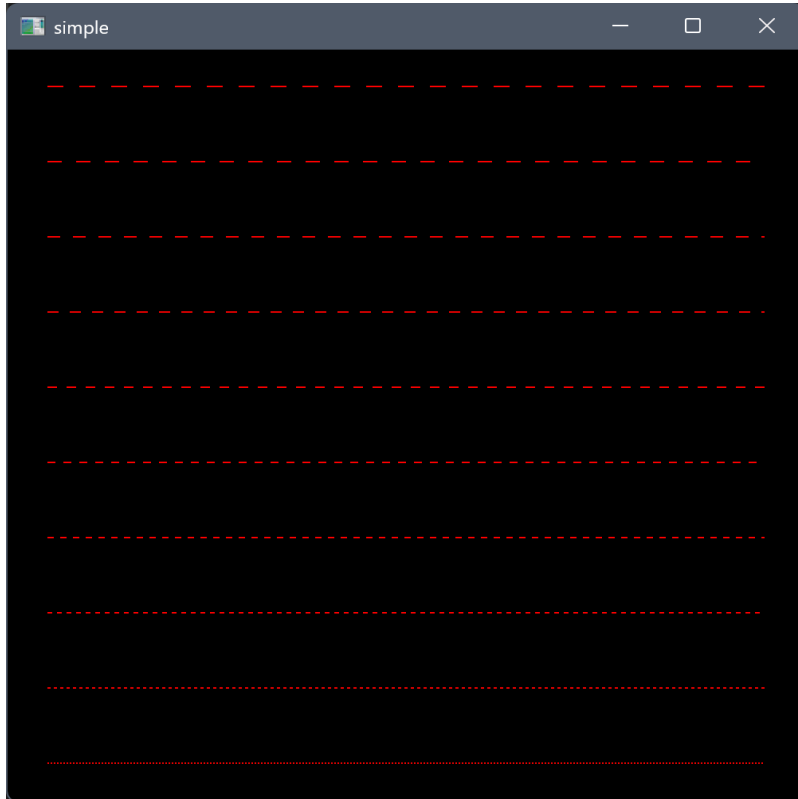


| 2025-04-19_CG_19_점선_02_점선 그리기 예제

| 📁 예제 설명:

| 📄 목표 출력



| 📁 해결 코드

| 📄 핵심 코드

```
void RenderScene(void) {  
  
    GLfloat y;  
    GLint factor = 1;  
    GLushort pattern = 0x5555;  
  
    glClear(GL_COLOR_BUFFER_BIT);  
    glColor3f(1.0f, 0.0f, 0.0f);  
    glPushMatrix();  
  
    glEnable(GL_LINE_STIPPLE); // ✖  
    for (y = -90.0f; y <= 90.0f; y += 20.0f) {
```

```

        glLineStipple(factor, pattern); // 📌

        glBegin(GL_LINES);
        glVertex2f(-90.0f, y);
        glVertex2f(90.0f, y);
        glEnd();

        factor++;

    }

    glPopMatrix();

    glFlush();
}

