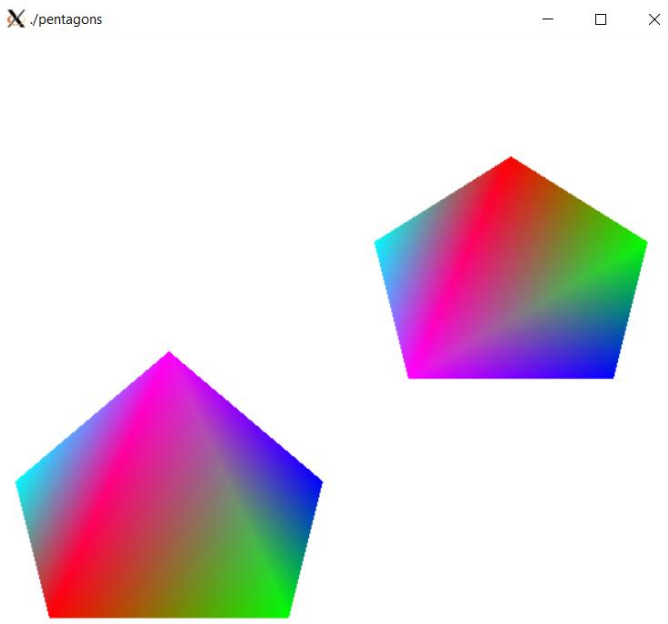
CSE 4200 Lab 6 – Spencer Wallace

Summary:

For this lab I was able to successfully draw the two pentagons using the vertex buffer object as well as draw the square and triangle with VBO and draw element. I also made it so the square and triangle would change colors on a mouse click, for both methods used. For the extra credit cube program I was able to put the cubes in their own quadrants using glViewport before the cubes were rendered, and I was able to change their colors by updating the color pointer before they were rendered. The cubes also swap colors when the mouse is clicked. Because I was able to complete all parts of the lab including the extra credit, I am giving myself **35/30** points for this lab.

Part 1: Pentagons



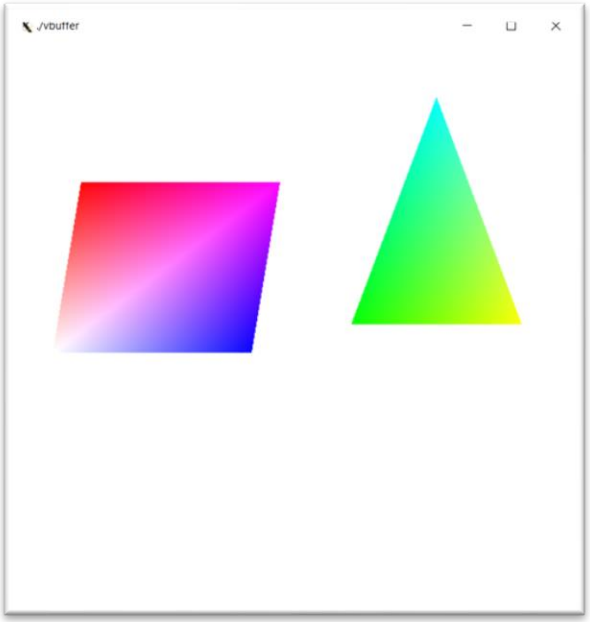
Code on next page (functions that weren't modified are not included. Ex: init, main, keyboard, etc.)

```
float data [] = { 50,108, 1,0,0,  95,70, 0,1,0, 85,30, 0,0,1, 15,30, 1,1,0, 5,70, 1,0,1,
                  120,100, 0,0,1,  180,100, 0,1,0, 190,140, 1,0,0, 150,165, 0,1,1, 110,140, 0,1,0};
bool ColorFlag = false;
unsigned int indices []  = { 0, 1, 2, 3, 4 };
unsigned int vbo;    //vertex buffer object
unsigned int ind;

void display( void )
{
  glClear( GL_COLOR_BUFFER_BIT );      //clear screen
  glEnableClientState ( GL_VERTEX_ARRAY );
  glEnableClientState ( GL_COLOR_ARRAY );
  glVertexPointer ( 2, GL_FLOAT, 5*sizeof(float), 0 );
  (ColorFlag)
  ?    glColorPointer ( 3, GL_FLOAT, 5*sizeof(float), (void*) (2*sizeof(float)) )
  :    glColorPointer ( 3, GL_FLOAT, 5*sizeof(float), (void*) (22*sizeof(float)) );
//glColorPointer ( 3, GL_FLOAT, 5*sizeof(float), (void*) (2*sizeof(float)) );
  glDrawElements ( GL_POLYGON, 5, GL_UNSIGNED_INT, 0 );  // last param=0  =>video card*/
  glVertexPointer ( 2, GL_FLOAT, 5*sizeof(float), (void*) (25*sizeof(float)) );
  glDrawElements ( GL_POLYGON, 5, GL_UNSIGNED_INT, 0 );  // last param  =>video card*/

  glDisableClientState ( GL_VERTEX_ARRAY );
  glDisableClientState ( GL_COLOR_ARRAY );

  glFlush();                        //send all output to screen
}

void myMouse( int button, int state, int x, int y )
{
```

```
 if ( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN )
      {
            ColorFlag = !ColorFlag;
            glutPostRedisplay();
      }
 glFlush();                    //send all output to screen
}
```

