Department of Computer Science and Engineering

Shivanirudh S G, 185001146, Semester VII

15 July 2021

UCS1712 - Graphics and Multimedia Lab

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

Aim:

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

```
#ifndef LOPENGL_H
#define LOPENGL_H

#include <GL/freeglut.h>
```

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <stdio.h>

#endif
```

```
#ifndef LUTIL_H
#define LUTIL_H

#include "Headers.h"
#include <stdio.h>

//Screen Constants
const int SCREEN_WIDTH = 640;
const int SCREEN_HEIGHT = 480;
const int SCREEN_FPS = 60;

bool initGL();

void update();

void render();

#endif
```

```
#include "Helpers.h"

void runMainLoop( int val );

int main( int argc, char* args[] )
{
    //Initialize FreeGLUT
    glutInit( &argc, args );

    //Create OpenGL 2.1 context
    glutInitContextVersion( 2, 1 );

    //Create Singlele Buffered Window
    glutInitDisplayMode( GLUT_SINGLE|GLUT_RGB );
    glutInitWindowSize( SCREEN_WIDTH, SCREEN_HEIGHT );
    glutCreateWindow( "OpenGL" );

    //Do post window/context creation initialization
    if( !initGL() )
    {
        printf( "Unable to initialize graphics library!\n" );
```

```
22          return 1;
23      }
24
25      //Set rendering function
26      glutDisplayFunc( render );
27
28      //Set main loop
29      glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, 0 );
30
31      //Start GLUT main loop
32      glutMainLoop();
33
34      return 0;
35  }
36
37  void runMainLoop( int val )
38  {
39      //Frame logic
40      update();
41      render();
42
43      //Run frame one more time
44      glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, val );
45  }
```

POINTS:

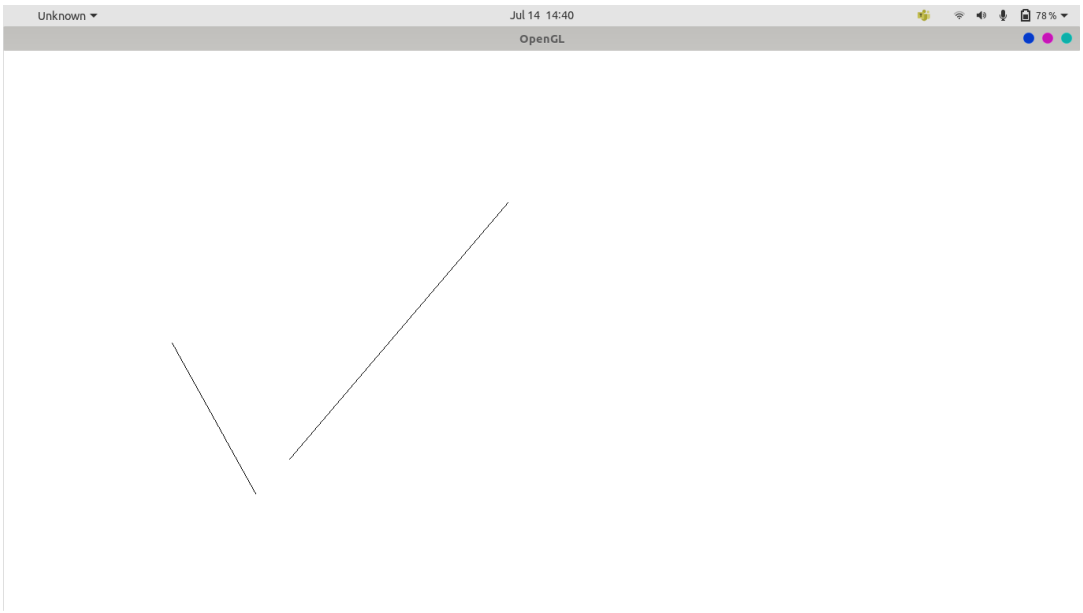
```
    glBegin(GL_POINTS);
        glVertex2d(70,130);
        glVertex2d(100,230);
        glVertex2d(170,130);
        glVertex2d(300,350);
    glEnd();
    glFlush();