```

📄 전체 코드

```

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

#define GL_PI 3.1415f

void RenderScene(void) {

    GLfloat y;
    GLint factor = 1;
    GLushort pattern = 0x5555;

    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 0.0f, 0.0f);
    glPushMatrix();

    glEnable(GL_LINE_STIPPLE);
    for (y = -90.0f; y <= 90.0f; y += 20.0f) {
        glLineStipple(factor, pattern);

        glBegin(GL_LINES);
        glVertex2f(-90.0f, y);
        glVertex2f(90.0f, y);
        glEnd();

        factor++;

    }

    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(100, 100);

    glutCreateWindow("simple");

    SetupRC();

    glutDisplayFunc(RenderScene);
    glutReshapeFunc(ChangeSize);

    glutMainLoop();
}

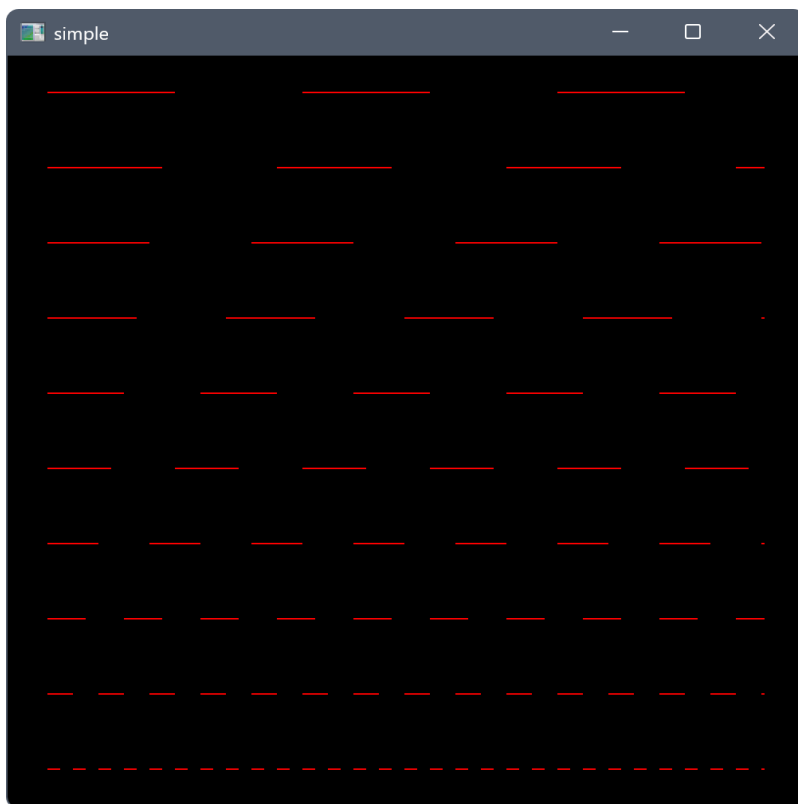
```

📁 설명

- pattern: `0x5555` = `0101 0101 0101 0101` -> 줄무늬 느낌
- factor 1 → 
- factor 2 → 
- factor 3 → 

📁 예제: 점선을 더 길게 하기

📄 목표 출력



📄 조건

- factor는 수정할 수 없다. pattern만 변경하기.

📄 해결 코드

```
GLushort pattern = 0x00FF;
```

- `0x00FF` 는 16비트 중 하위 8비트만 켜진 상태 (즉, `0000 0000 1111 1111`)
- 8 픽셀 그리기 → 8 픽셀 건너뛰기를 반복한다.

I 전체 코드

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

#define GL_PI 3.1415f

void RenderScene(void) {

    GLfloat y;
    GLint factor = 1;
    GLushort pattern = 0x00FF; // ✖

    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 0.0f, 0.0f);
    glPushMatrix();

    glEnable(GL_LINE_STIPPLE);
    for (y = -90.0f; y <= 90.0f; y += 20.0f) {
        glLineStipple(factor, pattern);

        glBegin(GL_LINES);
        glVertex2f(-90.0f, y);
        glVertex2f(90.0f, y);
        glEnd();

        factor++;
    }

    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(100, 100);

    glutCreateWindow("simple");

    SetupRC();

    glutDisplayFunc(RenderScene);
    glutReshapeFunc(ChangeSize);


    glutMainLoop();
}

```

- factor 가 1일 때 → 패턴 그대로 (8픽셀 그리고, 8픽셀 건너뛴)
- factor 가 2일 때 → 각 비트가 **2배 길이로 늘어남** → 16픽셀 그리고, 16픽셀 건너뛴
- factor 가 3이면 → 24픽셀 그리고, 24픽셀 건너뛴


추가 설명

0x1111, 0x5555, 0xFFFF의 차이는?


 1. 0x5555 = 0101 0101 0101 0101

- 1비트 그린 후 1비트 비움 → 규칙적인 "짧은 점선"
- 출력 느낌: ■ □ ■ □ ■ □ ■ □ ...
- 점선 중에서도 **가장 자주 끊기는 스타일**

2. $0x1111 = 0001\ 0001\ 0001\ 0001$

- 1비트만 간간이 찍힘
- 출력 느낌: 
- 매우 드문 간격으로 짧게 찍힘
- 매우 희미하고 드문 점선

3. $0xFFFF = 1111\ 1111\ 1111\ 1111$

- 전부 1 → 모든 비트 그리기
- 출력 느낌: 
- 사실상 실선과 동일