Lecture 2

2018년 3월 12일 월요일 오전 11:27

OpenGL 렌더링 구조

OpneGL에서 렌더링은 일종의 State Machine 형태로 동작한다.

병렬화가 되지 않으면 CPU가 데이터를 넘겨주고 GPU가 렌더링을 하는동안 CPU는 다른 일을 할 수 없다.

Q. CPU에서 만들어둔 데이터를 왜 GPU에서 바로 사용하지 못하는가?
A. CPU와 GPU의 메모리가 다르기 때문에

왜 메모리가 각각 존재할까?

CPU-GPU 병렬화를 위해서
CPU가 일하고 데이터를 만들어서 그 데이터를 GPU가 사용하는 형식으로 넘겨주면 GPU는 렌더링을 시작하고 CPU는 다른 일을 시작한다.

다이렉트에서도 그렇고 OpneGL에서도 그렇고 바꿔서 사용해야 하는 것

그렇다면 OpenGL에서의 렌더링 고유의 형식은 무엇일까

Vertex Buffer Object (Vertex에 관련된 오브젝트)

만약에 Index에 관련된 데이터라면 Index Buffer Object 등의 형식이다.

<시험 문제!>
OpenGL에서 그림을 그리기 위해서 가장 필수적이면서 기본이 되는 것 :: Vertex

즉, Vertex 없이 무언가 그리는 것은 불가능 하다.

DrawPoint 같은게 없다. (Point는 요소이다. 그 전의 데이터인 Vertex가 필요하다.)

오늘은 Vertex를 생성해서 GPU에 어떻게 넘기는지

최초 입력 (Vertex) 가 없다면 다음단계로 넘길 데이터가 없다.
-> 최종 출력물이 존재할 수 없다.

OPENGL 데이터 준비

OpenGL은 오른손 좌표계이다.

화면 오른쪽 끝이 1,0,0 왼쪽 끝이 -1,0,0

화면 맨위쪽 끝이 0,1,0 아래쪽 끝이 0,-1,0

오늘의 실습

(0,0,0), (1,1,0), (1,0,0) 의 데이터를 가지고 삼각형을 그려보자

배열에 넣을때 순서가 중요하다 와인딩 오더(시계방향, 반시계방향)

array는 RAM(메인 메모리)에 저장되어 있다.

위의 array는 OpenGL에서 바로 사용 가능할까?
- 바로사용이 불가능하다.
- 메인메모리에 GPU에서 접근하기 위해서는 복잡한 과정을 거쳐야 한다.
- GPU는 GPU만의 메모리가 있따. (VRAM)

즉, CPU 메모리에 있는 배열을 GPU 메모리에 똑같이 만들어 주어야 한다.

그것을 Vertex Buffer Object가 한다.(VBO)

OpenGL Buffer Object
- 다양한 목적으로 사용하기 위한 버퍼 오브젝트
- 오브젝트 하나씩 선언해서 사용한다.
-

정리

CPU와 GPU는 완벽히 병렬로 동작한다. 그렇기 때문에 둘은 각각 메모리를 가진다.

CPU에서 생성한 데이터는 렌더링을 위해서 OpneGL에서 관리하는 메모리 상에 올려야 한다.

명령어

glGenBuffers(Glsizei n, Gluint *ids)

- Buffer Object를 생성하고 Object ID를 ids에 넣어줌
- N은 만들고 싶은 개수, ids 주소
- Object ID가 Return 된다 (포인터가 리턴되서 직접 엑세스 하는 방식이 아님)

예시

```
Gluint VBO;
glGenBuffers(1, &VBO);
```

glBindBuffer(Glenum target, Gluint id)

- 생성된 VBO를 ID를 사용하여 Bind 함
    - Bind란?
        - 실제 OpneGL에서 작업할 대상을 선정해 주는 것
        - 데이터를 올리려고 하는데 그 데이터가 array 형식의 buffer를 가진다면 GL_ARRAY_BUFFER를 사용

예시

```
Gluint VBO;
glGenBuffers(1, &VBO); // 형태가 없는 상태

glBindBuffer(GL_ARRAY_BUFFER, VBO); // Bind 이후 array 형태를 가지는 상태
// 여기까지는 데이터가 없다.
```

glBufferData(Glenum target, Glsizeiptr size, const Glvoid *data, Glenum usage);

- Bind 된 VBO에 데이터를 할당

```
Gluint VBO;
glGenBuffers(1, &VBO); // 형태가 없는 상태

glBindBuffer(GL_ARRAY_BUFFER, VBO); // Bind 이후 array 형태를 가지는 상태
// 여기서부터 메모리상에 데이터가 할당 된다. ( malloc 느낌인가 )
glBufferData(GL_ARRAY_BUFFER,
             sizeof(vertices),      // 크기
             vertices,              // 메모리 주소
             GL_STATIC_DRAW);
```

메모리 주소부터 크기까지 복사를 하는 것 (CPU Memory -> GPU Memory)

OpenGL 데이터 사용

일단 Bind

- 중간에 다른 오브젝트가 Bind 되었을 가능성이 있기 때문에 한번 더
- OpenGL은 종류 당(GL_ARRAY_BUFFER와 같은 종류) 하나의 오브젝트 만 Bind 허용

```
glEnableVertexAttribArray();
glBindBuffer(GL_ARRAY_BUFFER, VBO);
```

glVertexAttribPointer(Gluint index, Glint size, Glenum type, Glboolean normalize, Glsizei stride, const Glvoid *pointer);

- Draw 시 데이터를 읽어갈 단위의 크기 및 시작점 설정

```
glVertexAttribPointer(0,            // 시작점?
                      3,            // 한번에 읽을 크기
                      GL_FLOAT,     // 데이터 타입
```

```
                GL_FALSE,       // normalize 인가?
                0,
                0);
```

glDrawArrays(Glenum mode, Glint first, Glsizei count);

- 어떠한 Primitive로 구성할 것인지
- Vertex를 몇 개를 그릴 것인지
- 이 함수를 호출 즉시 GPU가 동작

```
glDrawArrays(GL_POINTS, 0, 1);
```