Transformation 정리

Category of transformation: linear / affine(평행) / perspective(=projective, 평행 아님)

Linear Transformation

-rotation/translation/scaling/shear

-linear algebra equation -> Matrix\*vector

-Vector Shader에서 transform이 일어남

**Translation**

X’=x+dx

Y’=y+dy

Z’=z+dz

T=[1,0,0,0, 0,1,0,0, 0,0,1,0, dx,dy,dz,1]

v=[x,y,z,w] //vector -> **homogeneous coordinat**e (w는 초기값 =1 ) w는 계산을 한 뒤에 맨 나중에 1로 바꾼다. 최종적으로 좌표계를 scaling하는 역할이다.

T\*v=[ x+dx, y+dy, z+dz, 1] //vector

Q: homogeneous coordinate를 사용하는 이유는?

A: 다수의 행렬의 곱셈은 결합법칙이 성립한다. 따라서 여러 복잡한 변환 행렬이 있을 때 행렬을 모두 곱해서 복합 행렬로 하나를 만들어 놓는다. 그러면 처리해야 할 vertex가 많을 때 하나의 복합 행렬만 곱하면 되므로 성능의 효율성을 가져올 수 있다.

**Scaling**

X’=Sx\*X

Y’=Sy\*Y

Z’=Sz\*Z

P’=Sp

물체를 크게 키우는게 아니라 원점에서 멀리 이동시키는 것

원점으로 가져가서 scaling 하고 다시 이동

S=S(Sx,Sy,Sz)=[Sx,0,0,0, 0,Sy,0,0, 0,0,Sz,0, 0,0,0,1]

S\*v

Reflection

X반전 -> Sx=1 Sy=-1

Y반전 -> Sx=-1 Sy=1

X,Y 반전 -> Sx=Sy=-1

**Rotation**

X’=x\*cos(theta)-y\*sin(theta)

Y’=x\*sin(theta)+y\*cos(theta)

Z’=z

//z-rotate

Righted-handed coordinate

R=Rz(theta)=[cos(theta),sin(theta),0,0, -sin(theta),cos(theta),0,0, 0,0,1,0, 0,0,0,1]

R=Rx(theta)=[1,0,0,0, 0,cos(theta),sin(theta),0, 0,-sin(theta),cos(theta),0, 0,0,0,1]

R=Ry(theta)=[cos(theta),0,-sin(theta),0, 0,1,0,0, sin(theta),0,cos(theta),0, 0,0,0,1]

**Shear -> affine translation**

기우는 것

X’=x+y\*cot(theta)

Y’=y

Z’=z

H(theta)=[1,0,0,0, cot(theta),1,0,0, 0,0,1,0, 0,0,0,1]

**Inverse**

Translation: T-1(dx,dy,dz) = T(-dx,-dy,-dz) 그냥 마이너스만 붙이면 됨!!

T-1=[1,0,0,0, 0,1,0,0, 0,0,1,0, -dx,-dy,-dz,1]

Rotation: R-1(q) = R(-q) theta에 마이너스만 붙이면 됨

(cos(-theta)=cos(theta) / sin(theta)=-sin(theta))

R-1 z(theta)=[cos(theta),-sin(theta),0,0, sin(theta),cos(theta),0,0, 0,0,1,0, 0,0,0,1]

R-1 x(theta)=[1,0,0,0, 0,cos(theta),-sin(theta),0, 0,sin(theta),cos(theta),0, 0,0,0,1]

R-1 y(theta)=[cos(theta),0,sin(theta),0, 0,1,0,0, -sin(theta),0,cos(theta),0, 0,0,0,1]

Scaling: S-1(Sx,Sy,Sz) = S(1/Sx,1/Sy,1/Sz)

S-1=[1/Sx,0,0,0, 0,1/Sy,0,0, 0,0,1/Sz,0, 0,0,0,1]

Order of transformation

M=ABC -> M-1=C-1B-1A-1

Mv=vTMT

역행렬은 그냥 순서만 반대로 하면 된다.

(rotationM \* translateM)-1 = translateM \* rotationM

* 회전하고 이동의 역행렬은 다시 원점으로 이동하고 회전

특정한 지점에서의 회전 M=T(pf) R(pf) T(-pf)

1. T(-pf) : 원점으로 가서 2. R(pf): 회전하고 3. T(pf): 특정한 지점으로 이동

## drawarray 에서 가까운 것부터 실행됨!

mat4.fromYRotation ( mMat,rotY);

mat4.translate (mMat,mMat,[0.7,0.7,0.7,0.0]);

gl.drawArray(gl.TRIANGLES,0,36);

* Translate하고 rotation 하므로 큐브가 공전함 (M = ID \* Mrt \* Mtr)

mat4.translate (mMat,mMat,[0.7,0.7,0.7,0.0]);

mat4.fromYRotation ( mMat,rotY);

gl.drawArray(gl.TRIANGLES,0,36);

* Rotation하고 translate하므로 큐브가 자전 (M = ID \* Mtr \* Mrt)