of third vertex to blue
        glVertex3f( 1.0f,-1.0f, 0.0f); // Bottom Right
    glEnd(); // Finished Drawing The Triangle
    glTranslatef(3.0f,0.0f,0.0f); // Move Right 3 Units
    glRotatef(rquad, 1.0f, 0.0f, 0.0f); // Rotate The Quad On The X-axis
    glColor3f(0.5f,0.5f,1.0f); //Set The Color To Blue One Time Only
    glBegin(GL_QUADS); // Draw A Quad
        glVertex3f(-1.0f, 1.0f, 0.0f); // Top Left
        glVertex3f( 1.0f, 1.0f, 0.0f); // Top Right
        glVertex3f( 1.0f,-1.0f, 0.0f); // Bottom Right
        glVertex3f(-1.0f,-1.0f, 0.0f); // Bottom Left
    glEnd(); // Done Drawing The Quad
    rtri+=0.2f;      // Increase The Rotation Variable For The Triangle
    rquad-=0.15f;    // Decrease The Rotation Variable For The Quad
    return TRUE;                                           // Everything Went OK
}
```

Execute

- ‘Ctrl+F5’



3D Objects – Pyramid

- Modify glBegin()~glEnd() using following

```
glBegin(GL_TRIANGLES);
//Start Drawing The Pyramid
        // Draw Front Face
        glColor3f(1.0f,0.0f,0.0f);
        glVertex3f( 0.0f, 1.0f, 0.0f);
        glColor3f(0.0f,1.0f,0.0f);
        glVertex3f(-1.0f,-1.0f, 1.0f);
        glColor3f(0.0f,0.0f,1.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);

        // Draw Right Face
        glColor3f(1.0f,0.0f,0.0f);
        glVertex3f( 0.0f, 1.0f, 0.0f);
        glColor3f(0.0f,0.0f,1.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);
        glColor3f(0.0f,1.0f,0.0f);
        glVertex3f( 1.0f,-1.0f, -1.0f);
```

```
        // Draw Back Face
        glColor3f(1.0f,0.0f,0.0f);
        glVertex3f( 0.0f, 1.0f, 0.0f);
        glColor3f(0.0f,1.0f,0.0f);
        glVertex3f( 1.0f,-1.0f, -1.0f);
        glColor3f(0.0f,0.0f,1.0f);
        glVertex3f(-1.0f,-1.0f, -1.0f);

