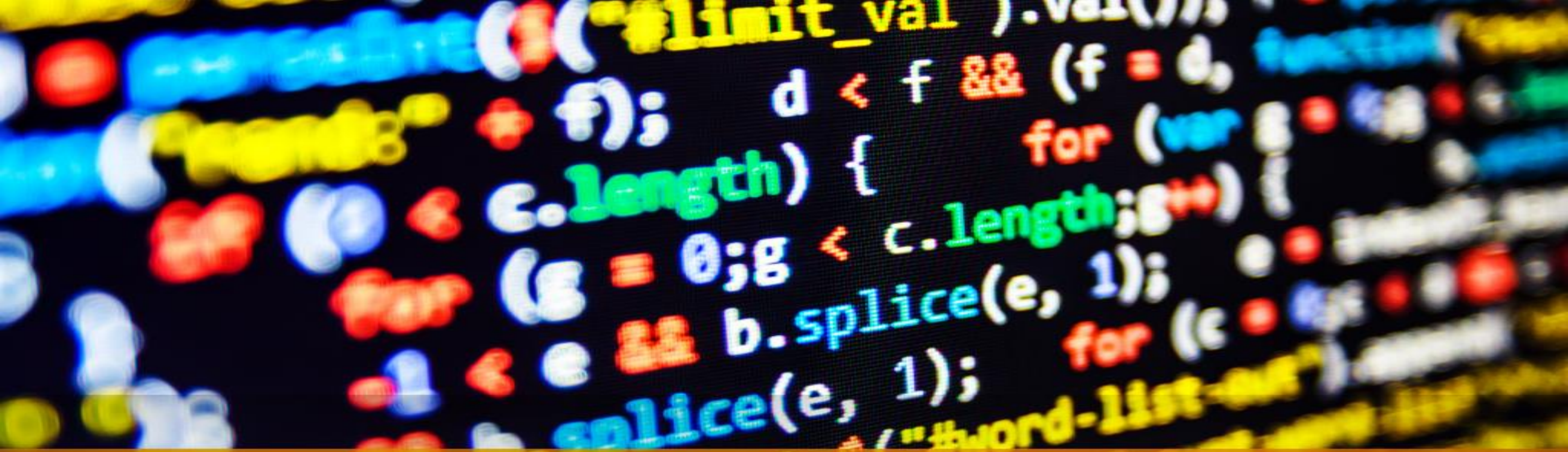


Lecture 05

Modelling & Transforms (Part 2)

Prepared by Ban Kar Weng (William)

Transformation (Part 2)

[Homogeneous Coordinates]

Homogeneous Coordinates | **Motivation**

A summary of basic transformations:

Translation:

$$p' = p + t$$

Rotation:

$$p' = R_a(\theta)p$$

Scaling:

$$p' = S(\vec{s})p$$

where:

p and p' are column vectors for coordinate positions.
t is a vector containing translational terms.
$R_a(\theta)$ is a rotation matrix at angle θ about the axis a.
$S(\vec{s})$ is a scaling matrix where $\vec{s}$ is a vector of scaling factors.

Homogeneous Coordinates | **Motivation**

Transformation like translation, rotation, and scaling can be expressed in a general matrix form:

$$p' = Mp + t$$

where:

p and p' are column vectors for coordinate positions.
M is a matrix containing multiplicative factors for p.
t is a vector containing the translational terms.

Homogeneous Coordinates | **Motivation**

Example 1 (Translation):

$$p' = p \ + t$$
$$p' = Ip + t$$

where:

$M = I$, the identity matrix

Homogeneous Coordinates | **Motivation**

Example 2 (Rotation):

$$p' = R_a(\theta)p$$
$$p' = R_a(\theta)p + \vec{0}$$

where:

$M = R_a(\theta)$

$t = \vec{0}$, the zero vector.

Homogeneous Coordinates | **Motivation**

Example 3 (Scaling):

$$p' = S(\vec{s})p$$
$$p' = S(\vec{s})p + \vec{0}$$

where:

$M = S(\vec{s})$

$t = \vec{0}$, the zero vector.

Homogeneous Coordinates | **Motivation**

Concatenating transformations **without translation** (e.g. a series of rotation and scaling only) is simple and elegant as the concatenated transformation can be combined into a single matrix using matrix multiplication.

Example 1 – Rotation followed by Scaling:

$$p' = S(\vec{s})\left[R_a(\theta)p + \vec{0}^T\right] + \vec{0}^T = \boldsymbol{S}(\vec{\boldsymbol{s}})\boldsymbol{R_a}(\boldsymbol{\theta})p = \boldsymbol{G}p$$

Example 2 – Scaling followed by Rotation:

$$p' = R_a(\theta)\left[S(\vec{s})p + \vec{0}^T\right] + \vec{0}^T = \boldsymbol{R_a}(\boldsymbol{\theta})\boldsymbol{S}(\vec{\boldsymbol{s}})p = \boldsymbol{G}p$$

Homogeneous Coordinates | **Motivation**

Concatenating transformations **with translation** does not results in a single matrix that combines all transformations.

