6/6/2011

3D Basic & OpenGLES 2.0

thuy.vuthiminh@gameloft.com

phong.caothai@gameloft.com

khiem.tranthien@gameloft.com

tam.la@gameloft.com

Content

- Introduction
- Rendering pipeline
- Shader
- Basic GLSL-ES
- Basic Math
- MVP matrices
- Textures
- Obj model
- Shader effect: Skydome using cube mapping

Basic knowledge in Opengles 2.0?

Content

Basic Math

- Point
- Vector
- Matrix
- Transformation (affine)
- Homogeneous coordinate
- Combination transformation

Basic Math concept

Basic Math

- **Point**
- Vector
- Matrix
- Transformation (affine)
- Homogeneous coordinate
- Combination transformation

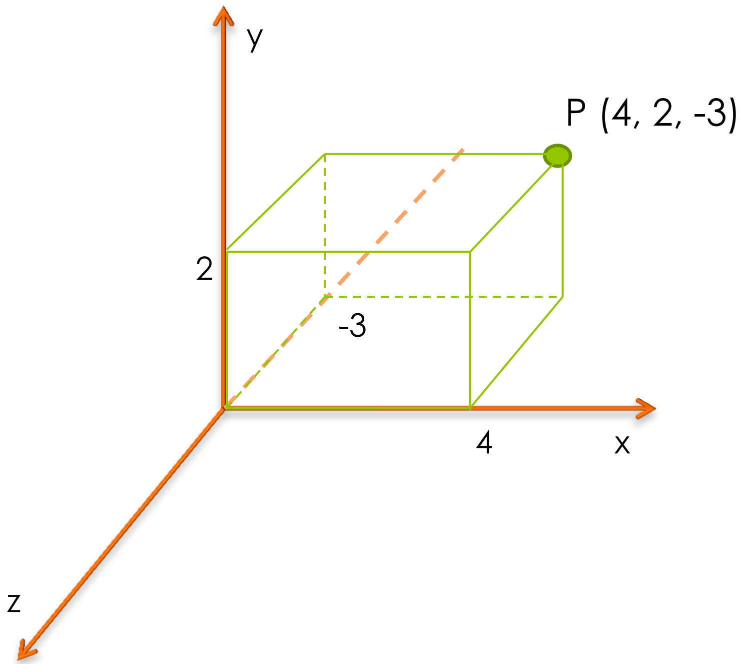
Basic Math concept

Point

- Position of a point P(px, py, pz)

$$P = \begin{bmatrix} Px \\ Py \\ Pz \end{bmatrix}$$

- Right – hand coordination system

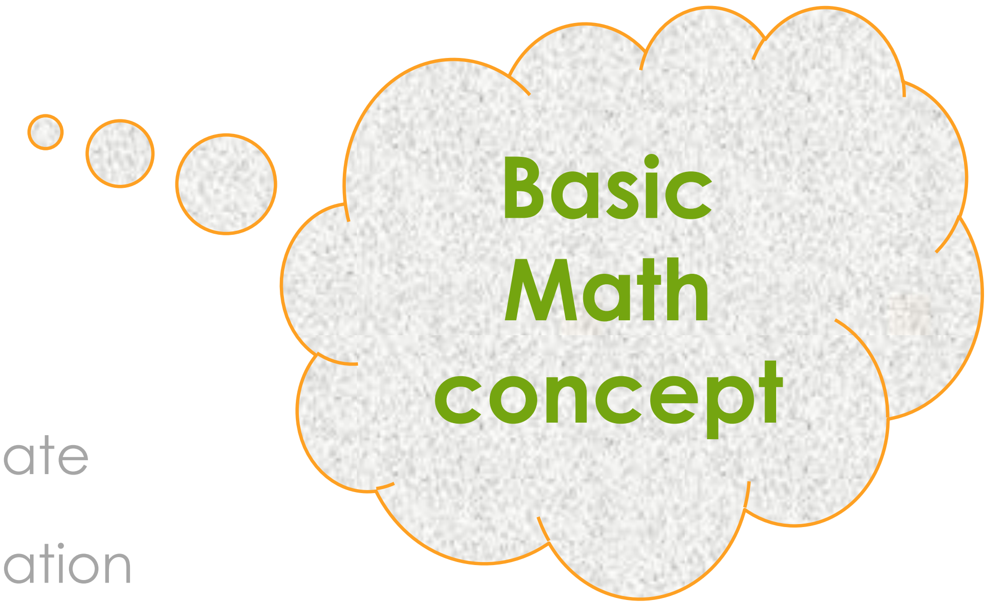


<u>Pratice:</u>

Draw point P(5, 3, 4) in Opengl coordination system

Basic Math

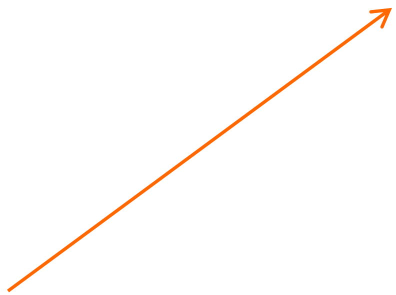
- Point
- **Vector**
- Matrix
- Transformation (affine)
- Homogeneous coordinate
- Combination transformation