        //Draw Left Face
        glColor3f(1.0f,0.0f,0.0f);
        glVertex3f( 0.0f, 1.0f, 0.0f);
        glColor3f(0.0f,0.0f,1.0f);
        glVertex3f(-1.0f,-1.0f,-1.0f);
        glColor3f(0.0f,1.0f,0.0f);
        glVertex3f(-1.0f,-1.0f, 1.0f);
glEnd();
// Done Drawing The Pyramid
```

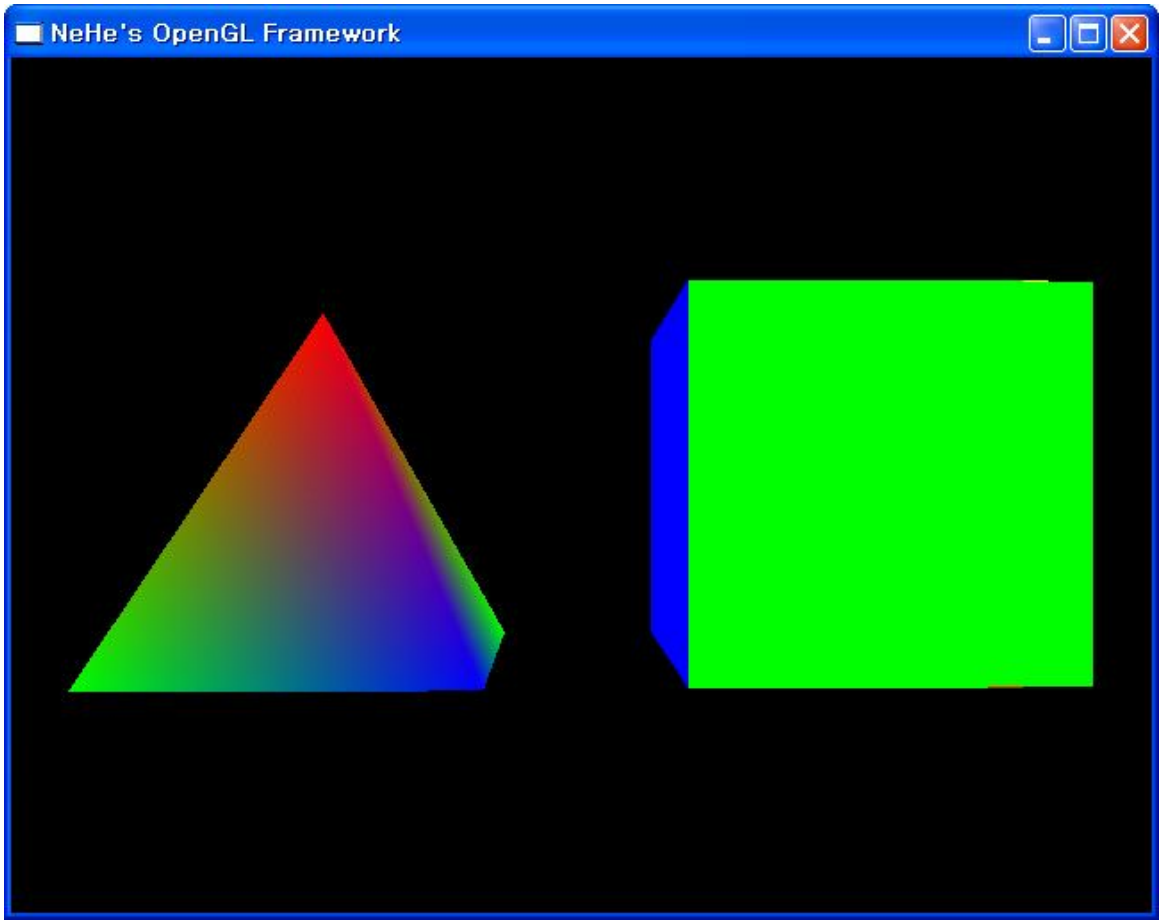
3D Objects – Cube

```
glBegin(GL_QUADS);
//Start Drawing The Cube
        // Draw Top Face (Green)
        glColor3f(0.0f,1.0f,0.0f);
        glVertex3f( 1.0f, 1.0f,-1.0f);
        glVertex3f(-1.0f, 1.0f,-1.0f);
        glVertex3f(-1.0f, 1.0f, 1.0f);
        glVertex3f( 1.0f, 1.0f, 1.0f);

        // Draw Bottom Face (Orange)
        glColor3f(1.0f,0.5f,0.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);
        glVertex3f(-1.0f,-1.0f, 1.0f);
        glVertex3f(-1.0f,-1.0f,-1.0f);
        glVertex3f( 1.0f,-1.0f,-1.0f);

        // Draw Front Face (Red)
        glColor3f(1.0f,0.0f,0.0f);
        glVertex3f( 1.0f, 1.0f, 1.0f);
        glVertex3f(-1.0f, 1.0f, 1.0f);
        glVertex3f(-1.0f,-1.0f, 1.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);
```

```
        // Draw Back Face (Yellow)
        glColor3f(1.0f,1.0f,0.0f);
        glVertex3f( 1.0f,-1.0f,-1.0f);
        glVertex3f(-1.0f,-1.0f,-1.0f);
        glVertex3f(-1.0f, 1.0f,-1.0f);
        glVertex3f( 1.0f, 1.0f,-1.0f);

        // Draw Left Face (Blue)
        glColor3f(0.0f,0.0f,1.0f);
        glVertex3f(-1.0f, 1.0f, 1.0f);
        glVertex3f(-1.0f, 1.0f,-1.0f);
        glVertex3f(-1.0f,-1.0f,-1.0f);
        glVertex3f(-1.0f,-1.0f, 1.0f);

