

Department of Computer Science and Engineering

Shivanirudh S G, 185001146, Semester VII

26 July 2021

UCS1712 - Graphics and Multimedia Lab

Exercise 1: Study of Basic Output Primitives in C++ using OpenGL

Objective:

To create an output window using OPENGL and to draw the following basic output primitives: POINTS, LINES, LINE_STRIP, LINE_LOOP, TRIANGLES, QUADS, QUAD_STRIP, POLYGON.

Code:

Common files

```
1 #ifndef LOPENGL_H
2 #define LOPENGL_H
3
4 #include <GL/freeglut.h>
```

```

5 #include <GL/gl.h>
6 #include <GL/glu.h>
7 #include <stdio.h>
8
9 #endif
10
11 #ifndef LUTIL_H
12 #define LUTIL_H
13
14 #include "LOpenGL.h"
15 #include <stdio.h>
16
17 //Screen Constants
18 const int SCREEN_WIDTH = 640;
19 const int SCREEN_HEIGHT = 480;
20 const int SCREEN_FPS = 60;
21
22 bool initGL();
23 /*
24 Pre Condition:
25 -A valid OpenGL context
26 Post Condition:
27 -Initializes matrices and clear color
28 -Reports to console if there was an OpenGL error
29 -Returns false if there was an error in initialization
30 Side Effects:
31 -Projection matrix is set to identity matrix
32 -Modelview matrix is set to identity matrix
33 -Matrix mode is set to modelview
34 -Clear color is set to black
35 */
36
37 void update();
38 /*
39 Pre Condition:
40 -None
41 Post Condition:
42 -Does per frame logic
43 Side Effects:
44 -None
45 */
46
47 void render();
48 /*
49 Pre Condition:
50 -A valid OpenGL context

```

```

41  -Active modelview matrix
42  Post Condition:
43  -Renders the scene
44  Side Effects:
45  -Clears the color buffer
46  -Swaps the front/back buffer
47  */
48
49  #endif

1  #include "LUtil.h"
2
3  void runMainLoop( int val );
4  /*
5  Pre Condition:
6  -Initialized freeGLUT
7  Post Condition:
8  -Calls the main loop functions and sets itself to be called
   back in 1000 / SCREEN_FPS milliseconds
9  Side Effects:
10 -Sets glutTimerFunc
11 */
12
13 int main( int argc, char* args[] )
14 {
15     //Initialize FreeGLUT
16     glutInit( &argc, args );
17
18     //Create OpenGL 2.1 context
19     glutInitContextVersion( 2, 1 );
20
21     //Create Singlele Buffered Window
22     glutInitDisplayMode( GLUT_SINGLE|GLUT_RGB );
23     glutInitWindowSize( SCREEN_WIDTH, SCREEN_HEIGHT );
24     glutCreateWindow( "OpenGL" );
25
26     //Do post window/context creation initialization
27     if( !initGL() )
28     {
29         printf( "Unable to initialize graphics library!\n" );
30         return 1;
31     }
32
33     //Set rendering function
34     glutDisplayFunc( render );
35

```

```

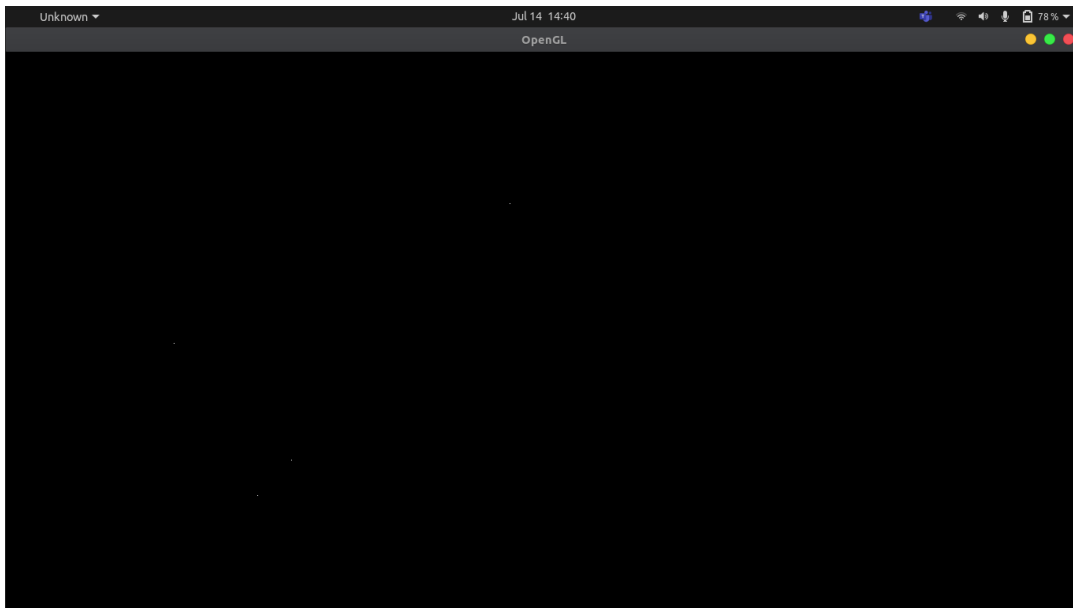
36     //Set main loop
37     glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, 0 );
38
39     //Start GLUT main loop
40     glutMainLoop();
41
42     return 0;
43 }
44
45 void runMainLoop( int val )
46 {
47     //Frame logic
48     update();
49     render();
50
51     //Run frame one more time
52     glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, val );
53 }