Vector

- **<u>Length (module)</u>**

$$\|\vec{v}\| = \sqrt{x^2 + y^2 + z^2}$$



$\|\vec{v}\| == 1 \rightarrow x^2 + y^2 + z^2 == 1 \rightarrow$ unit vector

- **<u>Normalization</u>**

Call $\vec{N}$ is normalize vector of $\vec{v}$

$\vec{N}$ ($\vec{v}$ (x, y, z)) = $\overrightarrow{v'}$ (x', y', z')

$$x' = \frac{x}{\|\vec{v}\|} = \frac{x}{\sqrt{x^2+y^2+z^2}} \qquad z' = \frac{z}{\|\vec{v}\|} = \frac{z}{\sqrt{x^2+y^2+z^2}}$$

$$y' = \frac{y}{\|\vec{v}\|} = \frac{y}{\sqrt{x^2+y^2+z^2}}$$

Vector: Dot Product

Dot product

- $\vec{a}$ dot $\vec{b}$ = $\|\vec{a}\|$ * $\|\vec{b}\|$ * cos(θ)

$\|\vec{a}\|$ = 1

$\|\vec{b}\|$ = 1

$\vec{a}$ dot $\vec{b}$ = cos(θ)

θ = 0 cos(θ) = 1 ($\vec{a}$ dot $\vec{b}$) get Max = $\|\vec{a}\|$ * $\|\vec{b}\|$

θ = 90 cos(θ) = 0 ($\vec{a}$ dot $\vec{b}$) get Min = 0

- Result is a scalar
- Implement:

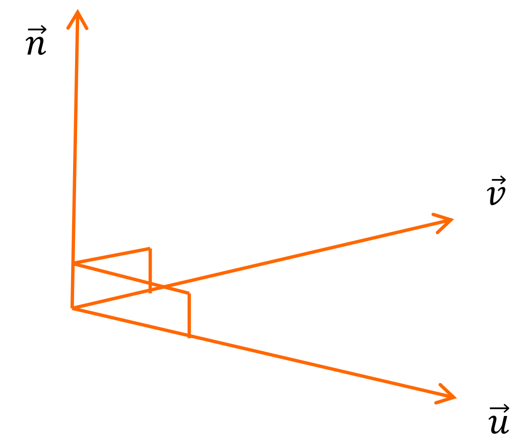
$\vec{a}\,(a1, a2, a3)$ dot $\vec{b}(\mathrm{b1}, \mathrm{b2}, \mathrm{b3})$ = a1*b1 + a2*b2 + a3*b3

Vector: Cross Product

- A binary operation between two vectors.
- Result is a third vector orthogonal to 2 first vectors.

$\overrightarrow{v1}$(x1,y1,z1) x $\overrightarrow{v2}$(x2,y2,z2) = $\overrightarrow{v3}$ (x3,y3,z3)

$$\begin{cases} x3 = y1 * z2 - z1 * y2 \\ y3 = z1 * x2 - x1 * z2 \\ z3 = x1 * y2 - y1 * x2 \end{cases}$$



- The cross product is anti-commutative

$\vec{a}$ x $\vec{b}$ = - $\vec{b}$ x $\vec{a}$

Practice 5.1

- Assume

$\overrightarrow{v1}$(5, -1, 7) and $\overrightarrow{v2}$(4, 2, 3)

Calc $\overrightarrow{v3} = \overrightarrow{v1} \times \overrightarrow{v2}$

- Result?

<u>Result:</u>
X3 = -17
Y3 = 13
Z3 = 14

Basic Math

- Point
- Vector
- **Matrix**
- Transformation (affine)
- Homogeneous coordinate
- Combination transformation

Basic Math concept

Matrix: **Addition & Multiplication**

- Addition of the same size

A[nxm] + B[nxm] = C[nxm]

Ex: $A\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} + \mathrm{B}\begin{bmatrix}2 & 0\\0 & 2\end{bmatrix} = C\begin{bmatrix}3 & 0\\0 & 3\end{bmatrix}$

- Direct addition

A[nxm] ⊕ B[pxq] = C[n+p, m+q]

$$\mathrm{A}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \oplus \mathrm{B}\begin{bmatrix}2 & 0 & 0\\0 & 2 & 0\\0 & 0 & 0\end{bmatrix} = \mathrm{C}\begin{bmatrix}1 & 0 & 0 & 0 & 0\\0 & 1 & 0 & 0 & 0\\0 & 0 & 2 & 0 & 0\\0 & 0 & 0 & 2 & 0\\0 & 0 & 0 & 0 & 2\end{bmatrix}$$

Matrix: Addition & Multiplication (conts)

- Direct addition

$$C[i,j] = \begin{cases} A[i,j]\ \{ i = 1 \ldots n,\ j = 1 \ldots m\} \\ B[i,j]\ \{i = n+1 \ldots p+n,\ j = m+1 \ldots q+m\} \end{cases}$$

- Multiplication

A * B = C ; $C_{ij}=\sum_{1}^{n} A_{ik} \times B_{kj}$

Matrix: Minor & Cofactor

Minor $M_{ij} = \begin{vmatrix} \square & \cdots & \square \\ \cdots & x & y \\ \square & z & w \end{vmatrix}$ = (x*w – z*y)

Cofactor $C_{ij} = (-1)^{i+j} * M_{ij} = (-1)^{i+j} \begin{vmatrix} \square & \cdots & \square \\ \cdots & x & y \\ \square & z & w \end{vmatrix} = (-1)^{i+j}$ (x*w – z*y)

A = $\begin{pmatrix} 1 & 4 & 7 \\ 3 & 0 & 5 \\ -1 & 9 & 11 \end{pmatrix}$

Practice 5.2:
calculate C_{12} ?

