Hello Triangle: 첫번째 그래픽스 프로그래밍 (심화)

김준호

Abstract

모던 OpenGL의 서버-클라이언트 모델을 이해한다. CPU가 접근할 수 있는 메모리에 있는 정점의 좌표, 색깔 등과 같은 정점의 속성 정보를 어떻게 GPU가 접근할 수 있는 비디오 메모리로 전송하는지 이해한다.

1 모던 OpenGL 프로그래밍 기본

모던 OpenGL은 Fig. 1과 같은 클라이언트-서버 프로그래밍 모델에 기반을 두고 있다. 여기서는 모던 OpenGL의 클라이언트-서버 모델을 이해하고, 어떤 방식으로 모던 OpenGL에서 삼각형을 그리기 위한 정점 데이터가 관리되어야 하는지 학습한다.

- 정점에 대한 위치(position), 색깔(color)과 같은 정점 속성(vertex attribute) 데이터를 클라이언트 메모리 측에서 설정하고 서버 메모리 측에 전송하는 방법을 학습한다.
- 서버 메모리에 전송된 정점 속성들을 이용해 모던 OpenGL로 화면을 그리는 방법을 학습한다.
- 정점 버퍼 객체(vertex buffer object, 이하 VBO)를 이해한다.

1.1 클라이언트-서버 모델

모던 OpenGL에서는 프로그래밍 가능한 렌더링 파이프라인을 지원하기 위해 Fig. 1과 같은 클라이언트-서버 모델을 따르는 프로그래밍 기법을 도입하고 있다.

클라이언트는 CPU와 메인메모리로 이루어지는 일반적인 컴퓨팅 환경을 의미한다. 일반적으로 클라이언트 측에서 돌아가는 코드는 C/C++로 작성된 OpenGL 코드이며, 때에 따라 Java/JavaScript 등 다른 언어로 작성될 수도 있다.

서버는 GPU와 GPU 메모리로 이루어지는 그래픽카드 상의 컴퓨팅 환경을 의미한다. 서버 측에서 돌아가는 코드는 OpenGL 쉐이딩 언어(OpenGL Shading Language, 이하 GLSL)로 작성된 쉐이더 코드이다. GLSL 언어는 OpenGL 버전에 따라 조금씩 다른데, 여기서는 OpenGL 2.x에서 사용하는 GLSL 1.2 버전을 사용하기로 한다.

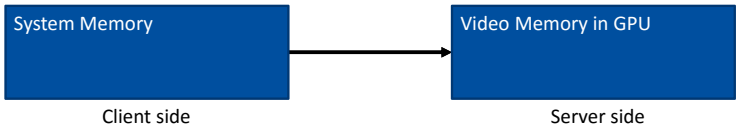

Fig. 1. OpenGL 클라이언트-서버 모델

모던 OpenGL이 도입하고 있는 클라이언트-서버 모델은 다음과 같은 방식으로 동작한다.

- CPU는 메인 메모리에 적힌 내용만 볼 수 있다.
- GPU는 GPU 메모리에 적힌 내용만 볼 수 있다.
- 메인 메모리에 적힌 내용을 GPU 메모리에 보내려면 특정 모던 OpenGL 명령들을 사용해야 한다.
- GPU는 GPU 메모리에 적힌 내용으로 화면을 렌더링 한다.

CPU와 GPU는 각자 자신이 볼 수 있는 메모리만 본다는 것이 중요한 점이다. 따라서, 모던 OpenGL을 이용해 삼각형을 그리려면 삼각형의 정점 데이터가 메인 메모리와 GPU 메모리 두 곳 모두 존재해야 한다.

2 핵심코드 파악하기

본 예제에서 삼각형을 그리기 위한 핵심코드는 main.cpp에 있는 아래 세 부분이다.

1) 메인 메모리에 정점 데이터를 정의하는 부분
2) 메인 메모리에 정의된 정점 데이터를 GPU 메모리로 전송하는 부분
3) GPU 메모리에 정의된 정점 데이터로 물체를 그리는 부분

국민대학교 소프트웨어학부, junho@kookmin.ac.kr

2.1 정점 데이터 정의 (CPU 메모리)

삼각형의 정점(vertex) 삼차원 위치(position)와 색깔(color)을 정의하는 코드 조각이다. 삼각형은 3개의 정점으로 이루어져 있기 때문에, 위치정보 3개와 색깔 정보 3개가 명시되어 있음을 주목하자.

각 정점의 삼차원 위치를 정의하기 위해서는 x, y, z로 구성된 3개의 float형 변수를 쓴다.

