2024-08-09-백민호

Camera

- camera coordinate systems(frames) : RHS, 앞이 -z 축
- default model-view matrix는 identity matrix, trans 아무것도 안한 것과 같다!
- Canonical View Volume(in NDC) : 길이 2짜리 cube로 모두 normalize 하는 것이다
- z축이 반전되어 있다 --> LHS, gl_Position은 NDC 상에서의 좌표 값이다
- Default projection : projection 안했을 때, 아무것도 안한 기본 상태
  - orthographic projection : z = 0
- Viewing : view matrix를 만드는 과정, look_at matrix를 만든다
  - 카메라의 좌표 표기 방식과, 객체의 좌표가 다르기 때문에 맞춰줘야 하기 때문!!
  - eye : 카메라 위치
  - at : 보고자 하는 객체의 중심
  - up : 카메라에서 봤을 때 수직 위(direction)
  - 저 세가지 config로 카메라 frame을 만들게 된다. basis vector(n, u, v)가 필요하다
- Perspective projection : perspective matrix 생성(원근법)
  - fovy, aspect, near, far
  - frustum(left, right, bottom, top, near, far)라는 projection matrix도 있음!
- Programming Example
  - Update 동안 cam의 aspect, projection matrix를 항상 갱신해주어야 한다
  - 이후 view와 projection matrix를 shader에 넘겨준다
- Virtual Trackball
  - 가상의 반 구형 껍질, x^2 + y^2 + z^2 = r^2 위의 점 p0에서 p1으로 이동
  - 커서의 좌표를 NDC로 normalize, 기존의 window coordinate에서 y축이 반전됨
- frame update에서 tracking motion, zoom, panning 등 다양한 좌표 이동이 가능하다

Shading

- shading은 빛과 material의 interaction
- phong illumination model
  - 물리적 완벽 X , 일부 반영
  - pipeline approach(ambient + diffuse + specular) -> compute rapidly

- Light Source Model
  - point light source : position, color, 거리가 멀면 밝기가 줄어듦)
  - distant(directional) light source : 무한정 멀리 있는 광원 가정, 마치 방향만 있는 빛을 생각한다, vec4에서 w 값을 0으로 놓음으로써 vector를 나타낸다
  - ambient light : 물리적 X, 모든 광원을 표현하기 어렵기 때문에 주변의 빛을 근사하기 위해 씬 전체적으로 approximation하게 빛을 주는 것
- Phong은 3가지 component로 분리하여 표시한다
  - ambient colors, diffuse colors, specular colors + position(homogeneous)
  - color : (r, g, b, a)
- Material Models
  - surface : smooth(specular), rough(diffuse), translucent(적당히 투과하는 것)
  - material도 세 가지 반사율에 대한 vectors(ambient, diffuse, specular, 0~1)
  - alpha : 빛이 모이는게 sharp한 정도(Shininess, 클수록 거울에 가까움, metal texture)
    shininess가 클 수록 specular가 퍼지지 않고 모여서 더 매끄러워 보임
  - material은 position이 없고(vertex에서 정해짐) color 3개 + shininess로 표현한다
  - 원래는 반사율인데 색깔처럼 표현한다
  - material을 상속해서 texture_material도 표현할 것이다(예정, material type)
- Update
  - fragment shader에 작업하기 위해 uniform으로 넘긴다
  - light model은 모든 vertex에 공통적으로 적용된다
    벗!! material에 대해서는 K, shininess가 따로 적용이 된다
- Blinn-Phong illumination Model
  - 4가지 vector : l(light source, surface point p에서 빛 쪽으로 가는 것, 방향 주의)
    v(p에서 camera까지, to Viewer)
    n(surface normal vector)
    h(halfway vector, l과 v의 가운데를 normalize한 벡터)
  - Ambient Reflections
    - 들어오는 빛과 반사율의 곱
  - Diffuse Reflections
    - scaling vector, l ・n(빛의 방향과 surface 방향, 음수 제거 max(0, x))
  - Specular Reflections
    - (n ・h)^β, β는 shininess(phong은 α, 거의 유사한 효과), 왜 halfway???
    - halfway는 l과 v 가운데, normal과 가까이 있을 수록 세진다(거울 반사되는 표면과 가까울수록??)

    - 이후 GLSL을 이용해서 표현!! 각각 더하기만 하면 되니까 계산이 독립적이다~
    - shading은 world space, camera space 등 에서 할 수 있다. camera 위치를 0으로 고정 할 수 있기 때문에 camera space에서 하는게 편하다~, v를 쉽게 구할 수 있음!!(epos)
        - norm = normalize(mat3(view_matrix *model_matrix)* normal);
        - point가 아니라 vector이기 때문에 view-model matrix에서 3x3 부분만 떼어낸다
    - norm by vertex shader -> rasterize -> norm by fragment shader
        - 따라서 normalize를 한 번 더 해줘야 함

Texture

- texture:반복되는 이미지 패턴이면서 현대에서는 gpu 메모리 상의 이미지를 말하기도 한다
- texture mapping : surface material에 쓰던 parameter를 이미지화 시키는 것, position에 따라서 변하는 function, geometry를 유지하면서 texture image를 잘 매핑해야 한다
    - texture를 만들고 attribute setting --> init에서 하면 됨
    - texture 좌표 assign, mapping 할 때 가져다 씀(만들어져 있음, preprocessing 과정)
    - shader에서 불러서 쓰기(CPU에서 하는게 아님)
- Geometry Pipeline
    - input : vertices -> geometry pipeline(raster pipeline) vertex shading cliping raster frag 등등 과정
- Pixel Pipeline
    - input : image -> 숨겨져 있음
- array of texels로 코딩하거나 texture file을 사용할 수 있음(image[64, 64, 3] or loader lib)
- glTexImage2D(target, level, internalFormat, width, height, border, format, type, texels)
- 이미지를 가져올 때는 openGL은 좌하단이 0,0이므로 y 방향으로 이미지를 한번 뒤집는다
- vertex와 마찬가지로 id를 들고 있을 texture_object를 generate하고 bind 해야한다
- texture space는 [0, 1]의 x, y 값 --> vertex 내부 texcoord를 raster가 interpolation
- Resampling --> approximation

- 얼마나 자세히 image를 샘플링 했는가, image는 raster이기 때문에~
- Aliasing : 서로 사이즈가 다른 image sampling에서 왜곡되는 것
- Magnification
  - texture가 커지는 경우, 텍스쳐의 한 셀(texel)이 raster에서 더 크게 매핑
  - nearest neighbor, bilinear interpolation, bicubic interpolation etc...
  - fragment가 texel 보다 많기 때문에 가장 가까운 texel에 선정하거나 또는 주위 탐색해서 적당한 값을 취하거나,,,
- Minification
  - 여러 texel들이 한 개의 fragment로 처리되는 경우
  - 축소하는 경우에 문제가 생긴다, 반드시 써야하는??(특징적인) texel이 제대로 반영이 안될 수 있기 때문에
  - 지글거리는 패턴이 나타나는 것을 막자
  - mipmapping
    - 미리 texel들을 섞어 놓는다(HW 수준에서 지원되어 빠르다, preintegration)
    - 해상도가 약간 떨어질 수 있음, blur 효과가 좀 있기 때문에~
    - summed-area tables
- Textured Shading
  - Blinn-Phong illumination model에서 diffuse reflector model
  - constant에서 texture에서 가져오는 것으로 바꾸기