

Data, Math and Methods

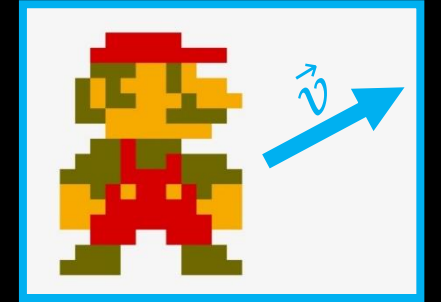
Week 11, Vectors & Matrices



Today

Vectors and matrices – Linear Algebra

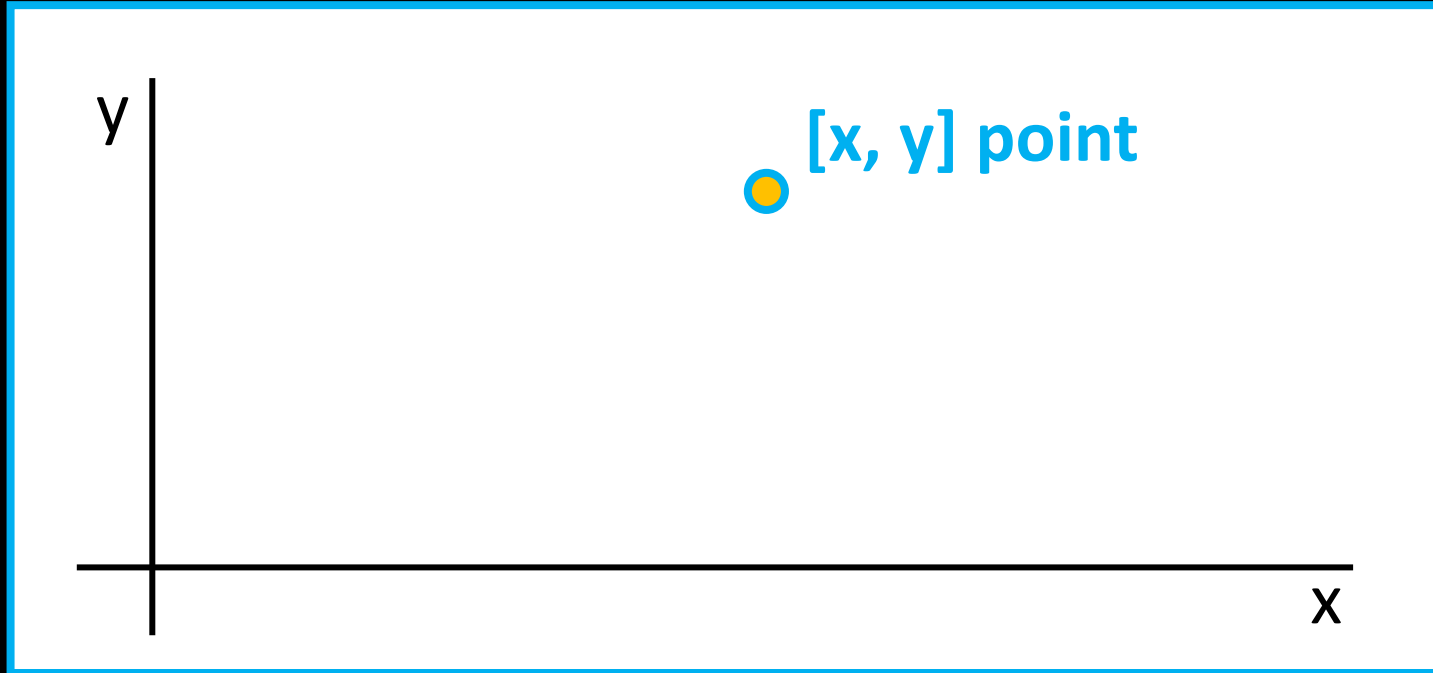
- **Vector** operations in math and programming
- **Matrix** operations and their uses in Computer Sciences



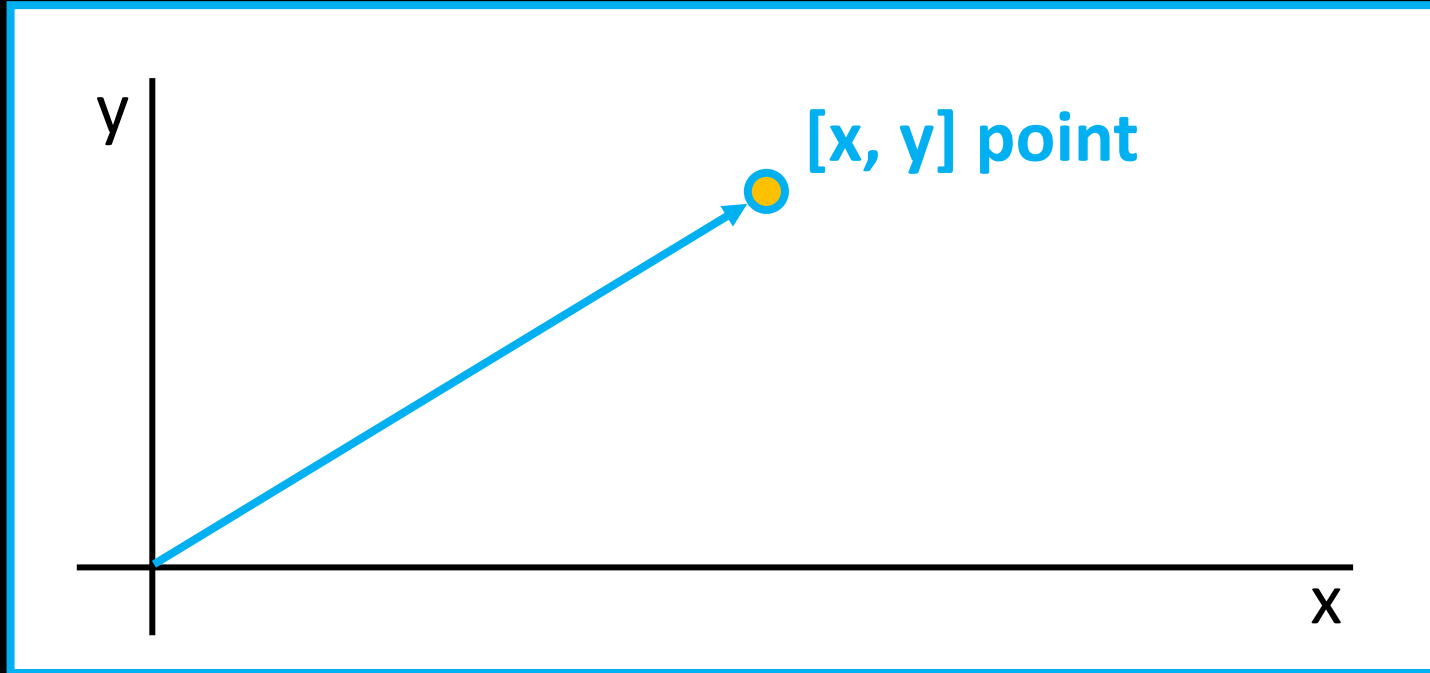
Practical session

- Vector manipulation in Python

Vectors



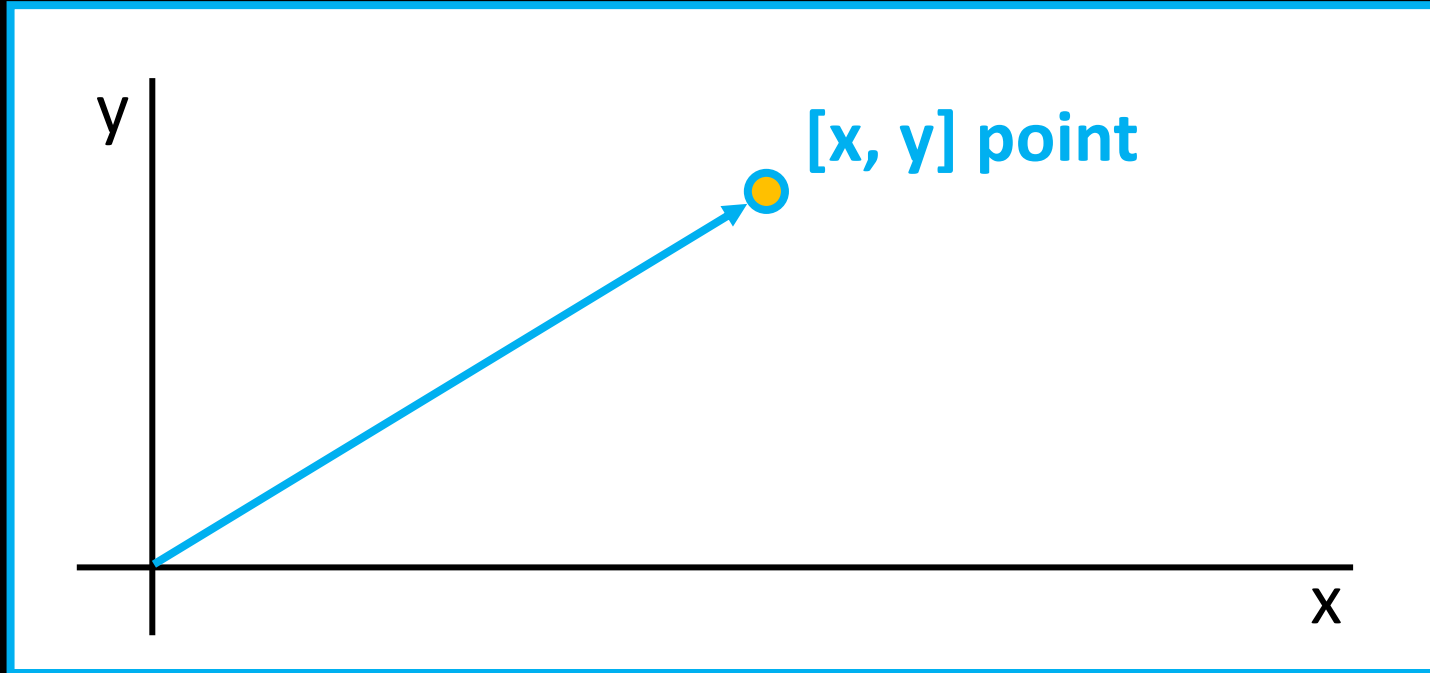
Vectors



(x, y) vector

- Vector, is without a fixed location
- Point, is with a fixed location

Vectors



(x, y) vector

- Vector, is without a fixed location
- Point, is with a fixed location

- The same in **higher dimensions**:
 - **(x,y,z, ...)** or in general **(x₁, x₂, x₃, ..., x_n)** → dimensionality = **n**
- In **programming** we have a similar structure:
 - array = **[x₁, x₂, x₃, ..., x_n]** or in python numpy: **np.asarray([x₁, x₂, x₃, ..., x_n])**

Operations with Vectors

What can we do with vectors?

