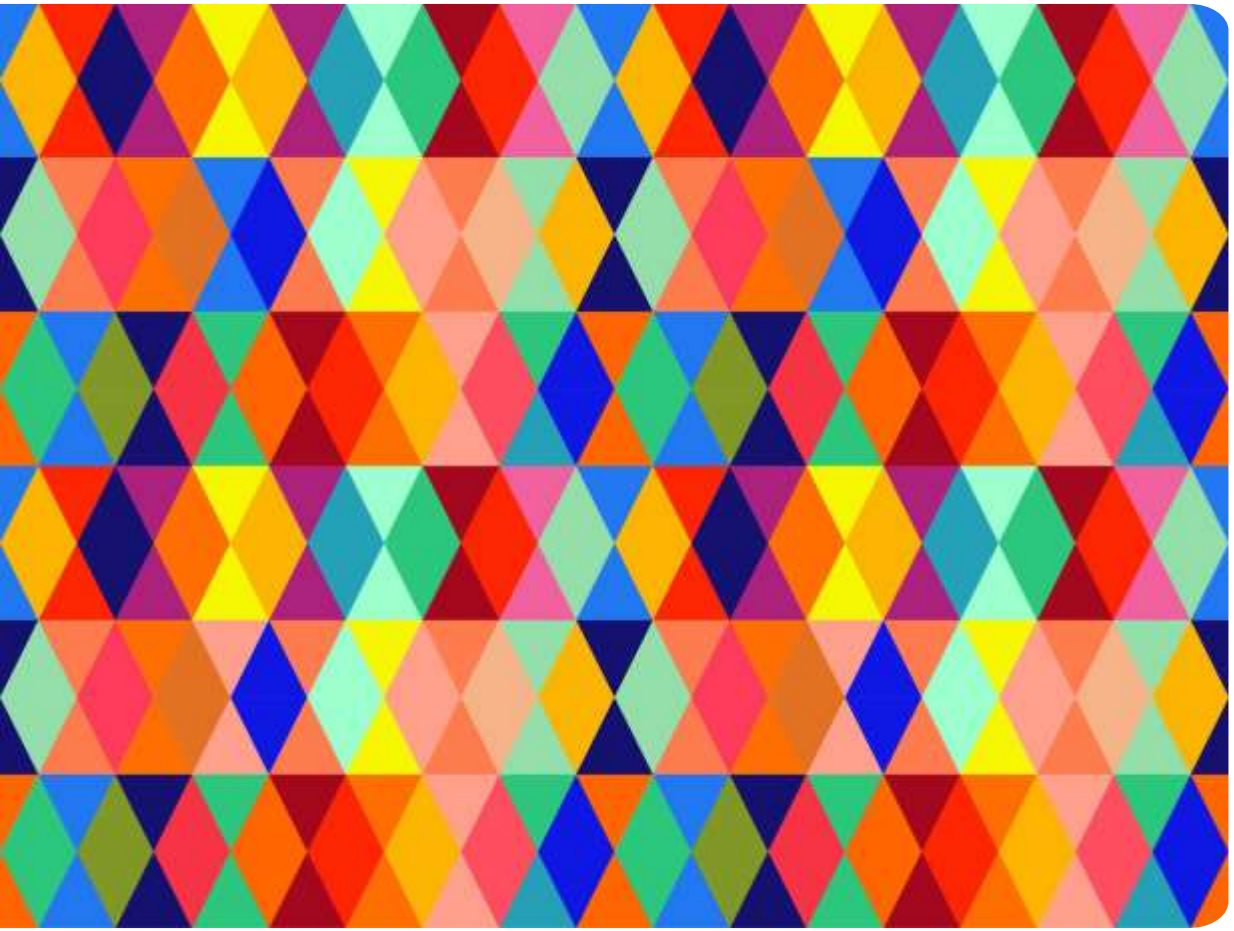


ICE3016 미니프로젝트

12191765 박승재

볼펜 3차원 제품 카탈로그 제작

프로젝트 주제

- 볼펜 3차원 제품 카탈로그 제작
- OpenGL과 마우스, 키보드를 이용한 interactive 프로젝트 작성
- Blender 모델링보다는 OpenGL을 활용한 동적인 기능 구현에 중점을 둘 것

프로젝트 주제 선정 이유

- 볼펜은 실생활에서 가장 쉽게 접할 수 있는 물체임.
- 대략적인 형태를 모델링하는 것은 쉽지만, 그립감을 위한 볼펜 몸통의 곡선을 섬세하게 구현하기 위해서는 많은 노력이 필요함.
- 사람들이 볼펜을 가지고 하는 interactive한 동작(볼펜 딸깍거리기, 이유없이 볼펜 분해하기)을 OpenGL 프로그램으로 구현해 볼 것임.
- 나아가 볼펜으로 종이에 그림을 그리는 기능까지 구현하며 bmp 텍스처와 카메라 이동을 깊게 학습함.

구현사양

① Blender를 이용해 **볼펜을 모델링**하고 OpenGL 프로그램을 이용해 보여준다. ‘space’를 누르면 볼펜촉을 꺼낼 수 있다.

② 마우스와 ‘w’, ‘a’, ‘s’, ‘d’, ‘z’, ‘x’를 이용해 **카메라의 시점과 위치를 이동**할 수 있다. ‘1’를 누르면 그리기 모드로 이동한다.

③ 마우스를 이용해 볼펜 색을 바꿀 수 있다. 볼펜 색이 바뀌면 볼펜 모델의 색도 바뀐다.

구현사양

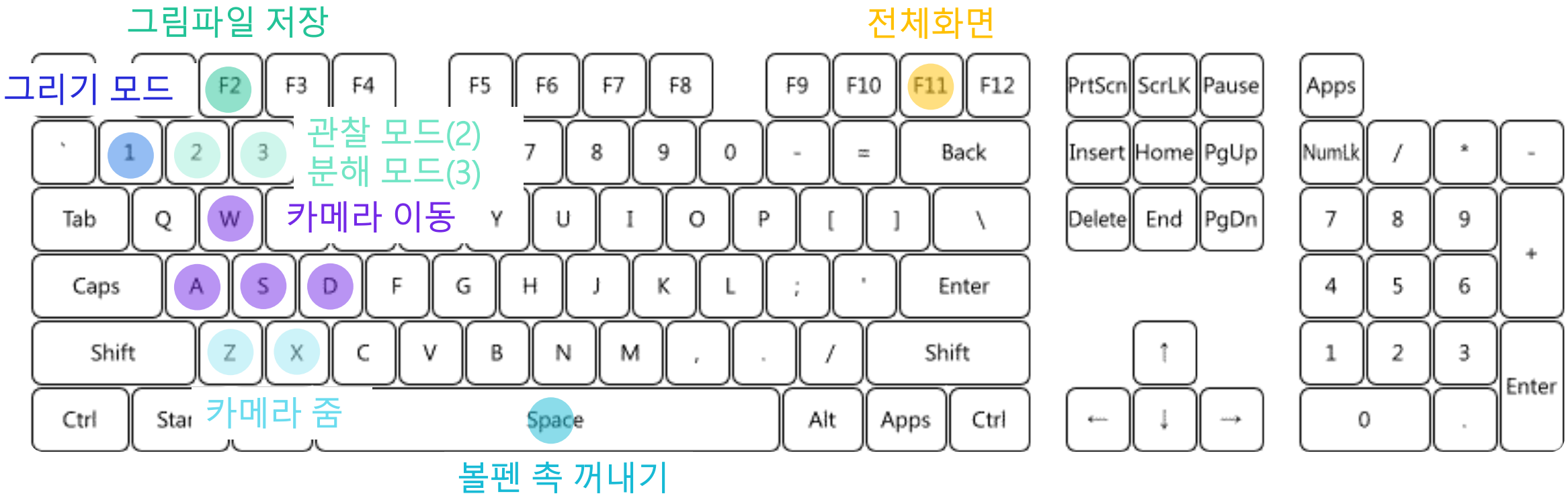
④ 디지털 카메라를 이용해 볼펜의 **텍스처**를 추출하고 매핑한다.

⑤ 텍스처를 재사용하는 등의 **리소스를 효율적**으로 사용할 수 있도록 구현한다.

⑥ 효과음과 애니메이션을 이용해 자연스러운 연출을 구현한다.

볼펜 모델링 세부 구현사양

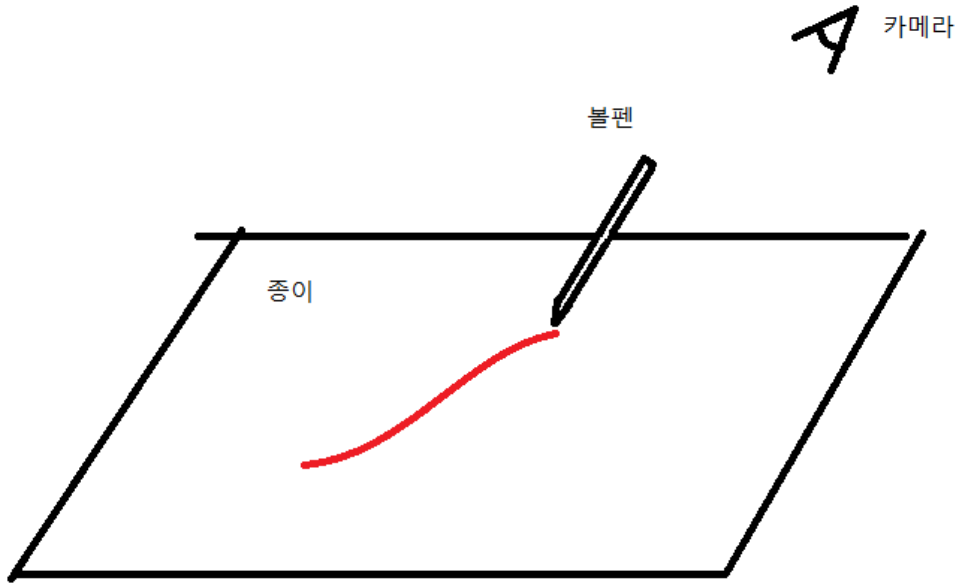
- 볼펜이 각각의 파트로 분해될 수 있도록, 볼펜 몸통, 볼펜 심(촉), 볼펜 뚜껑, 볼펜 버튼, 스프링으로 나뉘어 각각 모델링 된다.
- 볼펜이 움직일 때 모든 파트가 동시에 움직여야 한다. (결합 상태)
- 그리기 모드에서는 볼펜 촉이 종이와 맞닿아 있다. 볼펜 촉이 꺼내져 있지 않은 상태에서는 그림이 그려지지 않는다.



