Game Math

Ken-Yi Lee

Game Programming, Fall 2020 @ National Taiwan University

FeisStu

Game Programming

- Rendering
- Looping and control
- Math
- Behaviour and navigation (AI)
- Physics
- Animation and effects
- Networking

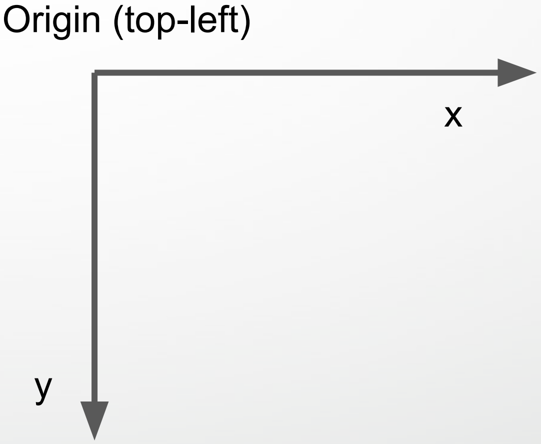
FeisStu

Game Programming

- Rendering
- Looping and control
- Math
- Behaviour and navigation (AI)
- Physics
- Animation and effects
- Networking

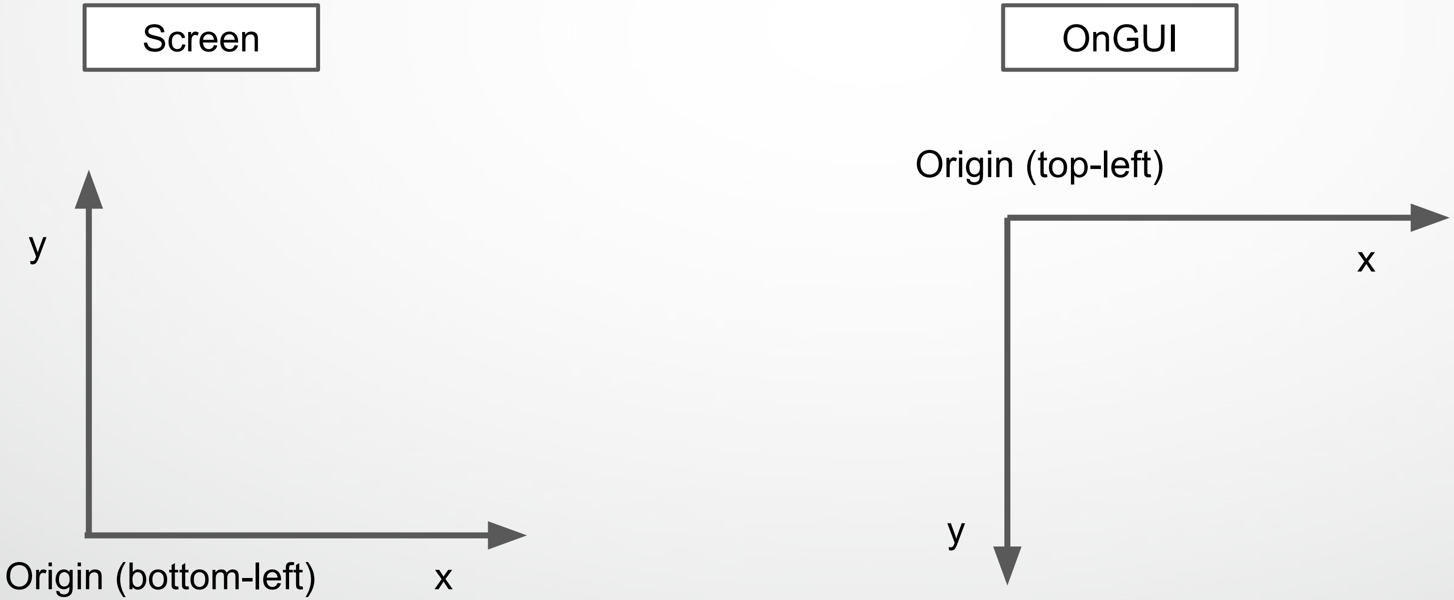
FeisStu

2D Coordinates Systems

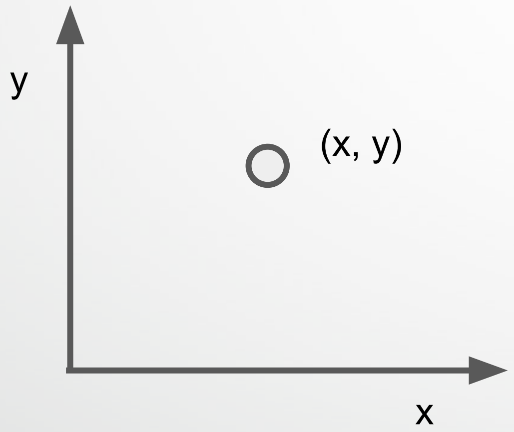


FeisStu

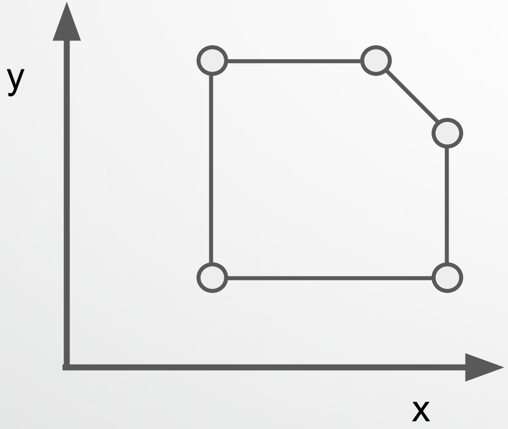
Screen and OnGUI coordinates



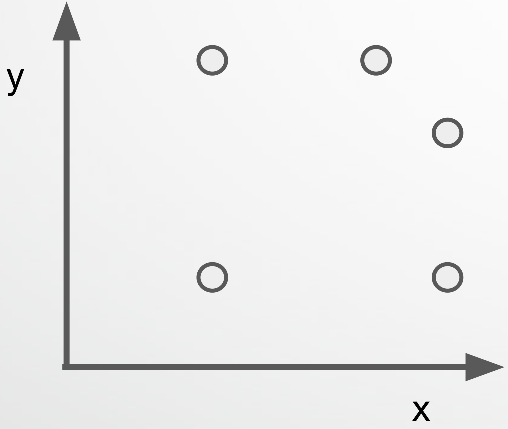
2D Point



2D Object (Mesh)

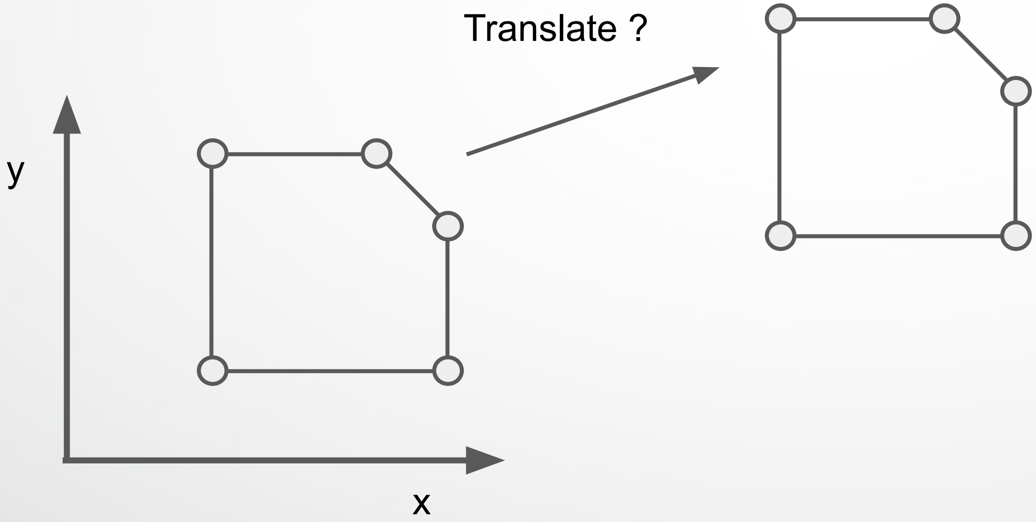


2D Object (vertices only)

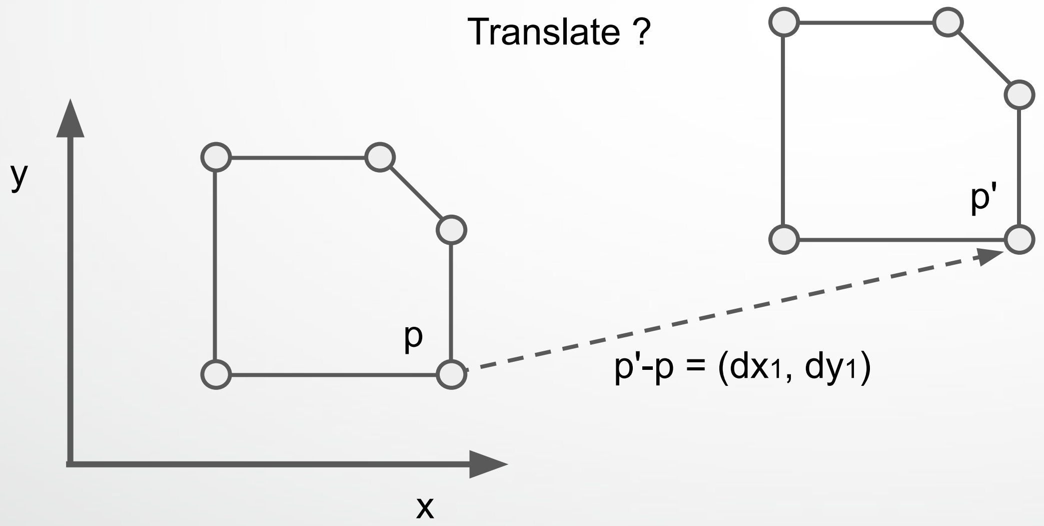


FeisStu

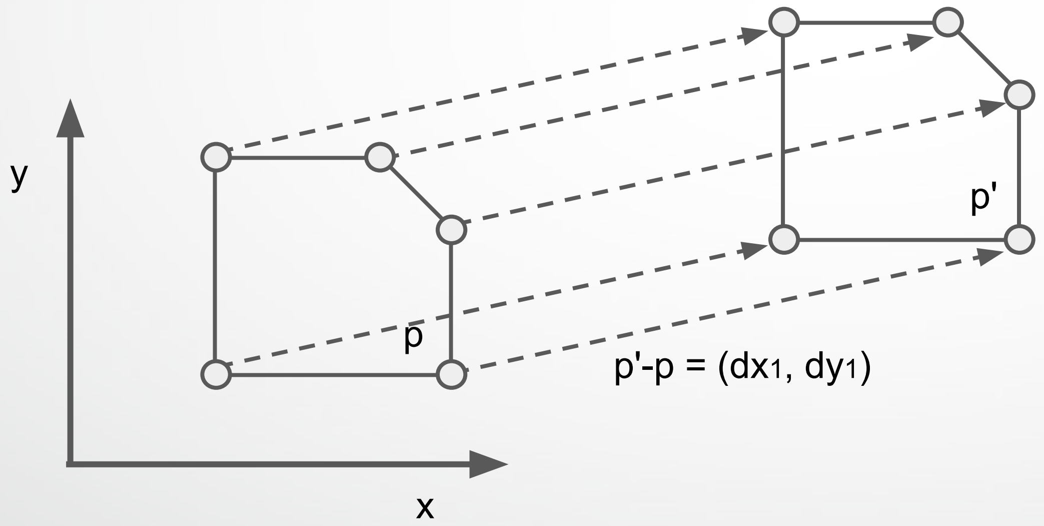
2D Translation



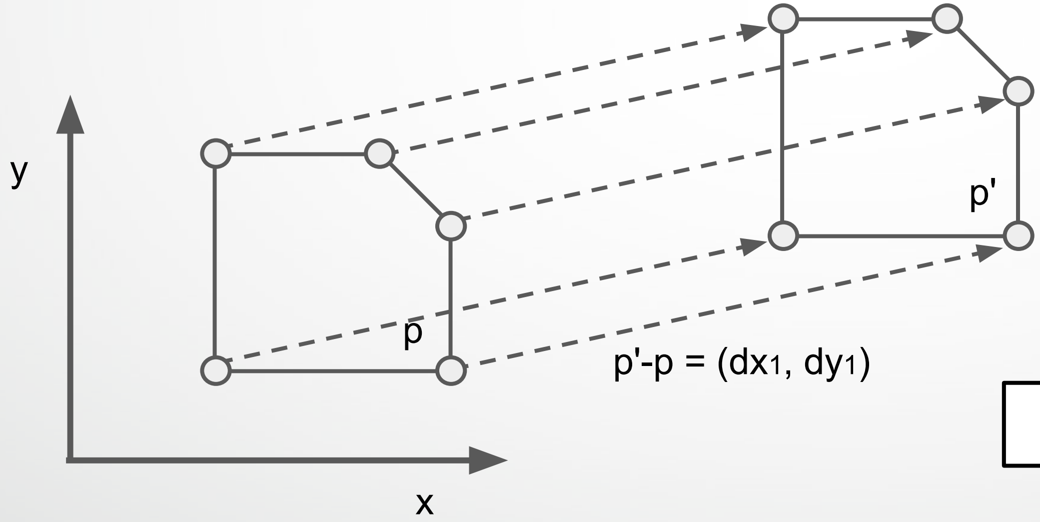
2D Translation



2D Translation

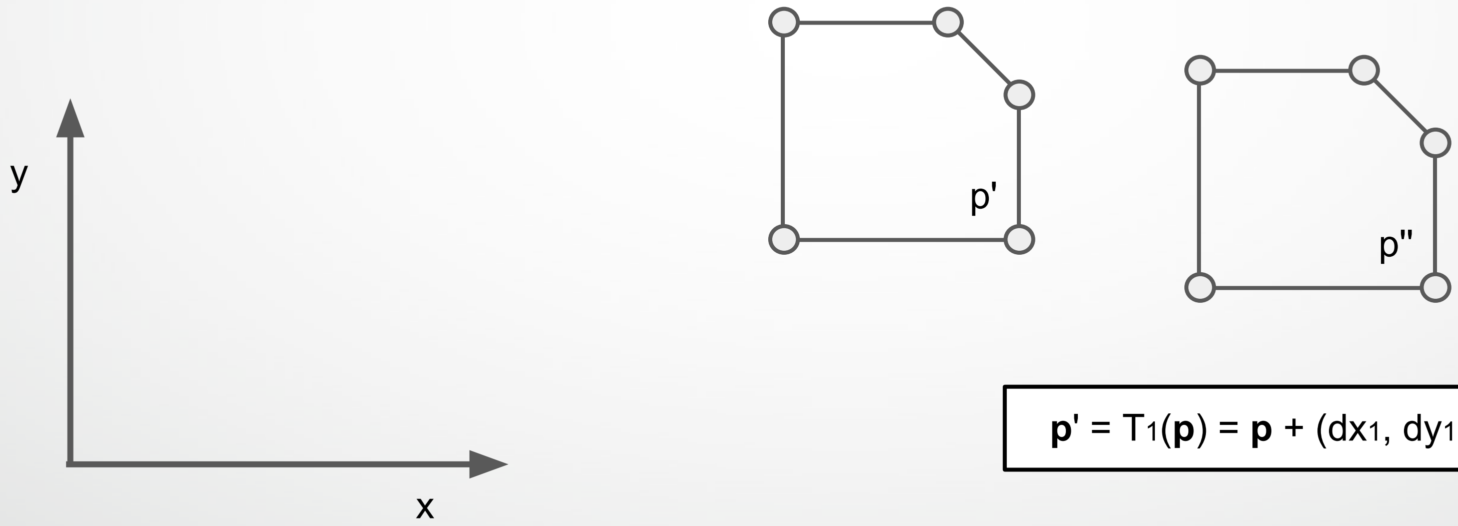


2D Translation



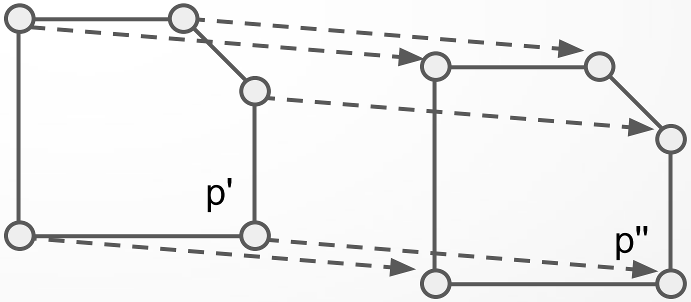
$\mathbf{p}' = T_1(\mathbf{p}) = \mathbf{p} + (dx_1, dy_1)$

2D Translation



$\mathbf{p}' = T_1(\mathbf{p}) = \mathbf{p} + (dx_1, dy_1)$

2D Translation



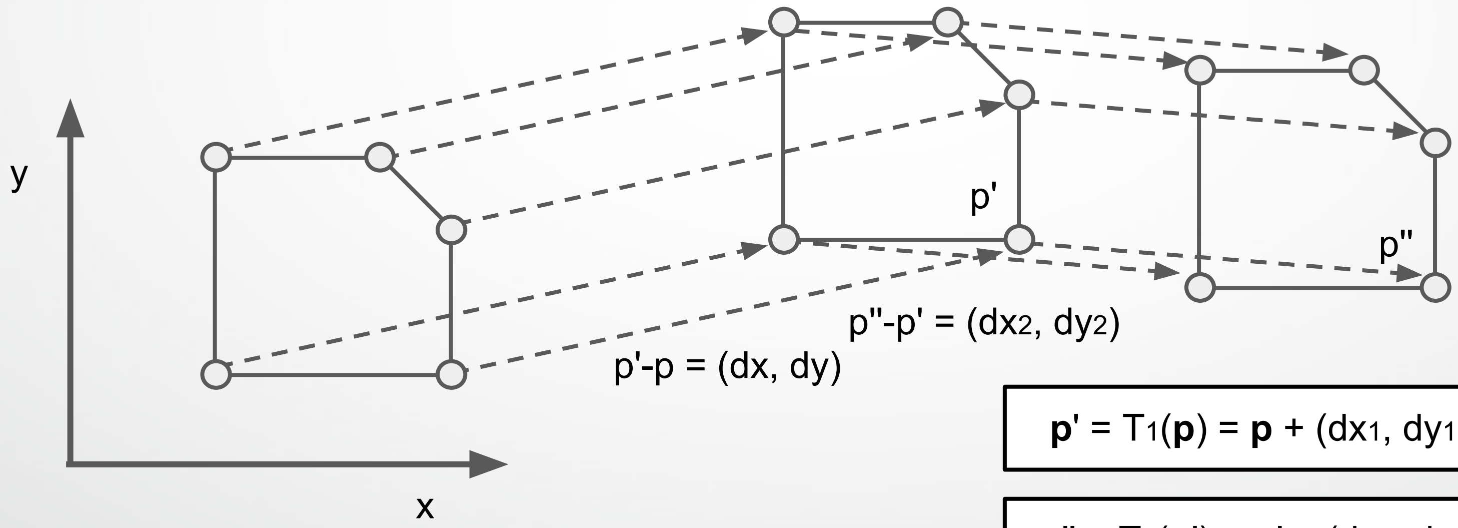
$p'' - p' = (dx_2, dy_2)$

$\mathbf{p}' = T_1(\mathbf{p}) = \mathbf{p} + (dx_1, dy_1)$

$\mathbf{p}'' = T_2(\mathbf{p}') = \mathbf{p}' + (dx_2, dy_2)$

2D Translation

$\mathbf{p}'' = T_2(T_1(\mathbf{p})) = \mathbf{p} + (dx_1, dy_1) + (dx_2, dy_2)$

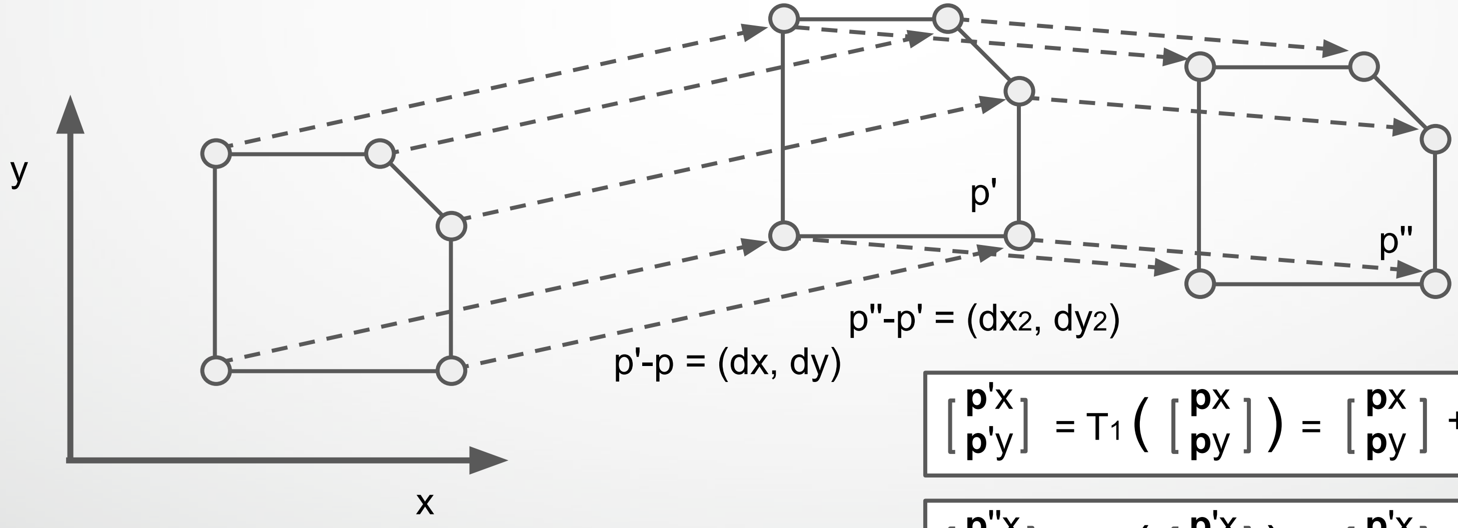


$\mathbf{p}' = T_1(\mathbf{p}) = \mathbf{p} + (dx_1, dy_1)$

$\mathbf{p}'' = T_2(\mathbf{p}') = \mathbf{p}' + (dx_2, dy_2)$

2D Translation

$$\begin{bmatrix} \mathbf{p}''_x \\ \mathbf{p}''_y \end{bmatrix} = T_2\left(T_1\left(\begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \end{bmatrix}\right)\right) = \begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \end{bmatrix} + \begin{bmatrix} d_{x1} \\ d_{y1} \end{bmatrix} + \begin{bmatrix} d_{x2} \\ d_{y2} \end{bmatrix}$$



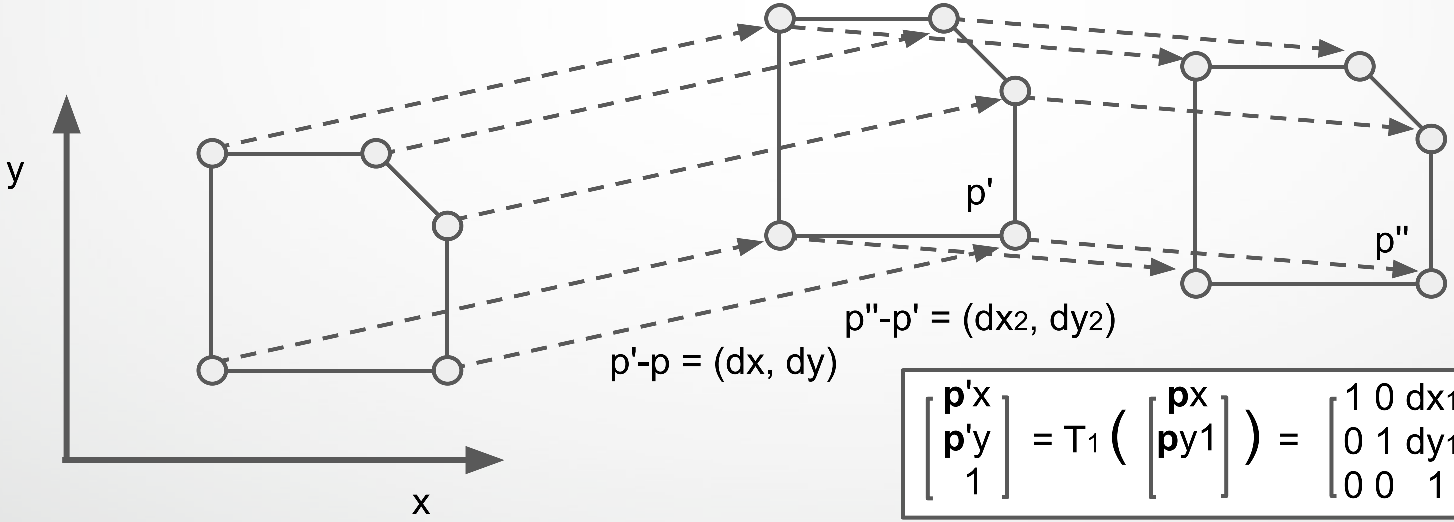
$$\begin{bmatrix} \mathbf{p}'_x \\ \mathbf{p}'_y \end{bmatrix} = T_1\left(\begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \end{bmatrix}\right) = \begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \end{bmatrix} + \begin{bmatrix} d_{x1} \\ d_{y1} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{p}''_x \\ \mathbf{p}''_y \end{bmatrix} = T_2\left(\begin{bmatrix} \mathbf{p}'_x \\ \mathbf{p}'_y \end{bmatrix}\right) = \begin{bmatrix} \mathbf{p}'_x \\ \mathbf{p}'_y \end{bmatrix} + \begin{bmatrix} d_{x2} \\ d_{y2} \end{bmatrix}$$

FeisStu

2D Translation

$$\begin{bmatrix} \mathbf{p}''x \\ \mathbf{p}''y \\ 1 \end{bmatrix} = T_2\left(T_1\left(\begin{bmatrix} \mathbf{p}x \\ \mathbf{p}y1 \\ \ \end{bmatrix}\right)\right) = \begin{bmatrix} 1 & 0 & dx_2 \\ 0 & 1 & dy_2 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & dx_1 \\ 0 & 1 & dy_1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \mathbf{p}x \\ \mathbf{p}y1 \\ \ \end{bmatrix}$$

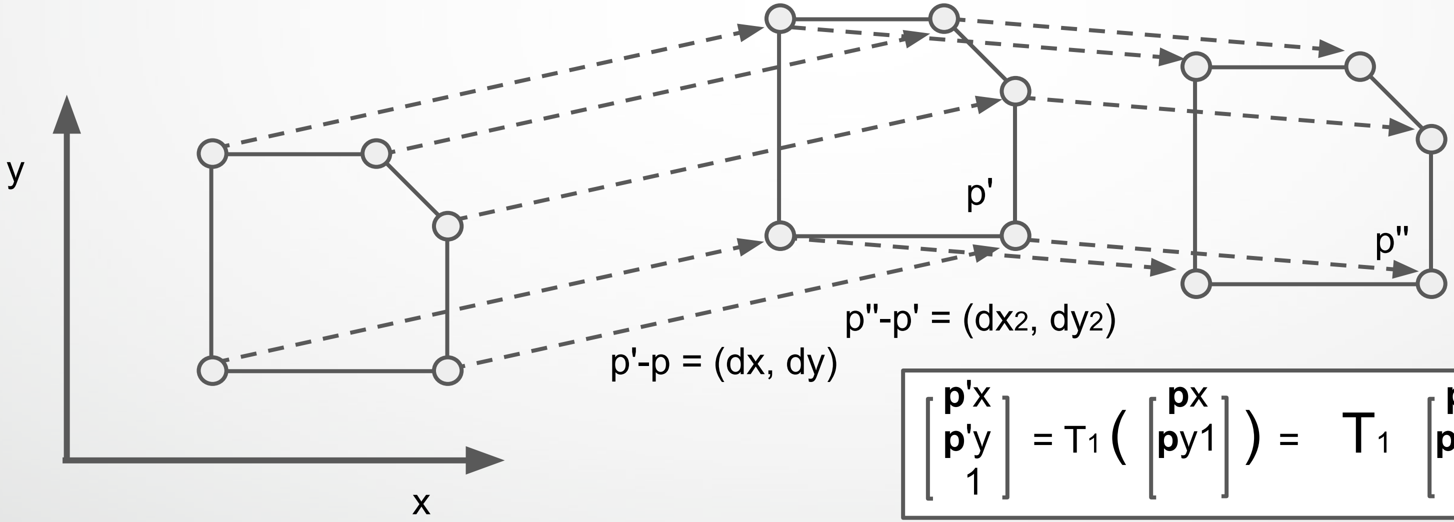


$$\begin{bmatrix} \mathbf{p}'x \\ \mathbf{p}'y \\ 1 \end{bmatrix} = T_1\left(\begin{bmatrix} \mathbf{p}x \\ \mathbf{p}y1 \\ \ \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 & dx_1 \\ 0 & 1 & dy_1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \mathbf{p}x \\ \mathbf{p}y1 \\ \ \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{p}''x \\ \mathbf{p}''y \\ 1 \end{bmatrix} = T_2\left(\begin{bmatrix} \mathbf{p}'x \\ \mathbf{p}'y \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 & dx_2 \\ 0 & 1 & dy_2 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \mathbf{p}'x \\ \mathbf{p}'y \\ 1 \end{bmatrix}$$

2D Translation

$$\begin{bmatrix} p''x \\ p''y \\ 1 \end{bmatrix} = T_2\left(T_1\left(\begin{bmatrix} px \\ py \\ 1 \end{bmatrix}\right)\right) = T_2 \; T_1 \begin{bmatrix} px \\ py \\ 1 \end{bmatrix}$$



$$\begin{bmatrix} p'x \\ p'y \\ 1 \end{bmatrix} = T_1\left(\begin{bmatrix} px \\ py \\ 1 \end{bmatrix}\right) = T_1 \begin{bmatrix} px \\ py \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} p''x \\ p''y \\ 1 \end{bmatrix} = T_2\left(\begin{bmatrix} p'x \\ p'y \\ 1 \end{bmatrix}\right) = T_2 \begin{bmatrix} p'x \\ p'y \\ 1 \end{bmatrix}$$

Vector2

Static Properties

down	Shorthand for writing Vector2(0, -1).
left	Shorthand for writing Vector2(-1, 0).
negativeInfinity	Shorthand for writing Vector2(float.NegativeInfinity, float.NegativeInfinity).
one	Shorthand for writing Vector2(1, 1).
positiveInfinity	Shorthand for writing Vector2(float.PositiveInfinity, float.PositiveInfinity).
right	Shorthand for writing Vector2(1, 0).
up	Shorthand for writing Vector2(0, 1).
zero	Shorthand for writing Vector2(0, 0).