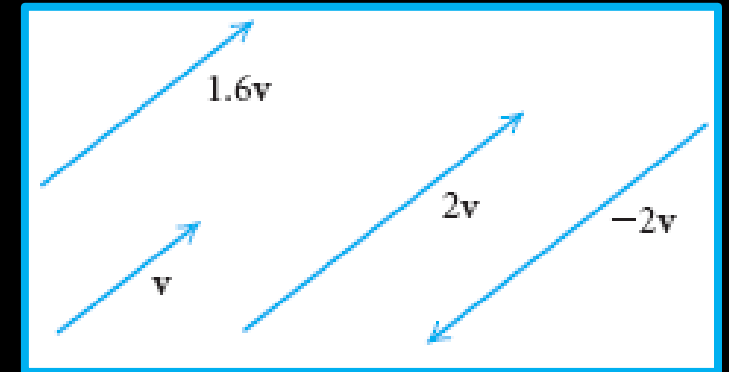
- $\vec{v} = (v_1, v_2)$

- Multiplication by a real number:

$$\vec{u} = a * \vec{v}$$

$$a \in \mathbb{R}$$

$$\vec{u} = (a * v_1, a * v_2)$$



Operations with Vectors

Given two vectors of the same dimensionality ...

- $\vec{v} = (v_1, v_2)$

- $\vec{u} = (u_1, u_2)$

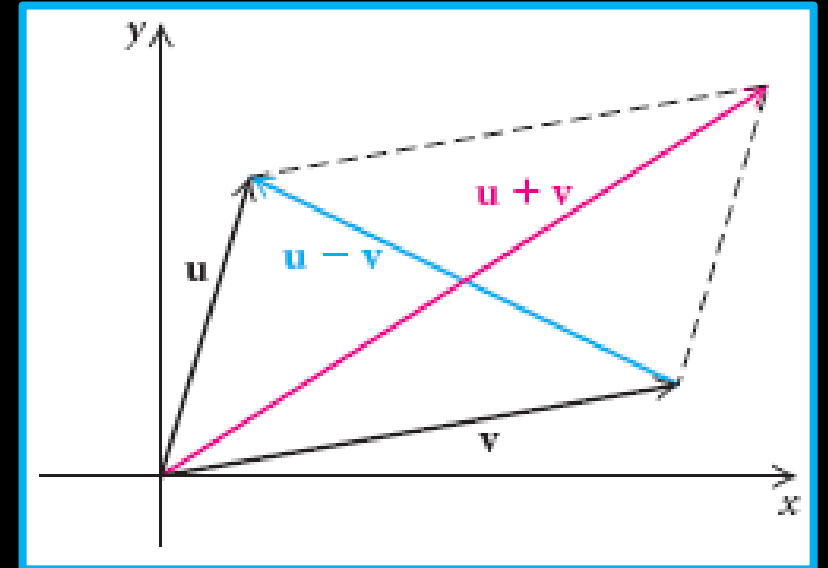
- Addition / Subtraction

$$\vec{u} + \vec{v}$$

$$(u_1 + v_1, u_2 + v_2)$$

$$\vec{u} - \vec{v}$$

$$(u_1 - v_1, u_2 - v_2)$$



Operations with Vectors

Normalization:

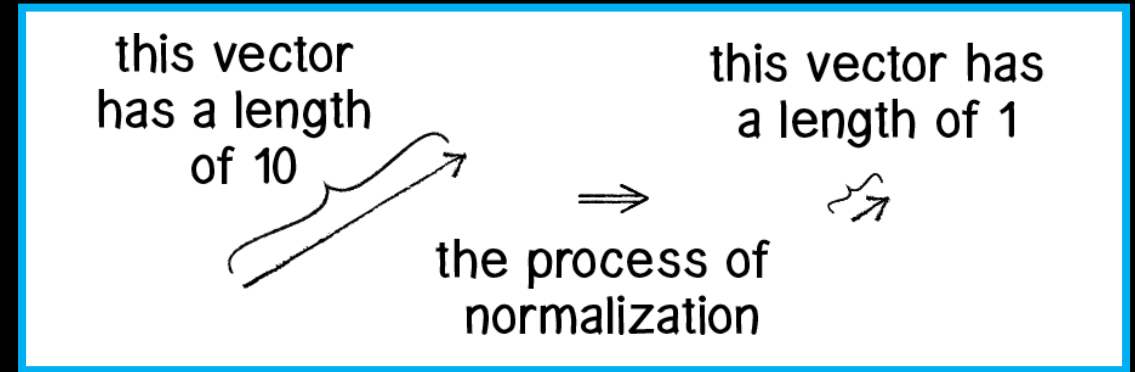
- Length of the vector:

$$|\vec{v}| = \sqrt{v_1^2 + v_2^2}$$

- Normalization:

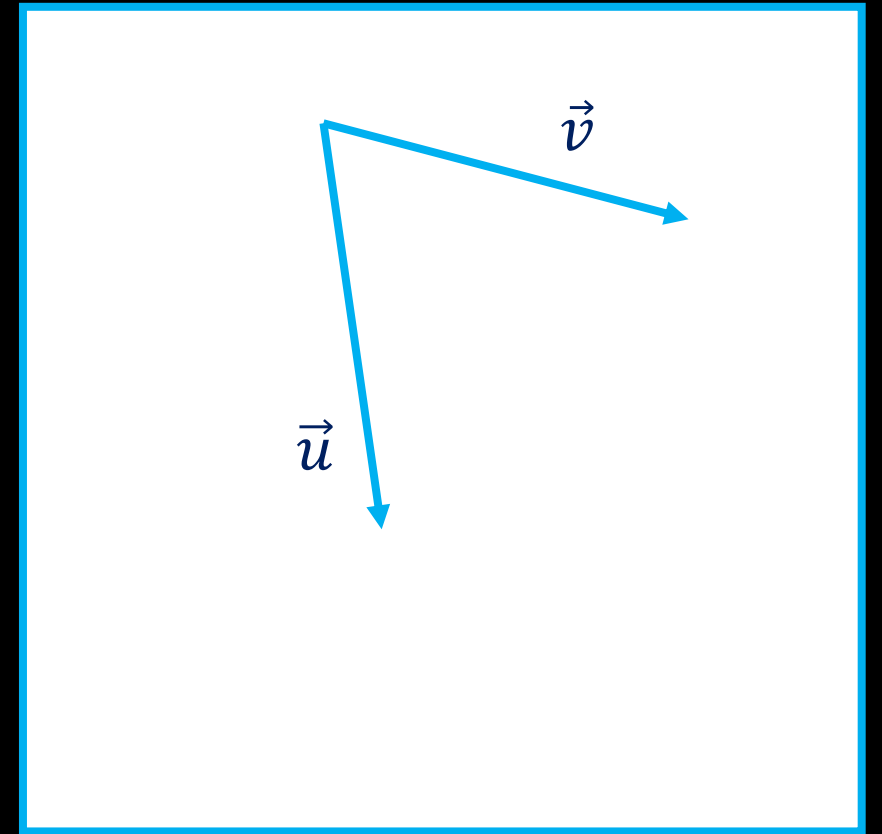
normalized vector $\frac{\vec{v}}{|\vec{v}|}$

The result is called **unit vector** and has a length of $\vec{1}$



Linear combination of vectors

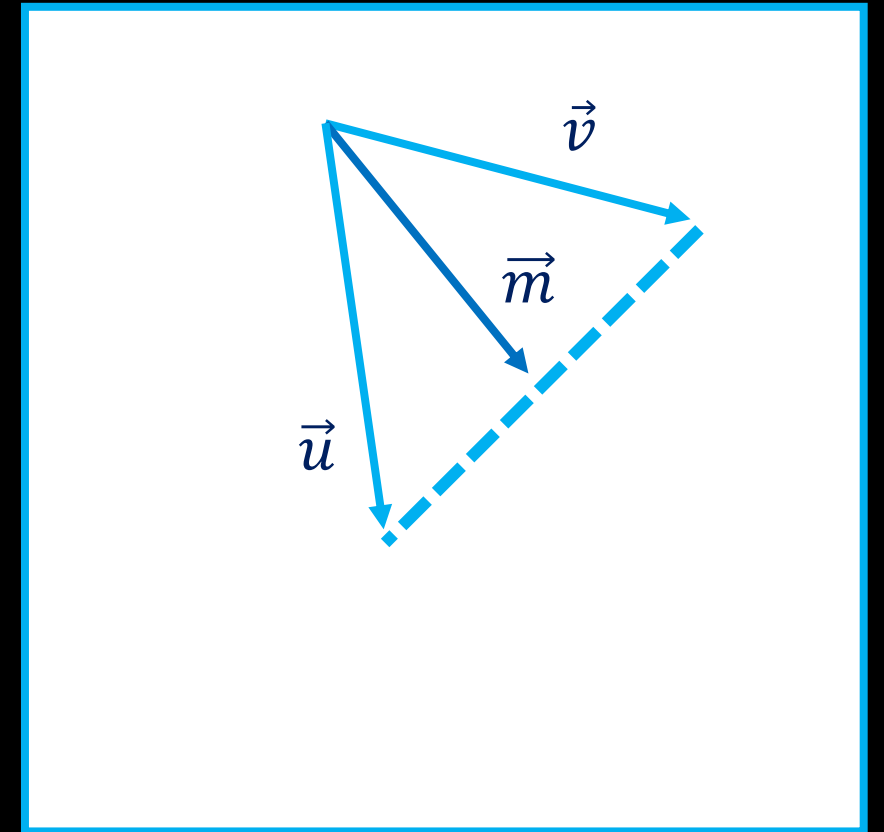
- Method of **combining vectors together** ...



Linear combination of vectors

- Method of **combining vectors together** ...
- Let's say we want to mix two vectors and **interpolate** between them
 - Useful in **animation** (if the vectors represented our graphics), ...
 - Useful when **going in between the data points**!