        // Draw Right Face (Violet)
        glColor3f(1.0f,0.0f,1.0f);
        glVertex3f( 1.0f, 1.0f,-1.0f);
        glVertex3f( 1.0f, 1.0f, 1.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);
        glVertex3f( 1.0f,-1.0f, -1.0f);
glEnd();
 // Done Drawing The Cube
```

Execute

- ‘Ctrl+F5’



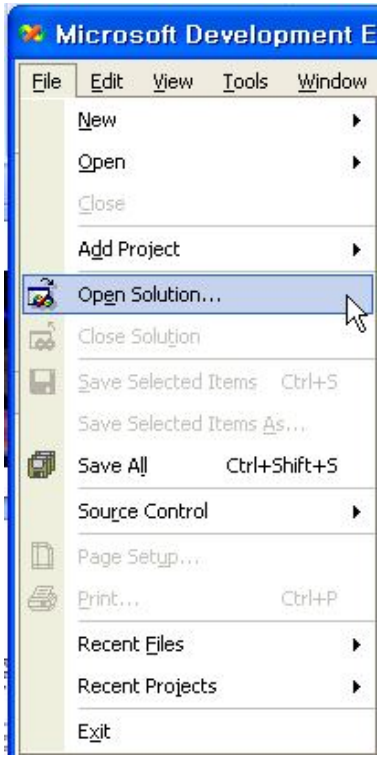
OpenGL programming with GLUT

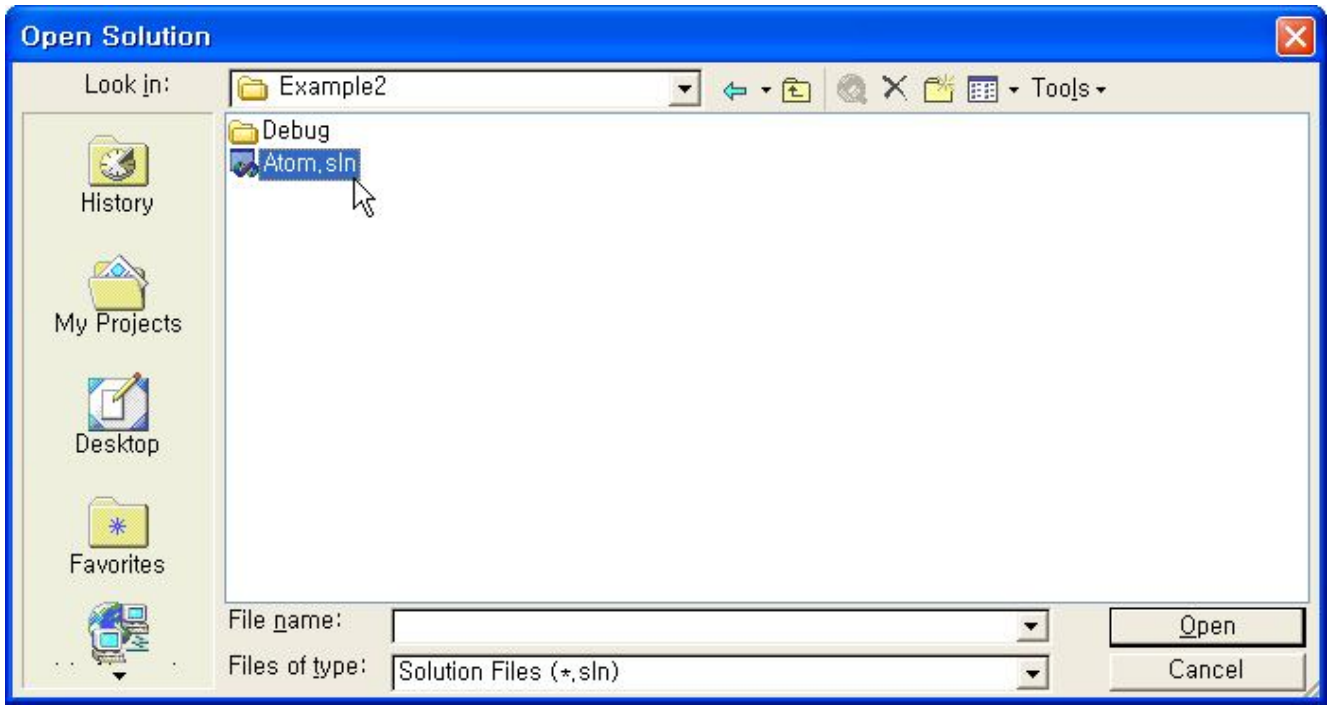
- GLUT
  - Graphic Library Utility Toolkit
  - Prefix “glut”

- To do…
  - Draw primitives using GLUT functions (Solid sphere)
  - Rotate the objects
  - Push / Pop Matrix
  - Parallel vs. Perspective projection
  - Lights