Vector2

Properties

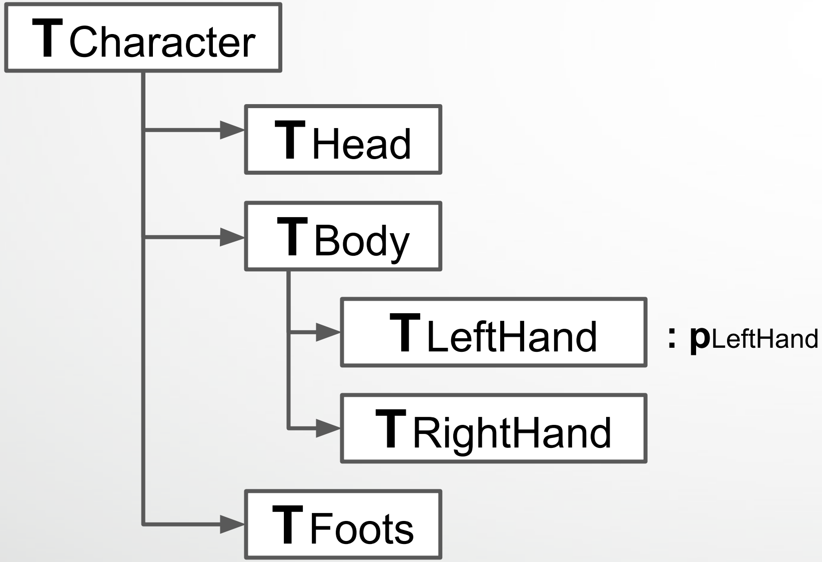
magnitude	Returns the length of this vector (Read Only).
normalized	Returns this vector with a magnitude of 1 (Read Only).
sqrMagnitude	Returns the squared length of this vector (Read Only).
this[int]	Access the x or y component using [0] or [1] respectively.
x	X component of the vector.
y	Y component of the vector.

FeisStu

Static Methods

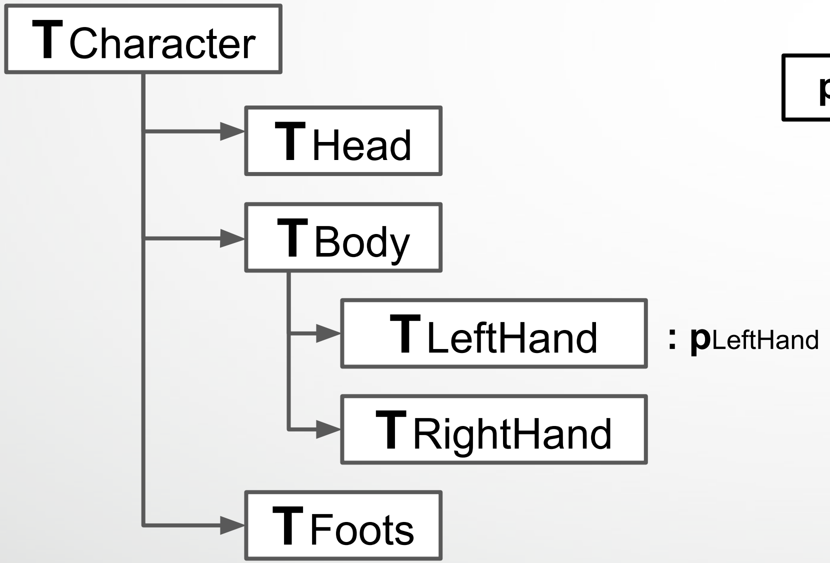
Angle	Returns the unsigned angle in degrees between from and to.
ClampMagnitude	Returns a copy of vector with its magnitude clamped to maxLength.
Distance	Returns the distance between a and b.
Dot	Dot Product of two vectors.
Lerp	Linearly interpolates between vectors a and b by t.
LerpUnclamped	Linearly interpolates between vectors a and b by t.
Max	Returns a vector that is made from the largest components of two vectors.
Min	Returns a vector that is made from the smallest components of two vectors.
MoveTowards	Moves a point current towards target.
Perpendicular	Returns the 2D vector perpendicular to this 2D vector. The result is always rotated 90-degrees in a counter-clockwise direction for a 2D coordinate system where the positive Y axis goes up.
Reflect	Reflects a vector off the vector defined by a normal.
Scale	Multiplies two vectors component-wise.
SignedAngle	Returns the signed angle in degrees between from and to.
SmoothDamp	Gradually changes a vector towards a desired goal over time.

Transformation Hierarchy



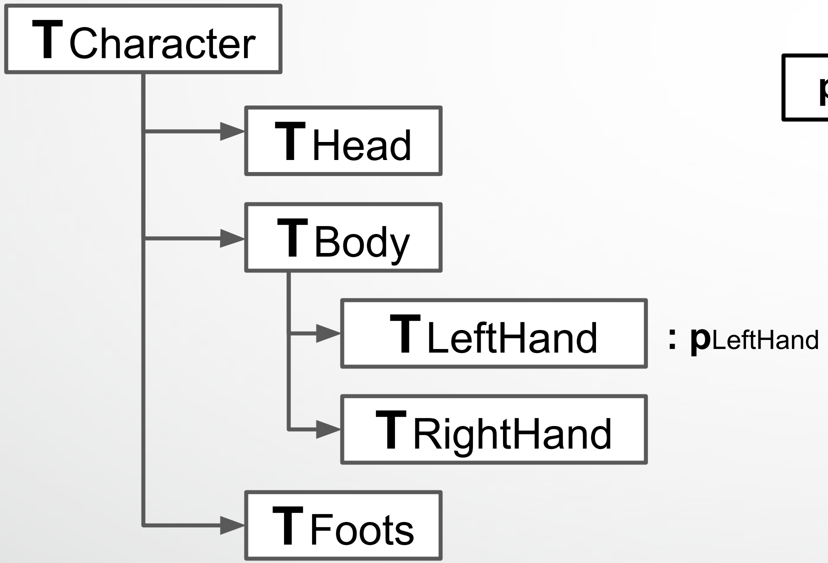
FeisStu

Transformation Hierarchy



$$\mathbf{p}'_{LeftHand} = \mathbf{T}_{Character}\ \mathbf{T}_{Body}\ \mathbf{T}_{LeftHand}\ \mathbf{p}_{LeftHand}$$

Transformation Hierarchy

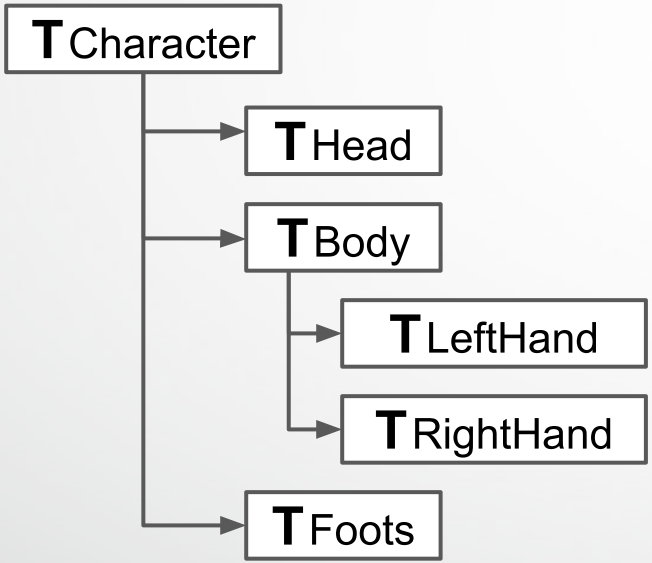


$$p'_{LeftHand} = T_{Character}\ T_{Body}\ T_{LeftHand}\ p_{LeftHand}$$

Precomputed ?

FeisStu

Transformation Hierarchy



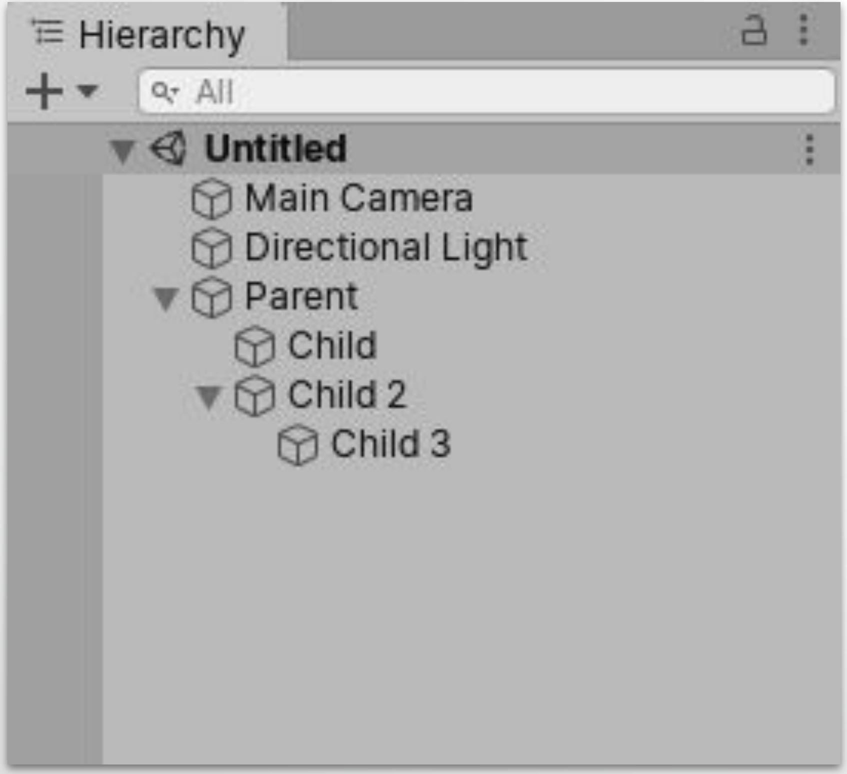
: $p_{LeftHand}$

$$p'_{LeftHand} = T_{Character}\ T_{Body}\ T_{LeftHand}\ p_{LeftHand}$$

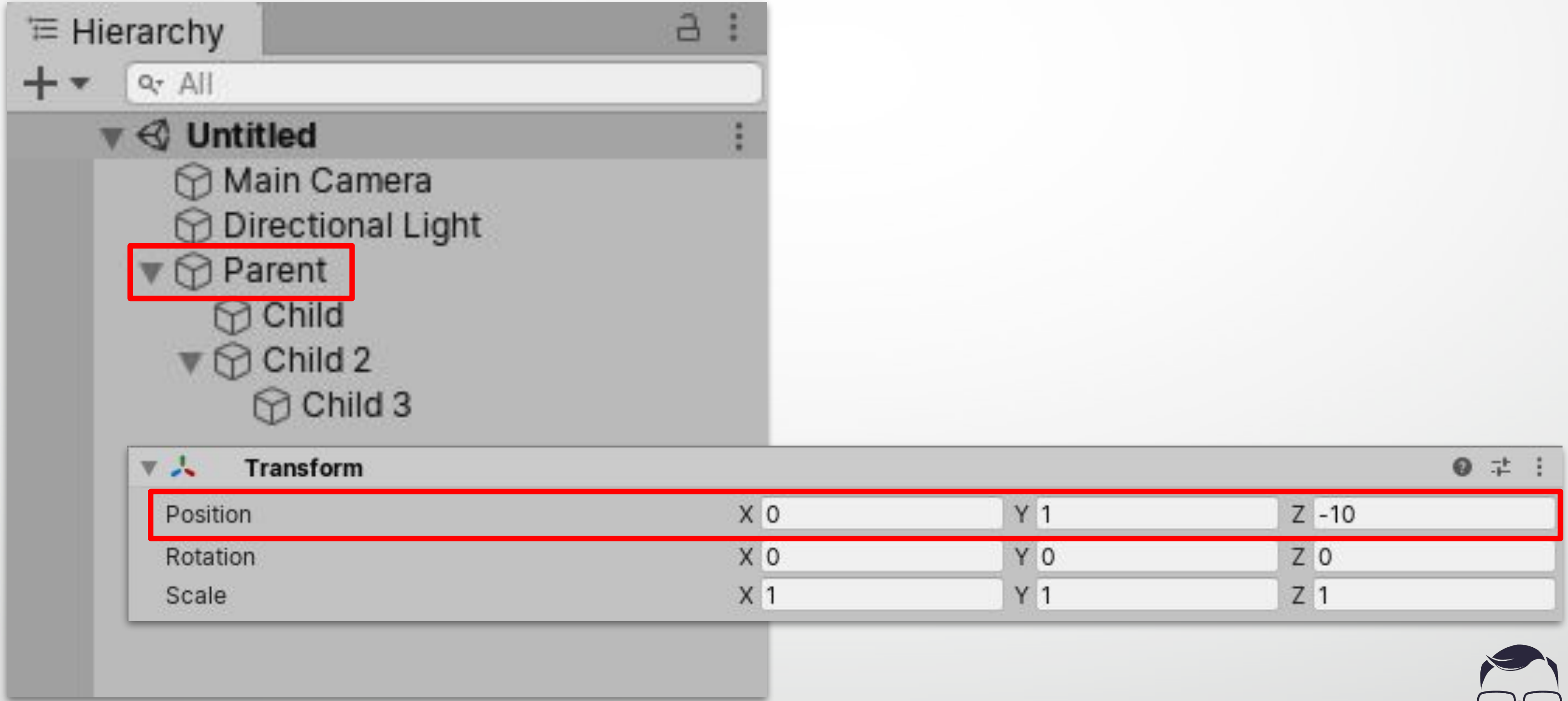
Precomputed ?

Object coordinates to world coordinates

Hierarchy window

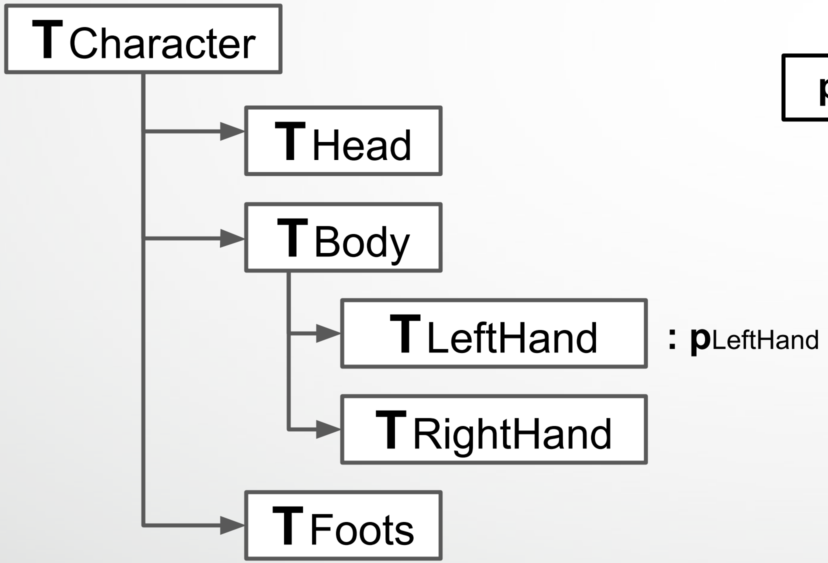


Hierarchy window



FeisStu

Transformation Hierarchy

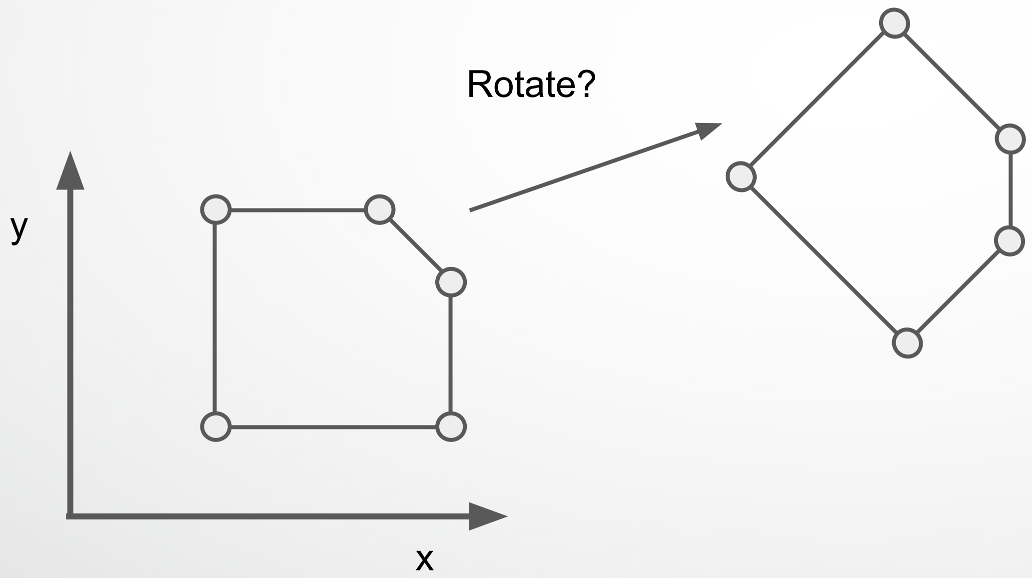


$$p'_{LeftHand} = T_{Character}\ T_{Body}\ T_{LeftHand}\ p_{LeftHand}$$

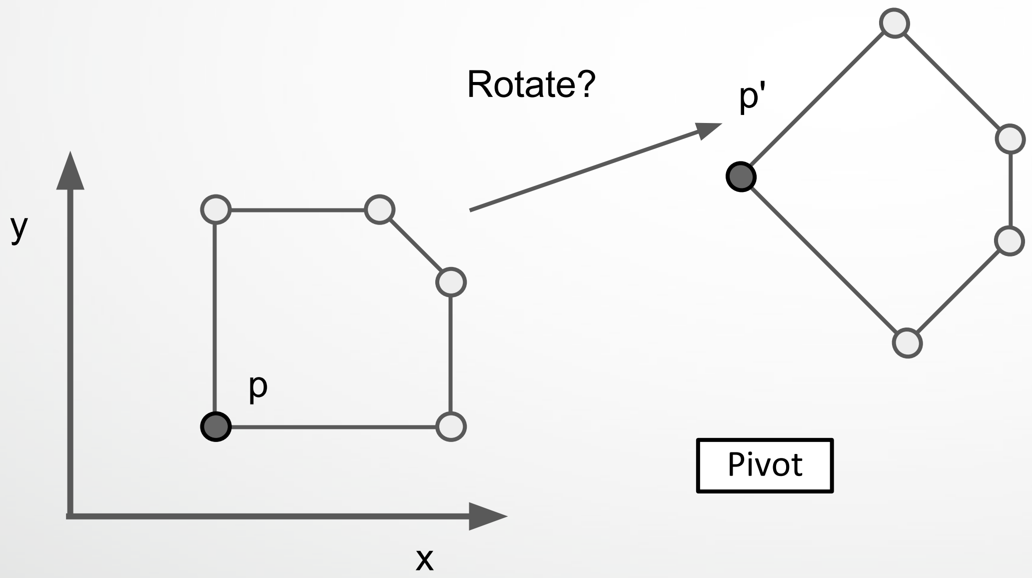
Order matters ?

Commutative ?

2D Rotation

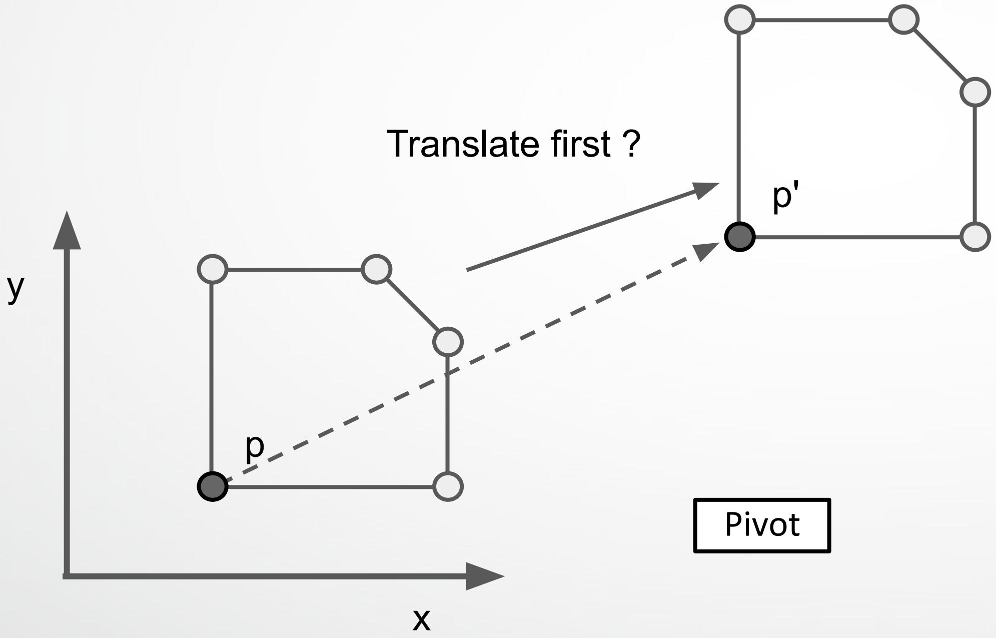


2D Rotation



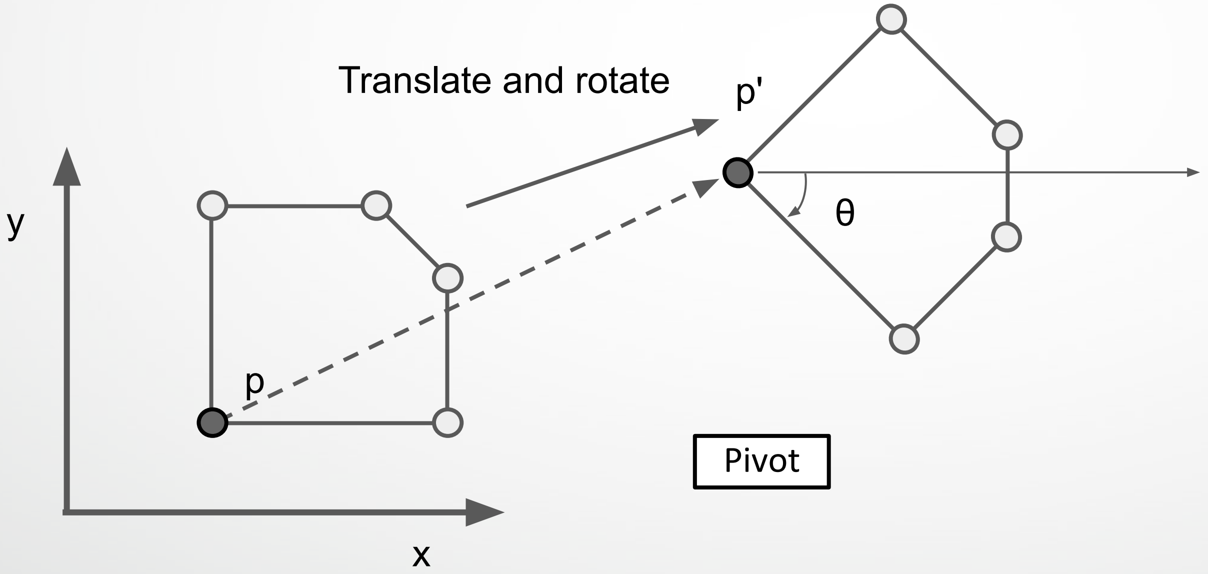
2D Rotation

$$\begin{bmatrix} p'_x \\ p'_y \\ 1 \end{bmatrix} = \begin{bmatrix} & ? & \end{bmatrix} T_1 \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}$$



2D Rotation

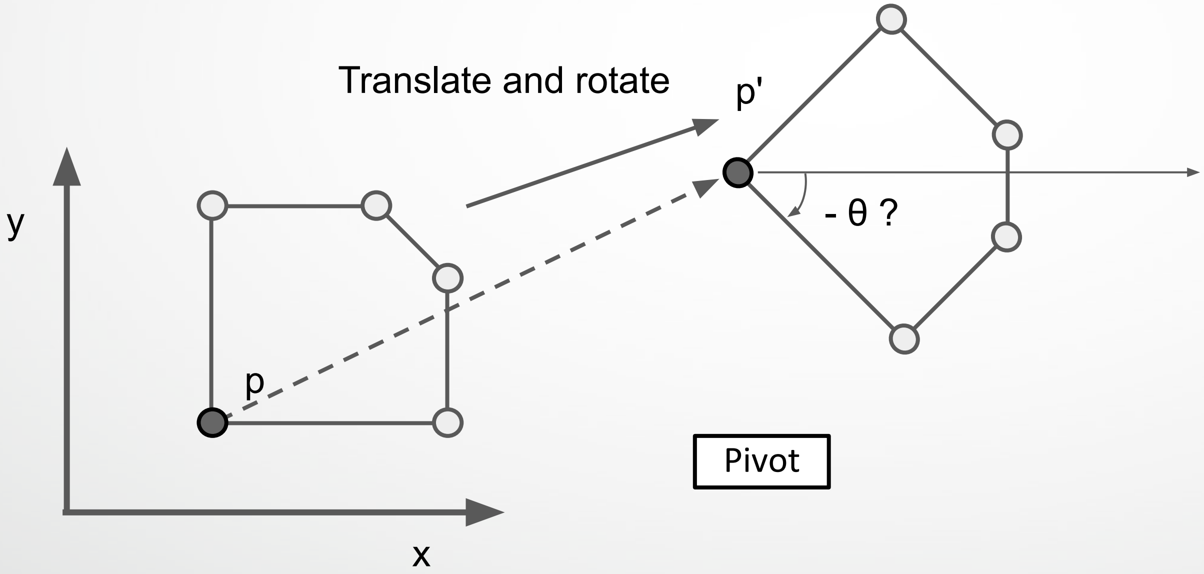
$$\begin{bmatrix} p'x \\ p'y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} T_1 \begin{bmatrix} px \\ py \\ 1 \end{bmatrix}$$



FeisStu

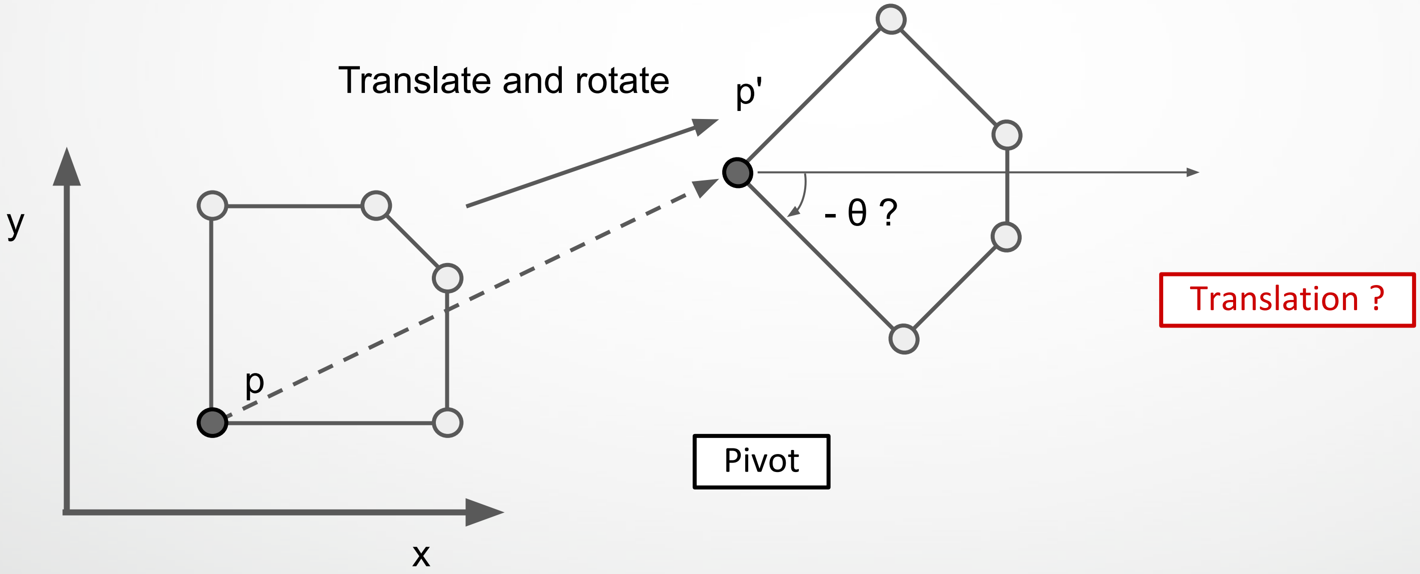
2D Rotation

$$\begin{bmatrix} \mathbf{p'}x \\ \mathbf{p'}y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\text{-}\theta & -\sin\text{-}\theta & 0 \\ \sin\text{-}\theta & \cos\text{-}\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} T_1 \begin{bmatrix} \mathbf{p}x \\ \mathbf{p}y1 \end{bmatrix}$$