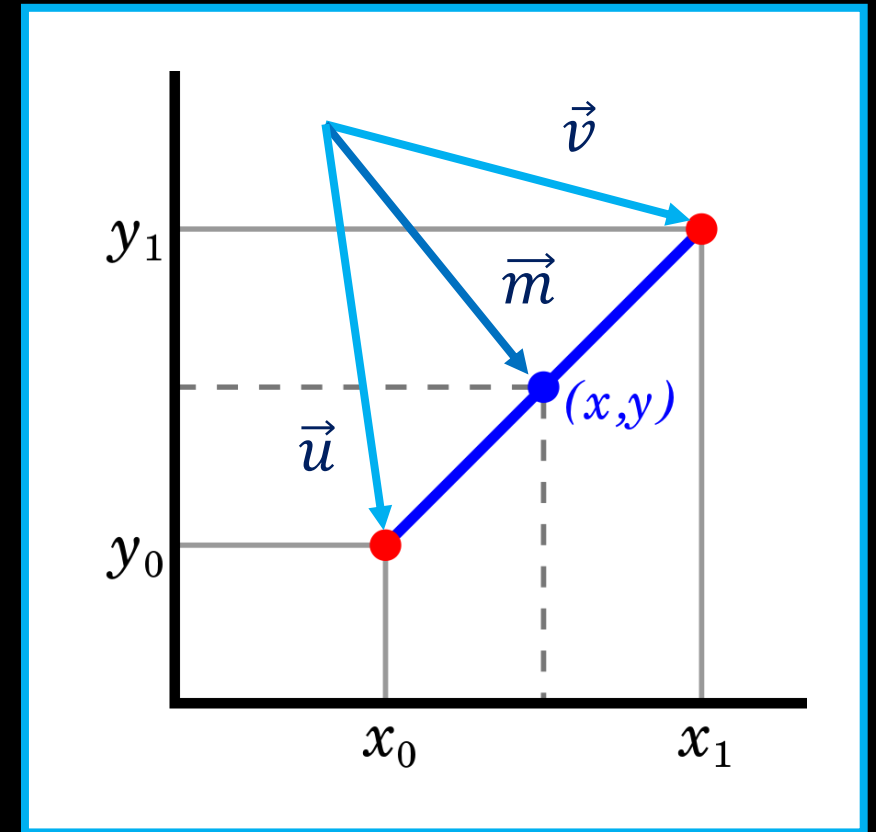
$$\vec{m} = a * \vec{u} + b * \vec{v}$$
$$a + b = 1$$



Linear combination of vectors

- Method of **combining vectors together** ...
- Let's say we want to mix two vectors and **interpolate** between them
 - Useful in **animation** (if the vectors represented our graphics), ...
 - Useful when **going in between the data points**!

$$\vec{m} = 0.5 * \vec{u} + 0.5 * \vec{v}$$
$$0.5 + 0.5 = 1$$



Demo: [geogebra.org/calculator/mqz6d4zw](https://www.geogebra.org/calculator/mqz6d4zw)

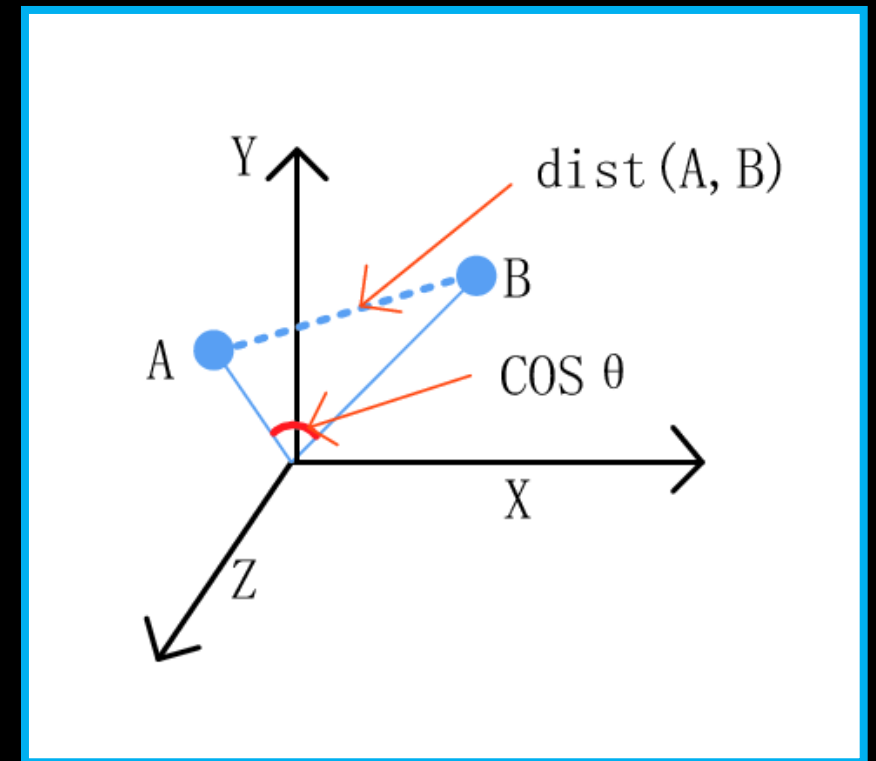
Angles between vectors

- **Dot product** = per element multiplication

$$\vec{u} \times \vec{v} = u_1 * v_1 + u_2 * v_2$$

- **Angle between two vectors** – cosine metric:

$$\cos \theta = \frac{\vec{u} \times \vec{v}}{|\vec{u}| * |\vec{v}|}$$



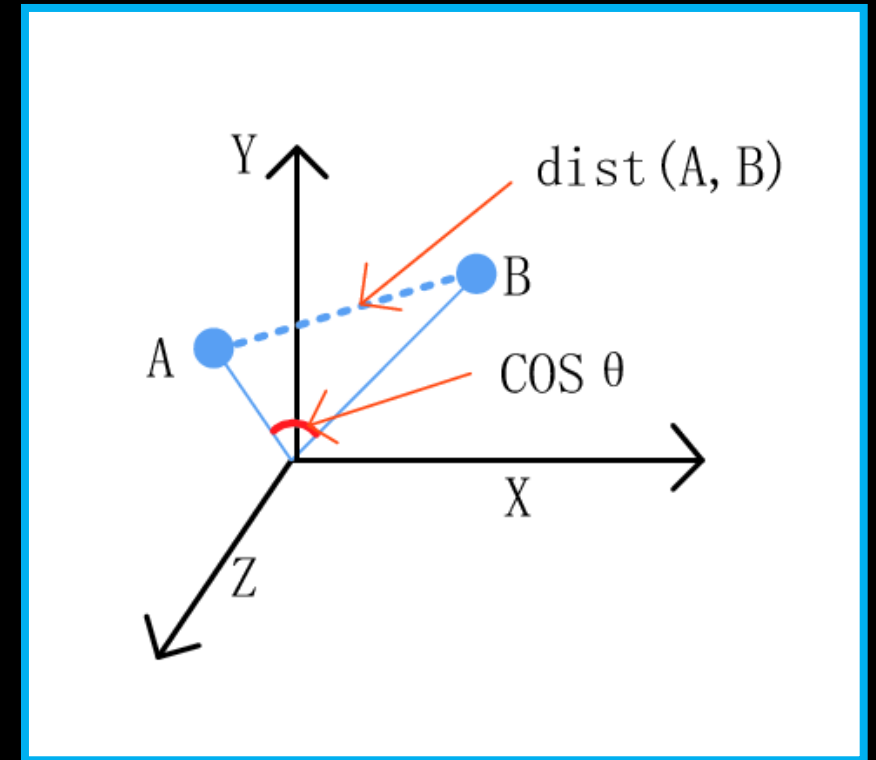
Angles between vectors

- **Dot product** = per element multiplication

$$\vec{u} \times \vec{v} = u_1 * v_1 + u_2 * v_2$$

- **Angle between two vectors** – cosine metric:

$$\cos \theta = \frac{\vec{u} \times \vec{v}}{|\vec{u}| * |\vec{v}|} = \frac{u_1 * v_1 + u_2 * v_2}{|\vec{u}| * |\vec{v}|}$$



Cosine vs Euclidian distance

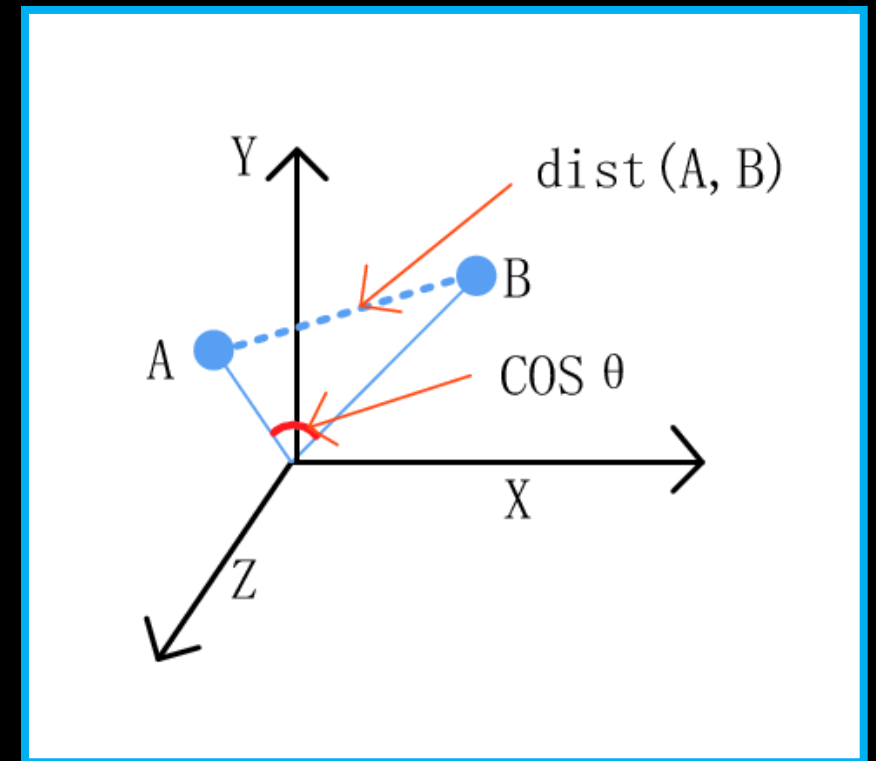
- Metrics of distance describe for us some kind of similarity between vectors ...
 - Small distance -> very similar vectors
 - Large distance -> very different vectors
- *PS: There are many types of metrics!*

- **Cosine similarity / distance:**

$$d_1 = \cos \theta = \frac{\vec{u} \times \vec{v}}{|\vec{u}| * |\vec{v}|}$$

- **Euclidian similarity / distance:**

$$d_2 = \sqrt{(v_1 - u_1)^2 + (v_2 - u_2)^2}$$



Vectors practically

- We may have some data saved in these vectors
 - \vec{v} = sample 1, \vec{u} = sample 2
- Quite often we want to **interpolate between samples**

$$\vec{w} = a * \vec{u} + (1 - a) * \vec{v}$$

Vectors practically

- We may have some data saved in these vectors
 - \vec{v} = sample 1, \vec{u} = sample 2
- Quite often we want to **interpolate between samples**

$$\vec{w} = a * \vec{u} + (1 - a) * \vec{v}$$

< Linear combination of vectors

Vectors practically

- We may have some data saved in these vectors
 - \vec{v} = sample 1, \vec{u} = sample 2
- Quite often we want to **interpolate between samples**

