Week 7 - Mathematics

Transformations and Coordinate Systems

Recap

- Up to this point:
  - Create object
  - Colour the object
  - Sample textures and wrap around polygon
- What we need to make things interesting – move vertices around.
- Cumbersome if we have to change each individual vertex and reconfigure their buffers.
- Hence we use transformations via vectors and matrices

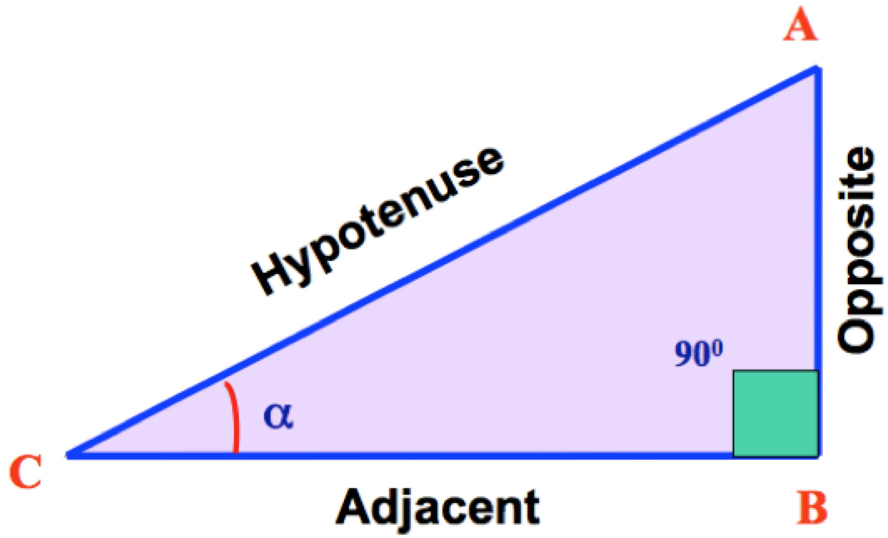
Agenda

- Transformations
  - Vectors
  - Matrices
  - Multiplication
  - Translation and Rotation
- Coordinate Systems
  - Local, world, view and clip space
  - Orthographic and Perspective Projection
  - Going 3D

Basics

Trigonometry, Vectors, Matrices and Multiplication

Trigonometry



$$\sin\alpha = \frac{AB}{CA} = \frac{opposite}{hypotenuse}$$

$$\cos\alpha = \frac{CB}{CA} = \frac{adjacent}{hypotenuse}$$

$$tan\,\alpha = \frac{AB}{CB} = \frac{opposite}{adjacent}$$

Transformations

Translation and Rotation

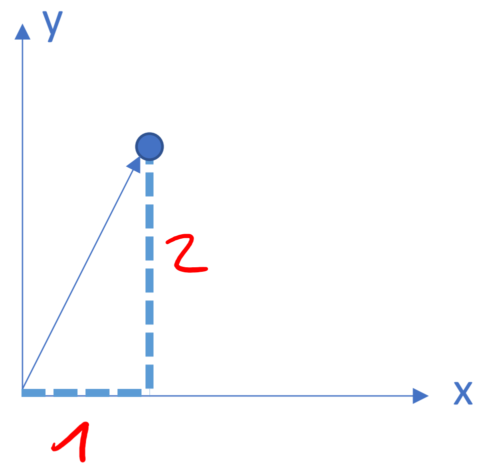
What we still need to know:
Coordinates for Points -> Lines -> Triangles -> Meshes