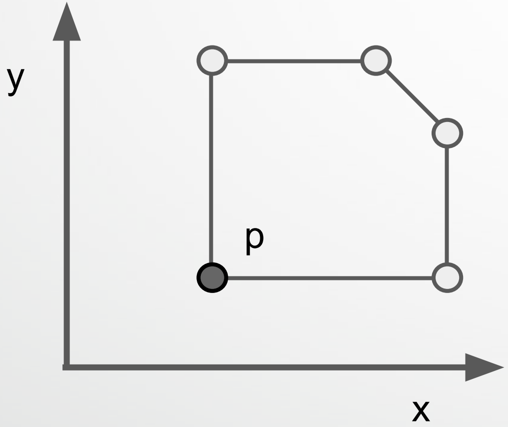
2D Rotation

$$\begin{bmatrix} p'x \\ p'y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\text{-}\theta & -\sin\text{-}\theta & 0 \\ \sin\text{-}\theta & \cos\text{-}\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} T_1 \begin{bmatrix} px \\ py \\ 1 \end{bmatrix}$$



2D Rotation

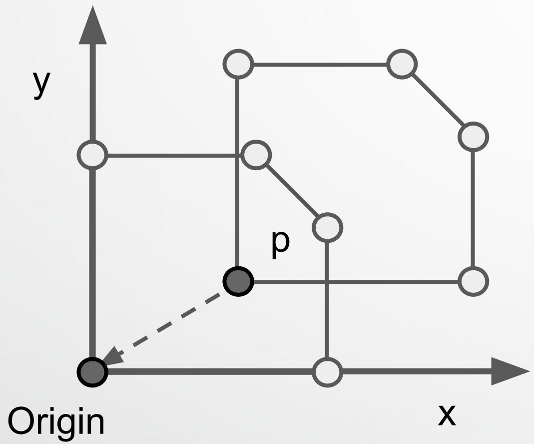
$$\begin{bmatrix} p'_x \\ p'_y \\ 1 \end{bmatrix} = \qquad \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}$$



2D Rotation

$$\begin{bmatrix} p'x \\ p'y \\ 1 \end{bmatrix} = T^{-1}_p \begin{bmatrix} px \\ py \\ 1 \end{bmatrix}$$

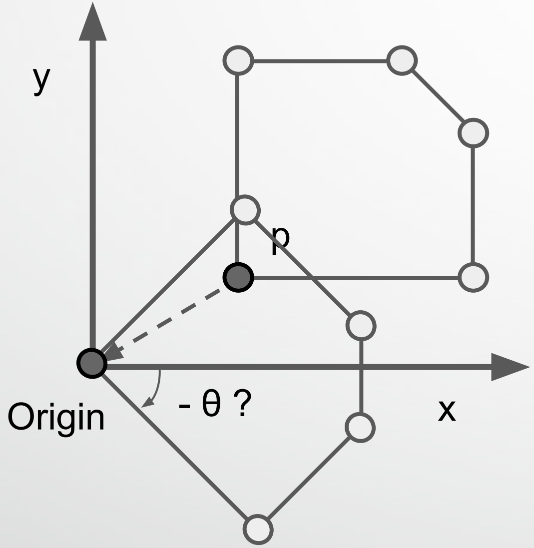
Translate



2D Rotation

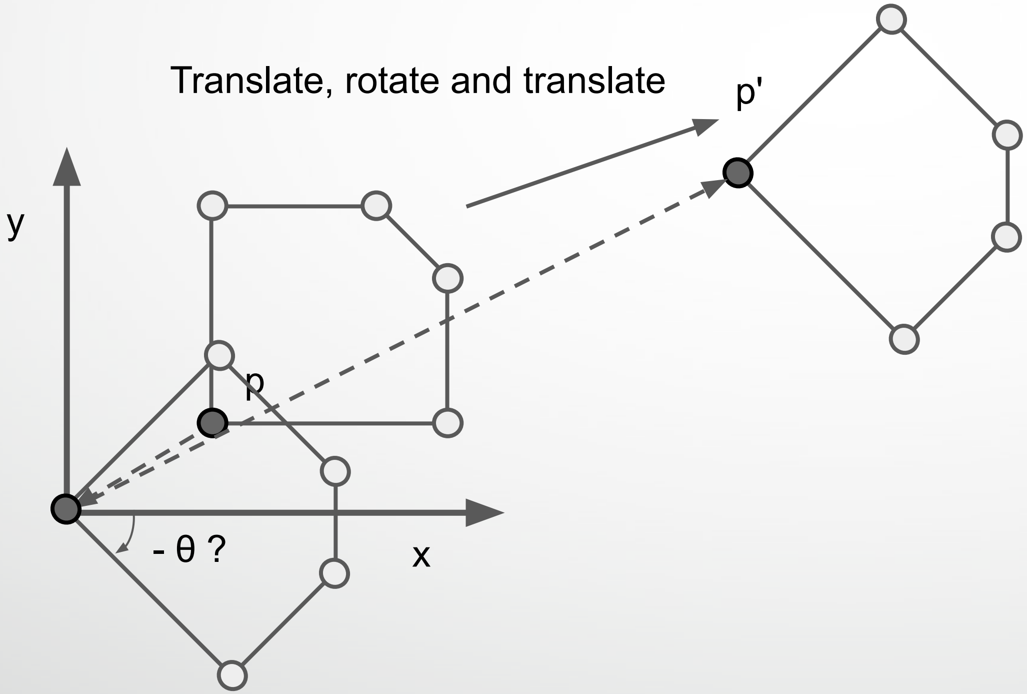
$$\begin{bmatrix} \mathbf{p}'x \\ \mathbf{p}'y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos-\theta & -\sin-\theta & 0 \\ \sin-\theta & \cos-\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} T^{-1}{}_{p} \begin{bmatrix} \mathbf{p}x \\ \mathbf{p}y \\ 1 \end{bmatrix}$$

Translate, rotate

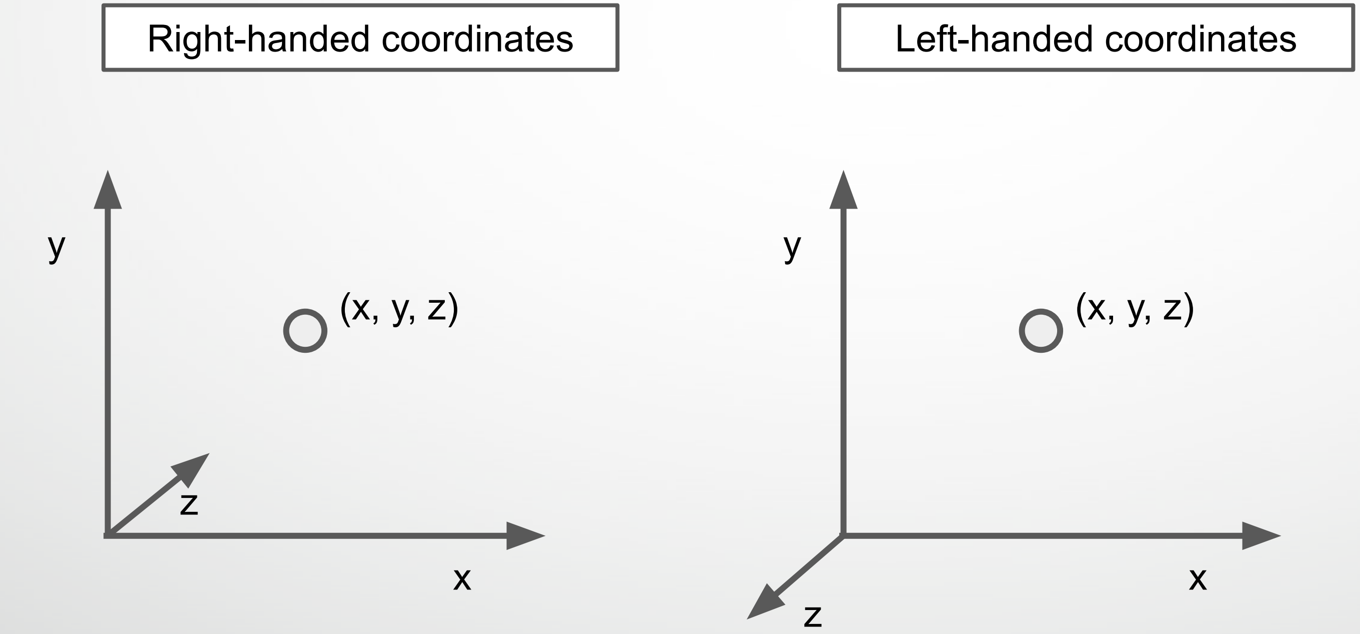


2D Rotation

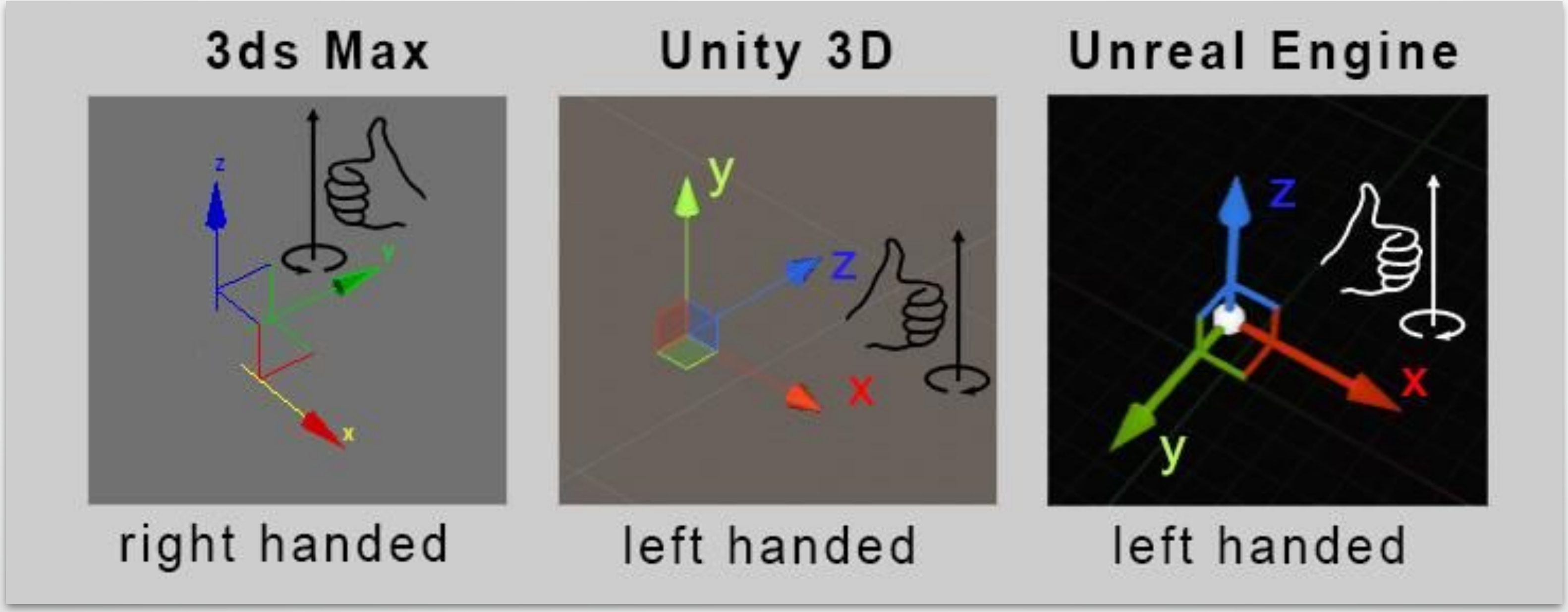
$$\begin{bmatrix} \mathbf{p}'x \\ \mathbf{p}'y \\ 1 \end{bmatrix} = T_{p'} \begin{bmatrix} \cos\text{-}\theta & -\sin\text{-}\theta & 0 \\ \sin\text{-}\theta & \cos\text{-}\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} T^{-1}_{p} \begin{bmatrix} \mathbf{p}x \\ \mathbf{p}y \\ 1 \end{bmatrix}$$



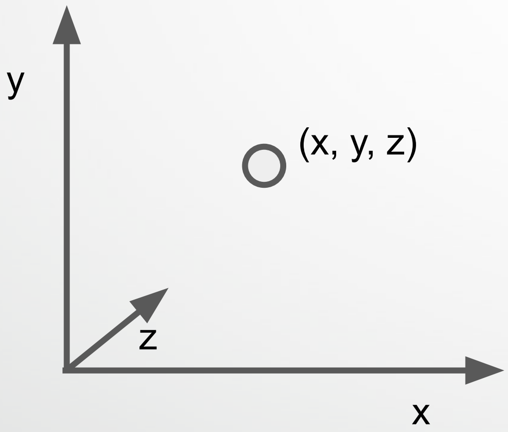
3D Coordinates Systems



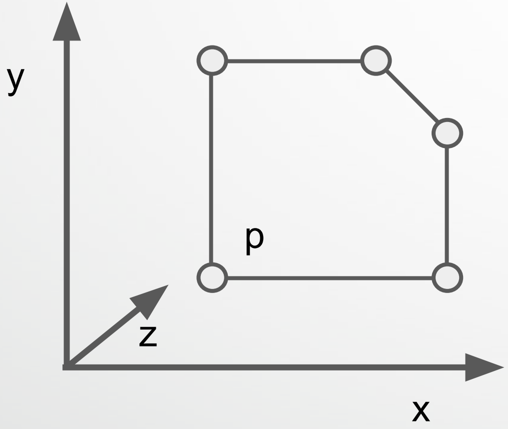
Left-handed coordinates



3D Point

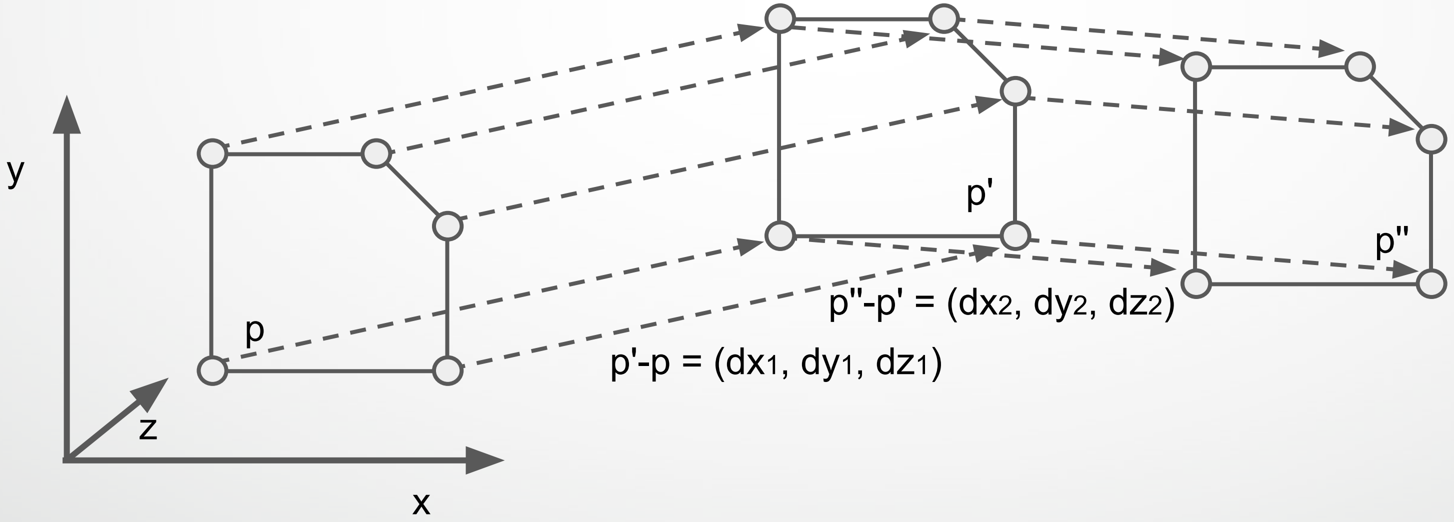


3D Object (Mesh)

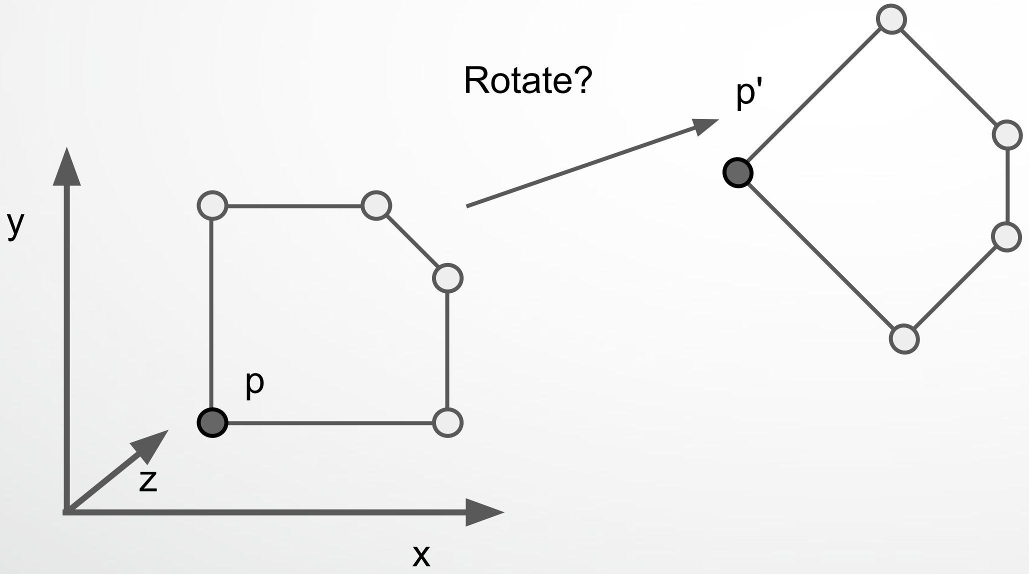


3D Translation

$$\begin{bmatrix} p''x \\ p''y \\ p''y \\ 1 \end{bmatrix} = T_2\left(T_1\left(\begin{bmatrix} px \\ py \\ pz1 \end{bmatrix}\right)\right) = T_2\, T_1 \begin{bmatrix} px \\ py \\ pz1 \end{bmatrix}$$

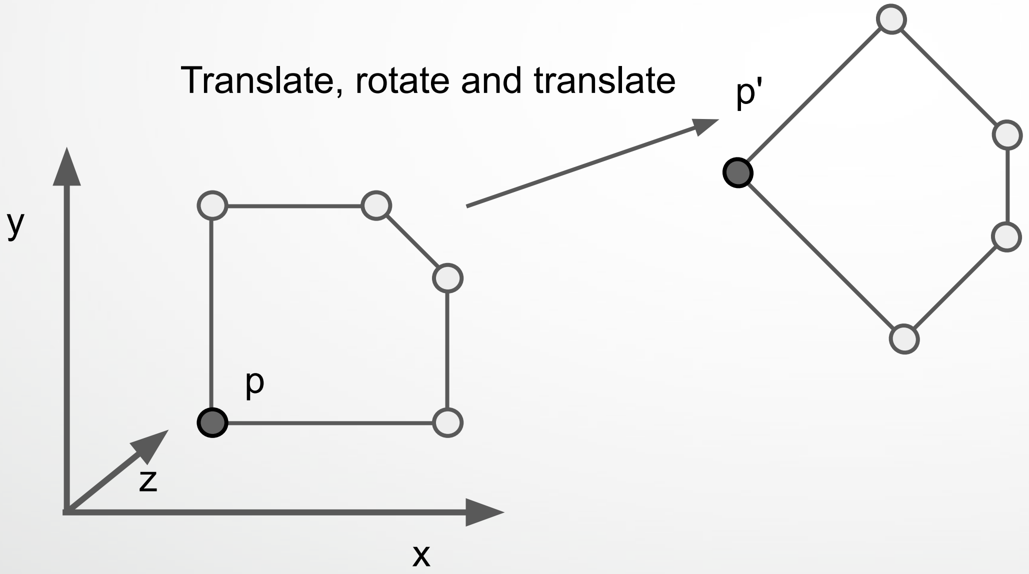


3D Rotation ?

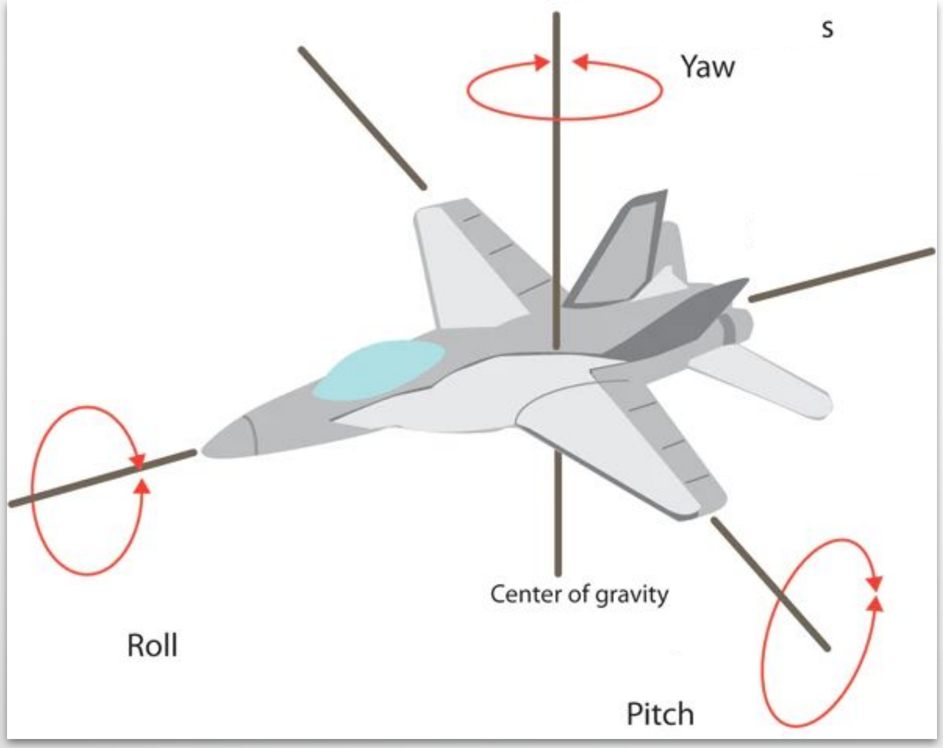


3D Rotation ?

$$\begin{bmatrix} \mathbf{p}''x \\ \mathbf{p}''y \\ \mathbf{p}''y \\ 1 \end{bmatrix} = T_{p'} \begin{bmatrix} \cos\text{-}\theta & \text{-}\sin\text{-}\theta & 0 & 0 \\ \sin\text{-}\theta & \cos\text{-}\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} T^{-1}{}_{p} \begin{bmatrix} \mathbf{p}x \\ \mathbf{p}y \\ \mathbf{p}z1 \end{bmatrix}$$

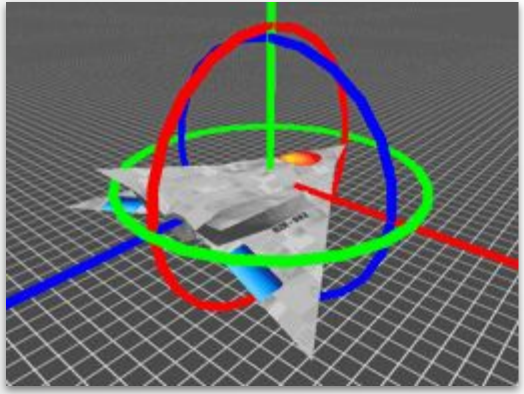


Euler angles



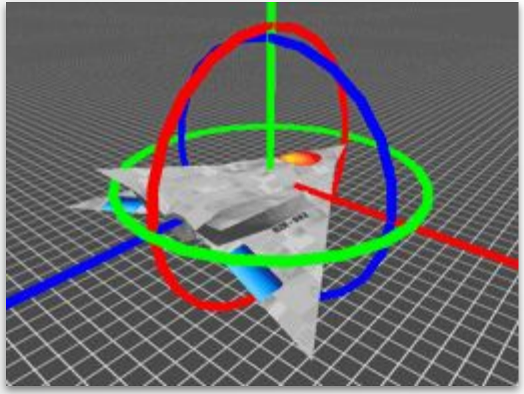
From: Wikipedia

Euler angles

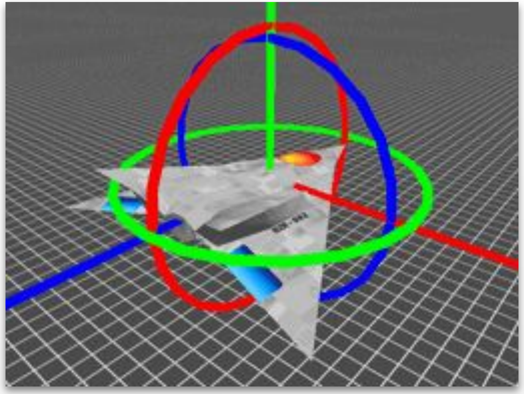


Yaw

Euler angles

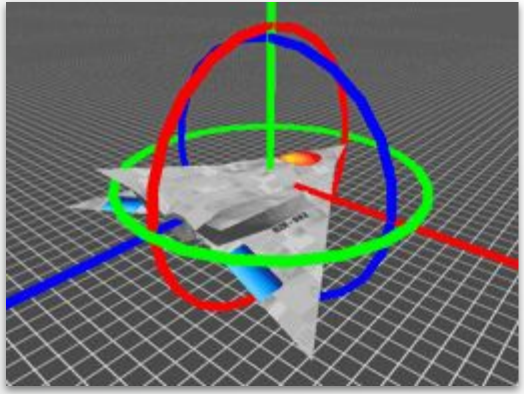


Yaw

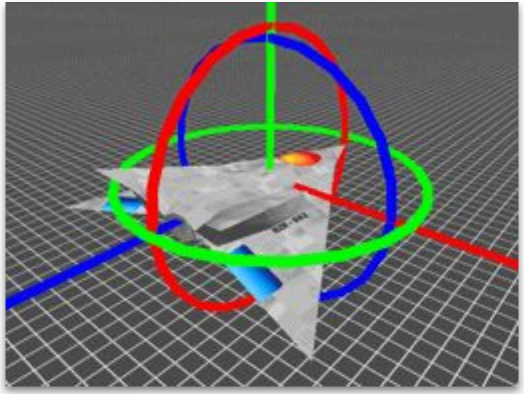


Pitch

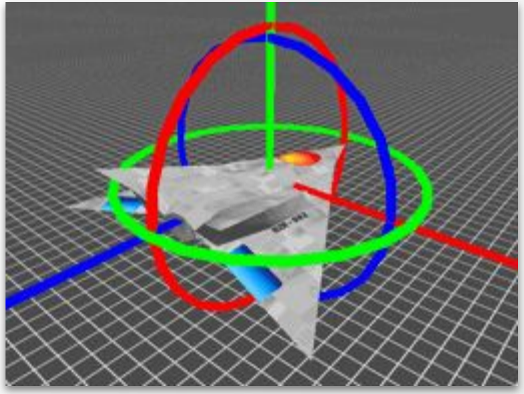
Euler angles



Yaw

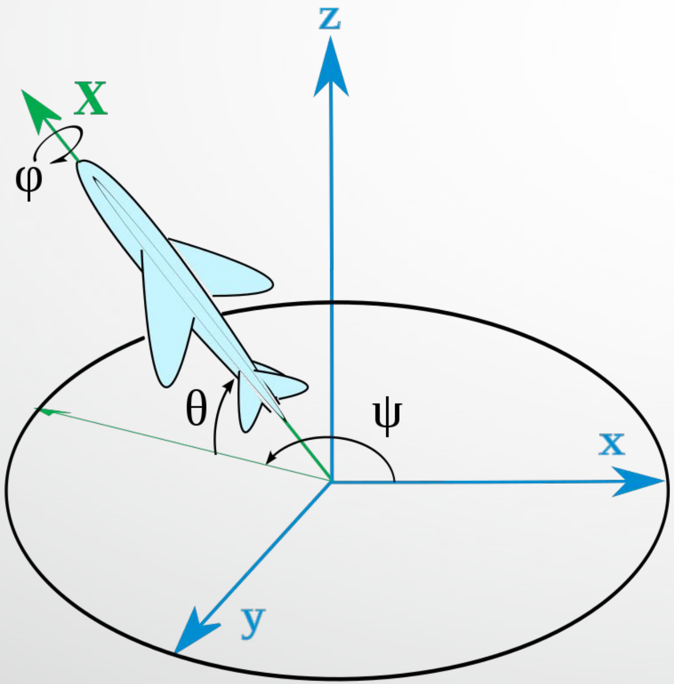


Pitch



Roll

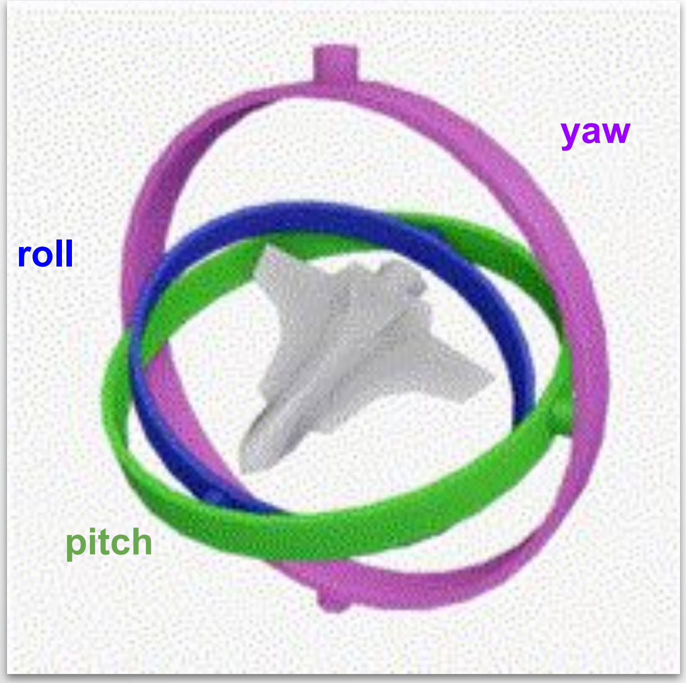
Euler angles



From: Wikipedia

$$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = R_{Yaw}(\psi)\, R_{Pitch}(\theta)\, R_{Roll}(\varphi) \begin{bmatrix} p_x \\ p_y \\ p_z 1 \end{bmatrix}$$