Part 2: Square and triangle with color change on click

For method 1 I used glDrawElements and for method 2 I used glArrayElement as instructed in the lab doc, the output for both methods is the same, thus only the two photos are shown to show the color change.



Code for Method 1

```
float data [] = { 15,90, 1,0,0,  85,90, 0,1,0, 95,150, 0,0,1, 25,150, 1,1,0,
                  120,100, 0,0,1,  180,100, 0,1,0, 150,180, 1,0,0};
bool ColorFlag = false;
unsigned int indices []  = { 0, 1, 2, 3 };
unsigned int vbo;    //vertex buffer object
unsigned int ind;


void display( void )
{
  glClear( GL_COLOR_BUFFER_BIT );      //clear screen
  glEnableClientState ( GL_VERTEX_ARRAY );
  glEnableClientState ( GL_COLOR_ARRAY );
```

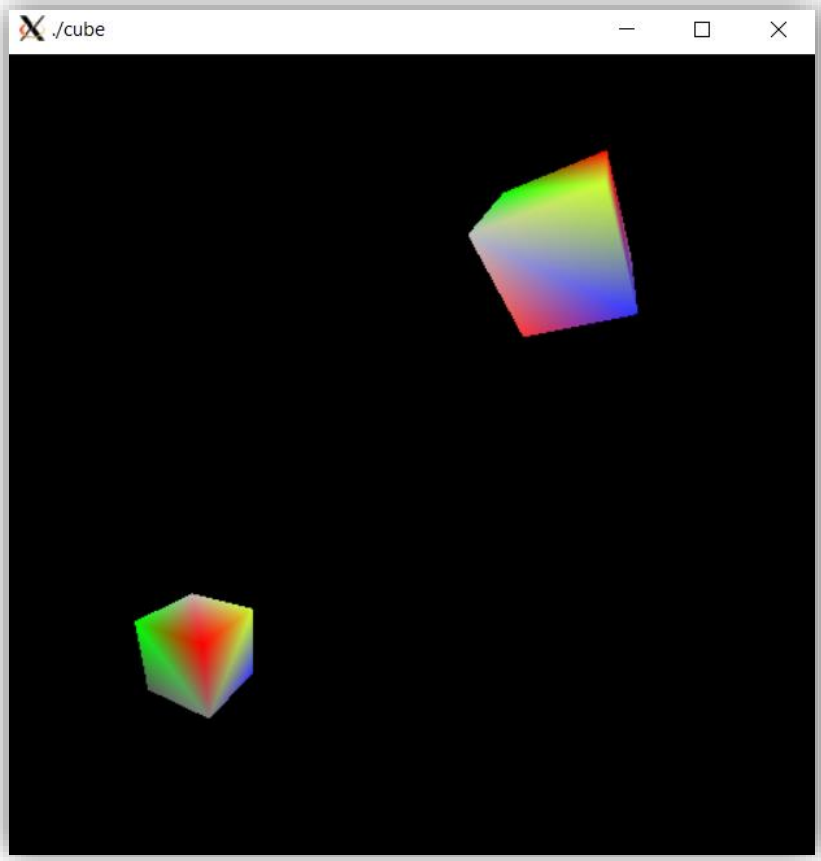
```
  glVertexPointer ( 2, GL_FLOAT, 5*sizeof(float), 0 );

 (ColorFlag)
 ?    glColorPointer ( 3, GL_FLOAT, 5*sizeof(float), (void*) (2*sizeof(float)) )
 :    glColorPointer ( 3, GL_FLOAT, 5*sizeof(float), (void*) (22*sizeof(float)) );

 glDrawElements ( GL_POLYGON, 4, GL_UNSIGNED_INT, 0 );  // last param=0  =>video card*/

 glVertexPointer ( 2, GL_FLOAT, 5*sizeof(float), (void*) (20*sizeof(float)) );
 glDrawElements ( GL_TRIANGLES, 3, GL_UNSIGNED_INT, 0 );  // last param  =>video card*/

  glDisableClientState ( GL_VERTEX_ARRAY );
  glDisableClientState ( GL_COLOR_ARRAY );
  glFlush();                        //send all output to screen
}

void myMouse( int button, int state, int x, int y )
{
  if ( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN )
      {
            ColorFlag = !ColorFlag;
            glutPostRedisplay();
      }
 glFlush();                        //send all output to screen
}
```