  //Update screen
  //glutSwapBuffers();
}
```

Output:

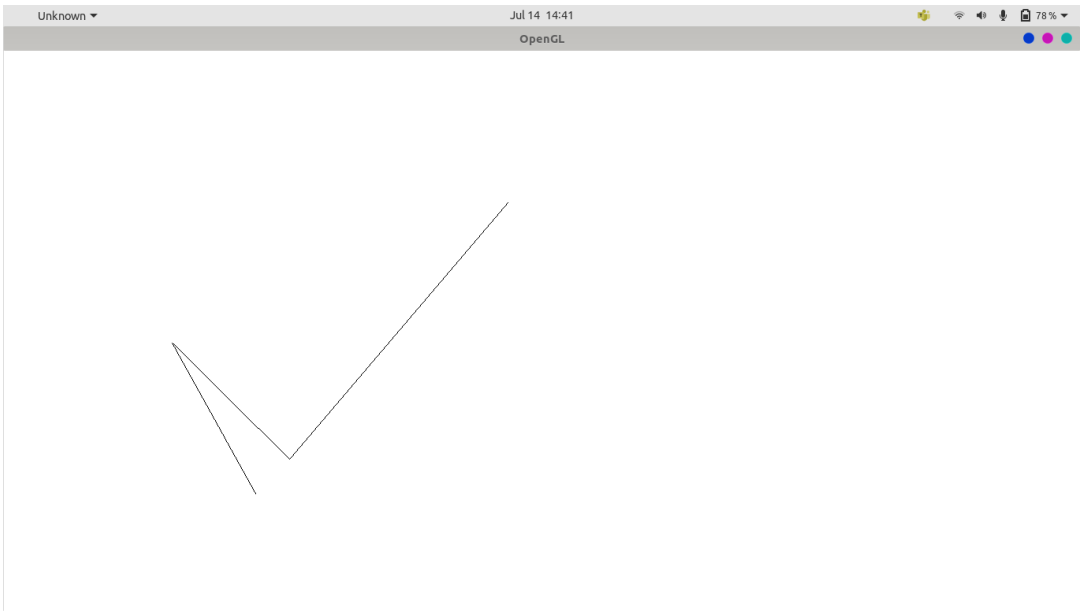
LINES:

Output:



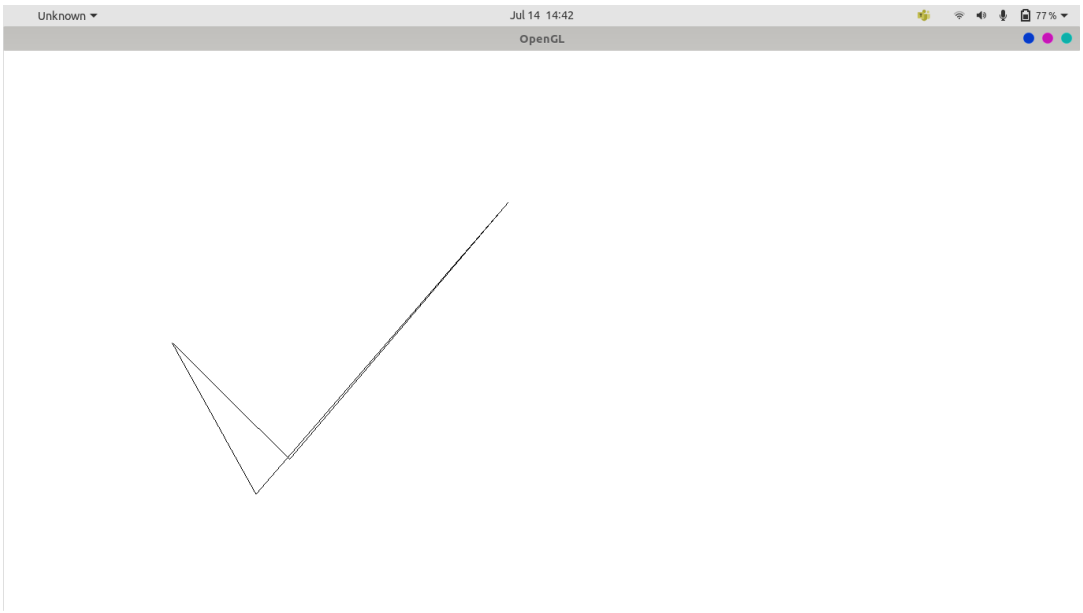
LINE_STRIP:

Output:



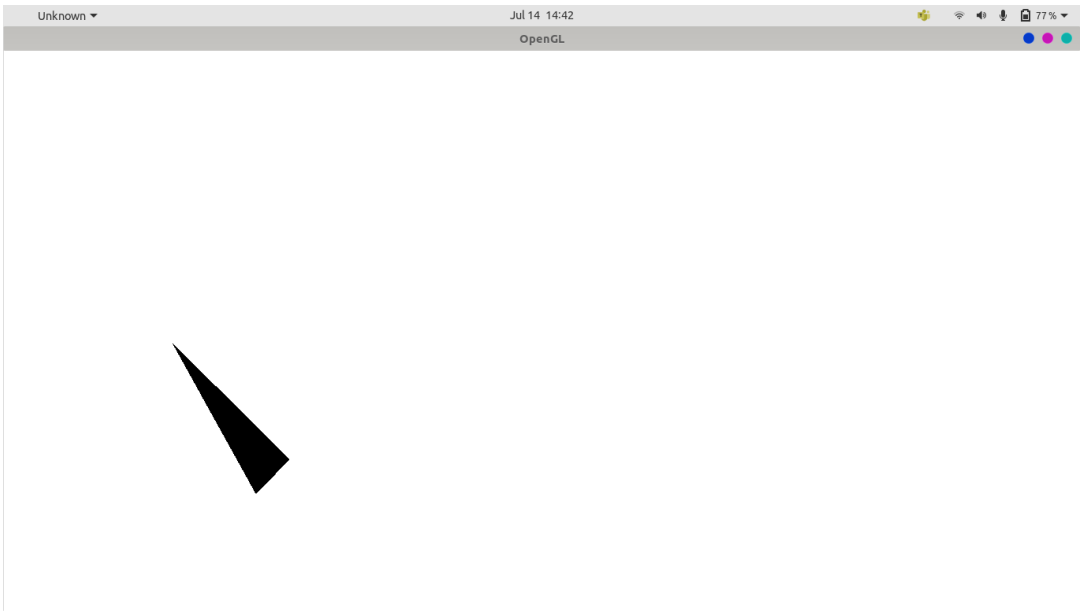
LINE_LOOP:

Output:



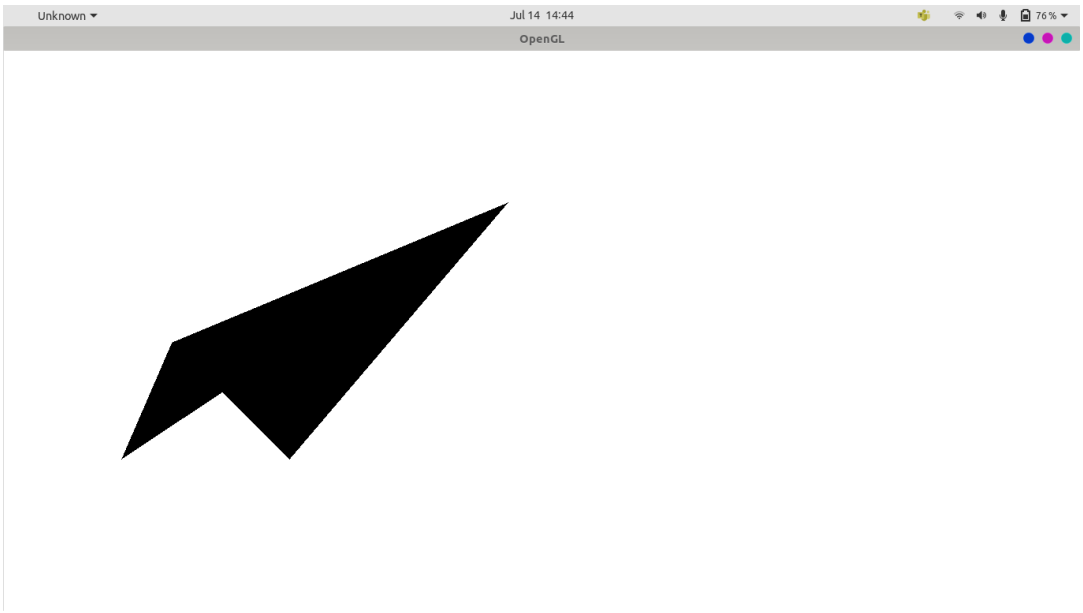
TRIANGLES:

Output:



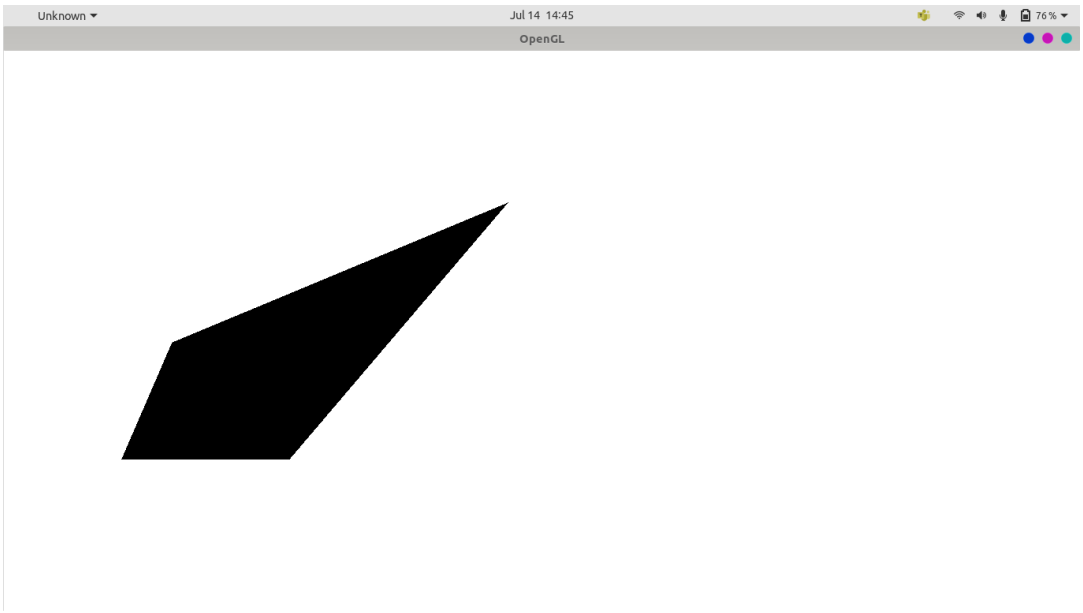