Euler angles

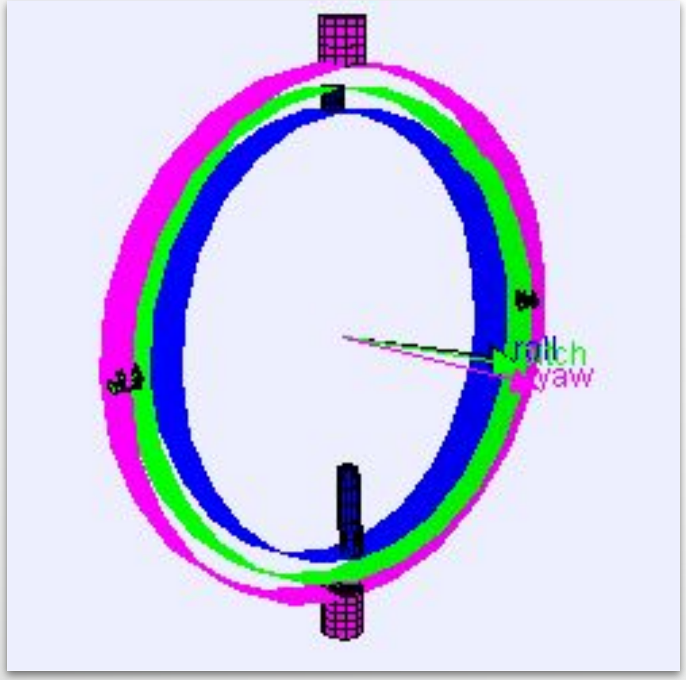


From: Wikipedia

$$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = R_{Yaw}(\psi)\, R_{Pitch}(\theta)\, R_{Roll}(\varphi) \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

FeisStu

Euler angles

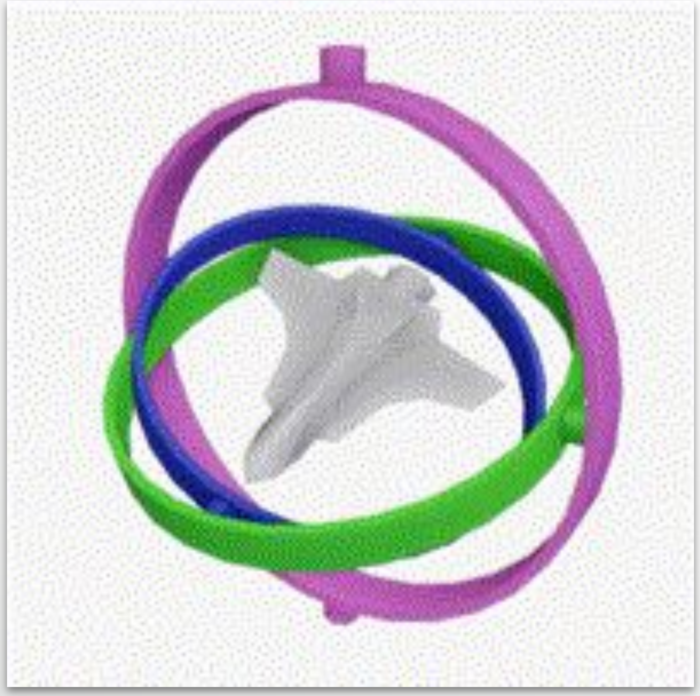


From: Wikipedia

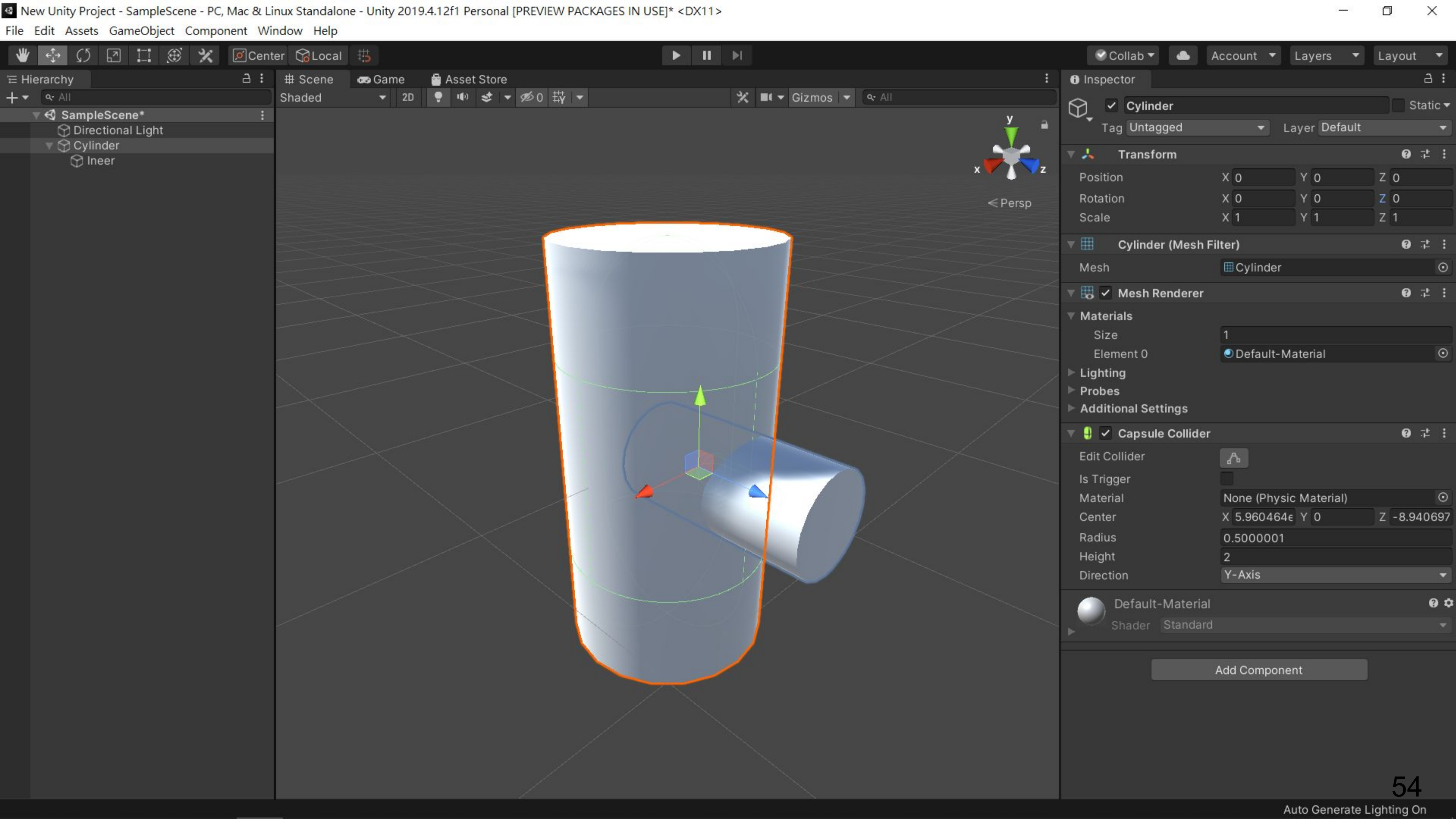
$$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = R_{Yaw}(\psi)\, R_{Pitch}(\theta)\, R_{Roll}(\varphi) \begin{bmatrix} p_x \\ p_y \\ p_z 1 \end{bmatrix}$$

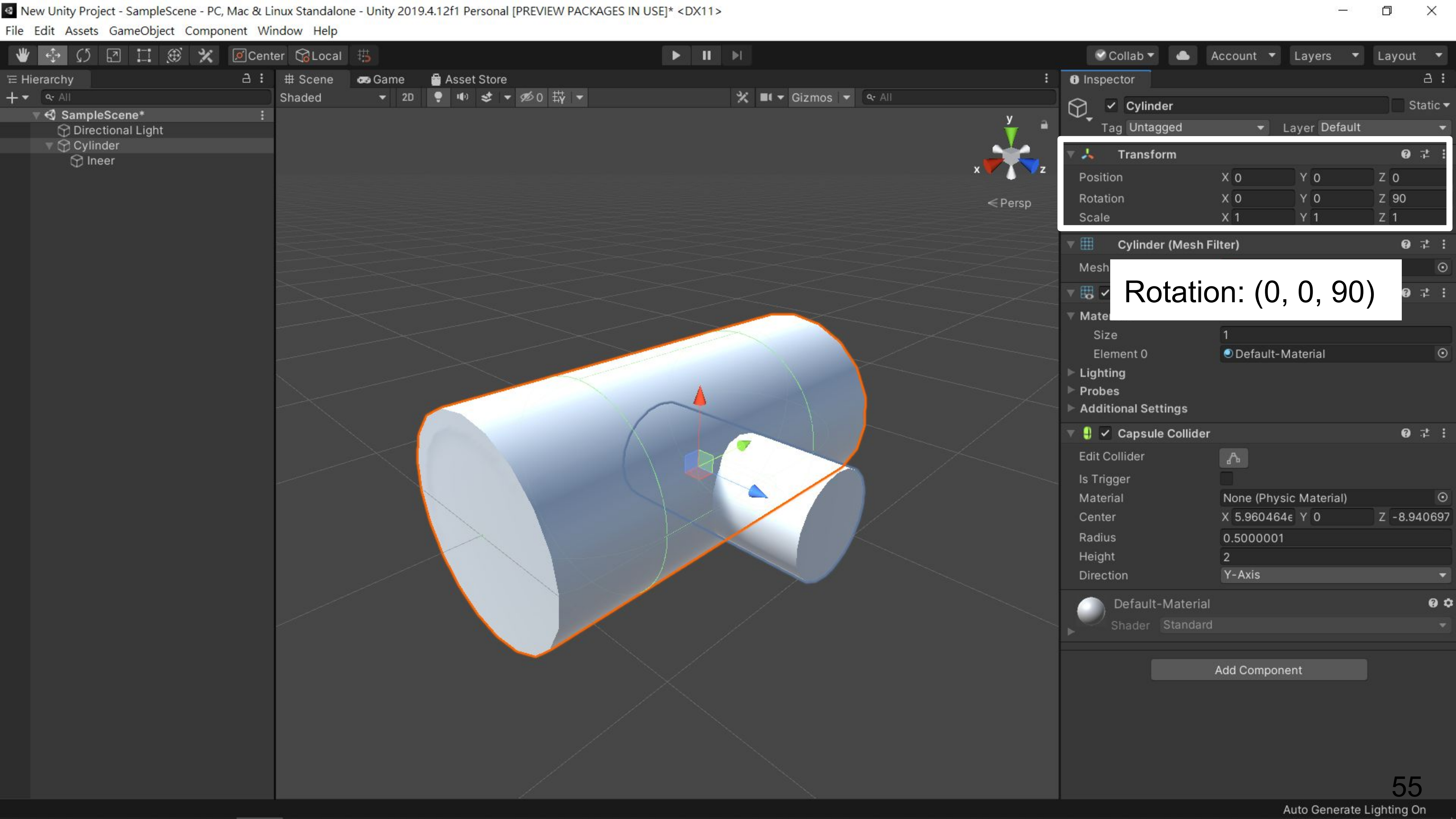
FeisStu

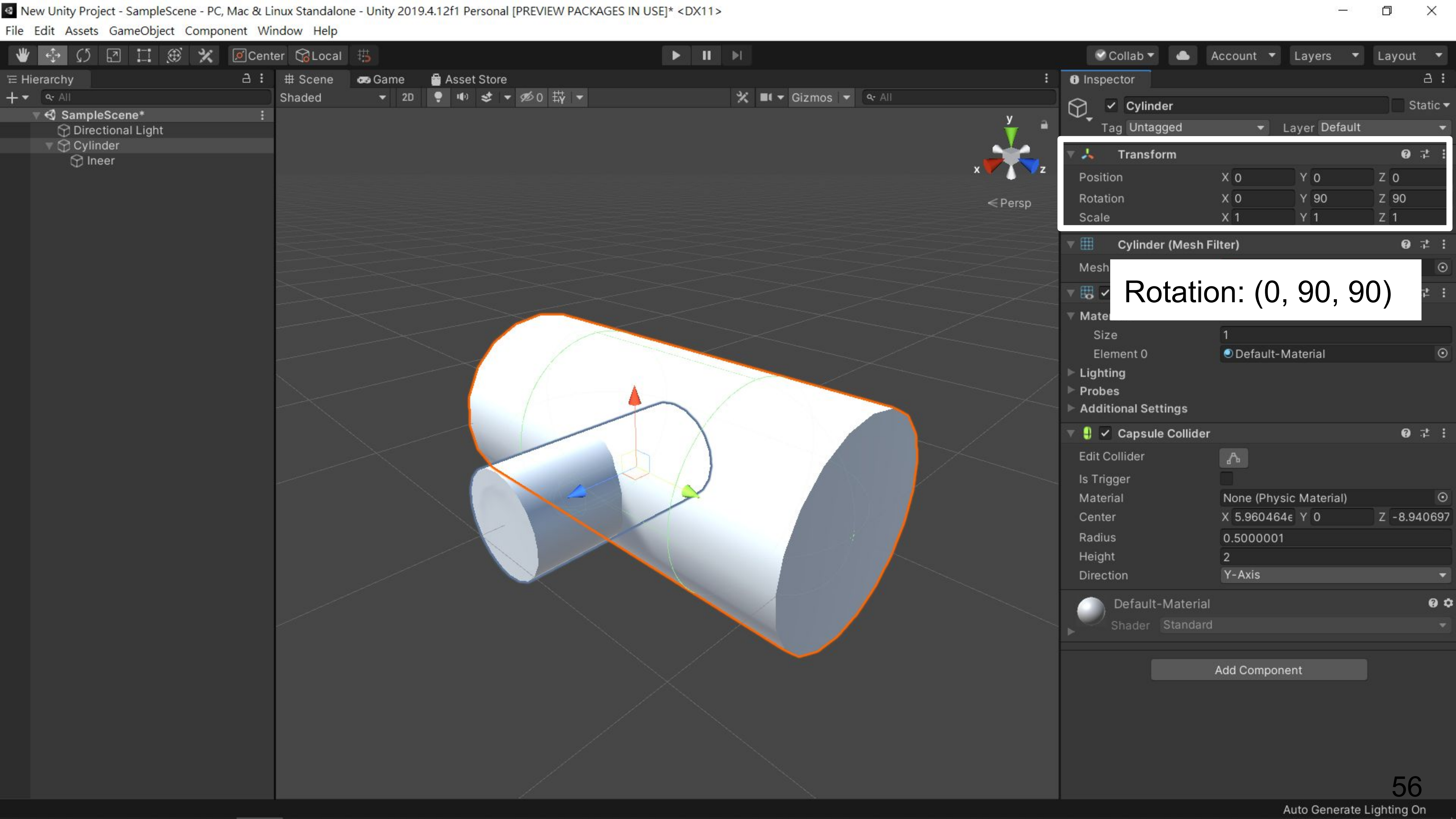
Gimbal lock

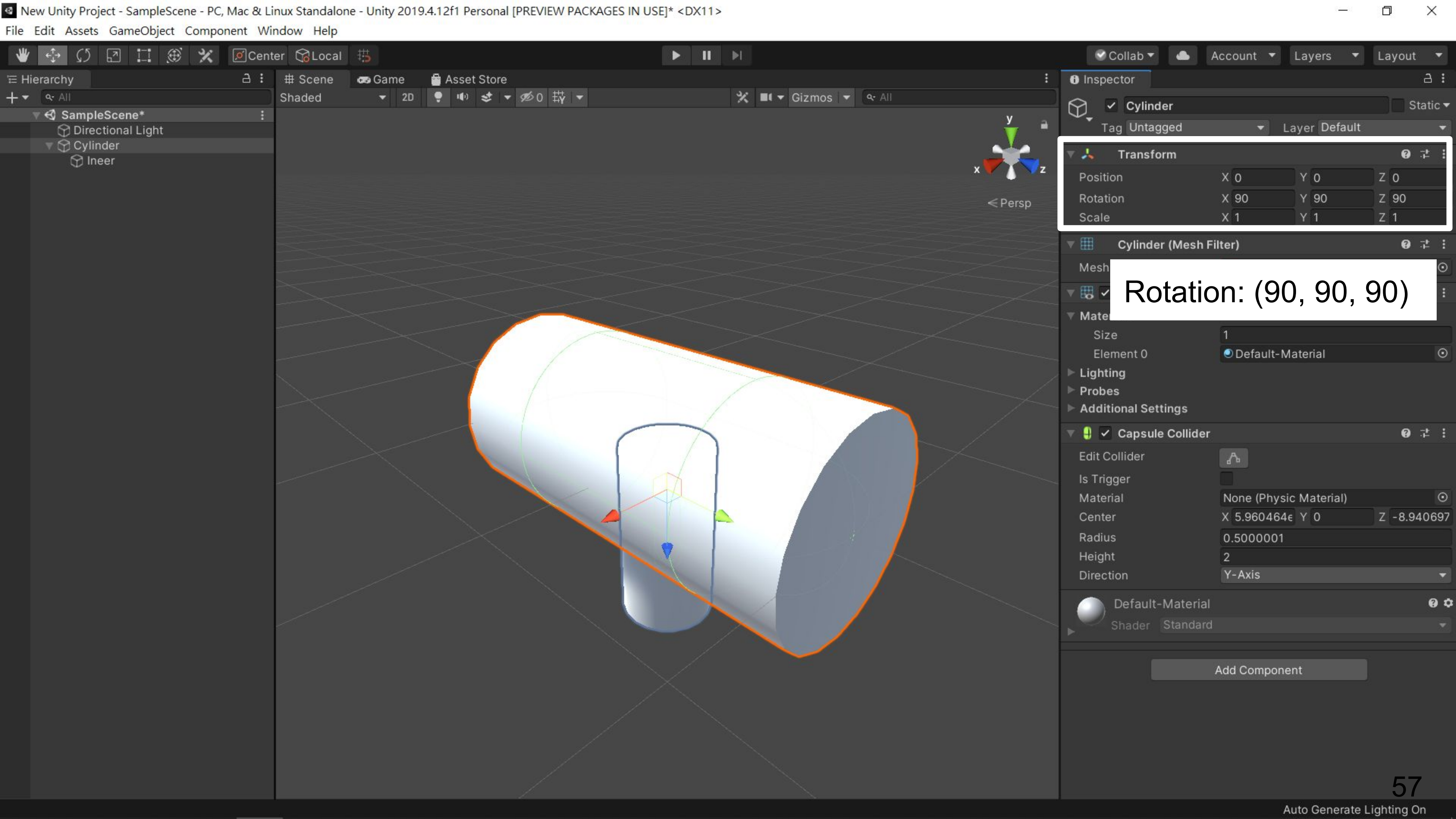


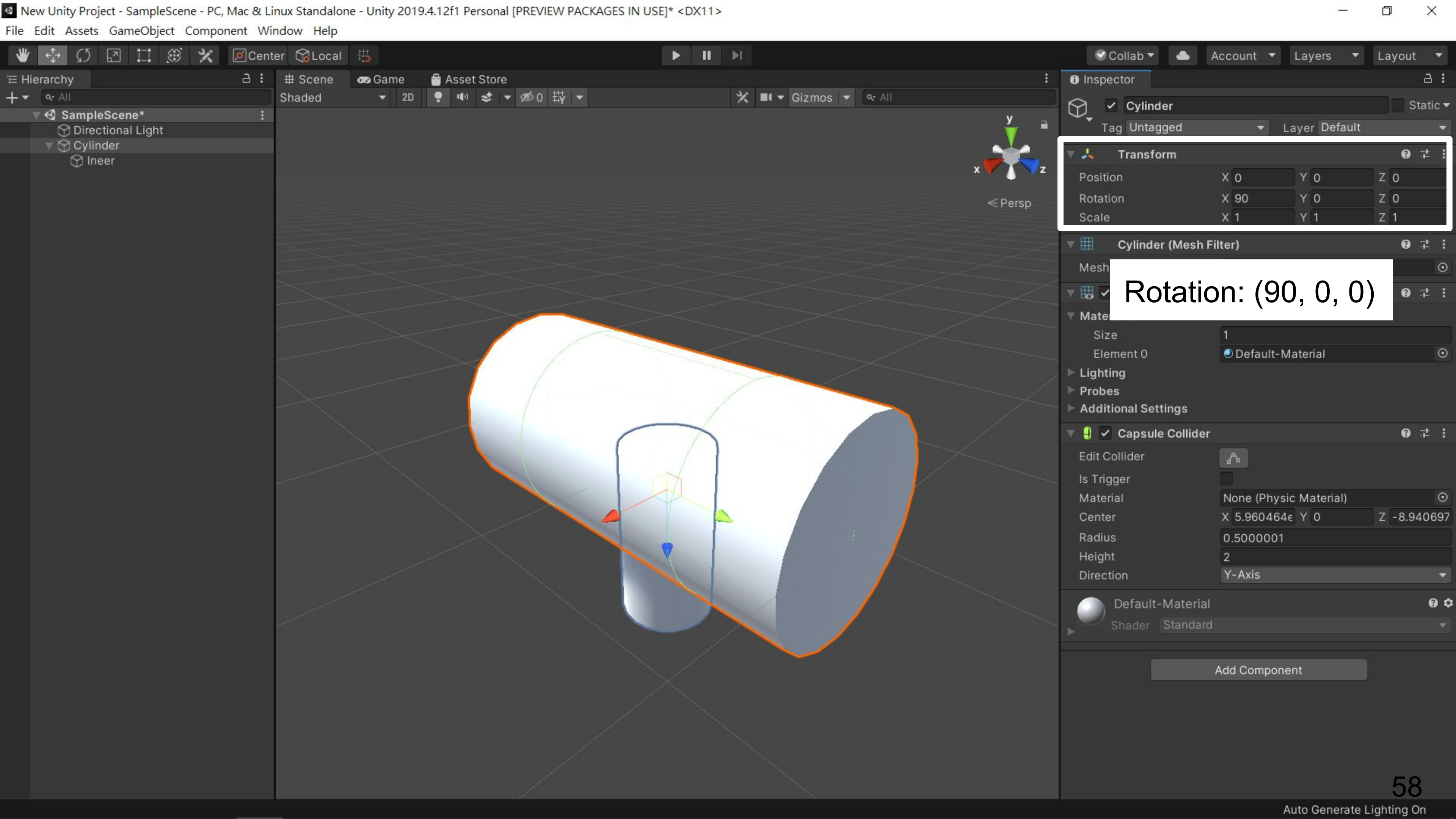
From: Wikipedia


New Unity Project - SampleScene - PC, Mac & Linux Standalone - Unity 2019.4.12f1 Personal [PREVIEW PACKAGES IN USE]* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
SampleScene*
Directional Light
Cylinder
Ineer
Scene Game Asset Store
Shaded 2D Gizmos
Persp
Inspector
Cylinder
Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Cylinder (Mesh Filter)
Mesh Cylinder
Mesh Renderer
Materials
Size 1
Element 0 Default-Material
Lighting
Probes
Additional Settings
Capsule Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 5.960464e Y 0 Z -8.940697
Radius 0.5000001
Height 2
Direction Y-Axis
Default-Material
Shader Standard
Add Component
Auto Generate Lighting On


New Unity Project - SampleScene - PC, Mac & Linux Standalone - Unity 2019.4.12f1 Personal [PREVIEW PACKAGES IN USE]* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
SampleScene*
Directional Light
Cylinder
Ineer
Scene Game Asset Store
Shaded 2D Gizmos
Persp
Inspector
Cylinder
Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 90
Scale X 1 Y 1 Z 1
Cylinder (Mesh Filter)
Rotation: (0, 0, 90)
Size 1
Element 0 Default-Material
Lighting
Probes
Additional Settings
Capsule Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 5.960464e Y 0 Z -8.940697
Radius 0.5000001
Height 2
Direction Y-Axis
Default-Material
Shader Standard
Add Component
Auto Generate Lighting On


New Unity Project - SampleScene - PC, Mac & Linux Standalone - Unity 2019.4.12f1 Personal [PREVIEW PACKAGES IN USE]* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
SampleScene*
Directional Light
Cylinder
Ineer
Scene Game Asset Store
Shaded 2D Gizmos
Persp
Inspector
Cylinder
Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 90 Z 90
Scale X 1 Y 1 Z 1
Cylinder (Mesh Filter)
Mesh
Rotation: (0, 90, 90)
Mate
Size 1
Element 0 Default-Material
Lighting
Probes
Additional Settings
Capsule Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 5.960464e Y 0 Z -8.940697
Radius 0.5000001
Height 2
Direction Y-Axis
Default-Material
Shader Standard
Add Component
Auto Generate Lighting On


New Unity Project - SampleScene - PC, Mac & Linux Standalone - Unity 2019.4.12f1 Personal [PREVIEW PACKAGES IN USE]* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
SampleScene*
Directional Light
Cylinder
Ineer
Scene Game Asset Store
Shaded 2D Gizmos
Persp
Inspector
Cylinder
Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 90 Y 90 Z 90
Scale X 1 Y 1 Z 1
Cylinder (Mesh Filter)
Rotation: (90, 90, 90)
Size 1
Element 0 Default-Material
Lighting
Probes
Additional Settings
Capsule Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 5.960464e Y 0 Z -8.940697
Radius 0.5000001
Height 2
Direction Y-Axis
Default-Material
Shader Standard
Add Component
Auto Generate Lighting On


New Unity Project - SampleScene - PC, Mac & Linux Standalone - Unity 2019.4.12f1 Personal [PREVIEW PACKAGES IN USE]* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
SampleScene*
Directional Light
Cylinder
Ineer
Scene Game Asset Store
Shaded 2D Gizmos
Persp
Inspector
Cylinder
Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 90 Y 0 Z 0
Scale X 1 Y 1 Z 1
Cylinder (Mesh Filter)
Mesh
Rotation: (90, 0, 0)
Size 1
Element 0 Default-Material
Lighting
Probes
Additional Settings
Capsule Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 5.960464e Y 0 Z -8.940697
Radius 0.5000001
Height 2
Direction Y-Axis
Default-Material
Shader Standard
Add Component
Auto Generate Lighting On

Euler angles

- Cons :
  - Order of rotation sequence (XYZ or ZYX) matters

Euler angles

- Cons :
  - Order of rotation sequence (XYZ or ZYX) matters
  - Gimbal lock

Euler angles

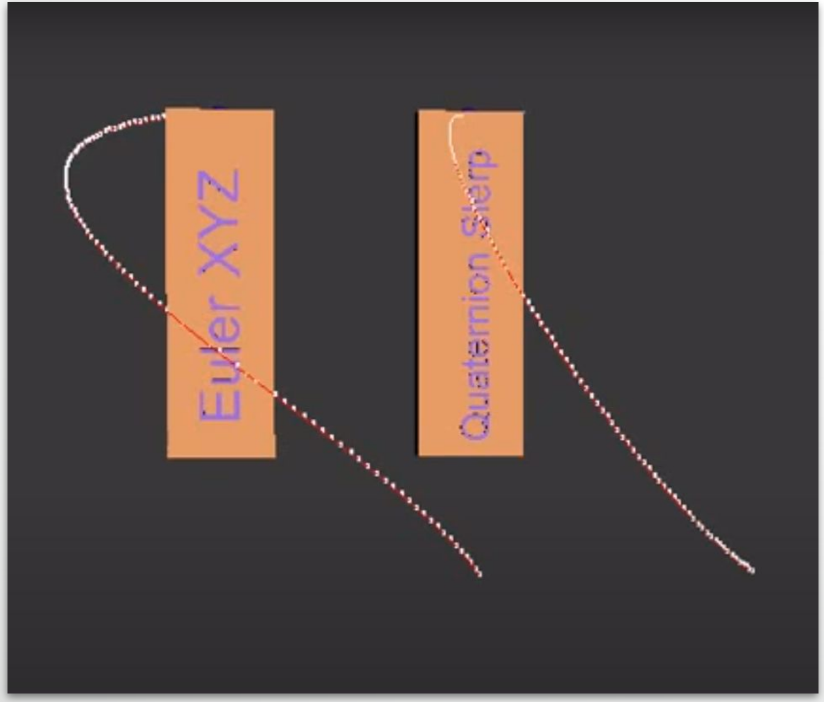
- Cons :
  - Order of rotation sequence (XYZ or ZYX) matters
  - Gimbal lock
  - Performance ?

