Hello Triangle: 모던 OpenGL 프로그래밍 시작하기

김준호

Abstract

모던 OpenGL 프로그래밍 기초와 원리를 학습한다. 삼각형 하나를 그리는 간단한 Hello Triangle을 작성함으로써 모던 OpenGL 프로그래밍 방법을 이해한다. 정점의 삼차원 위치, 색깔 등과 같은 정점 속성(vertex attribute)를 설정하는 방법을 학습한다. 모던 OpenGL의 서버-클라이언트 모델을 이해하고 모던 OpenGL 프로그래밍의 원리를 파악한다. CPU가 접근할 수 있는 메모리에 있는 정점의 좌표, 색깔 등과 같은 정점의 속성 정보를 어떻게 GPU가 접근할 수 있는 비디오 메모리로 전송하는지 이해한다.

1 HELLO TRIANGLE

OpenGL에서 그림을 그리는 가장 기본적인 단위는 삼각형(triangle)이다. 일반적으로 실시간 그래픽스에서는 물체를 그릴 때 해당 물체의 외형(surface)을 여러 개의 삼각형으로 근사(approximation)하여 표현한다. 따라서 OpenGL을 이용해서 임의의 물체를 그리려면 먼저 삼각형 하나를 그리는 방법부터 이해해야 한다.

여기서는 Fig. 1과 같이 삼각형 하나를 그리는 가장 간단한 형태의 모던 OpenGL 프로그램을 작성한다. 이를 통해 다음을 학습한다.

- 정점(vertex)에 대한 위치(position), 색깔(color)과 같은 정점 속성(vertex attribute) 데이터를 메인 메모리에 설정하는 방법을 학습한다.
- `glDrawArrays`를 이용하여 triangle soup 방식으로 삼각형을 그리는 방법을 학습한다.
- `glDrawElements`를 이용하여 vertex list & triangles 방식으로 삼각형을 그리는 방법을 학습한다.

Fig. 1. Hello Triangle 예제

1.1 소스코드 컴파일 및 실행

터미널에서 git 명령어로 소스코드를 다운로드 받는다.

```
$ git clone https://github.com/kmuvcl/2023_graphics
```

다운로드 받은 소스코드 디렉토리로 이동한 후, 터미널에서 make 명령어를 통해 컴파일을 수행하자. 컴파일이 성공적으로 끝나면 현재 디렉토리에 실행가능한 파일인 `triangle`이 생성된다.

