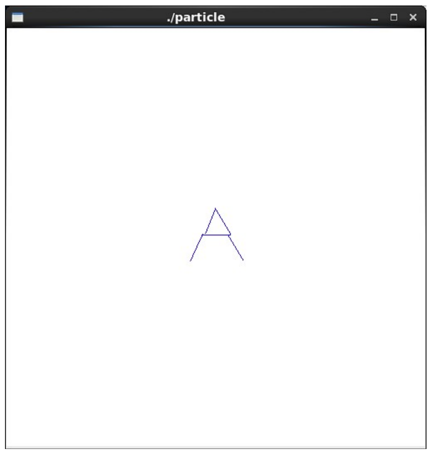
CSE520
Samuel Marrujo
Professor Yu
Homework 2

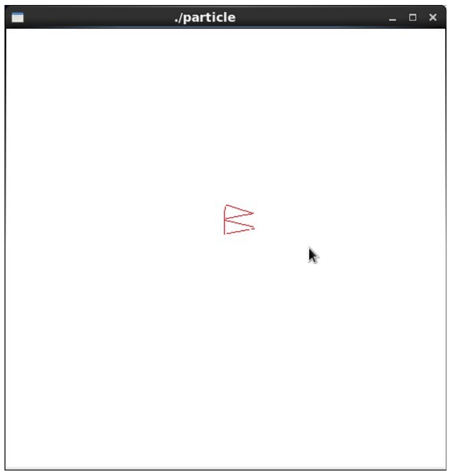
Homework 2

In this homework, we are to create multiple shader programs using openGL ES. I have successfully completed this task to the best of my ability for the time presented, and here are pictures of the following:

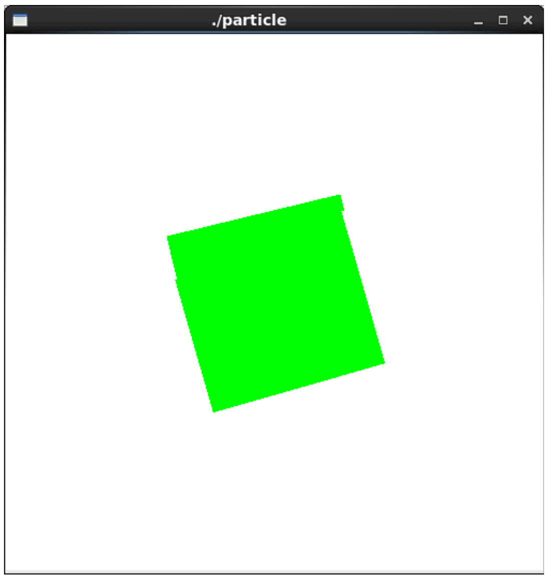
Each picture will be on a separate page as shown below.

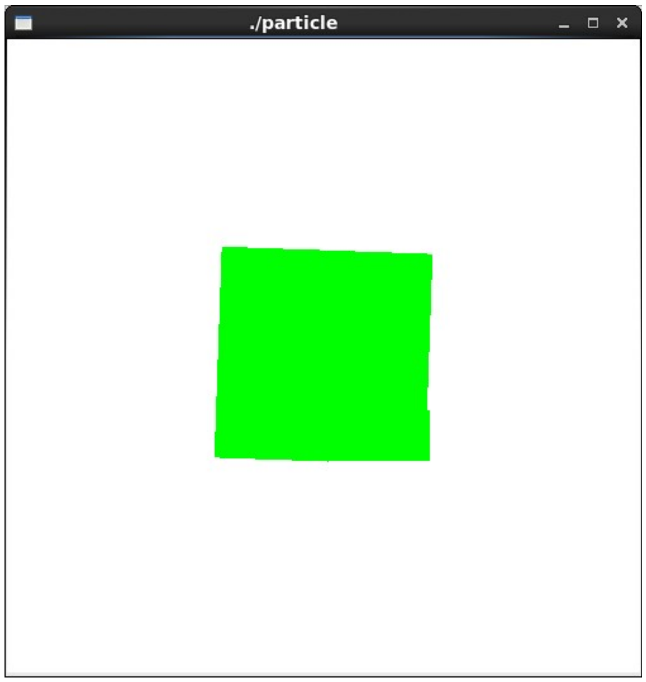
#1:



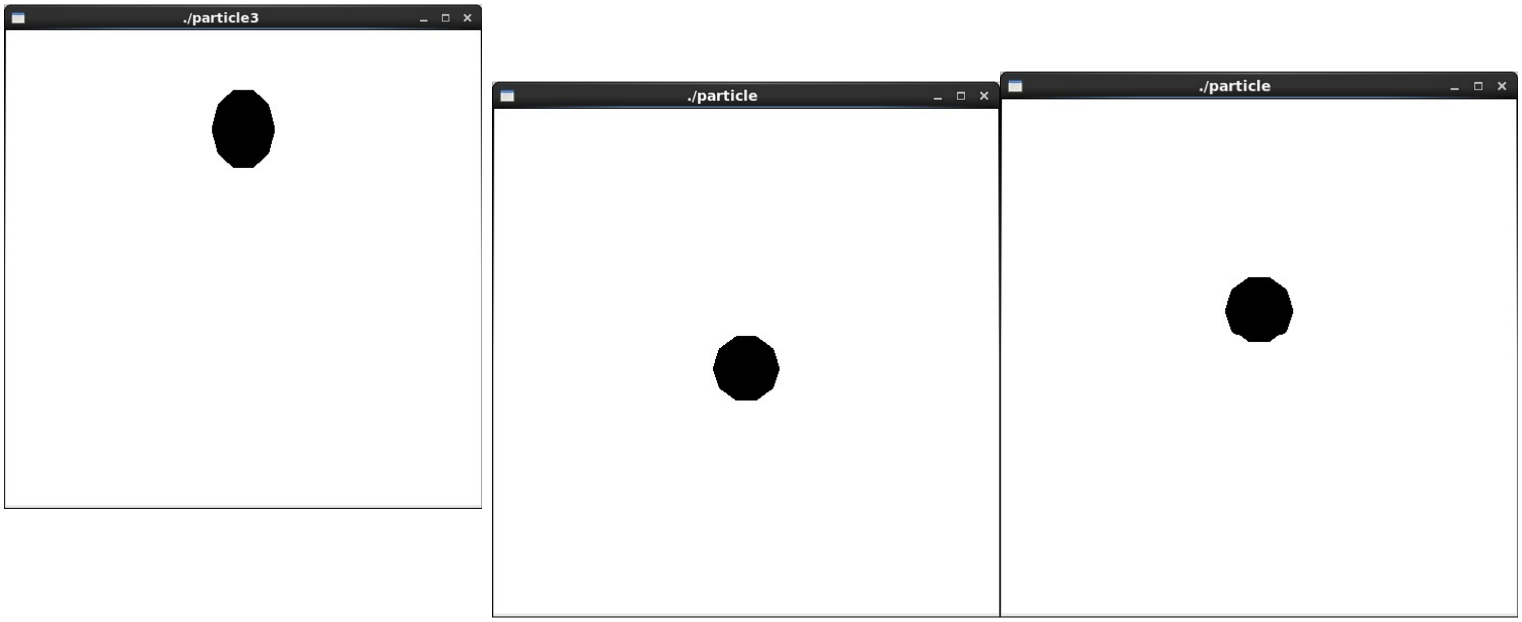


#2 :