Open Source Code

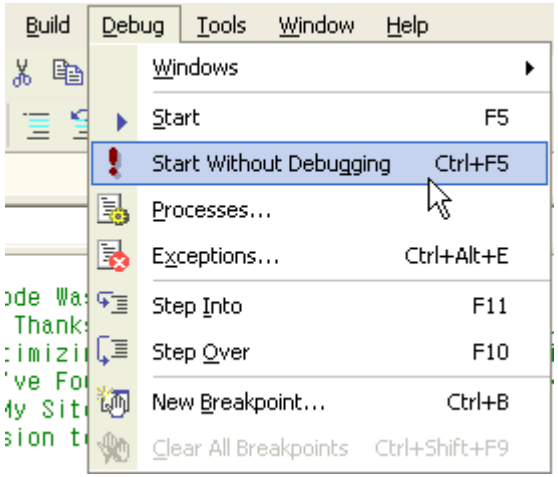
- Open Example2 (Atom.sln)
  - File >> Open Solution >> Atom.sln

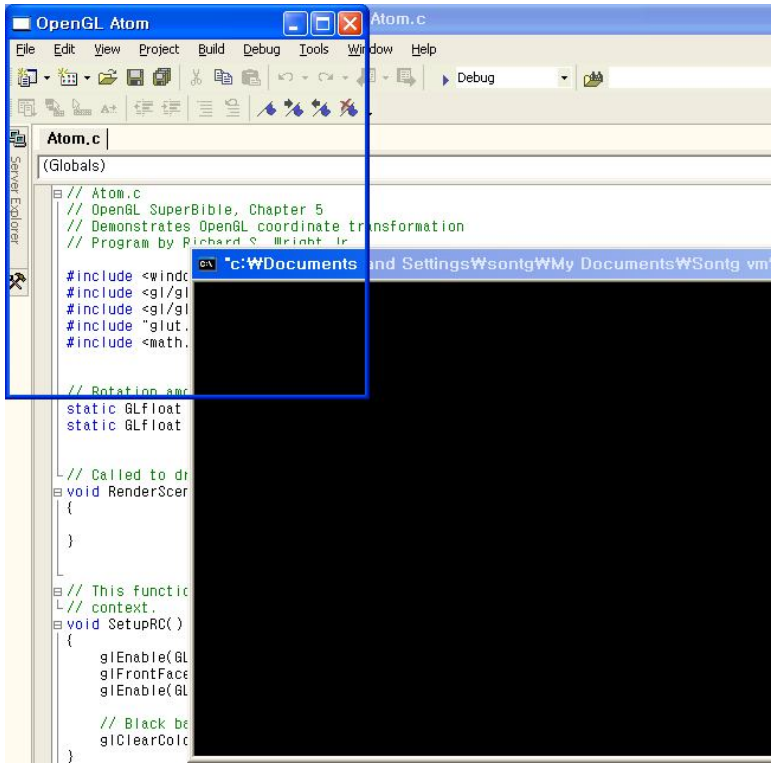




Open Source Code – cont'

- Execute the base code
  - Debug >> Start Without Debugging >> (Yes)





About Source Code

- Used Functions
  - RenderScene()
  - SetupRC()
  - SpecialKeys()
  - TimerFunc()
  - ChangeSize()
  - main()
- We will mainly modify RenderScene() to create different scene.
- ChangeSize() and SetupRC() will be modified for projection and lighting.

Draw a sphere

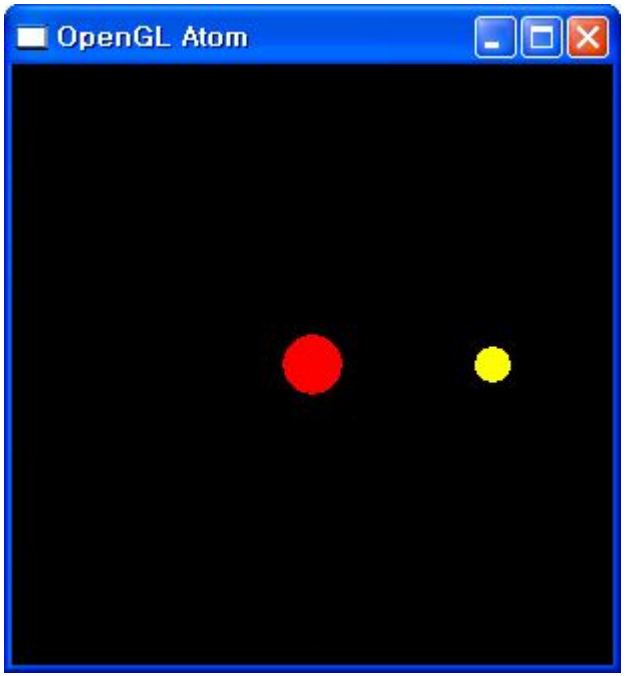
- Modify RenderScene()
  - Using glutSolidSphere()