Result:
-38

$C_{23} = (-1)^{2+3} \begin{vmatrix} 1 & 4 & \square \\ \square & \square & \square \\ -1 & 9 & \square \end{vmatrix}$ = (-1) * (9 – (-4)) = -13

Matrix: Determinant

- Give matrix A(nxm) (n = m)

Det(A) = $\sum_{j=1}^{m} a_{1j} * C_{1j}$

$$= \sum_{j=1}^{m} a_{1j} * (-1)^{1+j} * M_{1j}$$

- A = $\begin{pmatrix} 1 & 4 & 7 \\ 3 & 0 & 5 \\ -1 & 9 & 11 \end{pmatrix}$

Practice 5.3:
Find det A?

Result:

Det(A) = 1 * M_{11} - 4 * M_{12} + 7 * M_{13}

= 1* (-45) – 4*38 + 7*27 = -8

Matrix: Transpose

- Give A(nxm) $\begin{bmatrix} a_{11} & a_{12} & a_{1m} \\ a_{21} & \square & \square \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}$

A^t is transpose of A $\leftrightarrow$ A^t(mxn) $\begin{bmatrix} a_{11} & a_{21} & a_{n1} \\ a_{12} & \square & \square \\ \vdots & \ddots & \vdots \\ a_{1m} & \cdots & a_{nm} \end{bmatrix}$

- Ex:

A(3x4) $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 3 & 4 \end{bmatrix} \rightarrow A^t(4x3) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 1 & 3 \\ 1 & 0 & 4 \end{bmatrix}$

Matrix: Diagonal Matrix & Identity Matrix

- B is diagonal matrix means:

$$B = \begin{bmatrix} b_{11} & 0 & 0 & 0 \\ 0 & b_{22} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & b_{nn} \end{bmatrix}$$

$$b_{ii} = 1 \;\rightarrow \text{Identity Matrix } I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix: Inverse of a Matrix

- Inverse of Matrix A (noted A^{-1})

$$\boldsymbol{A} \;\; * \;\; \boldsymbol{A}^{-1} = \boldsymbol{I}$$

- How to calculate A^{-1} of A(nxm) ?

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} C_{11} & C_{21} & \cdots & C_{n1} \\ C_{12} & C_{22} & \cdots & C_{n2} \\ \cdots & \cdots & \ddots & \cdots \\ C_{1m} & C_{2m} & \cdots & C_{nm} \end{bmatrix}$$

Practice 5.4:

Assume A $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix}$. Find A^{-1}

Result:

$$\frac{1}{6}\begin{bmatrix} 6 & -3 & -1 \\ 0 & 3 & -1 \\ 0 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{6} \\ 0 & \frac{1}{2} & -\frac{1}{6} \\ 0 & 0 & \frac{1}{3} \end{bmatrix}$$

Basic Math

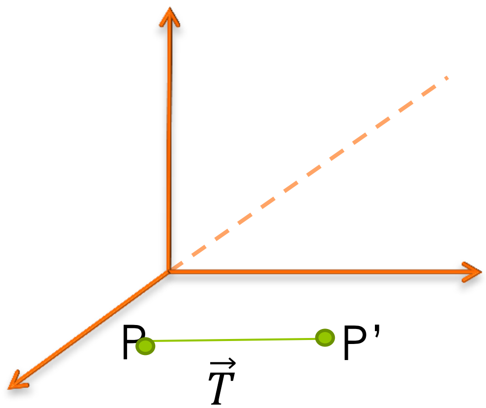
- Point
- Vector
- Matrix
- **Transformation (affine)**
- Homogeneous coordinate
- Combination transformation

Basic Math concept

Transformation: Translate, Scale

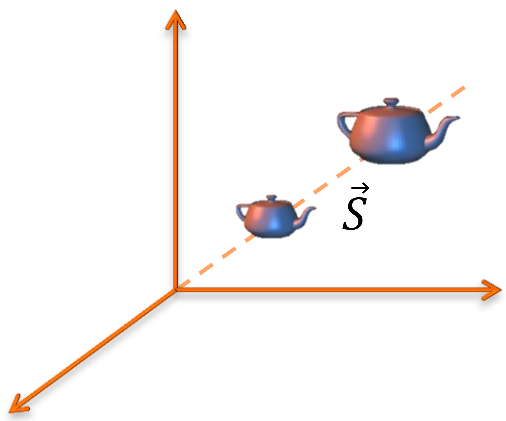
- Translate operator:

$$P' = P + \vec{T} \leftrightarrow \begin{bmatrix} Px' \\ Py' \\ Pz' \end{bmatrix} = \begin{bmatrix} Px + tx \\ Py + ty \\ Pz + tz \end{bmatrix}$$