QUADS:

Output:



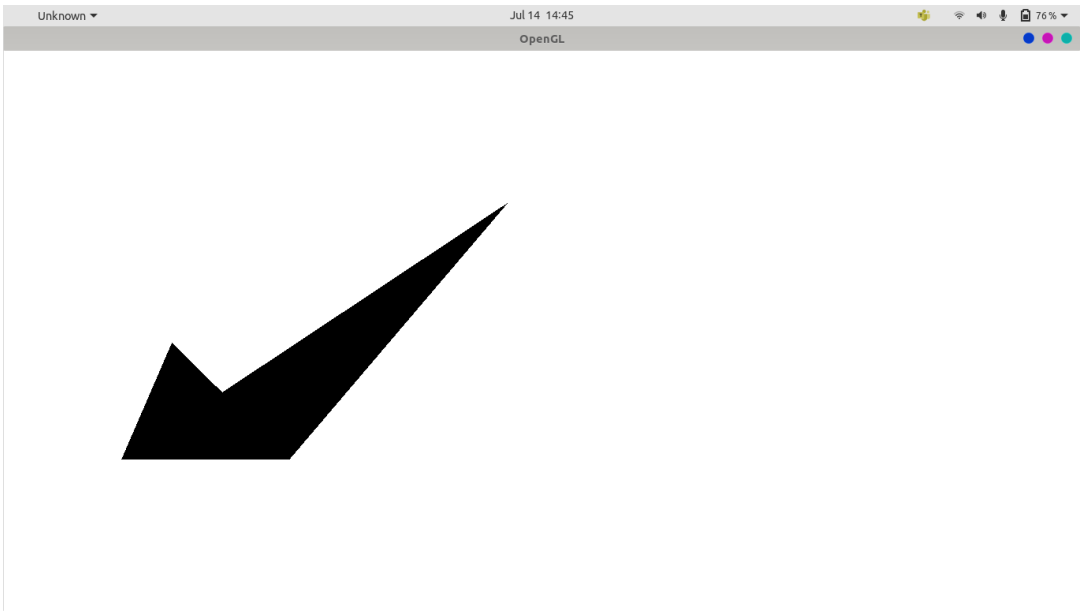
QUAD_STRIP:

Output:



POLYGON:

Output:



Aim:

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

Code:

```
#ifndef LOPENGL_H
#define LOPENGL_H

#include <GL/freeglut.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <stdio.h>

#endif
```

```
#ifndef LTEXTURE_H
#define LTEXTURE_H

#include "Headers.h"
#include<stdio.h>

class LTexture
{
    public:
        //Constructor
        LTexture();

        //Destructor
        ~LTexture();

        //Creates texture from pixels
        bool loadTextureFromPixels32( GLuint* pixels, GLuint
    width, GLuint height );

        //Delete Texture
        void freeTexture();

        void render( GLfloat x, GLfloat y );

```

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

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

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

```
1  #ifndef LUTIL_H
2  #define LUTIL_H
3
4  #include "Headers.h"
5  #include <stdio.h>
6
7  //Screen Constants
8  const int SCREEN_WIDTH = 640;
9  const int SCREEN_HEIGHT = 480;
10 const int SCREEN_FPS = 60;
11
12 bool initGL();
13
14 bool loadMedia();
15
16 void update();
```

```

void render();

#endif
```

```
#include "Signatures.h"
#include "Texture.h"

//Checkerboard texture
LTexture gCheckerBoardTexture;

bool initGL(){
    //Initialize Projection Matrix
    glViewport(0.f, 0.f, SCREEN_WIDTH, SCREEN_HEIGHT);
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluOrtho2D(0.0,640.0,0.0,480.0);

    //Initialize Modelview Matrix
    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();

    //Initialize clear color
    glClearColor( 0.f, 0.f, 0.f, 1.f );

    //Enable texturing
    glEnable( GL_TEXTURE_2D );

    //Check for error
    GLenum error = glGetError();
    if( error != GL_NO_ERROR )
    {
        printf( "Error initializing OpenGL! %s\n",
    gluErrorString( error ) );
        return false;
    }

    return true;
}

bool loadMedia(){
    //Checkerboard pixels
    const int CHECKERBOARD_WIDTH = 128;
    const int CHECKERBOARD_HEIGHT = 128;
    const int CHECKERBOARD_PIXEL_COUNT = CHECKERBOARD_WIDTH *
    CHECKERBOARD_HEIGHT;
```

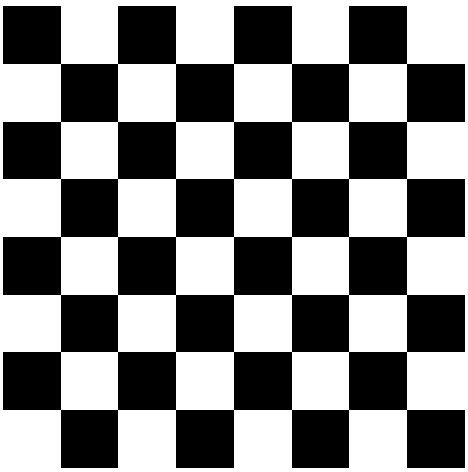
```
40      GLuint checkerBoard[ CHECKERBOARD_PIXEL_COUNT ];
41
42      //Go through pixels
43      for( int i = 0; i < CHECKERBOARD_PIXEL_COUNT; ++i )
44      {
45          //Get the individual color components
46          GLubyte* colors = (GLubyte*)&checkerBoard[ i ];
47
48          //If the 5th bit of the x and y offsets of the pixel
    do not match
49          if( i / 128 & 16 ^ i % 128 & 16 )
50          {
51              //Set pixel to white
52              colors[ 0 ] = 0xFF;
53              colors[ 1 ] = 0xFF;
54              colors[ 2 ] = 0xFF;
55              colors[ 3 ] = 0xFF;
56          }
57          else
58          {
59              //Set pixel to red
60              colors[ 0 ] = 0x00;
61              colors[ 1 ] = 0x00;
62              colors[ 2 ] = 0x00;
63              colors[ 3 ] = 0xFF;
64          }
65      }
66
67      //Load texture
68      if( !gCheckerBoardTexture.loadTextureFromPixels32(
    checkerBoard, CHECKERBOARD_WIDTH, CHECKERBOARD_HEIGHT ) )
69      {
70          printf( "Unable to load checkerboard texture!\n" );
71          return false;
72      }
73
74      return true;
75  }
76
77  void update(){
78
79  }
80
81  void render(){
82      //Clear color buffer
```

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

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

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

Output:



Aim:

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

Code:

```
#ifndef LOPENGL_H
#define LOPENGL_H

#include <GL/freeglut.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <stdio.h>

#endif
```

```
#include "Signatures.h"

bool initGL()
{
    //Initialize Projection Matrix
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluOrtho2D(0.0,640.0,0.0,480.0);