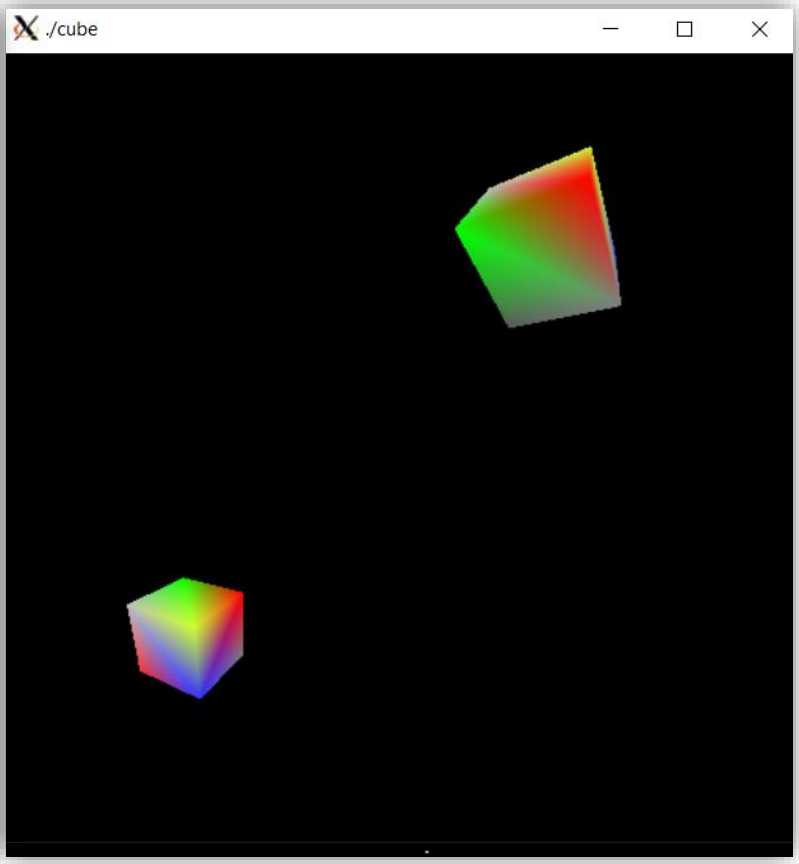
Code for Method 2

```
float data [] = { 15,90, 1,0,0,  85,90, 0,1,0, 95,150, 0,0,1, 25,150, 1,1,0,
                  120,100, 0,0,1,  180,100, 0,1,0, 150,180, 1,0,0};
bool ColorFlag = false;
unsigned int indices []  = { 0, 1, 2, 3};
unsigned int vbo;    //vertex buffer object
unsigned int ind;

void display( void )
{
  glClear( GL_COLOR_BUFFER_BIT );      //clear screen
  glEnableClientState ( GL_VERTEX_ARRAY );
  glEnableClientState ( GL_COLOR_ARRAY );

  glVertexPointer ( 2, GL_FLOAT, 5*sizeof(float), 0 );

  (ColorFlag)
  ?    glColorPointer ( 3, GL_FLOAT, 5*sizeof(float), (void*) (2*sizeof(float)) )
  :    glColorPointer ( 3, GL_FLOAT, 5*sizeof(float), (void*) (7*sizeof(float)) );

  glBegin( GL_POLYGON );
  glArrayElement(0);
  glArrayElement(1);
  glArrayElement(2);
  glArrayElement(3);
  glEnd();

  glBegin( GL_POLYGON );
  glArrayElement(4);
```

```
  glArrayElement(5);
  glArrayElement(6);
  glEnd();

  glDisableClientState ( GL_VERTEX_ARRAY );
  glDisableClientState ( GL_COLOR_ARRAY );

  glFlush();                    //send all output to screen
}
void myMouse( int button, int state, int x, int y )
{
  if ( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN )
      {
            ColorFlag = !ColorFlag;
            glutPostRedisplay();
      }
}
```