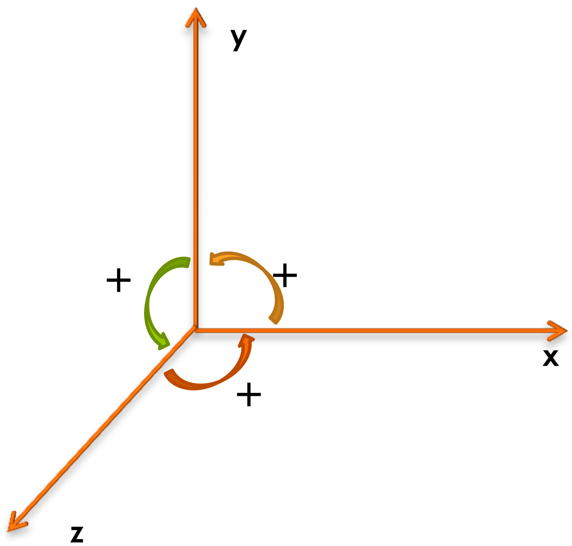
- Scale Operator

$$P' = P * \vec{S} \leftrightarrow \begin{bmatrix} Px' \\ Py' \\ Pz' \end{bmatrix} = \begin{bmatrix} Px * sx \\ Py * sy \\ Pz * sz \end{bmatrix}$$



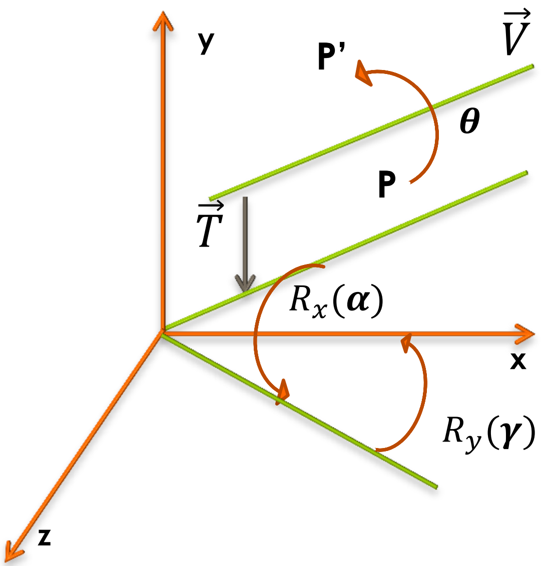
Transformation: Rotate operator

- Principle axis



Axis	Positive angle
Oz	X → y
Oy	Z → x
Ox	Y → z

- Arbitrary axis



How to rotate in an arbitrary axis?

$$R(\boldsymbol{\theta}) = T^{-1} . R_x{}^{-1} . R_y{}^{-1} . R_x . R_y . R_x . T$$

Transformation: Principle Rotate

- Suppose P' is result from rotating P through an angle θ (image 1)

$\overrightarrow{P'} = \vec{P} + \vec{Q}$

$= \|P'\| \cos\theta + \|P'\| \sin\theta$

$= \|P\| \cos\theta + \|P\| \sin\theta$

- Beside: P(x, y) = Q(-y, x) (image 2)
- Finally,

$P'_x = P_x \cos\theta - P_y \sin\theta$

$P'_y = P_y \cos\theta + P_x \sin\theta$

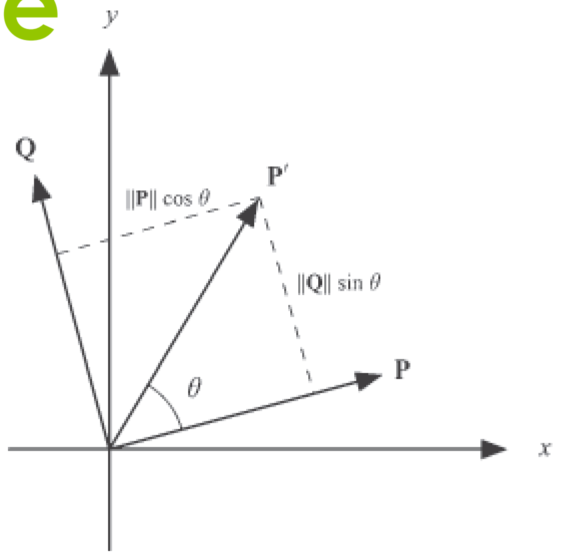


Image 1: P' as a linear combination of P and Q

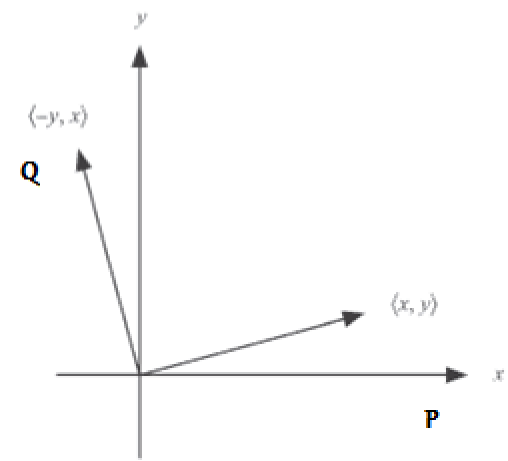


Image 2 : Rotation by 90 degrees in the x-y plane

Transformation: Principle Rotate

With R is a matrix:

- Ox

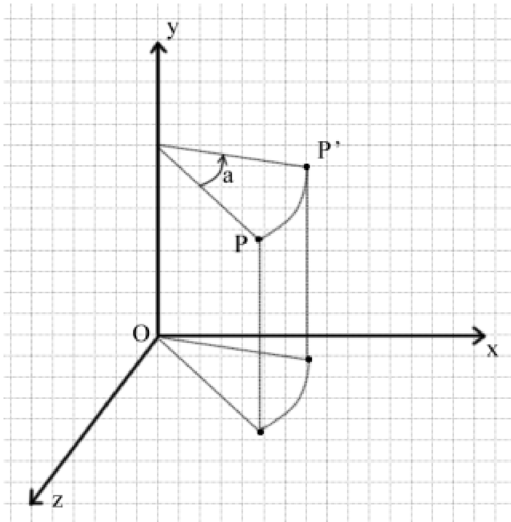
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\mathrm{a}) & -\sin(a) \\ 0 & \sin(a) & \cos(a) \end{bmatrix}$$