```

POINTS:

```
1 void render()  
2 {  
3     //Clear color buffer  
4     glClear(GL_COLOR_BUFFER_BIT);  
5     glBegin(GL_POINTS);  
6         glVertex2d(70,130);  
7         glVertex2d(100,230);  
8         glVertex2d(170,130);  
9         glVertex2d(300,350);  
10    glEnd();  
11    glFlush();  
12  
13    //Update screen  
14    //glutSwapBuffers();  
15 }
```

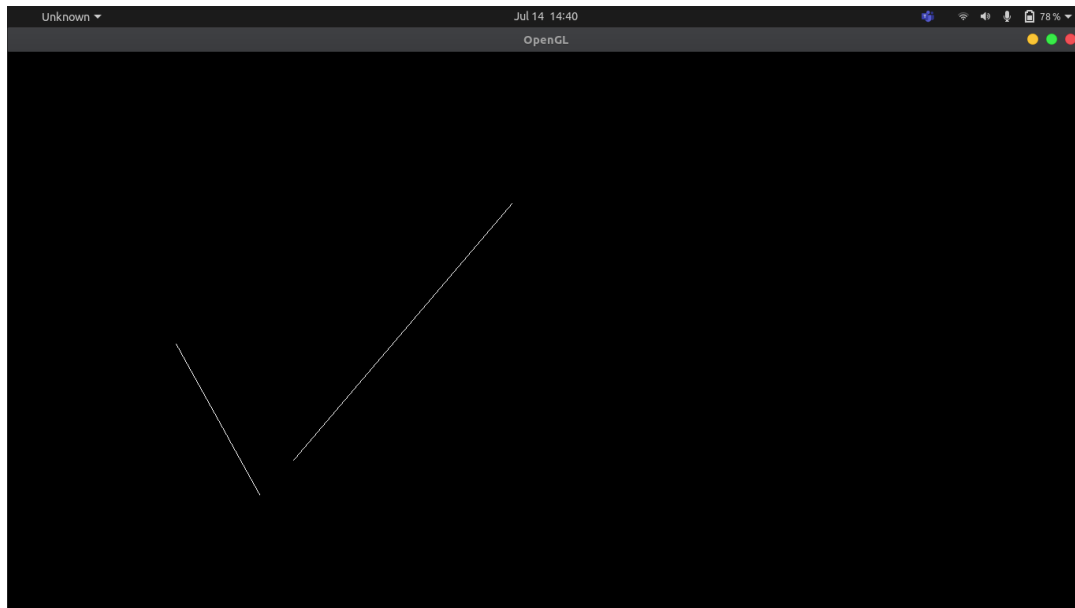
Output:



LINES:

```
1 void render()  
2 {  
3     //Clear color buffer  
4     glClear(GL_COLOR_BUFFER_BIT);  
5     glBegin(GL_LINES);  
6         glVertex2d(70,130);  
7         glVertex2d(100,230);  
8         glVertex2d(170,130);  
9         glVertex2d(300,350);  
10    glEnd();  
11    glFlush();  
12  
13    //Update screen  
14    //glutSwapBuffers();  
15 }
```

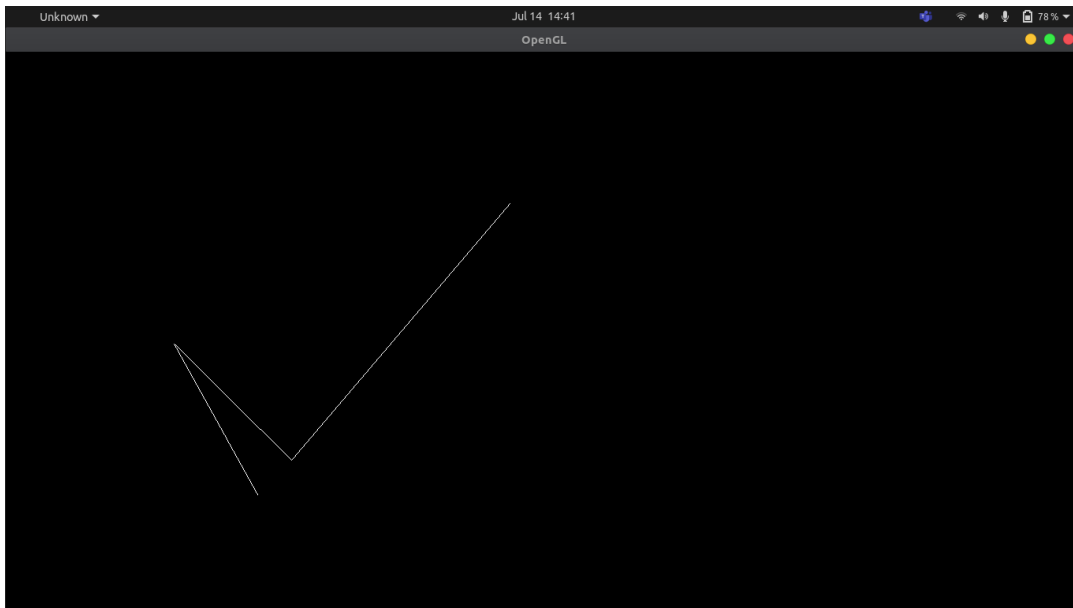
Output:



LINE_STRIP:

```
1 void render()  
2 {  
3     //Clear color buffer  
4     glClear(GL_COLOR_BUFFER_BIT);  
5     glBegin(GL_LINE_STRIP);  
6         glVertex2d(70,130);  
7         glVertex2d(100,230);  
8         glVertex2d(170,130);  
9         glVertex2d(300,350);  
10    glEnd();  
11    glFlush();  
12  
13    //Update screen  
14    //glutSwapBuffers();  
15 }
```

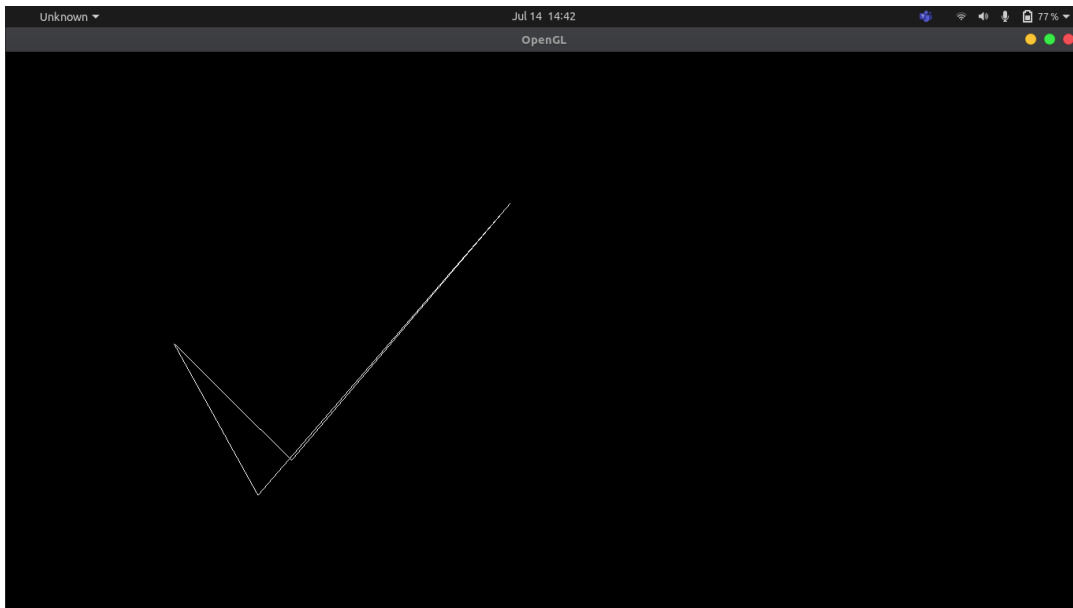
Output:



LINE_LOOP:

```
1 void render()
2 {
3     //Clear color buffer
4     glClear(GL_COLOR_BUFFER_BIT);
5     glBegin(GL_LINE_LOOP);
6         glVertex2d(70,130);
7         glVertex2d(100,230);
8         glVertex2d(170,130);
9         glVertex2d(300,350);
10    glEnd();
11    glFlush();
12
13    //Update screen
14    //glutSwapBuffers();
15 }
```

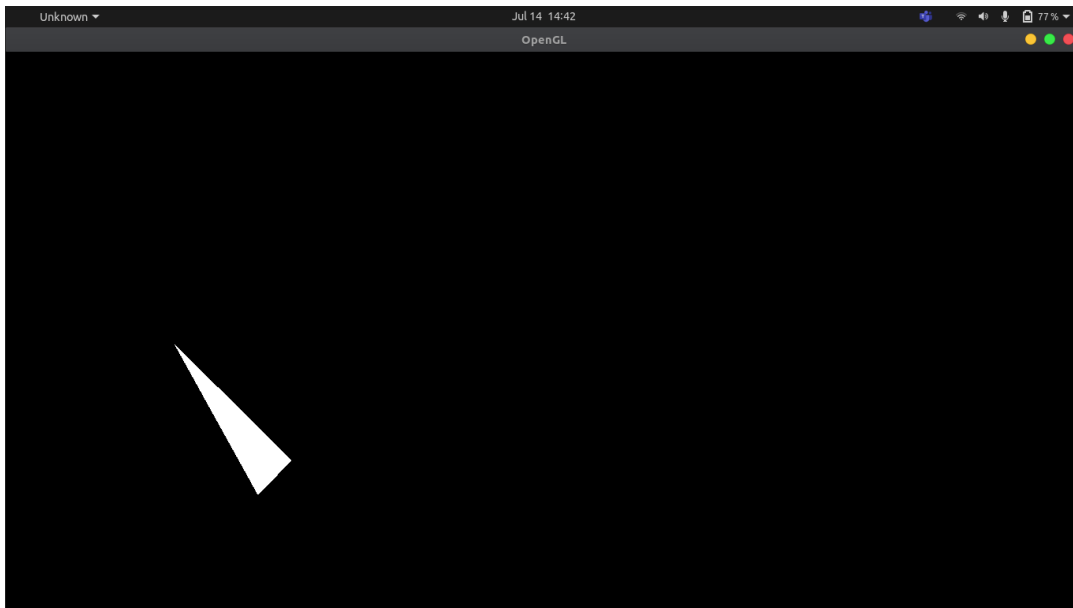
Output:



TRIANGLES:

```
1 void render()
2 {
3     //Clear color buffer
4     glClear(GL_COLOR_BUFFER_BIT);
5     glBegin(GL_TRIANGLES);
6         glVertex2d(70,130);
7         glVertex2d(100,230);
8         glVertex2d(170,130);
9         glVertex2d(300,350);
10    glEnd();
11    glFlush();
12
13    //Update screen
14    //glutSwapBuffers();
15 }
```

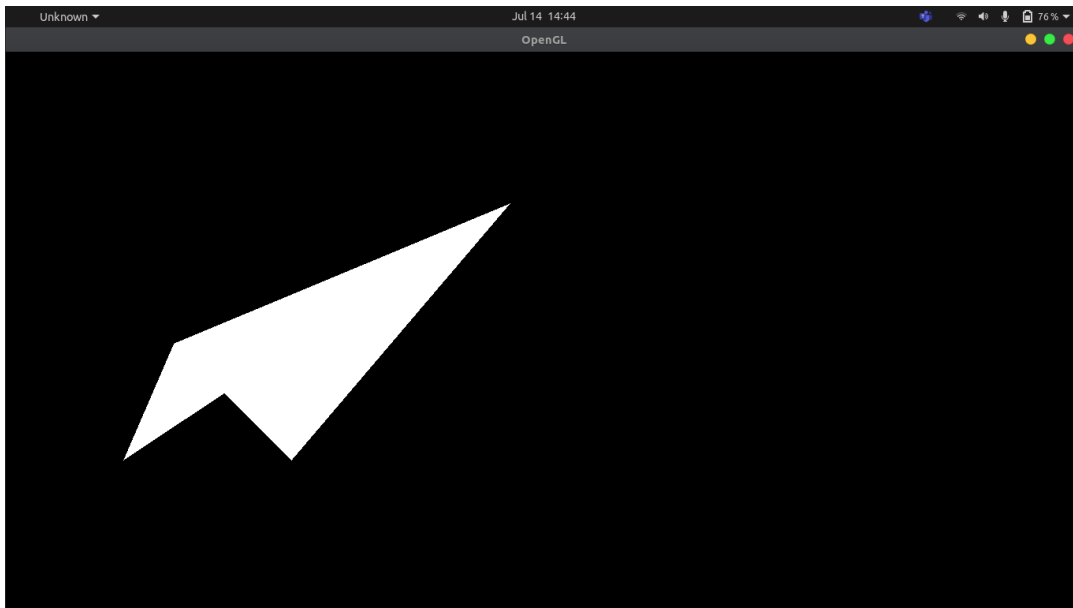
Output:



QUADS:

```
1 void render()  
2 {  
3     //Clear color buffer  
4     glClear(GL_COLOR_BUFFER_BIT);  
5     glBegin(GL_QUADS);  
6         glVertex2d(70,130);  
7         glVertex2d(100,230);  
8         glVertex2d(170,130);  
9         glVertex2d(300,350);  
10    glEnd();  
11    glFlush();  
12  
13    //Update screen  
14    //glutSwapBuffers();  
15 }
```