국민대학교 소프트웨어학부, junho@kookmin.ac.kr

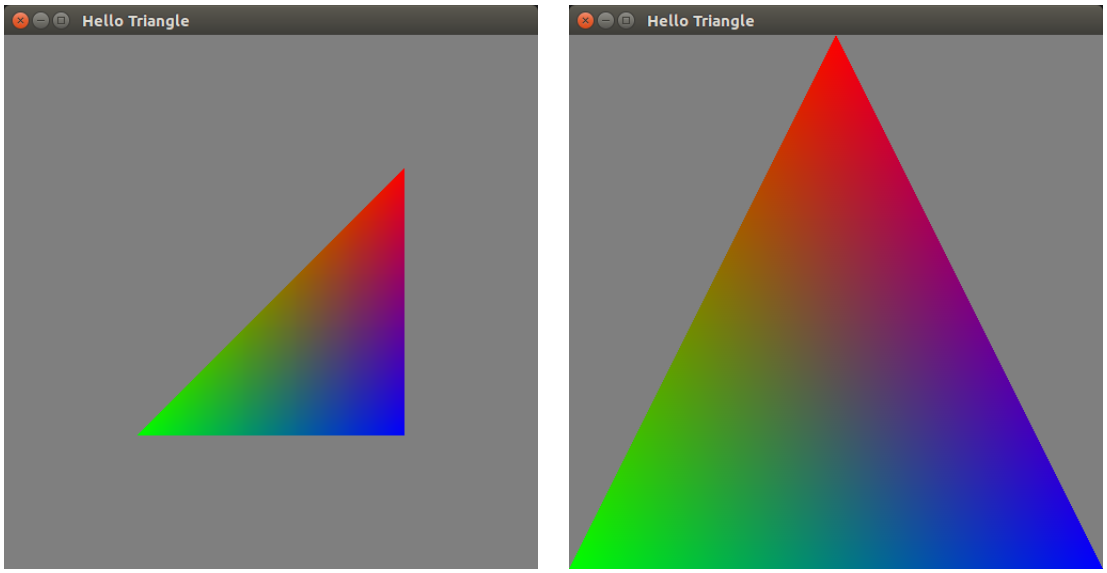

Fig. 2. 무작정 해보기: (a) 정점의 색깔을 바꾼 경우, (b) 정점의 위치까지 바꾼 경우

```
$ make
```

터미널에서 아래와 같이 `triangle`을 실행시켜, Fig. 1과 같은 화면이 뜨는 것을 확인하자.

```
$ ./triangle &
```

2 무작정 해보기

삼각형의 색깔과 모양을 바꿔보자.

소스코드 중 `main.cpp`를 에디터로 열어 살펴보면, 배열 `GLfloat g_position[]`과 배열 `GLfloat g_color[]`가 눈이 띌 것이다. 이 부분이 삼각형을 이루는 3개의 정점에 대한 삼차원 위치와 색깔을 지정한 부분이다.

2.1 색깔 바꾸기

먼저 삼각형의 색깔을 바꿔보자. 배열 `g_color[]` 내용을 다음과 같이 바꾼 후, 컴파일 및 실행을 해보자. Fig. 2(a)와 같이 나와야 정상이다.

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

2.2 모양 바꾸기

그 다음 삼각형의 모양을 바꿔보자. 배열 `g_position[]`의 내용을 다음과 같이 바꾼 후, 컴파일 및 실행을 해보자. Fig. 2(b)와 같이 나와야 정상이다.

```
// per-vertex 3D positions (x, y, z)
//GLfloat g_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
```

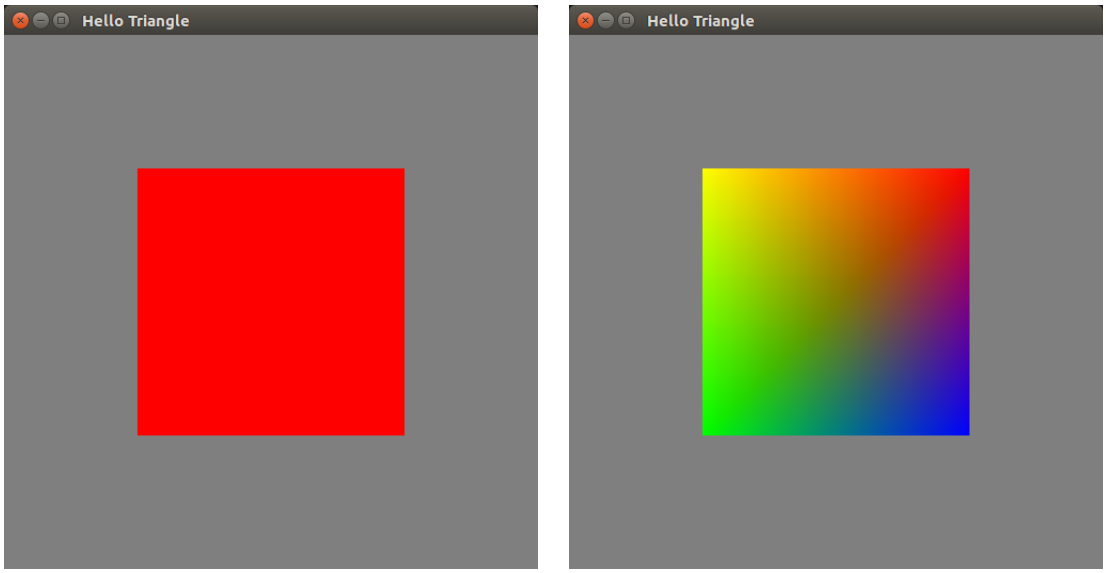

Fig. 3. 사각형 그리기: (a) 빨간색 삼각형 두 개로 그린 경우, (b) 정점의 색깔을 적절히 설정하여 사각형의 내부색이 바뀌도록 그린 경우

```
//};
GLfloat g_position[] = {
   0.0f,  1.0f,  0.0f,     // 0th vertex position
  -1.0f, -1.0f,  0.0f,     // 1st vertex position
   1.0f, -1.0f,  0.0f,     // 2nd vertex position
};
```

3 사각형 그리기

이제, Fig. 3(a)와 같이 사각형 하나를 그려보자. OpenGL에서 사각형을 직접 그릴 수도 있지만 여기서는 삼각형 2개를 이용해 사각형을 그리도록 한다.