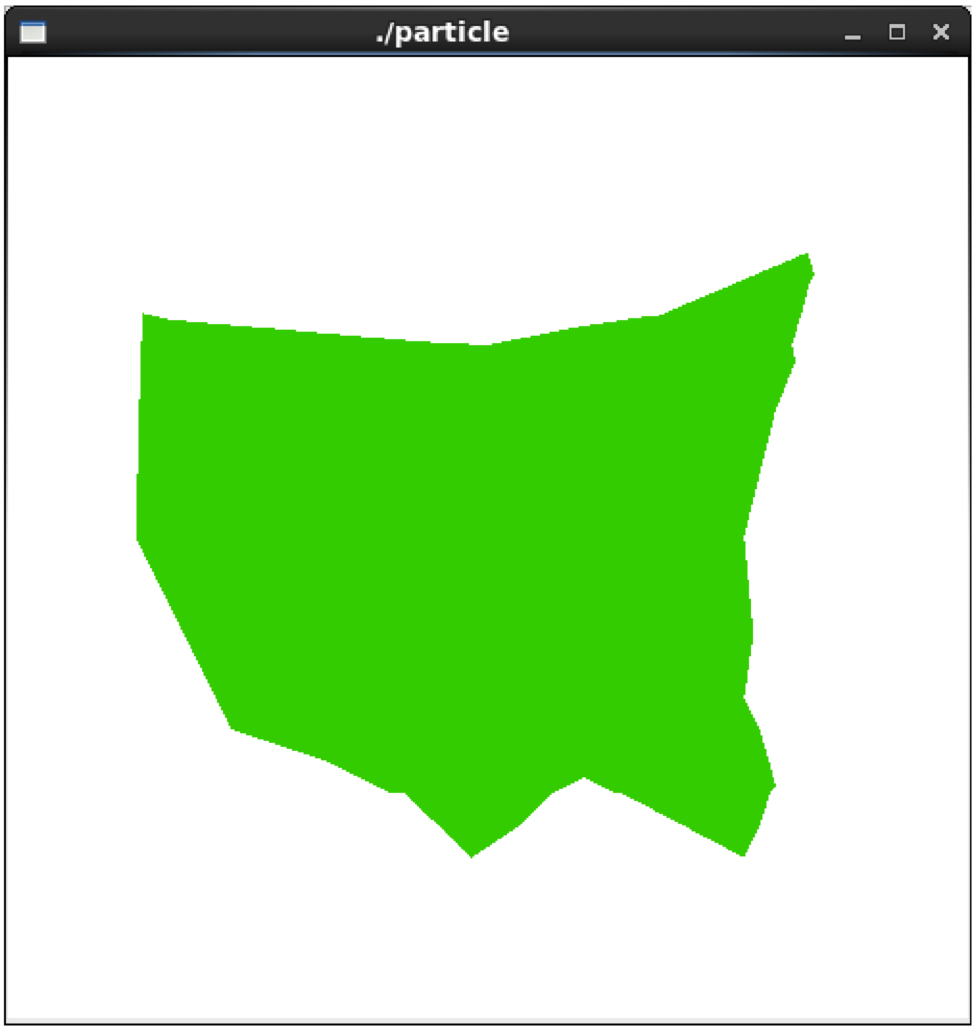




#3:



#4:



Code:
#1:
```
//particle.vert

uniform float time;                 //value provided by application program
attribute vec3 vel;                 //value provided by application program
varying vec3 color;

void main(void)
{

  float s = 1000.0;                 //scale factor
  float g = -10.0;
  float t;

  t =  time / s;              //time in ms
  vec4 object_pos = gl_Vertex;            //starting position
  float red = 0.0;
  float green = 0.0;
  float blue = 1.0;
  float j = 5.0;
  float k = j/2.0;



    if (t < k) {
        red = (t)/k;
        blue = (k-t)/k;
        object_pos.x = object_pos.x + ( (vel.x - object_pos.x) / k ) * t;
        object_pos.y = object_pos.y + ( (vel.y - object_pos.y) / k ) * t;
        object_pos.z = object_pos.z + ( (vel.z - object_pos.z) / k ) * t;
    }
    else {
        red = (j-t)/k;
        blue = (t-k)/k;
        object_pos.x = vel.x + ( (object_pos.x - vel.x) / k ) * (t - k);
        object_pos.y = vel.y + ( (object_pos.y - vel.y) / k ) * (t - k);
        object_pos.z = vel.z + ( (object_pos.z - vel.z) / k ) * (t - k);
    }

  color = vec3(red,green,blue);
  gl_Position = gl_ModelViewProjectionMatrix * object_pos;

}
```

This code is based on a time interval split in half, in which after the half-way point, the figure returns to it's original form.

#2:

```
//particle.vert

uniform float time;                 //value provided by application program
attribute vec3 vel;                 //value provided by application program
varying vec3 color;

void main(void)
{

  float s = 1000.0;                 //scale factor
  float g = -10.0;
  float t;
  float pi = 3.1415926535897932384626433;
  float angle = 360.0;
  float radius_angle = (angle*pi/(angle/2));
  t =  time / s;              //time in ms
  vec4 object_pos = gl_Vertex;            //starting position
  vec4 object2 = object_pos;
  float red = 0.0;
  float green = 0.0;
  float blue = 1.0;
  float j = 5.0;
  float k = j/2.0;

  object2.x = object_pos.x * cos(radius_angle/j*t) - object_pos.y * 
sin(radius_angle/j*t);
  object2.y = object_pos.y * cos(radius_angle/j*t) + object_pos.x * 
sin(radius_angle/j*t);


  color = vec3(red,green,blue);
  gl_Position = gl_ModelViewProjectionMatrix * object2;

}
```