Introducing

- Vectors and Operations
- Homogeneous Coordinates

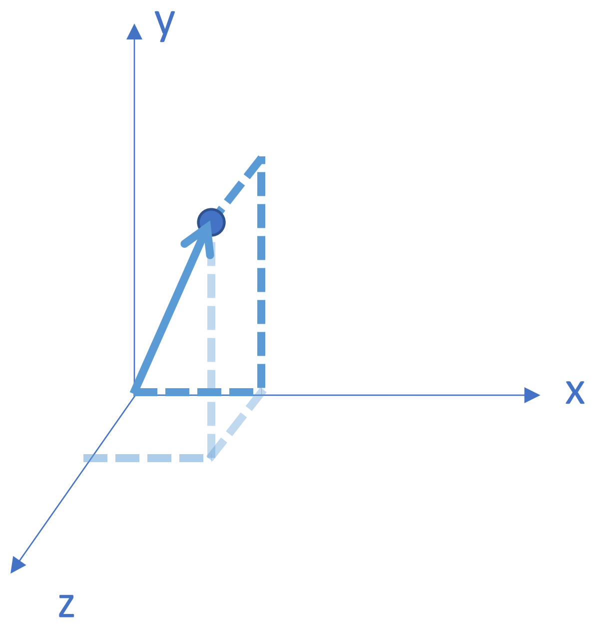
Vectors

$$p = \begin{bmatrix} 1 \\ 2 \end{bmatrix} = [1\ 2\ ]^T = 1i + 2j$$



Vectors

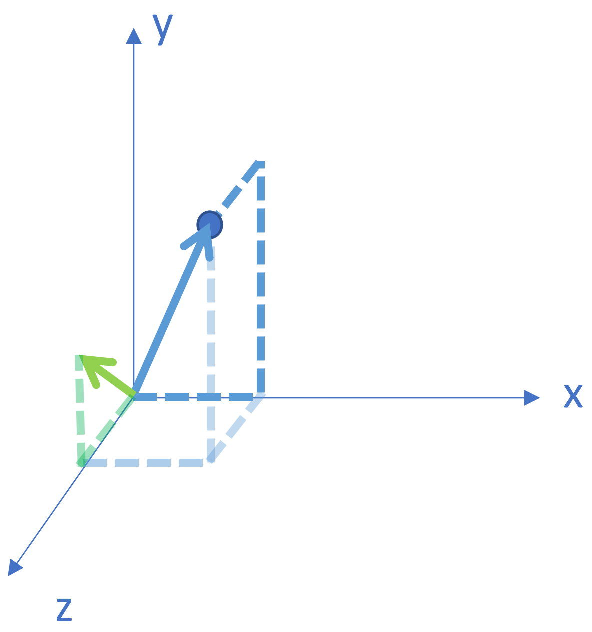
$$p=\begin{bmatrix}1\\2\\1\end{bmatrix}=[1\ 2\ 1]^T=1i+2j+1k$$



Vectors

$$p=\begin{bmatrix}1\\2\\1\end{bmatrix}= [1\ 2\ 1]^T$$

$$V_1=\begin{bmatrix}0\\1\\1\end{bmatrix}= [0\ 1\ 1]^T$$

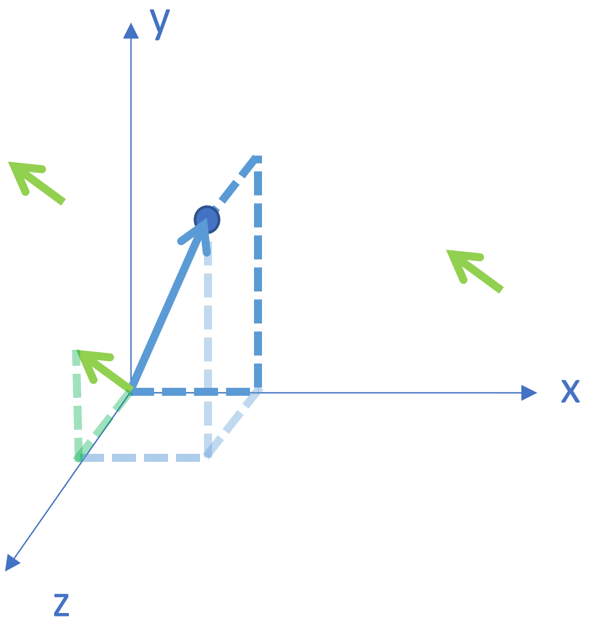


What is the difference between a vector and a point?

Vectors

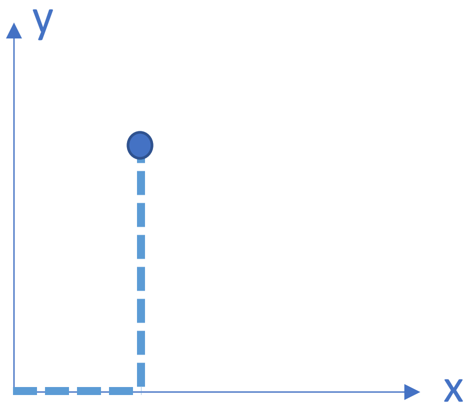
$$p=\begin{bmatrix}1\\2\\1\end{bmatrix}=[1\ 2\ 1]^T$$

$$V_1=\begin{bmatrix}0\\1\\1\end{bmatrix}=[0\ 1\ 1]^T$$



What is the difference between a vector and a point?

Going back to 2D

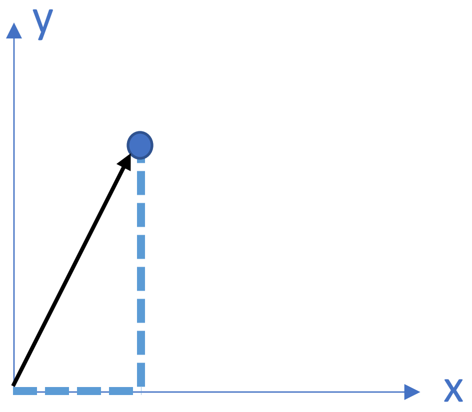


Operations in Space

$V_p = \begin{bmatrix} 1 \\ 2 \end{bmatrix} = [1\ 2\ ]^T = 1i + 2k$

$0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = [0\ 0\ ]^T = 0i + 0k$

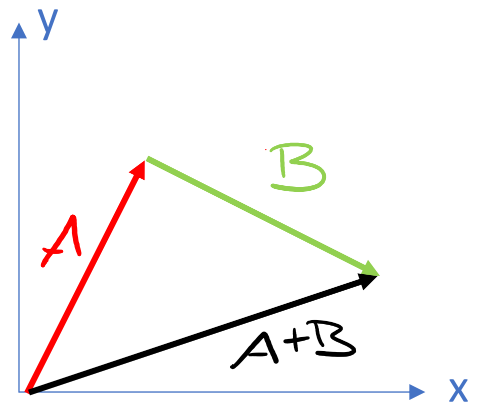
p=Op



Operations in Space

$A = \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 1i + 2k$

$B = \begin{bmatrix} 2 \\ -1 \end{bmatrix} = 2i - 1k$



Vector addition:

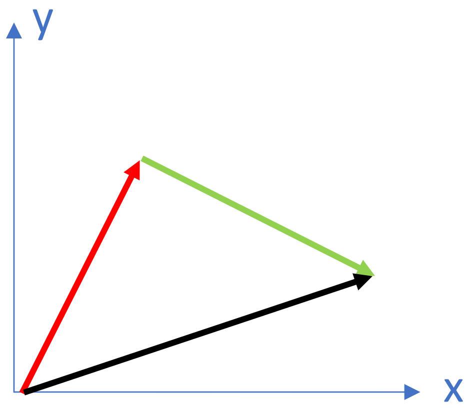
$A+B = B+A = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 1+2 \\ 2-1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$= (1i+2k)+(2i-1k) = 3i + 1k$

Operations in Space

$\color{red}{A}$ $=\begin{bmatrix}1\\2\end{bmatrix}$=1i+2k

$\color{green}{B}$ $=\begin{bmatrix}2\\-1\end{bmatrix}$=2i-1k



Vector addition:

A+B = B+A =$\begin{bmatrix}1\\2\end{bmatrix}$+$\begin{bmatrix}2\\-1\end{bmatrix}$ = $\begin{bmatrix}1+2\\2-1\end{bmatrix}$= $\begin{bmatrix}3\\1\end{bmatrix}$

= (1i+2k)+(2i-1k) = 3i + 1k

Vector Magnitude: |V|

|A|= $\sqrt{{a_x}^2 + ay^2}$

= $\sqrt{1^2 + 2^2}$ = $\sqrt{5}$

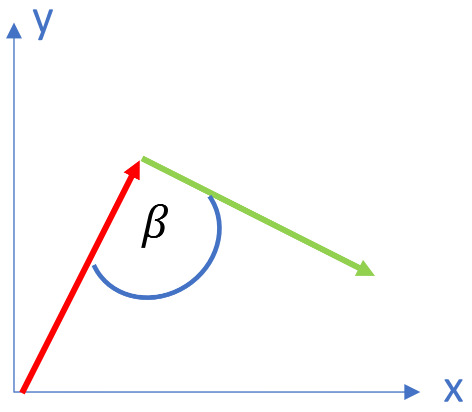
|B|= $\sqrt{{b_x}^2 + {b_y}^2}$

= $\sqrt{2^2 + (-1)^2}$ = $\sqrt{5}$

Operations in Space

$A = \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 1i+2k$

$B = \begin{bmatrix} 2 \\ -1 \end{bmatrix} = 2i-1k$



Scalar Product: A·B

$A \cdot B = |A|*|B|*\cos(\beta)$

$\cos(\beta) = \dfrac{A \cdot B}{|A|*|B|}$

Vector addition:

$A+B = B+A = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 1+2 \\ 2-1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$= (1i+2k)+(2i-1k) = 3i + 1k$

Vector Magnitude: |V|

$|A| = \sqrt{a_x{}^2 + ay^2}$

$= \sqrt{1^2 + 2^2} = \sqrt{5}$

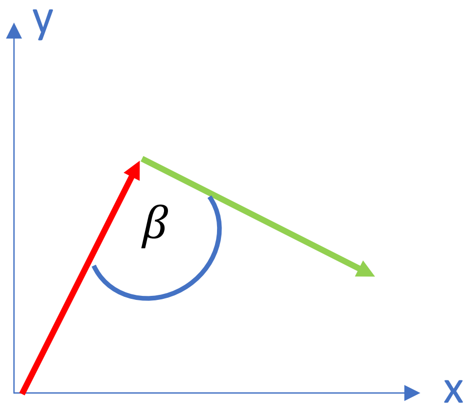
$|B| = \sqrt{b_x{}^2 + b_y{}^2}$

$= \sqrt{2^2 + (-1)^2} = \sqrt{5}$

Operations in Space

$A = \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 1i + 2k$

$B = \begin{bmatrix} 2 \\ -1 \end{bmatrix} = 2i - 1k$



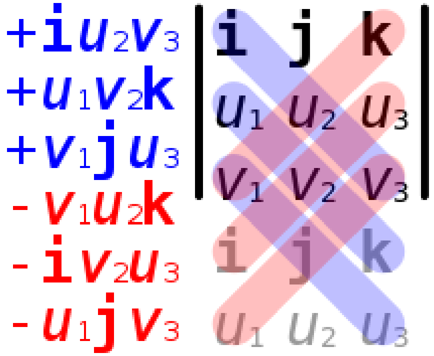
Scalar Product: A·B

$A \cdot B = |A| * |B| * \cos(\beta)$

$\cos(\beta) = \dfrac{A \cdot B}{|A| * |B|}$

Cross Product: A×B

$A \times B = (a_y b_z - a_z b_y)i +$
$(a_z b_x - a_x b_z)j +$
$(a_x b_y - a_y b_x)k$



Vector addition:

$A+B = B+A = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 1+2 \\ 2-1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$= (1i+2k)+(2i-1k) = 3i + 1k$