삼각형 데이터를 그리는 대표적인 방법으로는 1) triangle soup 방식과 2) vertex list & triangles 방식이 있다. OpenGL에서 첫번째 방식은 `glDrawArrays`함수로 제공하며, 두번째 방식은 `glDrawElements`함수로 제공한다.

3.1 Triangle Soup 방식으로 사각형 그리기

먼저 triangle soup 방식을 통해 삼각형 2개로 사각형 하나를 그려보도록 하자.

3.1.1 정점 데이터 변경

삼각형 하나를 표현하는데 3개의 정점이 필요하므로, 삼각형 2개를 그리려면 총 6개(=3x2)의 정점이 필요하다. 다음과 같이 정점 6개에 대한 데이터를 표현하도록, 배열 `g_position[]`과 배열 `g_color[]`를 수정하자.

```
// per-vertex 3D positions (x, y, z)
GLfloat g_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

   0.5f,  0.5f,  0.0f,     // 3nd vertex position
  -0.5f,  0.5f,  0.0f,     // 4nd vertex position
  -0.5f, -0.5f,  0.0f,     // 5nd vertex position
};
// per-vertex RGB color (r, g, b)
GLfloat g_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)

  1.0f, 0.0f, 0.0f,        // 3nd vertex color (red)
```

```
  1.0f, 0.0f, 0.0f,       // 4nd vertex color (red)
  1.0f, 0.0f, 0.0f,       // 5nd vertex color (red)
};
```

3.1.2 렌더링에 쓰이는 정점 개수 변경

다음으로 물체를 표현하는데 사용된 정점의 개수가 3개에서 6개로 변경되었으므로 OpenGL이 이를 알아챌 수 있도록 `glDrawArrays`의 정점 개수에 대한 파라메터를 3에서 6으로 수정한다.

```
void render_object()
{
  // ...
  // glDrawArrays(GL_TRIANGLES, 0, 3);
  glDrawArrays(GL_TRIANGLES, 0, 6);
  // ...
}
```

3.2 Vertex List & Triangles 방식으로 사각형 그리기

이제 vertex list & triangles 방식을 통해 삼각형 2개로 사각형 하나를 그려보도록 하자.

3.2.1 정점 데이터 변경

삼각형 두개를 이용해 사각형 하나를 표현할 때, 실제로 4개의 정점만 필요하다. 정점 4개에 대한 위치와 색상에 관한 데이터를 아래와 같이 수정한다.

```
// per-vertex 3D positions (x, y, z)
GLfloat g_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
  -0.5f,  0.5f,  0.0f,     // 3rd vertex position
};
// per-vertex RGB color (r, g, b)
GLfloat g_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)
  1.0f, 0.0f, 0.0f,       // 3rd vertex color (red)
};
```

3.2.2 인덱스 데이터 추가

삼각형 2개를 그리는데 필요한 정점의 인덱스 개수는 총 6개(=3x2)이다. 다음과 같이 정점 인덱스의 배열 `g_indices[]`를 추가하자.

```
// triangle-vertex indices
GLubyte g_indices[] = {0, 1, 2, 0, 3, 1};
```

3.2.3 렌더링에 쓰이는 함수 변경

물체를 vertex list & triangles 방식으로 표현하기로 했으므로 OpenGL의 `glDrawElements`함수를 이용해 아래와 같이 물체를 렌더링한다.

```
void render_object()
{
  // ...
  // glDrawArrays(GL_TRIANGLES, 0, 3);
  // glDrawArrays(GL_TRIANGLES, 0, 6);
  glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, g_indices);
  // ...
}
```

4 연습문제

소스파일 `main.cpp`를 수정해서 프로그램을 다음과 같이 변형해 보자.

1) Fig. 3(b)와 같이 사각형의 내부색이 바뀌도록 그려보자. 이때, `glDrawArrays`와 `glDrawElements`를 각각 활용하여 사각형을 그려보도록 하자.
2) 삼각형을 여러 개를 그리는 방법으로 자신이 원하는 복잡한 그림을 그려보자.