```
void RenderScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // Clear The Color And The Depth Buffer
    glMatrixMode(GL_MODELVIEW); // Specify ModelView matrix as current
    //matrix on which subsequent matrix operations act
    glLoadIdentity(); // Reset ModelView Matrix to Identity
    glTranslatef(0.0f, 0.0f, -100.0f); // Move 100 Units Along -Z Axis
    glColor3ub(255, 0, 0); // Change The Color To Red
    glutSolidSphere(10.0f, 15, 15); // Create A Sphere of Radius 10
    glutSwapBuffers(); // Swap the buffers to see the rending result.
}
```

Draw a smaller sphere

- With different radius

```
void RenderScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // Clear The Color And The Depth Buffer
    glMatrixMode(GL_MODELVIEW); // Specify ModelView matrix as current
    //matrix on which subsequent matrix operations act
    glLoadIdentity(); // Reset ModelView Matrix to Identity

    glTranslatef(0.0f, 0.0f, -100.0f); // Move 100 Units Along -Z Axis
    glColor3ub(255, 0, 0); // Change The Color To Red
    glutSolidSphere(10.0f, 15, 15); // Create A Sphere of Radius 10

    glTranslatef(60.0f, 0.0f, 0.0f); // Move 60 Units Along X Axis
    glColor3ub(255,255,0); // Change The Color To Yellow.
    glutSolidSphere(6.0f, 15, 15); // Draw A Solid Sphere With Radius Of 6.

    glutSwapBuffers(); // Swap the buffers to see the rending result.
}
```



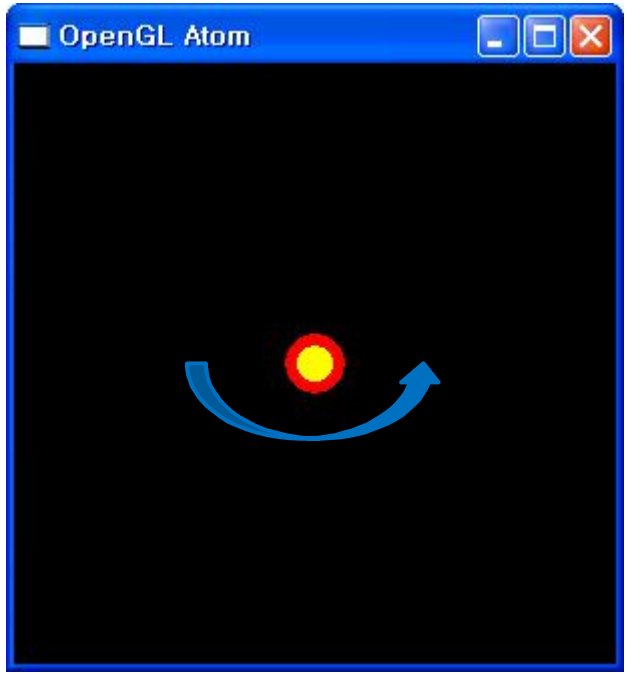
Add motion

- Yellow ball making a circle around Red ball

```
void RenderScene(void)
{
    static float fElect1 = 0.0f; //define a static variable for rotation angle.

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // Clear The Color And The Depth Buffer
    glMatrixMode(GL_MODELVIEW); // Specify ModelView matrix as current
    //matrix on which subsequent matrix operations act
    glLoadIdentity(); // Reset ModelView Matrix to Identity

    glTranslatef(0.0f, 0.0f, -100.0f); // Move 100 Units Along -Z Axis
    glColor3ub(255, 0, 0); // Change The Color To Red
    glutSolidSphere(10.0f, 15, 15); // Create A Sphere of Radius 10

    glColor3ub(255,255,0); // Change The Color To Yellow.
    glRotatef(fElect1, 0.0f, 1.0f, 0.0f); // Rotate fElect1 angle about Y axis
    glTranslatef(60.0f, 0.0f, 0.0f); // Move 60 Units Along X Axis
    glutSolidSphere(6.0f, 15, 15); // Draw A Solid Sphere With Radius Of 6.