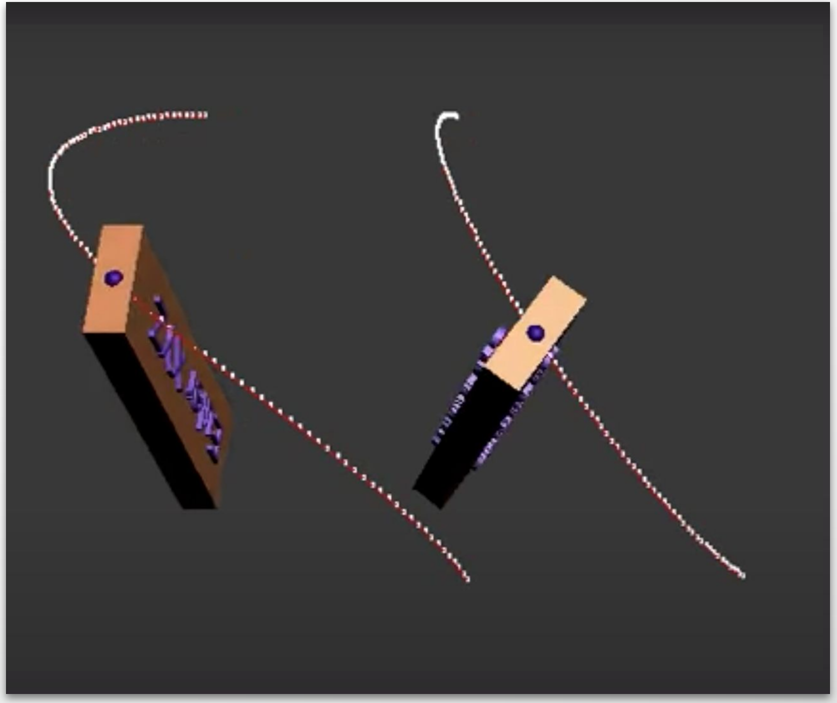
Euler angles

- Cons :
  - Order of rotation sequence (XYZ or ZYX) matters
  - Gimbal lock
  - Performance ?
  - Interpolation ?

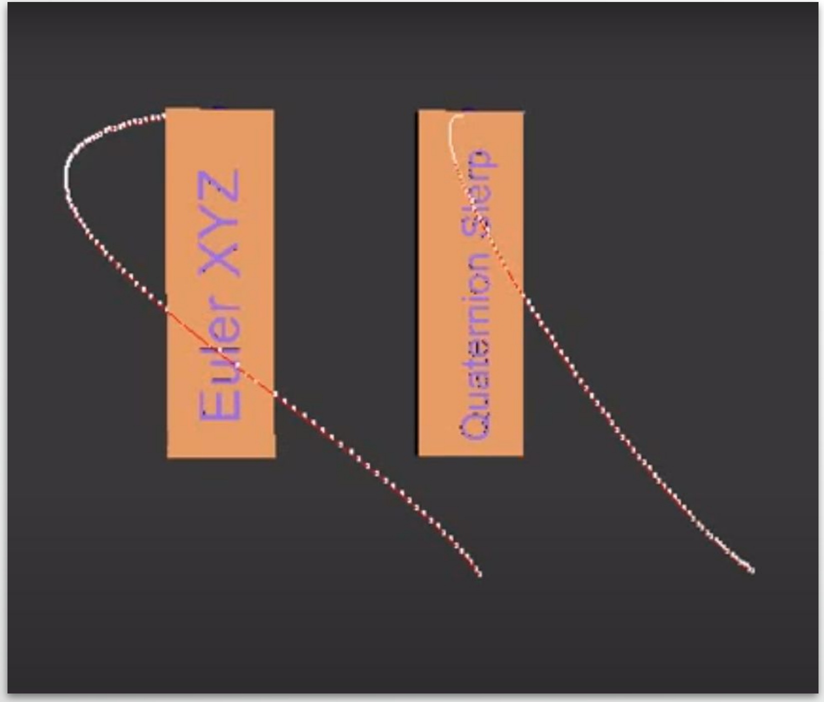
FeisStu

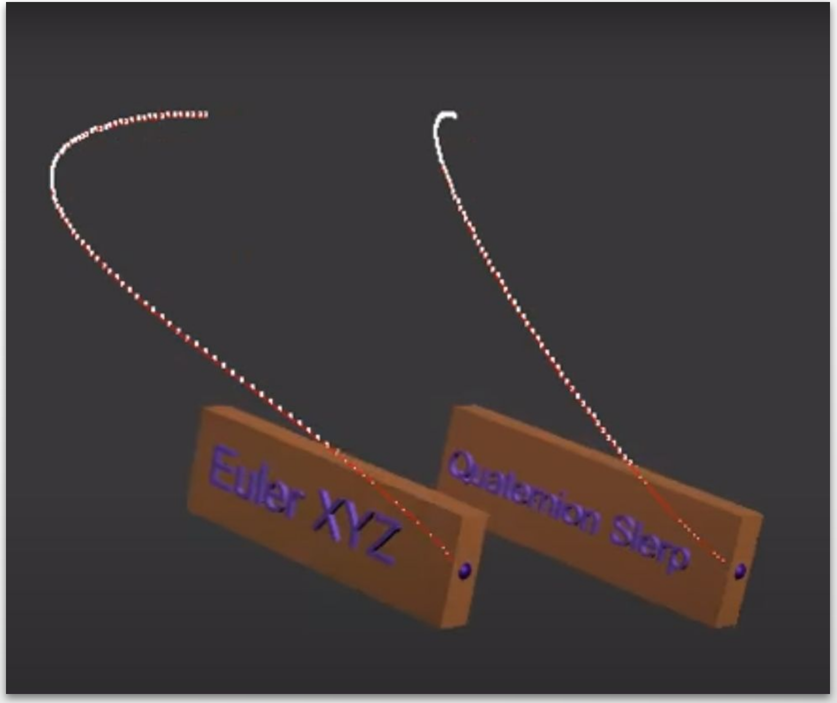
Euler angle vs. quaternion interpolation





Euler angle vs. quaternion interpolation





FeisStu

Euler's rotation theorem

- "When a sphere is moved around its centre it is always possible to find a diameter whose direction in the displaced position is the same as in the initial position."

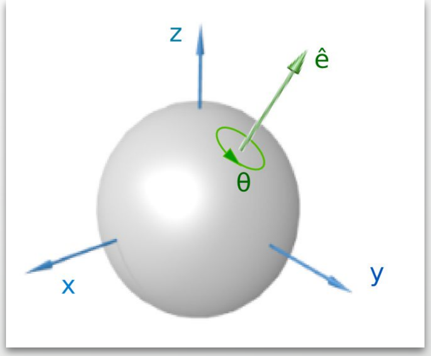
  - Euler (1776)

Euler's rotation theorem

- "When a sphere is moved around its centre it is always possible to find a diameter whose direction in the displaced position is the same as in the initial position."

  \- Euler (1776)

$$R(\hat{\mathbf{e}}, \theta)$$



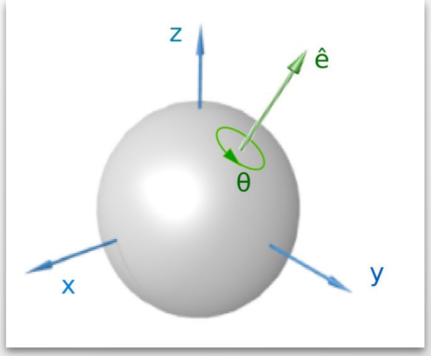
FeisStu

Euler's rotation theorem

- "When a sphere is moved around its centre it is always possible to find a diameter whose direction in the displaced position is the same as in the initial position."

  \- Euler (1776)

$$R(\hat{\mathbf{e}}_1, \theta_1)\ R(\hat{\mathbf{e}}_2, \theta_2) = R(\hat{\mathbf{e}}_3, \theta_3)$$



Complex number

a + b **i**

i^2 **=** -1

(a, b)

Quaternions (四元數)

a + b **i** + c **j** + d **k**

$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$

(a, b, c, d)

Like complex number

FeisStu

Quaternions (四元數)

$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = a + \mathbf{u}$

$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$

(let $\mathbf{u} = b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$)

FeisStu

Quaternions (四元數)

$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = a + \mathbf{u}$

$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$

(let $\mathbf{u} = b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$)

$R_1 = a_1 + \mathbf{u}_1$

$R_2 = a_2 + \mathbf{u}_2$

FeisStu

Quaternions (四元數)

$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = a + \mathbf{u}$

$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$

(let $\mathbf{u} = b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$)

$R_1 = a_1 + \mathbf{u}_1$

$R_2 = a_2 + \mathbf{u}_2$

$R_1R_2 = (a_1a_2 - \mathbf{u}_1 \cdot \mathbf{u}_2)+(a_1\mathbf{u}_2 + a_2\mathbf{u}_1 + \mathbf{u}_1 \times \mathbf{u}_2)$

Quaternions (四元數)

$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = a + \mathbf{u}$

$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$

(let $\mathbf{u} = b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$)

$R_1 = a_1 + \mathbf{u}_1$

$R_2 = a_2 + \mathbf{u}_2$

$$R_1R_2 = \underbrace{(a_1a_2 - \mathbf{u}_1 \cdot \mathbf{u}_2)}_{\text{Real part}} + \underbrace{(a_1\mathbf{u}_2 + a_2\mathbf{u}_1 + \mathbf{u}_1 \times \mathbf{u}_2)}_{\text{Imaginary part}}$$

FeisStu

Quaternions (四元數)

$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = a + \mathbf{u}$

$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$

(let $\mathbf{u} = b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$)

$R_1 = a_1 + \mathbf{u}_1$

$R_2 = a_2 + \mathbf{u}_2$

$$R_1R_2 = \underline{(a_1a_2 - \mathbf{u}_1 \cdot \mathbf{u}_2)} + \underline{(a_1\mathbf{u}_2 + a_2\mathbf{u}_1 + \mathbf{u}_1 \times \mathbf{u}_2)}$$

$$= a_3 + \mathbf{u}_3$$

FeisStu

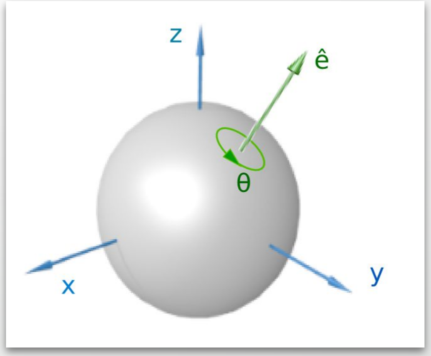
Quaternions (四元數)

$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = a + \mathbf{u}$

$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$

(let $\mathbf{u} = b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$)

Relationship ?



$R_1 = a_1 + \mathbf{u}_1$

$R_2 = a_2 + \mathbf{u}_2$

$R_1R_2 = \underline{(a_1a_2 - \mathbf{u}_1 \cdot \mathbf{u}_2)} + \underline{(a_1\mathbf{u}_2 + a_2\mathbf{u}_1 + \mathbf{u}_1 \times \mathbf{u}_2)}$

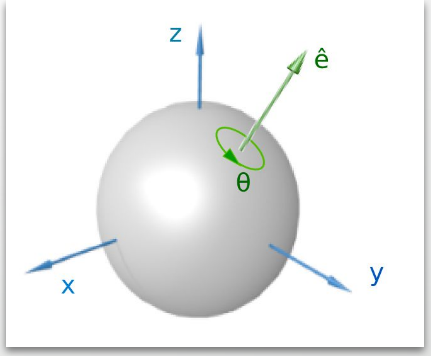
$= a_3 + \mathbf{u}_3$

$(\hat{\mathbf{e}}, \theta)$

FeisStu

Quaternions (四元數)

$$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = \underbrace{\cos(\theta/2)}_{a} + \underbrace{\sin(\theta/2)\,\hat{e}_x}_{b}\,\mathbf{i} + \underbrace{\sin(\theta/2)\,\hat{e}_y}_{c}\,\mathbf{j} + \underbrace{\sin(\theta/2)\,\hat{e}_z}_{d}\,\mathbf{k}$$



$(\hat{\mathbf{e}}, \theta)$

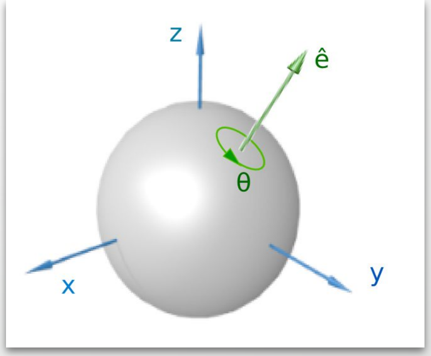
FeisStu

Quaternions (四元數)

$$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = \cos(\theta/2) + \sin(\theta/2)\,\hat{e}_x\,\mathbf{i} + \sin(\theta/2)\,\hat{e}_y\,\mathbf{j} + \sin(\theta/2)\,\hat{e}_z\,\mathbf{k}$$

$$a = \cos(\theta/2)$$

$$\mathbf{u} = \sin(\theta/2)(\hat{e}_x\,\mathbf{i} + \hat{e}_y\,\mathbf{j} + \hat{e}_z\,\mathbf{k})$$



$(\hat{\mathbf{e}}, \theta)$

FeisStu

Quaternions (四元數)

$$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = \cos(\theta/2) + \sin(\theta/2)\,\hat{e}_x\,\mathbf{i} + \sin(\theta/2)\,\hat{e}_y\,\mathbf{j} + \sin(\theta/2)\,\hat{e}_z\,\mathbf{k}$$

$$a = \cos(\theta/2)$$

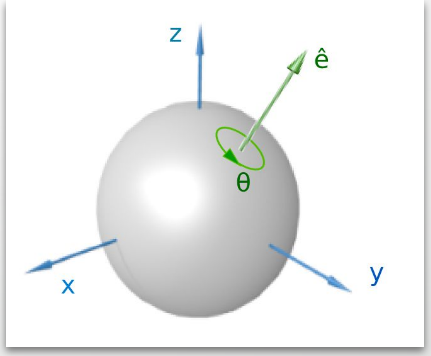
$$\mathbf{u} = \sin(\theta/2)(\hat{e}_x\,\mathbf{i} + \hat{e}_y\,\mathbf{j} + \hat{e}_z\,\mathbf{k})$$

$$R_1 = a_1 + \mathbf{u}_1$$

$$R_2 = a_2 + \mathbf{u}_2$$

$$R_1R_2 = \underline{(a_1a_2 - \mathbf{u}_1 \cdot \mathbf{u}_2)} + \underline{(a_1\mathbf{u}_2 + a_2\mathbf{u}_1 + \mathbf{u}_1 \times \mathbf{u}_2)}$$

$$= a_3 + \mathbf{u}_3$$



$(\hat{\mathbf{e}}, \theta)$

Quaternions (四元數)

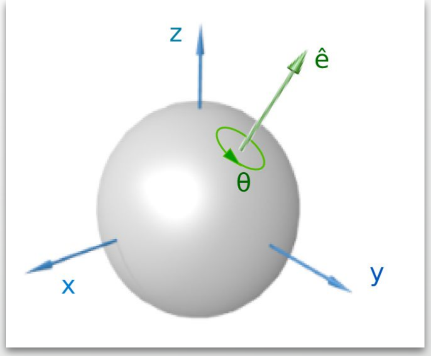
$$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = \cos(\theta/2) + \sin(\theta/2)\,\hat{e}_x\mathbf{i} + \sin(\theta/2)\,\hat{e}_y\mathbf{j} + \sin(\theta/2)\,\hat{e}_z\mathbf{k}$$

$$a = \cos(\theta/2)$$
$$\mathbf{u} = \sin(\theta/2)(\hat{e}_x\mathbf{i} + \hat{e}_y\mathbf{j} + \hat{e}_z\mathbf{k})$$

$$R_1 = a_1 + \mathbf{u}_1$$
$$R_2 = a_2 + \mathbf{u}_2$$

$(\hat{\mathbf{e}}_3, \theta_3)$?

$$R_1R_2 = (a_1a_2 - \mathbf{u}_1 \cdot \mathbf{u}_2) + (a_1\mathbf{u}_2 + a_2\mathbf{u}_1 + \mathbf{u}_1 \times \mathbf{u}_2)$$
$$= a_3 + \mathbf{u}_3$$



$(\hat{\mathbf{e}}, \theta)$

FeisStu

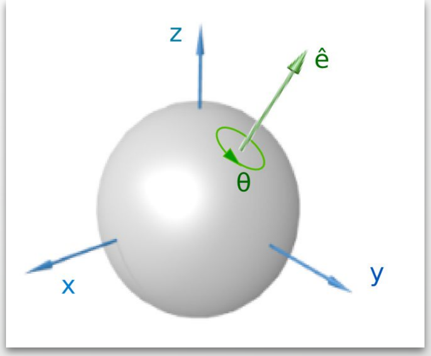
Quaternions (四元數)

$$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = \cos(\theta/2) + \sin(\theta/2)\,\hat{e}_x\,\mathbf{i} + \sin(\theta/2)\,\hat{e}_y\,\mathbf{j} + \sin(\theta/2)\,\hat{e}_z\,\mathbf{k}$$

$$a = \cos(\theta/2)$$

$$\mathbf{u} = \sin(\theta/2)(\hat{e}_x\,\mathbf{i} + \hat{e}_y\,\mathbf{j} + \hat{e}_z\,\mathbf{k})$$

$$R = a + \mathbf{u}$$



Quaternions (四元數)

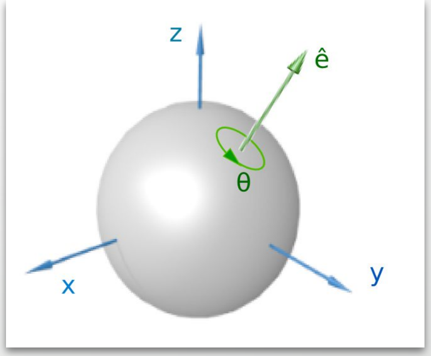
$$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = \cos(\theta/2) + \sin(\theta/2)\,\hat{e}_x\,\mathbf{i} + \sin(\theta/2)\,\hat{e}_y\,\mathbf{j} + \sin(\theta/2)\,\hat{e}_z\,\mathbf{k}$$

$$a = \cos(\theta/2)$$

$$\mathbf{u} = \sin(\theta/2)(\hat{e}_x\,\mathbf{i} + \hat{e}_y\,\mathbf{j} + \hat{e}_z\,\mathbf{k})$$

$$R = a + \mathbf{u}$$

$$\mathbf{p} = p_x\,\mathbf{i} + p_y\,\mathbf{j} + p_z\,\mathbf{k}$$



FeisStu

Quaternions (四元數)

$$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} = \cos(\theta/2) + \sin(\theta/2)\,\hat{e}_x\,\mathbf{i} + \sin(\theta/2)\,\hat{e}_y\,\mathbf{j} + \sin(\theta/2)\,\hat{e}_z\,\mathbf{k}$$

$$a = \cos(\theta/2)$$

$$\mathbf{u} = \sin(\theta/2)(\hat{e}_x\,\mathbf{i} + \hat{e}_y\,\mathbf{j} + \hat{e}_z\,\mathbf{k})$$

$$R = a + \mathbf{u}$$

$$\mathbf{p} = p_x\,\mathbf{i} + p_y\,\mathbf{j} + p_z\,\mathbf{k}$$

$$\mathbf{p}' = R\mathbf{p}R^{-1} = (a + \mathbf{u})\mathbf{p}(a + \mathbf{u})^{-1} = 0 + p'_x\,\mathbf{i} + p'_y\,\mathbf{j} + p'_y\,\mathbf{k}$$

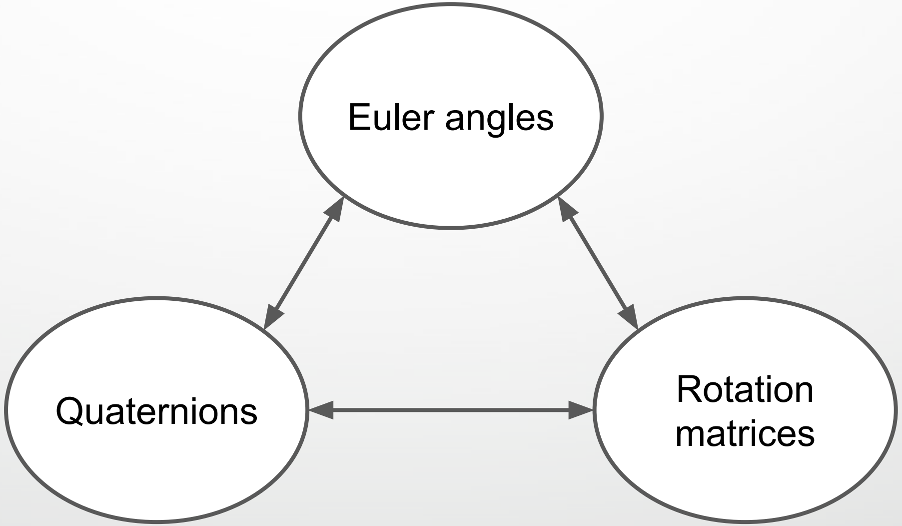
FeisStu

Quaternions visualization

- https://eater.net/quaternions/video/intro

Conversion between different representations

- https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles



UnityEngine.Quaternion

Properties

eulerAngles	Returns or sets the euler angle representation of the rotation.
normalized	Returns this quaternion with a magnitude of 1 (Read Only).
this[int]	Access the x, y, z, w components using [0], [1], [2], [3] respectively.
w	W component of the Quaternion. Do not directly modify quaternions.
x	X component of the Quaternion. Don't modify this directly unless you know quaternions inside out.
y	Y component of the Quaternion. Don't modify this directly unless you know quaternions inside out.
z	Z component of the Quaternion. Don't modify this directly unless you know quaternions inside out.

UnityEngine.Quaternion

Properties

eulerAngles	Returns or sets the euler angle representation of the rotation.
normalized	Returns this quaternion with a magnitude of 1 (Read Only).
this[int]	Access the x, y, z, w components using [0], [1], [2], [3] respectively.
w	W component of the Quaternion. Do not directly modify quaternions.
x	X component of the Quaternion. Don't modify this directly unless y out.
y	Y component of the Quaternion. Don't modify this directly unless you know quaternions inside out.
z	Z component of the Quaternion. Don't modify this directly unless you know quaternions inside out.

DON'T MODIFY THIS DIRECTLY

FeisStu

Static Methods

Angle	Returns the angle in degrees between two rotations a and b.
AngleAxis	Creates a rotation which rotates angle degrees around axis.
Dot	The dot product between two rotations.
Euler	Returns a rotation that rotates z degrees around the z axis, x degrees around the x axis, and y degrees around the y axis; applied in that order.
FromToRotation	Creates a rotation which rotates from fromDirection to toDirection.
Inverse	Returns the Inverse of rotation.
Lerp	Interpolates between a and b by t and normalizes the result afterwards. The parameter t is clamped to the range [0, 1].
LerpUnclamped	Interpolates between a and b by t and normalizes the result afterwards. The parameter t is not clamped.
LookRotation	Creates a rotation with the specified forward and upwards directions.
Normalize	Converts this quaternion to one with the same orientation but with a magnitude of 1.
RotateTowards	Rotates a rotation from towards to.
Slerp	Spherically interpolates between quaternions a and b by ratio t. The parameter t is clamped to the range [0, 1].
SlerpUnclamped	Spherically interpolates between a and b by t. The parameter t is not clamped.

Static Methods

Angle	Returns the angle in degrees between two rotations a and b.
AngleAxis	Creates a rotation which rotates angle degrees around axis.
Dot	The dot product between two rotations.
Euler	Returns a rotation that rotates z degrees around the z axis, x degrees around the x axis, and y degrees around the y axis; applied in that order.
FromToRotation	Creates a rotation which rotates from fromDirection to toDirection.
Inverse	Returns the Inverse of rotation.
Lerp	Interpolates between a and b by t and normalizes the result afterwards. The parameter t is clamped to the range [0, 1].
LerpUnclamped	Interpolates between a and b by t and normalizes the result afterwards. The parameter t is not clamped.
LookRotation	Creates a rotation with the specified forward and upwards directions.
Normalize	Converts this quaternion to one with the same orientation but with a magnitude of 1.
RotateTowards	Rotates a rotation from towards to.
Slerp	Spherically interpolates between quaternions a and b by ratio t. The parameter t is clamped to the range [0, 1].
SlerpUnclamped	Spherically interpolates between a and b by t. The parameter t is not clamped.

FeisStu

UnityEngine.Transform

Public Methods

LookAt	Rotates the transform so the forward vector points at /target/'s current position.
Rotate	Use Transform.Rotate to rotate GameObjects in a variety of ways. The rotation is often provided as an Euler angle and not a Quaternion.
RotateAround	Rotates the transform about axis passing through point in world coordinates by angle degrees.