일반적으로 컴퓨터그래픽스에서 한 색깔을 표현하기 위해 r, g, b로 구성된 3개의 숫자를 쓴다. r, g, b는 각각 빨강색(red), 초록색(green), 파랑색(blue) 요소가 얼마나 있는지를 나타낸다.

```
// per-vertex 3D positions (x, y, z)
GLfloat position[] = {
   0.5f,  0.5f,  0.0f,     // 0th vertex position
  -0.5f, -0.5f,  0.0f,     // 1st vertex position
   0.5f, -0.5f,  0.0f,     // 2nd vertex position
};

// per-vertex RGB color (r, g, b)
GLfloat color[] = {
  1.0f, 0.0f, 0.0f,        // 0th vertex color (red)
  1.0f, 0.0f, 0.0f,        // 1st vertex color (red)
  1.0f, 0.0f, 0.0f,        // 2nd vertex color (red)
};
```

2.2 정점 데이터 전송 (CPU 메모리 → GPU 메모리)

CPU 메모리에 있는 정점 데이터를 어떻게 GPU 메모리로 전송하는지 살펴보자. 먼저 정점 데이터 전송에 앞서 GPU 메모리에 정점 데이터가 보관된 장소를 마련해야 한다. 모던 OpenGL에서는 정점 데이터가 보관될 GPU 메모리를 할당하는 특수한 방법을 제공하는데 이를 정점 버퍼 객체(vertex buffer object, 이하 VBO)라고 한다.

다음은 GPU 메모리에 정점의 삼차원 위치를 담을 VBO와 정점의 칼라를 담을 VBO를 만들고, CPU 메모리의 정점 정보를 GPU 메모리 상의 VBO로 전송하는 코드 조각이다.

```
void init_buffer_objects()
{
  glGenBuffers(1, &position_buffer);
  glBindBuffer(GL_ARRAY_BUFFER, position_buffer);
  glBufferData(GL_ARRAY_BUFFER, sizeof(position), position, GL_STATIC_DRAW);

  glGenBuffers(1, &color_buffer);
  glBindBuffer(GL_ARRAY_BUFFER, color_buffer);
  glBufferData(GL_ARRAY_BUFFER, sizeof(color), color, GL_STATIC_DRAW);
}
```

2.3 정점 데이터로 삼각형 그리기 (GPU 메모리)

정점 데이터가 모두 GPU 메모리에 전송된 상태이므로 모던 OpenGL은 GPU 메모리에 있는 VBO에 기록된 삼각형 정점 데이터를 참조하여 그림을 그릴 수 있다.

다음은 정점의 삼차원 위치가 담긴 VBO와 칼라가 담긴 VBO를 각각 쉐이더(shader)의 특정 속성(attribute)에 결합시키고 삼각형을 렌더링하는 코드 조각이다. 아직 쉐이더를 배우지 않았기 때문에 쉐이더 관련 부분은 이해하지 않아도 된다.

```
void render_scene()
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  // 특정 쉐이더 프로그램 사용
  glUseProgram(program);

  // 정점 attribute 배열 loc_a_position 활성화
  glEnableVertexAttribArray(loc_a_position);
  // 앞으로 언급하는 배열 버퍼(GL_ARRAY_BUFFER)를 position_buffer로 지정
  glBindBuffer(GL_ARRAY_BUFFER, position_buffer);
  // 현재 배열 버퍼에 있는 데이터를 버텍스 쉐이더 loc_a_position 위치의 attribute와 연결
  glVertexAttribPointer(loc_a_position, 3, GL_FLOAT, GL_FALSE, 0, (void*)0);
```

```
  // 정점 attribute 배열 loc_a_color 활성화
  glEnableVertexAttribArray(loc_a_color);
  // 앞으로 언급하는 배열 버퍼(GL_ARRAY_BUFFER)를 color_buffer로 지정
  glBindBuffer(GL_ARRAY_BUFFER, color_buffer);
  // 현재 배열 버퍼에 있는 데이터를 버텍스 쉐이더 loc_a_color 위치의 attribute와 연결
  glVertexAttribPointer(loc_a_color, 3, GL_FLOAT, GL_FALSE, 0, (void*)0);

  // 삼각형 그리기
  glDrawArrays(GL_TRIANGLES, 0, 3);

  // 정점 attribute 배열 비활성화
  glDisableVertexAttribArray(loc_a_position);
  glDisableVertexAttribArray(loc_a_color);
  // 쉐이더 프로그램 사용해제
  glUseProgram(0);
}
```

3 반드시 이해해야할 사항

여기서 가장 주목해야 하는 사실은 모던 OpenGL에서 렌더링을 할 때, 메인 메모리의 데이터를 전혀 참조하지 않고 GPU 메모리에 있는 데이터만 참조해서 그림을 그렸다는 사실이다. 모던 OpenGL이 이런 방식으로 동작하기 때문에 정점에 대한 데이터가 메인 메모리와 GPU 메모리 두 곳에 존재한다는 사실 또한 중요하다. 따라서, 효과적인 모던 OpenGL 프로그램 작성을 위해서는 두 곳에 존재하는 데이터를 어떻게 효과적으로 동기화(synchronization)할 것인가를 항상 고민해야 한다.

4 연습문제

모던 OpenGL의 클라이언트-서버 모델을 이해하고 다음 사항을 고민해 보자.

1) 정점의 삼차원 위치나 칼라 데이터를 업데이트 시키려면 어떤 방식으로 해야 할까?
2) GPU가 메인 메모리를 접근하지 못하도록 설계되어 있는데 왜 그런 것일까?