Vector Magnitude: |V|

$|A| = \sqrt{a_x{}^2 + ay^2}$

$= \sqrt{1^2 + 2^2} = \sqrt{5}$

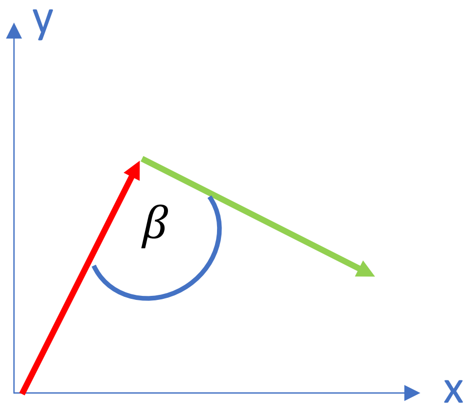
$|B| = \sqrt{b_x{}^2 + b_y{}^2}$

$= \sqrt{2^2 + (-1)^2} = \sqrt{5}$

Operations in Space

$A = \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 1i + 2k$

$B = \begin{bmatrix} 2 \\ -1 \end{bmatrix} = 2i - 1k$



Scalar Product: A·B

$A \cdot B = |A| * |B| * \cos(\beta)$

$\cos(\beta) = \frac{A \cdot B}{|A| * |B|}$

Vector addition:

$A+B = B+A = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 1+2 \\ 2-1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$= (1i+2k)+(2i-1k) = 3i + 1k$