- Oy

$$\begin{bmatrix} \cos(a) & 0 & \sin(a) \\ 0 & 1 & 0 \\ -\sin(a) & 0 & \cos(a) \end{bmatrix}$$

- Oz

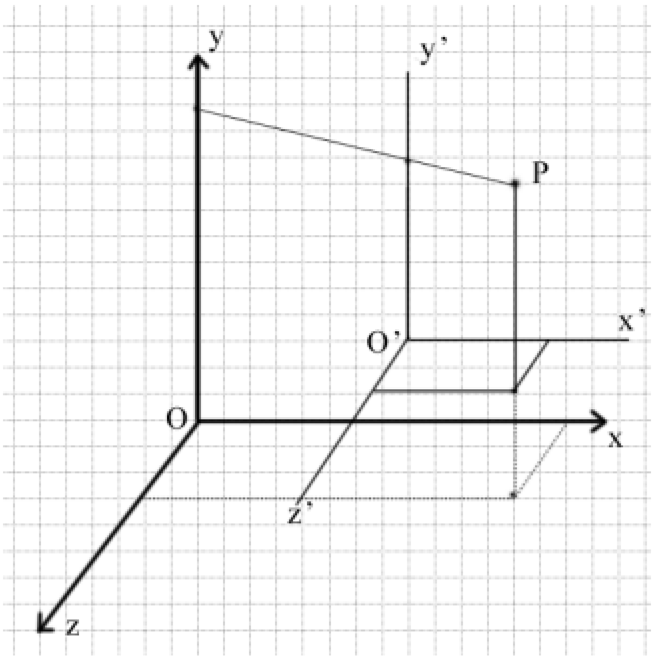
$$\begin{bmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\mathrm{P}' = R * P \quad \leftrightarrow \begin{bmatrix} Px' \\ Py' \\ Pz' \end{bmatrix} = R * \begin{bmatrix} Px \\ Py \\ Pz \end{bmatrix}$$

Coordination transformation

- 2 kinds of transformation:
  - Objects
  - Coordination
- Coordinate transformation is an invertible affine transformation



Basic Math

- Point
- Vector
- Matrix
- Transformation (affine)
- **Homogeneous coordinate**
- Combination transformation

Basic Math concept

Homogeneous coordinate

- Why need homogeneous coordinate?
  - *Use both "add" and "multiply" operator!*
- To unique operator → Homogeneous coordinate

Homogeneous coordinate (conts)

- Point or vector $\begin{bmatrix} Px \\ Py \\ Pz \\ Pw \end{bmatrix}$

§ For point pw != 0

$$\begin{bmatrix} Px \\ Py \\ Pz \\ Pw \end{bmatrix}$$

- Pw = 1
- If (Pw != 1), divide to Pw

$$\begin{bmatrix} Px/Pw \\ Py/Pw \\ Pz/Pw \\ Pw/Pw \end{bmatrix} \longrightarrow \begin{bmatrix} Px/Pw \\ Py/Pw \\ Pz/Pw \\ 1 \end{bmatrix}$$

§ For vector pw = 0

$$\begin{bmatrix} Px \\ Py \\ Pz \\ 0 \end{bmatrix}$$

Homogeneous coordinate: Translate & Scale

P' = M * P

- Translate

$$\begin{bmatrix} Px' \\ Py' \\ Pz' \\ Pw' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} Px \\ Py \\ Pz \\ Pw \end{bmatrix}$$

- Scale

$$\begin{bmatrix} Px' \\ Py' \\ Pz' \\ Pw' \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} Px \\ Py \\ Pz \\ Pw \end{bmatrix}$$

Homogeneous coordinate: Rotate

P' = M * P

- Ox

$$\begin{bmatrix} Px' \\ Py' \\ Pz' \\ Pw' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(a) & -\sin(a) & 0 \\ 0 & \sin(a) & \cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} Px \\ Py \\ Pz \\ Pw \end{bmatrix}$$

- Oy

$$\begin{bmatrix} Px' \\ Py' \\ Pz' \\ Pw' \end{bmatrix} = \begin{bmatrix} \cos(a) & 0 & \sin(a) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(a) & 0 & \cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} Px \\ Py \\ Pz \\ Pw \end{bmatrix}$$

- Oz

$$\begin{bmatrix} Px' \\ Py' \\ Pz' \\ Pw' \end{bmatrix} = \begin{bmatrix} \cos(a) & -\sin(a) & 0 & 0 \\ \sin(a) & \cos(a) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} Px \\ Py \\ Pz \\ Pw \end{bmatrix}$$

Practice 5.5

Assume that P(10, -7, 6). Find P' by

a) Moving along T(-1, 4, 6)

b) Scaling S(0.5, 1, 1.5)

c) Rotate around Ox 90^o

Answer?

a) P'(9, -3, 12)
b) P'(5, -7, 9)
c) P'(10, -6, -7)

Basic Math

- Point
- Vector
- Matrix
- Transformation (affine)
- Homogeneous coordinate
- **Combination transformation**