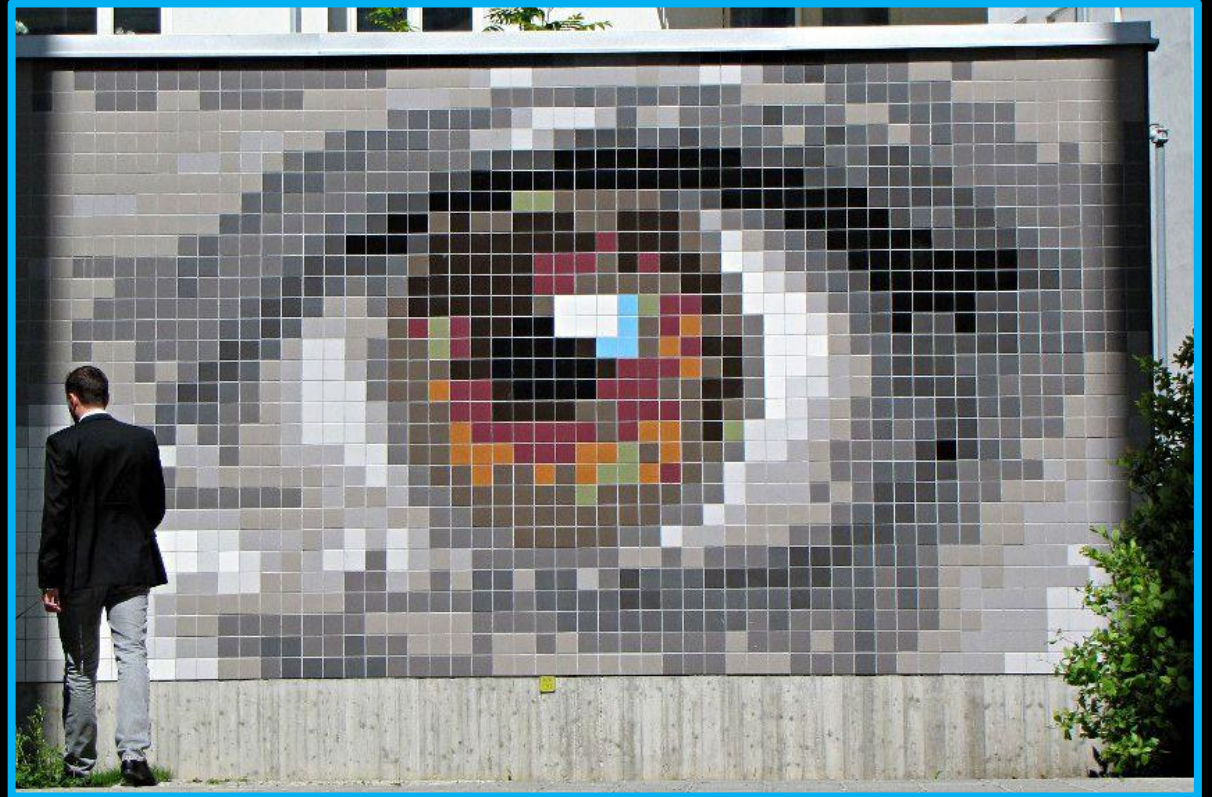
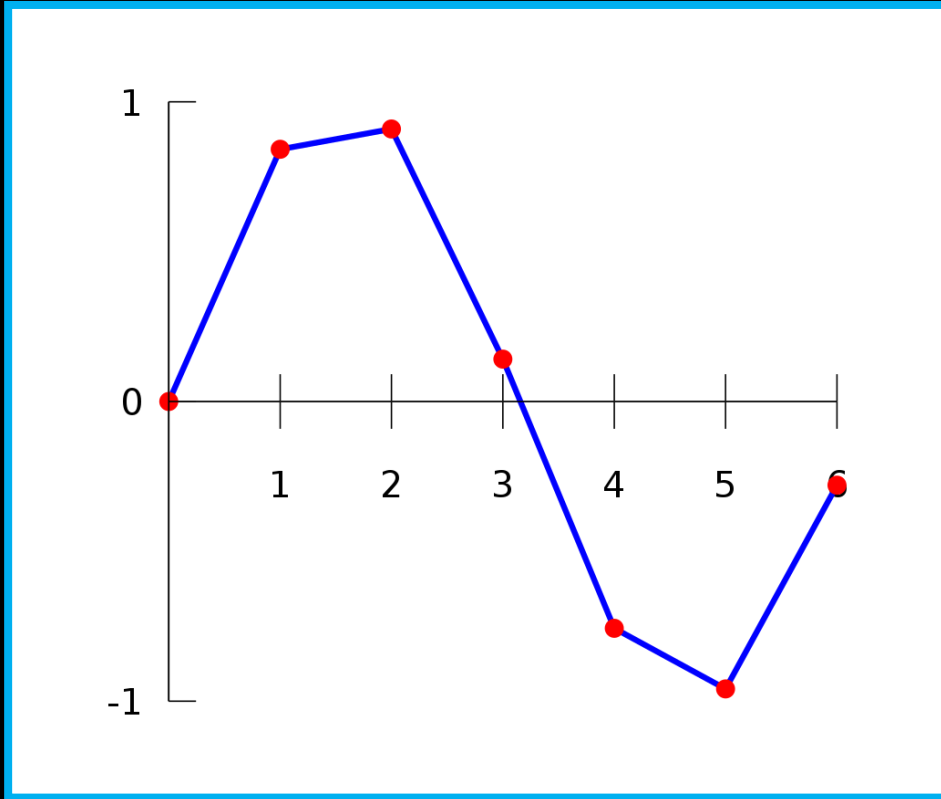
$$\vec{w} = a * \vec{u} + (1 - a) * \vec{v}$$

< Linear combination of vectors



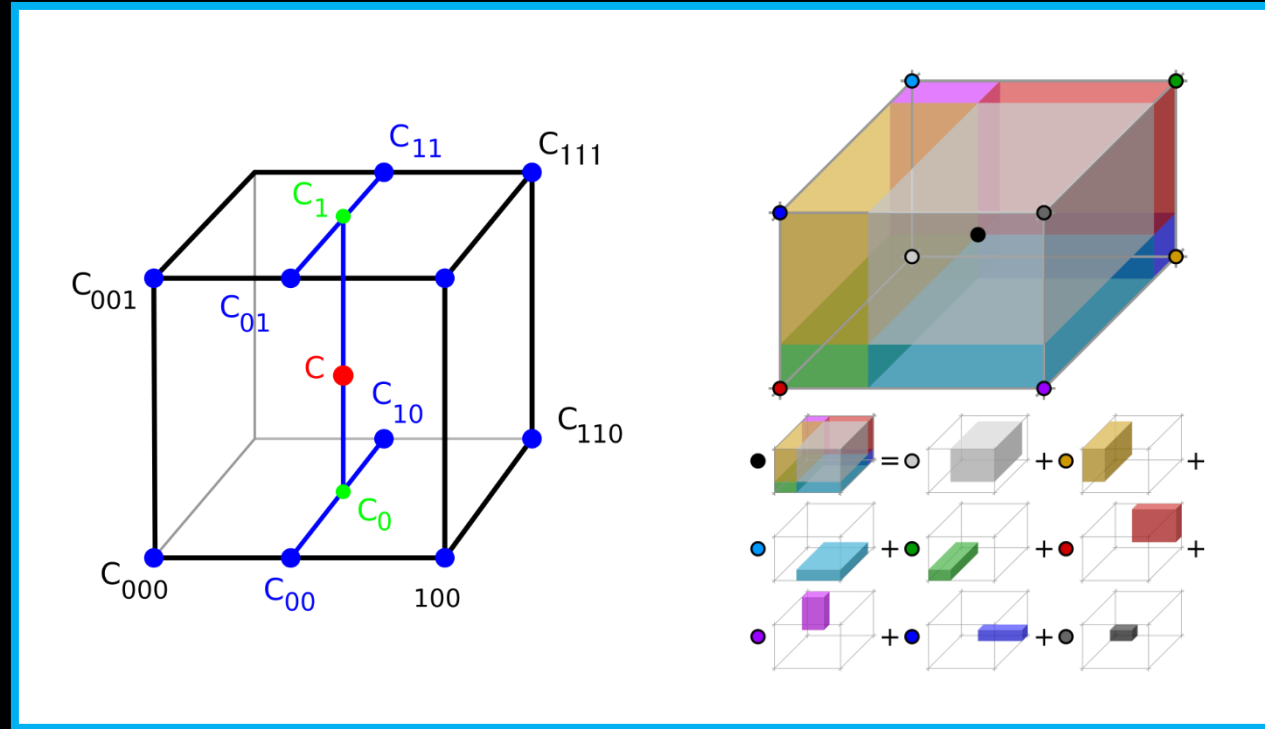
- *Ps: This is a special case where the vectors are used to generate images (we will get back to it in a demo at the end of class)*

Interpolation



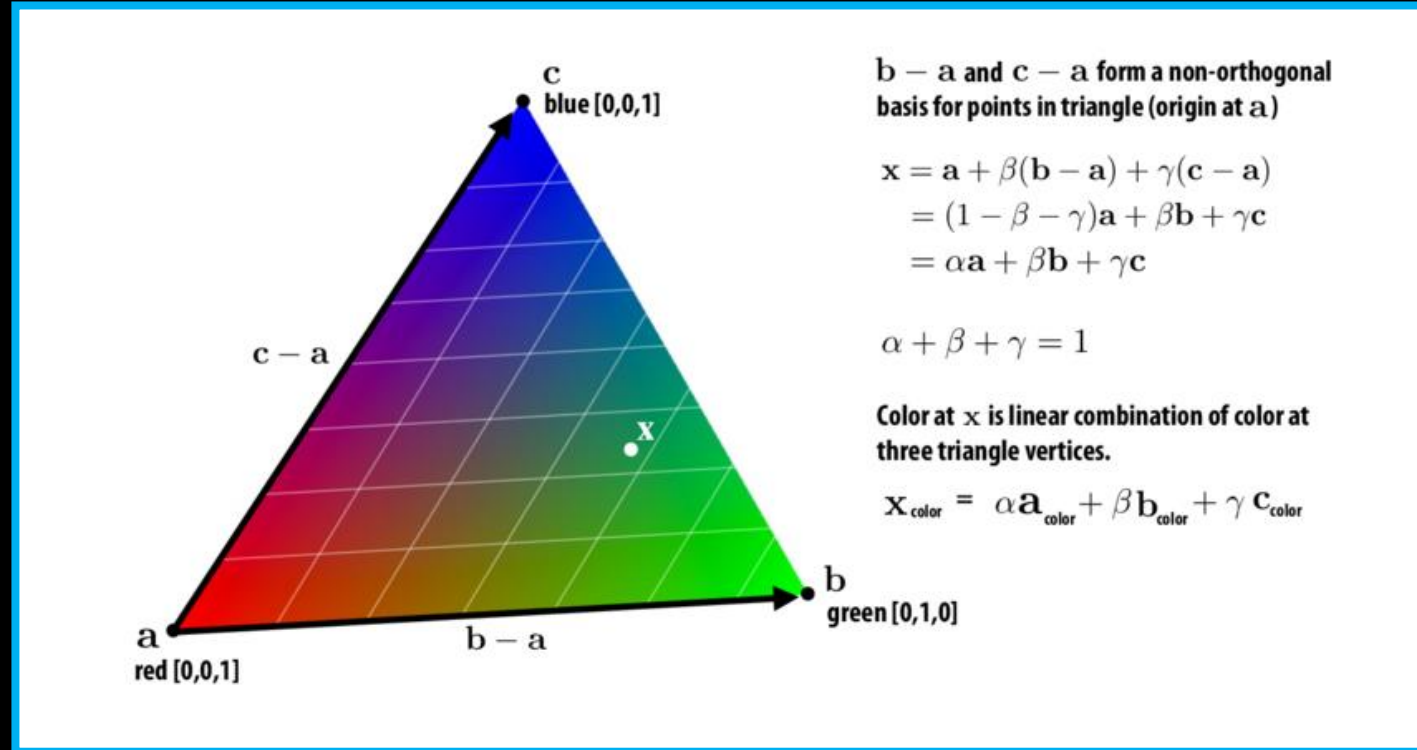
- We also may want to interpolate between data points on a plot.