Vector Magnitude: |V|

$|A| = \sqrt{a_x{}^2 + ay^2}$

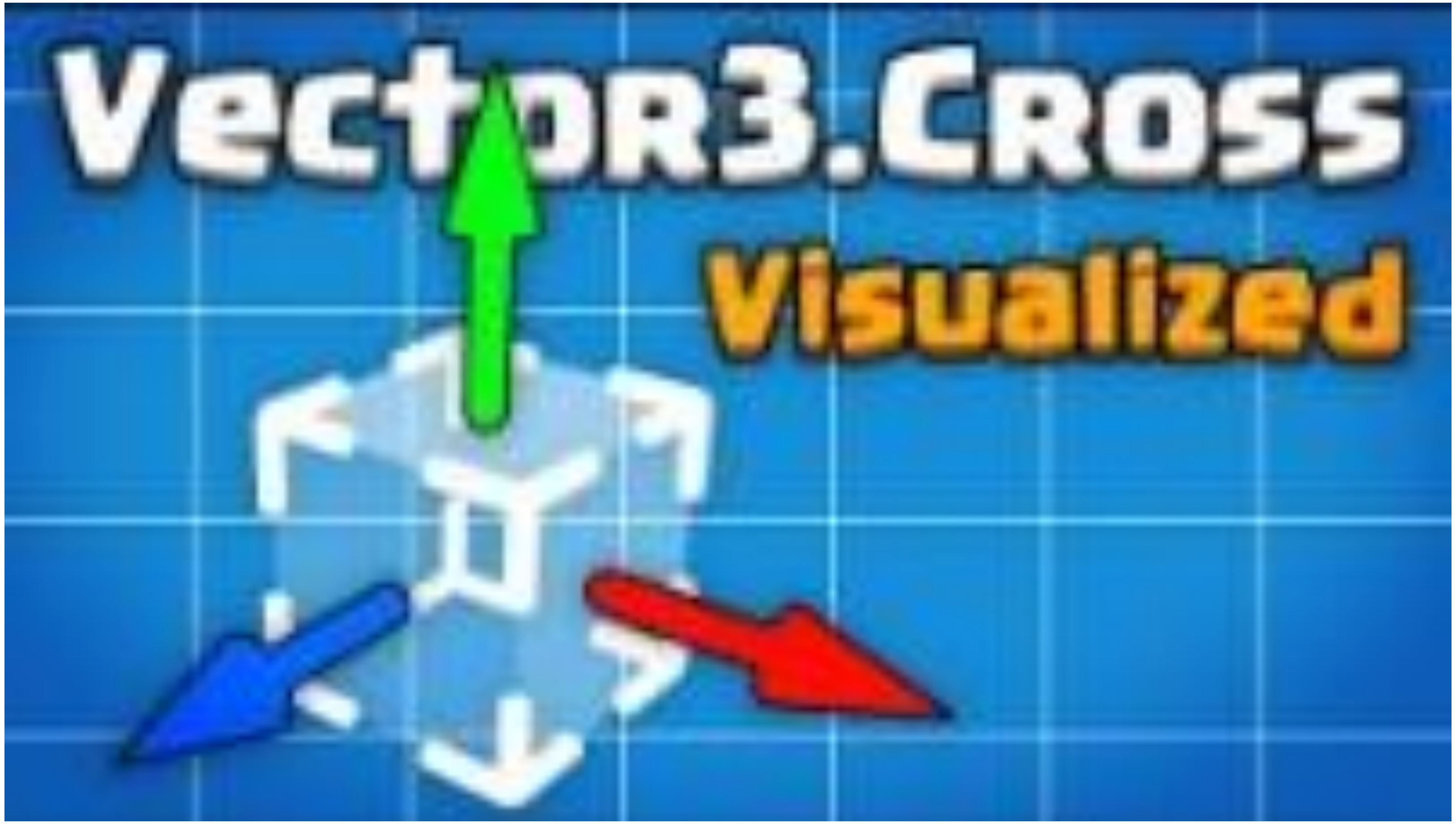
$= \sqrt{1^2 + 2^2} = \sqrt{5}$

$|B| = \sqrt{b_x{}^2 + b_y{}^2}$

$= \sqrt{2^2 + (-1)^2} = \sqrt{5}$

Add Another Dimension for 3D & Another one for Homogenous Coordinates

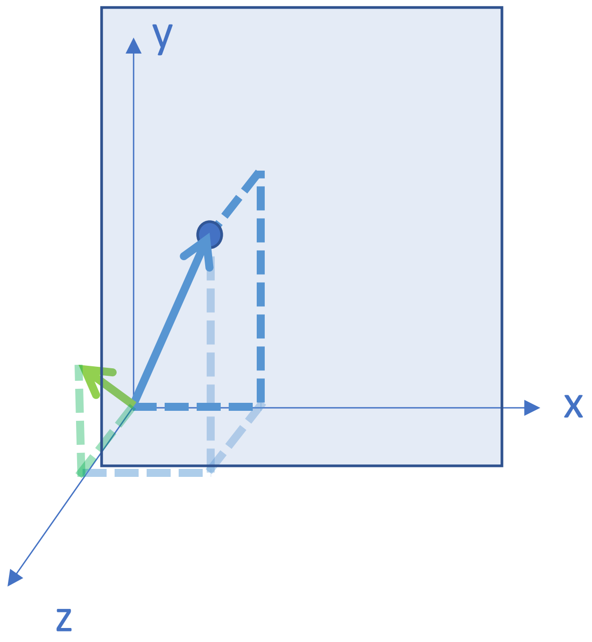
Cross product



Why Homogeneous Coordinates?

$$p=\begin{bmatrix}1\\2\\1\end{bmatrix}=[1\ 2\ 1]^T$$

$$V_1=\begin{bmatrix}0\\1\\0\end{bmatrix}=[0\ 1\ 0]^T$$

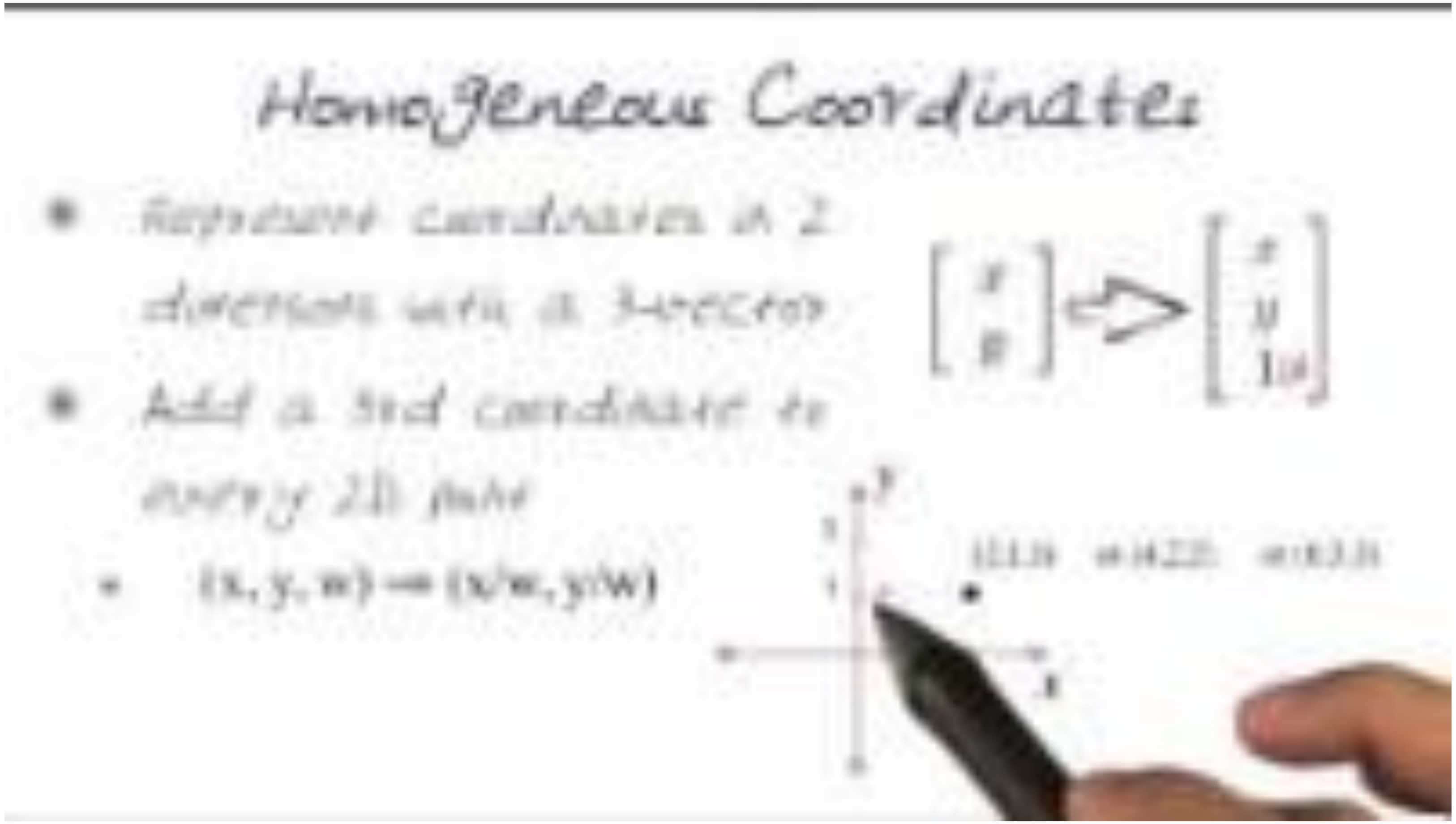


Why Homogeneous Coordinates?