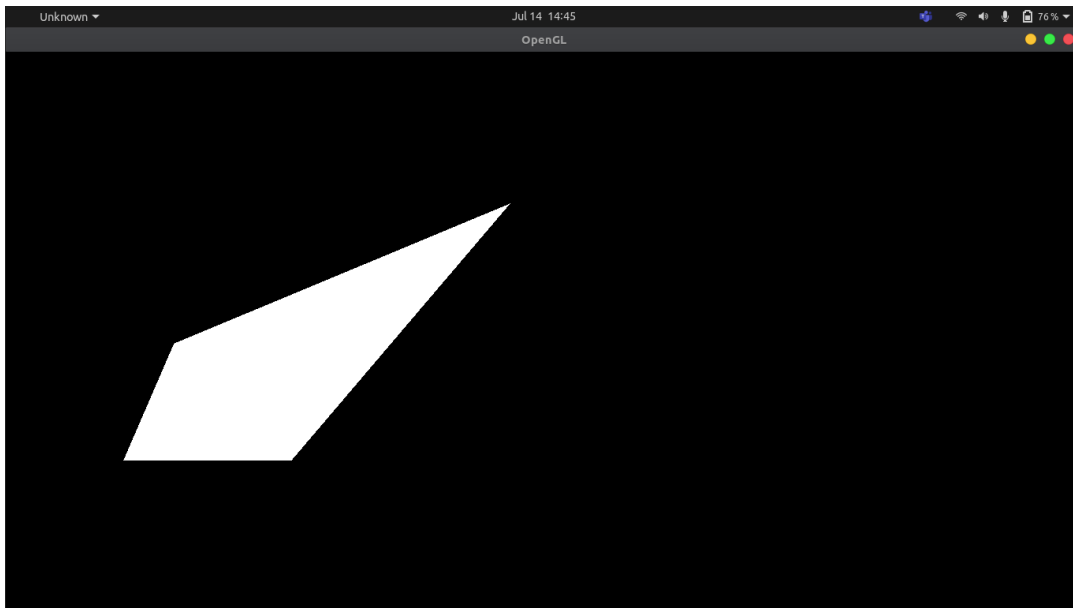
Output:



QUAD_STRIP:

```
1 void render()  
2 {  
3     //Clear color buffer  
4     glClear(GL_COLOR_BUFFER_BIT);  
5     glBegin(GL_LINE_STRIP);  
6         glVertex2d(70,130);  
7         glVertex2d(100,230);  
8         glVertex2d(170,130);  
9         glVertex2d(300,350);  
10    glEnd();  
11    glFlush();  
12  
13    //Update screen  
14    //glutSwapBuffers();  
15 }
```

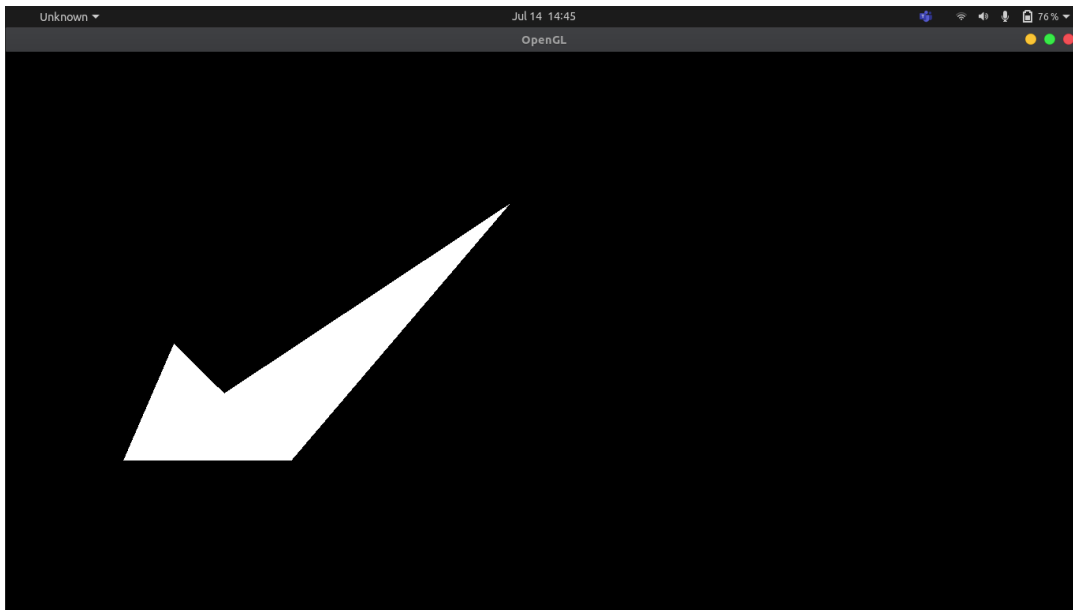
Output:



POLYGON:

```
1 void render()
2 {
3     //Clear color buffer
4     glClear(GL_COLOR_BUFFER_BIT);
5     glBegin(GL_LINE_LOOP);
6         glVertex2d(70,130);
7         glVertex2d(100,230);
8         glVertex2d(170,130);
9         glVertex2d(300,350);
10    glEnd();
11    glFlush();
12
13    //Update screen
14    //glutSwapBuffers();
15 }
```

Output:



Objective:

To create an output window and draw a checkerboard using OpenGL.