Interpolation demos



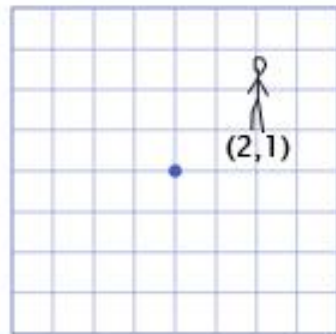
- Bilinear, Trilinear interpolation – interpolating between more than just two vectors
 - **Bilinear** demo: [geogebra.org/m/CTn3QkH9](https://www.geogebra.org/m/CTn3QkH9) – interpolations in a rectangle

Interpolation demos

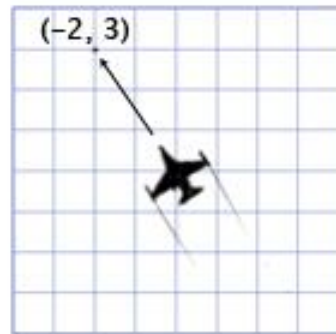


- Barycentric coordinates – interpolations **in a triangle** (*remember color gamuts?*)
 - **Barycentric** demo: [geogebra.org/m/rFQK2EH3#material/c8DwbVTP](https://www.geogebra.org/m/rFQK2EH3#material/c8DwbVTP)

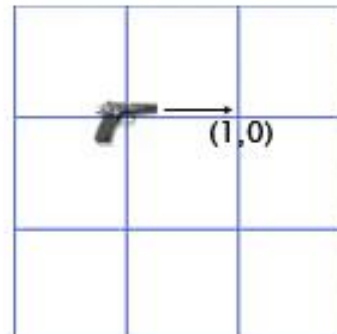
Vectors in games



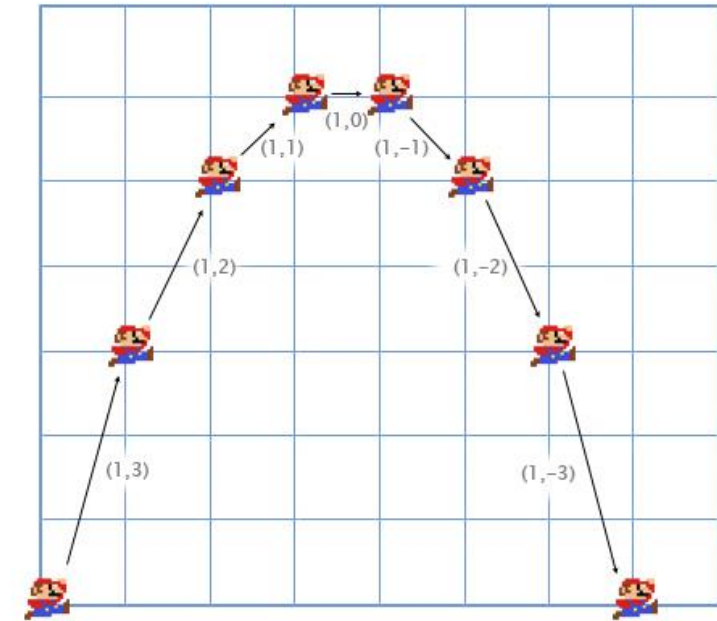
Position



Velocity



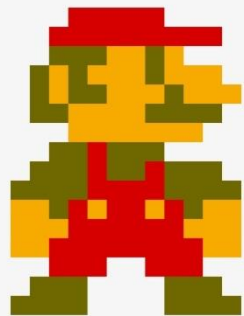
Direction



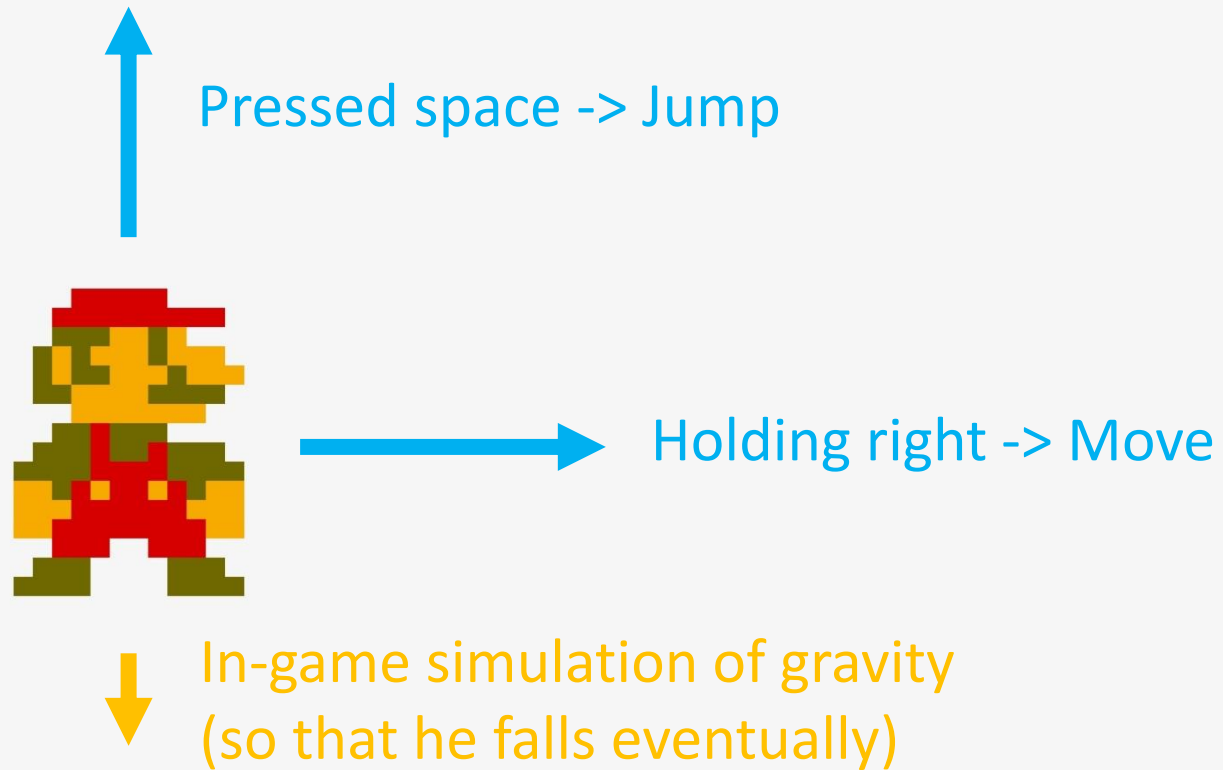
- **Simulating** gravitational **forces**, adding forces together, ...
- Checking **vision cones** (cosine metric to get an angle), **rotations** ...

Further read: [Blog](#)

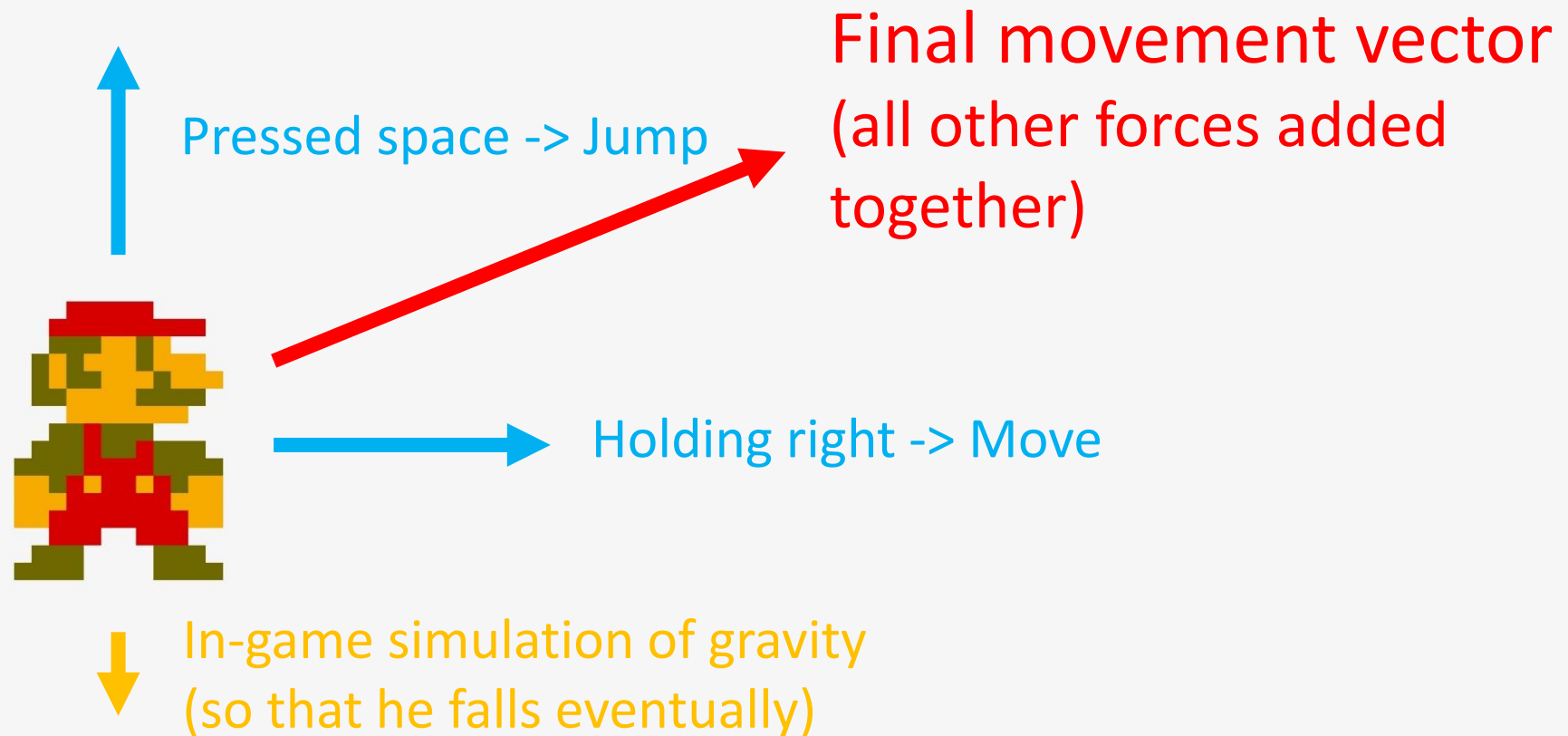
Vectors in games



Vectors in games



Vectors in games



Pause 1

Matrices

$$M = \begin{array}{c} \\ \\ m=3 \end{array} \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \begin{array}{c} n=3 \\ \\ \end{array}$$

