porblem 1

Here we arrange 10 teapots into circle around vertex

Here xRotated = 0, yRotated =0

```
void problem1() {
    for (int i = 0; i < 10; i++){
     glMatrixMode(GL_MODELVIEW);
      // clear the identity matrix.
     glLoadIdentity();
     // traslate the draw by x=(0.5*cos(0.628*i), y=0.5*sin(0.628*i), z = -3.0
This translate the position each teapot
     glPushMatrix();
       glTranslated(0.5*cos(0.628*i),0.5*sin(0.628*i),-3);
       glColor3f(0.8, 0.2, 0.1);
       glRotatef(xRotated,1.0,0.0,0.0);
       // rotation about Y axis
       glRotatef(yRotated,0.0,1.0,0.0);
       // rotation about Z axis
This translate the rotation about Z axis each angle(36)
       glRotatef(36*i,0.0,0.0,1.0);
       // scaling transfomation
       glScalef(1.0,1.0,1.0);
     // built-in (freeglut library) function , draw you a Teapot.
     glutSolidTeapot(0.1);
   }
//This is translate coordinate;
   glPushMatrix();
       glTranslated(0,0,-3);
       glRotated(0,0,0,1);
```

```
        glLoadIdentity();
        gluLookAt(cameraPos[0], cameraPos[1], cameraPos[2], 0, 0, 0, 0, 1, 0);


        glLightfv(GL_LIGHT0, GL_POSITION, cameraPos);


        glRotatef(yRot,0,1,0);
}
```

problem 2

As you can see stairs is divied into 15 spiece

First arrange 15 cubes into scene.

Second arrange one more stair than previous stair with 25 cubes scaled 0.02

```
void problem2() {
  for(int i = 0; i < 15; i++){ //arrange 15 cubes into scene
    glLoadIdentity();
    glTranslatef(-0.8 + i * 0.1, 0 ,-3);
    glColor3f(0.8, 0.2, 0.1);
    glRotatef(xRotated,1.0,0.0,0.0);
    glRotatef(yRotated,0.0,1.0,0.0);
    glRotatef(zRotated,0.0,0.0,1.0);
    glScalef(1.0,1.0,1.0);
    glutSolidCube(0.1);
    for(int j = 0; j < i+1; j++){//Second part
      for (float x = -2; x < 3; x++){// vertical arrow
        for (float z = -2; z < 3; z++){//Horizontal arrow
          glLoadIdentity();
          glTranslatef(-0.8 + i * 0.1 + x * 0.02, 0.05+j*0.01 ,-3 + z * 0.02);
          glColor3f(0.8, 0.2, 0.1);
          glRotatef(xRotated,1.0,0.0,0.0);
```

```
        glRotatef(yRotated,0.0,1.0,0.0);
        glRotatef(zRotated,0.0,0.0,1.0);
        glScalef(1.0,1.0,1.0);
        glutSolidCube(0.02);
      }
     }
   }
 }
```

This is translate coordinate;

```
   glPushMatrix();
    glTranslated(-0.8,0,-3);
    glLoadIdentity();
    gluLookAt(cameraPos[0], cameraPos[1], cameraPos[2], 0, 0, 0, 0, 1, 0);

    glLightfv(GL_LIGHT0, GL_POSITION, cameraPos);

    glRotatef(yRot,0,1,0);
}
```

Problem3

Here we arrange 21 teapots into pyramid

First arrange one teapot at the top

```
void problem3() {
   glMatrixMode(GL_MODELVIEW);
   for (int i = 0; i < 6; i++){//Each line
    for( float j = -0.5 * i/2; j <= 0.5 * i/2; j= j + 0.5){// The number of teapot each line
      glLoadIdentity();
      glPushMatrix();
       glTranslatef(j, 0.6 - i * 0.3,-3);
```

```
        glColor3f(0.8, 0.2, 0.1);
        glRotatef(xRotated,1.0,0.0,0.0);
        glRotatef(yRotated,0.0,1.0,0.0);
        glRotatef(zRotated,0.0,0.0,1.0);
        glScalef(1.0,1.0,1.0);
       glutSolidTeapot(0.1);
     }

   }
```

This is translate coordinate;

```
    glPushMatrix();
     glTranslated(0,0,-3);
     glLoadIdentity();
     gluLookAt(cameraPos[0], cameraPos[1], cameraPos[2], 0, 0, 0, 0, 1, 0);

     glLightfv(GL_LIGHT0, GL_POSITION, cameraPos);

     glRotatef(yRot,0,1,0);
}
```

Problem4

Here let draw apple.