Example 1 – Translation ($\boldsymbol{t_1}$), followed by Scaling, followed by Translation ($\boldsymbol{t_2}$):

$$p' = I\left[S(\vec{s})[Ip + t_1] + \vec{0}^T\right] + t_2 = I[S(\vec{s})p + S(\vec{s})t_1] + t_2 = \boldsymbol{S}(\vec{\boldsymbol{s}})\boldsymbol{p} + \boldsymbol{S}(\vec{\boldsymbol{s}})\boldsymbol{t_1} + \boldsymbol{t_2}$$

Example 2 – Scaling ($\boldsymbol{S_1}(\vec{\boldsymbol{s}})$), followed by Translation ($\boldsymbol{t}$), followed by Scaling ($\boldsymbol{S_2}(\vec{\boldsymbol{s}})$):

$$\begin{aligned} p' &= S_2(\vec{s})\left[I\left[S_1(\vec{s})p + \vec{0}^T\right] + t\right] + \vec{0}^T = S_2(\vec{s})[S_1(\vec{s})p + t] + \vec{0}^T \\ &= \boldsymbol{S_2}(\vec{\boldsymbol{s}})\boldsymbol{S_1}(\vec{\boldsymbol{s}})\boldsymbol{p} + \boldsymbol{S_2}(\vec{\boldsymbol{s}})\boldsymbol{t} \end{aligned}$$

Homogeneous Coordinates | **Motivation**

More generally, when performing concatenation of transformations of the form $p' = Mp + t$ results in rather messy equation:

$$
\begin{aligned}
p' &= M_2(M_1 p + t_1) + t_2 \\
&= (M_2 M_1)p + M_2 t_1 + t_2
\end{aligned}
$$

Is there a way to combine the transformations into a single matrix, instead of maintaining the multiplicative and translational terms separately?

Yes, by using homogeneous coordinates!

Homogeneous Coordinates | **Introduction**

Homogeneous coordinates extends Cartesian coordinates by one dimension, which is usually called the w coordinate, and setting $w = 1$:

(a) 2D point

Let $p = \begin{bmatrix} x_p \\ y_p \end{bmatrix}$, expressed in Cartesian Coordinate

The homogeneous coordinate of p, $p_h \coloneqq \begin{bmatrix} x_h \\ y_h \\ w \end{bmatrix}$

where

w is a non-zero scalar such that $\begin{bmatrix} x_h/w \\ y_h/w \\ w/w \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}$

Homogeneous Coordinates | **Introduction**

Converting Cartesian Coordinate to Homogeneous Coordinate:

<u>(b) 3D point</u>

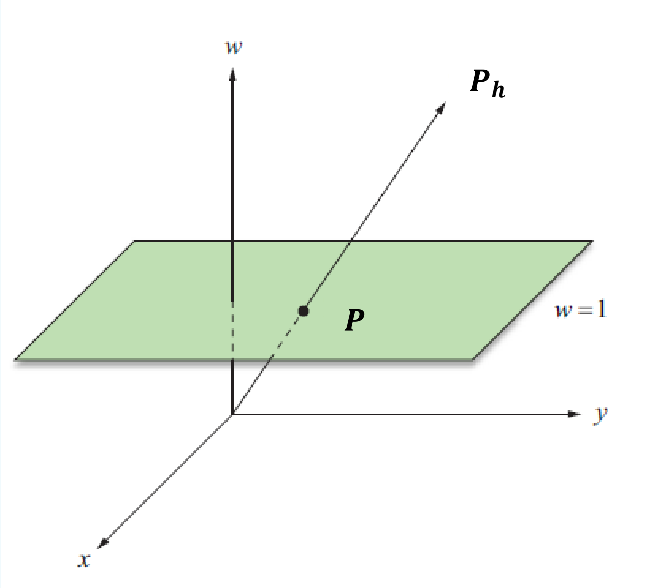
Let $p = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}$, expressed in Cartesian Coordinate

The homogeneous coordinate of p, $p_h \coloneqq \begin{bmatrix} x_h \\ y_h \\ z_h \\ w \end{bmatrix}$ where w is a non-zero scalar such that $\begin{bmatrix} \frac{x_h}{w} \\ \frac{y_h}{w} \\ \frac{z_h}{w} \\ \frac{w}{w} \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$

Homogeneous Coordinates | **Introduction**

Geometrical Interpretation of Homogeneous Coordinates in 2D space

- The Cartesian coordinates P is the projection of the homogeneous coordinates P_h into the 2D space in which $w = 1$
- Thus, any scalar multiple of P_h represents the same point in the 2D space.



Homogeneous Coordinates | **Transformations**

Steps to transform an n-dimensional Cartesian coordinate:

1. Convert it to $(n+1)$-dimensional homogeneous coordinates.
2. Use $(n+1) \times (n+1)$ transformation matrix corresponding to multiplicative term M and translational term T to transform the homogeneous coordinates.