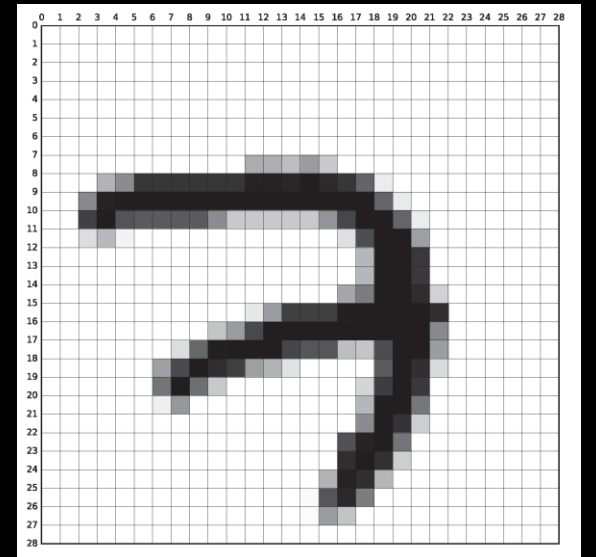
$$M \in \mathbb{R}^{m \times n}$$

Matrices

$$M = \begin{matrix} & & n=3 \\ & \begin{matrix} m_{1,1} & m_{1,2} & m_{1,3} \\ \dots & & \end{matrix} \\ m=3 & \begin{matrix} \\ \\ m_{3,3} \end{matrix} \end{matrix}$$

$$M \in \mathbb{R}^{m \times n}$$

- We can also consider loaded images as matrices:



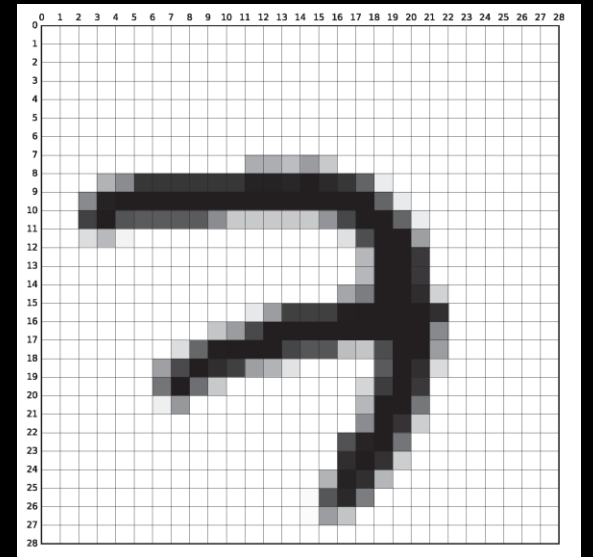
Matrices

$$M = \begin{matrix} & & n=3 \\ & \begin{matrix} m_{1,1} & m_{1,2} & m_{1,3} \\ \dots & & \end{matrix} \\ m=3 & \begin{matrix} \\ \\ m_{3,3} \end{matrix} \end{matrix}$$

< Indexing by row and column

$$M \in \mathbb{R}^{m \times n}$$

- We can also consider loaded images as matrices:



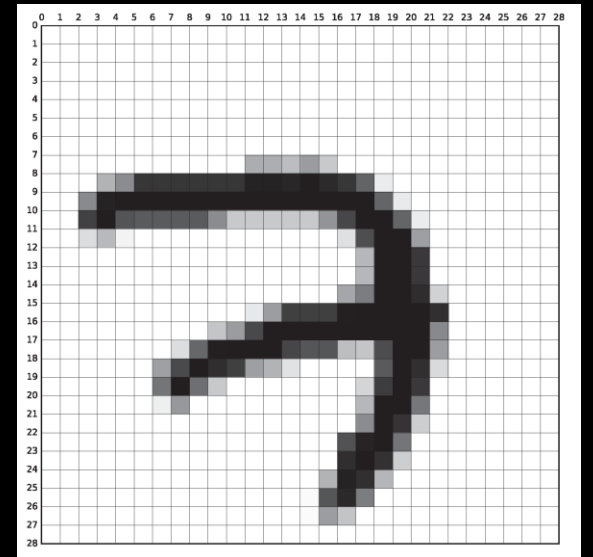
Matrices

$$\mathbf{M} = \begin{array}{c} \text{m=3} \\ \begin{array}{|c|c|c|} \hline m_1 & m_2 & m_3 \\ \hline m_4 & \dots & \\ \hline & \dots & m_9 \\ \hline \end{array} \\ \text{n=3} \end{array}$$

< Indexing as if we flattened it

$$\mathbf{M} \in \mathbb{R}^{m \times n}$$

- We can also consider loaded images as matrices:



Operations with matrices

- Multiplication by a real number

$$a * M$$

- $M \in \mathbb{R}^{m*n}$ (matrix)
- $a \in \mathbb{R}$ (a float number)

	n=3		
	$a * m_1$	$a * m_2$	$a * m_3$
m=3		...	$a * m_9$

Operations with matrices

Having two matrices of the same dimensionality:

- Addition

$$M + N$$

n=3		
m=3	$m_1 + n_1$	$m_2 + n_2$
	$m_3 + n_3$	
	...	$m_9 + n_9$

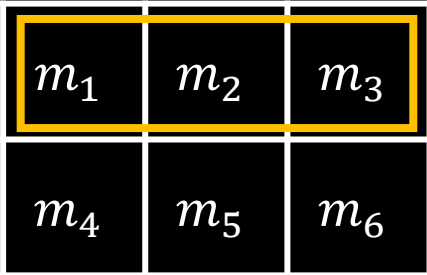
Operations with matrices

- Transpose M^T

$$M = \begin{array}{c} \begin{array}{ccc} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \end{array} \end{array}$$

$n=3$

$m=2$



Operations with matrices

- Transpose M^T

$$M = \begin{array}{|c|c|c|} \hline m_1 & m_2 & m_3 \\ \hline m_4 & m_5 & m_6 \\ \hline \end{array} \quad \begin{array}{l} n=3 \\ m=2 \end{array}$$

$$M^T = \begin{array}{|c|c|} \hline m_1 & m_4 \\ \hline m_2 & m_5 \\ \hline m_3 & m_6 \\ \hline \end{array} \quad \begin{array}{l} n=2 \\ m=3 \end{array}$$

< Just rotated and flipped

Matrix-matrix multiply

- Corresponds to a transformation
- Dimensions must match:

$$\begin{array}{ccccc} A & * & B & = & C \\ m * n & & n * p & & m * p \end{array}$$

Matrix-matrix multiply

- Corresponds to a transformation
- Dimensions must match:

$$\begin{array}{ccc} \mathbf{A} & * & \mathbf{B} = \mathbf{C} \\ m * n & & n * p \quad m * p \end{array}$$

$$\mathbf{A} = \begin{bmatrix} \text{---} & \text{row 1} & \rightarrow \\ \text{---} & \text{row 2} & \rightarrow \\ & \dots & \\ \text{---} & \text{row } m & \rightarrow \end{bmatrix} = \begin{bmatrix} \text{---} & \mathbf{r}_1 & \rightarrow \\ \text{---} & \mathbf{r}_2 & \rightarrow \\ & \dots & \\ \text{---} & \mathbf{r}_m & \rightarrow \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} | & | & \vdots & | \\ \text{col 1} & \text{col 2} & \vdots & \text{col } p \\ \downarrow & \downarrow & \downarrow & \downarrow \end{bmatrix} = \begin{bmatrix} | & | & \vdots & | \\ \mathbf{c}_1 & \mathbf{c}_2 & \vdots & \mathbf{c}_p \\ \downarrow & \downarrow & \downarrow & \downarrow \end{bmatrix}$$

< We look at separated
rows and columns ...

BTW: these are vectors!