Translation, rotation and scaling

$$T_{(px,py,pz)} = \begin{bmatrix} & & & px \\ & & & py \\ & & & pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation, rotation and scaling

$$T_{(px,py,pz)}\ R_{(rx,ry,rz)}\ S_{(sx,sy,sz)} = \begin{bmatrix} \boxed{RS} & \begin{matrix} px \\ py \\ pz \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}$$

$$S_{(sx,sy,sz)} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

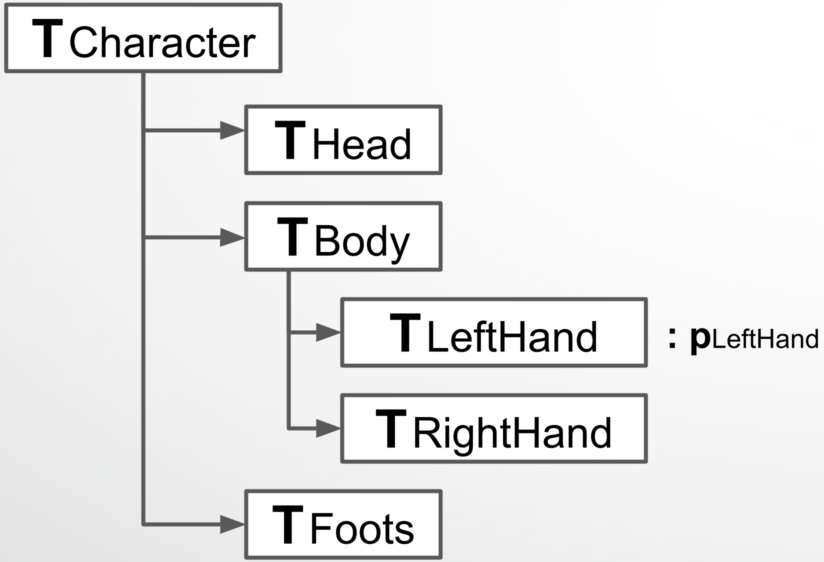
Translation, rotation and scaling

$$TRS_{(px,py,pz,rx,ry,rz,sx,sy,sz)} =$$

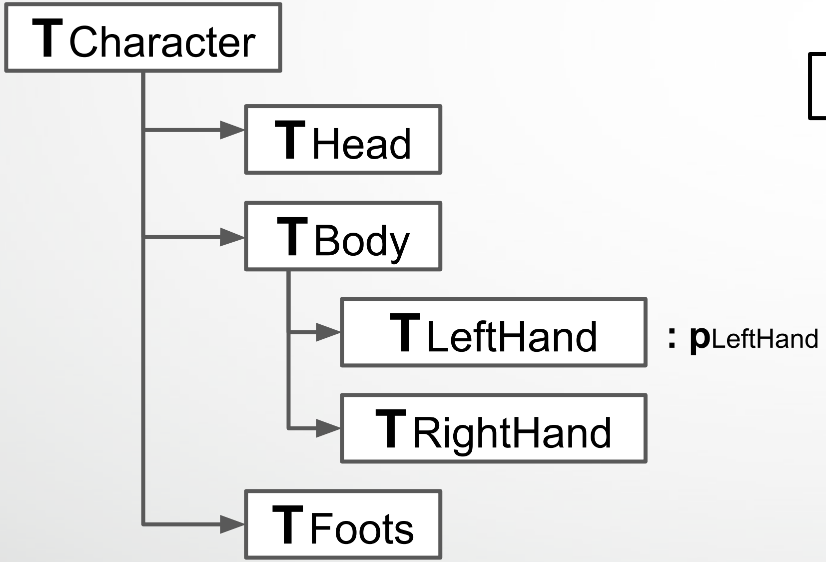
$$T_{(px,py,pz)}\ R_{(rx,ry,rz)}\ S_{(sx,sy,sz)} = \begin{bmatrix} \multicolumn{3}{c}{\multirow{3}{*}{RS}} & px \\ & & & py \\ & & & pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$S_{(sx,sy,sz)} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Object to World coordinates

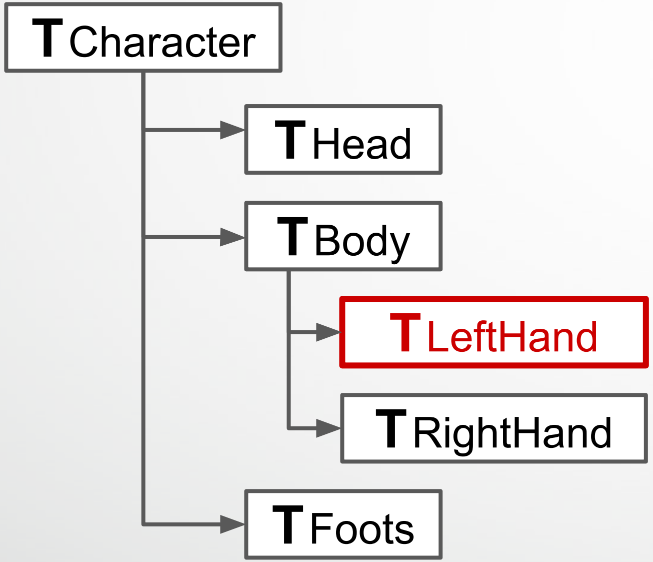


Object to World coordinates



$p'_{LeftHand} = \qquad p_{LeftHand}$

Object to World coordinates

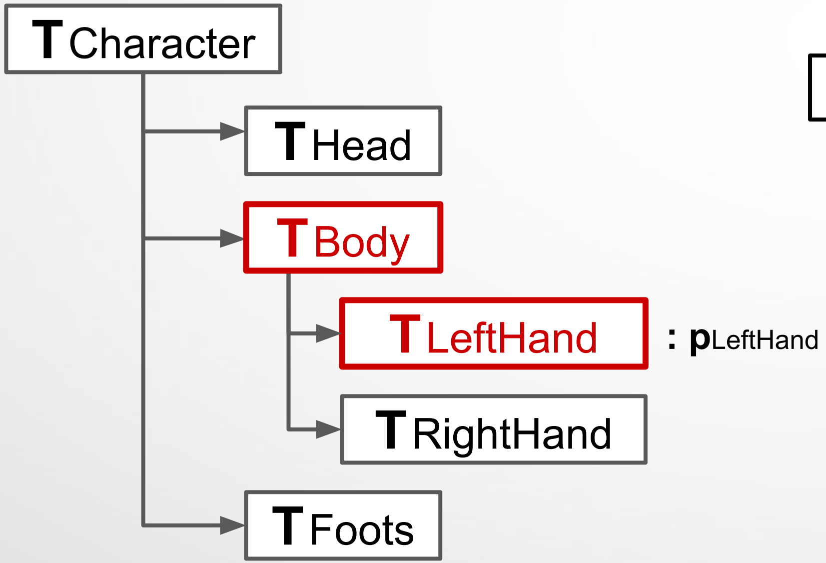


: $p_{LeftHand}$

$$p'_{LeftHand} = T_{LeftHand}\, p_{LeftHand}$$

$$T_{LeftHead} = TRS_{LeftHead}$$

Object to World coordinates



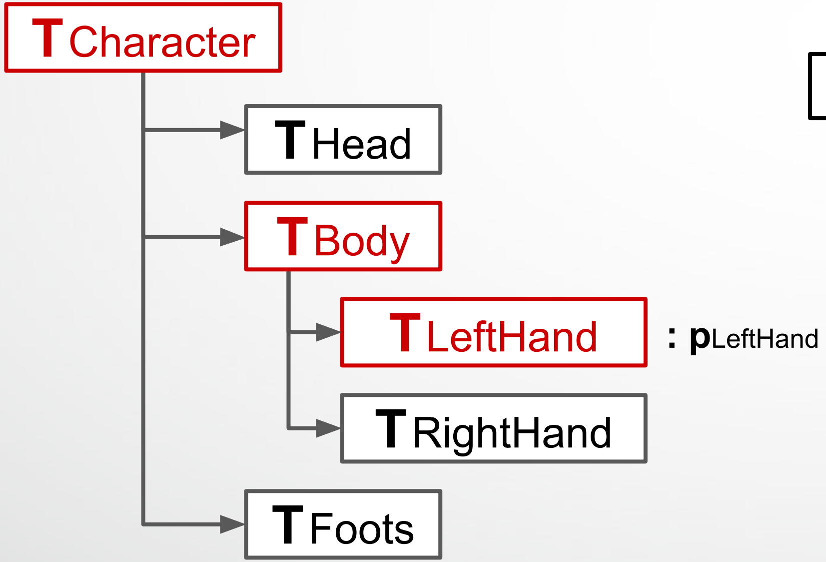
$$\mathbf{p}'_{LeftHand} = \mathbf{T}_{Body}\ \mathbf{T}_{LeftHand}\ \mathbf{p}_{LeftHand}$$

$$\mathbf{T}_{Body} = \mathbf{TRS}_{Body}$$

$$\mathbf{T}_{LeftHead} = \mathbf{TRS}_{LeftHead}$$

FeisStu

Object to World coordinates



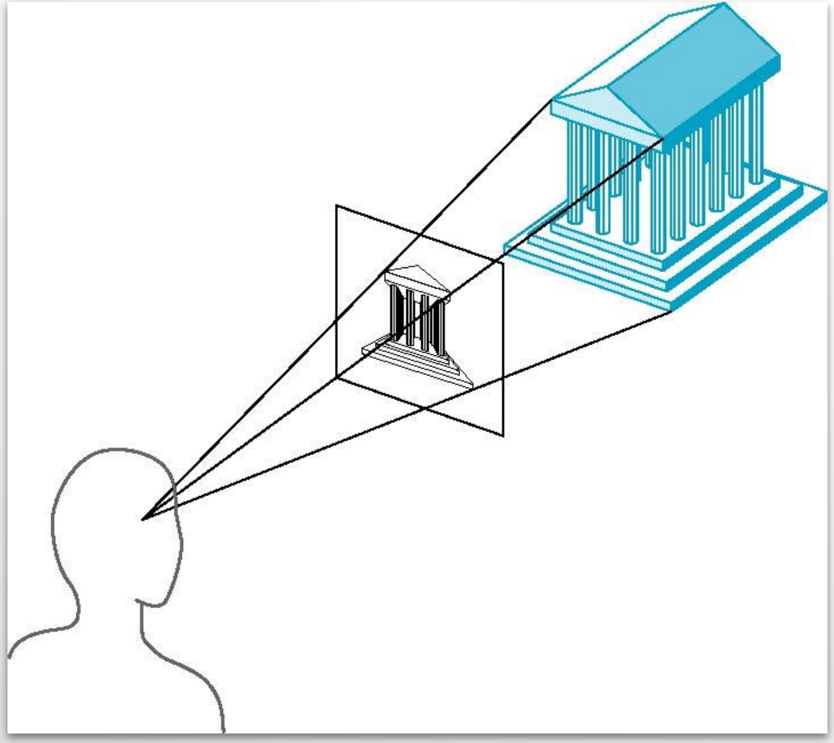
$$\mathbf{p}'_{LeftHand} = \mathbf{T}_{Character}\ \mathbf{T}_{Body}\ \mathbf{T}_{LeftHand}\ \mathbf{p}_{LeftHand}$$

$$\mathbf{T}_{Character} = \mathbf{TRS}_{Character}$$

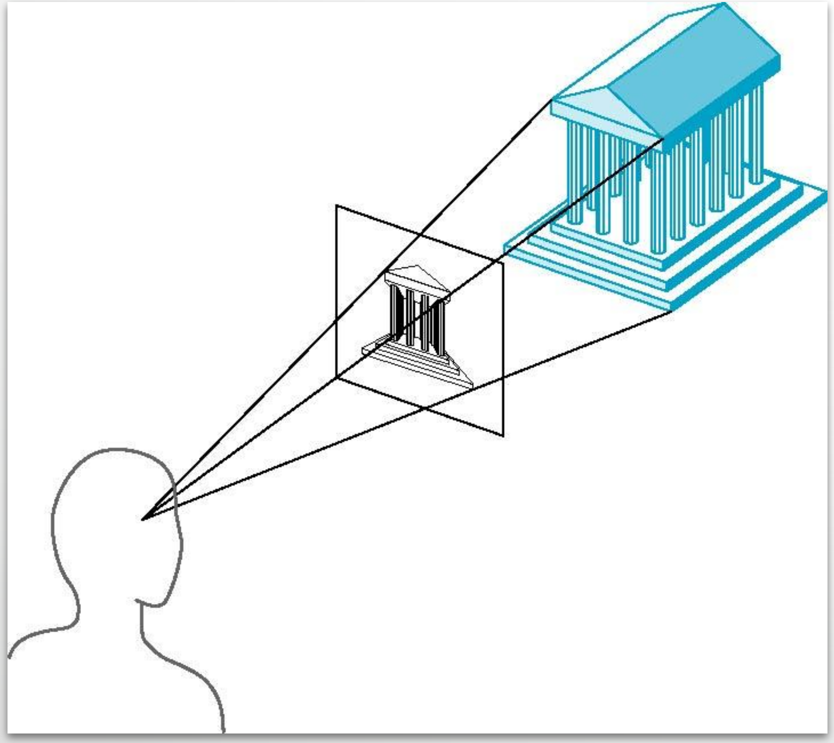
$$\mathbf{T}_{Body} = \mathbf{TRS}_{Body}$$

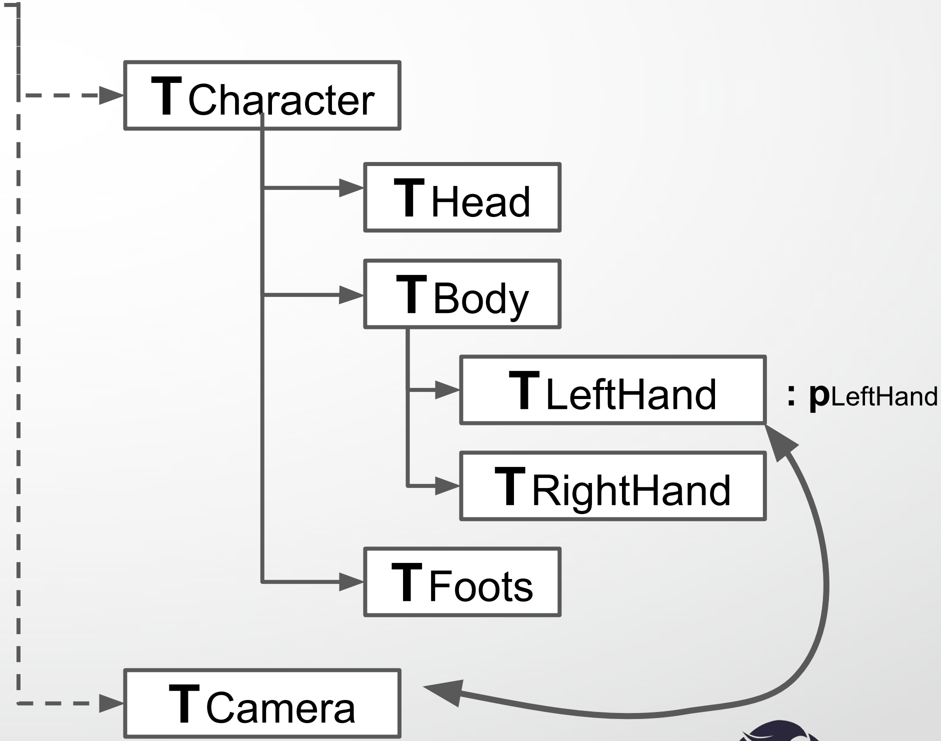
$$\mathbf{T}_{LeftHead} = \mathbf{TRS}_{LeftHead}$$

World to camera coordinates

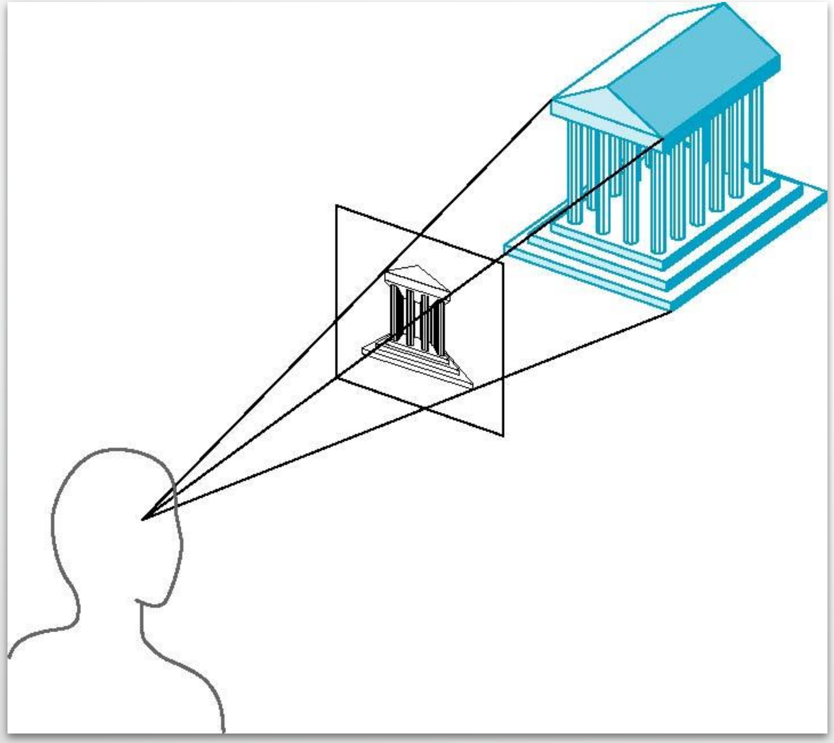


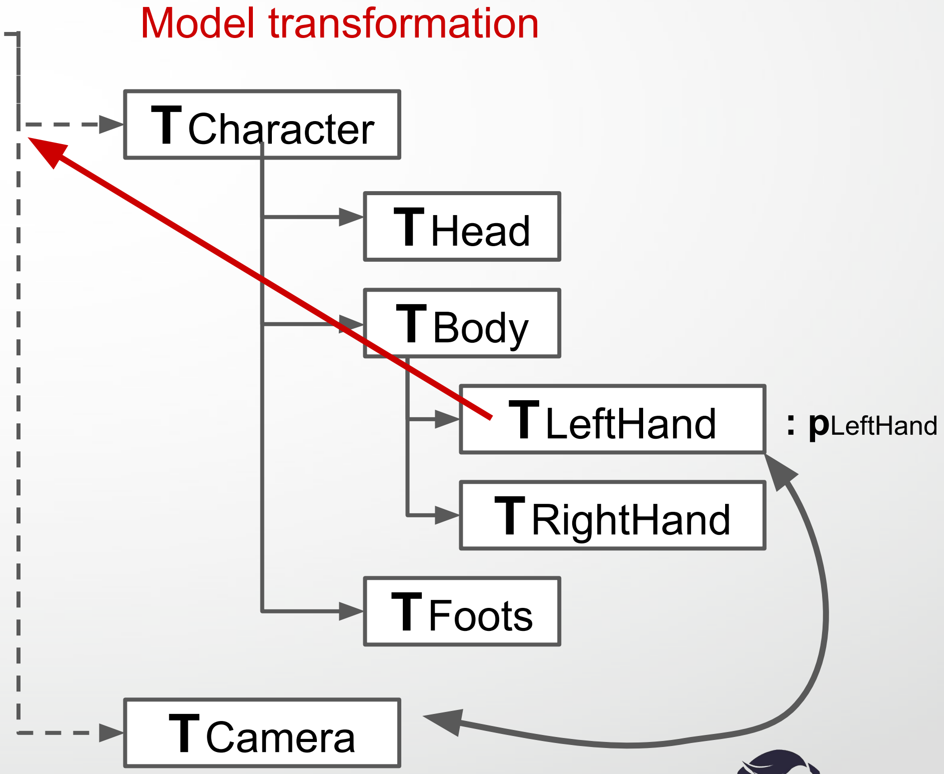
World to camera coordinates



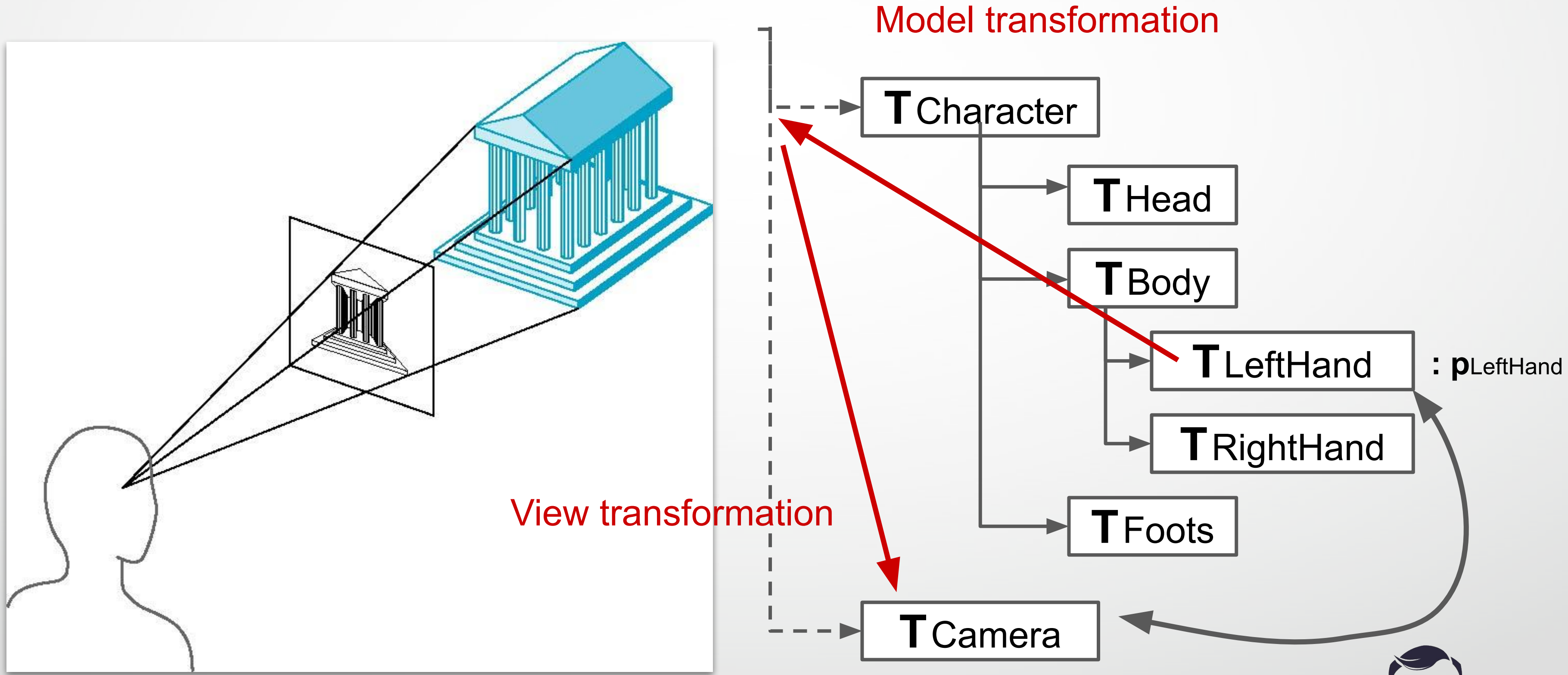


World to camera coordinates



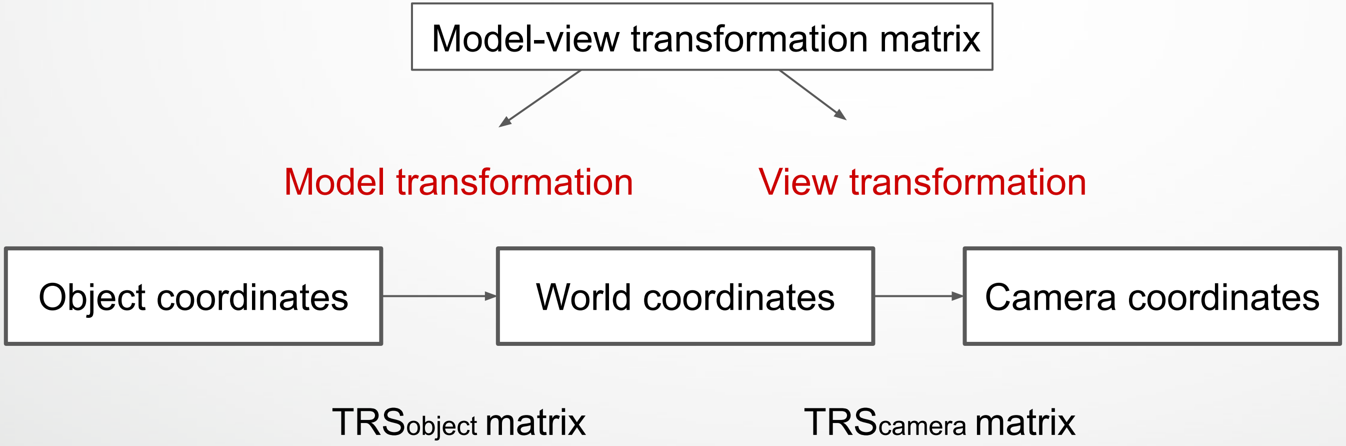


World to camera coordinates

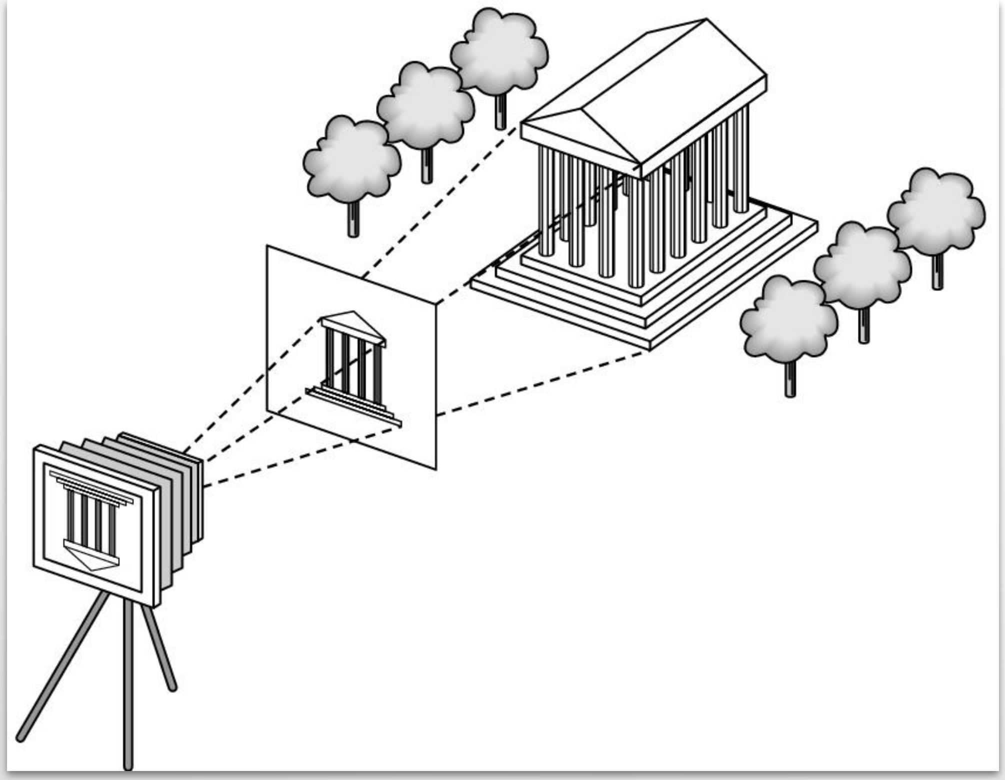


FeisStu

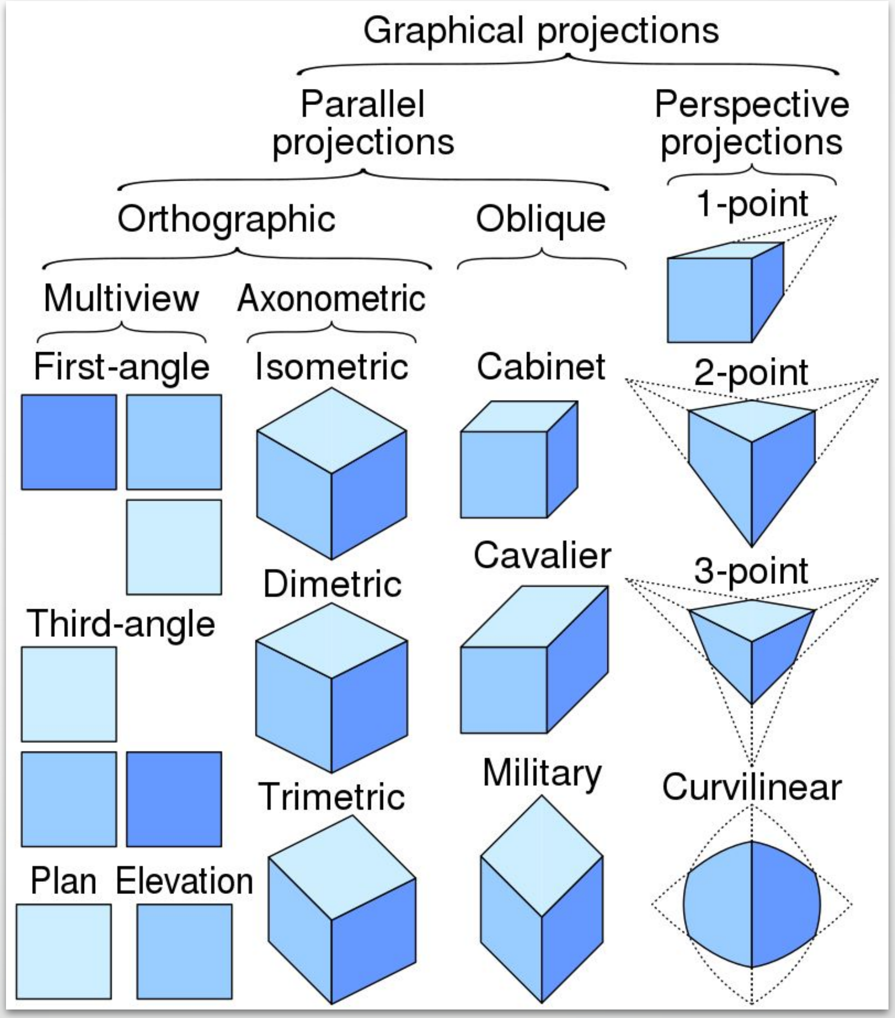
World to camera coordinates



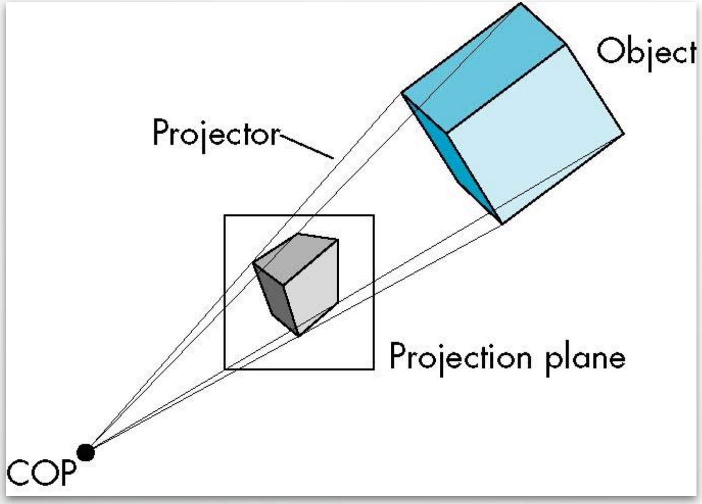
Camera to viewport coordinates

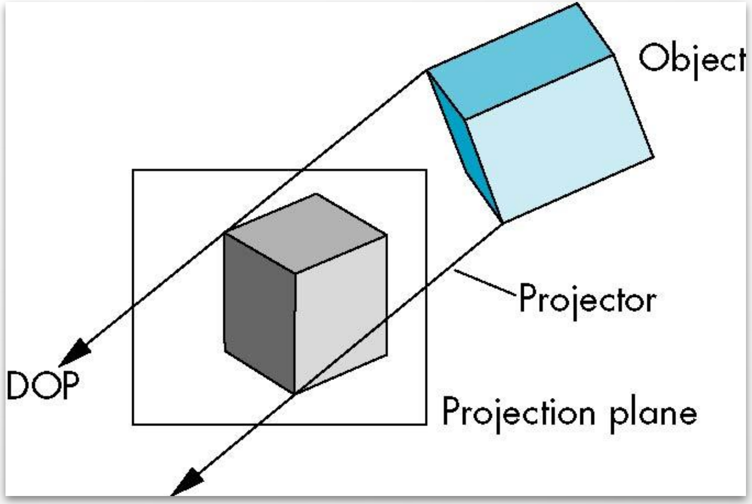


3D Projection

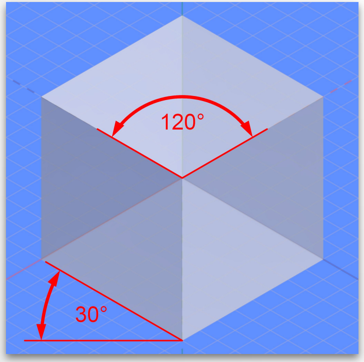


Perspective vs. parallel projections

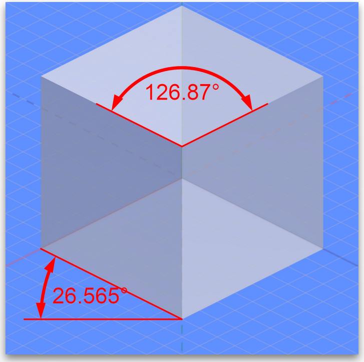




Axonometric projection



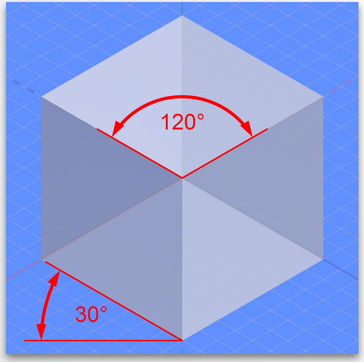
Isometric



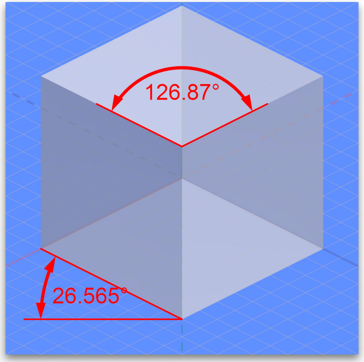
Dimetric

(2.5D ?)