Example: Matrix F combines multiplicative term M and translational term t as follows:

$$F = \begin{bmatrix} M & T \\ \vec{0}^T & 1 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} & t_x \\ M_{21} & M_{22} & M_{23} & t_y \\ M_{31} & M_{32} & M_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Homogeneous Coordinates | **Transformations**

Example 1 (Transforming a 2D point):

$p = \begin{bmatrix} x_p \\ y_p \end{bmatrix}$, Homogeneous coordinate of p, $p_h = \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} p \\ 1 \end{bmatrix}$

$$p_h' = \begin{bmatrix} {x_h}' \\ {y_h}' \\ w \end{bmatrix}$$

$$= Fp_h = \begin{bmatrix} M & t \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} = \begin{bmatrix} Mp + t \\ 1 \end{bmatrix}$$

Homogeneous Coordinates | **Transformations**

Example 2 (Transforming a 3D point):

$p = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}$, Homogeneous coordinate of p, $p_h = \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} p \\ 1 \end{bmatrix}$

$$p'_h = \begin{bmatrix} x'_h \\ y'_h \\ z_h{}' \\ w \end{bmatrix}$$

$$= Fp_h = \begin{bmatrix} M & t \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} = \begin{bmatrix} Mp + t \\ 1 \end{bmatrix}$$

Homogeneous Coordinates | **Transformations**

Basic Transformations for Homogeneous Coordinates:

Translation:

$$T'(t) = \begin{bmatrix} I & t \\ \overrightarrow{0^T} & 1 \end{bmatrix}$$

Rotation:

$$R'_a(\theta) = \begin{bmatrix} R_a(\theta) & \vec{0} \\ \overrightarrow{0^T} & 1 \end{bmatrix}$$

Scaling:

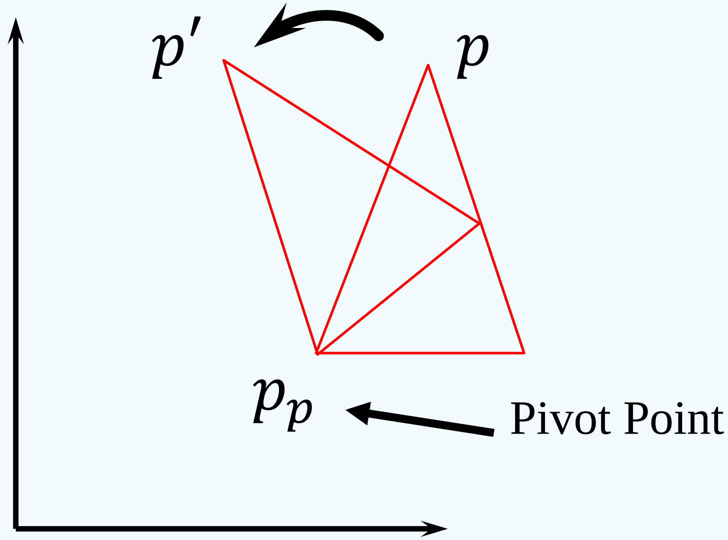
$$S'(\vec{s}) = \begin{bmatrix} S(\vec{s}) & \vec{0} \\ \overrightarrow{0^T} & 1 \end{bmatrix}$$

where:

t is a vector containing translational terms.
$R_a(\theta)$ is a rotation matrix at angle θ about the axis a.
$S(\vec{s})$ is a scaling matrix where $\vec{s}$ is a vector of scaling factors.

Homogeneous Coordinates | **Concatenation of Transformations**

Special Example 1 : Rotation About a Pivot Point $\boldsymbol{p_p}$

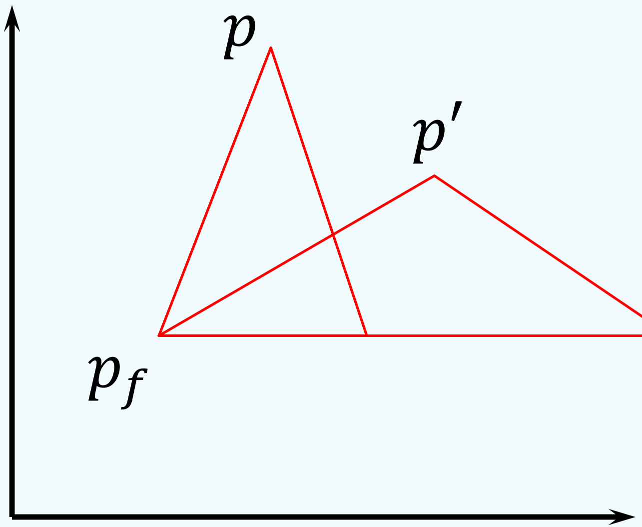


$$p' = T'(p_p)R'_z(\theta)T'(-p_p)p$$
$$= \begin{bmatrix} I & p_p \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} R_z(\theta) & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} I & -p_p \\ \vec{0}^T & 1 \end{bmatrix} p$$

Can concatenate all transformations into a single matrix $F = T'(p_p)R'_z(\theta)T'(-p_p)$