Basic Math concept

Combine Transformation

- In general P' = M*P

With M: combination transformation matrix

- Getting by multipling separated affine matrix

M = $T_1 * S_1 * R_1 * \ldots$

- Matrix multiplication is not commutative

$T_1 * S_1 \mathrel{!=} S_1 * T_1$

T_1(nxm) * S_1(mxp) → legal

T_1(nxm) * S_1(pxq) → illegal

Combine Transformation (conts)

- Stack of transformation is inverted

Assume we have point P → Translate T → Scale S

P' = T * P

P'' = S * P' = S * T * P = S * T * P

Combine Transformation

<u>Example:</u>

Point P(1, 0, 0) → Move along (1, 0, 0) → Rotate(Oz, 90) → P'(0, 2, 0)

$$M_1 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_1 = M_1 * P = (1, 1, 0) \rightarrow Wrong$$

$$P_2 = M_2 * P = (0, 2, 0) \rightarrow Right$$

Practice 5.6

Assume that P(10, -7, 6). Find P' by

- Rotate R(Ox, 90) → Translate T(-10, -1, -6)
- Rotate R(Ox, 90) → Translate T(-10, -1, -6) → Rotate R(Ox, 90)
- Translate T(10, 1, 6) → Rotate (Ox, 90) → Translate (-10, -1, -6)

Answer?

Content

MVP Matrix

- What is MVP matrix
- Model
- View
- Projection
- Practice
- References

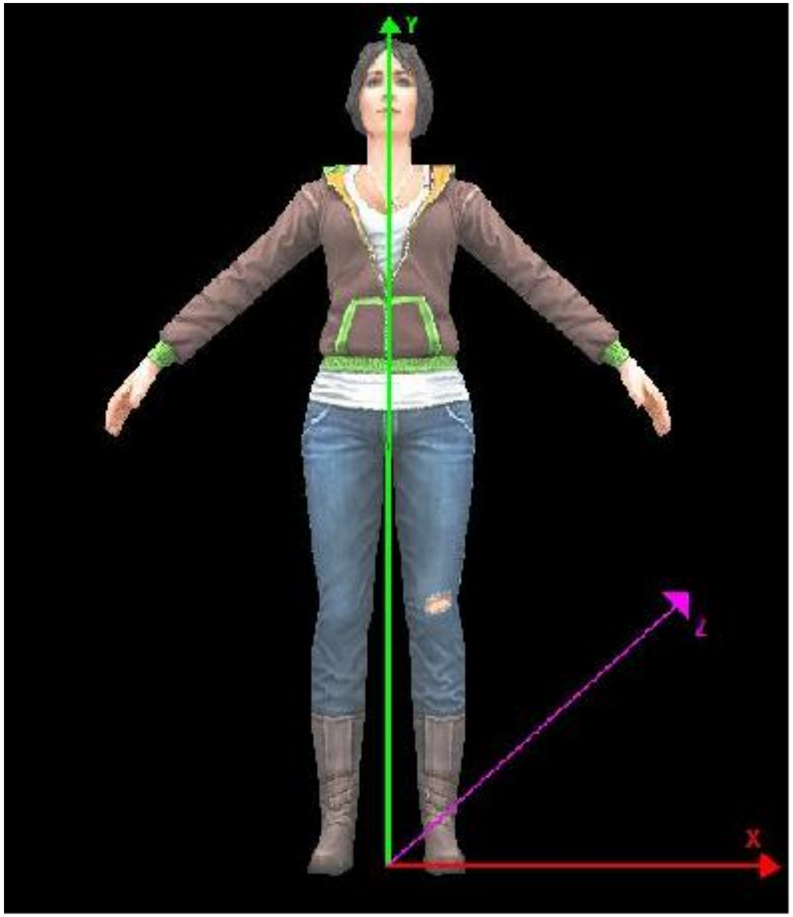
MVP Matrix

- **What is MVP matrix?**
  - MVP matrix is Model View Projection matrix.
  - MVPMatrix = ProjectionMatrix * ViewMatrix * ModelMatrix

MVP Matrix: Model

Object space

- All objects are in the object space, which means they will have the pivot in (0,0,0,1)

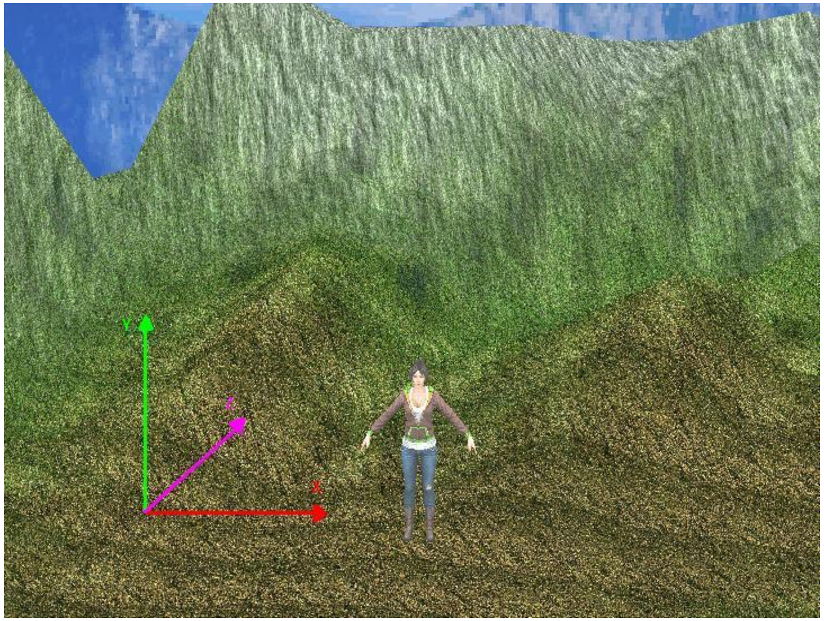


MVP Matrix: Model

Object space in View port

- Want to put objects at desired location in 3D world?