#3:
```
//particle.vert

uniform float time;                 //value provided by application program
attribute vec3 v;            //value provided by application program
varying vec3 color;

void main(void)
{

  float s = 1000.0;                 //scale factor
  float g = -10.0;
  float t;
  float h, h0;      //h0 = initial height
  float t0;
  float c = 0.9;
  float bounce = 40.0;
  float pi = 3.1415926535897932384626433;
  float angle = 360.0;
  float radius_angle = (angle*pi/(angle/2.0));
  t =  time / s;              //time in ms
  vec4 object_pos = gl_Vertex;            //starting position
  float red = 0.0;
  float green = 0.0;
  float blue = 1.0;
  float j = 5.0;
  float k = j/2.0;
  h0 = gl_Vertex.y;   //initial height of ball
  vec3 norm = vec3 ( 0, 1, 0 );
  vec3 v1, v2;

  t0 = sqrt ( 2.0 * h0 / g ); //time to reach ground

  v1.x = v.x;
  v1.y = v.y - g * t0;
  v1.z = v.z;

  //initial height is gl_vertex.y
  h =  h0 - g/(2.0)*t*t; //height of ball at time t
  /*
  object_pos.x = object_pos.x + v.x*t;
  object_pos.y = object_pos.y + v.y*t + g/(2.0)*t*t;
  object_pos.z = object_pos.z + v.z*t;
*/
  while ( h <= 0.0 ){       //ball should always be above ground
    v2 = c * reflect( v1, norm );    //bouncing vocity reduced
    t = t - t0;
    h = v2.y * t - g/(2.0) * t * t;
    h0 = c * h0;        //reduced bouncing height
    t0 = 2.0 * sqrt ( 2.0 * h0 / g );   //time for one bounce
    v1.y = v2.y - g * t0;

  }

  color = vec3(red,green,blue);
  gl_Position = gl_ModelViewProjectionMatrix * vec4(gl_Vertex.x, h,
gl_Vertex.z,1);
}
```

#4:
```
void display(void)
{
   //float t = glutGet ( GLUT_ELAPSED_TIME );
   GLfloat vec[4];
   int loc = glGetAttribLocation(programObject, "color");
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glClearColor(1.0, 1.0, 1.0, 0.0);            //get white background color
   glColor3f(1, 0, 0);                    //red, this will have no effect if
shader is loaded
   glPointSize(20);
   //"shoot" a particle at 45 degrees
   glBegin (GL_POLYGON);
     glVertex3f(-11.0,0,0);
     glVertex3f(-10.9,4,0);
     glVertex3f(-10.8,7,0);
     glVertex3f(-10,6.8,0);
     glVertex3f(-2,6.1,0);
     glVertex3f(0,6,0);
     glVertex3f(3,6.6,0);
     glVertex3f(5,6.9,0);
     glVertex3f(7,7,0);
     glVertex3f(8,7.2,0);
     glVertexAttrib3f(loc,0.8,0,0);
     glVertex3f(10,8.9,0);
     glVertex3f(10.2,8.2,0);
     glVertex3f(10,7.8,0);
     glVertex3f(9.8,7,0);
     glVertex3f(9.5,6,0);
     glVertex3f(9.6,5.5,0);
     glVertex3f(9,4,0);
     glVertex3f(8,0,0);
     glVertex3f(8.2,-2,0);
     glVertex3f(8.3,-3,0);
     glVertex3f(8,-5,0);
     glVertex3f(8.5,-6,0);
     glVertex3f(9,-7.8,0);
     glVertex3f(8.8,-8,0);
     glVertex3f(8.5,-9,0);
     glVertex3f(8.25,-9.5,0);
     glVertex3f(8,-10,0);
     glVertex3f(7.5,-9,0);
     glVertex3f(7,-8,0);
     glVertex3f(6.5,-5,0);
     glVertex3f(7,-3,0);
     glVertex3f(5,-8,0);
     glVertex3f(4,-8,0);
     glVertex3f(3,-7.5,0);
     glVertex3f(2,-8,0);
     glVertex3f(1,-9,0);
     glVertexAttrib3f(loc,0.8,0,0);
     glVertex3f(-0.5,-10,0);
     glVertex3f(-2,-8,0);
     glVertex3f(-3,-8,0);
     glVertex3f(-5,-7,0);
     glVertexAttrib3f(loc,0.8,0,0);
     glVertex3f(-8,-6,0);
   glEnd();
```

```
    glutSwapBuffers();
    glFlush();

}
```