    fElect1 += 10.0f; // Increase the rotation angle
    if(fElect1 > 360.0f)
        fElect1 = 0.0f;

    glutSwapBuffers(); // Swap the buffers to see the rending result.
}
```

Execute

- ‘Ctrl+F5’



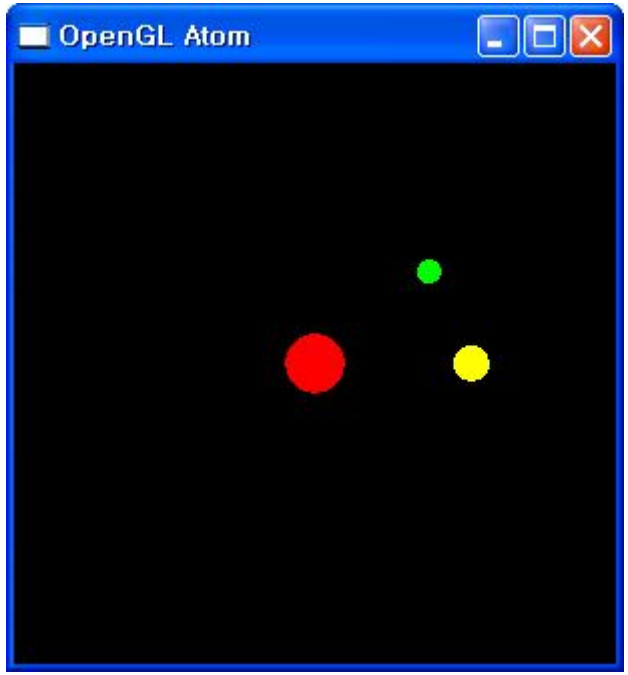
Draw another sphere

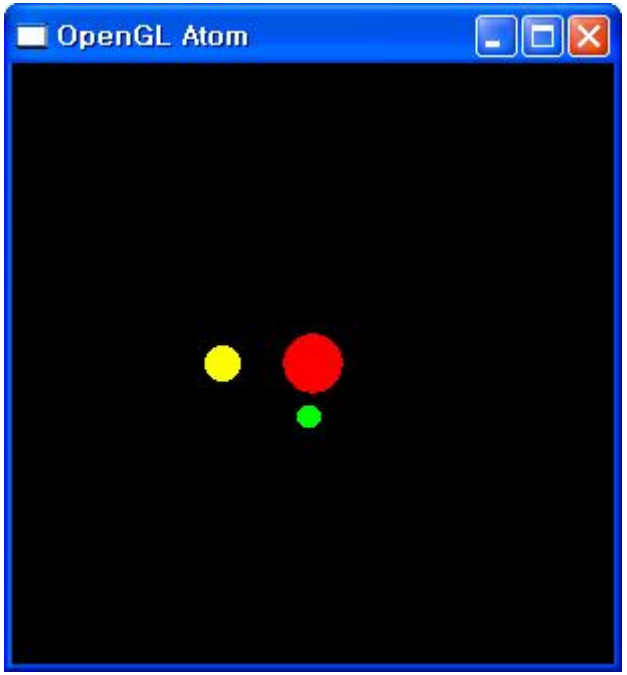
- Green ball which circle around the yellow ball

```
void RenderScene(void)
{
    static float fElect1 = 0.0f; //define a static variable for rotation angle.

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // Clear The Color And The Depth Buffer
    glMatrixMode(GL_MODELVIEW); // Specify ModelView matrix as current
    //matrix on which subsequent matrix operations act
    glLoadIdentity(); // Reset ModelView Matrix to Identity

    glTranslatef(0.0f, 0.0f, -100.0f); // Move 100 Units Along -Z Axis
    glColor3ub(255, 0, 0); // Change The Color To Red
    glutSolidSphere(10.0f, 15, 15); // Create A Sphere of Radius 10

    glColor3ub(255,255,0); // Change The Color To Yellow.
    glRotatef(fElect1, 0.0f, 1.0f, 0.0f); // Rotate fElect1 angle about Y axis
    glTranslatef(60.0f, 0.0f, 0.0f); // Move 60 Units Along X Axis
    glutSolidSphere(6.0f, 15, 15); // Draw A Solid Sphere With Radius Of 6.