Homogeneous Coordinates | **Concatenation of Transformations**

Special Example 2 : Scaling About a Fixed Point

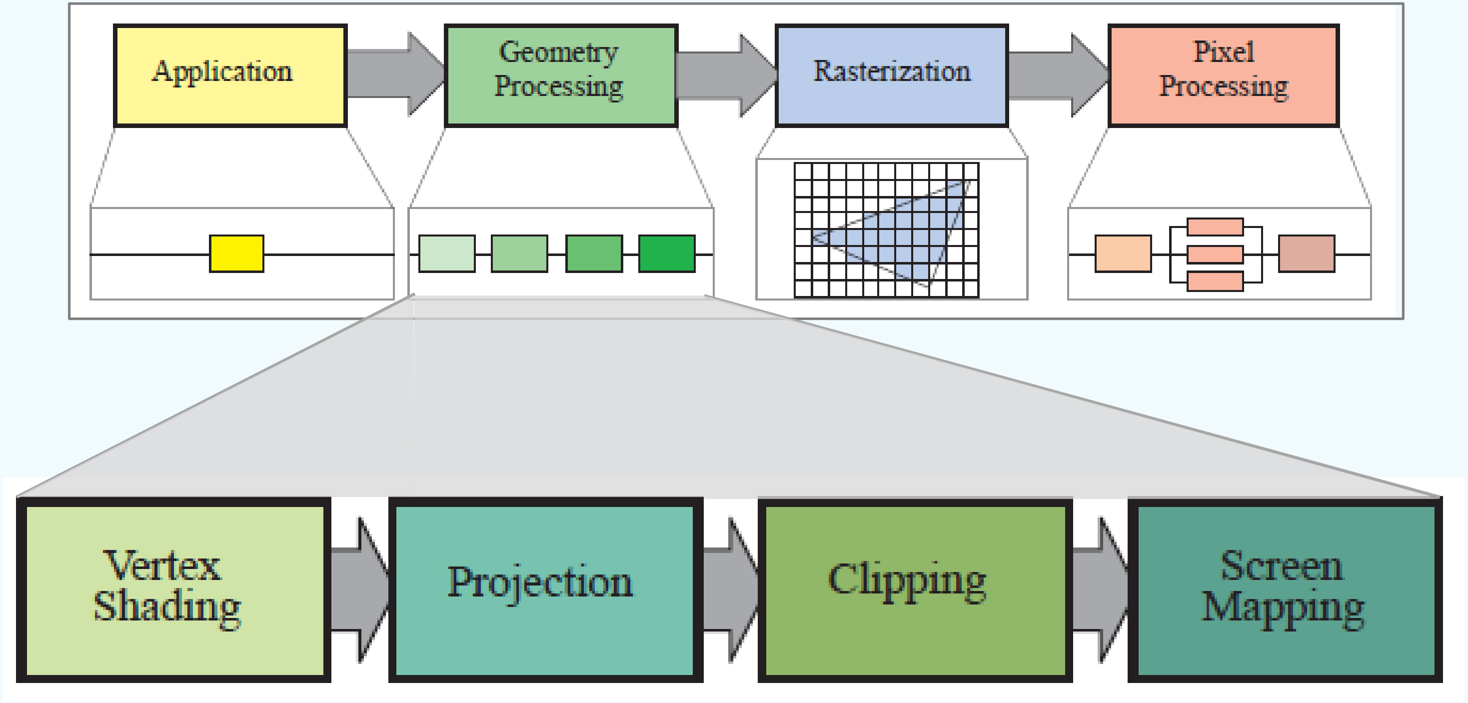


$$p' = T'(p_p)S'(\vec{s})T'(-p_p)p$$
$$= \begin{bmatrix} I & p_p \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} S'(\vec{s}) & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} I & -p_p \\ \vec{0}^T & 1 \end{bmatrix} p$$

Can concatenate all transformations into a single matrix $F = T'(p_p)S'(\vec{s})T'(-p_p)$

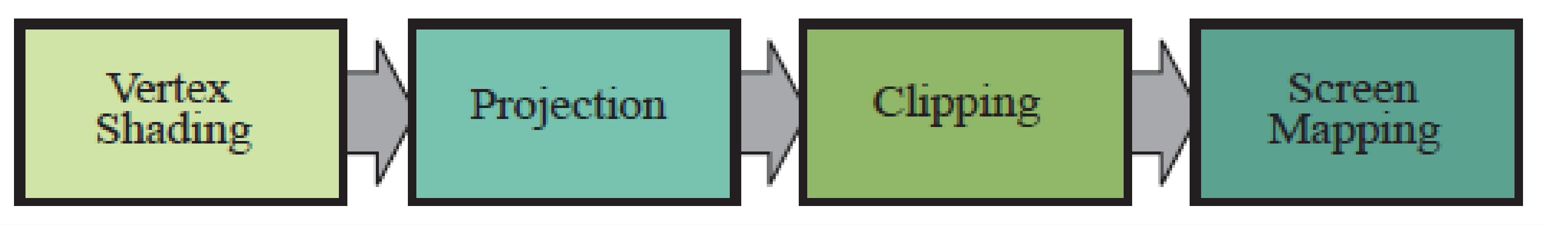
Coordinate Space

Recall that in the Graphics Pipeline ……



Recall that in the Graphics Pipeline ……

Geometry Processing involves a series of coordinate space transformations



Model Space → World Space → View (Camera) Space → Clip Space → Normalized Device Space → Window (Screen) Space

Modeling Transformation	View Transformation	Projection Transformation and Clipping	Perspective Division	Screen Mapping

* We will look at model space, world space, and View (camera) space in this lecture in the coming slides.

