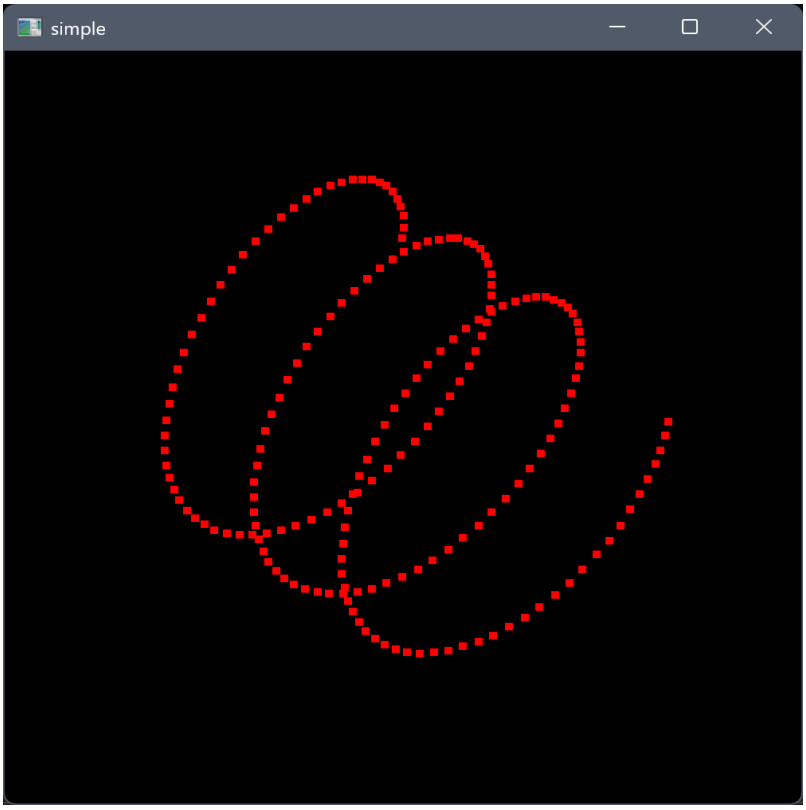
2025-04-19_CG_09_나선_01_점으로 나선 그리기

📁 예제 설명: 점으로 나선 그리기

📝 목표 출력

- 

📝 조건

- Z축으로 회전하며 나선을 그린다.
- 3번 회전한다.

📁 해결 코드

📝 핵심 코드

```
glBegin(GL_POINTS);
z = -50.0f;
for (angle = 0.0f; angle <= (2.0f * GL_PI) * 3.0f; angle += 0.1f) {
    x = 50.0f * cos(angle);
    y = 50.0f * sin(angle);
    glVertex3f(x, y, z);
    z += 0.5f;
```

```
}
glEnd();
```

📝 전체 코드

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

#define GL_PI 3.1415f

void RenderScene(void) {

    GLfloat x, y, z, angle;

    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 0.0f, 0.0f);
    glPointSize(5.0f);

    // 📌
    glPushMatrix();
    glRotatef(45, 1.0f, 0.0f, 0.0f);
    glRotatef(45, 0.0f, 1.0f, 0.0f);

    // 📌
    glBegin(GL_POINTS);
    z = -50.0f;
    for (angle = 0.0f; angle <= (2.0f * GL_PI) * 3.0f; angle += 0.1f) {
        x = 50.0f * cos(angle);
        y = 50.0f * sin(angle);
        glVertex3f(x, y, z);
        z += 0.5f;
    }
    glEnd();

    glPopMatrix(); // 📌

    glFlush();
}

void ChangeSize(GLsizei w, GLsizei h) {

    GLint wSize = 100.0f;
    GLfloat aspectRatio;

    if (h == 0) h = 1;

    glViewport(0, 0, w, h);
```

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

    aspectRatio = (GLfloat)w / (GLfloat)h;
    if (aspectRatio >= 1.0f) {
        glOrtho(-wSize*aspectRatio, wSize*aspectRatio, -wSize, wSize, -wSize, wSize);
    }
    else {
        glOrtho(-wSize, wSize, -wSize/aspectRatio, wSize/aspectRatio, -wSize, wSize);
    }

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void SetupRC(void) {
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
}

int main(int argc, char** argv) {

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE ¦ GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(400, 400);

    glutCreateWindow("simple");

    SetupRC();

    glutDisplayFunc(RenderScene);
    glutReshapeFunc(ChangeSize);

    glutMainLoop();
}
```

📁 설명

📝 Z 방향으로 이동하며 원 그리기

```
glBegin(GL_POINTS);
z = -50.0f;
for (angle = 0.0f; angle <= (2.0f * GL_PI) * 3.0f; angle += 0.1f) {
    x = 50.0f * cos(angle);
    y = 50.0f * sin(angle);
```

```
    glVertex3f(x, y, z);
    z += 0.5f;
}
glEnd();
```

- `(2.0f * GL_PI) * 3.0f` : 2π * 3 -> 세 바퀴를 돌기 위한 총 각도이다.
- `z += 0.5f` : z 좌표를 0.5씩 증가시켜서 나선 형태로 만든다.

📝 glPushMatrix(), glPopMatrix()

- OpenGL에서 **변환 행렬**(Transformation Matrix)을 스택에 저장하고, 이를 다시 복원하는 함수.
- `glPushMatrix()` :
  - 여러 변환을 중첩해서 적용할 때, **이전 상태를 저장**해두고, 필요할 때 복원하는 데 사용한다.
  - 변환 작업을 한 후, 나중에 `glPopMatrix()`를 호출하면 저장된 이전 상태로 되돌릴 수 있다.
- `glPopMatrix()` :
  - 여러 번의 변환 작업이 끝난 후, **이전 상태로 돌아가고자** 할 때 사용된다.
  - 즉, `glPushMatrix()`로 저장했던 변환 상태를 복원한다.

📎 예시

```
glPushMatrix();  // 현재 변환 행렬을 스택에 저장

// 변환 적용
glRotatef(45.0f, 1.0f, 0.0f, 0.0f);  // x축 기준으로 45도 회전
glTranslatef(1.0f, 2.0f, 0.0f);     // (1, 2, 0)만큼 이동

// 변환된 상태에서 그리기
glBegin(GL_TRIANGLES);
// 삼각형 그리기
glEnd();

glPopMatrix();  // 변환 행렬을 이전 상태로 복원
```

📝 glRotate()

```
glRotatef(angle, x, y, z);
```

- (1.0f, 0.0f, 0.0f) → X축
- (0.0f, 1.0f, 0.0f) → Y축
- (0.0f, 0.0f, 1.0f) → Z축

```
glRotatef(45, 1.0f, 0.0f, 0.0f);  // X축을 기준으로 45도 회전
glRotatef(45, 0.0f, 1.0f, 0.0f);  // Y축을 기준으로 45도 회전
```

📝 ChangeSize() -> glOrtho 수정

- 원을 z축 방향으로 나선처럼 이동시키며 그리기 때문에, z축의 클리핑 범위도 충분히 확보해야 한다.
- 만약 z축 범위를 `-1`에서 `1`로 제한하면, 대부분의 점들이 클리핑 범위를 벗어나기 때문에 화면에 보이지 않는다.

```
aspectRatio = (GLfloat)w / (GLfloat)h;
if (aspectRatio >= 1.0f) {
    // glOrtho(-wSize*aspectRatio, wSize*aspectRatio, -wSize, wSize, 1, -1);
    glOrtho(-wSize*aspectRatio, wSize*aspectRatio, -wSize, wSize, -wSize, wSize);
}
else {
    //  glOrtho(-wSize, wSize, -wSize/aspectRatio, wSize/aspectRatio, 1, -1);
    glOrtho(-wSize, wSize, -wSize/aspectRatio, wSize/aspectRatio, -wSize, wSize);
}
```

🖊 클리핑?

- **카메라(view volume)** 바깥에 있는 그래픽 요소를 **잘라내는 과정.**
- 화면에 그려질 수 있는 **3차원 공간 범위(view volume)** 가 있다.
  - 이 공간은 `glOrtho()`나 `gluPerspective()` 등을 이용해 직접 지정한다.
  - 그 범위 바깥에 있는 점이나 도형은 **보이지 않게 잘려 나간다.**
  - 이 과정을 **"클리핑"** 이라고 한다.