    glColor3ub(0,255,0); // Change The Color To Green
    glRotatef(45.0f, 0.0f, 0.0f, 1.0f); // Rotate 45 degree about Z axis
    glRotatef(fElect1, 0.0f, 1.0f, 0.0f); // Rotate fElect1 angle about Y axis
    glTranslatef(-50.0f, 0.0f, 0.0f); // Translate 50 unites along minus X axis
    glutSolidSphere(4.0f, 15, 15); // Draw a sphere of radius

    fElect1 += 10.0f; // Increase the rotation angle
    if(fElect1 > 360.0f)
        fElect1 = 0.0f;

    glutSwapBuffers(); // Swap the buffers to see the rending result.
}
```

Execute

- ‘Ctrl+F5’



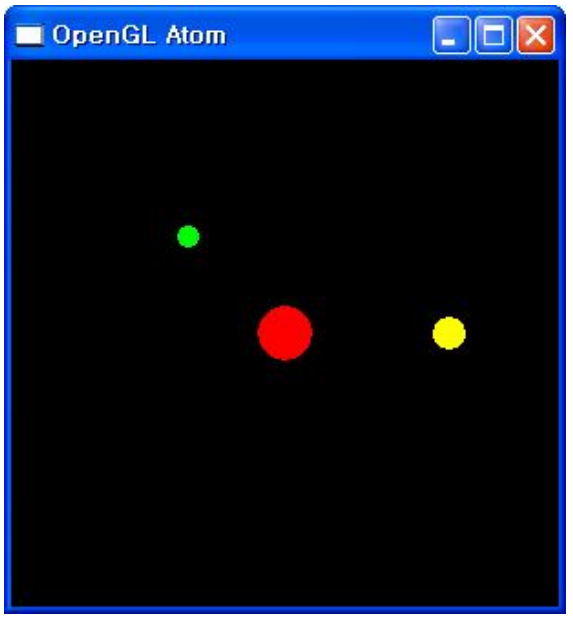


The movement of Green ball is Strange!!!

Push / Pop Matrix

- Push Matrix
  - Store current transformation state in Stack
- Pop Matrix
  - Recover the saved state from Stack

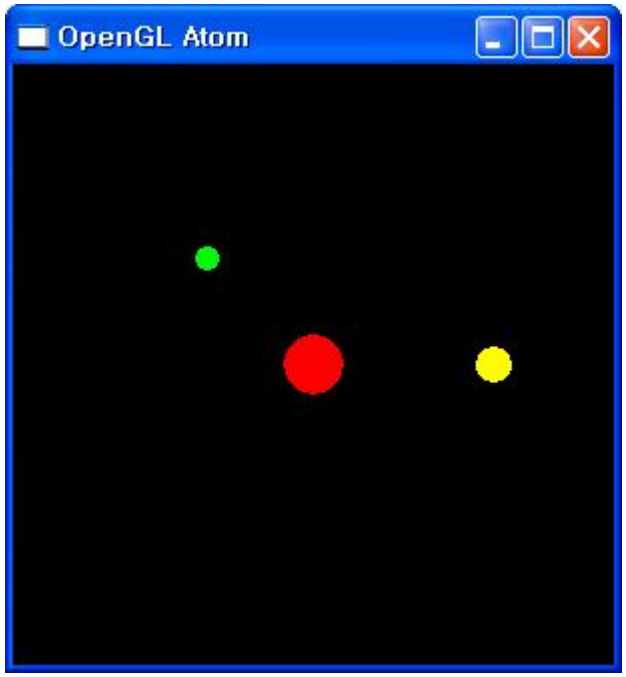
```
glTranslatef(0.0f, 0.0f, -100.0f); // Move 100 Units Along -Z Axis
glColor3ub(255, 0, 0); // Change The Color To Red
glutSolidSphere(10.0f, 15, 15); // Create A Sphere of Radius 10

glPushMatrix();     // Push matrix stack to keep the current transformation state
glColor3ub(255,255,0); // Change The Color To Yellow.
glRotatef(fElect1, 0.0f, 1.0f, 0.0f); // Rotate fElect1 angle about Y axis
glTranslatef(60.0f, 0.0f, 0.0f); // Move 60 Units Along X Axis
glutSolidSphere(6.0f, 15, 15); // Draw A Solid Sphere With Radius Of 6.
glPopMatrix();      // Pop up the matrix to recover the saved state