Code:

```
1
2 #ifndef LOPENGL_H
3 #define LOPENGL_H
4
5 #include <GL/freeglut.h>
6 #include <GL/gl.h>
7 #include <GL/glu.h>
8 #include <stdio.h>
9
10 #endif
1
2 #ifndef LTEXTURE_H
3 #define LTEXTURE_H
4
5 #include "LOpenGL.h"
6 #include <stdio.h>
7
8 class LTexture
9 {
10     public:
11         //Constructor
12         LTexture();
13
14         //Destructor
15         ~LTexture();
16
17         //Creates texture from pixels
18         bool loadTextureFromPixels32( GLuint* pixels, GLuint
19 width, GLuint height );
20
21         //Delete Texture
22         void freeTexture();
23
24         void render( GLfloat x, GLfloat y );
```

```

23
24         GLuint getTextureID();
25
26         GLuint textureWidth();
27
28         GLuint textureHeight();
29
30     private:
31         //Texture name
32         GLuint mTextureID;
33
34         //Texture dimensions
35         GLuint mTextureWidth;
36         GLuint mTextureHeight;
37 };
38
39 #endif

```



```

1
2 #include "LTexture.h"
3
4 LTexture::LTexture(){
5     //Initialize texture ID
6     mTextureID = 0;
7
8     //Initialize texture dimensions
9     mTextureWidth = 0;
10    mTextureHeight = 0;
11 }
12
13 LTexture::~~LTexture(){
14     //Free texture data if needed
15     freeTexture();
16 }
17
18 bool LTexture::loadTextureFromPixels32( GLuint* pixels,
19                                         GLuint width, GLuint height ){
20     //Free texture if it exists
21     freeTexture();
22
23     //Get texture dimensions
24     mTextureWidth = width;
25     mTextureHeight = height;
26
27     //Generate texture ID
28     glGenTextures( 1, &mTextureID );

```

```

28
29 //Bind texture ID
30 glBindTexture( GL_TEXTURE_2D, mTextureID );
31
32 //Generate texture
33 glTexImage2D( GL_TEXTURE_2D, 0, GL_RGBA, width, height,
34 0, GL_RGBA, GL_UNSIGNED_BYTE, pixels );
35
36 //Set texture parameters
37 glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
38 GL_LINEAR );
39 glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
40 GL_LINEAR );
41
42 //Unbind texture
43 glBindTexture( GL_TEXTURE_2D, NULL );
44
45 //Check for error
46 GLenum error = glGetError();
47 if( error != GL_NO_ERROR ){
48     printf( "Error loading texture from %p pixels! %s\n",
49     pixels, gluErrorString( error ) );
50     return false;
51 }
52
53 return true;
54 }
55
56 void LTexture::freeTexture(){
57     //Delete texture
58     if( mTextureID != 0 ){
59         glDeleteTextures( 1, &mTextureID );
60         mTextureID = 0;
61     }
62
63     mTextureWidth = 0;
64     mTextureHeight = 0;
65 }
66
67 void LTexture::render( GLfloat x, GLfloat y ){
68     //If the texture exists
69     if( mTextureID != 0 ){
70         //Remove any previous transformations
71         glLoadIdentity();

```

```

69         //Move to rendering point
70         glTranslatef( x, y, 0.f );
71
72         //Set texture ID
73         glBindTexture( GL_TEXTURE_2D, mTextureID );
74
75         //Render textured quad
76         glBegin( GL_QUADS );
77             glTexCoord2f( 0.f, 0.f ); glVertex2f(
0.f,                0.f );
78             glTexCoord2f( 1.f, 0.f ); glVertex2f(
mTextureWidth,      0.f );
79             glTexCoord2f( 1.f, 1.f ); glVertex2f(
mTextureWidth, mTextureHeight );
80             glTexCoord2f( 0.f, 1.f ); glVertex2f(
0.f, mTextureHeight );
81         glEnd();
82     }
83 }
84
85 GLuint LTexture::getTextureID(){
86     return mTextureID;
87 }
88
89 GLuint LTexture::textureWidth(){
90     return mTextureWidth;
91 }
92
93 GLuint LTexture::textureHeight(){
94     return mTextureHeight;
95 }

```



```

1
2 #ifndef LUTIL_H
3 #define LUTIL_H
4
5 #include "LOpenGL.h"
6 #include <stdio.h>
7
8 //Screen Constants
9 const int SCREEN_WIDTH = 640;
10 const int SCREEN_HEIGHT = 480;
11 const int SCREEN_FPS = 60;
12
13 bool initGL();
14

```



```

15 bool loadMedia();
16
17 void update();
18
19 void render();
20
21 #endif

1
2 #include "LUtil.h"
3 #include "LTexture.h"
4
5 //Checkerboard texture
6 LTexture gCheckerBoardTexture;
7
8 bool initGL(){
9     //Initialize Projection Matrix
10    glViewport(0.f, 0.f, SCREEN_WIDTH, SCREEN_HEIGHT);
11    glMatrixMode( GL_PROJECTION );
12    glLoadIdentity();
13    gluOrtho2D(0.0,640.0,0.0,480.0);
14
15    //Initialize Modelview Matrix
16    glMatrixMode( GL_MODELVIEW );
17    glLoadIdentity();
18
19    //Initialize clear color
20    glClearColor( 0.f, 0.f, 0.f, 1.f );
21
22    //Enable texturing
23    glEnable( GL_TEXTURE_2D );
24
25    //Check for error
26    GLenum error = glGetError();
27    if( error != GL_NO_ERROR )
28    {
29        printf( "Error initializing OpenGL! %s\n",
30            gluErrorString( error ) );
31        return false;
32    }
33    return true;
34 }
35
36 bool loadMedia(){
37     //Checkerboard pixels