Homogeneous coordinates are key to all computer graphics systems

- All standard transformations (rotation, translation, scaling) can be implemented with matrix multiplications using 4 x 4 matrices
- Hardware pipeline works with 4 dimensional representations
- For orthographic viewing, we can maintain $w=0$ for vectors and $w=1$ for points

Homogeneous Coordinates



Matrix Recap

- Addition, substraction and Scalar multiplication

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 2 \\ 1 & 6 \end{bmatrix} - \begin{bmatrix} 2 & 4 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 4-2 & 2-4 \\ 1-0 & 6-1 \end{bmatrix} = \begin{bmatrix} 2 & -2 \\ 1 & 5 \end{bmatrix}$$

$$2 \cdot \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 \cdot 1 & 2 \cdot 2 \\ 2 \cdot 3 & 2 \cdot 4 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

Matrix Multiplication

- Only possible if left side matrix has same number of columns as the rows of the right matrix.
- Not commutative! A.B is not same as B.A

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1 \cdot 5 + 2 \cdot 7 & 1 \cdot 6 + 2 \cdot 8 \\ 3 \cdot 5 + 4 \cdot 7 & 3 \cdot 6 + 4 \cdot 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 2 & 0 \\ 0 & 8 & 1 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 4 & 2 & 1 \\ 2 & 0 & 4 \\ 9 & 4 & 2 \end{bmatrix} = \begin{bmatrix} 4 \cdot 4 + 2 \cdot 2 + 0 \cdot 9 & 4 \cdot 2 + 2 \cdot 0 + 0 \cdot 4 & 4 \cdot 1 + 2 \cdot 4 + 0 \cdot 2 \\ 0 \cdot 4 + 8 \cdot 2 + 1 \cdot 9 & 0 \cdot 2 + 8 \cdot 0 + 1 \cdot 4 & 0 \cdot 1 + 8 \cdot 4 + 1 \cdot 2 \\ 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 9 & 0 \cdot 2 + 1 \cdot 0 + 0 \cdot 4 & 0 \cdot 1 + 1 \cdot 4 + 0 \cdot 2 \end{bmatrix}$$

$$= \begin{bmatrix} 20 & 8 & 12 \\ 25 & 4 & 34 \\ 2 & 0 & 4 \end{bmatrix}$$

Matrix-Vector Multiplication Use Cases

- Identity Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 \\ 1 \cdot 2 \\ 1 \cdot 3 \\ 1 \cdot 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

- Scaling

$$\begin{bmatrix} S_1 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 \\ 0 & 0 & S_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} S_1 \cdot x \\ S_2 \cdot y \\ S_3 \cdot z \\ 1 \end{pmatrix}$$

- Translation

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + T_x \\ y + T_y \\ z + T_z \\ 1 \end{pmatrix}$$

Rotation

- X-axis

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ \cos\theta \cdot y - \sin\theta \cdot z \\ \sin\theta \cdot y + \cos\theta \cdot z \\ 1 \end{pmatrix}$$

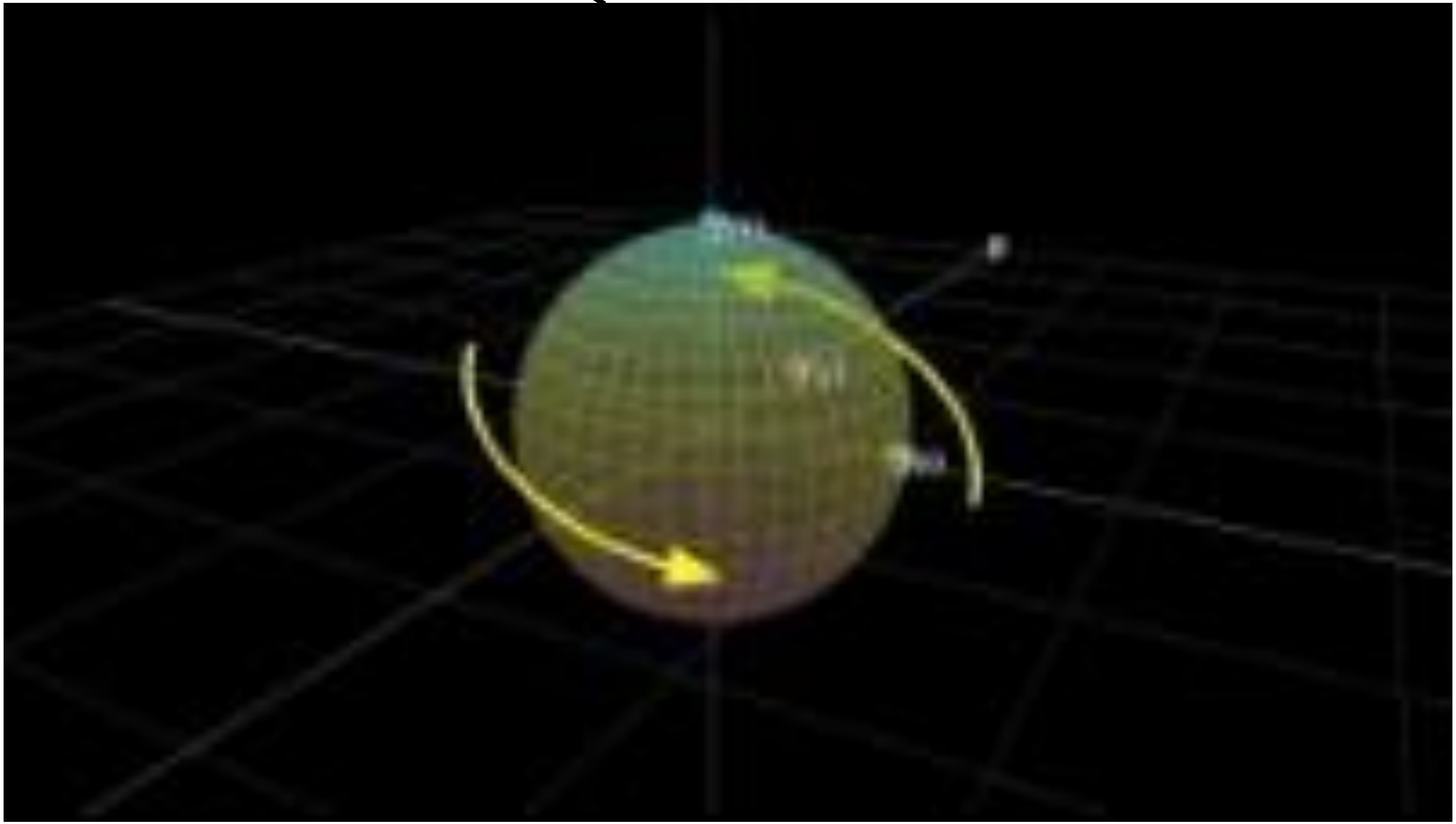
- Y-axis

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot x + \sin\theta \cdot z \\ y \\ -\sin\theta \cdot x + \cos\theta \cdot z \\ 1 \end{pmatrix}$$

- Z-axis

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot x - \sin\theta \cdot y \\ \sin\theta \cdot x + \cos\theta \cdot y \\ z \\ 1 \end{pmatrix}$$

Gimbal Lock and Quarternions



Combining Transforms

- Transform then scale:

$$Trans.Scale = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 2 & 0 & 2 \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 2 & 0 & 2 \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 2x + 1 \\ 2y + 2 \\ 2z + 3 \\ 1 \end{bmatrix}$$

In Practice: Use GLM

Declare and define transform matrix.

Use methods like translate, rotate and scale with the proper params.

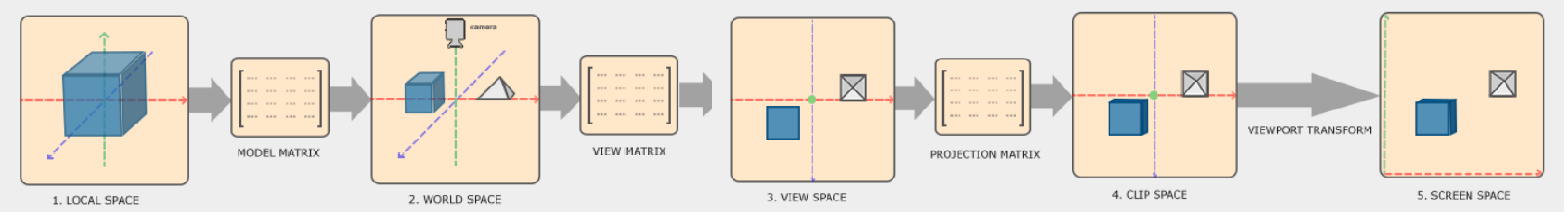
```
glm::mat4 trans = glm::mat4(1.0f);
trans = glm::rotate(trans, glm::radians(90.0f), glm::vec3(0.0, 0.0, 1.0));
trans = glm::scale(trans, glm::vec3(0.5, 0.5, 0.5));
```

For shaders, use a uniform mat4 variable and pass the transformation matrix to the shader:

```
unsigned int transformLoc = glGetUniformLocation(ourShader.ID, "transform");
glUniformMatrix4fv(transformLoc, 1, GL_FALSE, glm::value_ptr(trans));
```

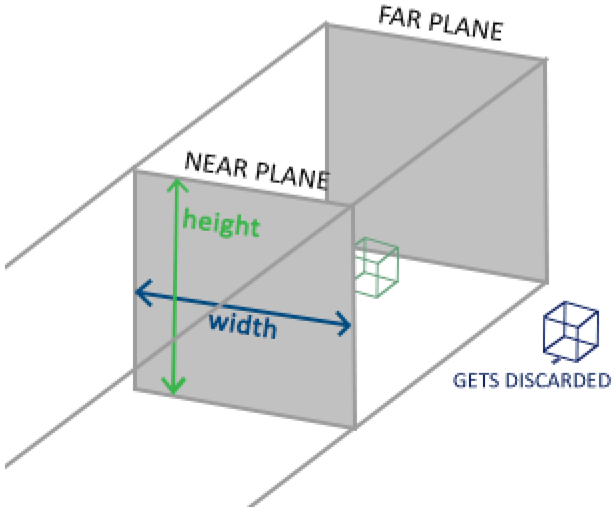
Coordinate systems

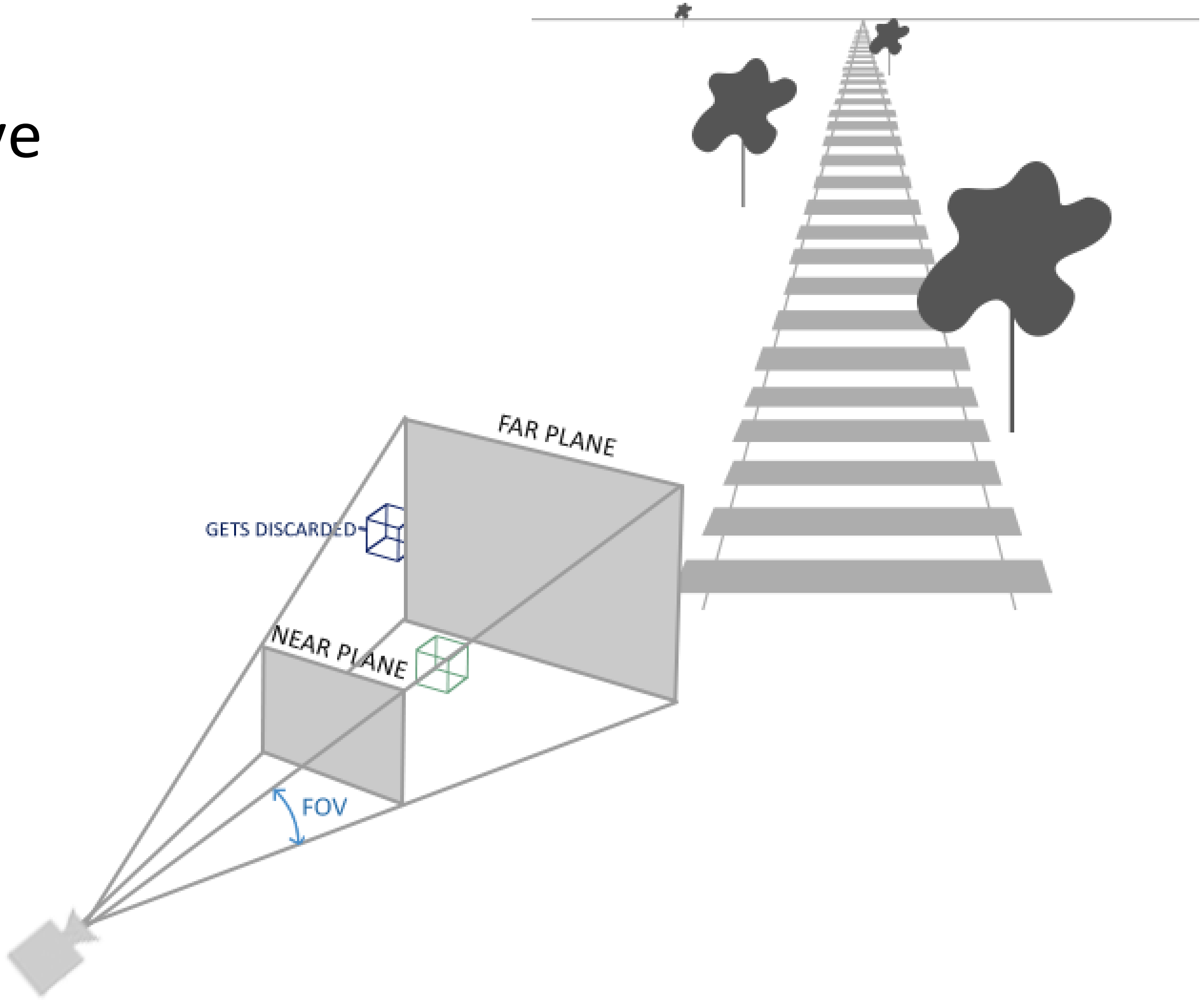
- Local coordinates - the coordinates of your object relative to its local origin
- World-space coordinates - coordinates relative to some global origin of the world, together with other objects in scene relative to this world's origin.
- View-space coordinates – how each coordinate is as seen from the camera or viewer POV.
- Clip space coordinates - processed to the -1.0 and 1.0 range and determine which vertices will end up on the screen. Adds perspective if using perspective projection.
- Viewport transform that transforms the coordinates from -1.0 and 1.0 to the coordinate range defined by glViewport. The resulting coordinates are sent to the rasterizer to turn into fragments.



Orthographic vs Perspective Projection

- Glm::ortho and glm::perspective





Orthographic vs Perspective

Combining the Transforms

- Create the matrix for each of the steps: model, view and then projection.

$$V_{clip} = M_{projection} \cdot M_{view} \cdot M_{model} \cdot V_{local}$$

- Note that order of multiplication is reversed.
- Result is assigned to gl_Position in the vertex shader.
- OpenGL will do perspective division and clipping.