→ Appropriated scale, translate and rotation!

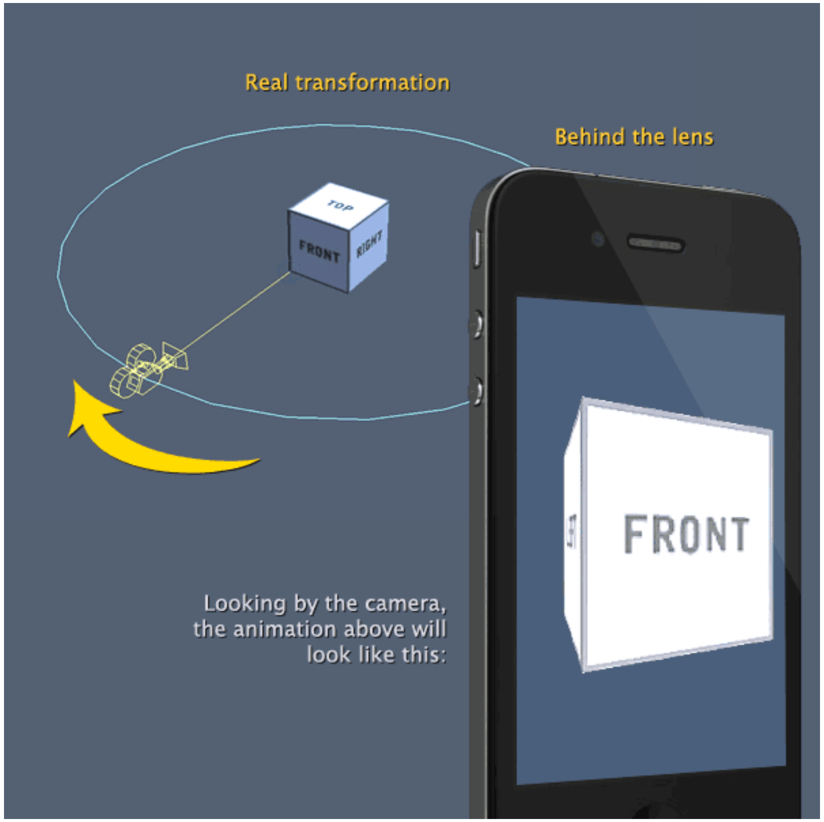


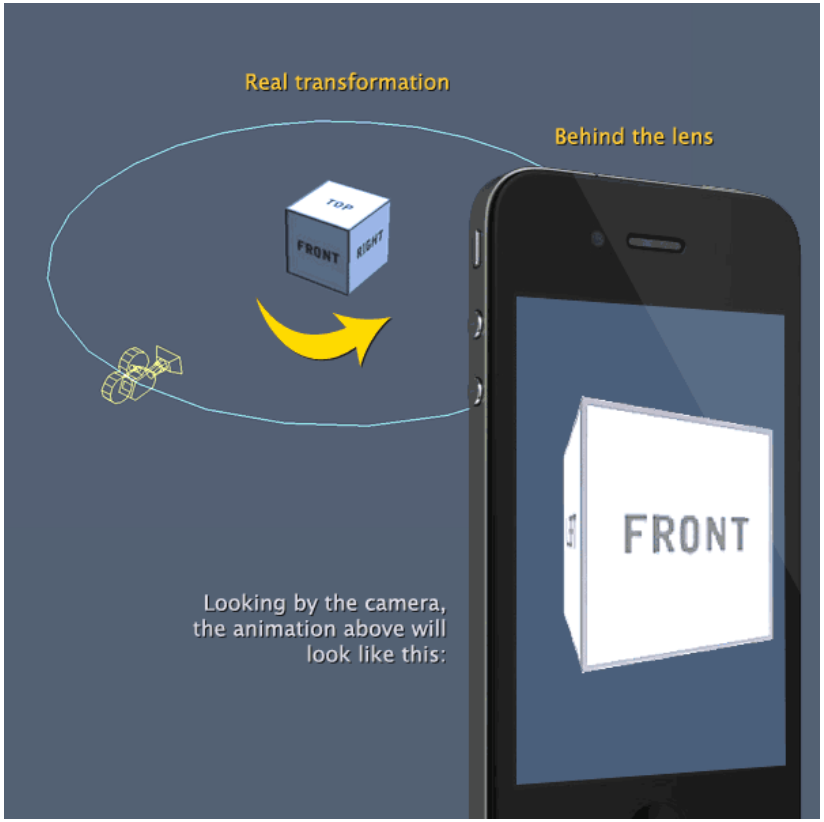
- To bring object from local space to work space:

World Matrix * Object

MVP Matrix: View

Rotate Camera vs Rotate Object:





MVP Matrix: View

Now is:
View Matrix * World Matrix * Object

- Want to move all objects as little as possible?

