20181005 그래픽스

Conceptual하게 window의 크기는 변하지만 실제로 현실에서 카메라 센서기준으로 윈도우 크기는 변한다!

Viewing pipeline

Mapping world to canonical -> clipping -> mapping …?

**Clipping in Viewing pipeline이 굉장히 중요하다고 하셨는데 집중을 못해서 하나도 이해하지 못했다!! Ch.4 page 16**

gluLookat에서 ex ey ez는 카메라의 위치, ex ey ez는 카메라가 바라보는 방향, ux uy uz는 카메라의 up vector인데 이 방향이 카메라의 윗방향이 되는듯하다. 이 것에 해당하는 transformation matrix가 stack에 쌓여있을 듯.

1. 모델을 디자인한다
2. 모델의 위치를 결정해주는 transformation matrix를 준다
3. 이제 이 모델을 어느방향에서 어떤 크기의 윈도우로 바라볼지 결정한다.
4. 그 윈도우를 디바이스의 어느위치에 나타낼지(viewport) 결정한다!

Identitiy

Gluortho2d

glulookat

Gluviewport

glRotatef 함수를 사용했을 경우 물체만 rotate시키는 것이 아니라 물체가 define되어있는 coordinate system 자체를 transform하기 때문에 물체를 위 아래로 움직이는 상황의 경우 transformation된 상황에서 그대로 움직인다.

glMatrixMode(GL\_MODELVIEW/GL\_PROJECTION)

위 함수를 사용하면 각각의 transformation matrix가 다른 stack에 쌓여서 각각 곱해지는 순서를 predefine해주기 때문에 순서를 크게 신경쓰지 않고 적용시켜도 된다! gluOrtho2D나 gluLookAt 같은 함수들을 !

M3 \* M2 \* M1 \* P

M1은 transformation, M2는 3D->2D projection(glulookat), M3는 window를 viewport에 띄워주는것??(ortho2d and viewport??)-> viewport까지 포함되어있는지는 확실하지 않긴함

교수님이 적기로는

1. ModelView
2. Projection
3. Viewport라고 적긴했다

이제 programmable shader를 다루는 듯하다!

지금까지 배운건 opengl의 구버전이다! Shader가 포함되어있지 않은1

따라서 이제 shader가 포함된 신버전을 배워야할것이다!!

두번째 어싸인은 shader를 이용한 것이 될 것이다!!

Opengl Architecture에서 geometry processor는 이제까지의 modelview to device까지의 변환을 계산해준다. 그런 다음 rasterizer는 이것을 픽셀로 바꾸어준다. 그런다음 Fragment Processor에서 depth test를 해서 어떤 object가 앞에 있을지 결정해주는것같다!!