Coordinate Space | **Introduction**

Model Space

- The coordinate space in which coordinates of the points of a model are specified.

World Space

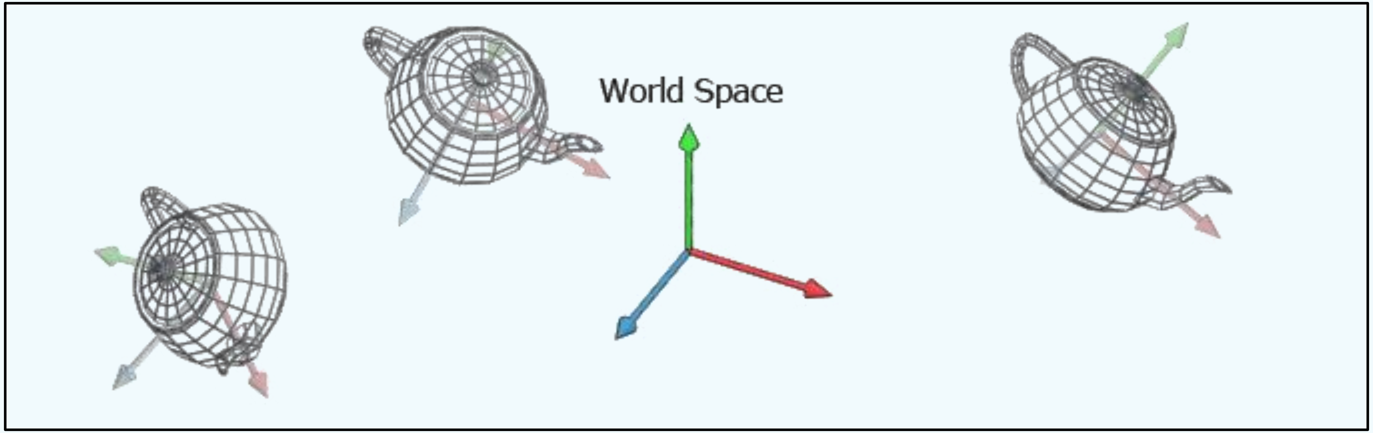
- The coordinate space in which models are placed.

View (Camera) Space

- The coordinate space of which the origin is typically the viewpoint.

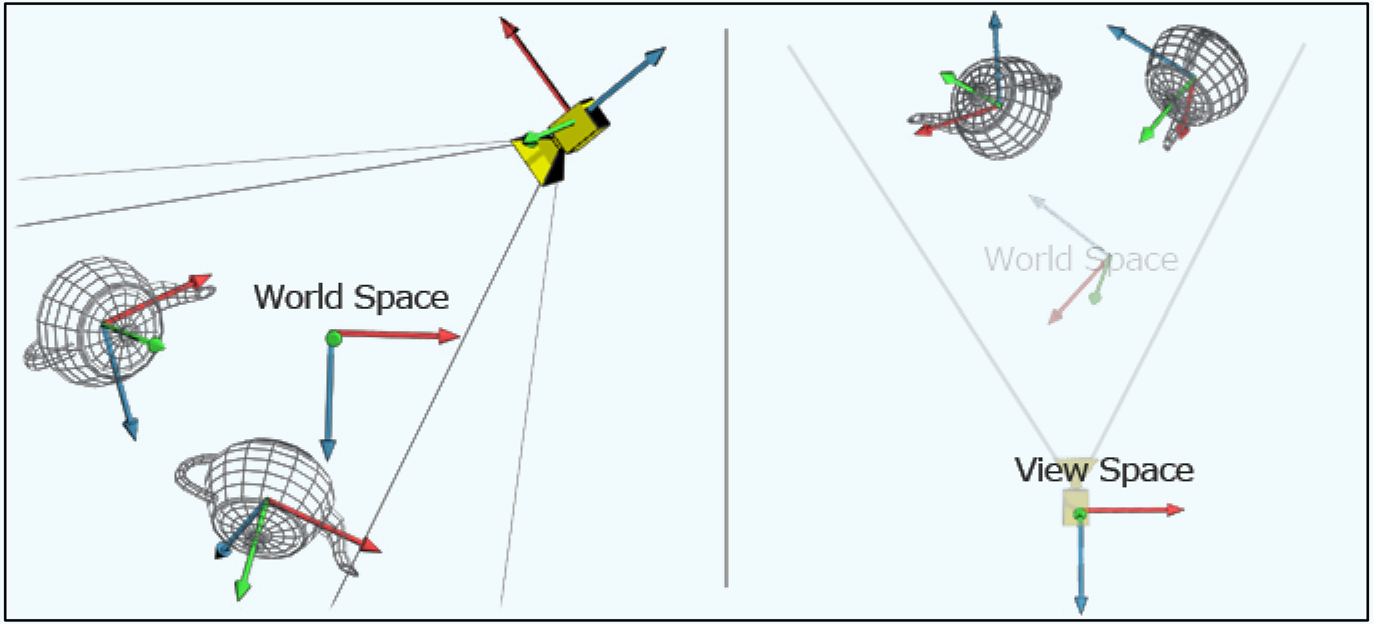
1. There are other coordinate spaces (e.g. Clip space, Screen Space) that will only be covered in detail later.