    //Initialize Modelview Matrix
    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();

    //Initialize clear color
    glClearColor( 0.f, 0.f, 0.f, 1.f );

    //Check for error
    GLenum error = glGetError();
    if( error != GL_NO_ERROR )
    {
        printf( "Error initializing OpenGL! %s\n",
    gluErrorString( error ) );
        return false;
```

```
23      }
24
25      return true;
26  }
27
28  void update()
29  {
30
31  }
32
33  void render()
34  {
35      //Clear color buffer
36      glClear(GL_COLOR_BUFFER_BIT);
37
38      building();
39      roof();
40      door();
41      window();
42      chimney();
43
44      glFlush();
45
46      //Update screen
47      //glutSwapBuffers();
48  }
49
50
51  void line(int x1, int y1, int x2, int y2) {
52
53      glBegin(GL_LINES);
54
55      glVertex2d(x1,y1);
56      glVertex2d(x2,y2);
57
58      glEnd();
59  }
60
61  void lineloop(int x1, int y1, int x2, int y2) {
62
63      glBegin(GL_LINE_LOOP);
64
65      glVertex2d(x1,y1);
66      glVertex2d(x2,y1);
67      glVertex2d(x2,y2);
```

```
68      glVertex2d(x1,y2);
69
70      glEnd();
71  }
72
73  void triangle(int x1, int y1, int x2, int y2, int x3, int y3)
        {
74
75      glBegin(GL_TRIANGLES);
76
77      glVertex2d(x1,y1);
78      glVertex2d(x2,y2);
79      glVertex2d(x3,y3);
80
81      glEnd();
82  }
83
84  void quad(int x1, int y1, int x2, int y2) {
85
86      glBegin(GL_QUADS);
87
88      glVertex2d(x1,y1);
89      glVertex2d(x2,y1);
90      glVertex2d(x2,y2);
91      glVertex2d(x1,y2);
92
93      glEnd();
94  }
95
96  void building() {
97      lineloop(250,100, 330,250);
98
99      lineloop(330,100, 530,250);
100 }
101
102 void roof() {
103     triangle(250,250, 330,250, 290,300);
104     line(290,300, 490,300);
105     line(490,300, 530,250);
106
107 }
108
109 void door(){
110     quad(280,100, 283,160);
111     quad(280,158, 300,160);
```

```
    quad(297,100, 300,160);
}

void window(){
    quad(418,155, 420,167);
    quad(418,165, 442,167);
    quad(418,153, 442,155);
    quad(440,155, 442,167);
    line(430,155, 430,165);
    line(420,160, 440,160);
}

void chimney(){
    quad(500,260, 515,310);
}
```

```
#ifndef LUTIL_H
#define LUTIL_H

#include "Headers.h"
#include <stdio.h>

//Screen Constants
const int SCREEN_WIDTH = 640;
const int SCREEN_HEIGHT = 480;
const int SCREEN_FPS = 60;

bool initGL();

void update();

void render();

void building();

void roof();

void door();

void window();

void chimney();

void line(int x1, int y1, int x2, int y2);

void lineloop(int x1, int y1, int x2, int y2);
```

```
31
32 void triangle(int x1, int y1, int x2, int y2, int x3, int y3)
       ;
33
34 void quad(int x1, int y1, int x2, int y2);
35
36
37 #endif
```

```
1 #include "Helpers.h"
2
3 void runMainLoop( int val );
4
5
6 int main( int argc, char* args[] )
7 {
8     //Initialize FreeGLUT
9     glutInit( &argc, args );
10
11    //Create OpenGL 2.1 context
12    glutInitContextVersion( 2, 1 );
13
14    //Create Singlele Buffered Window
15    glutInitDisplayMode( GLUT_SINGLE|GLUT_RGB );
16    glutInitWindowSize( SCREEN_WIDTH, SCREEN_HEIGHT );
17    glutCreateWindow( "OpenGL" );
18
19    //Do post window/context creation initialization
20    if( !initGL() )
21    {
22        printf( "Unable to initialize graphics library!\n" );
23        return 1;
24    }
25
26    //Set rendering function
27    glutDisplayFunc( render );
28
29    //Set main loop
30    glutTimerFunc( 1000 / SCREEN_FPS, runMainLoop, 0 );
31
32    //Start GLUT main loop
33    glutMainLoop();
34
35    return 0;
36 }
37
```

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

Output: