INFOGR – Computer Graphics

J. Bikker - April-July 2015 - Lecture 5: “3D Engine Fundamentals”

Welcome!

Today's Agenda:

- Rendering Overview
- Matrices
- Transforms

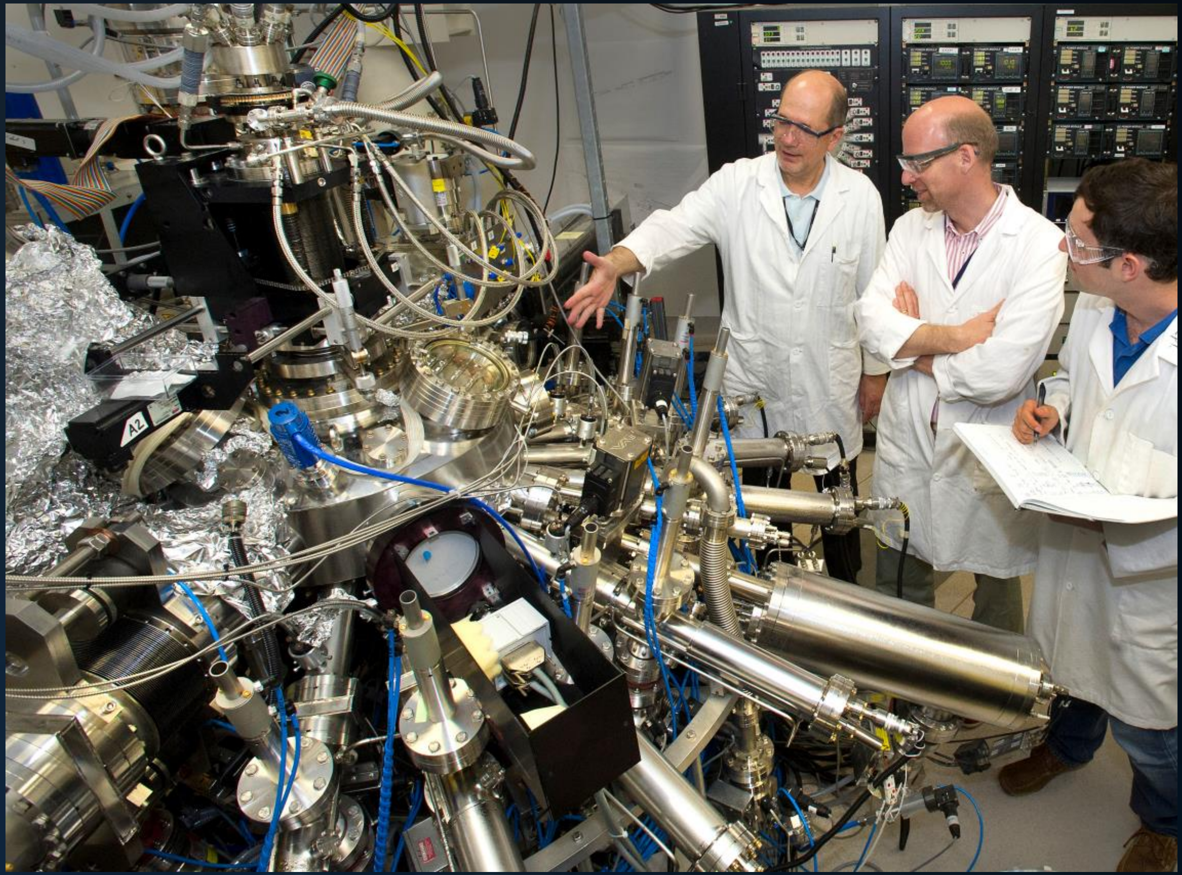
Rendering

Topics covered so far:

Lecture 1:
- Field study

Lecture 2:
- Rasters
- Vectors
- Color representation

Lecture 3:
- 2D primitives
- 3D primitives
- Textures



Rendering

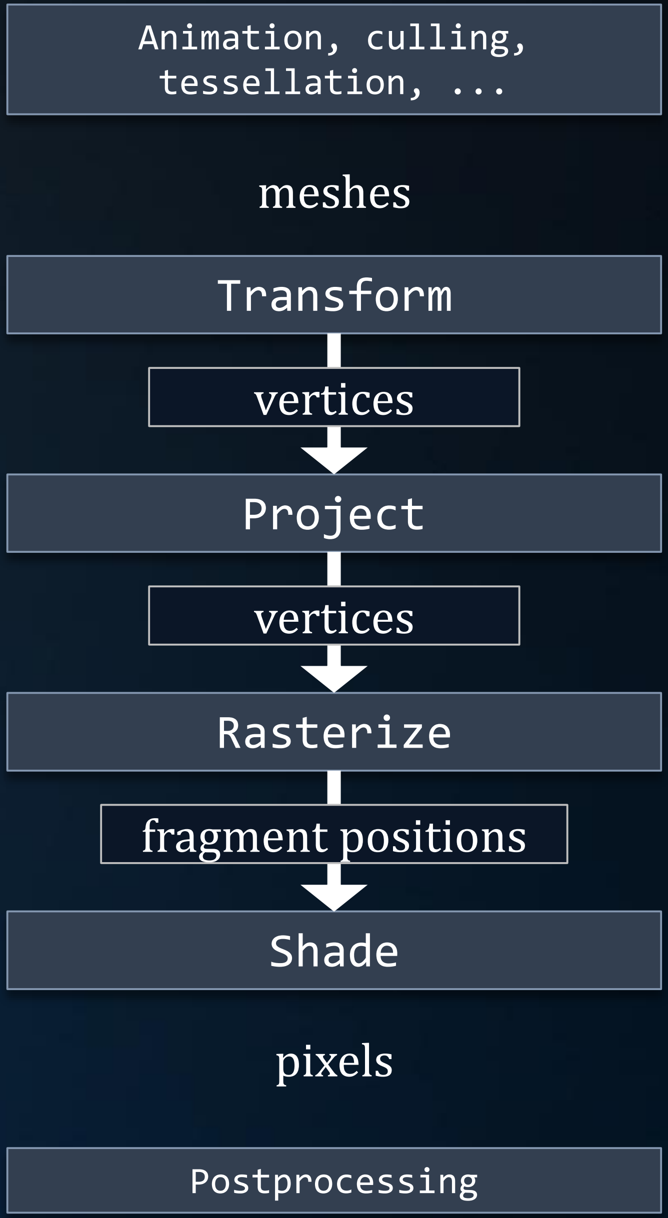
Rendering – Functional overview

1. Transform:
   translating / rotating / scaling meshes

2. Project:
   calculating 2D screen positions

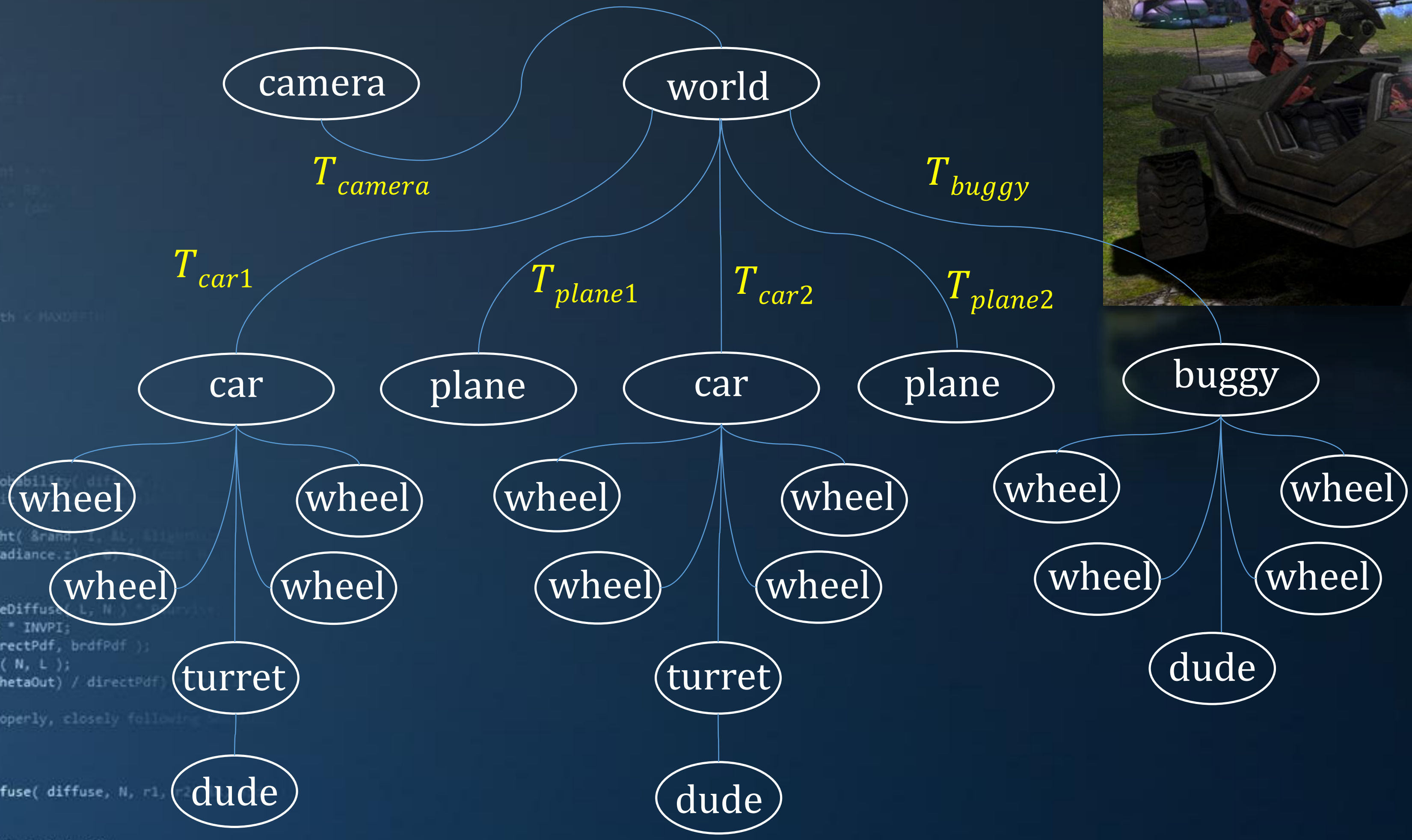
3. Rasterize:
   determining affected pixels

4. Shade:
   calculate color per affected pixel



Rendering

Rendering – Data overview



Rendering

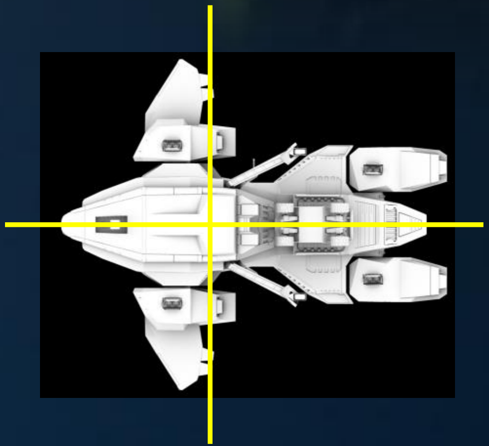
Rendering – Data overview

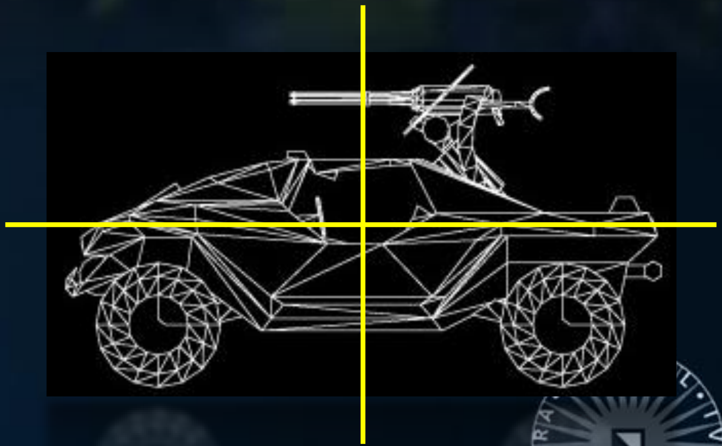
Objects are organized in a hierarchy: the *scenegraph*.

In this hierarchy, objects have translations and orientations relative to their parent node.

Relative translations and orientations are specified using matrices.

Mesh vertices are defined in a coordinate system known as *object space*.



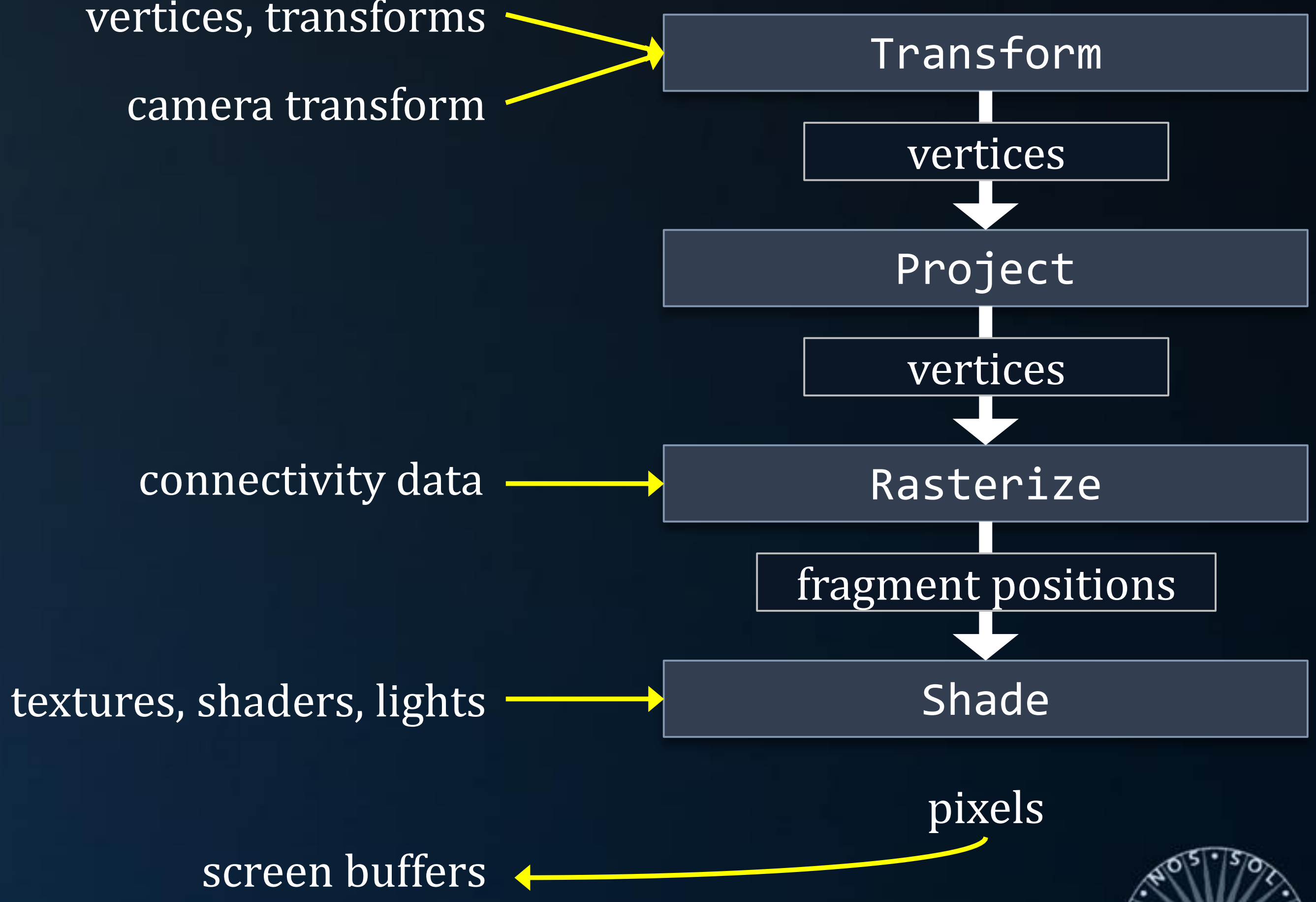


Rendering

Rendering – Data overview

Transform takes our meshes from object space (3D) to camera space (3D).

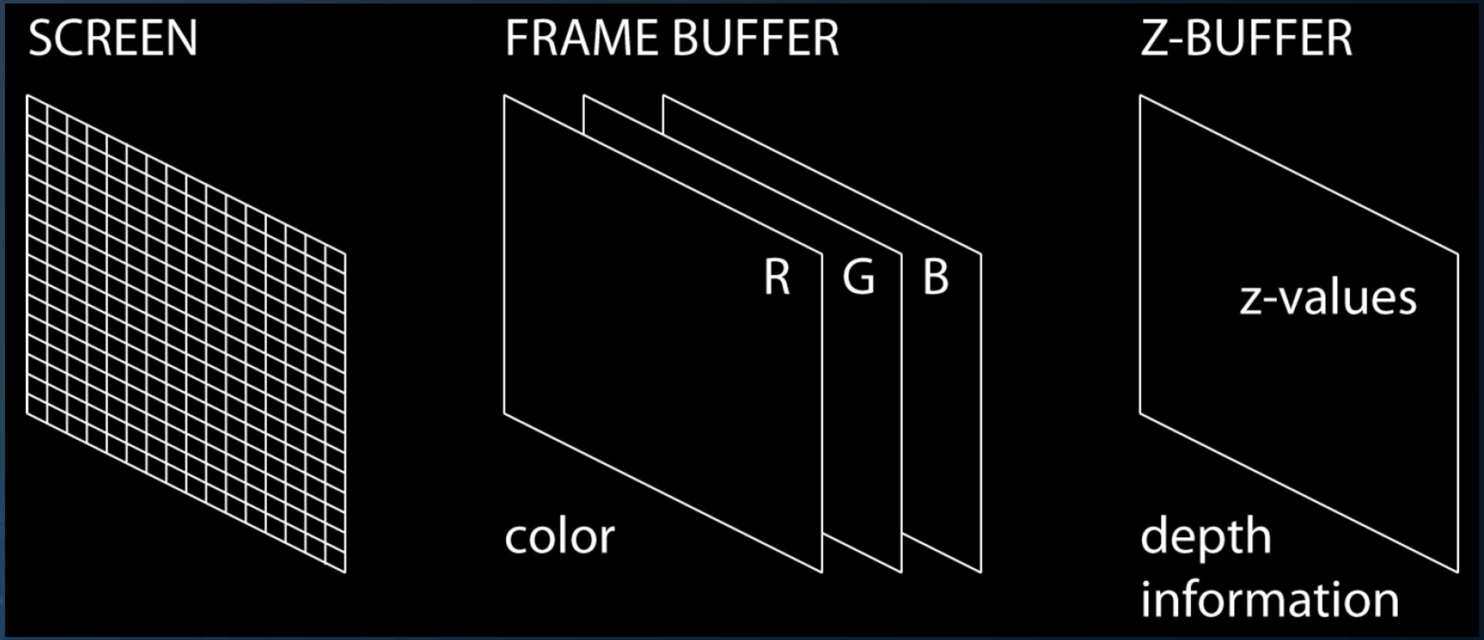
Project takes the vertex data from camera space (3D) to screen space (2D).



Rendering

Rendering – Data overview

The screen is represented by (at least) two buffers:



Rendering

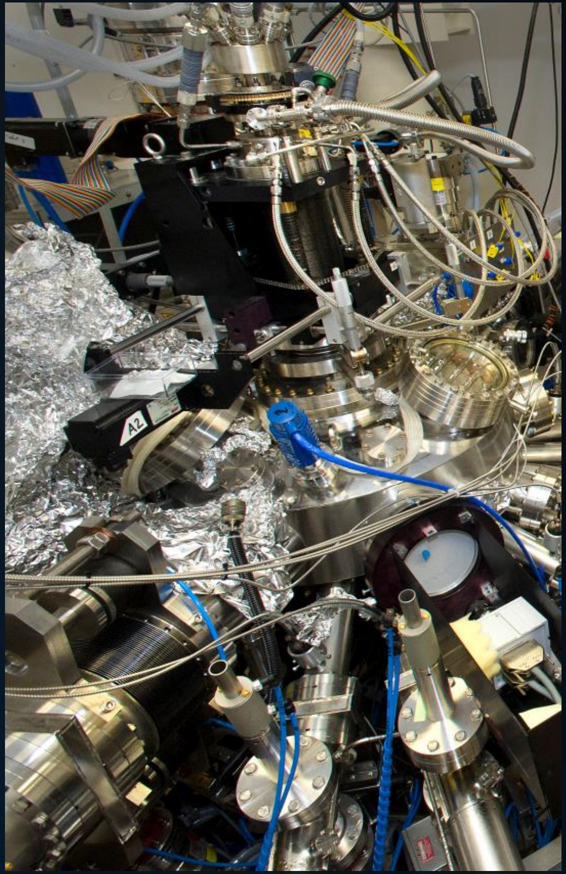
Rendering – Components

Scenegraph
- Culling — *Lecture 7*

Vertex transform pipeline
- Matrices to convert from one space to another — *Lecture 5*
- Perspective — *Lecture 6*

Rasterization
- Interpolation
- Clipping — *Lecture 7*
- Depth sorting: z-buffer — *Lecture 7*

Shading
- Light / material interaction — *P2*
- Shadows / reflections / etc. — *P3*



Today’s Agenda:

- Rendering Overview
- Matrices
- Transforms

Matrices

Bases in $\mathbb{R}^2$ and $\mathbb{R}^3$

Recall:

- Two linearly independent vectors form a base.
- We can reach any point in using:

$$\vec{a} = \lambda_1 \vec{u} + \lambda_2 \vec{v}$$

- If $\vec{u}$ and $\vec{v}$ are perpendicular unit vectors, the base is orthonormal.
- The Cartesian coordinate system is an example of this, with $\vec{u} = (1,0)$ and $\vec{v} = (0,1)$.

By manipulating $\vec{u}$ and $\vec{v}$, we can create a ‘coordinate system’ within a coordinate system.

Matrices

Bases in $\mathbb{R}^2$ and $\mathbb{R}^3$

This extends naturally to $\mathbb{R}^3$:

Three vectors, $\vec{u}$, $\vec{v}$ and $\vec{w}$ allow us to reach any point in 3D space;

$$a = \lambda_1 \vec{u} + \lambda_2 \vec{v} + \lambda_3 \vec{w}$$

Again, manipulating $\vec{u}$, $\vec{v}$ and $\vec{w}$ changes where coordinates specified as $(\lambda_1, \lambda_2, \lambda_3)$ end up.

Matrices

Matrices

A vector is an ordered set of d scalar values (i.e., a d-tuple):

$$\vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \text{ or } (v_1, v_2, v_3) \text{ or } \ldots$$

A $m \times n$ matrix is an array of $m \cdot n$ scalar values, sorted in m rows and n columns:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

The elements a_{ij} are referred to as the *coefficients* of the matrix (or elements, entries). Note that here i is the row; j is the column.

Matrices

Terminology – special matrices

- A *diagonal matrix* is a matrix for which all elements a_{ij} are zero if $i \neq j$.
- An *identity matrix* is a diagonal matrix where each element $a_{ii} = 1$.
- The *zero matrix* contains only zeroes.

$$A = \begin{pmatrix} 1.5 & 0 & 0 \\ 0 & 0.99 & 0 \\ 0 & 0 & 3.14 \end{pmatrix} \qquad A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

x y z

Before we continue, what *is* a matrix?

- Just a group of numbers;
- In graphics: often a representation of a coordinate system.

Matrices

Matrices - operations

Matrix addition is defined as:

$A = B + C$, with: $c_{ij} = a_{ij} + b_{ij}$

Note that addition is only defined for matrices with the same dimensions.

Example:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 2 & 2 \\ 4 & 4 \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 4 & 5 \end{pmatrix}$$

Subtraction works the same.

Matrices

Matrices - operations

Multiplying a matrix with a scalar is defined as follows:

$A = \lambda B$, with: $a_{ij} = \lambda b_{ij}$

Example:

$$2\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} = \begin{pmatrix}2 & 0\\0 & 2\end{pmatrix}$$

Matrices

Matrices - operations

Multiplying a matrix (dimensions $m_A \times n_A$) with another matrix (dimensions $m_B \times n_B$):

$$C = AB, \text{ with: } \quad c_{ij} = \sum_{k=1}^{n_A} a_{ik}\, b_{kj}$$

Example:

$$\begin{pmatrix} 2 & 6 & 1 \\ 5 & 2 & 4 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} 17 & 44 \\ 21 & 54 \end{pmatrix}$$

$$c_{11} = \sum_{k=1}^{2} a_{1k}\, b_{k1} = 2*1+6*2+1*3 = 17$$

$$c_{21} = \sum_{k=1}^{2} a_{2k}\, b_{k1} = 5*1+2*2+4*3 = 21$$

$$c_{12} = \sum_{k=1}^{2} a_{1k}\, b_{k2} = 2*4+6*5+1*6 = 44$$

$$c_{22} = \sum_{k=1}^{2} a_{2k}\, b_{k2} = 5*4+2*5+4*6 = 54$$

Note the dimensions of the resulting matrix: $m_A \times n_B$.

Matrix multiplication is only defined if $n_A = m_B$.

Matrices

Matrices - operations

Doing matrix multiplication manually:

$$\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 6 & 1 \\ 5 & 2 & 4 \end{pmatrix} \quad \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}$$

Note that each cell in the resulting matrix is essentially the dot product of a row and a column.

Some properties:

Matrix multiplication is distributive over addition:

$$A(B+C) = AB + AC$$
$$(A+B)C = AC + BC$$

...and associative:

$$(AB)C = A(BC)$$

However, matrix multiplication is not commutative, i.e., in general:

$$AB \neq BA$$

Matrices

Matrices - operations

Doing matrix multiplication manually:

$$\begin{pmatrix} 2 & 6 & 1 \\ 5 & 2 & 4 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$$

Multiplying by the zero matrix yields the zero matrix:

$$0A = A0 = 0$$

Multiplying by the identity matrix yields the original matrix:

$$IA = AI = A$$

Matrices

Matrices - operations

The *transpose* A^T of an $m \times n$ matrix is an $n \times m$ matrix that is obtained by interchanging rows and columns: a_{ij} becomes a_{ji} for all i, j:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \qquad A^T = \begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$$

The transpose of the product of two matrices is:

$$(AB)^T = B^T A^T$$

Matrices

Matrices - operations

The *inverse* of a matrix A is a matrix A^{-1} such that

$$AA^{-1} = A^{-1}A = I$$

Note: only square matrix possibly have an inverse.

Matrices

Matrices - operations

We can multiply a *d*-dimensional vector by an $m \times d$ matrix:

$$\begin{pmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{md} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} = \begin{pmatrix} a_{11}v_1 + \cdots + a_{1d}v_d \\ \cdots \quad + \cdots + \quad \cdots \\ a_{m1}v_1 + \cdots + a_{md}v_d \end{pmatrix}$$

Note:

This is the same as matrix concatenation; the vector is simply an $m \times 1$ *matrix.*

Example: multiply a 3D vector by a 3x3 matrix:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_{11}x + a_{12}y + a_{13}z \\ a_{21}x + a_{22}y + a_{23}z \\ a_{31}x + a_{32}y + a_{33}z \end{pmatrix}$$

Matrices

Matrices - operations

We can multiply a *d*-dimensional vector by an $m \times d$ matrix:

$$\begin{pmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{md} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} = \begin{pmatrix} a_{11}v_1 + \cdots + a_{1d}v_d \\ \cdots \quad + \cdots + \quad \cdots \\ a_{m1}v_1 + \cdots + a_{md}v_d \end{pmatrix}$$

Note:

This is the same as matrix concatenation; the vector is simply an $m \times 1$ matrix.

Example: multiply a 3D vector by a 3x3 matrix:

$$\begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} u_x x + v_x y + w_x z \\ u_y x + v_y y + w_y z \\ u_z x + v_z y + w_z z \end{pmatrix} = x\vec{u} + y\vec{v} + z\vec{w}$$

u **v** **w**

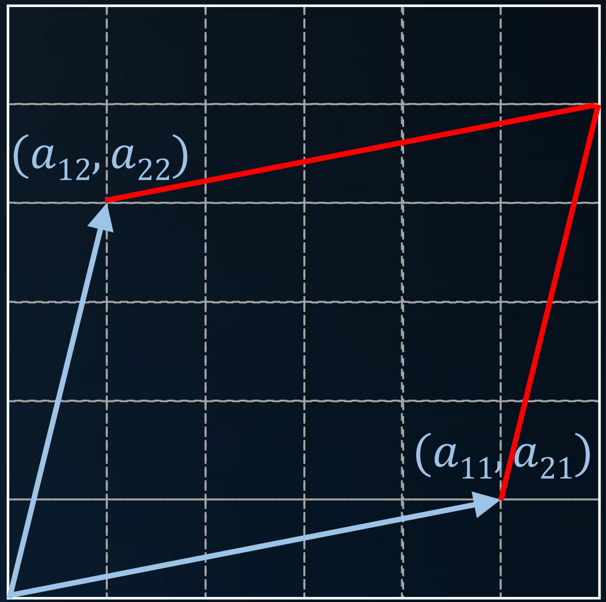
Matrices

Matrices – determinant

The determinant $|A|$ of an $n \times n$ matrix A is the signed area or volume spanned by its column vectors.

Example (in $\mathbb{R}^2$):

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad \det A = |A| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$$



In this case, the determinant is the oriented area of the *parallelogram* defined by the two column vectors.

The determinant is positive if the vectors are counter-clockwise, or negative if they are clockwise. Therefore:

$$\det\left|\overrightarrow{a1}\ \overrightarrow{a2}\right| = -\det\left|\overrightarrow{a2}\ \overrightarrow{a1}\right|$$

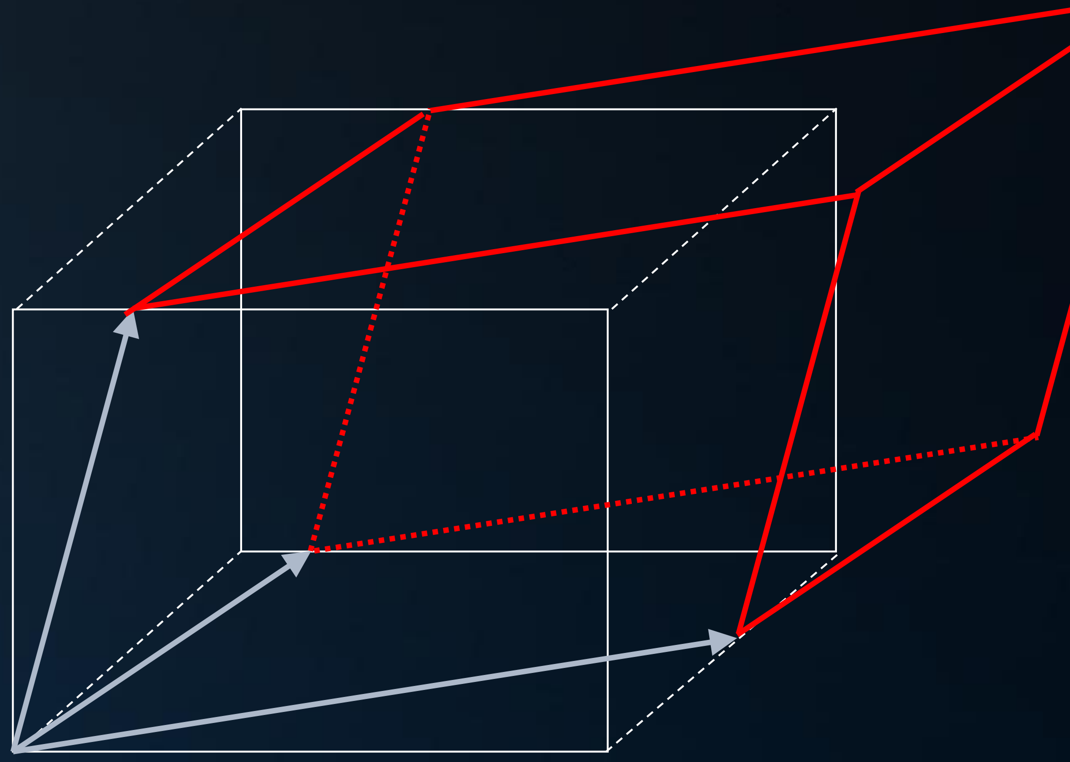
Matrices

Matrices – determinant

The determinant $|A|$ of an $n \times n$ matrix A is the signed volume spanned by its column vectors.

In $\mathbb{R}^3$, the determinant is the oriented area of the parallelepiped defined by the three column vectors.

$$\det A = |A| = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$



Matrices

Matrices – determinant

Calculating determinants: Laplace’s expansion.

The determinant of a matrix is the sum of the products of the elements of any row or column of the matrix with their *cofactors*.

The cofactor of an entry a_{ij} in an $n \times n$ matrix A is:

- The determinant of the $(n-1) \times (n-1)$ matrix A',
- that is obtained from A by removing the i-th row and j-th column,
- multiplied by -1^{i+j}.

Example:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$a_{11}^{c} = \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} * (-1^{2})$$

$$a_{12}^{c} = \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} * (-1^{3})$$

$$a_{13}^{c} = \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} * (-1^{4})$$

$$|A| = a_{11}\, a_{11}^{c} + a_{12}\, a_{12}^{c} + a_{13}\, a_{13}^{c}$$

Matrices

Matrices – determinant

Full example for 3×3 matrix:

$$\begin{vmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{vmatrix} = 0 \begin{vmatrix} 4 & 5 \\ 7 & 8 \end{vmatrix} - 1 \begin{vmatrix} 3 & 5 \\ 6 & 8 \end{vmatrix} + 2 \begin{vmatrix} 3 & 4 \\ 6 & 7 \end{vmatrix}$$

$$\begin{vmatrix} 3 & 5 \\ 6 & 8 \end{vmatrix} = 3 * 8 * -1^2 + 5 * 6 * -1^3 = -6$$

$$\begin{vmatrix} 3 & 4 \\ 6 & 7 \end{vmatrix} = 3 * 7 * -1^2 + 4 * 6 * -1^3 = -3$$

$0 - 1 * -6 + 2 * -3 = 0.$

Generic approach for a for 3×3 matrix:

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - \cdots$$

$$= (aei + bfg + cdh) - (ceg + afh + bdi)$$

$$\begin{vmatrix} a & b & c & a & b & c \\ d & e & f & d & e & f \\ g & h & i & g & h & i \end{vmatrix}$$

Rule of Sarrus for 2×2: $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$

Matrices

Matrices – adjoint

The *adjoint* (or *adjugate*) $\tilde{A}$ of matrix A is the transpose of the cofactor matrix of A.

Example:

$$A = \begin{pmatrix} 2 & 5 \\ 1 & 3 \end{pmatrix} \rightarrow C = \begin{pmatrix} 3 * (-1^2) & 1 * (-1^3) \\ 5 * (-1^3) & 2 * (-1^4) \end{pmatrix} = \begin{pmatrix} 3 & -1 \\ -5 & 2 \end{pmatrix}$$

$$adj(A) = C^T = \begin{pmatrix} 3 & -5 \\ -1 & 2 \end{pmatrix}.$$

The cofactor of an entry a_{ij} in an $n \times n$ matrix A is:

- The determinant of the $(n-1) \times (n-1)$ matrix A',
- that is obtained from A by removing the i-th row and j-th column,
- multiplied by -1^{i+j}.

Matrices

Matrices – inverse

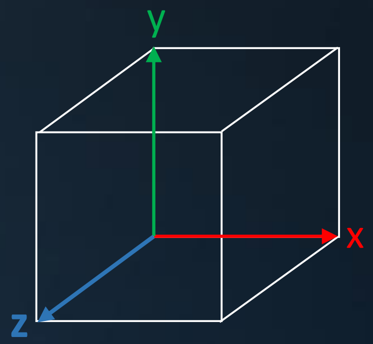
The adjoint is used to calculate the inverse A^{-1} of a matrix A:

$$A^{-1} = \frac{\tilde{A}}{|A|}$$

Matrices

Matrices – overview

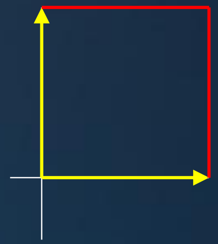
$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



$n \times m$: n rows, m columns

$$\det(A) = |A| = 1 = (aei + bfg + cdh) - (ceg + afh + bdi)$$

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



note: $\begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix} = -1$, and: $\det|\overrightarrow{a1}\ \overrightarrow{a2}| = -\det|\overrightarrow{a2}\ \overrightarrow{a1}|$

cofactor $a_{11}^{c} = \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} * (-1^2)$

Adjoint $\tilde{A}$ of A is C^T; inverse A^{-1} is $\frac{\tilde{A}}{|A|}$.

Today’s Agenda:

- Rendering Overview
- Matrices
- Transforms

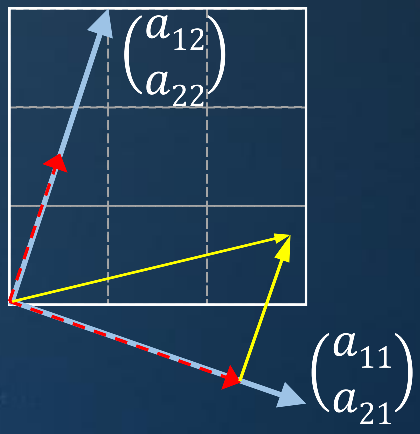
Transforms

Spaces - introduction

As we have seen before, we can multiply a matrix with a vector.

In 2D: $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{pmatrix} \quad = x \begin{pmatrix} a_{11} \\ a_{21} \end{pmatrix} + y \begin{pmatrix} a_{12} \\ a_{22} \end{pmatrix}$

In 3D: $\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_{11}x + a_{12}y + a_{13}z \\ a_{21}x + a_{22}y + a_{23}z \\ a_{31}x + a_{32}y + a_{33}z \end{pmatrix} \quad = x \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix} + y \begin{pmatrix} a_{12} \\ a_{22} \\ a_{32} \end{pmatrix} + z \begin{pmatrix} a_{13} \\ a_{23} \\ a_{33} \end{pmatrix}$



Geometric interpretation:

scalar multiplication of $\begin{pmatrix} a_{11} \\ a_{21} \end{pmatrix}$ by x, plus

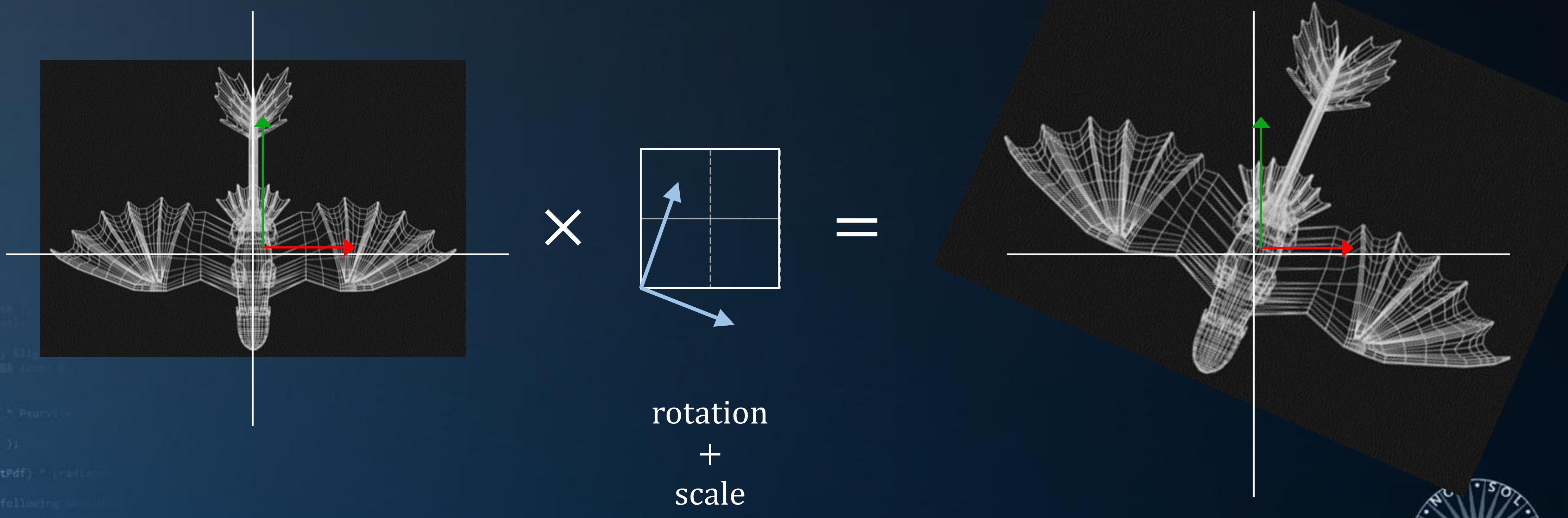
scalar multiplication of $\begin{pmatrix} a_{12} \\ a_{22} \end{pmatrix}$ by y yields

transformed point.

Transforms

Spaces – introduction

A matrix allows us to *transform* a coordinate system.



Transforms

Spaces – scaling

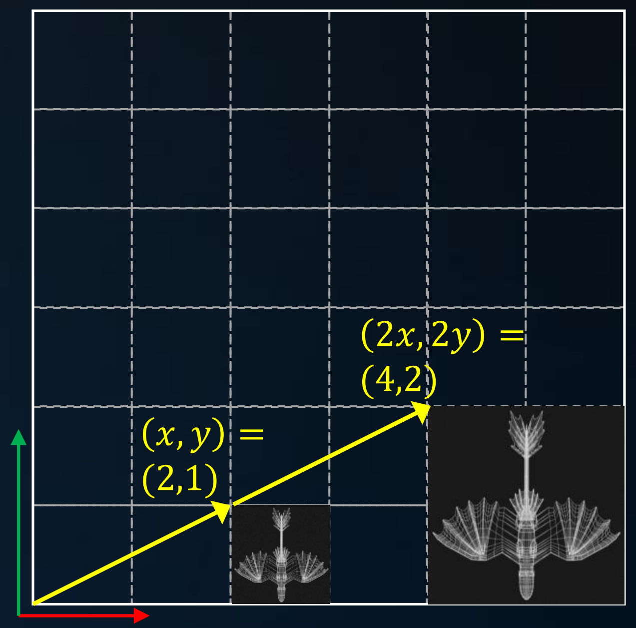
To scale by a factor 2 with respect to the origin, we apply the matrix

$$\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

Applied to a vector, we get:

$$\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2x + 0y \\ 0x + 2y \end{pmatrix} = \begin{pmatrix} 2x \\ 2y \end{pmatrix}$$

This is called *uniform scaling*.



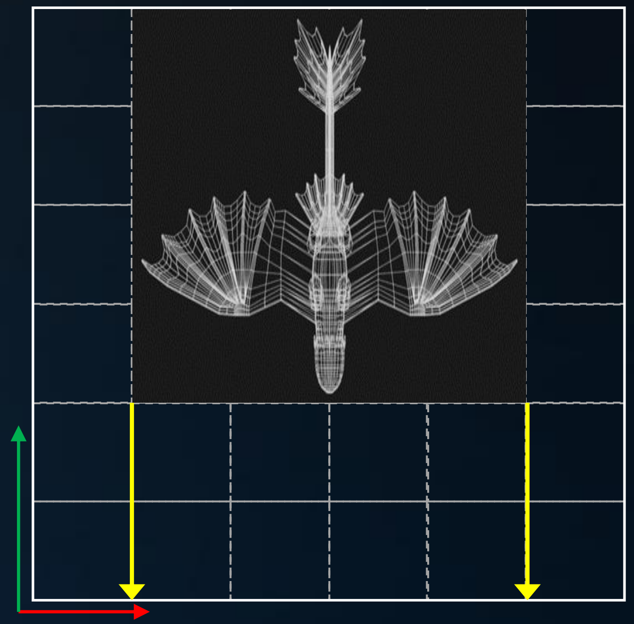
Transforms

Spaces – projection

If we set one of the a_{ii} to 0, we get an *orthographic projection*.

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

This is useful for projecting a shadow of the dragon on the x-axis, or to draw a 3D object on a 2D screen.

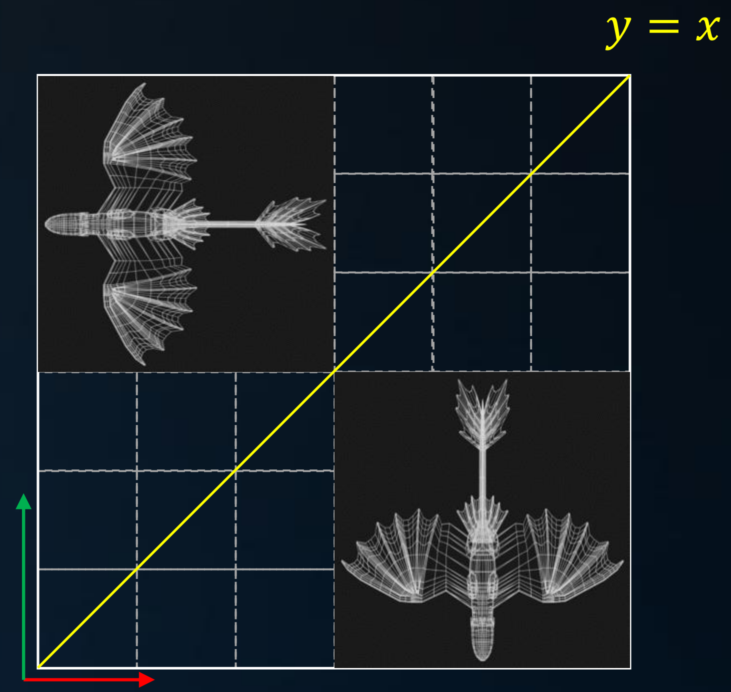


Transforms

Spaces – reflection

We can construct a matrix that will swap x and y coordinates to get a reflection in the line $y = x$:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0x + 1y \\ 1x + 0y \end{pmatrix} = \begin{pmatrix} y \\ x \end{pmatrix}$$



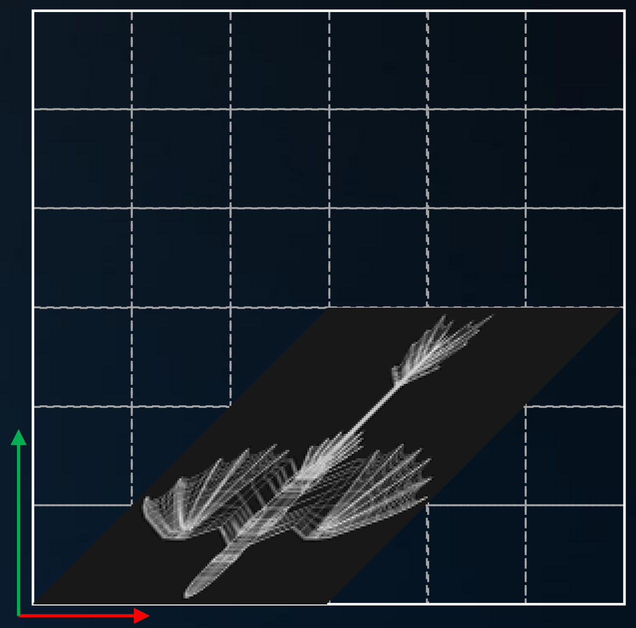
Transforms

Spaces – shearing

Pushing things sideways:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1x + 1y \\ 1y \end{pmatrix} = \begin{pmatrix} x + y \\ x \end{pmatrix}$$

This is called *shearing*.



Transforms

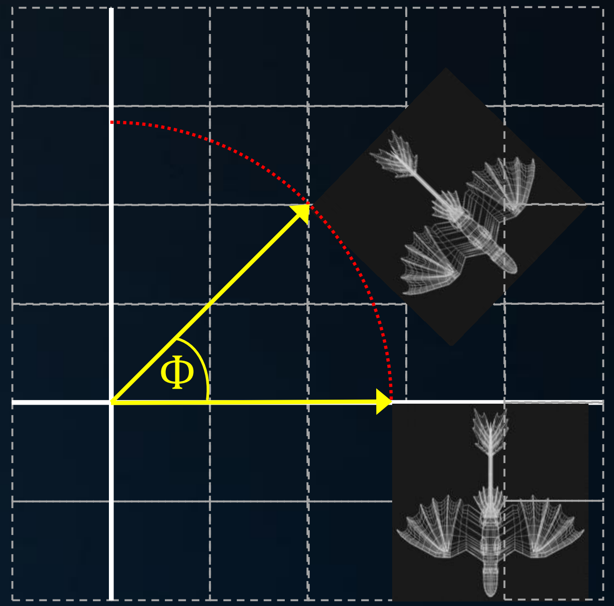
Spaces – rotation

To rotate counter-clockwise about the origin, we use the following matrix:

$$\begin{pmatrix} \cos\emptyset & -sin\emptyset \\ \sin\emptyset & cos\emptyset \end{pmatrix}$$

For clockwise rotation, we use

$$\begin{pmatrix} \cos\emptyset & sin\emptyset \\ -\sin\emptyset & cos\emptyset \end{pmatrix}$$



Transforms

Spaces – linear transformations

A function $T\colon\ \mathbb{R}^n \rightarrow \mathbb{R}^m$ is called a linear transformation, if it satisfies:

1. $T(\vec{u}+\vec{v}) = T(\vec{u}) + T(\vec{v})$
   for all $\vec{u}, \vec{v} \in \mathbb{R}^n$.
2. $T(c\vec{v}) = cT(\vec{v})$
   for all $\vec{v} \in \mathbb{R}^n$ and all scalars c.

Linear transformations can be represented by matrices.

We can summarize both conditions into one equation:

$T(c_1\vec{u} + c_2\vec{v}) = c_1 T(\vec{u}) + c_2 T(\vec{v})$
for all $\vec{u}, \vec{v} \in \mathbb{R}^n$ and all scalars c_1, c_2.

Transforms

Spaces – linear transformations

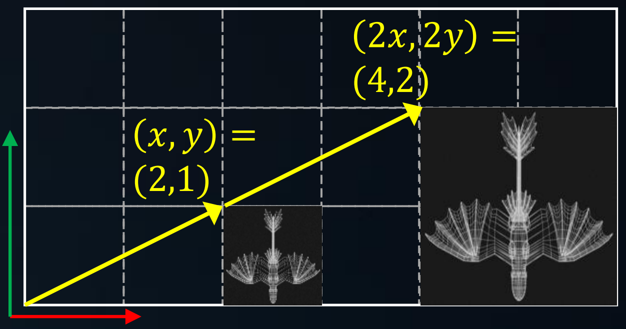
$T(c_1\vec{u} + c_2\vec{v}) = c_1T(\vec{u}) + c_2T(\vec{v})$
for all $\vec{u}, \vec{v} \in \mathbb{R}^n$ and all scalars c_1, c_2.

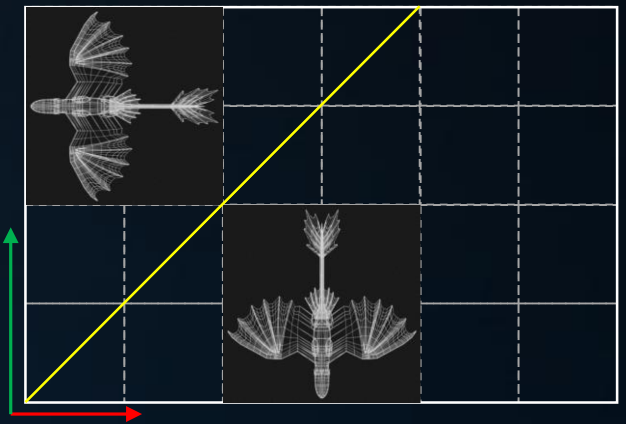
Remember Cartesian coordinates, where each vector $\vec{w}$ can be expressed as a linear combination of base vectors $\vec{u}$ and $\vec{v}$:

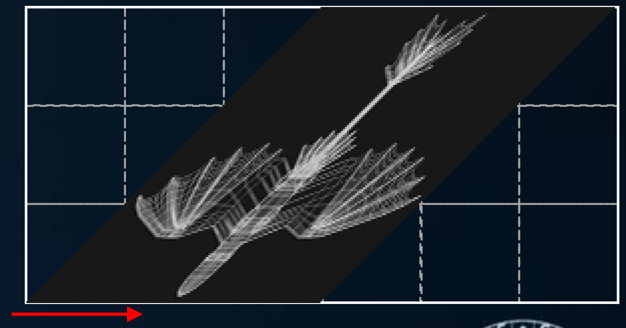
$$\vec{w} = \binom{x}{y} = x\binom{1}{0} + y\binom{0}{1}$$

If we apply a linear transform T to this vector, we get

$$T\left(\binom{x}{y}\right) = \mathrm{T}\left(x\binom{1}{0} + y\binom{0}{1}\right) = xT\left(\binom{1}{0}\right) + yT\left(\binom{0}{1}\right)$$





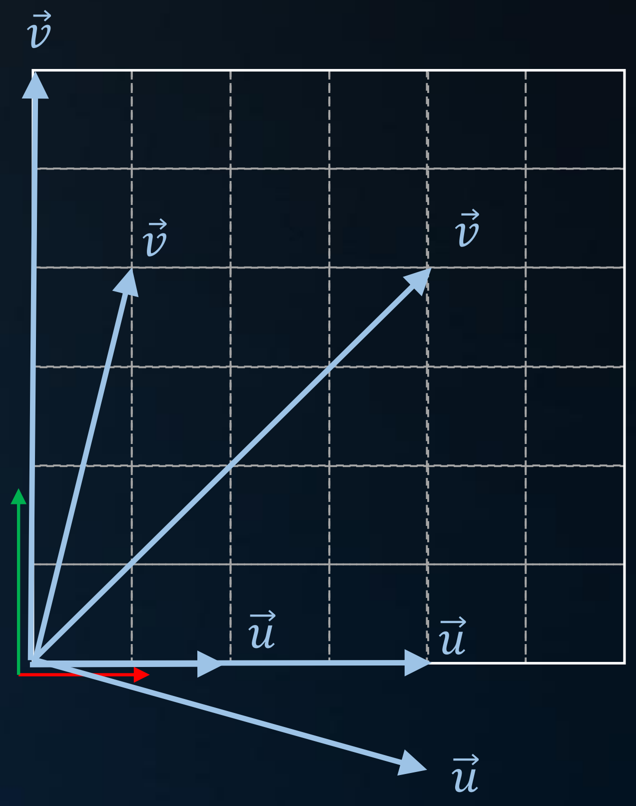


Transforms

Spaces – linear transformations

$T(c_1\vec{u} + c_2\vec{v}) = c_1T(\vec{u}) + c_2T(\vec{v})$
for all $\vec{u}, \vec{v} \in \mathbb{R}^n$ and all scalars c_1, c_2.

Matrices are constructed conveniently using two base vectors.



Transforms

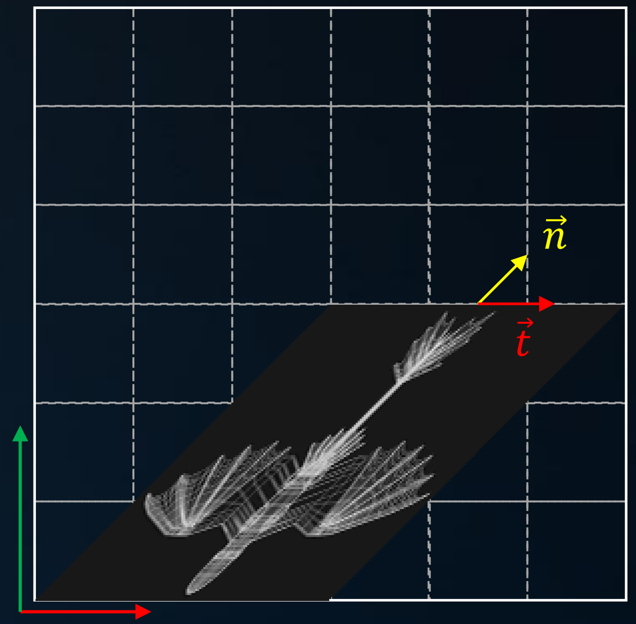
Spaces – transforming normals

Unfortunately, normals are not always transformed correctly.

To transform a normal vector $\vec{n}$ correctly under a given linear transformation A, we have to apply the matrix

$$(A^{-1})^T$$

Why?



Note: if the transform is orthonormal,
$A^{-1} = A^T$; therefore $(A^{-1})^T = A$.

Transforms

Spaces – transforming normals

We know that tangent vectors are transformed correctly: $A\vec{t} = \vec{t}_A$. But: $A\vec{n} \neq \vec{n}_A$.
Goal: **find a matrix M that transforms $\vec{n}$ correctly**, i.e. $M\vec{n} = \vec{n}_M$, where $\vec{n}_M$ is the correct normal of the transformed surface.

Because the original normal vector $\vec{n}$ is perpendicular to the original tangent vector $\vec{t}$, we know that $\vec{n}^T\vec{t} = 0$. This is the same as $\vec{n}^T\, I\, \vec{t} = 0$. Since $I = A^{-1}A$, this is the same as $\vec{n}^T\,(A^{-1}A)\,\vec{t} = 0$.

Because $A\vec{t} = t_A$ is the correctly transformed tangent vector, we have $\vec{n}^T A^{-1}\,\vec{t}_A = 0$.

Because their scalar product is 0, $\vec{n}^T A^{-1}$ must be orthogonal to $\vec{t}_A$. So, the vector we are looking for must be: $\vec{n}_M^T = \vec{n}^T A^{-1}$ (which suggests $M = A^{-1}$).

Because of how matrix multiplication is defined, $\vec{n}_M^T$ and $\vec{n}^T$ are transposed vectors. We can rewrite this to $\vec{n}_M = (\vec{n}^T A^{-1})^T$. And finally, remember that $(AB)^T = B^T A^T$, which gets us $\vec{n}_M = (A^{-1})^T\vec{n}$.

Transforms

Spaces – needful things

Three things left undiscussed:

1. Reverting a transform
2. Combining transforms
3. Translation

Reverting a transform:

Invert the matrix.

Note: doesn't always work; e.g. the matrix for orthographic projection has no inverse.

Combining transforms:

Use matrix multiplication.

Note: matrix multiplication is not commutative, mind the order!

Transforms

Spaces – translation

Translation is not a linear transform.

With linear transforms, we get:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{pmatrix}$$

But we need something like:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + x_t \\ y + y_t \end{pmatrix}$$

We can do this with a combination of linear transformations and translations called *affine transformations*.

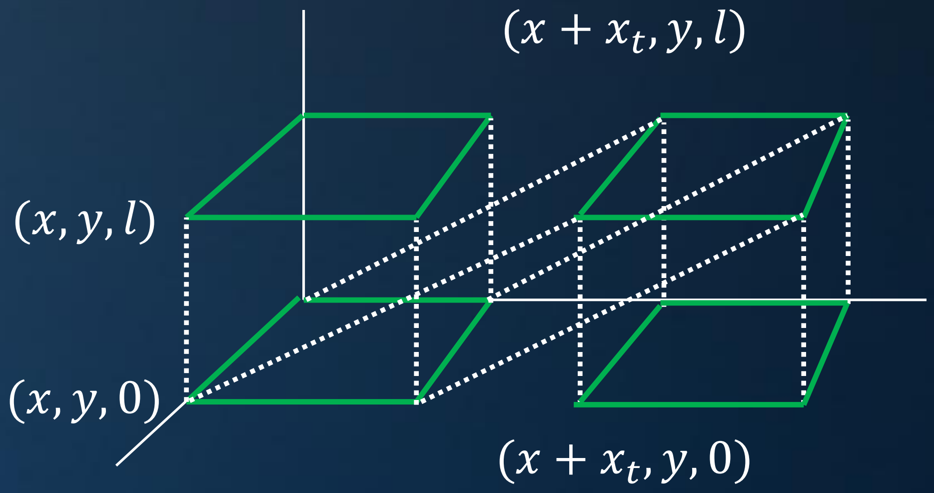
Transforms

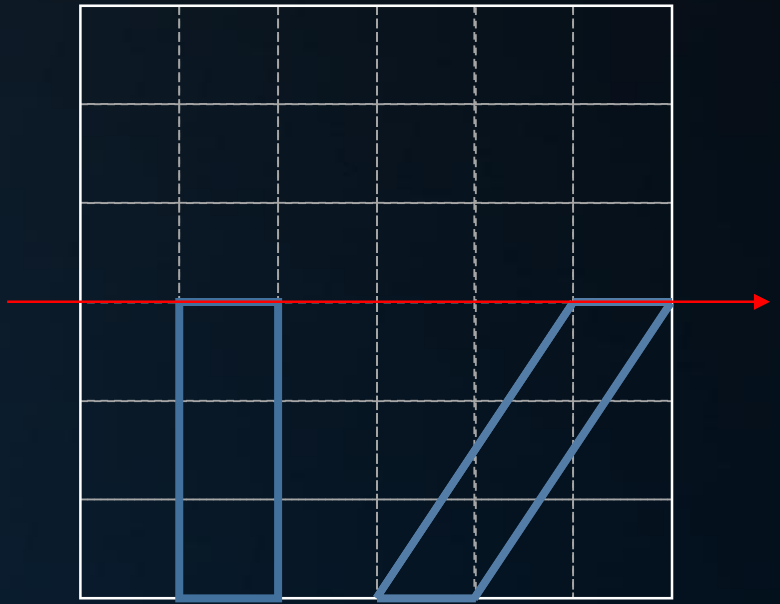
Spaces – translation

Observe: in 2D, shearing “pushes things sideways” (e.g., in the x direction), in a “fixed level” (the y value).

We are thus performing a translation in a 1D subspace (a line), using matrix multiplication in 2D.

In 3D, shearing leads to translation in a 2D subspace, i.e. a plane.





Transforms

Spaces – translation

By adding a 3rd dimension to 2D space, we can use matrix multiplication to do translation.

$$M \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x + x_t \\ y + y_t \\ z \end{pmatrix}$$

But: what does matrix M look like? What about x_t and y_t?
And how do we deal with the third coordinate z?

Transforms

Spaces – translation

Shearing in 3D based on the z coordinate is a simple generalization of 2D shearing:

$$\begin{pmatrix} 1 & 0 & x_t \\ 0 & 1 & y_t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + zxt + 0z \\ 1y + zyt + 0z \\ 0x + 0y + z \end{pmatrix} = \begin{pmatrix} x + x_t z \\ y + y_t z \\ z \end{pmatrix}$$

The final step is to set z to 1.

$$\begin{pmatrix} 1 & 0 & x_t \\ 0 & 1 & y_t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + x_t \\ y + y_t \\ 1 \end{pmatrix}$$

Transforms

Spaces – translation

Translations in 2D can be represented as shearing in 3D by looking at the plane $z = 1$.

By representing our 2D points (x, y) by 3D vectors $(x, y, 1)$, we can translate them about (x_t, y_t) by applying the following matrix:

$$\begin{pmatrix} 1 & 0 & x_t \\ 0 & 1 & y_t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + x_t \\ y + y_t \\ 1 \end{pmatrix}$$

That works for points. What about vectors? We use the following transform:

$$\begin{pmatrix} 1 & 0 & x_t \\ 0 & 1 & y_t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$$

Transforms

Spaces – translation

Affine transformations (i.e., linear transformations and translations) can be done with matrix multiplication if we add *homogeneous coordinates*, i.e.

- A third coordinate $z = 1$ to each *point*: $p = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$
- A third coordinate z $= 0$ to each vector: $\vec{v} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$
- A third row (0 0 1) to each matrix.

Transforms

Spaces – translation

These concepts apply naturally to 3D, in which case we again add a homogeneous coordinate, i.e.

- A fourth coordinate $w = 1$ to each *point;*
- A fourth coordinate $w = 0$ to each vector;
- A fourth row (0 0 0 1) to each matrix.

Today's Agenda:

- Rendering Overview
- Matrices
- Transforms

INFOGR – Computer Graphics

J. Bikker - April-July 2015 - Lecture 5: “3D Engine Fundamentals”

END of “3D Engine Fundamentals”

next lecture: “Transformations”