Coordinate Space | **Introduction**



Three teapots set in World Space

Coordinate Space | **Introduction**

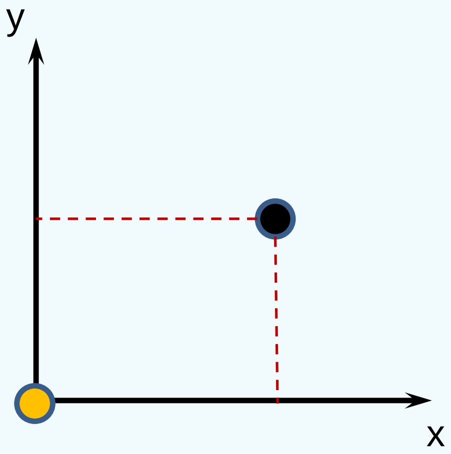


Left : Two teapots and a camera in World Space.
Right : Everything is transformed into View space (World space is represented only to help visualize the transformation).

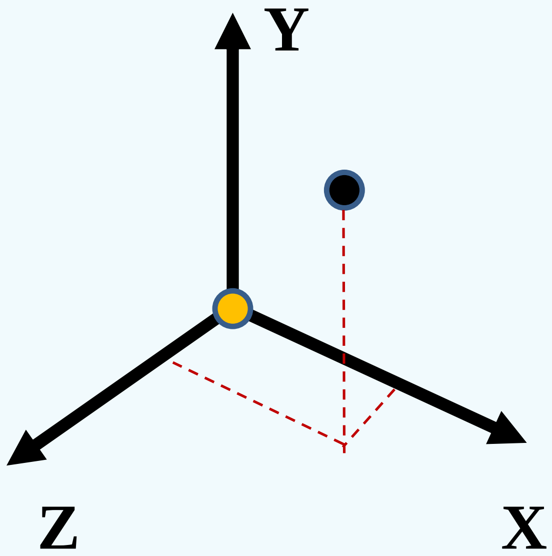
Coordinate Space | **Introduction**

An n-dimensional coordinate space can be represented using $\boldsymbol{n}$ **vectors** and **an origin**. These n vectors must be **linearly independent** and, in most cases, **orthogonal** to each other.

2D



3D



Coordinate Space | **Matrix Representation**

A coordinate space with n vectors and an origin can be expressed in homogeneous coordinate and packed into a single matrix.

2D

$$\begin{bmatrix} X & Y & P \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{3 \times 3}$$

3D

$$\begin{bmatrix} X & Y & Z & P \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

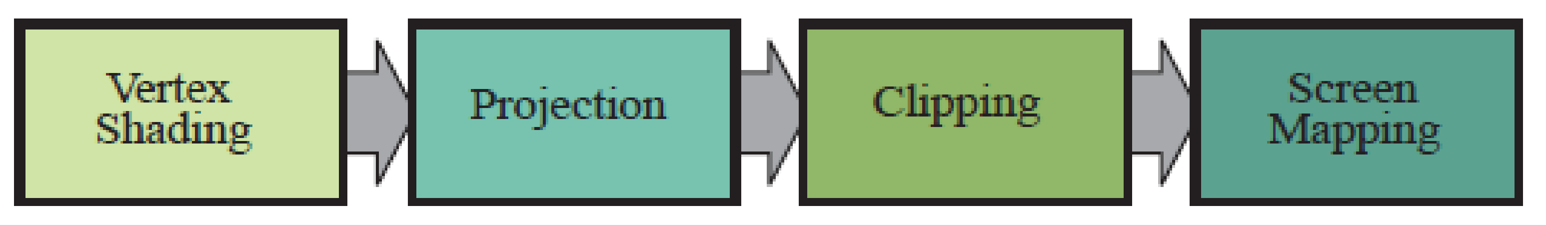
$$\mathbf{4 \times 4}$$

* We shall see why this is helpful in interpreting the result of transformations in the coming slides.

Modelling Transformation

Modelling Transformation

Geometry Processing involves a series of coordinate space transformations



Model Space → World Space → View (Camera) Space → Clip Space → Normalized Device Space → Window (Screen) Space

Modeling Transformation

View Transformation

Projection Transformation and Clipping

Perspective Division

Screen Mapping

Modelling Transformation | Motivation

Consider an example of concatenated transformations:

$$p' = R_z(45°)T\left(\begin{bmatrix}1\\0\end{bmatrix}\right)p$$

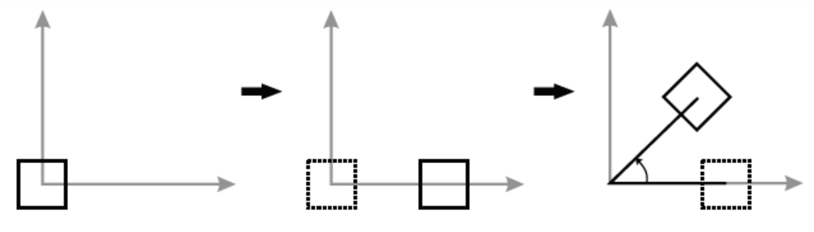
- When the transformations are multiplied **from the right** (i.e. translation then rotation), it **transforms the objects directly in the world space**. No other coordinate space is used.