Axonometric projection

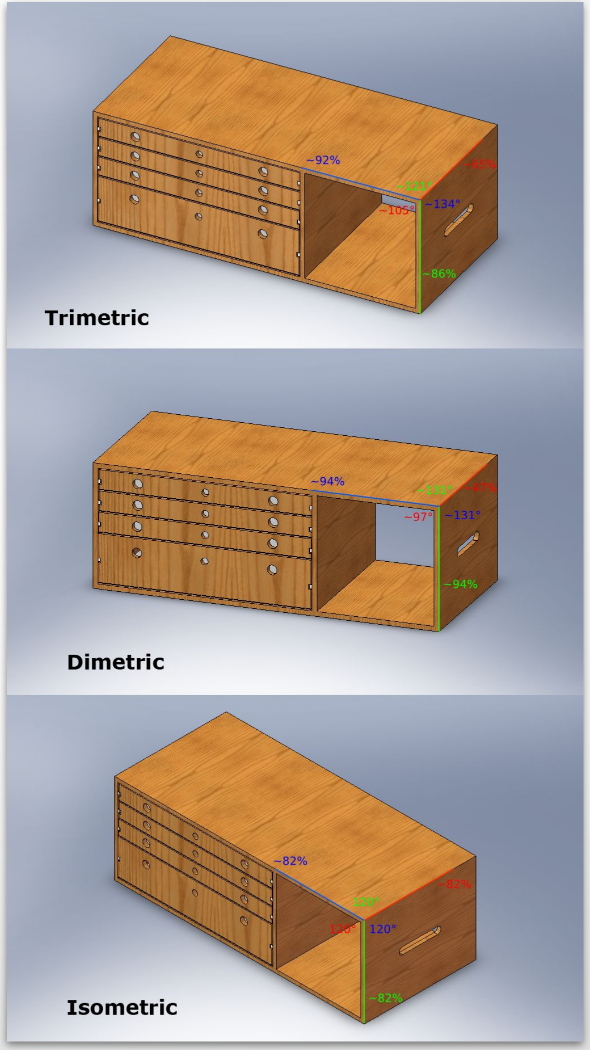


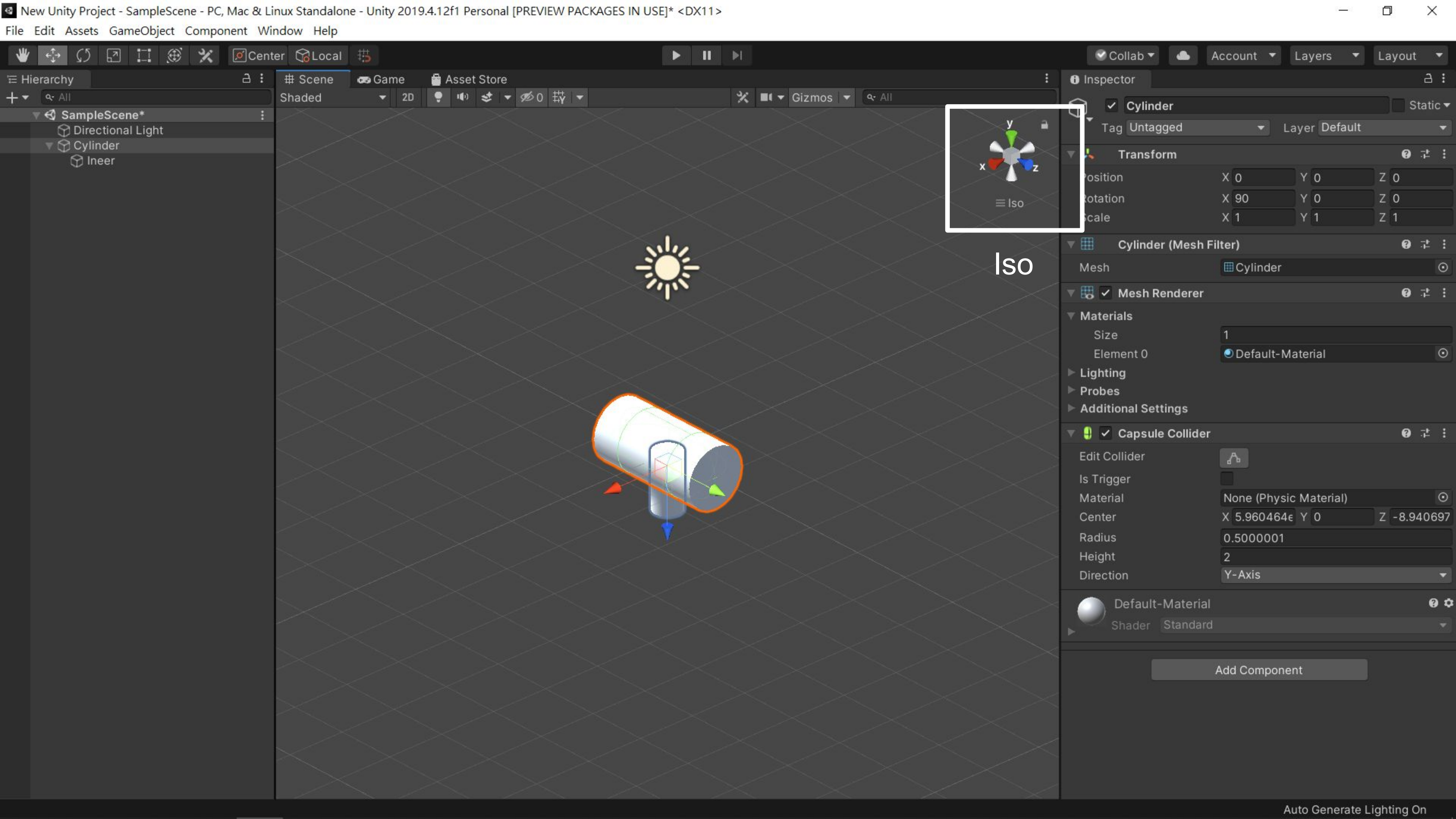
Isometric

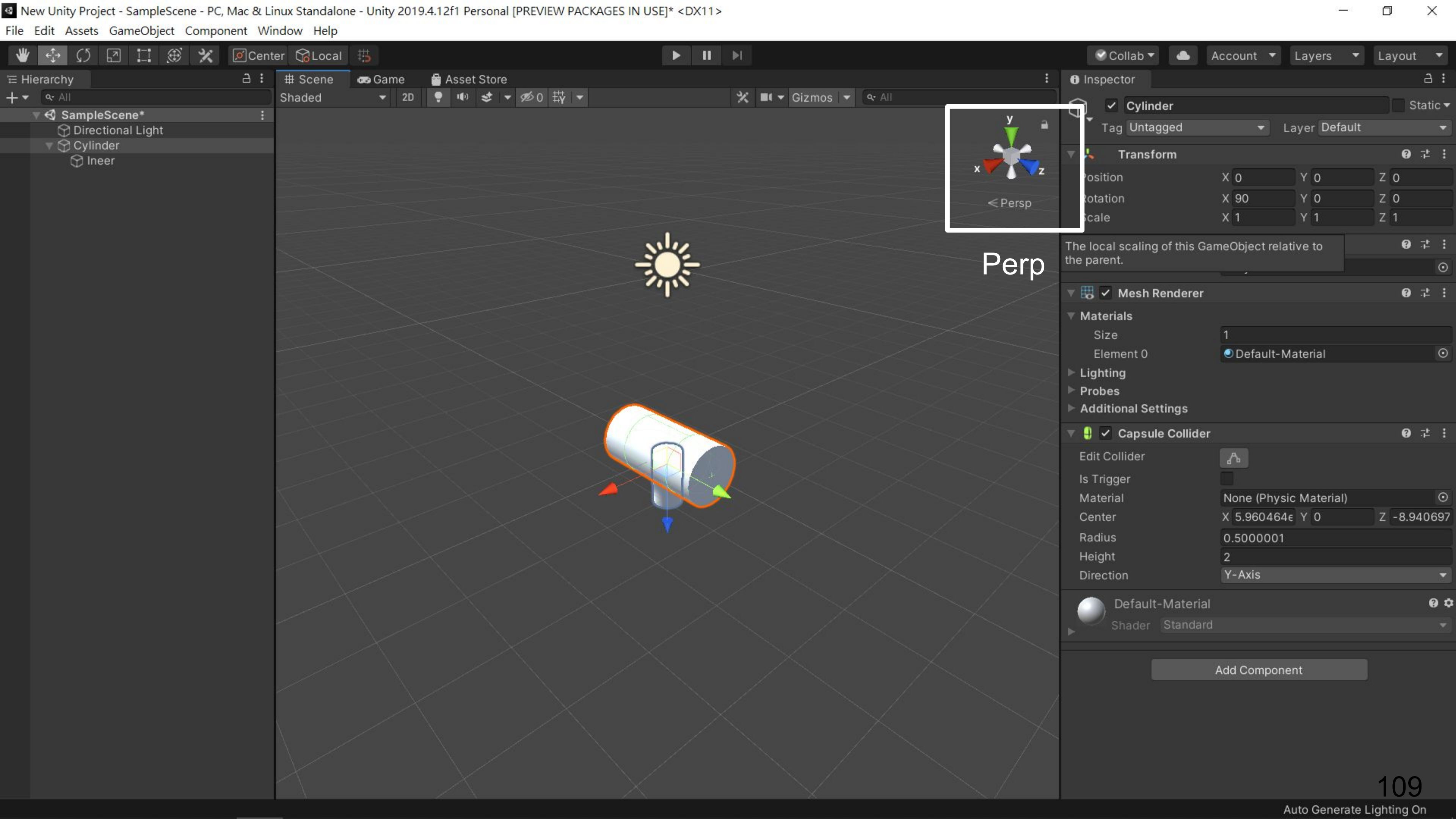


Dimetric

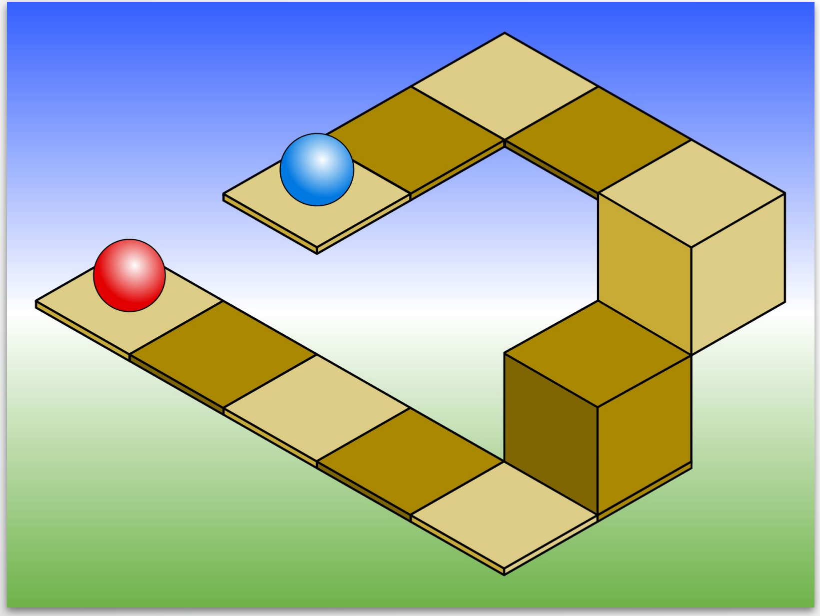
(2.5D ?)



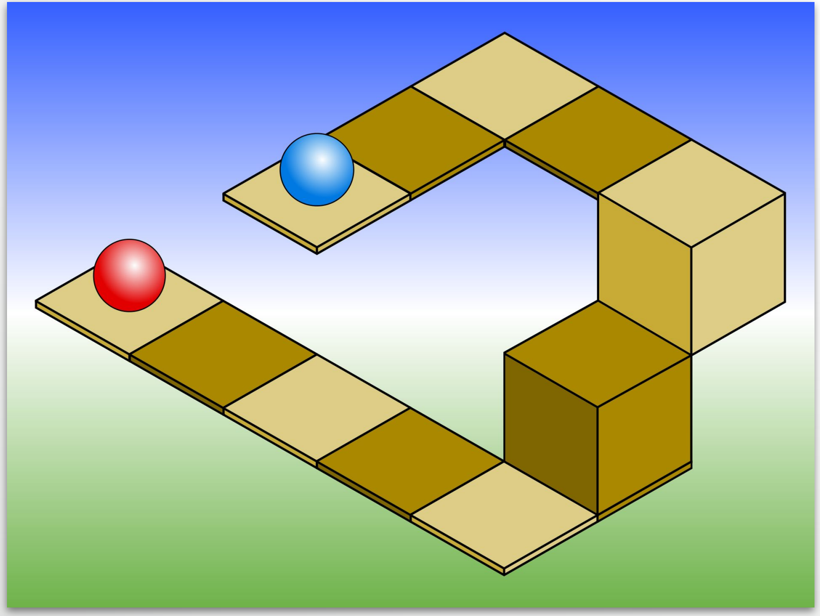

New Unity Project - SampleScene - PC, Mac & Linux Standalone - Unity 2019.4.12f1 Personal [PREVIEW PACKAGES IN USE]* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
SampleScene*
Directional Light
Cylinder
Ineer
Scene Game Asset Store
Shaded 2D Gizmos
Iso
Iso
Inspector
Cylinder
Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 90 Y 0 Z 0
Scale X 1 Y 1 Z 1
Cylinder (Mesh Filter)
Mesh Cylinder
Mesh Renderer
Materials
Size 1
Element 0 Default-Material
Lighting
Probes
Additional Settings
Capsule Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 5.960464e Y 0 Z -8.940697
Radius 0.5000001
Height 2
Direction Y-Axis
Default-Material
Shader Standard
Add Component
Auto Generate Lighting On

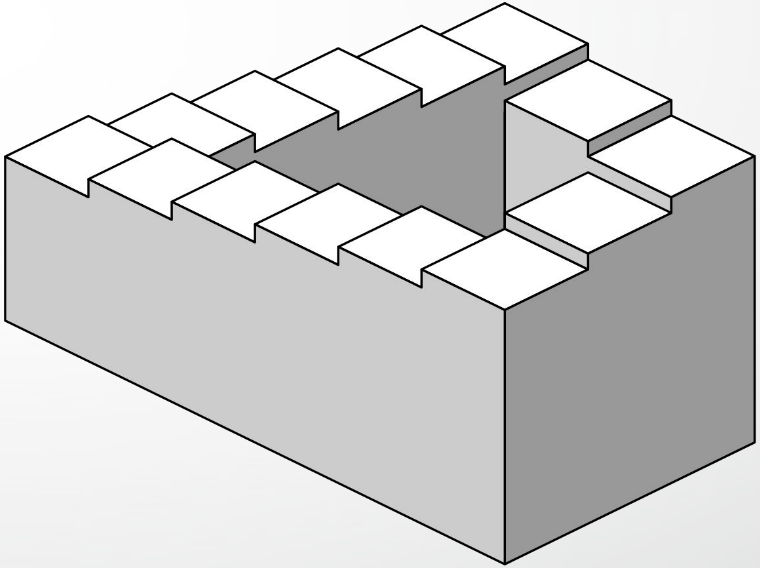

New Unity Project - SampleScene - PC, Mac & Linux Standalone - Unity 2019.4.12f1 Personal [PREVIEW PACKAGES IN USE]* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
SampleScene*
Directional Light
Cylinder
Ineer
Scene Game Asset Store
Shaded 2D Gizmos
Persp
Perp
Inspector
Cylinder Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 90 Y 0 Z 0
Scale X 1 Y 1 Z 1
The local scaling of this GameObject relative to the parent.
Mesh Renderer
Materials
Size 1
Element 0 Default-Material
Lighting
Probes
Additional Settings
Capsule Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 5.960464e Y 0 Z -8.940697
Radius 0.5000001
Height 2
Direction Y-Axis
Default-Material
Shader Standard
Add Component
Auto Generate Lighting On

Limitation of parallel projections

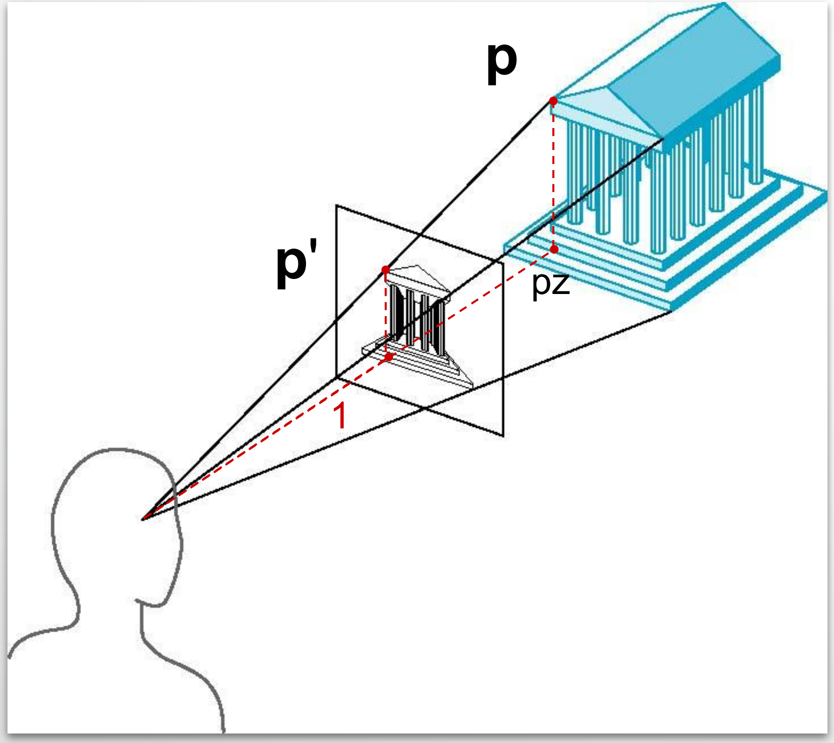


Limitation of parallel projections





Perspective projection



$$\mathbf{p'} = \mathbf{p} / pz = \begin{bmatrix} px/pz \\ py/pz \\ 1 \end{bmatrix}$$

Perspective divide in homogeneous coordinates

$$\mathbf{p}' = \mathbf{p} \;/\; \mathrm{pz} = \begin{bmatrix} \mathrm{px}/\mathrm{pz} \\ \mathrm{py}/\mathrm{pz} \\ 1 \end{bmatrix}$$

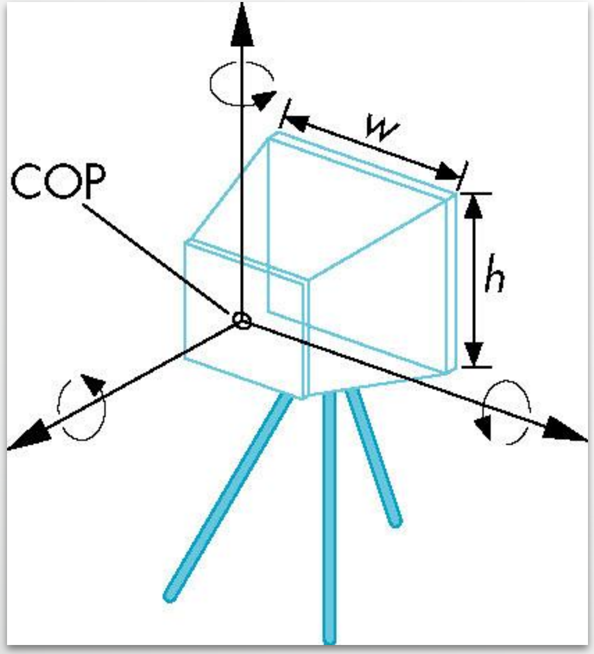
Perspective divide in homogeneous coordinates

$$\mathbf{p'} = \mathbf{p} \,/\, pz = \begin{bmatrix} px/pz \\ py/pz \\ 1 \end{bmatrix}$$

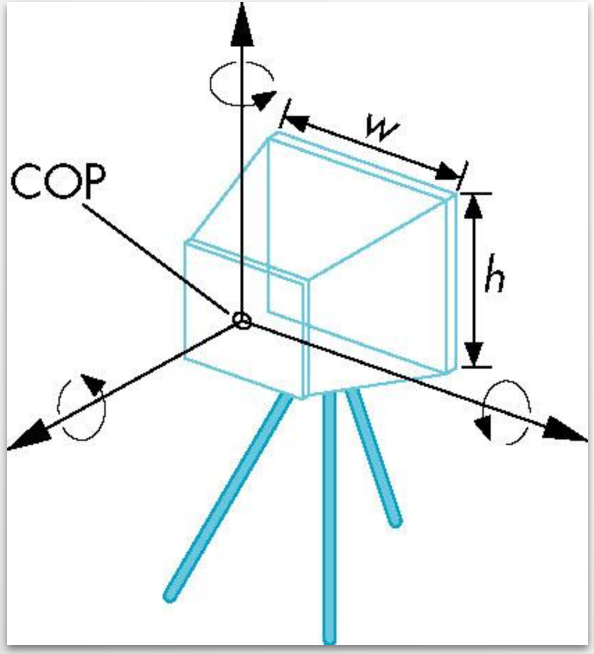
$$\mathbf{p'} = M\,\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} px \\ py \\ pz \\ 1 \end{bmatrix} = \begin{bmatrix} px \\ py \\ pz \\ pz \end{bmatrix} \sim \begin{bmatrix} px/pz \\ py/pz \\ 1 \\ 1 \end{bmatrix}$$

FeisStu

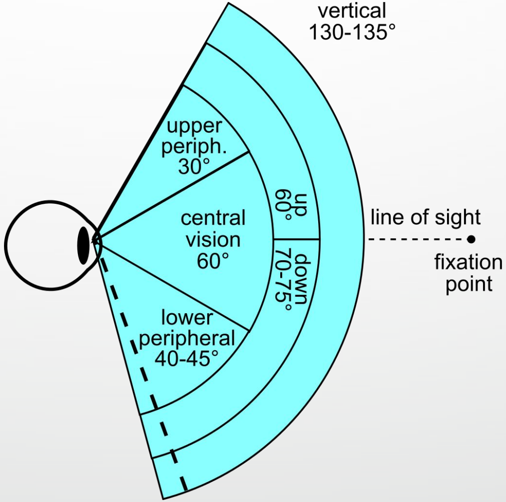
Field of View ?



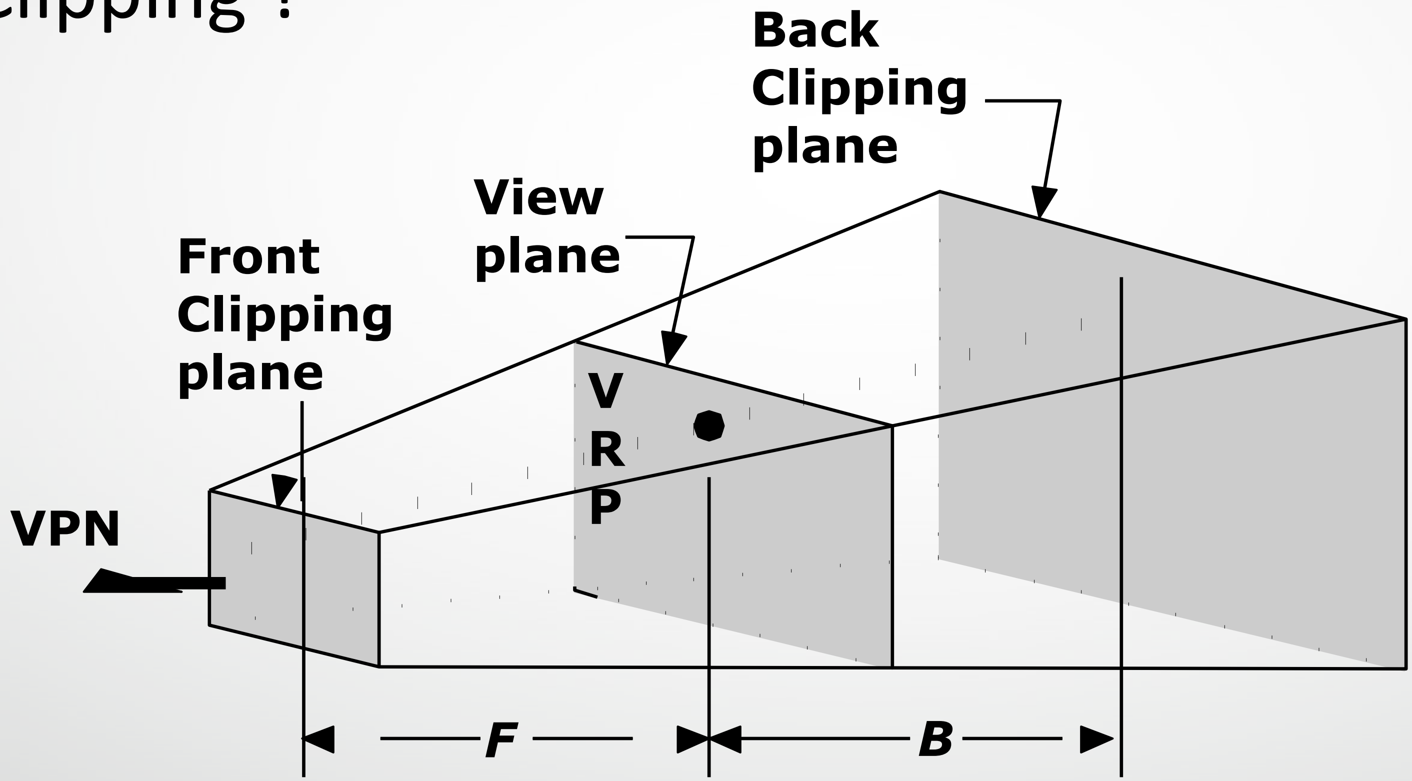
Field of View ?



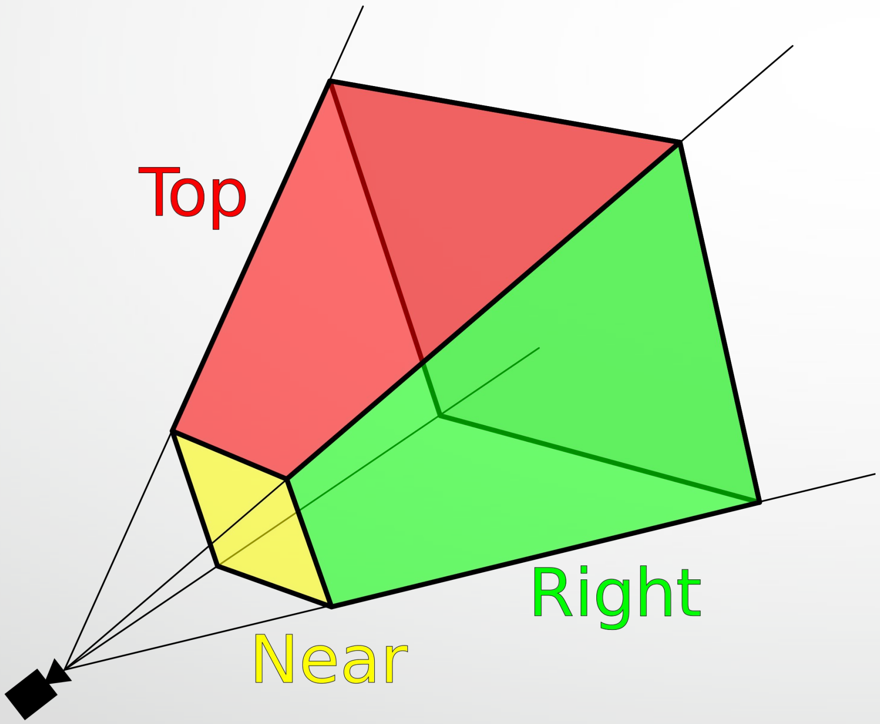
- Vertical field of view + viewport aspect ratio (w/h)



Clipping ?