glColor3ub(0,255,0); // Change The Color To Green
glRotatef(45.0f, 0.0f, 0.0f, 1.0f); // Rotate 45 degree about Z axis
glRotatef(fElect1, 0.0f, 1.0f, 0.0f); // Rotate fElect1 angle about Y axis
glTranslatef(-50.0f, 0.0f, 0.0f); // Translate 50 unites along minus X axis
glutSolidSphere(4.0f, 15, 15); // Draw a sphere of radius
```



The movement of balls are reasonable now!!!

Projection Mode

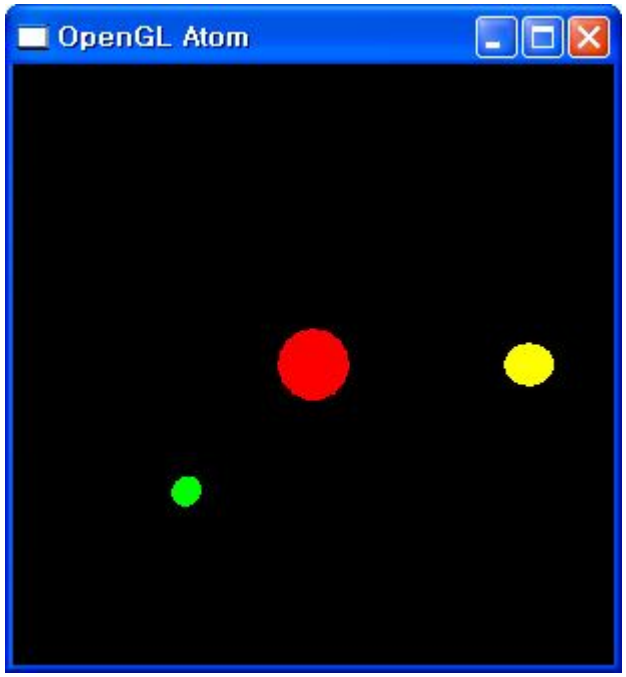
- Parallel Projection
  - glOrtho()




```
void ChangeSize(int w, int h)
{
    GLfloat nRange = 100.0f;

    if(h == 0)      // Prevent a divide by zero
        h = 1;

    glViewport(0, 0, w, h); // Set Viewport to window dimensions

    glMatrixMode(GL_PROJECTION);    // Reset coordinate system
    glLoadIdentity();

    // Establish clipping volume (left, right, bottom, top, near, far)
    if (w <= h)
        glOrtho (-nRange, nRange, nRange*h/w, -nRange*h/w, -nRange*2.0f, nRange*2.0f);
    else
        glOrtho (-nRange*w/h, nRange*w/h, nRange, -nRange, -nRange*2.0f, nRange*2.0f);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
```

Projection Mode – cont'

- Perspective Projection
  - gluPerspective()




```
void ChangeSize(int w, int h)
{
    GLfloat fAspect; //ratio of width to height of window

    if(h == 0)      // Prevent a divide by zero
        h = 1;

    glViewport(0, 0, w, h); // Set Viewport to window dimensions

    glMatrixMode(GL_PROJECTION);     // Reset coordinate system
    glLoadIdentity( );

    fAspect = (float)w/(float)h; // calculate the ratio of width to height
//define perspective projection and its view clipping volume (angle, ratio, near, far)
    gluPerspective(80.0, fAspect, 1.0, 500.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity( );
//Translate the scene into screen (far from viewpoint). It is equal to changing viewpoint.
    glTranslatef(0.0f, 0.0f, -250.0f);
}
```

More realistic!!!

Lighting

- Modify SetupRC()

```
void SetupRC( )
{
    GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };//ambient property of the light
    GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };//diffuse property of the light
    GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };// specular property of the light
    GLfloat light_position[] = { 1.0, 100.0, 200.0, 0.0 };//position of light
    /*Set properties of the light*/
    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHTING); //Enable lighting
    glEnable(GL_LIGHT0);//Enable LIGHT0
    glEnable(GL_COLOR_MATERIAL); //Enable color material

    glEnable(GL_DEPTH_TEST);    // Hidden surface removal
    glFrontFace(GL_CCW);        // Counter clock-wise polygons face out
    glEnable(GL_CULL_FACE);     // Do not calculate inside of jet

    // Black background
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f );
}
```

Final Result