First we have to point over 1000 vertex onto surface of apple which we want to draw.

And save it into apple.dat

Second we import apple.dat

Here fname = apple.dat

```
void ReadModel()
{
```

```
	FILE* f1; char s[81]; int i;
	if (mpoint != NULL) delete mpoint;
	if (mface != NULL) delete mface;
	if ((f1 = fopen(fname.c_str(), "rt")) == NULL) { printf("No file\n"); exit(0); }
	fscanf(f1, "%s", s); printf("%s", s); fscanf(f1, "%s", s); printf("%s", s);
	fscanf(f1, "%d", &pnum); printf("%d\n", pnum);
	mpoint = new Point[pnum];
	for (i = 0; i < pnum; i++) {
		fscanf(f1, "%f", &mpoint[i].x); fscanf(f1, "%f", &mpoint[i].y); fscanf(f1, "%f",
&mpoint[i].z);
		printf("%f %f %f\n", mpoint[i].x, mpoint[i].y, mpoint[i].z);
	}
	fscanf(f1, "%s", s); printf("%s", s);
	fscanf(f1, "%s", s); printf("%s", s);
	fscanf(f1, "%d", &fnum); printf("%d\n", fnum);
	mface = new Face[fnum];
	for (i = 0; i < fnum; i++) {
		fscanf(f1, "%d", &mface[i].ip[0]); fscanf(f1, "%d", &mface[i].ip[1]); fscanf(f1,
"%d", &mface[i].ip[2]);
		printf("%d %d %d\n", mface[i].ip[0], mface[i].ip[1], mface[i].ip[2]);
	}
	fscanf(f1, "%s", s); printf("%s", s); fscanf(f1, "%s", s); printf("%s", s); fscanf(f1, "%f", s);
printf("%f\n", s);
	fscanf(f1, "%f", &ground); printf("%f\n", ground);
	fclose(f1);
}
```

Third draw vector with all points

```
Point cnormal(Point a, Point b, Point c) {
	Point p, q, r;
```

```
        double val;
        p.x = a.x - b.x; p.y = a.y - b.y; p.z = a.z - b.z;
        q.x = c.x - b.x; q.y = c.y - b.y; q.z = c.z - b.z;
        r.x = p.y * q.z - p.z * q.y;
        r.y = p.z * q.x - p.x * q.z;
        r.z = p.x * q.y - p.y * q.x;
        val = sqrt(r.x * r.x + r.y * r.y + r.z * r.z);
        r.x = r.x / val; r.y = r.y / val; r.z = r.z / val;
        return r;
}
```

Fourth draw all points and draw into triangle mesh

```
void DrawModel(float varx, float vary, float varz, float movex, float movey, float movez) {
        int i;
        glPushMatrix();
        glRotatef(rotatex, 0.0, 0.0, 1.0);
        glRotatef(rotatey + xRotAngle, 0.0, 1.0, 0.0);
        glRotatef(rotatez + yRotAngle, 1.0, 0.0, 0.0);
        glScalef(1, 1, 1);
        glColor3f(varx, vary, varz);
        for (i = 0; i < fnum; i++) {
                Point norm = cnormal(mpoint[mface[i].ip[2]], mpoint[mface[i].ip[1]],
mpoint[mface[i].ip[0]]);
                glBegin(GL_TRIANGLES);
                glNormal3f(norm.x, norm.y, norm.z);
                glVertex3f(mpoint[mface[i].ip[0]].x , mpoint[mface[i].ip[0]].y,
mpoint[mface[i].ip[0]].z);
                glNormal3f(norm.x, norm.y, norm.z);
  glVertex3f(mpoint[mface[i].ip[1]].x , mpoint[mface[i].ip[1]].y, mpoint[mface[i].ip[1]].z);
```

```
                glNormal3f(norm.x, norm.y, norm.z);
  glVertex3f(mpoint[mface[i].ip[2]].x , mpoint[mface[i].ip[2]].y, mpoint[mface[i].ip[2]].z);
                glNormal3f(norm.x, norm.y, norm.z);
                glEnd();
        }
        glPopMatrix();

}
```

Last display the result.