키보드/마우스 입력

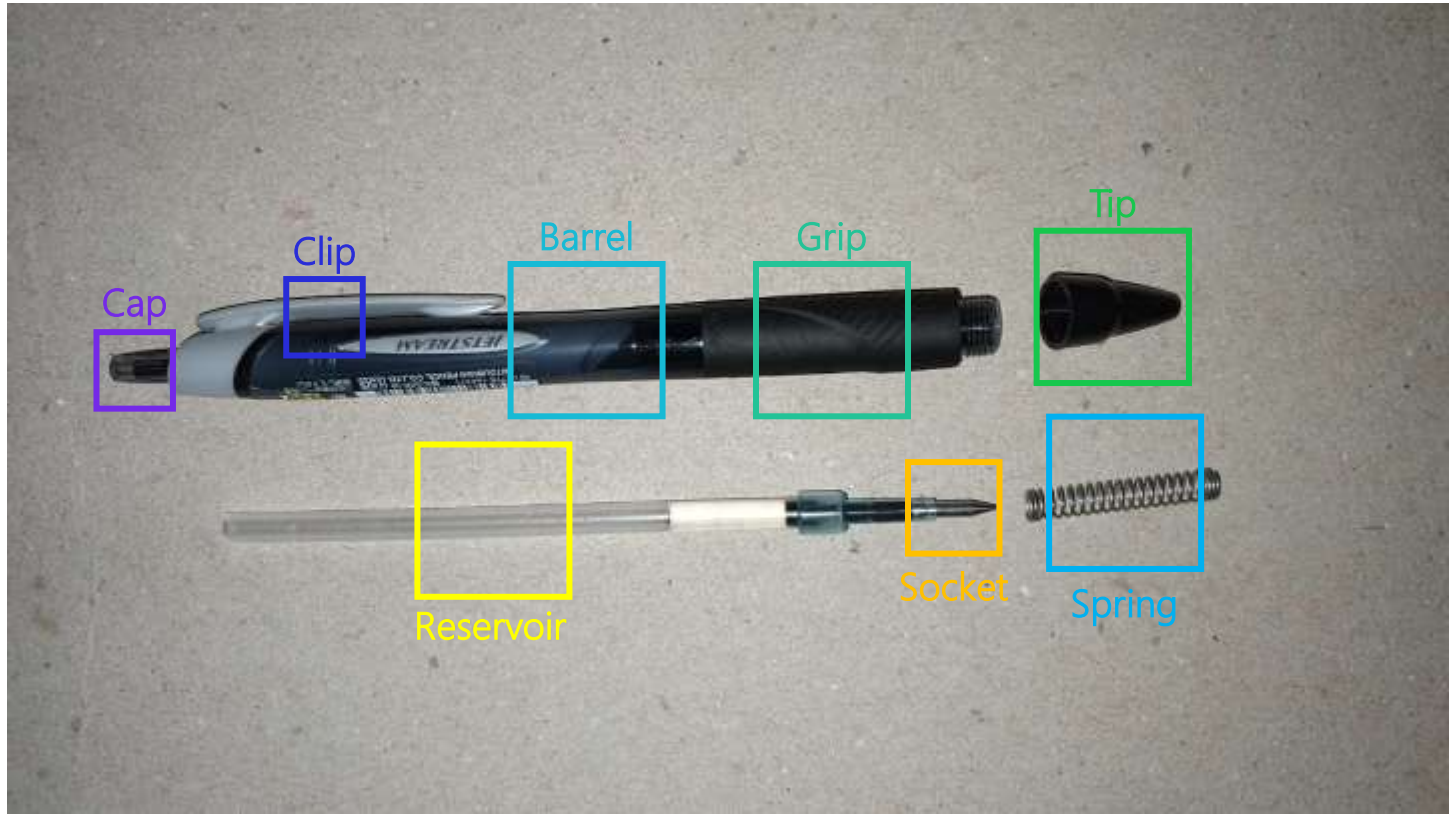
- 그리기 모드에서는 종이가 한 눈에 보이게끔 시점이 올라간다.
- 관찰 모드와 분해 모드는 볼펜이 잘 보이도록 시점이 확대된다. 분해 모드는 분해 애니메이션을 제공한다.
- 그림파일 저장은 프린터가 종이를 출력하는 듯한 애니메이션을 보여준다.
- 전체화면 상태에서 F11을 한 번 더 누르면 원래 크기로 돌아온다.

그리기 모드



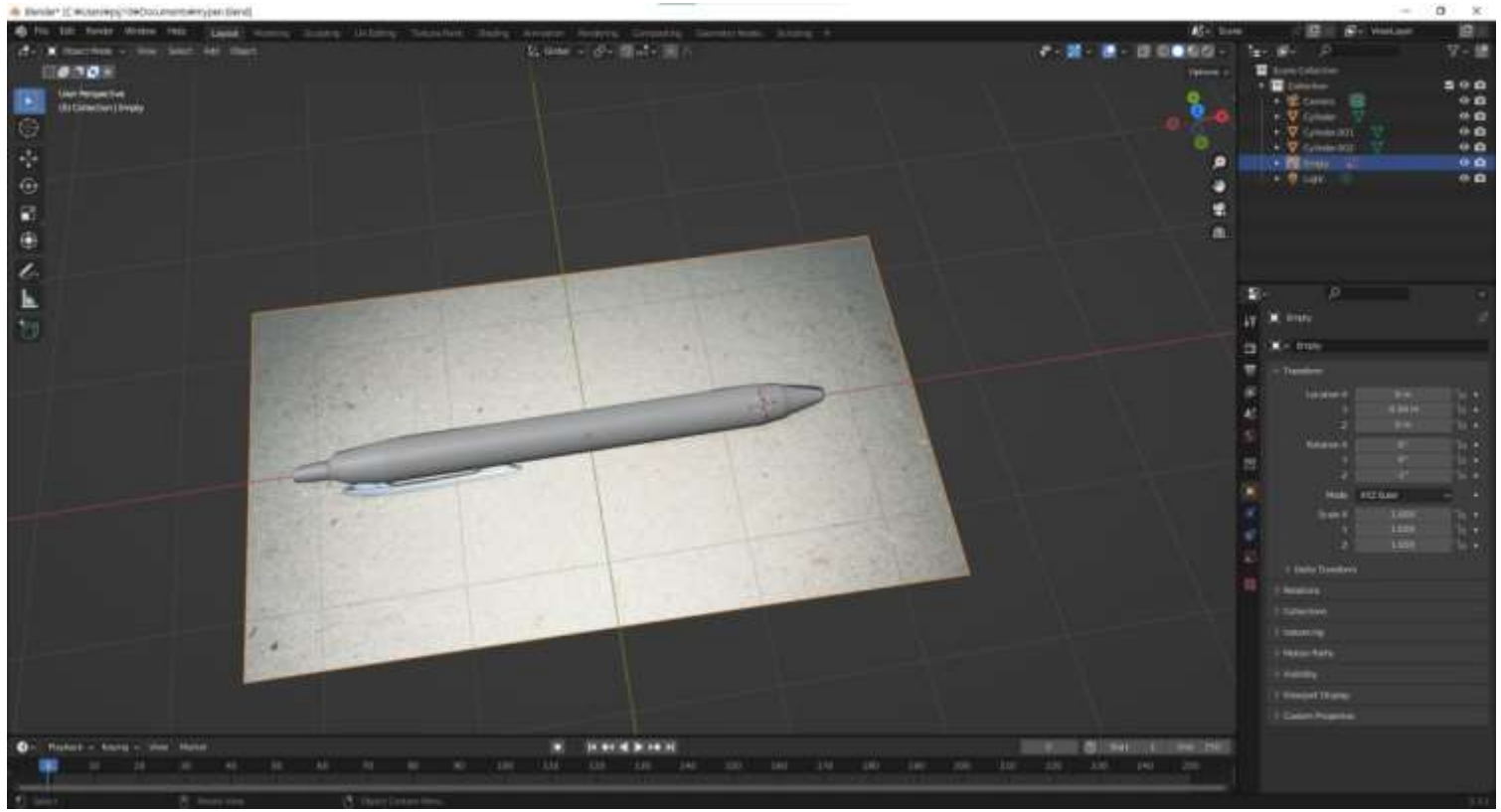
- 그리기 모드에서는 카메라 시점은 고정되고 볼펜이 마우스의 움직임을 따라한다.
- 마우스의 움직임은 볼펜과 함께 모델링 되는 종이에 획의 형태로 그려진다.
- 볼펜 색은 **검정**, 빨강, 파랑 3종류가 있다.