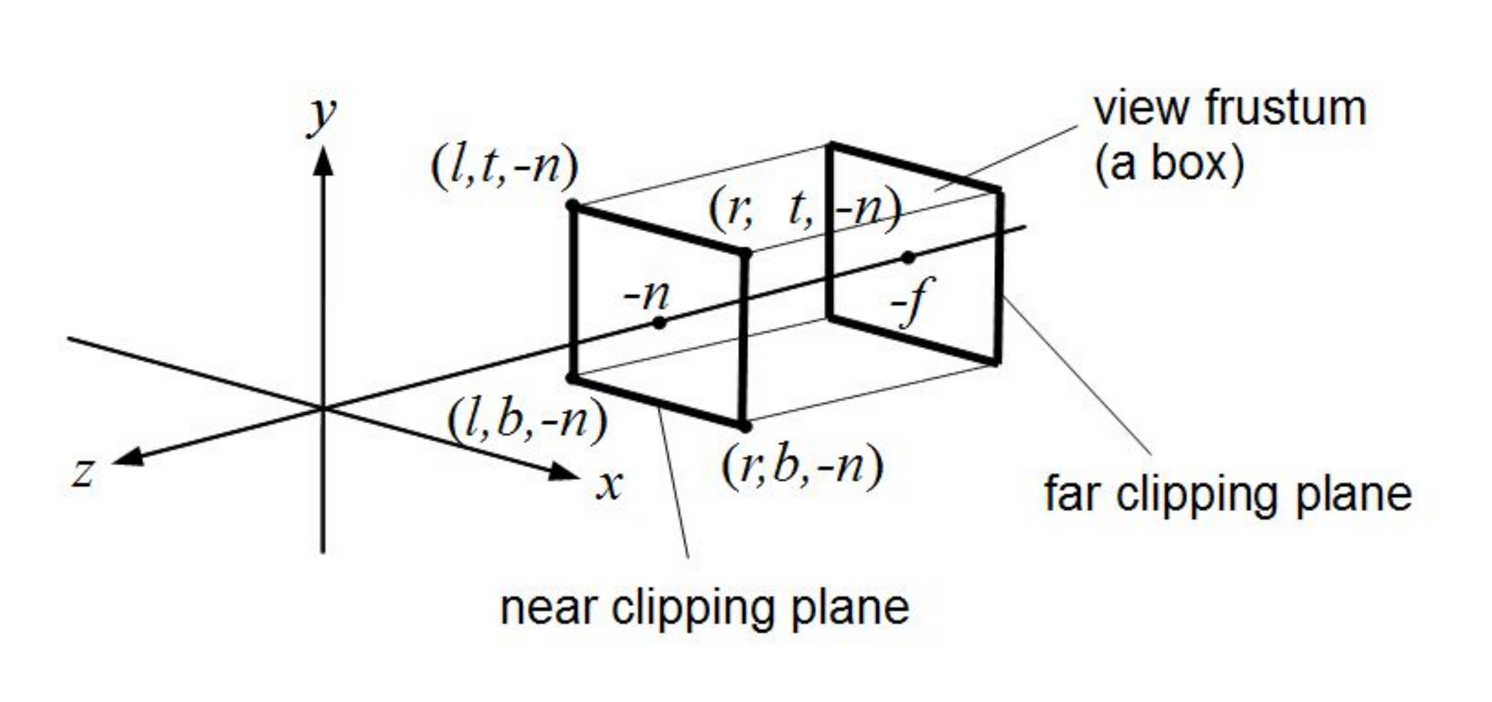

Back
Clipping
plane
View
plane
Front
Clipping
plane
V
R
P
VPN
F
B

Viewing frustum

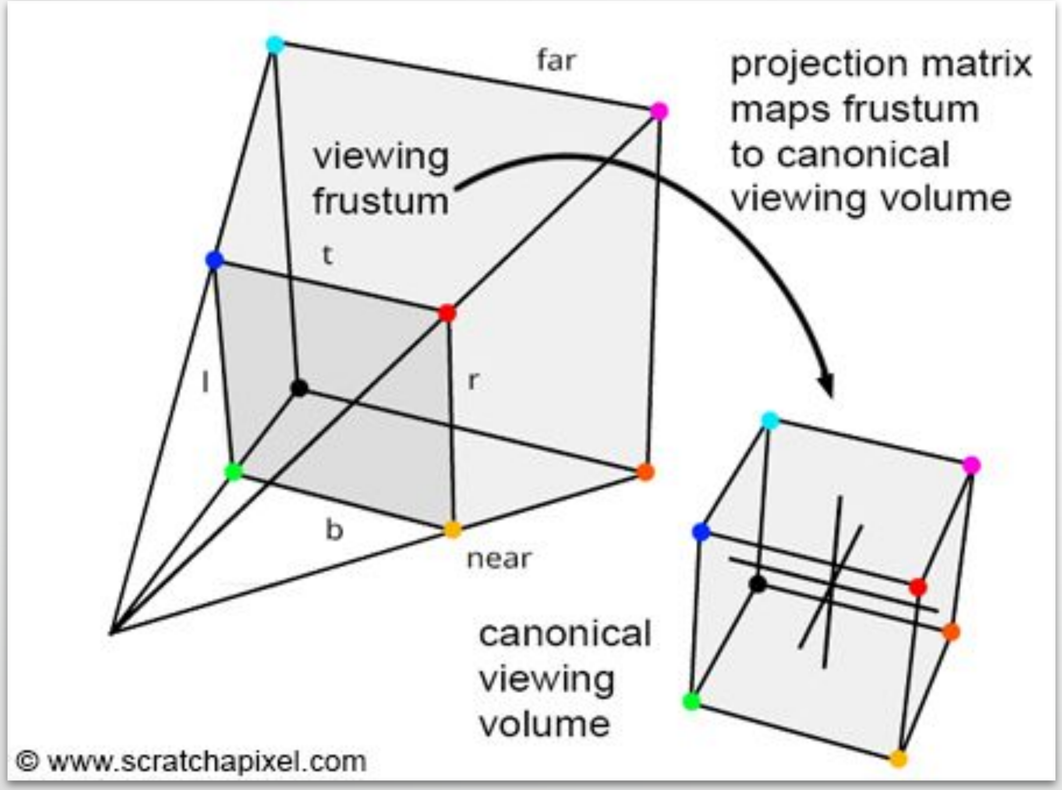


- Vertical field of view + viewport aspect ratio (w/h) + near and far plane

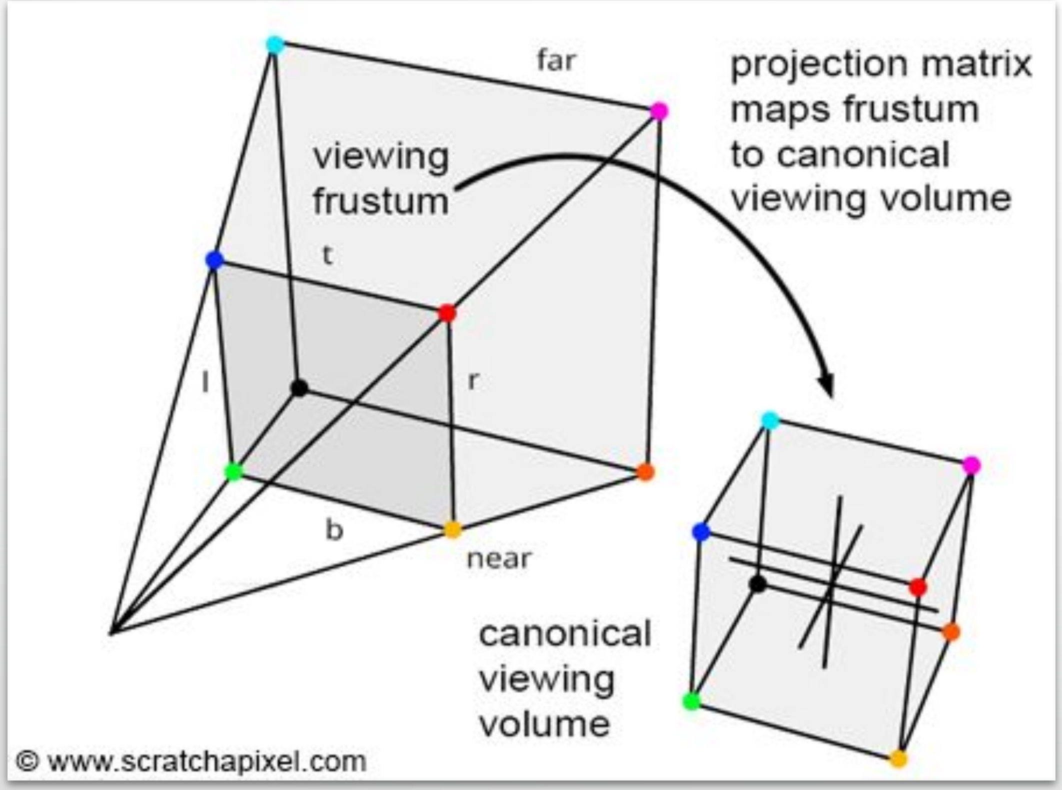
Viewing frustum of parallel projections

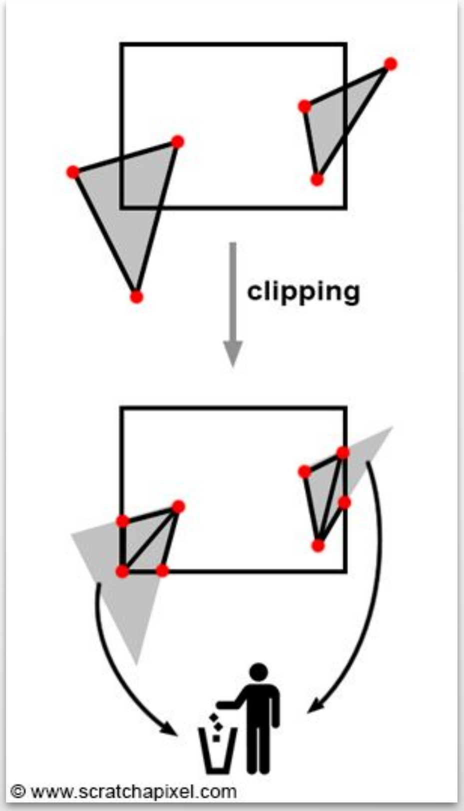


Canonical viewing volume

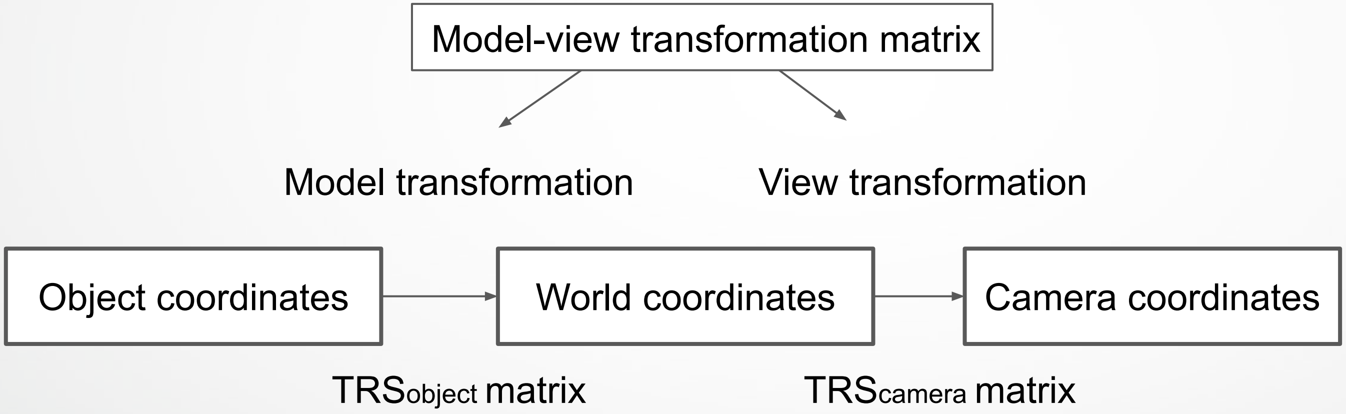


Canonical viewing volume

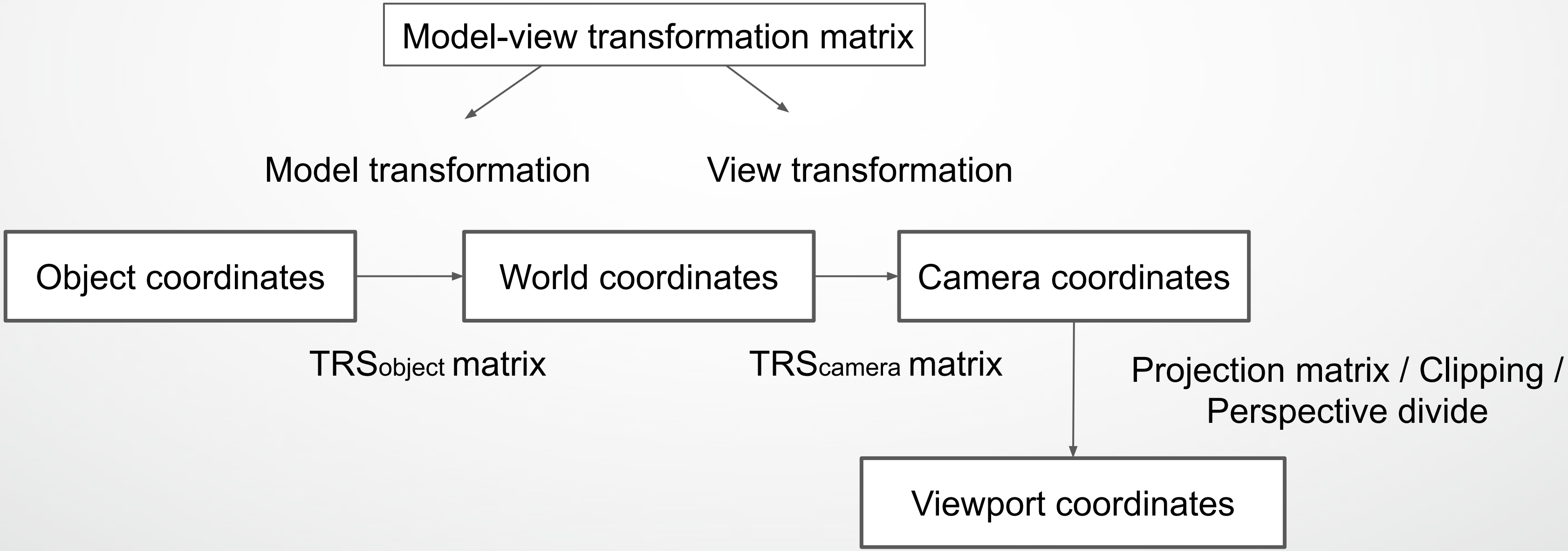




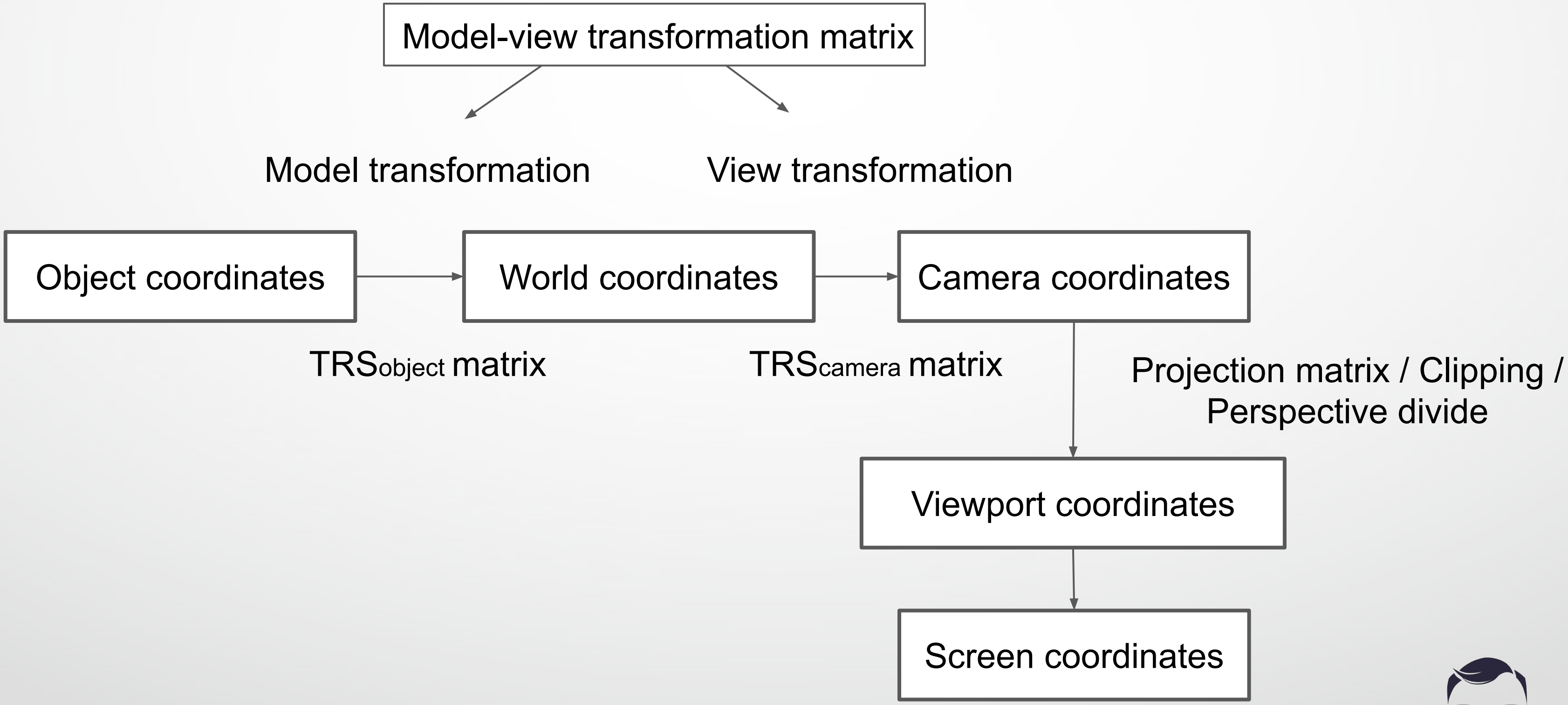
Object to screen coordinates



Object to screen coordinates

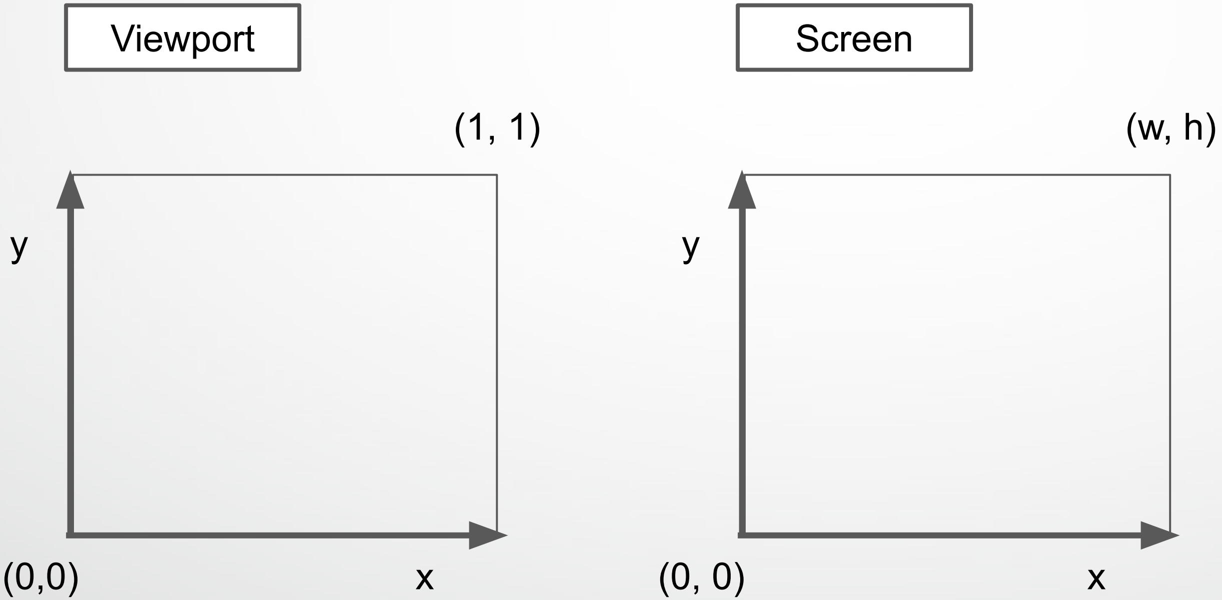


Object to screen coordinates

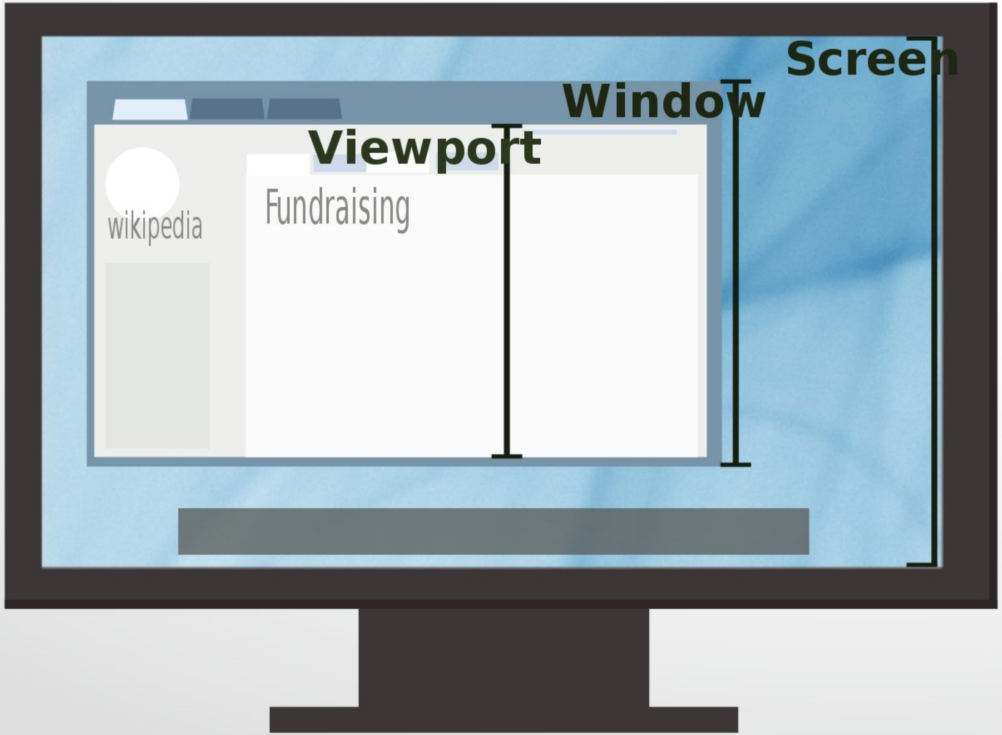


FeisStu

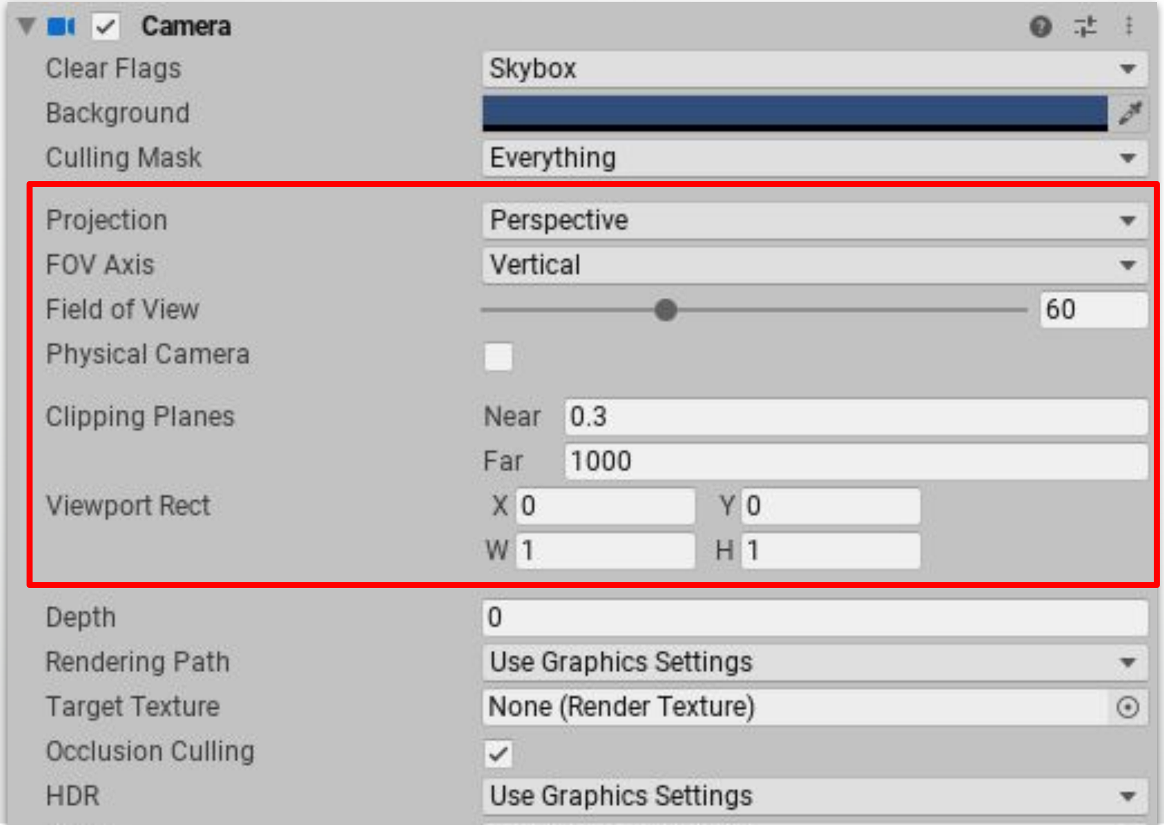
Viewport to screen coordinates



Viewport to screen coordinates



Camera component



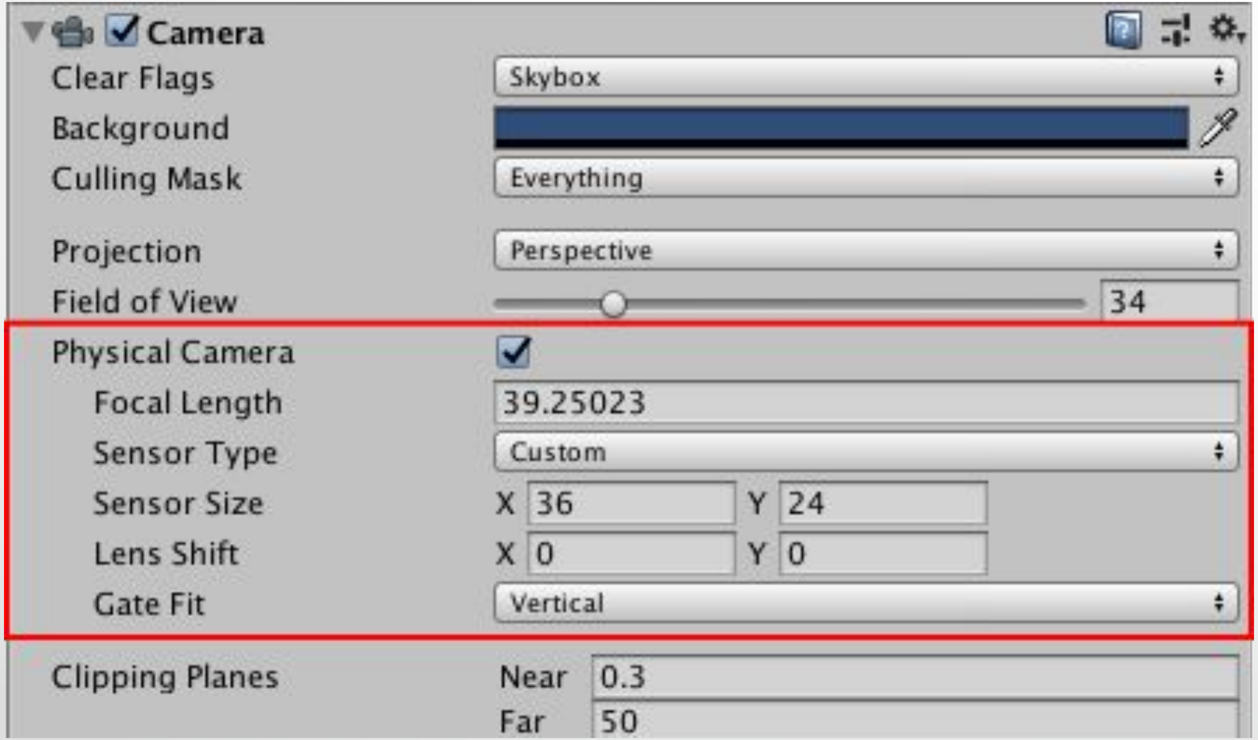
FeisStu

UnityEngine.Camera

Public Methods

ScreenPointToRay	Returns a ray going from camera through a screen point.
ScreenToViewportPoint	Transforms position from screen space into viewport space.
ScreenToWorldPoint	Transforms a point from screen space into world space, where world space is defined as the coordinate system at the very top of your game's hierarchy.
ViewportPointToRay	Returns a ray going from camera through a viewport point.
ViewportToScreenPoint	Transforms position from viewport space into screen space.
ViewportToWorldPoint	Transforms position from viewport space into world space.
WorldToScreenPoint	Transforms position from world space into screen space.
WorldToViewportPoint	Transforms position from world space into viewport space.

Unity physical camera



Q & A

FeisStu