$$\mathbf{A} * \mathbf{B} = \mathbf{C}$$

$m * n$

$n * p$

$m * p$

a_1	a_2	a_3
a_4	a_5	a_6

$*$

b_1	b_2	b_3
b_4	...	
	...	b_9

$=$

2 rows x 3 columns

3 rows x 3 columns

$m = 2, n = 3$

$p = 3$

$$A * B = C$$

$m * n$

$n * p$

$m * p$

a_1	a_2	a_3
a_4	a_5	a_6

$*$

b_1	b_2	b_3
b_4	...	
	...	b_9

$=$

2 rows x 3 columns

3 rows x 3 columns

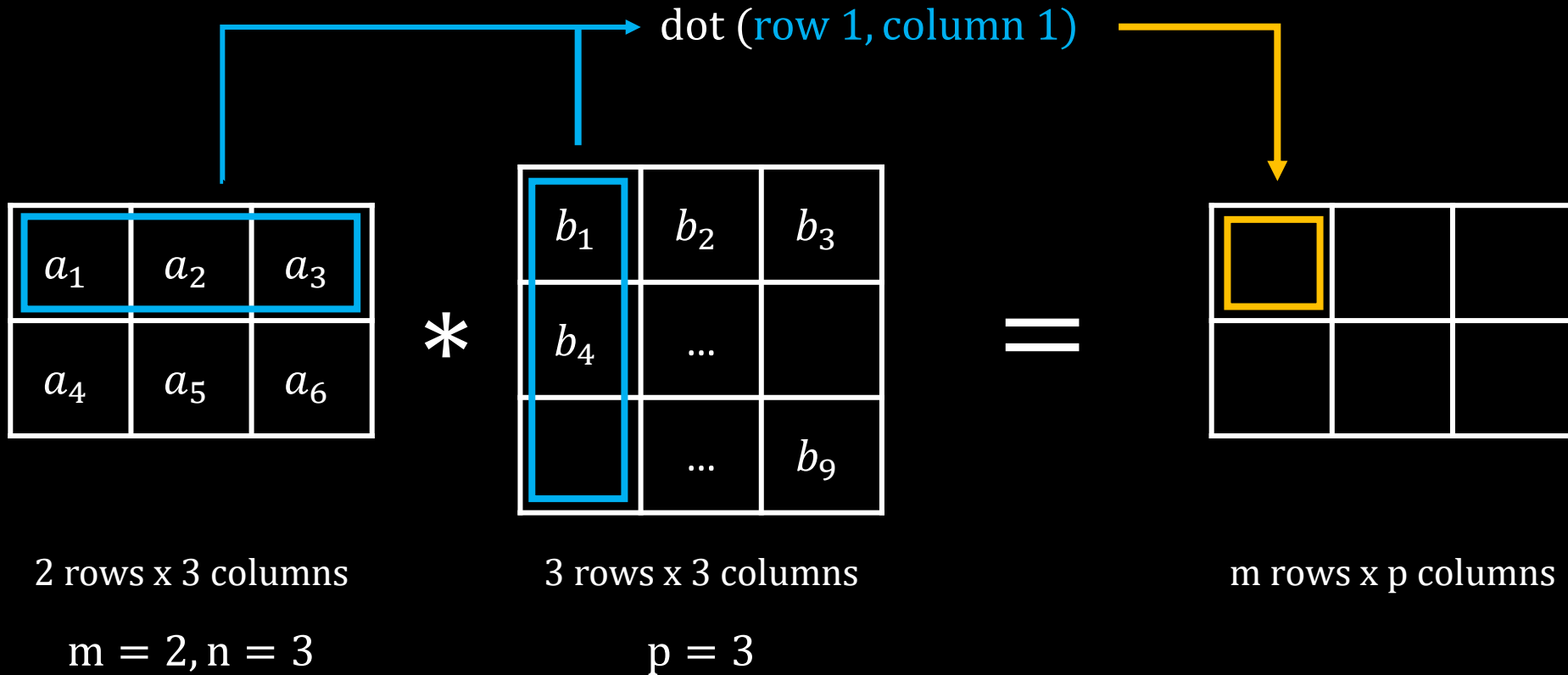
m rows x p columns

$m = 2, n = 3$

$p = 3$

$$\begin{array}{ccc}
 \mathbf{A} & * & \mathbf{B} = \mathbf{C} \\
 m * n & & n * p \quad m * p
 \end{array}$$

$$\mathbf{A} = \begin{bmatrix} \text{row 1} \rightarrow \\ \text{row 2} \rightarrow \\ \vdots \\ \text{row } m \rightarrow \end{bmatrix} = \begin{bmatrix} \text{r}_1 \rightarrow \\ \text{r}_2 \rightarrow \\ \vdots \\ \text{r}_m \rightarrow \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \text{col 1} & \text{col 2} & \vdots & \text{col } p \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} = \begin{bmatrix} \text{c}_1 & \text{c}_2 & \vdots & \text{c}_p \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}$$



$$\begin{matrix}
 \mathbf{A} & * & \mathbf{B} & = & \mathbf{C} \\
 m * n & & n * p & & m * p
 \end{matrix}$$

$$\mathbf{A} = \begin{bmatrix} \text{row 1} \rightarrow \\ \text{row 2} \rightarrow \\ \vdots \\ \text{row } m \rightarrow \end{bmatrix} = \begin{bmatrix} \text{r}_1 \rightarrow \\ \text{r}_2 \rightarrow \\ \vdots \\ \text{r}_m \rightarrow \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} | & | & \vdots & | \\ \text{col 1} & \text{col 2} & & \text{col } p \\ | & | & \vdots & | \end{bmatrix} = \begin{bmatrix} | & | & \vdots & | \\ \mathbf{c}_1 & \mathbf{c}_2 & & \mathbf{c}_p \\ | & | & \vdots & | \end{bmatrix}$$

$$\text{dot}(\text{row 1, column 1}) = (1*1 + 0*-2 + -3*0) = 1$$

1	0	-3
-2	4	1

*

1	0	4
-2	3	-1
0	-1	2

=

1		

2 rows x 3 columns

$m = 2, n = 3$

3 rows x 3 columns

$p = 3$

m rows x p columns

$$\begin{matrix}
 \mathbf{A} & * & \mathbf{B} & = & \mathbf{C} \\
 m * n & & n * p & & m * p
 \end{matrix}$$

$$\mathbf{A} = \begin{bmatrix} \text{row 1} \rightarrow \\ \text{row 2} \rightarrow \\ \vdots \\ \text{row } m \rightarrow \end{bmatrix} = \begin{bmatrix} \text{r}_1 \rightarrow \\ \text{r}_2 \rightarrow \\ \vdots \\ \text{r}_m \rightarrow \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} | & | & \vdots & | \\ \text{col 1} & \text{col 2} & & \text{col } p \\ | & | & \vdots & | \end{bmatrix} = \begin{bmatrix} | & | & \vdots & | \\ \mathbf{c}_1 & \mathbf{c}_2 & & \mathbf{c}_p \\ | & | & \vdots & | \end{bmatrix}$$

$$\text{dot}(\text{row 1, column 2}) = (1*0 + 0*3 + -3*-1) = 3$$

1	0	-3
-2	4	1

*

1	0	4
-2	3	-1
0	-1	2

=

1	3	

2 rows x 3 columns

$m = 2, n = 3$

3 rows x 3 columns

$p = 3$

m rows x p columns

$$\begin{matrix}
 \mathbf{A} & * & \mathbf{B} & = & \mathbf{C} \\
 m * n & & n * p & & m * p
 \end{matrix}$$

$$\mathbf{A} = \begin{bmatrix} \text{row 1} \rightarrow \\ \text{row 2} \rightarrow \\ \vdots \\ \text{row } m \rightarrow \end{bmatrix} = \begin{bmatrix} \text{r}_1 \rightarrow \\ \text{r}_2 \rightarrow \\ \vdots \\ \text{r}_m \rightarrow \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} | & | & \vdots & | \\ \text{col 1} & \text{col 2} & & \text{col } p \\ | & | & \vdots & | \end{bmatrix} = \begin{bmatrix} | & | & \vdots & | \\ \mathbf{c}_1 & \mathbf{c}_2 & & \mathbf{c}_p \\ | & | & \vdots & | \end{bmatrix}$$

$$\text{dot}(\text{row 1, column 3}) = (1*4 + 0*-1 + -3*2) = -2$$

1	0	-3
-2	4	1

*

1	0	4
-2	3	-1
0	-1	2

=

1	3	-2

2 rows x 3 columns

$m = 2, n = 3$

3 rows x 3 columns

$p = 3$

m rows x p columns

$$\begin{matrix}
 \mathbf{A} & * & \mathbf{B} & = & \mathbf{C} \\
 m * n & & n * p & & m * p
 \end{matrix}$$

$$\mathbf{A} = \begin{bmatrix} \text{row 1} \rightarrow \\ \text{row 2} \rightarrow \\ \vdots \\ \text{row } m \rightarrow \end{bmatrix} = \begin{bmatrix} \text{r}_1 \rightarrow \\ \text{r}_2 \rightarrow \\ \vdots \\ \text{r}_m \rightarrow \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} | & | & \vdots & | \\ \text{col 1} & \text{col 2} & & \text{col } p \\ | & | & \vdots & | \end{bmatrix} = \begin{bmatrix} | & | & \vdots & | \\ \mathbf{c}_1 & \mathbf{c}_2 & & \mathbf{c}_p \\ | & | & \vdots & | \end{bmatrix}$$

$$\text{dot}(\text{row 2, column 1}) = (-2 * 1 + 4 * -2 + 1 * 0) = -10$$

1	0	-3
-2	4	1

*

1	0	4
-2	3	-1
0	-1	2

=

1	3	-2
-10		

2 rows x 3 columns

$m = 2, n = 3$

3 rows x 3 columns

$p = 3$

m rows x p columns

$$\begin{matrix}
 \mathbf{A} & * & \mathbf{B} & = & \mathbf{C} \\
 m * n & & n * p & & m * p
 \end{matrix}$$

$$\mathbf{A} = \begin{bmatrix} \text{row 1} \rightarrow \\ \text{row 2} \rightarrow \\ \vdots \\ \text{row } m \rightarrow \end{bmatrix} = \begin{bmatrix} \text{r}_1 \rightarrow \\ \text{r}_2 \rightarrow \\ \vdots \\ \text{r}_m \rightarrow \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} | & | & \vdots & | \\ \text{col 1} & \text{col 2} & \vdots & \text{col } p \\ | & | & \vdots & | \end{bmatrix} = \begin{bmatrix} | & | & \vdots & | \\ \mathbf{c}_1 & \mathbf{c}_2 & \vdots & \mathbf{c}_p \\ | & | & \vdots & | \end{bmatrix}$$

$$\text{dot}(\text{row 2, column 1}) = (-2 * 1 + 4 * -2 + 1 * 0) = -10$$

1	0	-3
-2	4	1

*

1	0	4
-2	3	-1
0	-1	2

=

1	3	-2
-10		

... ETC!

2 rows x 3 columns

$m = 2, n = 3$

3 rows x 3 columns

$p = 3$

m rows x p columns

Matrix-matrix multiply

Practically ... **why do we care** about *matrix-matrix multiply*?

Matrix-matrix multiply

Practically ... **why do we care** about *matrix-matrix multiply*?

- Matrices can have encoded data (images, points, 3D coordinates, ...)
- Multiplication by another matrix can serve as a **transformation operation**

Matrix-matrix multiply

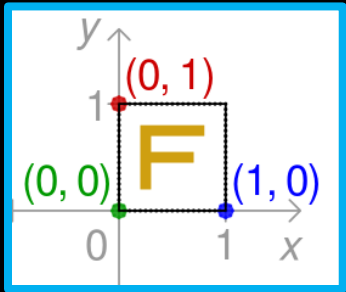
Practically ... **why do we care** about *matrix-matrix multiply*?