- When the transformations are multiplied **from the left** (i.e. rotation then translation), it is treated as **transforming the model space** on which the object is created. This is more intuitive and must easier to understand.

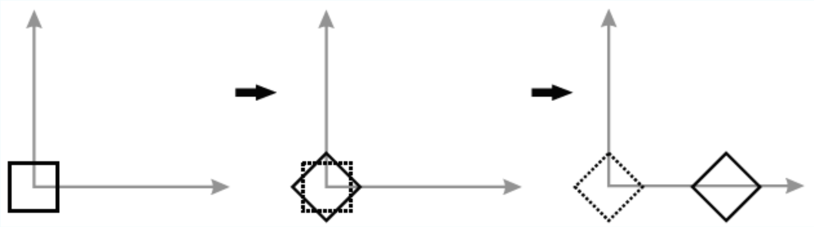
Modelling Transformation | Motivation

$$p' = R_z(45°)T\left(\begin{bmatrix}1\\0\end{bmatrix}\right)p$$

- Translation then rotation.



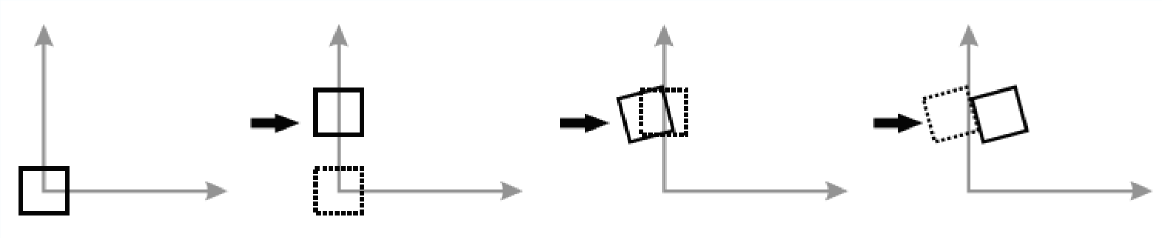
- Rotation then translate on the x-axis.



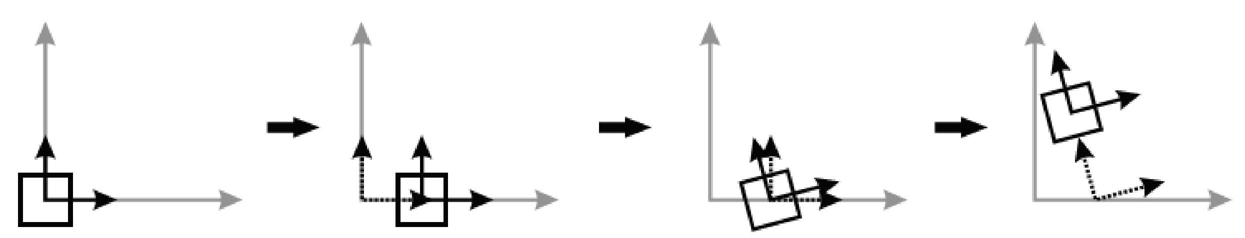
Modelling Transformation | Motivation

$$p' = T(\begin{bmatrix}1\\0\end{bmatrix})R_z(45°)T\left(\begin{bmatrix}0\\1\end{bmatrix}\right)p$$

- Translate on the y-axis, then rotate, then translate on the x-axis.

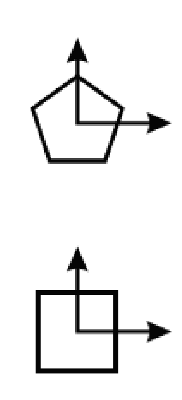


- Translate on the x-axis, then rotate, then translate on the y-axis.



Modelling Transformation | Introduction

- Focus on how to **place** objects in the virtual **world**.
- Objects are specified in their individual coordinate space, called **model space**.