```

```

38     const int CHECKERBOARD_WIDTH = 128;
39     const int CHECKERBOARD_HEIGHT = 128;
40     const int CHECKERBOARD_PIXEL_COUNT = CHECKERBOARD_WIDTH *
        CHECKERBOARD_HEIGHT;
41     GLuint checkerBoard[ CHECKERBOARD_PIXEL_COUNT ];
42
43     //Go through pixels
44     for( int i = 0; i < CHECKERBOARD_PIXEL_COUNT; ++i )
45     {
46         //Get the individual color components
47         GLubyte* colors = (GLubyte*)&checkerBoard[ i ];
48
49         //If the 5th bit of the x and y offsets of the pixel
do not match
50         if( i / 128 & 16 ^ i % 128 & 16 )
51         {
52             //Set pixel to white
53             colors[ 0 ] = 0xFF;
54             colors[ 1 ] = 0xFF;
55             colors[ 2 ] = 0xFF;
56             colors[ 3 ] = 0xFF;
57         }
58         else
59         {
60             //Set pixel to red
61             colors[ 0 ] = 0x00;
62             colors[ 1 ] = 0x00;
63             colors[ 2 ] = 0x00;
64             colors[ 3 ] = 0xFF;
65         }
66     }
67
68     //Load texture
69     if( !gCheckerBoardTexture.loadTextureFromPixels32(
checkerBoard, CHECKERBOARD_WIDTH, CHECKERBOARD_HEIGHT ) )
70     {
71         printf( "Unable to load checkerboard texture!\n" );
72         return false;
73     }
74
75     return true;
76 }
77
78 void update(){
79

```

```

80 }
81
82 void render(){
83     //Clear color buffer
84     glClear(GL_COLOR_BUFFER_BIT);
85
86     //Calculate centered offsets
87     GLfloat x = ( SCREEN_WIDTH - gCheckerBoardTexture.
textureWidth() ) / 2.f;
88     GLfloat y = ( SCREEN_HEIGHT - gCheckerBoardTexture.
textureHeight() ) / 2.f;
89
90     //Render checkerboard texture
91     gCheckerBoardTexture.render( x, y );
92
93     //Update screen
94     glutSwapBuffers();
95 }

1
2 #include "LUtil.h"
3
4 void runMainLoop( int val );
5
6
7 int main( int argc, char* args[] )
8 {
9     //Initialize FreeGLUT
10    glutInit( &argc, args );
11
12    //Create OpenGL 2.1 context
13    glutInitContextVersion( 2, 1 );
14
15    //Create Singlelele Buffered Window
16    glutInitDisplayMode( GLUT_DOUBLE );
17    glutInitWindowSize( SCREEN_WIDTH, SCREEN_HEIGHT );
18    glutCreateWindow( "OpenGL" );
19
20    //Do post window/context creation initialization
21    if( !initGL() ){
22        printf( "Unable to initialize graphics library!\n" );
23        return 1;
24    }
25
26    if(!loadMedia()){
27        printf("Unable to load media\n");

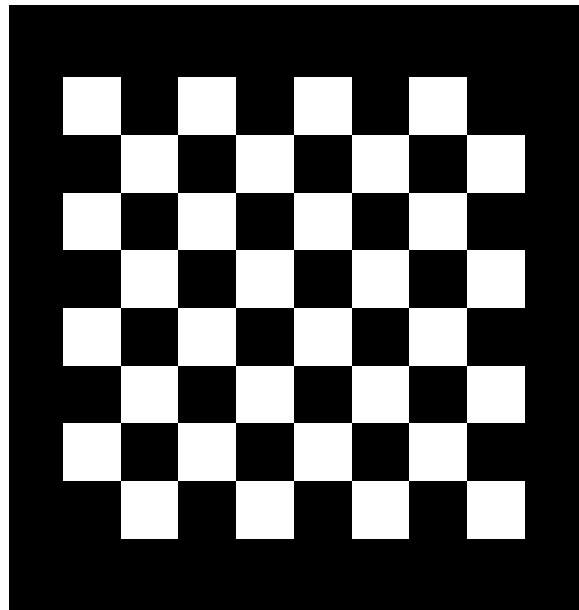
```

```

28         return 2;
29     }
30
31     //Set rendering function
32     glutDisplayFunc( render );
33
34     //Set main loop
35     glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, 0 );
36
37     //Start GLUT main loop
38     glutMainLoop();
39
40     return 0;
41 }
42
43 void runMainLoop( int val )
44 {
45     //Frame logic
46     update();
47     render();
48
49     //Run frame one more time
50     glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, val );
51 }

```

Output:



Objective:

To create an output window and draw a house using POINTS,LINES,TRAINGLS and QUADS/POLYGON.

