Yazhuo Liu
Final Exam

1. (30 points) A parametric curve C(t) is described by:
C(t) = (-cos(t), √3 t, sin(t))

a) Find a Frenet frame (N(t), B(t), T(t)) for this curve.

b) Find the 4x4 transformation matrix M that brings the world coordinate system (i, j, k) into this new coordinate system (N(t), B(t), T(t)).

a)

$$C'(t) = \begin{pmatrix} \sin(t) \\ \sqrt{3} \\ \cos(t) \\ 0 \end{pmatrix}$$

$$T(t) = 1/2 \begin{pmatrix} \sin(t) \\ \sqrt{3} \\ \cos(t) \\ 0 \end{pmatrix}$$

$$N(t) = \begin{pmatrix} \cos(t) \\ 0 \\ -\sin(t) \\ 0 \end{pmatrix}$$

$$B(t) = T(t) * N(t) = 1/2 \begin{pmatrix} \sqrt{3}\sin(t) \\ 1 \\ -\sqrt{3}\cos(t) \\ 0 \end{pmatrix} = \begin{pmatrix} \sqrt{3}/2 \sin(t) \\ 1/2 \\ -\sqrt{3}/2 \cos(t) \\ 0 \end{pmatrix}$$

b) In homogeneous coordinates,

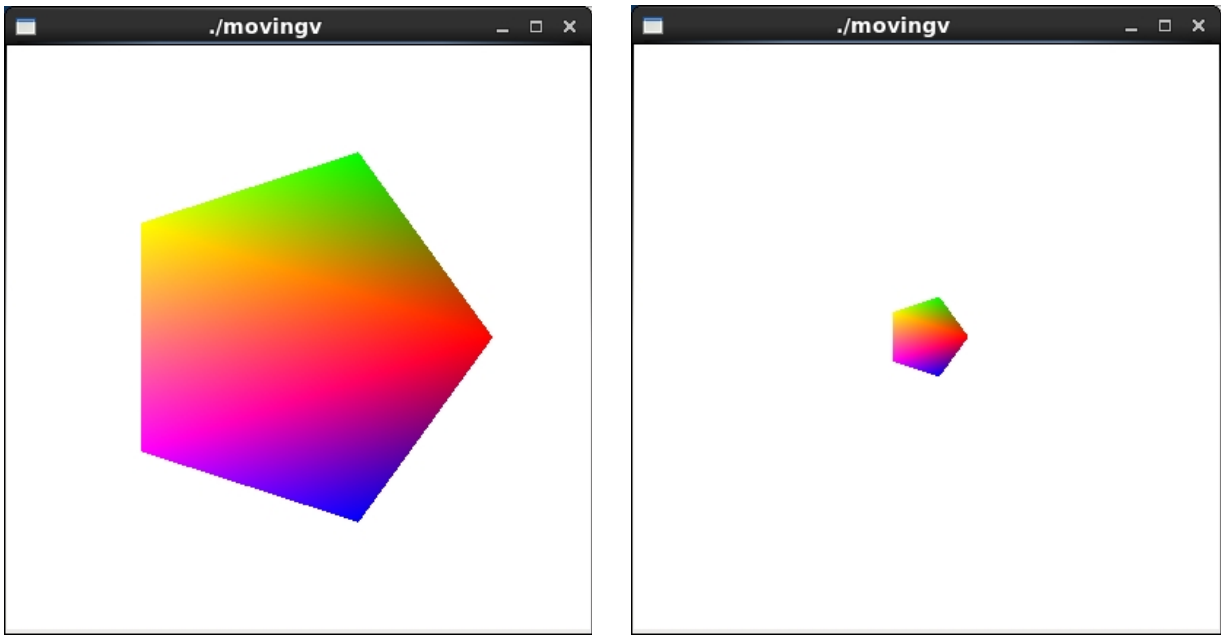
$$M = \begin{Bmatrix} \cos(ti) & \sqrt{3}/2 \sin(ti) & \sin(ti) & -\cos(ti) \\ 0 & 1/2 & \sqrt{3} & \sqrt{3}ti \\ -\sin(ti) & -\sqrt{3}/2 \cos(ti) & \cos(ti) & \sin(ti) \\ 0 & 0 & 0 & 0 \end{Bmatrix}$$

2. (30 points) Write a shader program that renders a scene where a red particle is moving along a diagonal of the screen window.

Code:

```
//particle.vert

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

void main(void)
{
  color = temp;

  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

  object_pos.x = object_pos.x + vel.x*t;
  object_pos.y = object_pos.y - vel.y*t;
  object_pos.z = object_pos.z + vel.z*t;

  gl_Position = gl_ModelViewProjectionMatrix * object_pos;
}
```

3. (30 points) Write an shader program that renders a regular pentagon with different colors at the vertices. Pass a time parameter to the shader from the application, so that the pentagon shrinks and expands periodically.



Code:

```
//pentagon.cpp

...

void display(void)
{
  GLfloat vec[4];

  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glClearColor( 1.0, 1.0, 1.0, 0.0 );   //get white background color

  int loc = glGetAttribLocation(programObject, "temp" );

  glPushMatrix();
  glRotatef( anglex, 1.0, 0.0, 0.0);          //rotate the cube along x-axis
  glRotatef( angley, 0.0, 1.0, 0.0);          //rotate along y-axis
  glRotatef( anglez, 0.0, 0.0, 1.0);          //rotate along z-axis

  float a = 2 * 3.14159265 / 5;
  float a1 = 0;

  glBegin ( GL_POLYGON );
    glVertexAttrib3f(loc,1,0,0);
    glVertex2f (cos (a1 ), sin (a1) );
    a1 += a;
```

```
    glVertexAttrib3f(loc,0,1,0);
    glVertex2f (cos (a1 ), sin (a1) );
    a1 += a;
    glVertexAttrib3f(loc,1,1,0);
    glVertex2f (cos (a1 ), sin (a1) );
    a1 += a;
    glVertexAttrib3f(loc,1,0,1);
    glVertex2f (cos (a1 ), sin (a1) );
    a1 += a;
    glVertexAttrib3f(loc,0,0,1);
    glVertex2f (cos (a1 ), sin (a1) );
  glEnd();

  glPopMatrix();

  glutSwapBuffers();
  glFlush();
}

...

//pentagon.vert

uniform float time;     //value provided by application program

attribute vec3 temp;
varying vec3 color;

void main(void)
{
 float s = 0;

 angle += 2.0 * sin ( 0.0005 * time );

 s = s + 2.0 * sin ( 0.0005 * time );

 color = temp;

 gl_Position = gl_ModelViewProjectionMatrix * (vec4(s, s, s, 1.0)  * gl_Vertex );
}

//pentagon.frag

varying vec3 color;

void main(void)
{
  gl_FragColor = vec4( color, 1);
}
```

Report:

I have finished all parts of the final, and I have shown the professor the output of my program. Therefore, I am giving myself 100 points.