중간고사 대비 정리

- 컴퓨터 그래픽의 정의
    - 실세계의 물체를 컴퓨터를 이용하여 표현하는 것
    - 물체를 모델링, 랜더링하여 Display하는 것
- 컴퓨터 그래픽 3가지 단계
    - 모델링
        - 물체를 표현하는 방법
        - 단순 데이터로 존재
    - 랜더링
        - 모델링된 물체를 화면에 표현하는 방법
        - 빛과 광원을 계산하여 물체를 표현
    - 애니메이션
        - 물체의 움직임을 표현하는 방법
- 컴퓨터 그래픽 키워드
    - Polygons
    - Rendering
    - Visibility
- 물체를 화면에 그리기 위해 필요한 정보
    - 카메라 정보, 시각 정보(위치, 원근감): Viewer
    - 조명 정보(빛의 위치, 색상, 세기): Light Source
    - 물체 정보(물체의 위치, 모양, 재질): Objects
    - ++ Attributes(Material, Texture, Transparency, ...)
        - 빛이 재질과 어떻게 상호작용하는지
- 실세계의 색을 결정하는 방법
    - 광원에서 빛이 시작되어 물체에 반사되어 눈에 보이는 것
    - 반사, 투과, 흡수의 3가지 방법으로 빛이 물체에 반사되어 눈에 보이는 것
- Ray Tracing
    - 빛을 쏘고 반사되는 과정을 계산하는 것
    - 즉, 시야각에서 빛을 쏘고 반사되는 과정을 계산하는 것 (역추적)

실세계의 실제 빛 반사를 컴퓨터로 표현하려면 매우 느리고 무겁다. (모든 데이터가 있어야 함)

따라서 레이트레이싱 기술이 발달함.

초기에는 스트리밍 방식으로 발달

- 단순화된 실용적 접근 방식으로 Pipeline architecture

    - 순서대로 한번에 하나씩 처리
    - Vertices -> Vertex Processing -> Primitive Assembly -> Rasterization -> Fragment Processing -> Frame Buffer

- Vertex Processing

    - 대부분의 작업은 객체 표현을 한 좌표계에서 다른 좌표계로 변환하는 것이다.
        - 객체 좌표계 -> 카메라 좌표계 -> 화면 좌표계
        - 이때 변환은 행렬을 이용하여 수행한다.
    - 3차원 객체를 2차원 화면에 표현하기 위해 변환한다.

꼭지점을 읽는 과정이 Projection

- Clipper and Primitive Assembler: 2D image -> Primitives

    - 점을 선으로 연결

- Rasterization

    - 객체가 보이려면 프레임 버퍼에 적절한 픽셀에 색상을 할당해야 한다.
    - 래스터라이저는 각 개체에 대한 일련의 조각을 생성한다. (픽셀)
    - 해당 픽셀은 프레임 버퍼에 위치하며 생상과 깊이 속성을 가진다.
    - 선을 삼각형으로 채움

- Fragment Processing

    - 프래그먼트는 래스터라이저에 의해 생성된 픽셀이다.
    - 프래그먼트 프로세서는 프래그먼트의 색상을 계산한다.
    - 꼭기점 색상을 보간하는 것이 일반적이다.
    - 삼각형을 픽셀로 채워서 2D image로 변환

- OepnGl API

    - 대부분은 이미지를 형성하는 데 필요한 기능들을 제공한다. (object, viewer, light source, ...)
    - 이 외에도 keyboard, mouse같은 기능도 제공한다.
    - 다음 기능도 제공한다
        - Point, Line, Polygon, Curve, Surface, ...

위에서 다룬 연산들은 대부분 벡터, 행렬 연산으로 이뤄짐 (실수 계산)

- CPU와 GPU

    - OpenGl에서 쉐이더를 사용하기 위해서는 GPU가 필요하다.
    - GPU는 단순계산에 매우 유리

- OpenGL 최신 (3.0 이상)

    - 직접 쉐이더를 작성해야 한다.