분해된 볼펜과 각 부분의 명칭



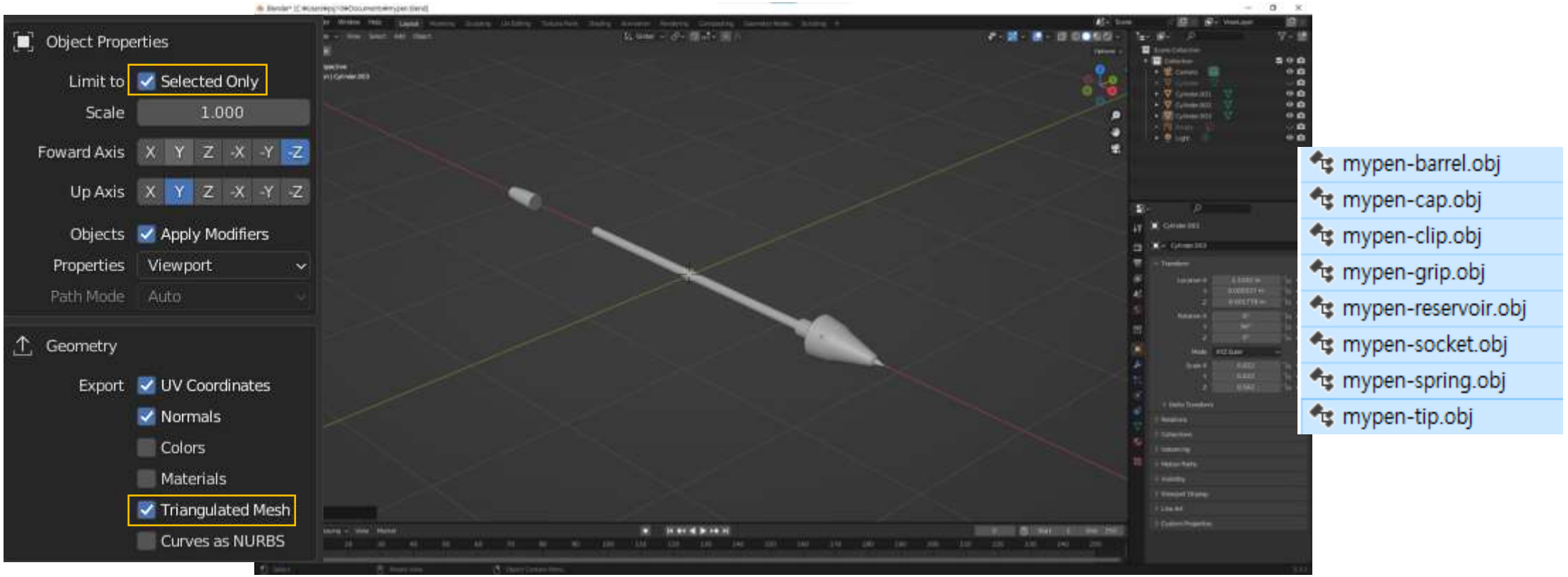
일반적으로 분해 가능한 부분까지 분해한 볼펜입니다

Blender 렌더링 과정



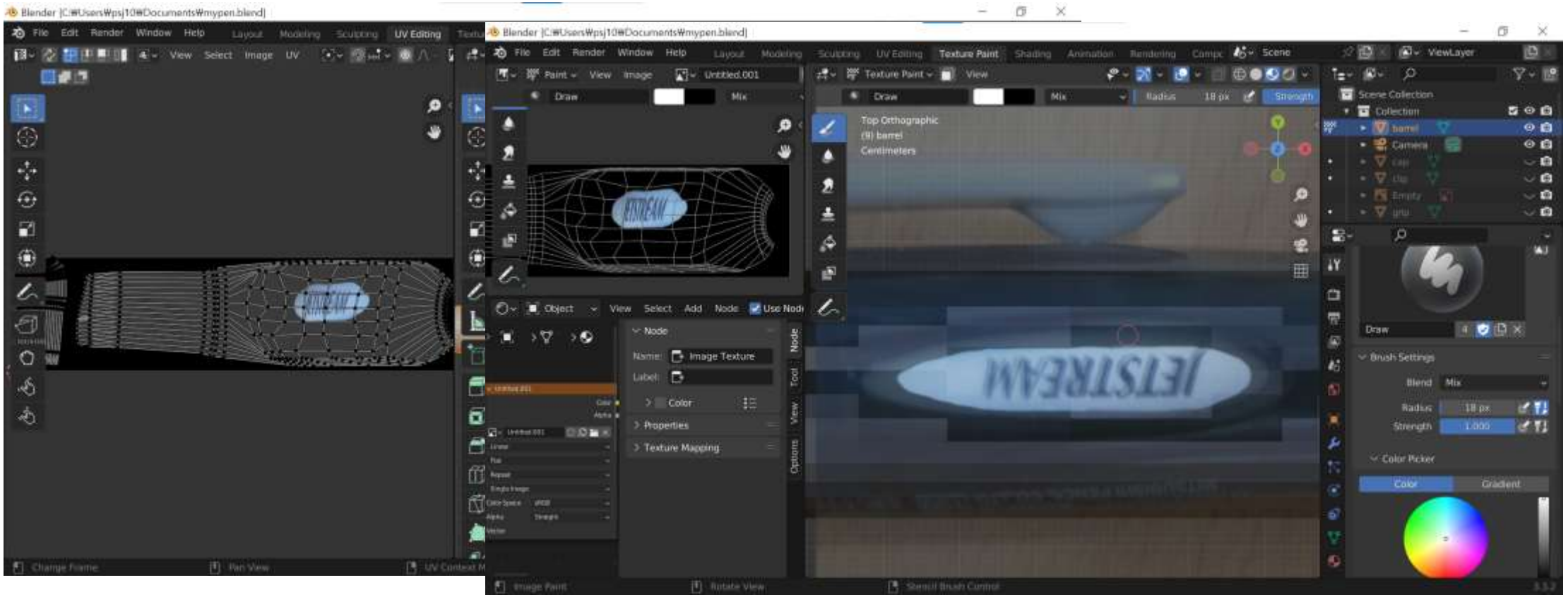
볼펜 이미지를 Blender에 올려 놓고 Cylinder을 이용해 모양을 만듦

Blender 렌더링 과정



볼펜의 각 부품 별로 따로 모델링하여 OpenGL 프로그램 상에서 개별적으로 조작 가능하게 함

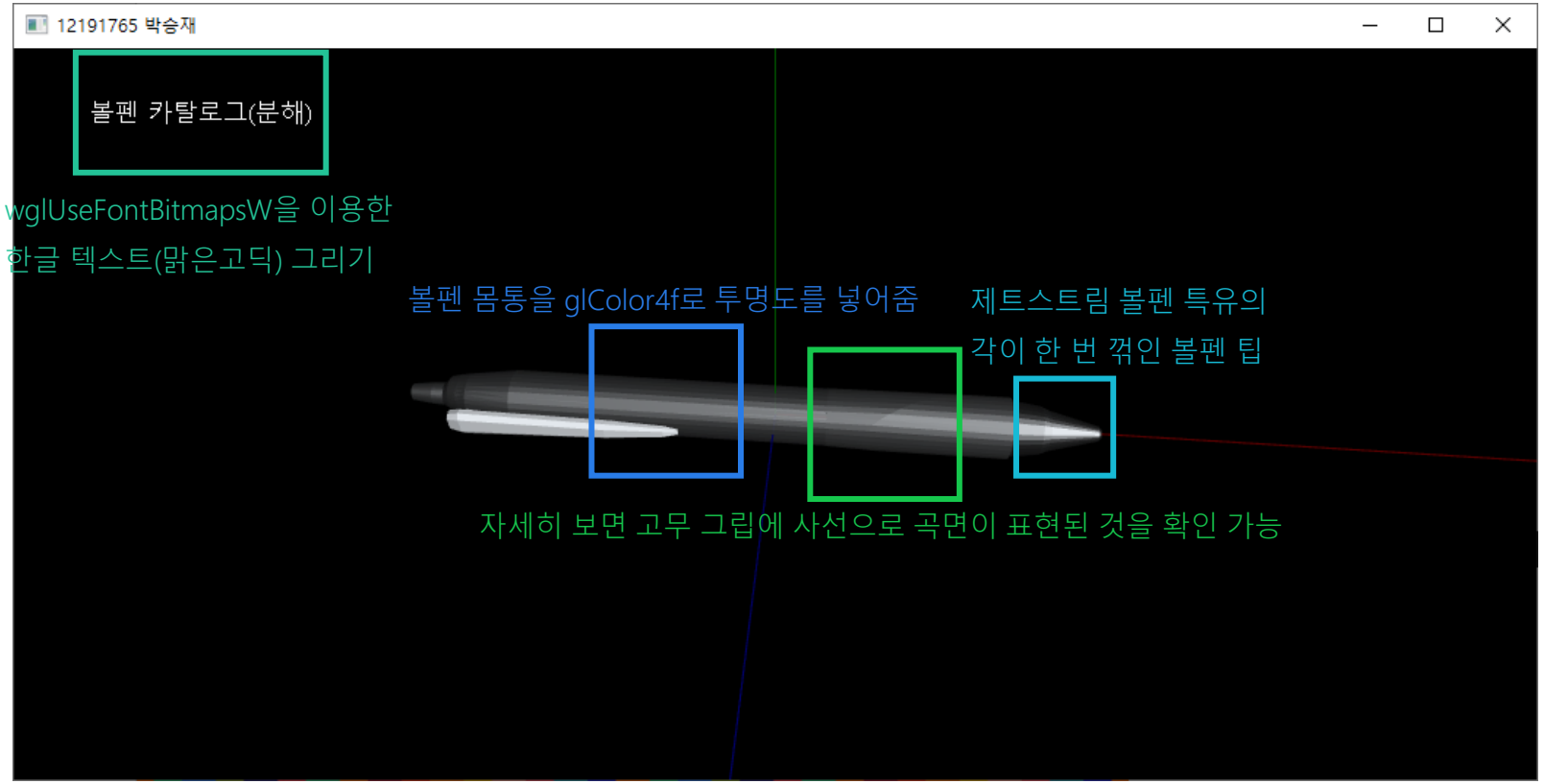
Blender UV 매핑 과정



스마트폰 카메라로 볼펜을 촬영하고 로고 부분의 텍스처를 매핑

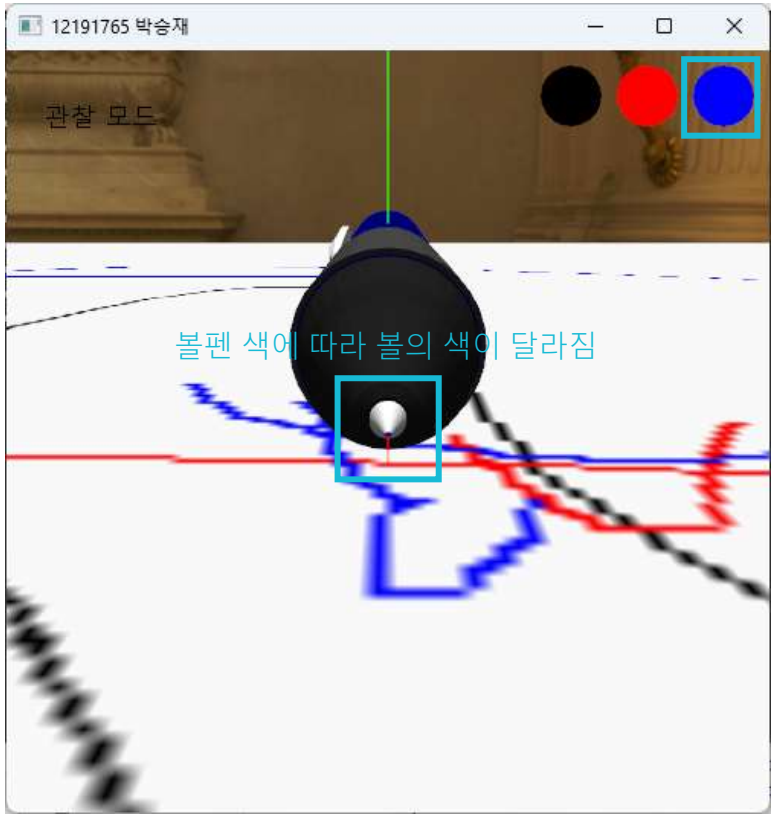
로고가 선명하게 보이게 하기 위해 UV 맵에서 로고 부분만 확대하여 상대적으로 높은 해상도의 텍스처가 들어가게 함

볼펜 모델 렌더링 테스트



각각의 .obj 파일을 불러와 개별적으로 색을 입힘
텍스처는 볼펜 몸통만 적용

볼펜 렌더링 테스트



.obj 불러오기

```
struct Vec2f {
    float x;
    float y;
};

struct Vec3f {
    float x;
    float y;
    float z;
};

class Object {
public:
    Object(const std::string& filename);
    ~Object() = default;
    void render() const;

private:
    std::string name;
    std::vector<Vec3f> vertices; // v
    std::vector<Vec2f> textures; // vt
    std::vector<Vec3f> normals; // vn
    std::vector<std::size_t> vertex_indices; // f
    std::vector<std::size_t> texture_indices; // f
    std::vector<std::size_t> normal_indices; // f
};
```

```
void Object::render() const {
    glBegin(GL_TRIANGLES);
    for (std::size_t n = 0; n < vertex_indices.size(); n += 3) {
        glTexCoord2f(textures[texture_indices[n] - 1].x, textures[texture_indices[n] -
1].y);
        glNormal3f(normals[normal_indices[n] - 1].x, normals[normal_indices[n] - 1].y,
normals[normal_indices[n] - 1].z);
        glVertex3f(vertices[vertex_indices[n] - 1].x, vertices[vertex_indices[n] - 1].y,
vertices[vertex_indices[n] - 1].z);

        glTexCoord2f(textures[texture_indices[n + 1] - 1].x, textures[texture_indices[n +
1] - 1].y);
        glNormal3f(normals[normal_indices[n + 1] - 1].x, normals[normal_indices[n + 1] -
1].y, normals[normal_indices[n + 1] - 1].z);
        glVertex3f(vertices[vertex_indices[n + 1] - 1].x, vertices[vertex_indices[n + 1]
- 1].y, vertices[vertex_indices[n + 1] - 1].z);

        glTexCoord2f(textures[texture_indices[n + 2] - 1].x, textures[texture_indices[n +
2] - 1].y);
        glNormal3f(normals[normal_indices[n + 2] - 1].x, normals[normal_indices[n + 2] -
1].y, normals[normal_indices[n + 2] - 1].z);
        glVertex3f(vertices[vertex_indices[n + 2] - 1].x, vertices[vertex_indices[n + 2]
- 1].y, vertices[vertex_indices[n + 2] - 1].z);
    }
    glEnd();
}
```

실습시간에 제공받은 ObjParser.h 코드를 참고해 객체지향적인 Object Parser와 Drawer를 코드를 작성

Bitmap 클래스

실습시간에 제공받은 bmpfunc.cpp 코드를 참고해
객체지향적인 Bitmap Parser를 코드를 작성함

```
class Bitmap {
public:
    Bitmap() = delete;
    Bitmap(int width, int height);
    Bitmap(const std::string& filename);
    ~Bitmap(); 소멸자를 이용한 메모리 회수
    void fill_pixel(int x, int y, const Color& color);
    int get_channels() const;
    int get_width() const;
    int get_height() const;
    unsigned char* get_pixels() const;
    void save(const std::string& out_filename, const
std::string& ref_filename) const;

private:
    int width;
    int height;
    int channels;
    unsigned char* pixels;
};
```

```
Bitmap::Bitmap(int width, int height)
: width(width), height(height), channels(3) {
    int size = width * height * 3;
    pixels = new unsigned char[size]; // image pixel data
    std::fill(pixels, pixels + size, 0xff);
}
```

흰 색 종이 텍스처를 위한 Bitmap(int width, int height);

```
Bitmap::Bitmap(const std::string& filename) {
    FILE* fp = fopen(filename.c_str(), "rb");
    if (!fp) {
        perror("File opening failed\n");
    }

    unsigned char header[54]; // bitmap header
    if (fread(header, sizeof(unsigned char), 54, fp) != 54 || header[0] != 'B' || header[1] != 'M') {
        perror("Invalid BMP file\n");
        fclose(fp);
    }

    unsigned int offset = *(unsigned int*) &(header[0x0A]);
    width = *(int*) &(header[0x12]);
    height = *(int*) &(header[0x16]);
    unsigned int size = *(unsigned int*) &(header[0x22]); // image size

    if (size == width * height) {
        channels = 1;
    } else {
        channels = 3;
    }
    size = width * height * channels;
    if (offset == 0) {
        offset = 54; // The BMP header is done that way
    }
    pixels = new unsigned char[size]; // image pixel data
    fread(pixels, sizeof(unsigned char), size, fp);
    fclose(fp);

    if (channels == 3) { // BGR order -> RGB order
        reverse_each_pixel(width, height, pixels);
    }
}
```

MyPen 클래스

```
class MyPen {
public:
    MyPen() = default;
    ~MyPen() = default;
    const Color& get_line_color() const;
    bool is_clicked() const;
    bool is_drawing_mode() const;
    void move(float x, float y);
    void disable_drawing_mode();
    void enable_drawing_mode();
    void perform_click();
    void perform_draw_line(Paper* paper, int x1,
int y1, int x2, int y2);
    void render(float disassembled = 0.0f) const;
    void set_line_color(const Color& color);

    private:
    ...

    float x = 0.0f;
    float y = 0.0f;
    bool clicked = false;
    bool drawing = false;
    Color line_color = { 0.0f, 0.0f, 0.0f };
};
```

float disassembled는 볼펜 분해 애니메이션을 보여줄 때 사용하는 계수

조립 상태일 때는 disassembled=-1.0

볼펜 색에 따라 볼펜 촉 끝의 색이 달라지도록 glColor3f와 glutSolidSphere으로 구현

```
void MyPen::render(float animated) const {
    if (drawing) {
        glTranslatef(x, 1.89f, y);
        glRotatef(-90.0f, 0.0f, 0.0f, 1.0f);
    } else {
        glTranslatef(x, 0.5f, y);
    }

    glTranslatef(-1.5f * animated, 0.0f, 0.0f);
    pen_barrel.render();
    pen_clip.render();
    pen_grip.render();

    if (!clicked) {
        glTranslatef(-0.1f, 0.0f, 0.0f);
    }
    glPushMatrix();
    glTranslatef(-0.5f * animated, 0.0f, 0.0f);
    pen_cap.render();
    glPopMatrix();

    glTranslatef(1.0f * animated, 0.0f, 0.0f);
    pen_reservoir.render();
    pen_socket.render();

    glTranslatef(0.75f * animated, 0.0f, 0.0f);
    pen_spring.render();

    glColor3f(line_color.red, line_color.green, line_color.blue);
    glPushMatrix();
    glTranslatef(1.88f, 0.0f, 0.0f);
    glutSolidSphere(0.004, 10, 10); // ball
    glPopMatrix();

    if (!clicked) {
        glTranslatef(0.1f, 0.0f, 0.0f);
    }
    glTranslatef(0.75f * animated, 0.0f, 0.0f);
    pen_tip.render();
}
```

볼펜 그리기 모드

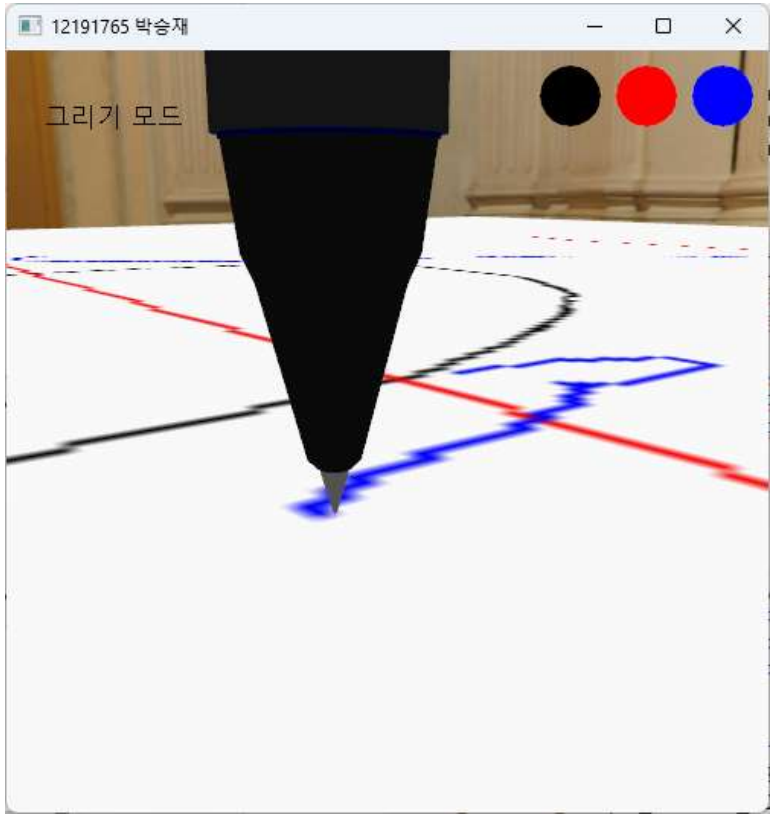
- 키보드 1을 누르면 카메라 시점이 올라가면서 종이가 한 눈에 보임
- 키보드 스페이스바를 누르면 효과음과 함께 펜촉이 튀어나오면서 마우스를 드래그 하는대로 선이 그려짐 (Bresenham 알고리즘 이용)
- 우측 상단의 원을 클릭하면 볼펜 색이 바뀜
- 마우스 우클릭 > Clear로 그림을 지울 수 있음

볼펜 그리기 모드

```
void motion_cb(int x, int y) {
    float pen_x = (x * paper->get_width() / window_width) - (paper->get_width() / 2);
    float pen_y = (y * paper->get_height() / window_height) - (paper->get_height() / 2);
    mypen->move(pen_x, pen_y);
    int paper_x = x * paper->get_image_width() / window_width;
    int paper_y = paper->get_image_height() - (y * paper->get_image_height() /
window_height);
    if (prev_paper_x >= 0 && prev_paper_y >= 0 && mypen->is_clicked() && mypen-
>is_drawing_mode()) {
        mypen->perform_draw_line(paper, paper_x, paper_y, prev_paper_x, prev_paper_y);
        paper->update_texture();
    }
    prev_paper_x = paper_x;
    prev_paper_y = paper_y;
}
```

Bresenham 알고리즘 이용해 개선

```
void Bitmap::fill_pixel(int x, int y, const Color& color) {
    pixels[channels * (y * width + x)] = color.red * 255;
    pixels[channels * (y * width + x) + 1] = color.green * 255;
    pixels[channels * (y * width + x) + 2] = color.blue * 255;
}

void Paper::update_texture() const {
    glBindTexture(GL_TEXTURE_2D, texture);
    glTexImage2D(GL_TEXTURE_2D, 0, 3, image->get_width(), image->get_height(), 0,
GL_RGB, GL_UNSIGNED_BYTE, image->get_pixels());
}
```



마우스를 드래그 하는대로 종이에 선을 그릴 수 있습니다.

볼펜 그리기 모드 – 색상 선택

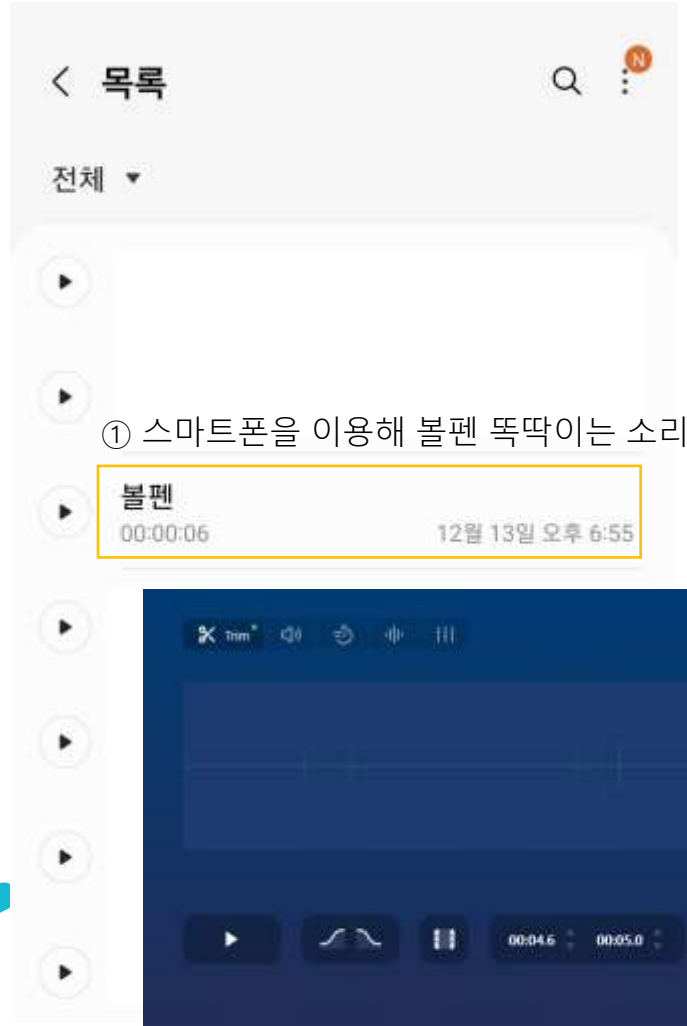
```
void pick(int x, int y) {
    int viewport_width = 160;
    int viewport_height = 60;
    unsigned int select_buf[256];
    glSelectBuffer(256, select_buf);
    int viewport[4] = { window_width - viewport_width, 0, viewport_width, viewport_height }; // (x, y, width, height)
    glRenderMode(GL_SELECT);
    glMatrixMode(GL_PROJECTION);
    glPushMatrix();
    glLoadIdentity();
    gluPickMatrix(x, y, 0.1, 0.1, viewport);
    glOrtho(-viewport_width / 2, viewport_width / 2, -viewport_height / 2, viewport_height / 2, -50.0, 50.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    draw_color_spheres();
    glMatrixMode(GL_PROJECTION);
    glPopMatrix();
    glFlush();

    int hits = glRenderMode(GL_RENDER);
    if (hits >= 1) {
        unsigned int idx = 0;
        unsigned int selected_z = -1; // max value of unsigned int(overflow)
        sphere_selected = -1;
        for (int i = 1; i <= hits; i += 1) {
            unsigned int name_count = select_buf[idx]; // always 1
            unsigned int z_min = select_buf[idx + 1];
            unsigned int z_max = select_buf[idx + 2];
            unsigned int name = select_buf[idx + 3];
            idx += name_count + 3;
            if (selected_z > z_max) {
                selected_z = z_max;
                sphere_selected = name;
            }
        }
    }
}
```

```
    // mini viewport
    int viewport_width = 160;
    int viewport_height = 60;
    glViewport(window_width - viewport_width,
window_height - viewport_height, viewport_width,
viewport_height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-viewport_width / 2, viewport_width / 2, -
viewport_height / 2, viewport_height / 2, -50.0, 50.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glDisable(GL_LIGHTING);
    draw_color_spheres();
    glEnable(GL_LIGHTING);
```

Multi-viewport를 이용해 색상 선택 원을 그렸기 때문에,

glGetIntegerv(GL_VIEWPORT, viewport) 대신 viewport 크기를 따로 지정해야 함

볼펜 클릭 효과음 출력



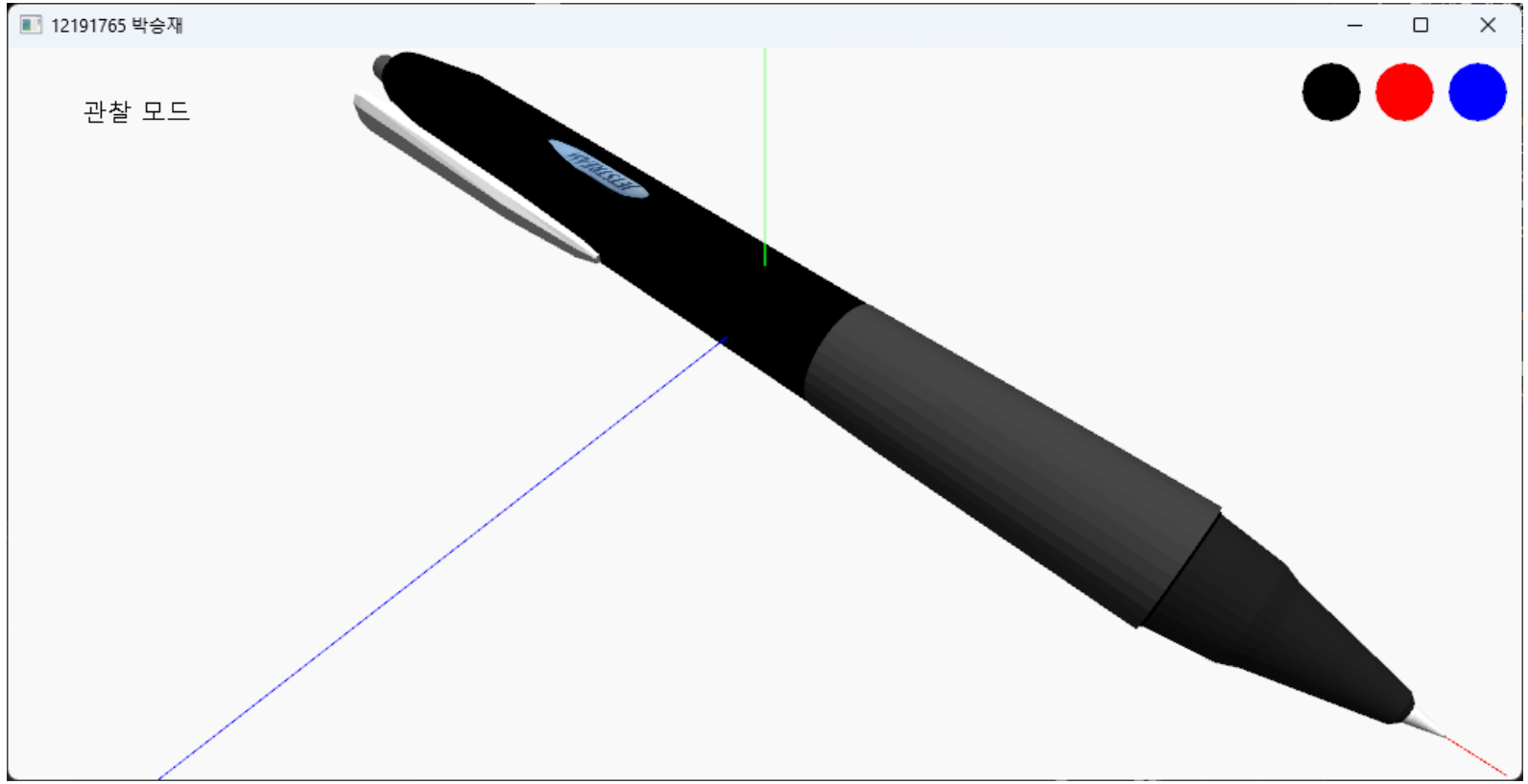
① 스마트폰을 이용해 볼펜 똑딱이는 소리를 녹음

② https://mp3cut.net/ko/ 에서 원하는 부분만 잘라내고 wav로 변환

③ Spacebar를 누를 때마다 효과음 출력

```
#include <mmsystem.h>

mypen->perform_click();
PlaySound(TEXT("res/click.wav"), NULL, SND_ASYNC);
```

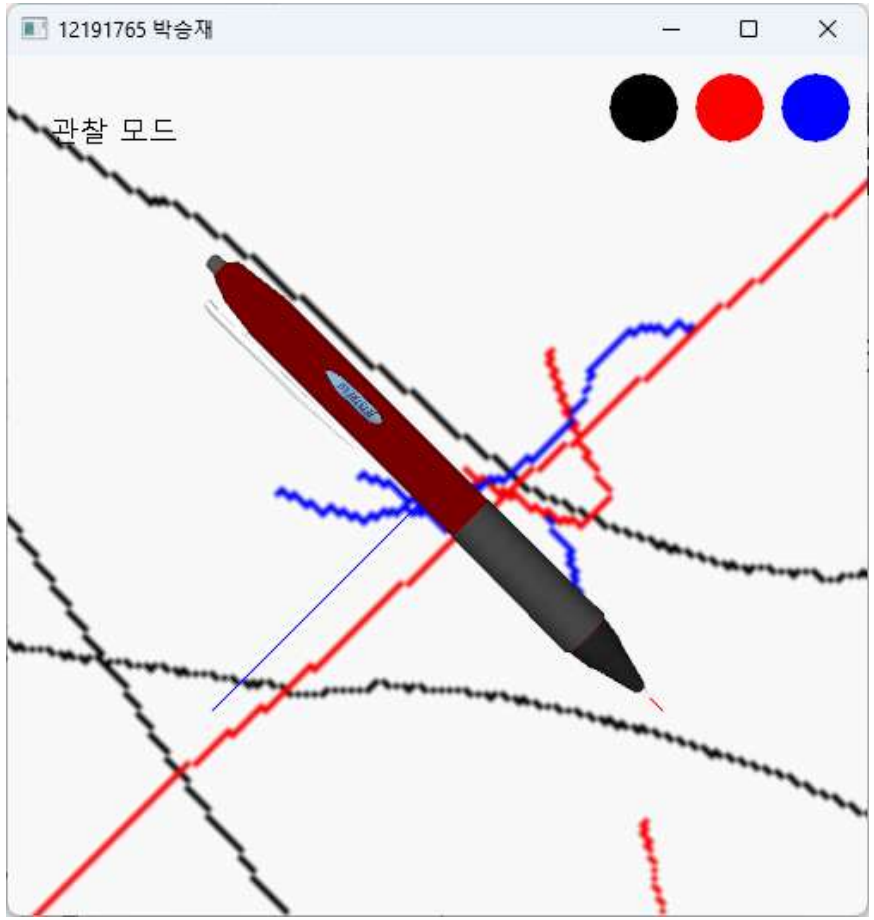
볼펜 관찰 모드

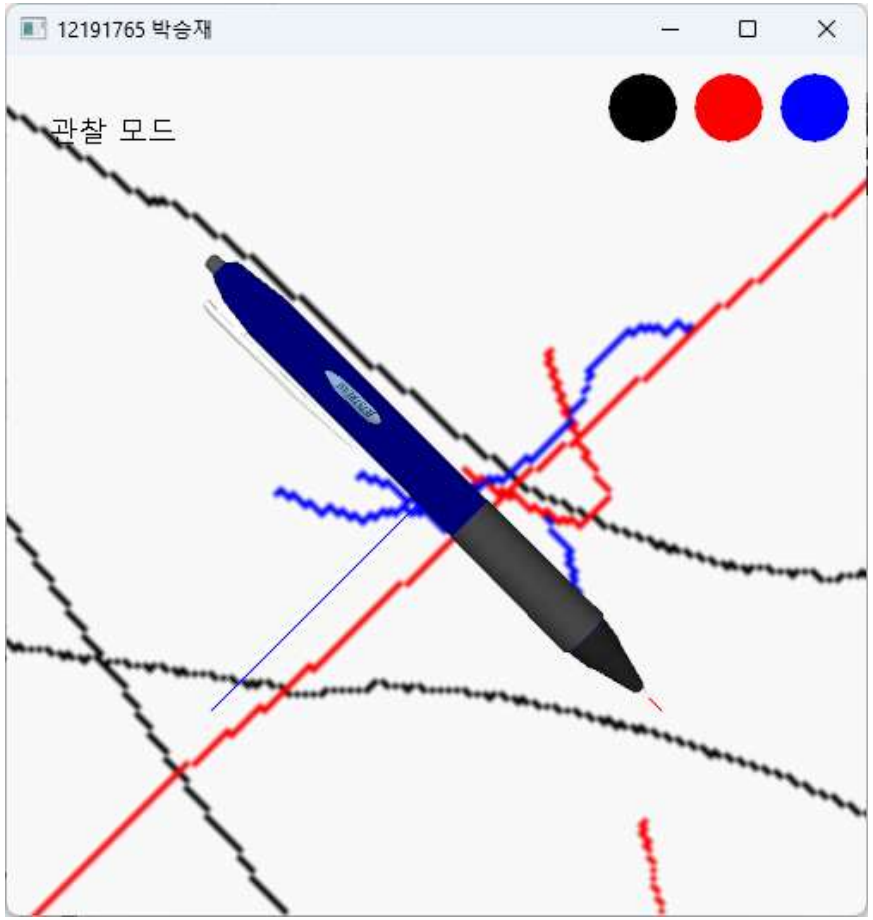


키보드 2를 누르면 카메라가 확대되고 카메라를 이동하며 볼펜을 관찰할 수 있음

볼펜 관찰 모드

볼펜 색이 바뀌면 볼펜 몸통색도 바뀜





볼펜 관찰 모드 – 텍스처 색상 변경

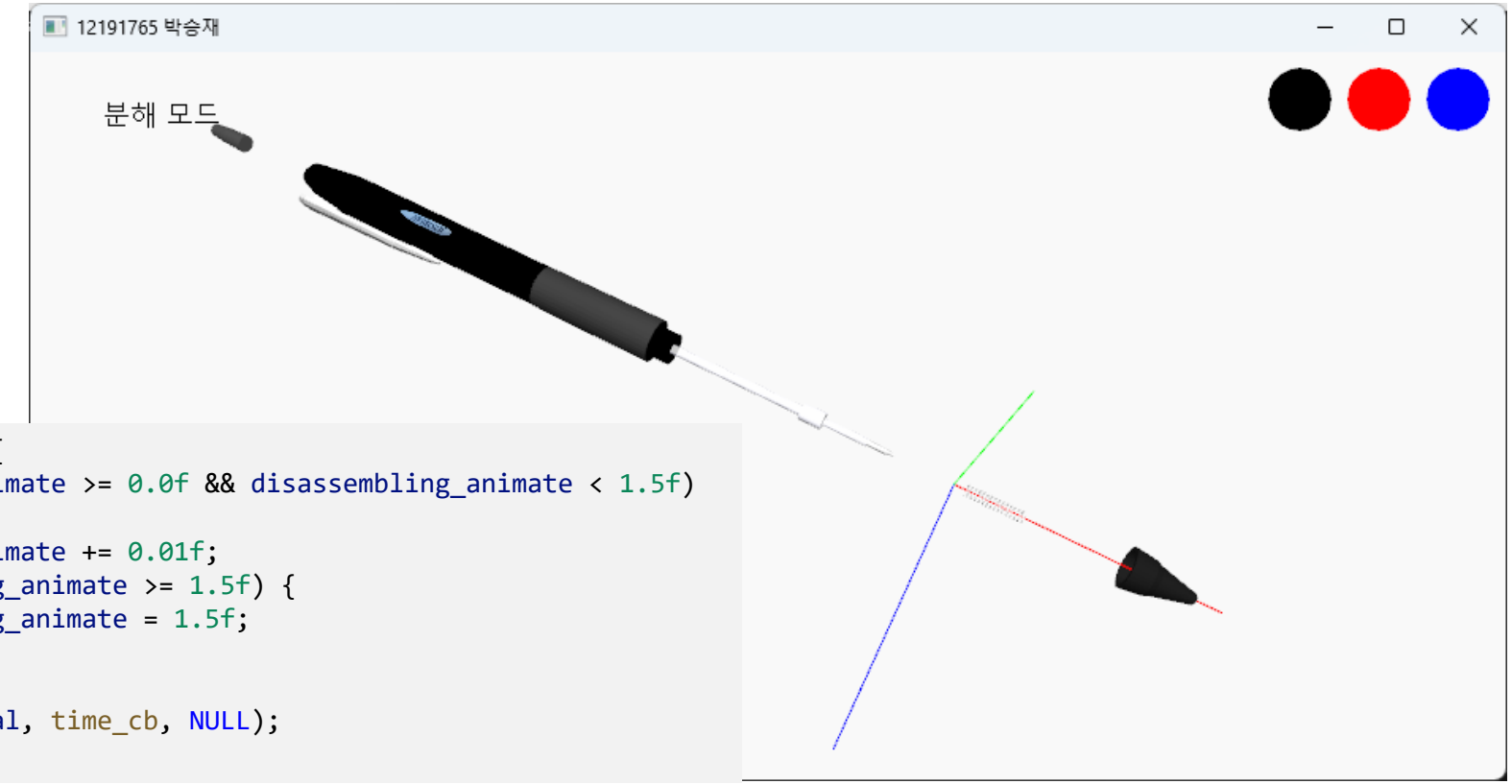
```
void MyPen::Barrel::set_color(const Color& color) {
    int width = image->get_width();
    int height = image->get_height();
    int channels = image->get_channels();
    unsigned char* pixels = new unsigned char[width * height * channels];
    std::memcpy(pixels, image->get_pixels(), width * height * channels);
    for (int x = 0; x < width; x += 1) {
        for (int y = 0; y < height; y += 1) {
            if (pixels[channels * (y * width + x)] == 0 && pixels[channels * (y * width +
x) + 1] == 0 && pixels[channels * (y * width + x) + 2] == 0) {
                pixels[channels * (y * width + x)] = color.red * 120;
                pixels[channels * (y * width + x) + 1] = color.green * 120;
                pixels[channels * (y * width + x) + 2] = color.blue * 120;
            }
        }
    }
    glBindTexture(GL_TEXTURE_2D, texture);
    glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, pixels);
    delete[] pixels;
}
```

mypen-barrel.bmp

볼펜 몸통 텍스처의 검은 부분을 원하는 색으로 치환하면 볼펜 몸통의 색을 바꿀 수 있음

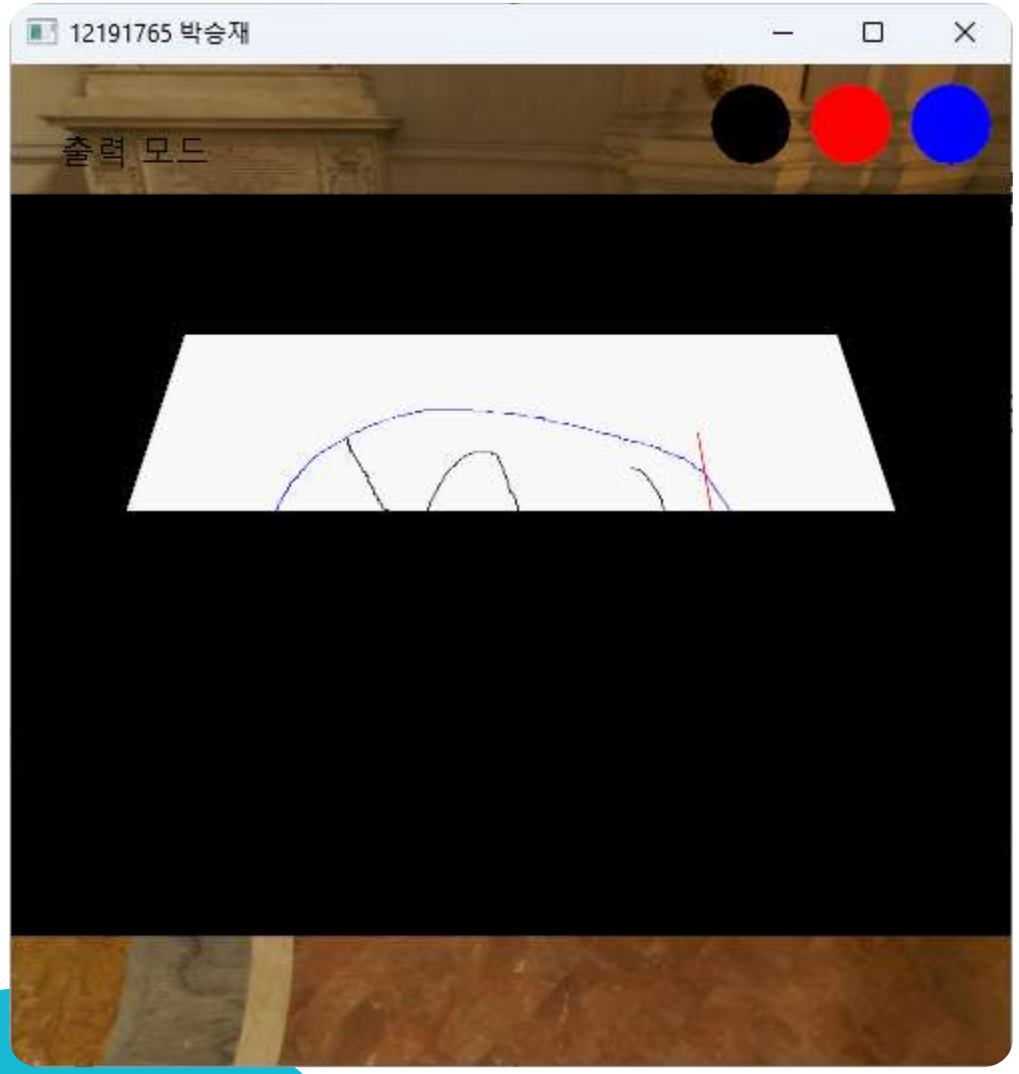
하나의 텍스처로 다양한 볼펜 몸통 색을 표현 가능

볼펜 분해 모드



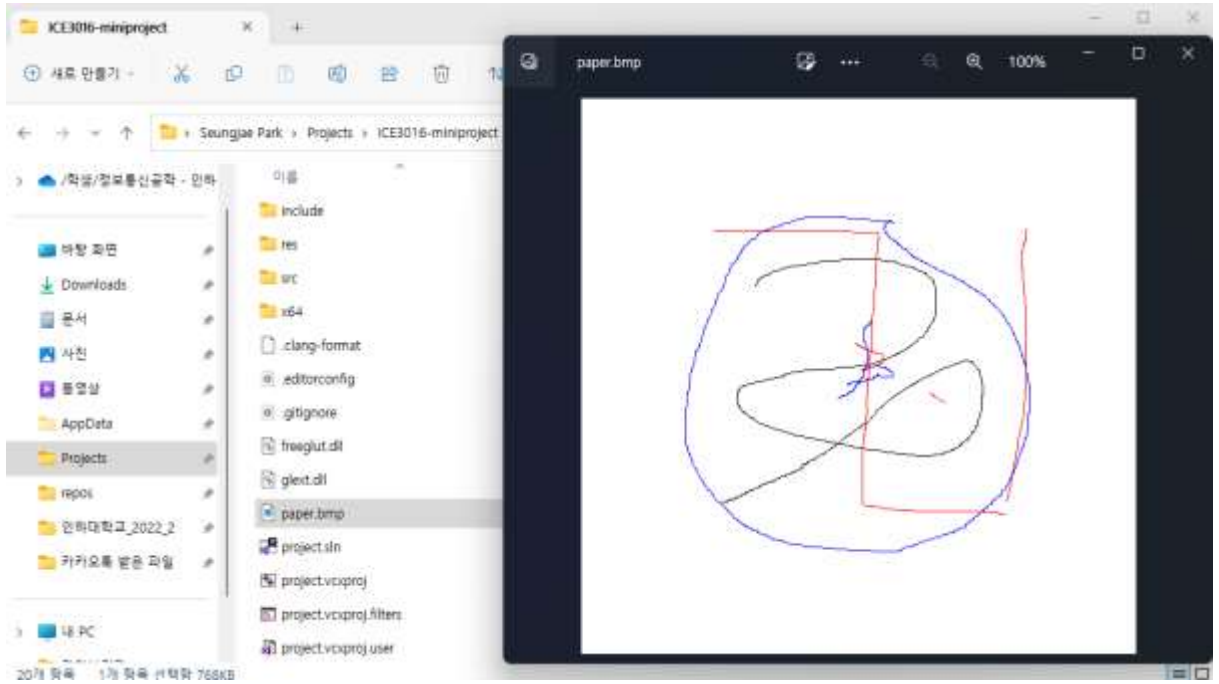

```
void time_cb(int value) {
    if (disassembling_animate >= 0.0f && disassembling_animate < 1.5f)
{
        disassembling_animate += 0.01f;
        if (disassembling_animate >= 1.5f) {
            disassembling_animate = 1.5f;
        }
    }
    glutTimerFunc(interval, time_cb, NULL);
}
```

키보드 3를 누르면 카메라가 확대되고 볼펜이 분해되는 애니메이션이 연출



그림파일 출력 모드

- F2를 누르면 프린터로 종이를 인쇄하는 듯한 애니메이션이 보여지며 파일이 저장됨



그림파일 출력 모드

```cpp
// 다른 .bmp 파일 헤더를 복사해 새로운 BMP 파일을 만듭니다.
void Bitmap::save(const std::string& out_filename, const std::string& ref_filename) const {
    FILE* ifp = fopen(ref_filename.c_str(), "rb");
    if (!ifp) {
        perror("File opening failed\n");
        return;
    }
    fseek(ifp, 10, SEEK_SET);

    int ref_offset;
    fread(&ref_offset, sizeof(int), 1, ifp);

    fseek(ifp, 18, SEEK_SET);

    int ref_width;
    int ref_height;
    fread(&ref_width, sizeof(int), 1, ifp);
    fread(&ref_height, sizeof(int), 1, ifp);

    fseek(ifp, 0, SEEK_SET);

    unsigned char* ref_header = new unsigned char[ref_offset];
    fread(ref_header, sizeof(unsigned char), ref_offset, ifp);
    fclose(ifp);

    FILE* ofp = fopen(out_filename.c_str(), "wb");
    if (!ofp) {
        perror("File opening failed\n");
        return;
    }
    fwrite(ref_header, sizeof(unsigned char), ref_offset, ofp);
    delete[] ref_header;

    int size = channels * width * height;
    unsigned char* out_pixels = new unsigned char[size];
    std::copy(pixels, pixels + size, out_pixels);
    if (channels == 3) {
        reverse_each_pixel(width, height, out_pixels);
    }
    fwrite(out_pixels, sizeof(unsigned char), size, ofp);
    delete[] out_pixels;

    fclose(ofp);
}
```

```cpp
// main viewport
glViewport(0, 0, window_width, window_height);

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
if (printing_animate < 0.0f) {
    gluPerspective(45.0, (double) window_width / window_height, 0.1, 100.0);
} else {
    gluPerspective(45.0, (double) window_width / window_height, printing_animate, 50.0);
}
```

Z near를 조절해 프린트 애니메이션 효과를 구현

실습시간에 제공받은 bmpfunc.cpp 코드를 참고해

종이에 적용된 텍스처를 BMP 파일로 저장하는 기능 구현

전체화면 모드

```
void toggle_fullscreen() {
    fullscreen = !fullscreen;
    if (fullscreen) {
        prev_window_width = glutGet(GLUT_WINDOW_WIDTH);
        prev_window_height = glutGet(GLUT_WINDOW_HEIGHT);
        glutFullScreen();
    } else {
        glutReshapeWindow(prev_window_width, prev_window_height);
    }
}

void special_keyboard_cb(int key, int x, int y) {
    switch (key) {
    case GLUT_KEY_F11:
        std::cout << "Toggle fullscreen\n";
        toggle_fullscreen();
        break;
    }
}
```

glutFullScreen을 이용해 화면을 전체화면 모드로 변경

이전 윈도우 크기를 저장해두고 F11를 한 번 더 누르면 원래 크기로 되돌아옴

한글 텍스트 그리기

전역변수 선언

```
#include <Windows.h>

HDC hdc; // handle display context
```

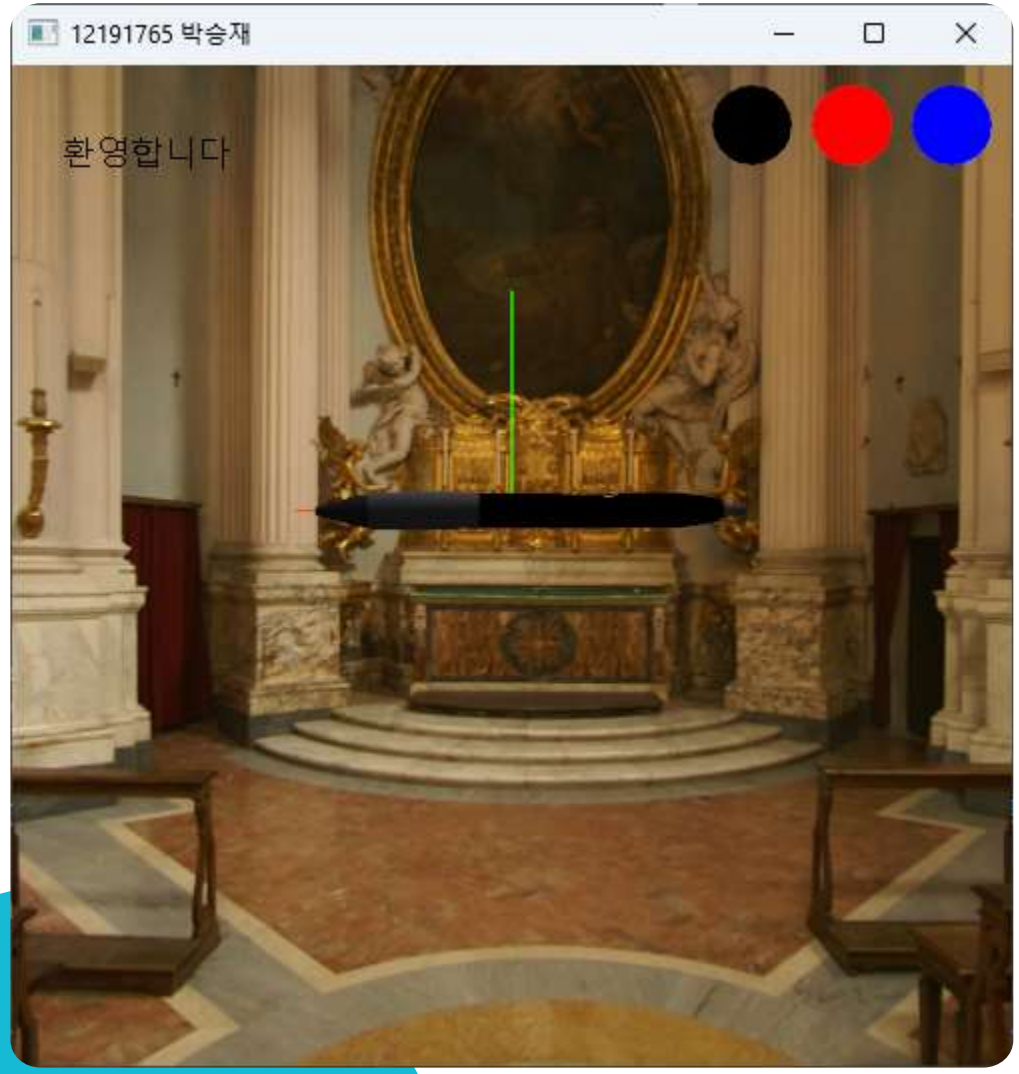
init 함수에서 초기화

```
void init_font(HDC& hdc) {
    hdc = wglGetCurrentDC();            글자 굵기 중간                    한글 charset
    HFONT font = CreateFont(24, 0, 0, 0, FW_NORMAL, FALSE, FALSE, 0, HANGUL_CHARSET,
OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY, DEFAULT_PITCH | FF_SWISS,
TEXT("맑은 고딕"));한글 폰트
    SelectObject(hdc, font);
}
```

draw_string에서 글자를 비트맵으로 변환하고 glCallList로 화면에 그림

```
void draw_text(const HDC& hdc, const std::wstring& text) {
    glDisable(GL_LIGHTING);  한글이기 때문에 wide character 사용
    glMatrixMode(GL_PROJECTION);
    glPushMatrix();
    glLoadIdentity();
    gluOrtho2D(0.0, 10.0, 10.0, 0.0);

    glMatrixMode(GL_MODELVIEW);
    glPushMatrix();
    glLoadIdentity();

    glRasterPos3f(0.5f, 1.0f, 0.0f);
    for (int i = 0; i < text.size(); i += 1) {
        int list = glGenLists(1);
        wglUseFontBitmapsW(hdc, text[i], 1, list);
        glCallList(list);
        glDeleteLists(list, 1);
    }
    glPopMatrix();

    glMatrixMode(GL_PROJECTION);
    glPopMatrix();

    glMatrixMode(GL_MODELVIEW);
    glEnable(GL_LIGHTING);
}
```

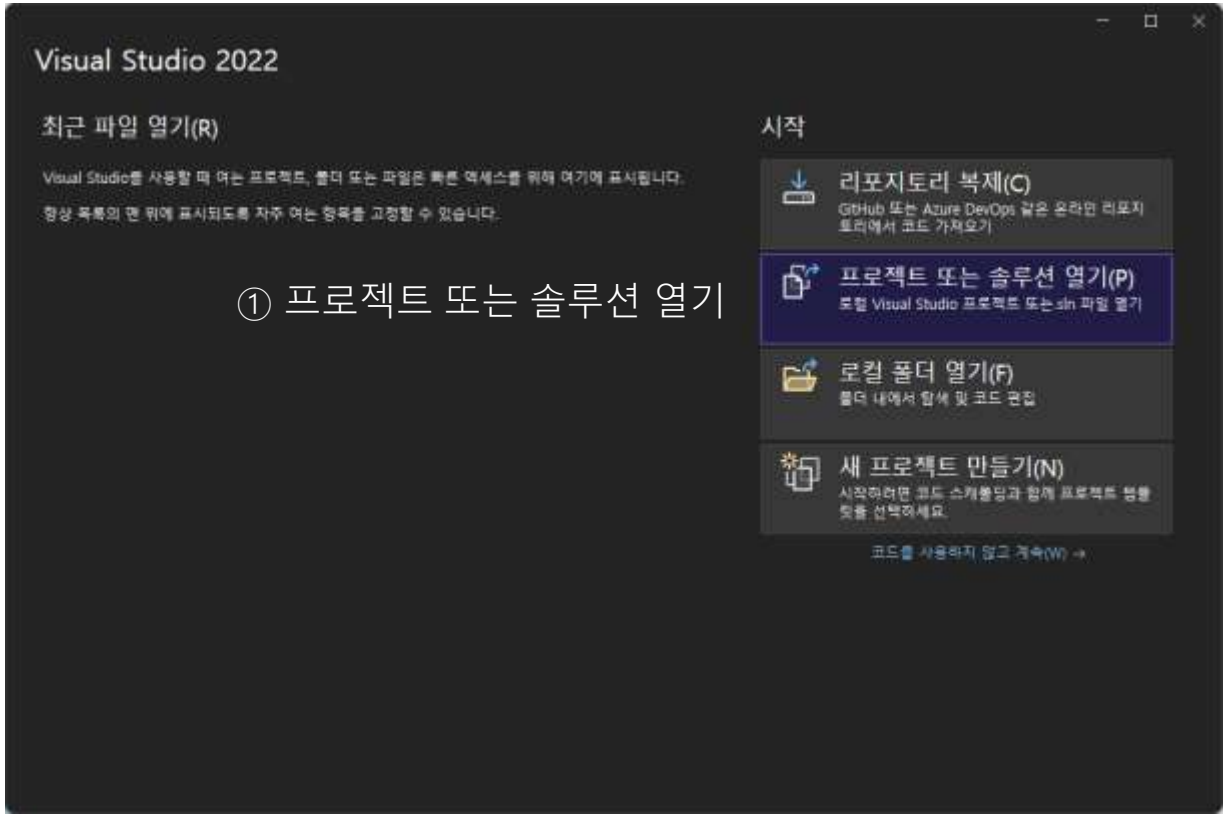
https://www.humus.name/index.php?page=Textures&ID=110



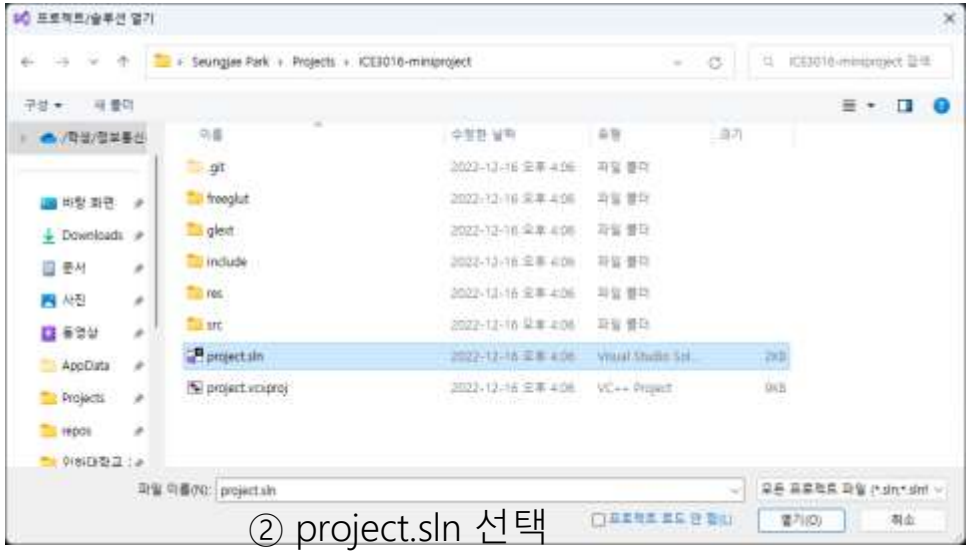
큐브맵 배경

```
CubeMap::CubeMap(float size)
: size(size) {
    glGenTextures(1, &texture);
    glEnable(GL_TEXTURE_CUBE_MAP);
    glBindTexture(GL_TEXTURE_CUBE_MAP, texture);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);

    Bitmap image_px{ "res/cubemap/px.bmp" };
    glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X, 0, GL_RGBA, image_px.get_width(),
image_px.get_height(), 0, GL_RGB, GL_UNSIGNED_BYTE, image_px.get_pixels());
    Bitmap image_nx{ "res/cubemap/nx.bmp" };
    glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X, 0, GL_RGBA, image_nx.get_width(),
image_nx.get_height(), 0, GL_RGB, GL_UNSIGNED_BYTE, image_nx.get_pixels());
    Bitmap image_py{ "res/cubemap/py.bmp" };
    glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y, 0, GL_RGBA, image_py.get_width(),
image_py.get_height(), 0, GL_RGB, GL_UNSIGNED_BYTE, image_py.get_pixels());
    Bitmap image_ny{ "res/cubemap/ny.bmp" };
    glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y, 0, GL_RGBA, image_ny.get_width(),
image_ny.get_height(), 0, GL_RGB, GL_UNSIGNED_BYTE, image_ny.get_pixels());
    Bitmap image_pz{ "res/cubemap/pz.bmp" };
    glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Z, 0, GL_RGBA, image_pz.get_width(),
image_pz.get_height(), 0, GL_RGB, GL_UNSIGNED_BYTE, image_pz.get_pixels());
    Bitmap image_nz{ "res/cubemap/nz.bmp" };
    glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z, 0, GL_RGBA, image_nz.get_width(),
image_nz.get_height(), 0, GL_RGB, GL_UNSIGNED_BYTE, image_nz.get_pixels());

    glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
    glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
    glTexGeni(GL_R, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
}
```

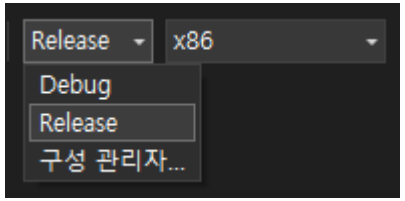
프로젝트 실행 참고사항



① 프로젝트 또는 솔루션 열기



② project.sln 선택



③ Release x86 선택

④ Ctrl+5로 실행

프로젝트 실행 참고사항

freeglut	2022-12-16 오후 4:06	파일 폴더	
glext	2022-12-16 오후 4:06	파일 폴더	
include	2022-12-16 오후 4:06	파일 폴더	
Release	2022-12-16 오후 4:08	파일 폴더	
res	2022-12-16 오후 4:06	파일 폴더	
src	2022-12-16 오후 4:06	파일 폴더	
.clang-format	2022-12-16 오후 4:06	CLANG-FORMAT ...	3KB
.editorconfig	2022-12-16 오후 4:06	Editor Config 원...	1KB
.gitignore	2022-12-16 오후 4:06	Git Ignore 원본 파...	7KB
paper.bmp	2022-12-16 오후 4:09	BMP 파일	769KB
project.sln	2022-12-16 오후 4:06	Visual Studio Sol...	2KB
project.vcxproj	2022-12-16 오후 4:06	VC++ Project	9KB
project.vcxproj.filters	2022-12-16 오후 4:06	VC++ Project Filt...	2KB
project.vcxproj.user	2022-12-16 오후 4:06	Per-User Project ...	2KB

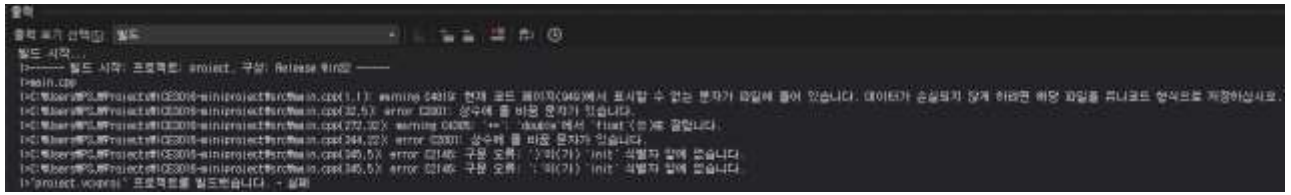
⑤ F2로 저장한 그림파일은 프로젝트 폴더 paper.bmp으로 저장됨

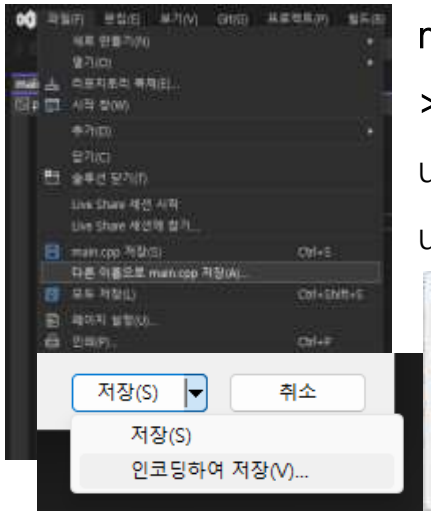
ⓐ *.dll이 없어 코드 실행을 진행할 수 없습니다.

freeglut/bin과 glext/bin의 dll 파일을 프로젝트 루트로 복사합니다.

project.vcxproj.user에 환경변수로 지정해 뒀는데 Visual Studio 버전이 달라 발생하는 문제로 추정됩니다.

ⓑ 현재 코드 페이지(949)에서 표시할 수 없는 문자가 파일에 들어 있습니다.





main.cpp 파일 열기 > 다른 이름으로 저장 > 인코딩하여 저장 > 유니코드(서명이 있는 UTF-8)

utf-8-bom 인코딩으로 main.cpp를 저장했는데 utf-8이나 euc-kr으로 인식된 것으로 추정됩니다.