→ Need Camera

- Camera object has its own world matrix.
- To make the world be relative to the camera's location called View Matrix → multiply the object with the inverse of the camera's world matrix (View Matrix)

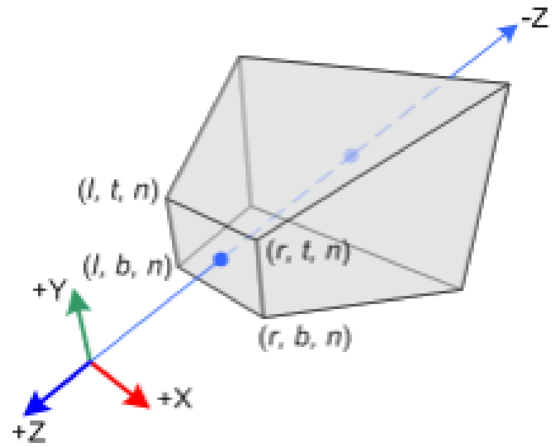
MVP Matrix: Projection

- 2 ways to define frustum (perspective projection volume)

1. 6 planes of frustum
2. FOV + nearPlan + farPlan

❶ 6 planes of frustum (left, right, near, far, top, bottom)

- Near Plane: any geometry closer to the camera will be clipped.
- Far Plane : any geometry beyond this plane will be clipped.



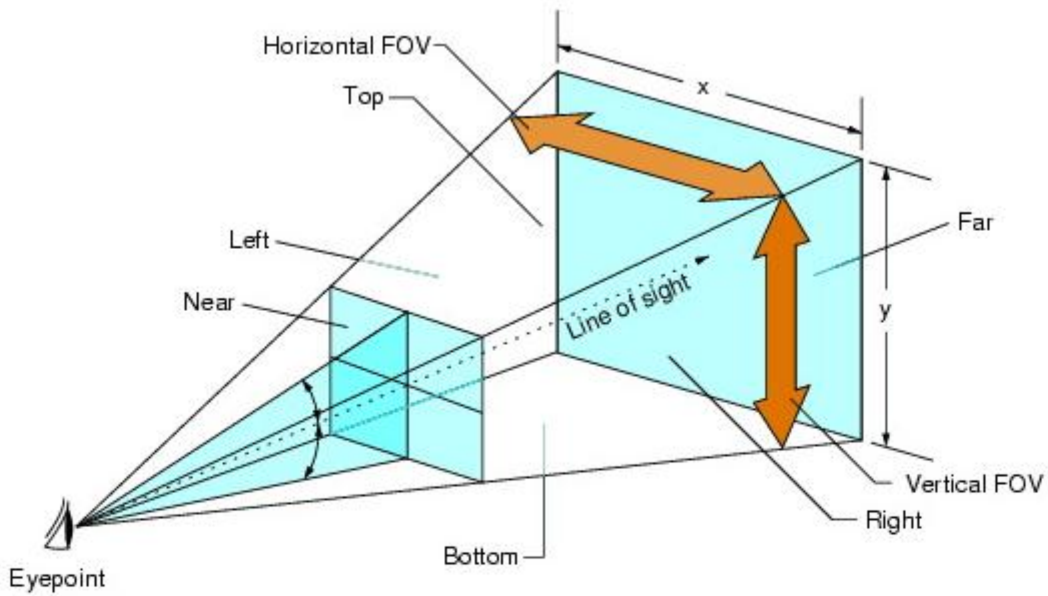
Perspective volume (Frustum)

MVP Matrix: Projection (conts)

❷ FOV (fovy, aspect, near, far)

Ex: FOV (45, $^4/_3$, 1, 10000)

- Fovy: the camera will open 45 x 2= 90 degrees
- Aspect: All images display with aspect ratio is $^{SCREEN\_WIDTH}/_{SCREEN\_HEIGHT}$ of render screen



MVP Matrix: Perspective Projection

- We will skip the math and we'll just present the resulting matrix form:

$$\text{Projection Matrix} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

- Where:
  - r = right
  - l = left
  - t = top
  - b = bottom
  - n = near
  - f = far

Now is:
Projection Matrix * View Matrix * World Matrix * Object

MVP Matrix

- Finally, formula to render a scene with regards to the viewpoint (camera) and the perspective projection:
  - In a right handed coordinate system, transposed matrix:

```
Final(x,y,z,w) = ProjMatrix*ViewMatrix*WorldMatrix*Initial(x,y,z,w);
```

- Notice that all the matrices have to be changed accordingly
- The final position will be in Homogenous Clip space

MVP Matrix: Coding

```
GLint mvpMatrixLoc =
glGetUniformLocation(programHandle,"u_mvpMatrix");

Matrix maProjection, maView, maModel; //Matrix is declared by 2 dimensions array

The value of matrix???

Matrix maMvpMatrix = m_maProjection * m_maView * m_maModel;

maMvpMatrix.MakeTranspose(); //transpose matrix

float *fMvpMatrix = maMvpMatrix.ToArray(); // convert to 1 dimension array

glUniformMatrix4fv(mvpMatrixLoc, 1, GL_FALSE, fMvpMatrix );

 …

 glVertexAttribPointer(…);

 glDrawArrays ( GL_TRIANGLES, …);
```

MVP Matrix: Practice

- Assume triangle $\begin{Bmatrix} 0.0f & 0.0f & 0.0f \\ 0.0f & 4.0\,f & 0.0f \\ 4.0f & 3.0f & 0.0f \end{Bmatrix}$

- Find P’ in 3D world of Opengl coordination system?

References

- http://www.songho.ca