- Matrices can have encoded data (images, points, 3D coordinates, ...)
- Multiplication by another matrix can serve as a **transformation operation**:

$$\begin{matrix} A_{\text{data}} & * & T & = & A_{\text{transformedData}} \\ m * n & & n * p & & m * p \end{matrix}$$

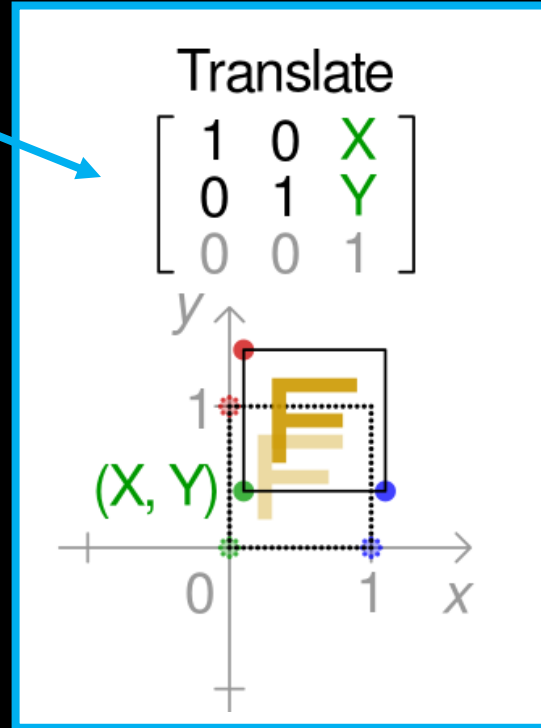
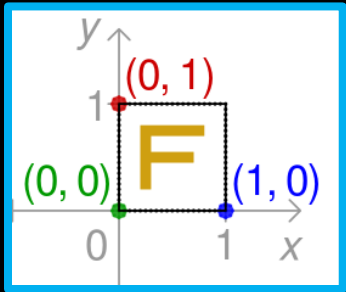
Transformation Matrix

$$A_{\text{data}} * T$$



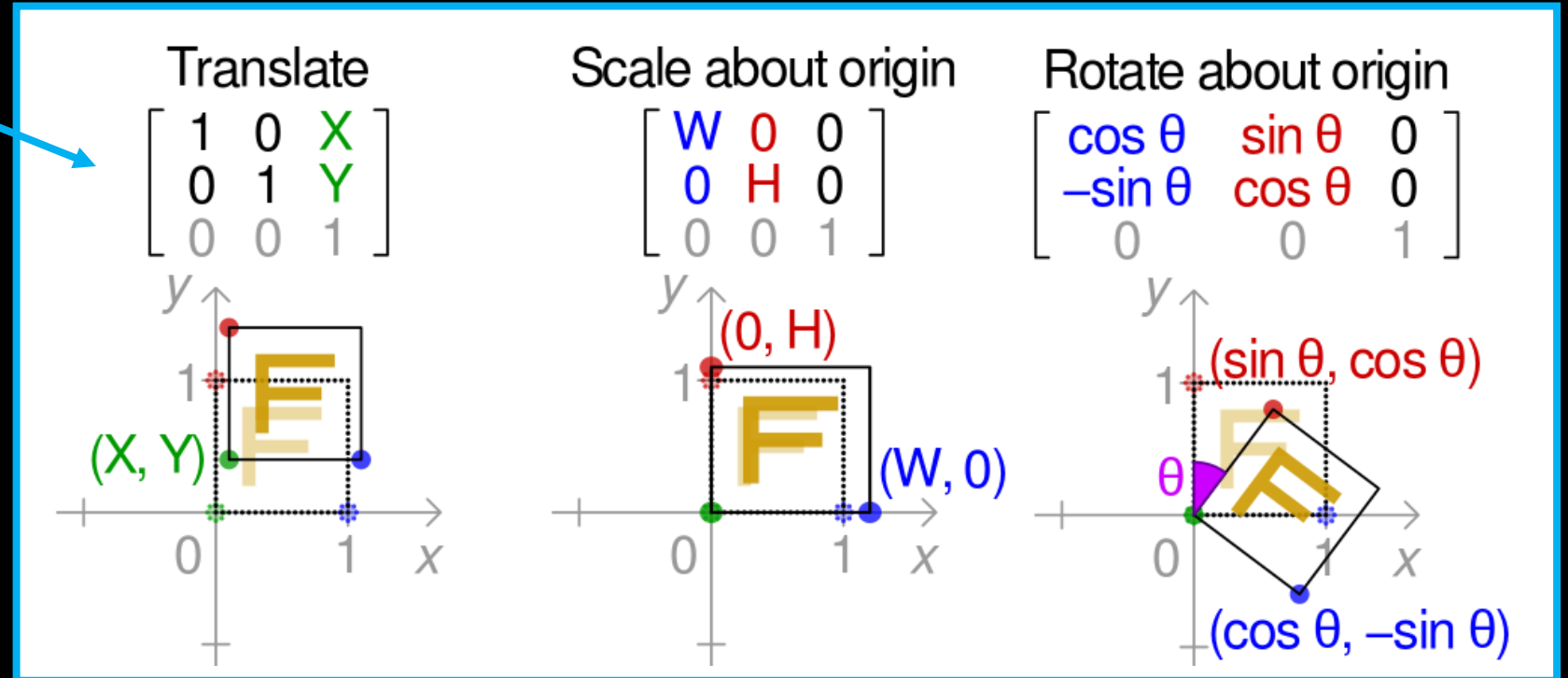
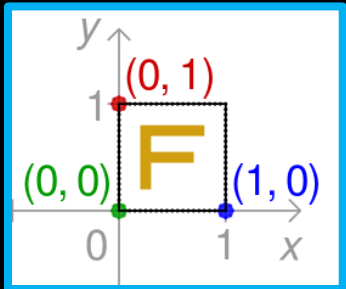
Transformation Matrix

$$A_{\text{data}} * T$$



Transformation Matrix

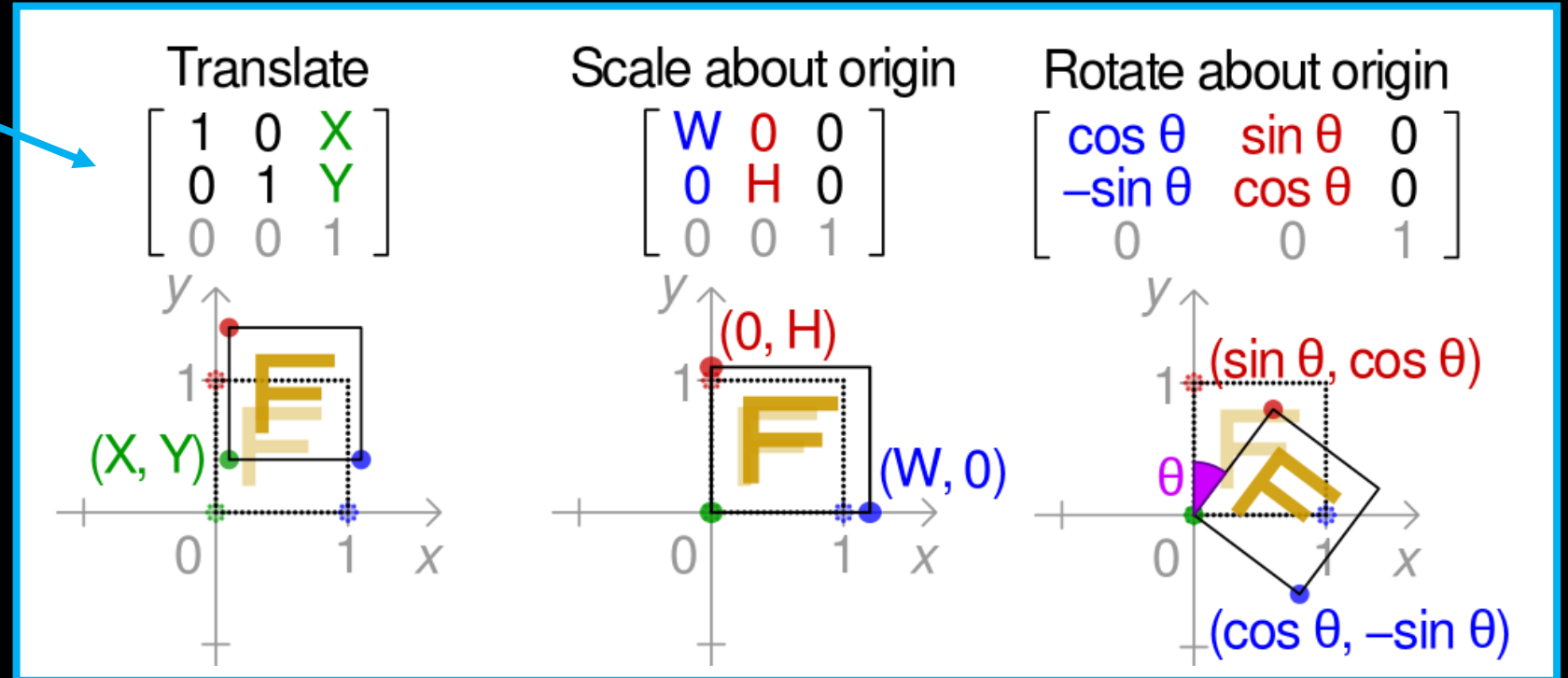
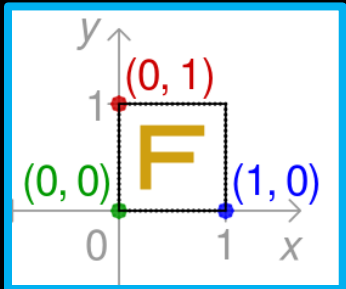
$$A_{\text{data}} * T$$



- And many other operations too (*translation, rotation, scale, ...*)
- We often use this in Computer Graphics applications (Games)

Transformation Matrix

$$A_{\text{data}} * T$$

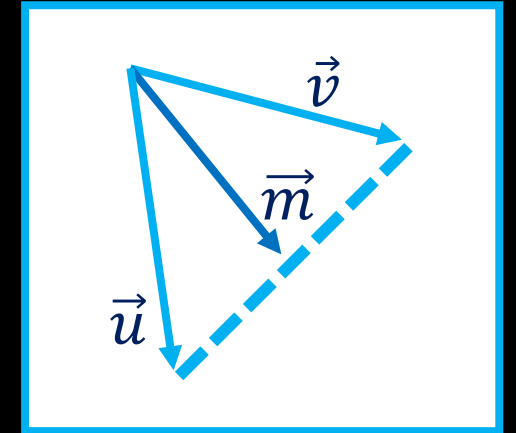


- And many other operations too (*translation, rotation, scale, ...*)
- We often use this in Computer Graphics applications (Games)
- **Matrix-Matrix Multiply is very fast on GPUs**

Pause 2

Programming task

- **Vector manipulation in Python**



- Starter code with tasks:
 - [week11_vectors-matrices/w11_vectors_matrices_tasks.ipynb](#)

Bonus!

- Let's explore a **Project ArtBreeder** (*GAN Breeder*)
 - Behind it, there is a machine learning model, which can generate an image if you give it a vector of 512 numbers
 - $\vec{v} = (v_1, v_2, \dots, v_{512}) \rightarrow$ Generate an image #1
 - $\vec{u} = (u_1, u_2, \dots, u_{512}) \rightarrow$ Generate an image #2
 - Using linear interpolation between vectors, they can mix them and **generate images somewhere in between!**

Project ArtBreeder – artbreeder.com/

Portraits



Album Covers



Landscapes



Anime Portraits



Additional readings?

- About **Vectors**:
 - On Khan Academy with some interactive demos: khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-vectors/a/intro-to-vectors
 - More on the math side: math10.com/en/geometry/vectors-operations/vectors-operations.html
- About **Matrices**: cliffsnotes.com/study-guides/algebra/linear-algebra/matrix-algebra/operations-with-matrices
- Bonus: in Nature of Code natureofcode.com/book/chapter-1-vectors/

The End