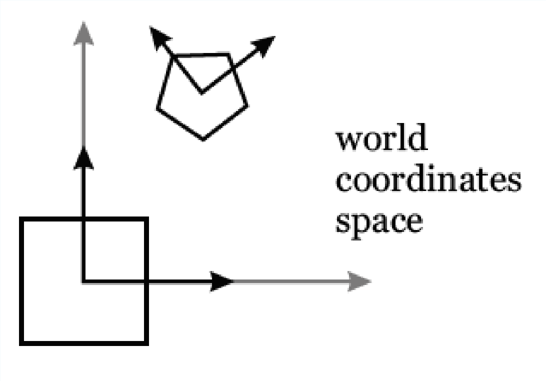
Object 1
Model space

Object 2
Model space

rotation &
translation

MODELLING TRANSFORMATION

scaling



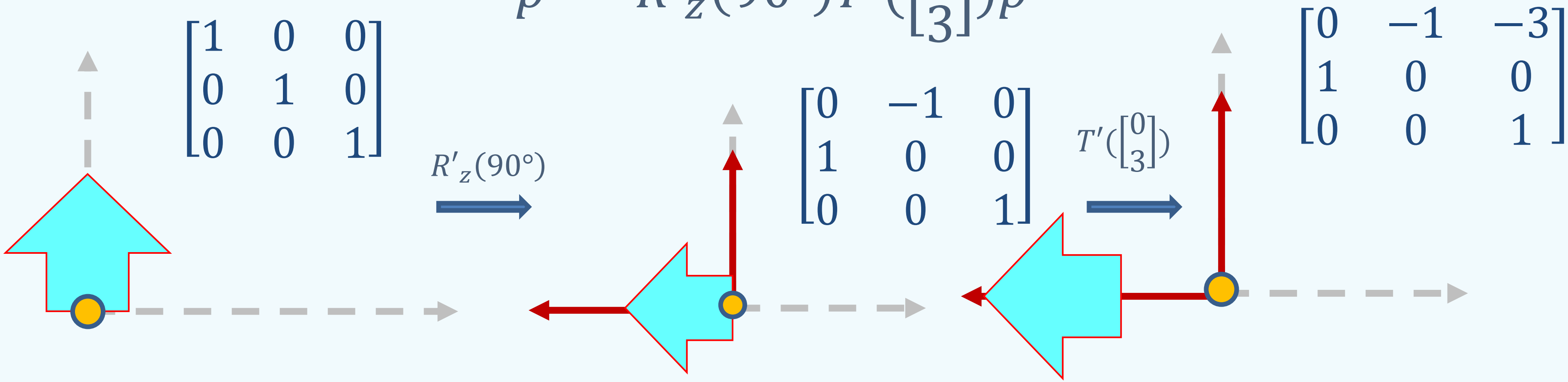
* Notice that both objects' relative position to their own coordinate space doesn't change.

Reference: Interactive Computer Graphics – A Top Down Approach using WebGL (7th Edition)

Modelling Transformation | **Visualization**

We can visualize the result of a concatenated transformation as the resulting coordinate space after the transformation.

$$p' = R'_z(90°)T'(\begin{bmatrix}0\\3\end{bmatrix})p$$



GLM

[Transformation in Practice]

GLM

- An OpenGL Mathematics library designed to match the mathematical functionalities of GLSL and features of the now deprecated OpenGL functions (i.e. gl*() and glu*()).
- Links:
  - Downloads
  - API Documentations
  - Manual

GLM | **Vectors**

$$v_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, v_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, v_3 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

```
#include <glm/glm.hpp>

glm::vec2 v1(1.0f, 2.0f);
std::cout << v1 << std::endl;

glm::vec3 v2(1.0f, 2.0f, 3.0f);
std::cout << v2 << std::endl;

glm::vec4 v3(v2, 1.0f);
std::cout << v3 << std::endl;
```

GLM | **Matrices**

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, t = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$M_1 = \begin{bmatrix} I & t \\ \overrightarrow{0^T} & 1 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} I & -t \\ \overrightarrow{0^T} & 1 \end{bmatrix}$$

$$M_3 = M_1 M_2$$

```
#include <glm/glm.hpp>

glm::mat3 I(1.0f);
glm::vec3 t(1.0f, 2.0f, 3.0f);

glm::mat4 M1(I);
M1[3] = glm::vec4( t, 1.0f);

glm::mat4 M2(I);
M2[3] = glm::vec4(-t, 1.0f);

glm::mat4 M3 = M1 * M2;
```

GLM | Transformations

$$\theta = \frac{\pi}{2}, t = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} s = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix}$$

$$T = \begin{bmatrix} I & t \\ \vec{0}^T & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} R_z(\theta) & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} S(s) & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix}$$

$$F = TRS$$

Method 1:

```
#include <glm/glm.hpp>
#include <glm/gtc/constants.hpp>
#include <glm/gtx/transform.hpp>

constexpr float theta = glm::half_pi<float>();
glm::vec3 t(1.0f, 2.0f, 3.0f);
glm::vec3 s(1.0f, 1.0f, 2.0f);
glm::vec3 z_axis(0.0f, 0.0f, 1.0f);

glm::mat4 T = glm::translate(t);
glm::mat4 R = glm::rotate(theta, z_axis);
glm::mat4 S = glm::scale(s);

glm::mat4 F = T * R * S;
```

GLM | Transformations

$$\theta = \frac{\pi}{2}, t = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} s = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix}$$

$$T = \begin{bmatrix} I & t \\ \vec{0}^T & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} R_z(\theta) & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} S(s) & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix}$$

$$F = TRS$$

Method 2:

```
#include <glm/glm.hpp>
#include <glm/gtc/constants.hpp>
#include <glm/gtc/matrix_transform.hpp>

constexpr float theta = glm::half_pi<float>();
glm::vec3 t(1.0f, 2.0f, 3.0f);
glm::vec3 s(1.0f, 1.0f, 2.0f);
glm::vec3 z_axis(0.0f, 0.0f, 1.0f);

glm::mat4 F(1.0f);
F = glm::translate(F, t);
F = glm::rotate(F, theta, z_axis);
F = glm::scale(F, s);
```

Q & A

Acknowledgement

- This presentation has been designed using resources from PoweredTemplate.com