- OpenGL Libraries

    - GLUT
        - OpenGL Utility Toolkit

        - 윈도우 열기, 닫기, 키보드, 마우스 등의 이벤트 처리
    - GLEW
        - OpenGL Extension Wrangler Library
        - 특정 시스템에서 사용 가능한 OpenGl 확장에 쉽게 액세스 가능
        - glew.h만 포함하면 된다.

- OpenGl Pipeline

    - Vertex Processor
        - 화면의 정보를 점으로 읽음
        - 꼭지점을 읽음
        - Vertex Shader
    - Rasterizer
        - Primitives -> Fragments
        - 점을 선으로 연결, 주어진 도형 가장 많이 삼각형으로 사용
    - Fragment Processor
        - Fragments -> 2D image
        - 삼각형을 채워서 2D image로 변환
        - 삼각형의 내부를 채워서 색을 채움
        - Fragment Shader

- OpenGL Pipeline Process

    - 버퍼 개체를 생성하고 개체에 데이터를 로드한다.
    - 쉐이더 프로그램을 만든다.
    - 세이터 프로그램과 데이터 위치를 연결한다.
    - 랜더링 한다.

- Vertex Array Objects(VAO)

    - VAO는 해당 물체의 데이터를 저장한다.
    - glGenVertexArrays()로 VAO를 생성한다.
    - glBindVertexArray()로 VAO를 바인딩한다.

정점 데이터는 VBO에 저장되어야 하며 VAO와 연결되어야 한다.

VBO 데이터가 여러개 존재할 수 있기 때문에 VAO 에 넣어주는 방식

- Vertex Shader

    - 꼭지점 데이터 위치를 결정
    - 스케일링, 회전, 변형등에 사용
    - Fragment Shader에 데이터를 전달

- Fragment Shader

    - 프래그먼트의 색상을 결정

- GLSL 특징

    - C언어와 유사

    - 포인터, 재귀, 동적할당은 지원하지 않음
    - 벡터, 행렬은 기본 자료형
    - 구조체, 수학 라이브러리 지원

- GLSL 변수

    - float, int, bool
    - vec2, vec3, vec4 (float vector)
    - ivec2, ivec3, ivec4 (int vector)
    - bvec2, bvec3, bvec4 (bool vector)
    - mat2, mat3, mat4 (float matrix)
    - sampler2D (texture)

- 벡터 데이터 접근

    - x, y, z, w
    - r, g, b, a
    - s, t, p, q

- 벡터는 colume major

- 쉐이더 in, out

    - in: Vertex Shader -> Fragment Shader
    - out: Fragment Shader -> Frame Buffer

- gl_position

    - Vertex Shader의 실제 표시될 위치

- Uniform

    - CPU에서 GPU로 데이터를 전달하는 역할
    - 즉 실제 코드에서 쉐이더관련을 직접 제어하기 좋음

- GLUT callbacks

    - glutDisplayFunc()
        - 화면을 그리는 함수
    - glutReshapeFunc()
        - 윈도우 크기가 변경될 때 호출되는 함수
    - 등등

- Depth-Buffer(Z-Buffer)

    - z버퍼는 현재 깊이 값을 저장하기 위한 버퍼이다.
    - 각 픽셀 위치에 해당하는 메모리가 있다.
    - 이전에 저장된 거리보다 거리가 가까울 때 표시된다. 즉 깊이감
    - 화면 z축으로 그려야 하는 레이어 우선순위라고 생각
    - **심화**
        - 컬러버퍼와 깊이 버퍼를 세트로 선언하여 스왑하여 사용하면 깜빡임을 없앨 수 있다.

- Scalars
    - 스칼라는 단일 값을 가진다.
- Vectors
    - 더하기, 빼기, 내적, 외적, 크로스, 닷 등등
- 점과 벡터
    - 점과 방향벡터
    - 점과 점
        - 두 점을 잇는 벡터