2024-08-02-백민호

OpenGL

Hello Triangle

- Event-Driven Programming
  - 종료되지 않고 항상 무한 루프를 돌면서 user의 event를 기다린다
  - event 입력이 주어지면 해당하는 callback function을 실행한다
  - Retained mode(modern style) <-> Immediate mode
    - 실행 중에는 data나 program을 건드리지 않는 것(미리 준비해야 함!!)
    - 오고 가는 data를 최소화(성능이 좋아짐)
    - Main Memory -> GPU Memory
      - glBufferData : vertex data
      - glUniform : uniform type var(read-only), vert shad 파일 등 모두 볼 수 있다
      - glDrawElement 실행 할 때 GPU가 돌아가면서 pipeline이 동작한다
- OpenGL as State Machine
  - 전역 변수(상태)가 많다, 객체들이 attribute를 들고 있지 않기 때문!! (OOL 이전)
  - 어떤 function을 부를 때 bind를 해줘야 한다. function을 실행할 버퍼들을 bind해야 함
- Coordinate Systems
  - 카메라는 RHS, projection을 통과하면 왼손으로 넘어감
  - Normalized Device Coordinate(NDC) system
    - projection이후 객체는 NDC system에 위치한다(LHS), 2D는 이미 일어났다 생각
- OpenGL code
  - vertex shader : input, output, uniform type, gl_Position을 정해준다(무조건!)
    그 외 attribute도 아래 써서 fragment shader로 넘겨주는 역할을 한다
  - fragment shader : in, out, uniform, vertex의 output과 frag의 input이 이름이 같다고 같은 것이라고 착각하면 안됨!! interpolation되어서 넘어간 것임!!
    - uniform은 읽기 전용 전역 변수, vertex shader와 같고 vertex, fragment 동일하다
  - event callback의 type 같은 것은 os 마다 정해져 있다

- Event inf loop를 돌면서 user event를 기다리거나 update(render) 시행한다
- user init
  - Retained mode에 맞게 vertex buffer와 vertex array objects를 생성
  - depth test, back-face culling(뒤는 묘사 x) 등 GL state init
  - buffer 담을 GPU buffer memory 생성
  - Vertex Array Object(VAO)
    - vertex link 정보를 담는 array (generate -> bind)
    - 실제 값과 링크를 분리함으로써 object에 포함된 vertex를 매 프레임마다 반복적으로 작업을 해야하는 것을 막을 수 있다.

OpenGL Shading Language(GLSL)

- https://thebookofshaders.com/
- https://www.khronos.org/opengl/wiki/
- shading : light-surface interaction physical simulation, flat, phong etc...
- 빛이 보이는 과정을 계산하는 프로그램 --> shader(user defined GPU program)
- GPU shading : run-time compile, linked, launched
- vertex, fragment 따로 code 및 compile 되고 linker를 통해 하나의 program object 생성
  program id 1개를 받게 된다
- vertex shader, fragment 둘 다 한 번에 하나씩 처리한다(각 프로세서 1개 당 병렬로)
- vertex shader
  - input, output, uniform, main function
  - vertex buffer attribute를 access 할 때 layout(location=0)
  - input은 vertex buffer에 들어온다, gl_Position 반드시 써야한다
- fragment shader
  - input은 vertex shader의 output임!!(보간되어 들어옴)
  - fragColor를 정의한다(반드시 쓸 필요는 없음!)
- matrix, vectors, texture 등 type과 c++과 유사한 오버로딩과 생성자
  - but pointer, dynamic malloc 쓸 수 없다(gpu 주소는 access 불가능)
  - 모든 함수는 inline function, no call stack, no recursion(모두 main으로 들어감)
  - no strings, char, double, short, long, file I/O, console I/O
  - global 변수들은 되도록 in, out, uniform 이라는 qualifiers를 써줘야 한다

- shader는 vertex 하나 마다 쓰기 때문에 최적화가 중요하다