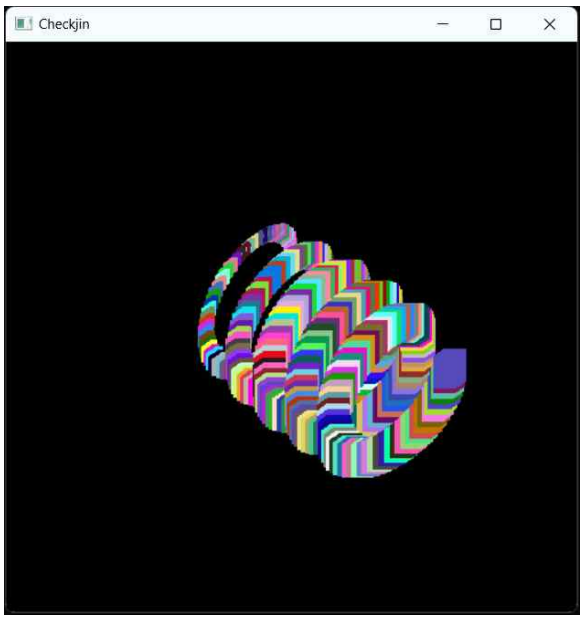
<컴퓨터 그래픽스 문제>

아래의 그림처럼 출력하도록 OpenGL 코드를 C++언어로 작성하시오.



조건)

1. DisplayMode는 GLUT_SINGE 버퍼를 사용할 것.
2. 윈도우 창의 크기는 500x500으로 설정할 것.
3. 윈도우 창의 위치는 (100,100)위치로 설정할 것.
4. 윈도우 창의 Title은 “Checkjin”으로 설정할 것.
5. 윈도우 창의 배경색은 검정색으로 설정할 것.
6. 윈도우 창의 크기 변화에 따른 모델의 크기 변화를 일정하게 유지할 것.
7. 모델을 윈도우 창의 1/4 크기로 설정할 것.
8. 모델의 회전을 5바퀴로 설정할 것.
9. 카메라의 회전 각도를 주어진 사진을 보고 올바르게 설정할 것.
10. 기본으로 제공해주는 코드를 최대한 활용할 것.

기본제공코드)
- nomal -> hard 순으로 해결해볼 것.
- 주석으로 처리된 부분을 코딩하여 주석을 해제할 것.

*** nomal code ***

```
#include <GL/glut.h>
#include <stdio.h>
#include <iostream>
#include <math.h>

#define GL_PI 3.1415f

void glPointSize(GLfloat s);

float getRandomFloat() { //색상의 랜덤 값 설정
    return (float)rand() / (float)RAND_MAX;
}

void RenderScene() {
        GLfloat x, y, z, angle;
        GLfloat r=0.0f, g=0.0f, b=0.0f;
        GLfloat sizes[2], step;
        GLfloat pointSize;
        glGetFloatv(GL_POINT_SIZE_RANGE, sizes);
        glGetFloatv(GL_POINT_SIZE_GRANULARITY, &step);

        glClear(GL_COLOR_BUFFER_BIT);
        glMatrixMode(GL_PROJECTION);
        glPushMatrix();

        // glRotatef();
        // glRotatef();

        // z = ;
        // float squaresize = ;
        // float spacing = ;
        // float halfsize = ;

        pointSize = sizes[0];

        // for (angle = 0.0f; angle <= ; angle += 0.1f) { // angle의 최대값을 설정할 것.
                // x =  * cos(angle);
```

```
                // y =  * sin(angle);

                // r = ;
                // g = ;
                // b = ;

                glColor3f(r, g, b);
                glPointSize(pointSize);
                glBegin(GL_POINTS);
                // for (float dx = -halfsize; dx <= halfsize; dx += spacing) { //왼쪽에서
오른쪽으로 점을 찍음
                        // glVertex3f(x + , y + , z); // 윗변
                        // glVertex3f(x + , y - , z); // 아랫변
                //}
                //for (float dy = -halfsize; dy <= halfsize; dy += spacing) { //아래쪽에서
위쪽으로 점을 찍음
                //      glVertex3f(x - , y + , z); //좌측변
                //      glVertex3f(x + , y + , z); //우측변
                //}
                glEnd();

                // z += f;
                // pointSize += ;
                if (pointSize > sizes[1]) {
                        pointSize = sizes[1];
                }

        }

        glPopMatrix();
        glFlush();
}

void SetupRC(void) {
        std::cout << "SetupRC" << std::endl;
        // glClearColor();
}

void changeSize(GLsizei w, GLsizei h) {
        GLfloat t = 100;
        GLfloat wSizeRatio;
        // glViewport();
```

```
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();

	// wSizeRatio = ;
	//if (w <= h) {
	//	glOrtho();
	//}
	//else {
	//	glOrtho();
	//}

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

int main(int argc, char** argv) {
	glutInit(&argc, argv);
	// glutInitDisplayMode( | GLUT_RGB);
	// glutInitWindowSize(, );
	// glutInitWindowPosition(, );
	// glutCreateWindow();

	SetupRC();
	//glutDisplayFunc();
	//glutReshapeFunc();
	glutMainLoop();
}
```

*** hard code ***

```
#include <GL/glut.h>
#include <stdio.h>
#include <iostream>
#include <math.h>

#define GL_PI 3.1415f

void glPointSize(GLfloat s);

float getRandomFloat() {
    return ;
}

void RenderScene() {

}

void SetupRC(void) {
        std::cout << "SetupRC" << std::endl;
}

void changeSize(GLsizei w, GLsizei h) {

}

int main(int argc, char** argv) {
        glutInit(&argc, argv);
        // glutInitDisplayMode( | GLUT_RGB);
        // glutInitWindowSize(, );
        // glutInitWindowPosition(, );
        // glutCreateWindow();

        SetupRC();
        //glutDisplayFunc();
        //glutReshapeFunc();
        glutMainLoop();
}
```