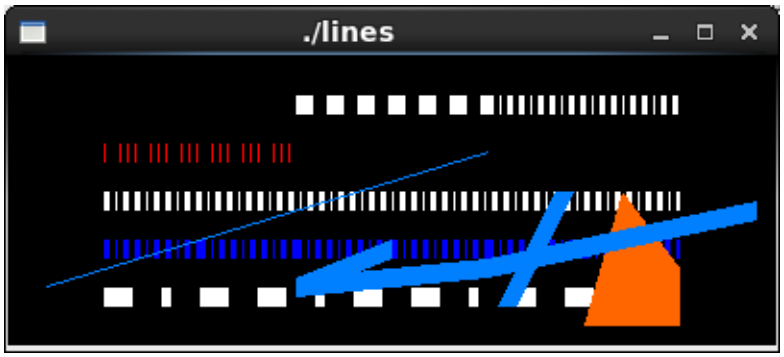
CSE420
Samuel Marrujo
Professor Yu
Lab 04

Lines

In this part of the lab, it was a modification of the original lines program. After some modifications to the program, and adding in the polygons asked for, the result became interesting. Especially the Cull face part of the lab. Furthermore, I added the Bresenham line algorithm to produce the line given. Because of this change, I believe I was able to accomplish this task successfully, since there were no errors and the display window was changed. Here are my results for this program:



```
/*
 *  lines.cpp
 *  This program demonstrates geometric primitives and
 *  their attributes.
 */
#include <GL/glut.h>
#include <stdlib.h>

#define drawOneLine(x1,y1,x2,y2)  glBegin(GL_LINES);  \
   glVertex2f ((x1),(y1)); glVertex2f ((x2),(y2)); glEnd();

void init(void)
{
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_FLAT);
}

void setPixel(GLint x,GLint y)
{
  glBegin(GL_POINTS);
     glVertex2i(x,y);
  glEnd();
}

void display(void)
{
   int i;
   int x0 = 20, y0 = 30, xn = 250, yn = 100, x, y;
   int     dx, dy,          //deltas
           pk,              //decision parameter
           k;               //looping variable

   glClear (GL_COLOR_BUFFER_BIT);

/* select white for all lines  */
   glColor3f (1.0, 1.0, 1.0);

/* in 1st row, 3 lines, each with a different stipple  */
   glEnable (GL_LINE_STIPPLE);

   glLineStipple (1, 0x01FF);
   drawOneLine (150.0, 125.0, 250.0, 125.0);
   glLineStipple (1, 0x1C47);  /*  dash/dot/dash  */
   drawOneLine (250.0, 125.0, 350.0, 125.0);

/* in 2nd row, 3 wide lines, each with different stipple */
   glColor3f (1.0, 0, 0); // Color Red
   glLineWidth (15.0);
   glLineStipple (1, 0x1101);
```

```
   drawOneLine (50.0, 100.0, 150.0, 100.0);

   glColor3f (1,1,1); //Resets color to White
/* in 3rd row, 6 lines, with dash/dot/dash stipple  */
/* as part of a single connected line strip         */
   glLineStipple (1, 0x1C47);  /*  dash/dot/dash  */
   glBegin (GL_LINE_STRIP);
   for (i = 0; i < 7; i++)
      glVertex2f (50.0 + ((GLfloat) i * 50.0), 75.0);
   glEnd ();

/* in 4th row, 6 independent lines with same stipple  */
   glColor3f(0,0,1);
   for (i = 0; i < 6; i++) {
      drawOneLine (50.0 + ((GLfloat) i * 50.0), 50.0,
         50.0 + ((GLfloat)(i+1) * 50.0), 50.0);
   }
   glColor3f(1,1,1);
/* in 5th row, 1 line, with dash/dot/dash stipple    */
/* and a stipple repeat factor of 5                  */
   glLineStipple (5, 0x1C47);  /*  dash/dot/dash  */
   drawOneLine (50.0, 25.0, 350.0, 25.0);

   glDisable (GL_LINE_STIPPLE);

   glColor3f(1,0.4,0);
   glPolygonMode( GL_FRONT, GL_FILL );
   glBegin (GL_POLYGON);
      glVertex2i(300,10);
      glVertex2i(350,10);
      glVertex2i(350,40);
      glVertex2i(320,80);
      glVertex2i(300,10);
   glEnd();
   glColor3f(0,0.5,1);
   glEnable(GL_CULL_FACE);
   glCullFace(GL_BACK);
   glPolygonMode( GL_FRONT, GL_LINE);
   glBegin(GL_POLYGON);
      glVertex2i(200,50);
      glVertex2i(150,30);
      glVertex2i(250,40);
      glVertex2i(390,70);
      glVertex2i(260,20);
      glVertex2i(290,80);
      glVertex2i(200,50);
   glEnd();
   glDisable(GL_CULL_FACE);
```

```
  setPixel(x0, y0);      //plot first point

  // difference between starting and ending points
  dx = xn - x0;
  dy = yn - y0;
  pk = 2 * dy - dx;
  x = x0;         y = y0;

  for ( k = 0; k < dx-1; ++k ) {
    if ( pk < 0 ) {
      pk = pk + 2 * dy;                          //calculate next pk
                                                 //next pixel: (x+1, y )
    } else {
                                                 //next pixel: (x+1, y+1)
      pk = pk + 2*dy - 2*dx;             //calculate next pk
      ++y;
    }
    ++x;
    setPixel( x, y );
  }
  glFlush();
}

void reshape (int w, int h)
{
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  gluOrtho2D (0.0, (GLdouble) w, 0.0, (GLdouble) h);
}

void keyboard(unsigned char key, int x, int y)
{
  switch (key) {
    case 27:
      exit(0);
      break;
  }
}

int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (400, 150);
  glutInitWindowPosition (100, 100);
  glutCreateWindow (argv[0]);
  init ();
  glutDisplayFunc(display);
```

```
  glutReshapeFunc(reshape);
  glutKeyboardFunc(keyboard);
  glutMainLoop();
  return 0;
}
```

Extra Credit Part I

Given:

- x0 = 20 xn = 250
 x = 20
dx=250-20 = 230

- y0 = 30 yn = 100
 y = 30
dy=100-30 = 70

pk = 2*70-230 = -90
=> pk = -90 + 140 = 50
x=21
then
pk = 50 + 140 – 180 = 10
y = 31
x=22
then
pk = 10 + 140 – 180 = -30
y=32
x=23
=> pk = -30 + 140 = 110
x=24
then
pk = 110 + 140 – 180 = 70
y=33
x=24
then
pk = 70 + 140-180 = 30
y=34
x=25
then
pk = 30 + 140-180 = -10
y=35
x=26
=> pk = -10 +140 = 130
x=27

Therefore, this line doesn't actually reach (33,34).

Now,
suppose the point is (33,34). That is, suppose x0 = 33, y0 = 34:
xn = 250, yn = 100
x = 33, y = 34
x0 = 33, y0 = 34

dx = 250 – 33 = 217
dy = 100 – 34 = 66

pk = 2 * 66 – 217 = -217 + 132 = -85
=> pk = -85 + 132 = 47
y = 35
then
pk = 47 + 132 – 434 = -255
x = 34
y = 36

Therefore, the next two points in this line from (33,34) are:
(33, 35)
(34, 36)