Code:

```
1
2 #ifndef LOPENGL_H
3 #define LOPENGL_H
4
5 #include <GL/freeglut.h>
6 #include <GL/gl.h>
7 #include <GL/glu.h>
8 #include <stdio.h>
9
10 #endif
11
12
13 #ifndef LUTIL_H
14 #define LUTIL_H
15
16 #include "LOpenGL.h"
17 #include <stdio.h>
18
19 //Screen Constants
20 const int SCREEN_WIDTH = 640;
21 const int SCREEN_HEIGHT = 480;
22 const int SCREEN_FPS = 60;
23
24 bool initGL();
25
26 void update();
27
28 void render();
29
30 void building();
31
32 void roof();
33
```

```

23 void door();
24
25 void window();
26
27 void chimney();
28
29 void line(int x1, int y1, int x2, int y2);
30
31 void lineloop(int x1, int y1, int x2, int y2);
32
33 void triangle(int x1, int y1, int x2, int y2, int x3, int y3)
    ;
34
35 void quad(int x1, int y1, int x2, int y2);
36
37
38 #endif

1
2 #include "LUtil.h"
3
4 bool initGL()
5 {
6     //Initialize Projection Matrix
7     glMatrixMode( GL_PROJECTION );
8     glLoadIdentity();
9     gluOrtho2D(0.0,640.0,0.0,480.0);
10
11     //Initialize Modelview Matrix
12     glMatrixMode( GL_MODELVIEW );
13     glLoadIdentity();
14
15     //Initialize clear color
16     glClearColor( 0.f, 0.f, 0.f, 1.f );
17
18     //Check for error
19     GLenum error = glGetError();
20     if( error != GL_NO_ERROR )
21     {
22         printf( "Error initializing OpenGL! %s\n",
gluErrorString( error ) );
23         return false;
24     }
25
26     return true;
27 }

```

```

28
29 void update()
30 {
31
32 }
33
34 void render()
35 {
36     //Clear color buffer
37     glClear(GL_COLOR_BUFFER_BIT);
38
39     building();
40     roof();
41     door();
42     window();
43     chimney();
44
45     glFlush();
46
47     //Update screen
48     //glutSwapBuffers();
49 }
50
51
52 void line(int x1, int y1, int x2, int y2) {
53
54     glBegin(GL_LINES);
55
56     glVertex2d(x1,y1);
57     glVertex2d(x2,y2);
58
59     glEnd();
60 }
61
62 void lineloop(int x1, int y1, int x2, int y2) {
63
64     glBegin(GL_LINE_LOOP);
65
66     glVertex2d(x1,y1);
67     glVertex2d(x2,y1);
68     glVertex2d(x2,y2);
69     glVertex2d(x1,y2);
70
71     glEnd();
72 }

```



```

73
74 void triangle(int x1, int y1, int x2, int y2, int x3, int y3)
    {
75
76     glBegin(GL_TRIANGLES);
77
78     glVertex2d(x1,y1);
79     glVertex2d(x2,y2);
80     glVertex2d(x3,y3);
81
82     glEnd();
83 }
84
85 void quad(int x1, int y1, int x2, int y2) {
86
87     glBegin(GL_QUADS);
88
89     glVertex2d(x1,y1);
90     glVertex2d(x2,y1);
91     glVertex2d(x2,y2);
92     glVertex2d(x1,y2);
93
94     glEnd();
95 }
96
97 void building() {
98     lineloop(250,100, 330,250);
99
100    lineloop(330,100, 530,250);
101 }
102
103 void roof() {
104     triangle(250,250, 330,250, 290,300);
105     line(290,300, 490,300);
106     line(490,300, 530,250);
107
108 }
109
110 void door(){
111     quad(280,100, 283,160);
112     quad(280,158, 300,160);
113     quad(297,100, 300,160);
114 }
115
116 void window(){

```

```

117     quad(418,155, 420,167);
118     quad(418,165, 442,167);
119     quad(418,153, 442,155);
120     quad(440,155, 442,167);
121     line(430,155, 430,165);
122     line(420,160, 440,160);
123 }
124
125 void chimney(){
126     quad(500,260, 515,310);
127 }

1
2 #include "LUtil.h"
3
4 void runMainLoop( int val );
5
6
7 int main( int argc, char* args[] )
8 {
9     //Initialize FreeGLUT
10    glutInit( &argc, args );
11
12    //Create OpenGL 2.1 context
13    glutInitContextVersion( 2, 1 );
14
15    //Create Singlele Buffered Window
16    glutInitDisplayMode( GLUT_SINGLE|GLUT_RGB );
17    glutInitWindowSize( SCREEN_WIDTH, SCREEN_HEIGHT );
18    glutCreateWindow( "OpenGL" );
19
20    //Do post window/context creation initialization
21    if( !initGL() )
22    {
23        printf( "Unable to initialize graphics library!\n" );
24        return 1;
25    }
26
27    //Set rendering function
28    glutDisplayFunc( render );
29
30    //Set main loop
31    glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, 0 );
32
33    //Start GLUT main loop
34    glutMainLoop();

```

```

35
36     return 0;
37 }
38
39 void runMainLoop( int val )
40 {
41     //Frame logic
42     update();
43     render();
44
45     //Run frame one more time
46     glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, val );
47 }

```

Output:

