컴퓨터그래픽스

컴퓨터그래픽스
기말고사 대체과제 발표

- 2D Everyday Scene Simulation -

ICT융합공학부
202204152 추예진

목차

프로젝트 개요

프로젝트 이름

: 2D Everyday Scene Simulation

주요 특징

- 직교 투영 기반의 2D 장면: 복잡한 3D 투영 없이 2D 공간에서 효과적인 표현.
- 반복적 배치: 배열과 반복문을 활용해 물체를 효율적으로 배치.
- 셰이더 기반 렌더링: 각 물체에 적합한 색상을 셰이더를 통해 표현.

개발 목표

- OpenGL을 활용한 간단한 2D 그래픽 렌더링 구현.
- 도로, 나무, 가로등, 구름 등 일상적인 장면 구성 요소의 표현.
- OpenGL 기본 지식 습득: 셰이더 프로그래밍, 뷰포트 설정, 반복적 렌더링.

▼

생활 주변의 물체를 단순화시켜 모델링해서 장면을 구성하고

해당 장면에 모델 변환, 시점 변환, 투상 변환이 가능하도록 사용자 인터페이스를 구현

개발 환경 및 기술 스택

개발 환경

OS

IDE

빌드 도구

기술 스택

- OpenGL: 그래픽 렌더링.
- GLFW: 윈도우 생성 및 입력 처리.
- GLAD: OpenGL 확장 로더.
- GLM: 행렬 및 벡터 연산 라이브러리.
- C++: 프로젝트 구현 언어.

기술적 요소

1.셰이더 코드:

- Vertex Shader: 물체의 위치 계산.
- Fragment Shader: 물체의 색상 정의.

2.렌더링 파이프라인:

- OpenGL 초기화 → 셰이더 생성 → 투영 행렬 설정 → 물체 그리기.

3.배치 논리:

- 반복문과 배열을 사용해 도로, 나무, 가로등의 위치를 관리.

개발 내용 - 기능 설명

---

1. 도로 :

•화면 하단에 고정된 크기의 사각형으로 렌더링.

•회색 셰이더(fragmentShaderSourceGray)로 색상 표현.

2. 가로수 :

•줄기와 잎으로 구성.

•줄기: 갈색 셰이더(fragmentShaderSourceBrown) 사용.

•잎: 초록색 셰이더(fragmentShaderSourceGreen) 사용.

•일정 간격으로 도로 양옆에 배치.

개발 내용 - 기능 설명

---

3. 가로등 :

•기둥과 등불로 구성.

•기둥: 회색 셰이더(fragmentShaderSourceGray) 사용.

•등불: 노란색 셰이더(fragmentShaderSourceYellow) 사용.

4.구름 :

•여러 개의 타원으로 구성된 구름 모양.

•화면 상단에 배치, 흰색 셰이더(fragmentShaderSourceWhite) 사용.

코드 분석 - 주요 파일

1. main.cpp:

- 프로그램의 시작점으로서 물체 배치, 셰이더 설정, 투영 행렬 처리.

2. objects.cpp:

- 물체의 렌더링 로직 구현.
- 각 물체를 그리기 위한 OpenGL 코드 포함.

3. shader.cpp:

- 셰이더 로드, 컴파일, 프로그램 생성 및 코드 재사용성을 높이기 위해 분리.

코드 분석

```
while (!glfwWindowShouldClose(window)) {
    processInput(window);
    glClear(GL_COLOR_BUFFER_BIT);
    // 물체 렌더링 함수 호출
    drawRoad();
    drawTreeTrunk(x);
    drawTreeLeaves(x);
    drawLampPost(x);
    drawLampLight(x);
    drawCloud(x, y);
    glfwSwapBuffers(window);
    glfwPollEvents();
}
```

▶ 렌더링 루프

```
float treePositions[] = { -1.6f, -0.8f, 0.0f, 0.8f };  // 나무 위치
for (float x : treePositions) {
    drawTreeTrunk(x);
    drawTreeLeaves(x);
}
```

▶ 물체 배치 알고리즘:

-> 반복문과 배열

코드 분석

```glsl
layout(location = 0) in vec3 aPos;
uniform mat4 projection;
void main() {
    gl_Position = projection * vec4(aPos, 1.0);
}
```

▶ 셰이더 코드 - Vertex Shader

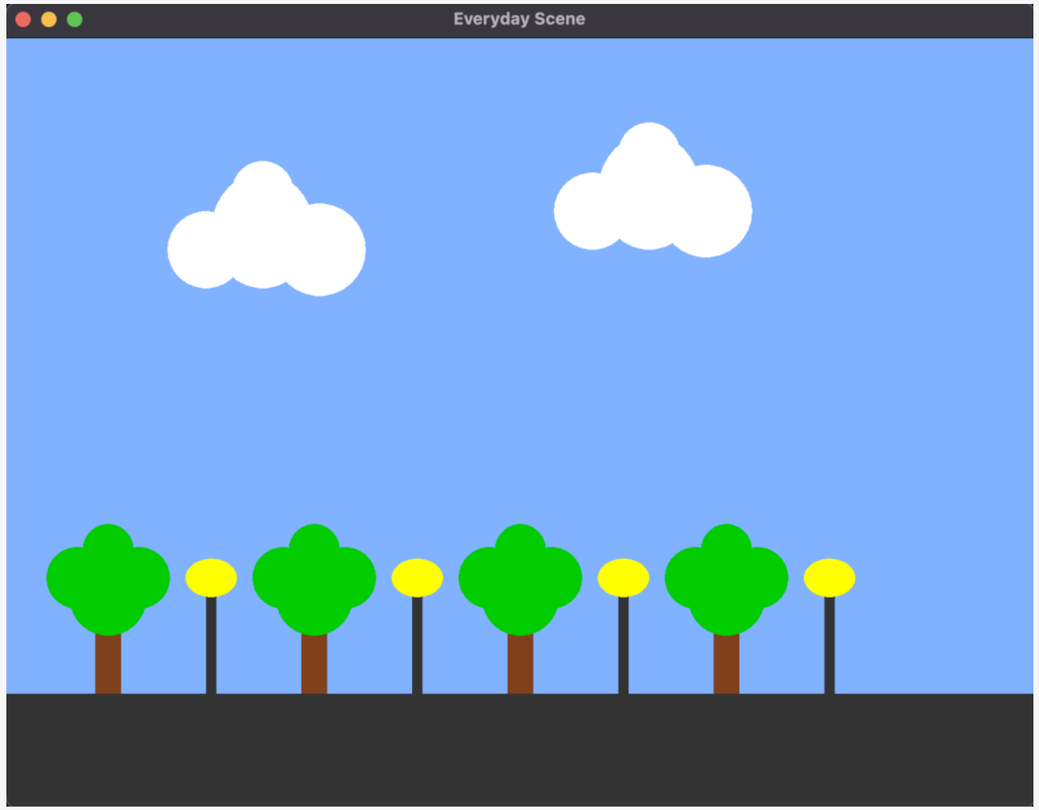
-> projection 행렬을 사용해 물체의 위치를 계산.

```glsl
out vec4 FragColor;
void main() {
    FragColor = vec4(0.5, 0.25, 0.1, 1.0); // 예: 갈색
}
```

▶ 셰이더 코드 - Fragment Shader

-> 물체의 색상을 설정.

시연 장면



▶ 구름, 나무, 가로등, 도로 등
생활 주변의 물체를 단순화 시켜 모델링한 모습

▶ OpenGL 초기화 → 셰이더 생성 →
투영 행렬 설정 → 물체를 그린 모습

▶ 반복문과 배열을 사용해 도로, 나무, 가로등의
위치를 관리한 모습

결론 및 개선 방향

결론 (성과)

- OpenGL 기본 구조를 학습 및 적용.
- 반복문을 활용해 효율적인 렌더링 구조 구현.
- 물체 간 간격 조정, 배열을 통한 동적 배치.

개선 방향

1. 애니메이션 추가:

- 구름 이동, 가로등 불빛 깜빡임.

2. 사용자 인터랙션:

- 키보드나 마우스로 물체 이동 제어.

3. 3D 확장 가능성:

- 직교 투영에서 원근 투영으로 전환 및 3D 공간 배치