Extra Credit: Cubes





Code (only the display function was modified and a mouse function was added with a global boolean variable to use as a flag)

```
bool ColorFlag = true;


void display(void)
{
  glClear (GL_COLOR_BUFFER_BIT);
  glColor3f (1.0, 1.0, 1.0);
  glLoadIdentity ();             /* clear the matrix */
        /* viewing transformation  */
  gluLookAt (3.0, 3.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
//   glScalef (1.0, 2.0, 1.0);      /* modeling transformation */
//   glutWireCube (1.0);

static GLint vertices[] = {-1, -1, -1,   //0
                  1, -1, -1,
                  1, 1, -1,
                  -1, 1, -1,       //3
                   -1, -1, 1,
                   1, -1, 1,
                   1, 1, 1,        //6
                   -1, 1, 1};      //7

  static GLfloat colors[] = {1.0, 0.2, 0.2,
                  0.2, 0.2, 1.0,
                  0.8, 1.0, 0.2,
                  0.75, 0.75, 0.75,
                  0.35, 0.35, 0.35,
                  0.5, 0.5, 0.5,
```

```
                1.0, 0.0, 0.0,
                0.0, 1.0, 0.0
                };
    static GLfloat colors2[] = {
            0.35, 0.35, 0.35,
            0.5, 0.5, 0.5,
            1.0, 0.0, 0.0,
            0.0, 1.0, 0.0,
            1.0, 0.2, 0.2,
            0.2, 0.2, 1.0,
            0.8, 1.0, 0.2,
            0.75, 0.75, 0.75
     };

  glVertexPointer (3, GL_INT, 0, vertices);
  (ColorFlag)
  ? glColorPointer (3, GL_FLOAT, 0, colors)
  : glColorPointer (3, GL_FLOAT, 0, colors2);

  glEnable( GL_CULL_FACE );
  glCullFace ( GL_BACK );
  static GLubyte frontIndices[] = {4, 5, 6, 7};
  static GLubyte rightIndices[] = {1, 2, 6, 5};
  static GLubyte bottomIndices[] = {0, 1, 5, 4};
  static GLubyte backIndices[] = {0, 3, 2, 1};
  static GLubyte leftIndices[] = {0, 4, 7, 3};
  static GLubyte topIndices[] = {2, 3, 7, 6};

  glViewport(25,25,250,250);
```

```
  glTranslatef ( -1.5, -1.2, 0 );
  glDrawElements(GL_QUADS, 4, GL_UNSIGNED_BYTE, frontIndices);
  glDrawElements(GL_QUADS, 4, GL_UNSIGNED_BYTE, rightIndices);
  glDrawElements(GL_QUADS, 4, GL_UNSIGNED_BYTE, bottomIndices);
  glDrawElements(GL_QUADS, 4, GL_UNSIGNED_BYTE, backIndices);
  glDrawElements(GL_QUADS, 4, GL_UNSIGNED_BYTE, leftIndices);
  glDrawElements(GL_QUADS, 4, GL_UNSIGNED_BYTE, topIndices);

  glRotatef ( 30, 1, 1, 1 );
  glTranslatef ( 3, 2.4, 0 );
  static GLubyte allIndices[] = {4, 5, 6, 7, 1, 2, 6, 5,
     0, 1, 5, 4, 0, 3, 2, 1,
     0, 4, 7, 3, 2, 3, 7, 6};
  glViewport(200,200,250,250);
  (ColorFlag)
  ? glColorPointer (3, GL_FLOAT, 0, colors2)
  : glColorPointer (3, GL_FLOAT, 0, colors);

  glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, allIndices);
/*
  glBegin( GL_QUADS );
    glArrayElement ( 4 );    //front face (see notes)
    glArrayElement ( 5 );
    glArrayElement ( 6 );
    glArrayElement ( 7 );

    glArrayElement ( 0 );   //back face
    glArrayElement ( 3 );
    glArrayElement ( 2 );
```

```
    glArrayElement ( 1 );

    glArrayElement ( 1 );   //back facing (right)
    glArrayElement ( 2 );
    glArrayElement ( 6 );
    glArrayElement ( 5 );

  glEnd();
  */
  glFlush ();
}

void myMouse( int button, int state, int x, int y )
{
  if ( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN )
      {
            ColorFlag = !ColorFlag;
            glutPostRedisplay();
      }

}
```