Data, Math and Methods

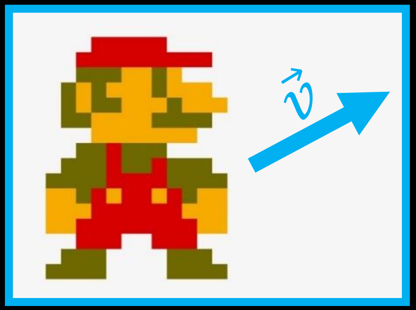
Week 11, Vectors & Matrices

~ Vítek Růžička

Today

Vectors and matrices – Linear Algebra

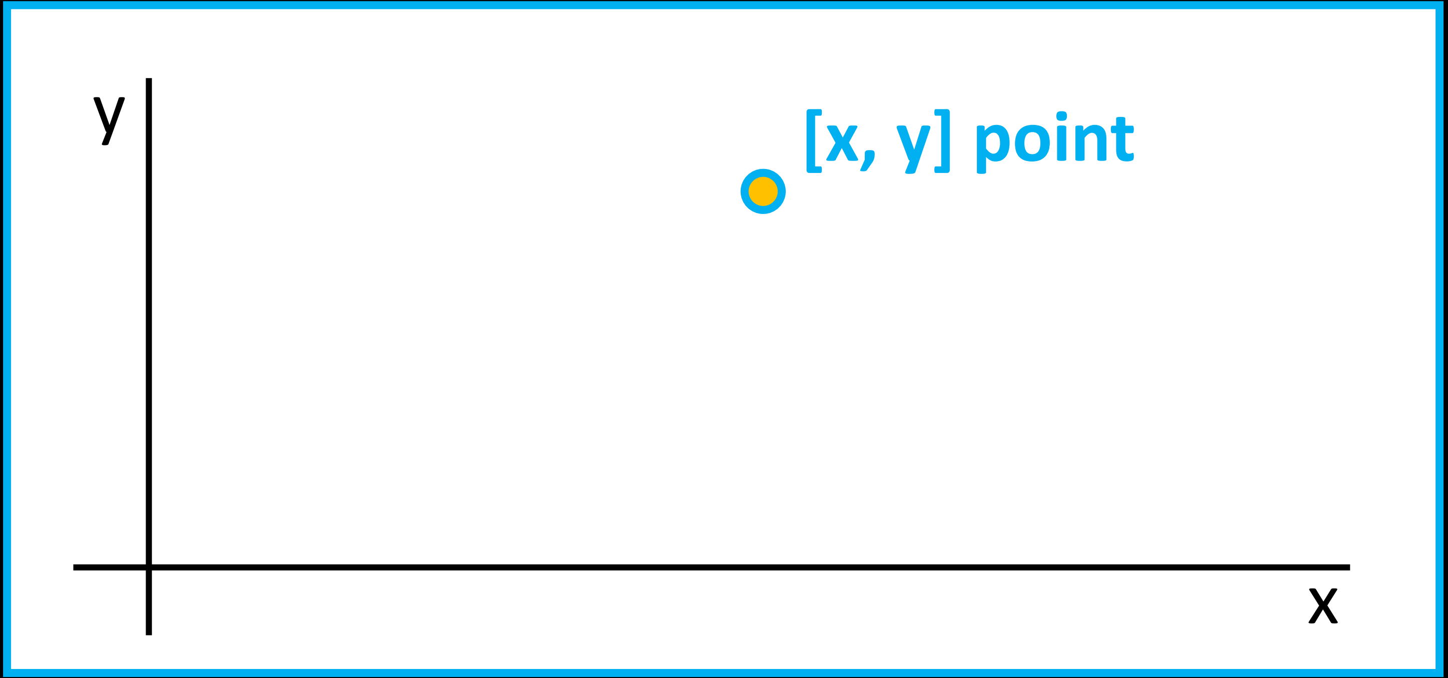
- **Vector** operations in math and programming
- **Matrix** operations and their uses in Computer Sciences



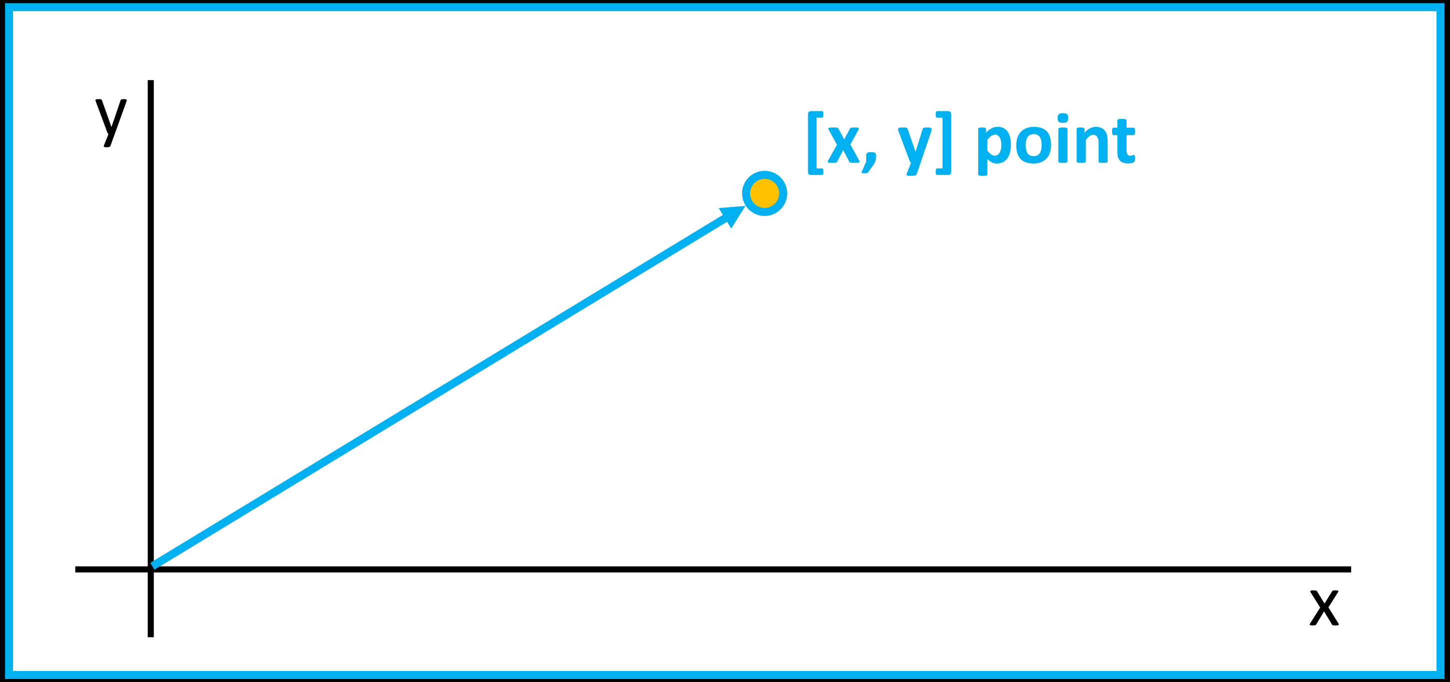
Practical session

- Vector manipulation in Python

Vectors



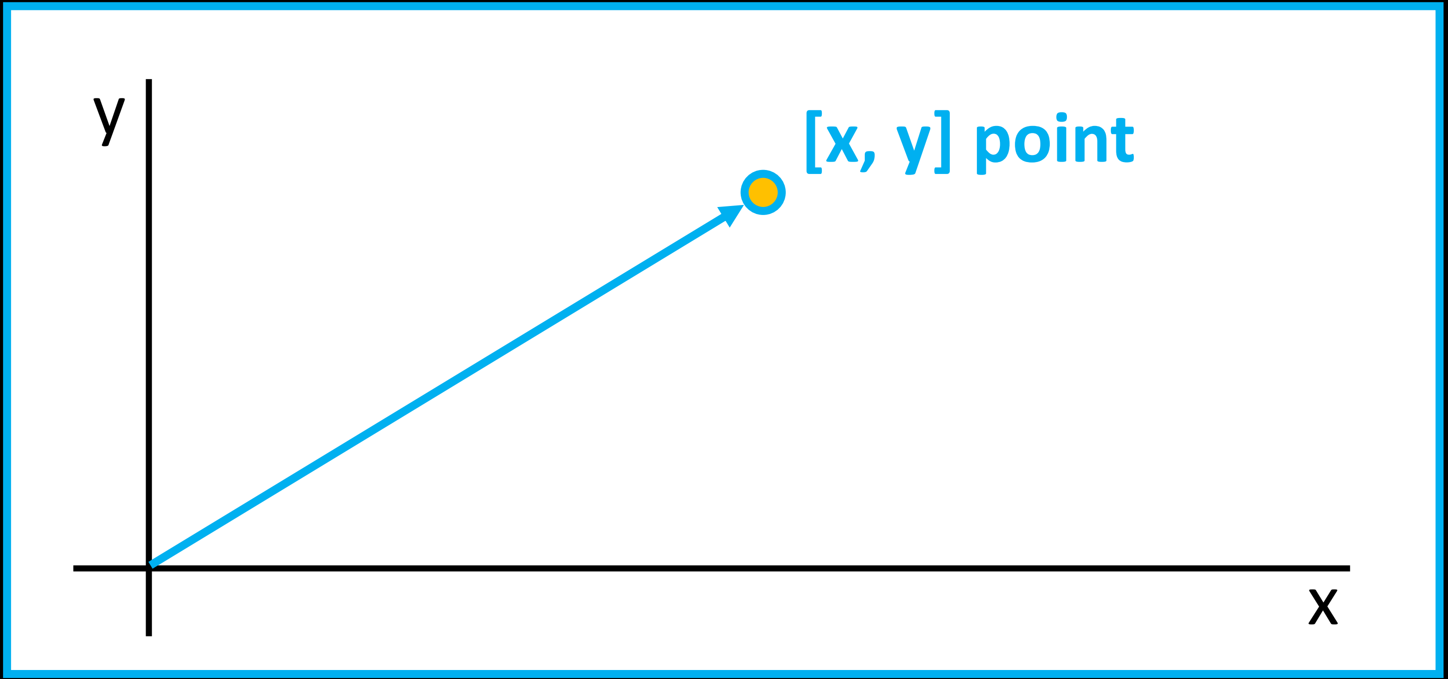
Vectors



(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 $(x_1, x_2, x_3, \ldots, x_n)$ -> dimensionality = **n**
- In **programming** we have a similar structure:
  - array = $[x_1, x_2, x_3, \ldots, x_n]$ or in python numpy: np.asarray($[x_1, x_2, x_3, \ldots, x_n]$)

Operations with Vectors

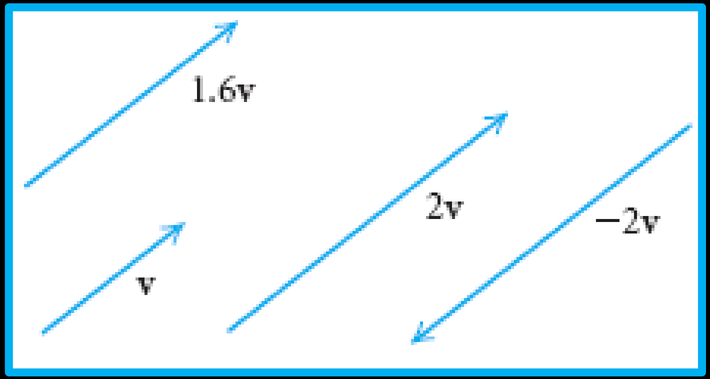
What can we do with vectors?

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

- <u>Multiplication by a real number:</u>

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

  $a \in \mathbb{R}$

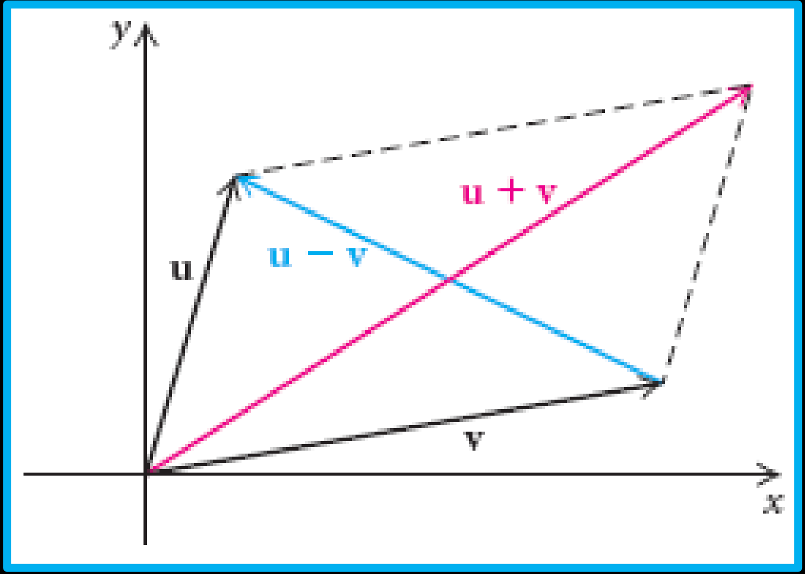


Operations with Vectors

Given two vectors of the same dimensionality …

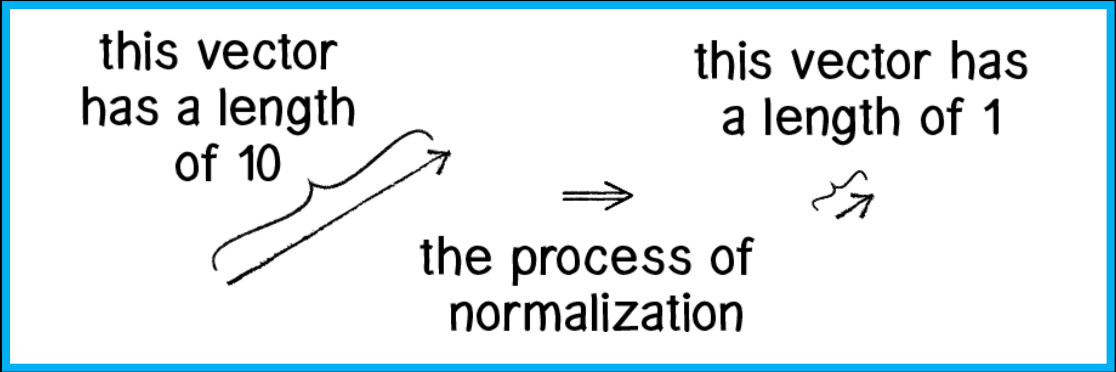
- $\vec{v}$ = (v_1, v_2)
- $\vec{u}$ = (u_1, u_2)

- <u>Addition / Subtraction</u>
  $\vec{u} + \vec{v}$ $(u_1 + v_1, u_2 + v_2)$
  $\vec{u} - \vec{v}$ $(u_1 - v_1, u_2 - v_2)$



Operations with Vectors

Normalization:



- <u>Length of the vector:</u>

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

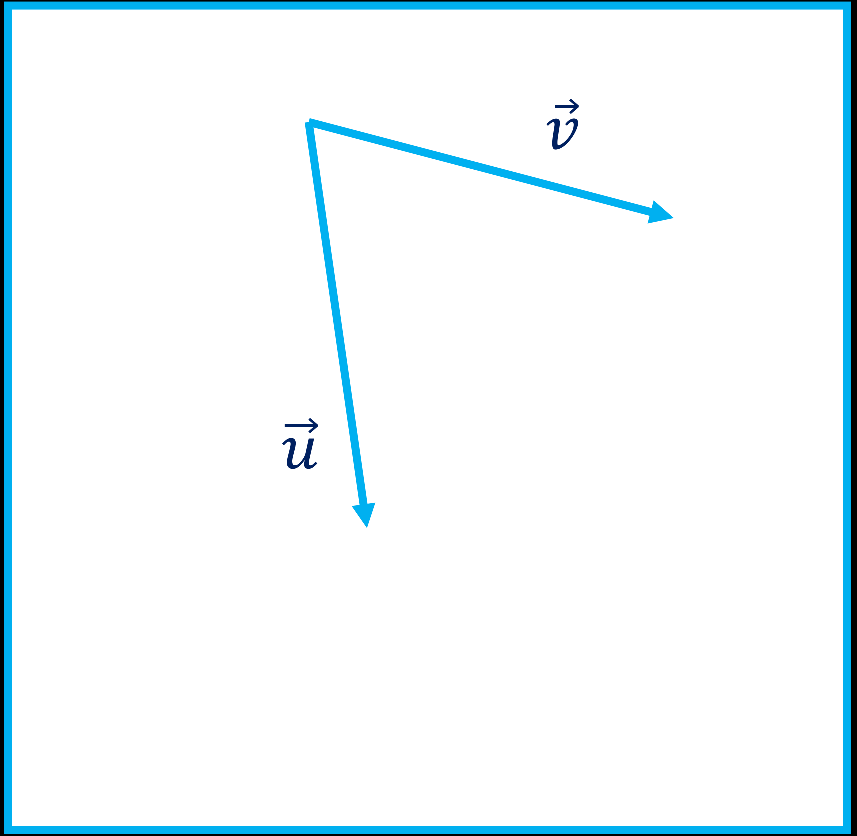
- <u>Normalization:</u>

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

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

Linear combination of vectors

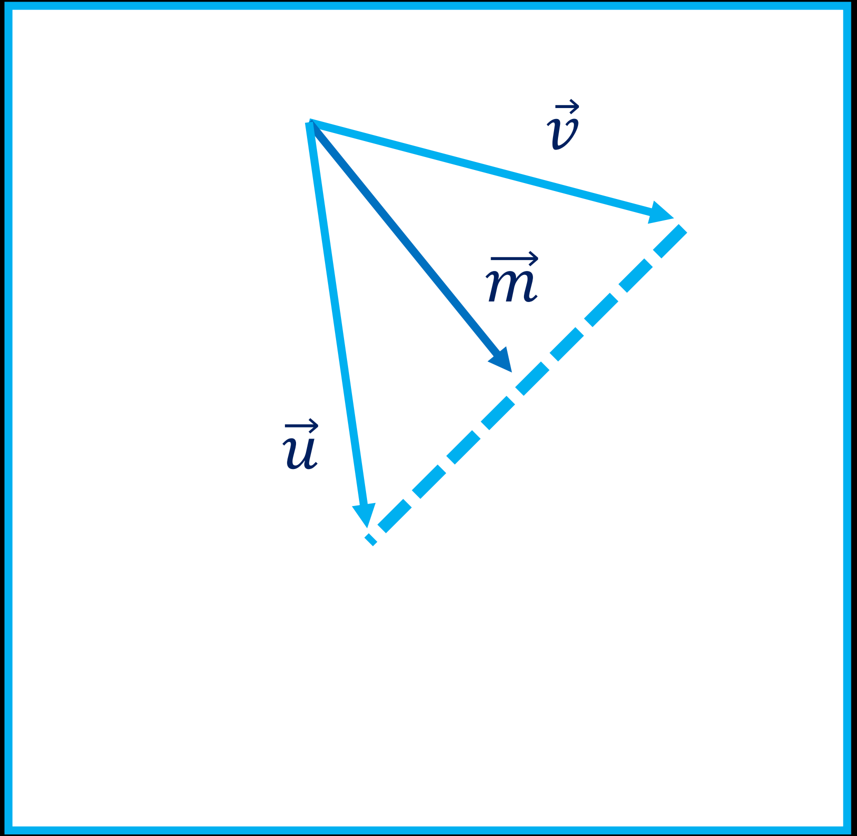
- Method of **combining vectors together** …



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!

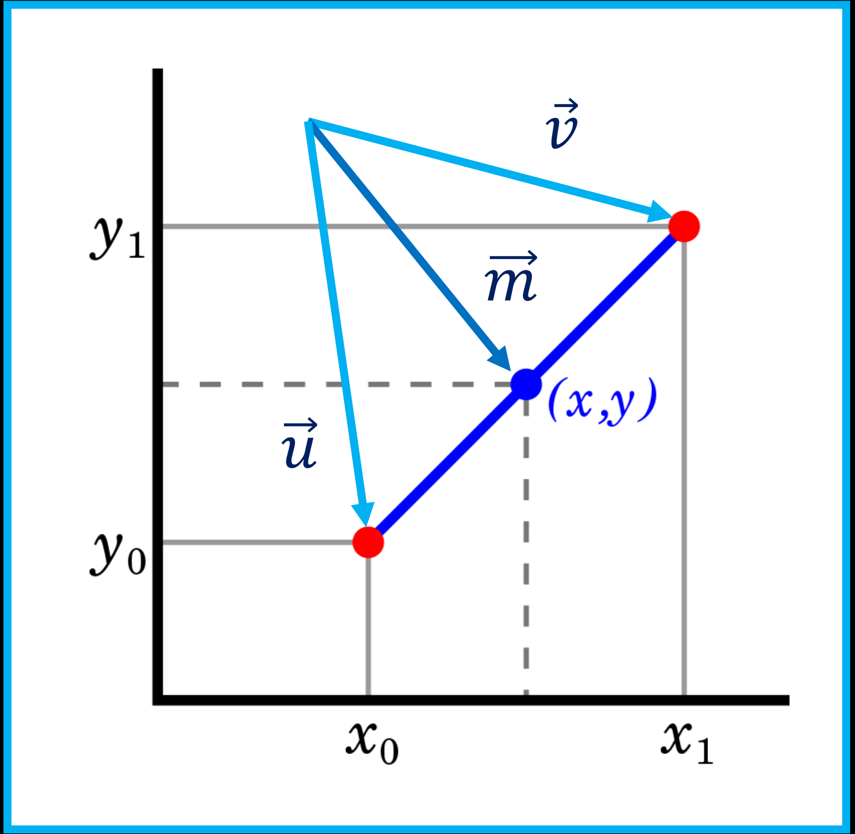
$$\vec{m} = a * \vec{u} + b * \vec{v}$$
$$a + b = 1$$



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

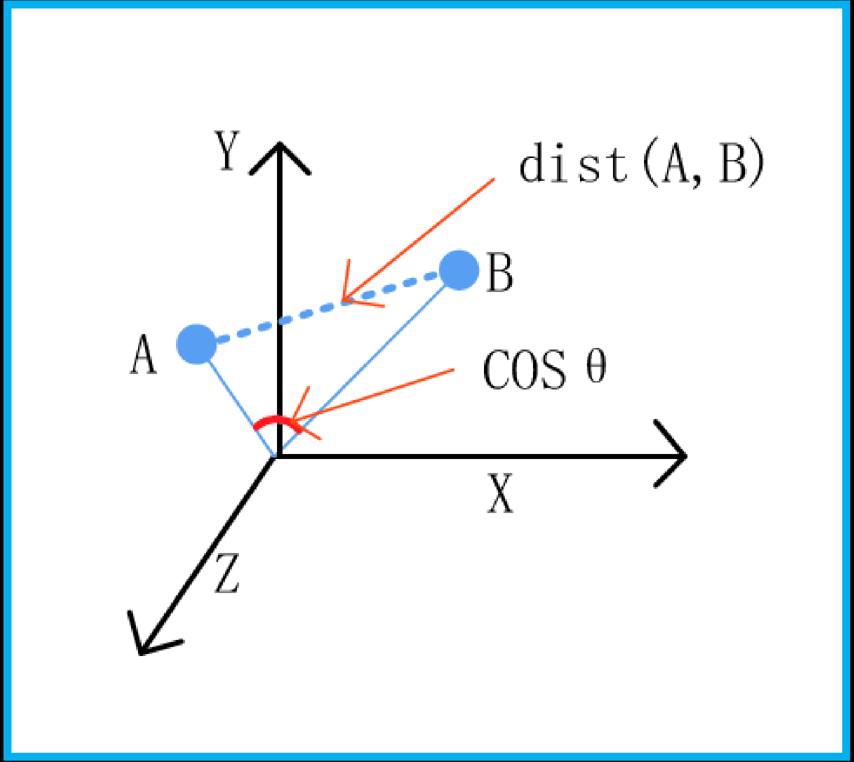
Angles between vectors

- **Dot product** = per element multiplication

$$\vec{u} \times \vec{v} = (u_1 * v_1, u_2 * v_2)$$

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

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



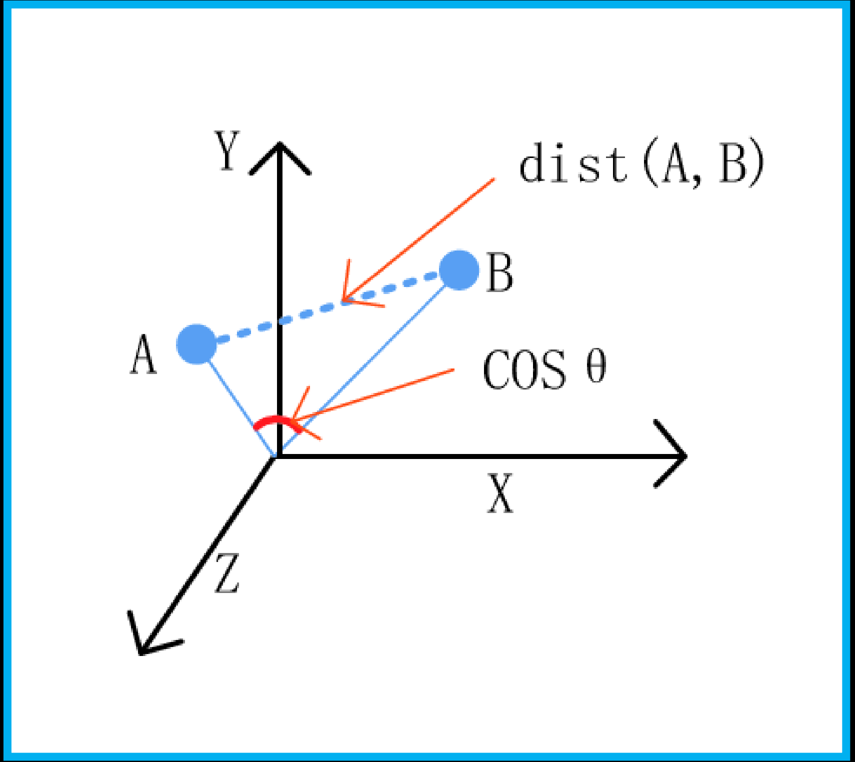
Angles between vectors

- **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

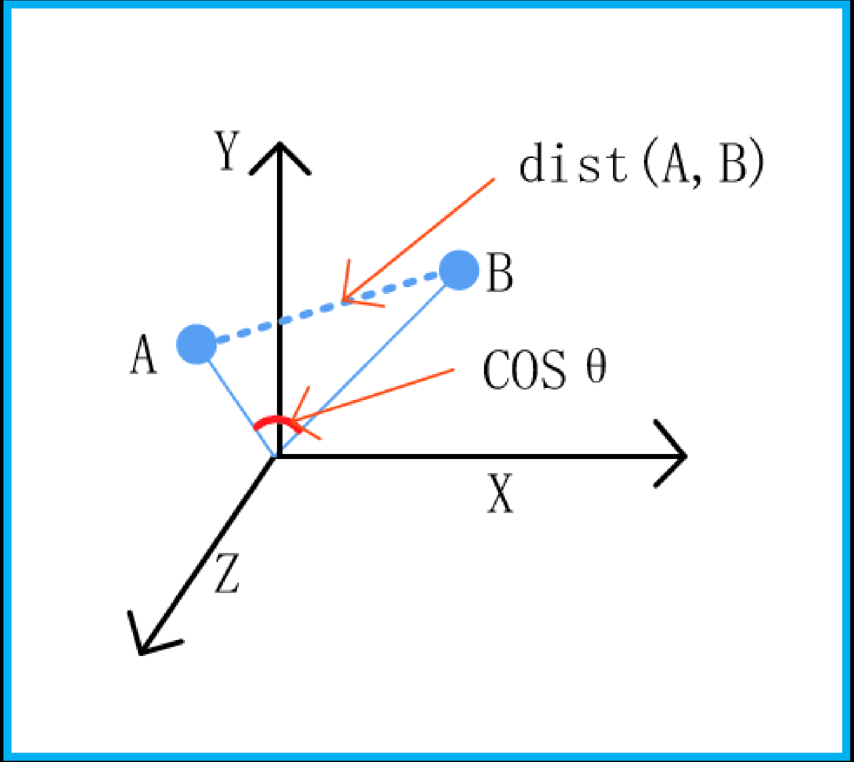
- Metrics of distance describe for us some kind of similarity between vectors …
  - Small distance -> very similar vectors
  - Large distance -> very different vectors
- *PS: There are many types of metrics!*

- **Cosine similarity / distance:**

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

- **Euclidian similarity / distance:**

$$\mathrm{d}_2 = \sqrt{(v_1 - u_1)^2 + (v_2 - u_2)^2}$$



Vectors practically

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

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

< Linear combination of vectors

Vectors practically

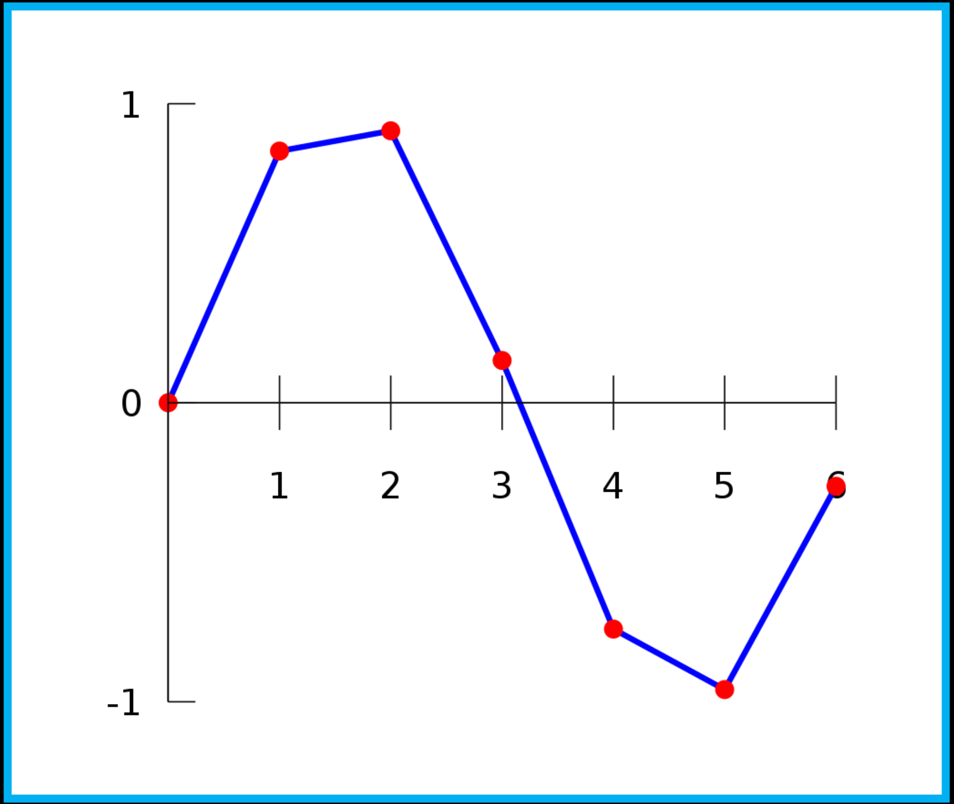
- We may have some data saved in these vectors
  - $\vec{v} = \text{sample 1}, \ \vec{u} = \text{sample 2}$
- Quite often we want to **interpolate between samples**

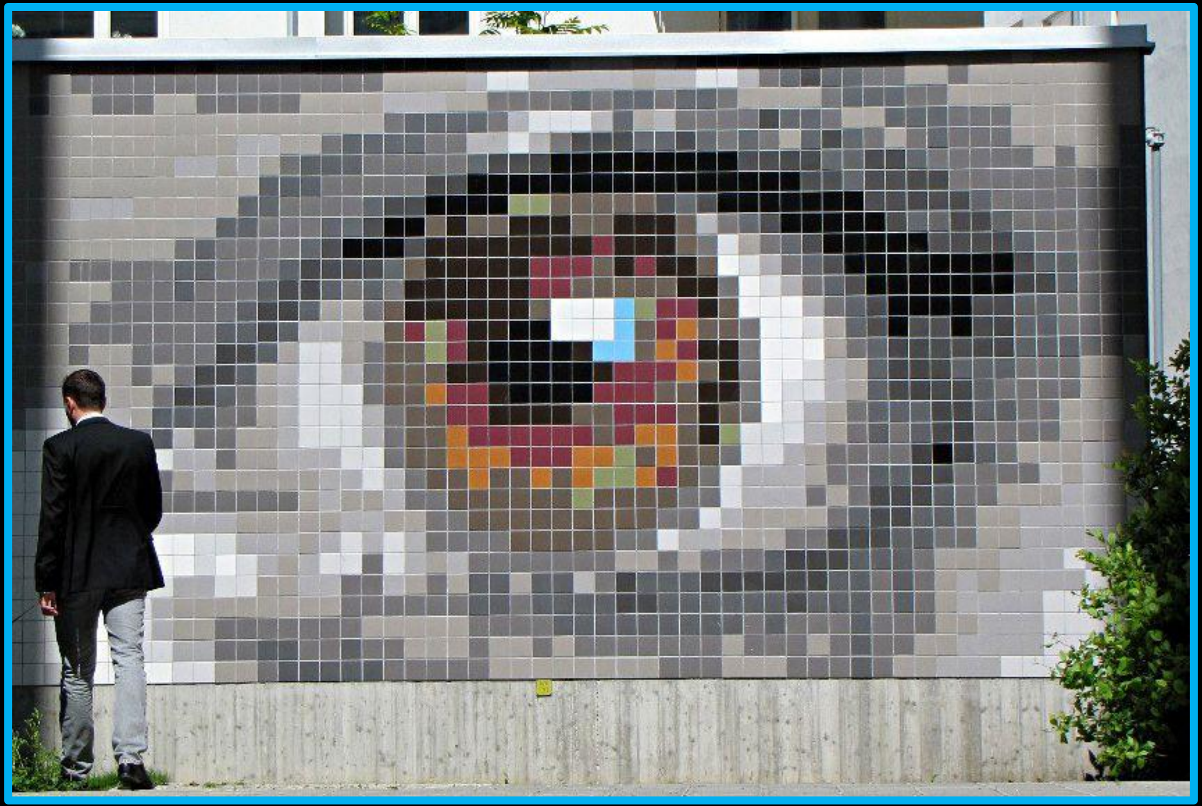
$$\vec{w} = a * \vec{u} + (1 - a) * \vec{v}$$

< 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)*

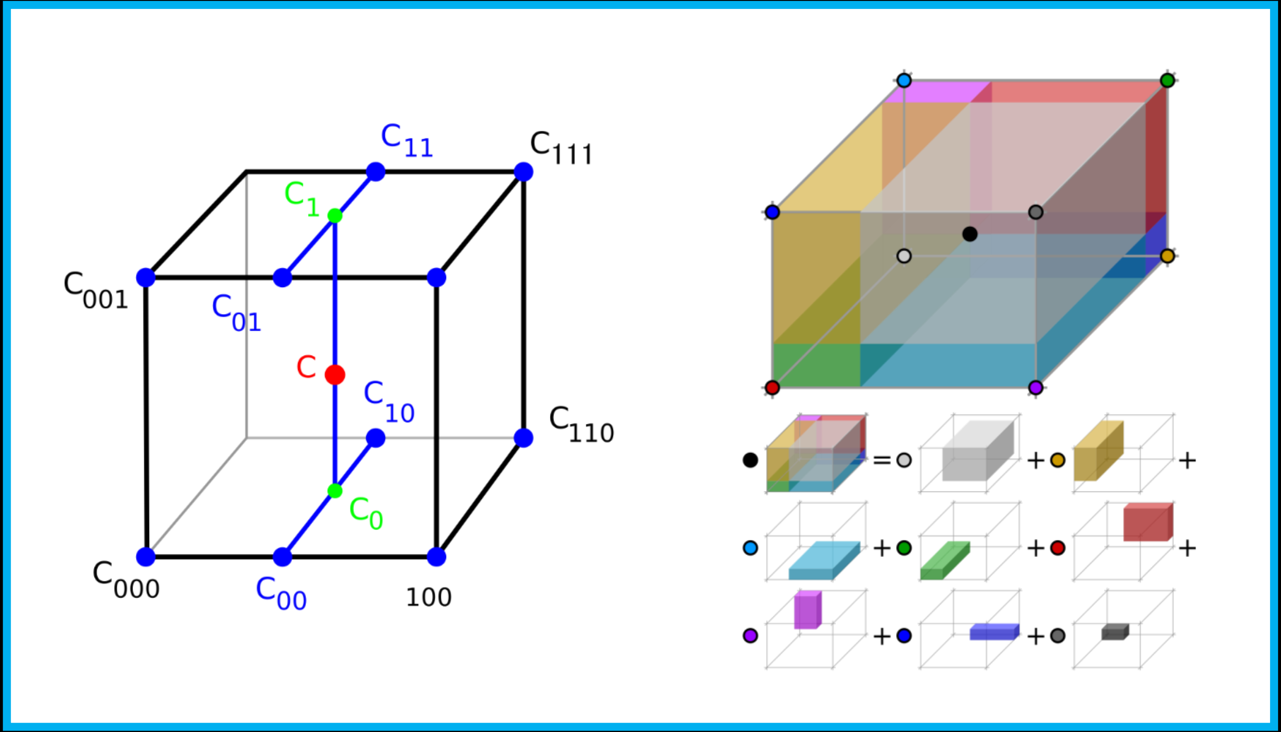
Interpolation





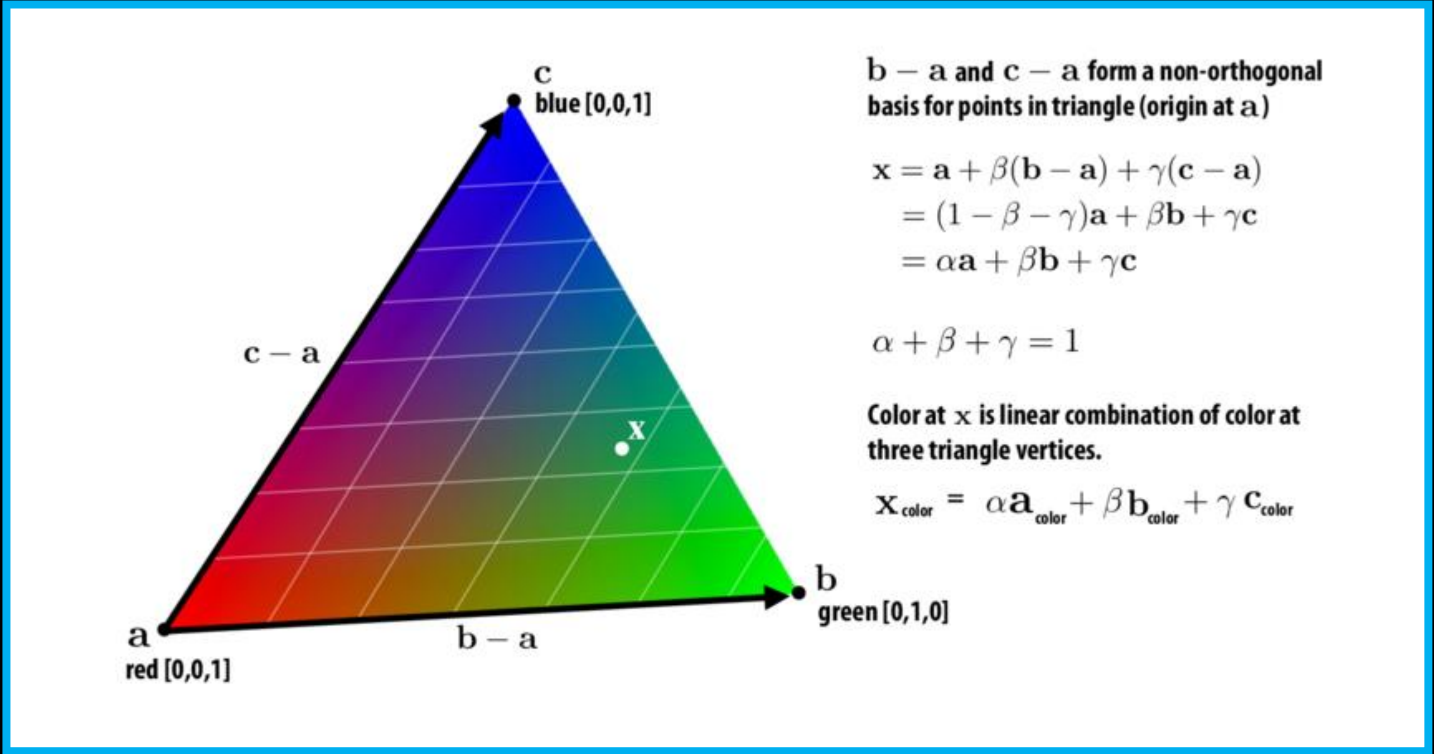
- We also may want to interpolate between data points on a plot.

Interpolation demos



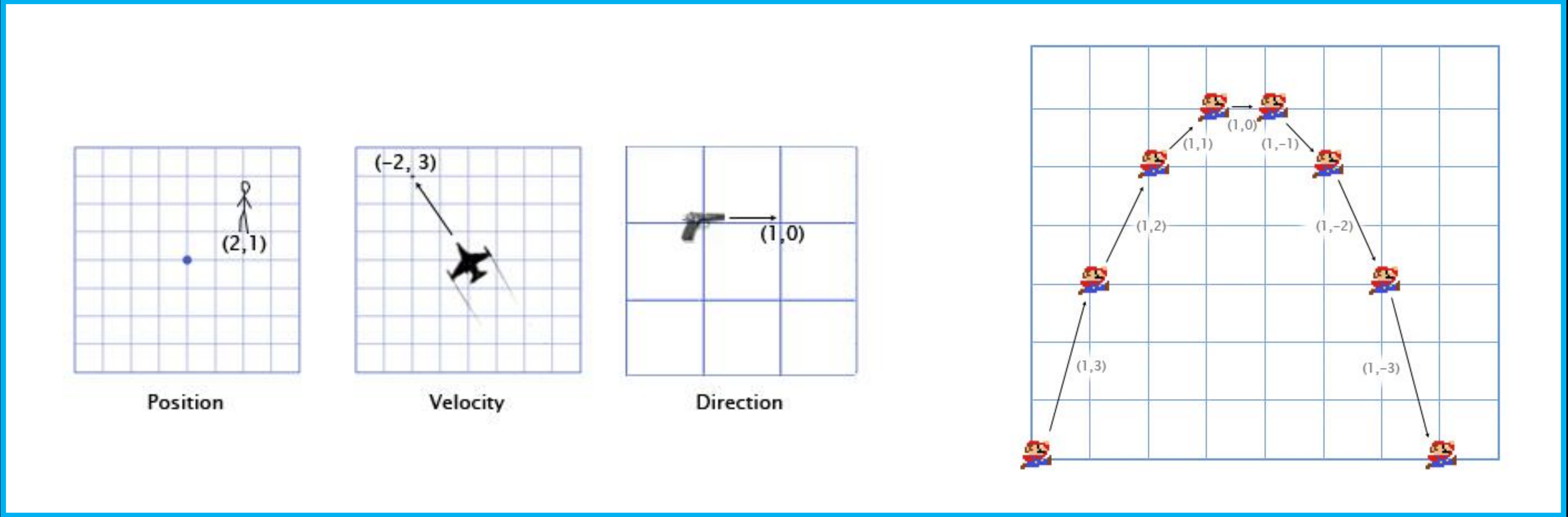
- Bilinear, Trilinear interpolation – interpolating between more than just two vectors
  - **Bilinear** demo: geogebra.org/m/CTn3QkH9 – **interpolations in a rectangle**

Interpolation demos



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

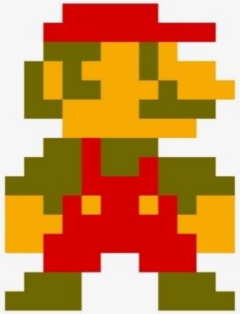
Vectors in games



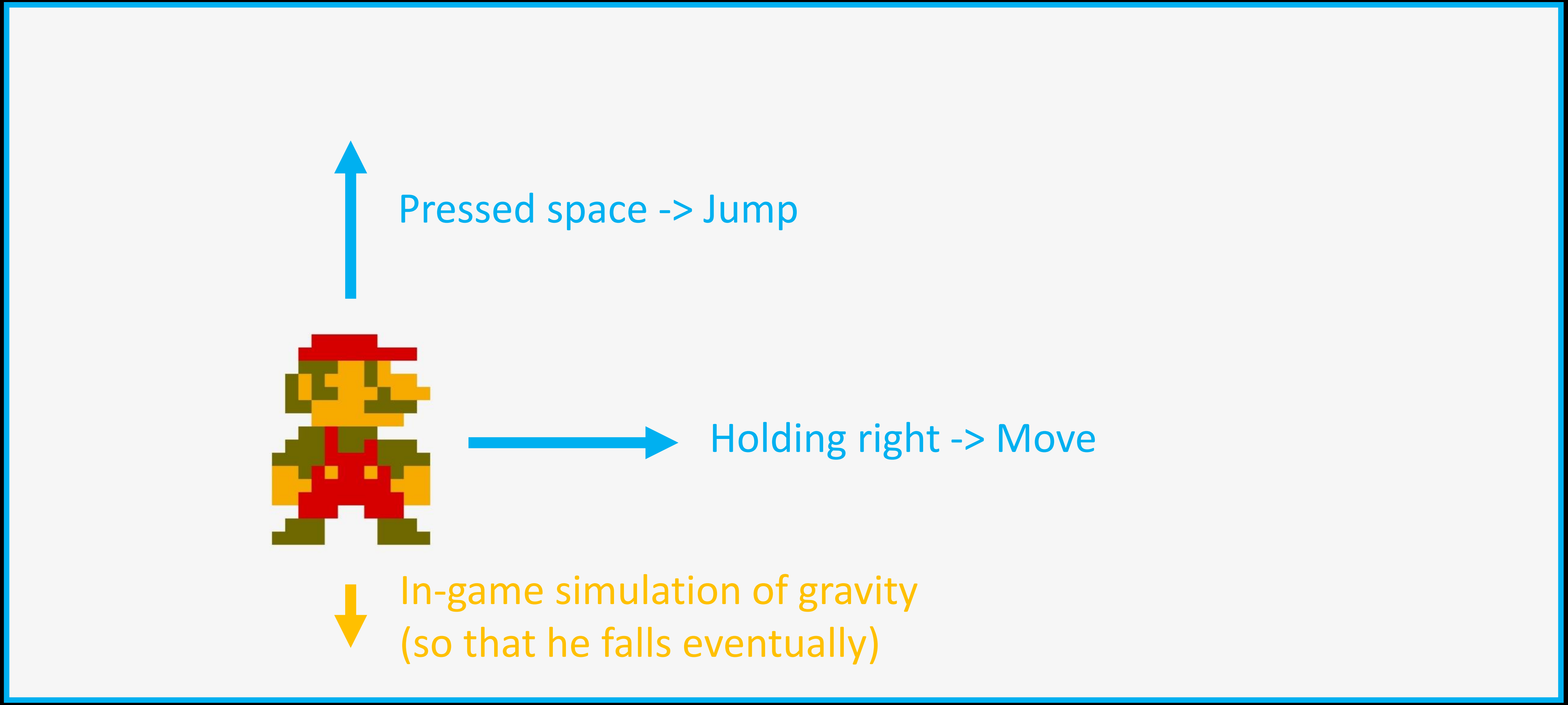
- **Simulating** gravitational **forces**, adding forces together, …
- Checking **vision cones** (cosine metric to get an angle), **rotations** …

Further read: [Blog]

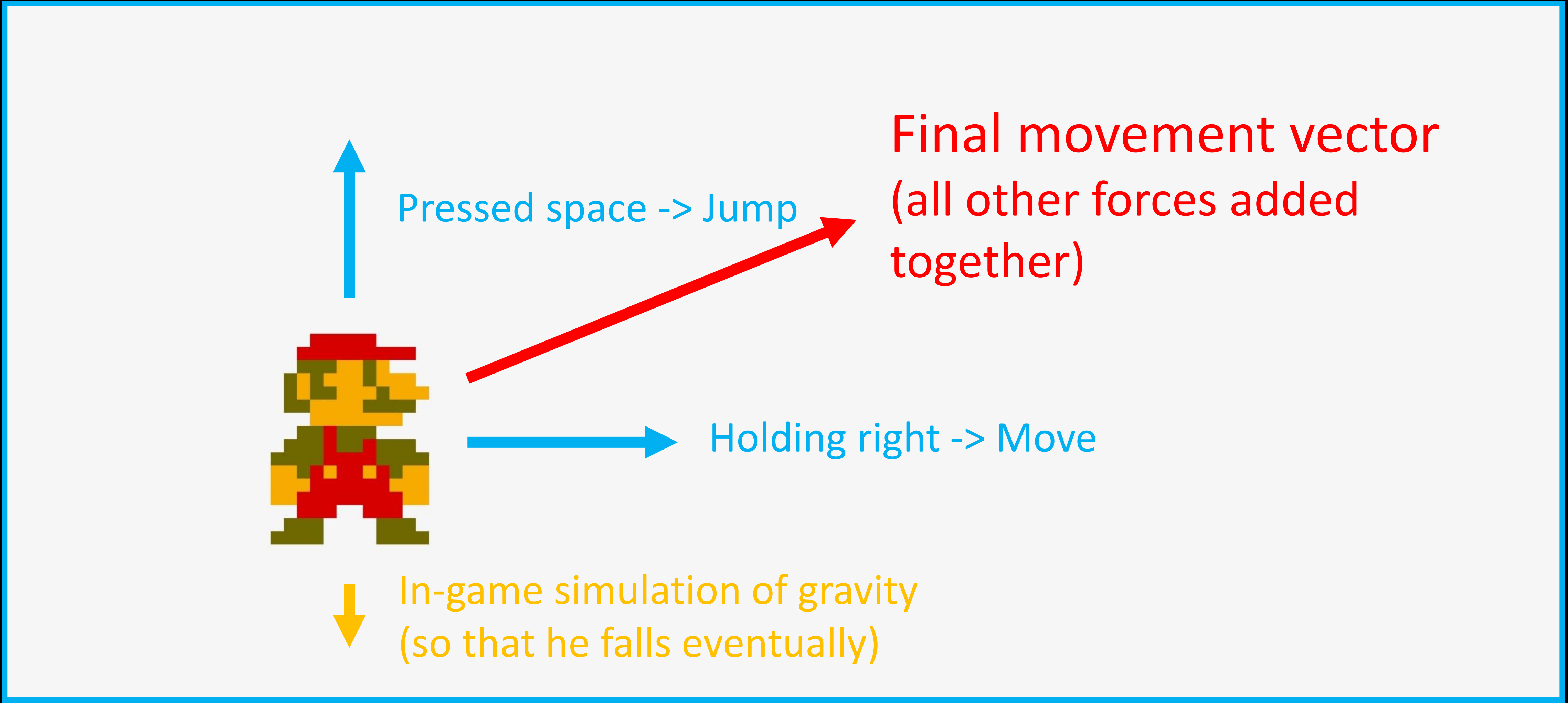
Vectors in games



Vectors in games



Vectors in games



Pause 1

Matrices

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

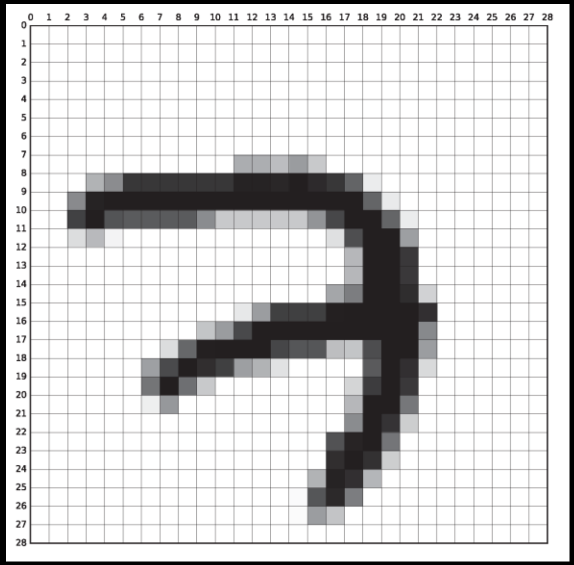
$$\mathrm{M} \in \mathbb{R}^{m*n}$$

Matrices

$$\begin{array}{cc} & \begin{array}{ccc} & & n=3 \end{array} \\ \mathrm{M} = & \begin{array}{|c|c|c|} \hline m_{1,1} & m_{1,2} & m_{1,3} \\ \hline \dots & & \\ \hline & \dots & m_{3,3} \\ \hline \end{array} \\ m=3 & \end{array}$$

$$\mathrm{M} \in \mathbb{R}^{m*n}$$

- We can also consider loaded images as matrices:



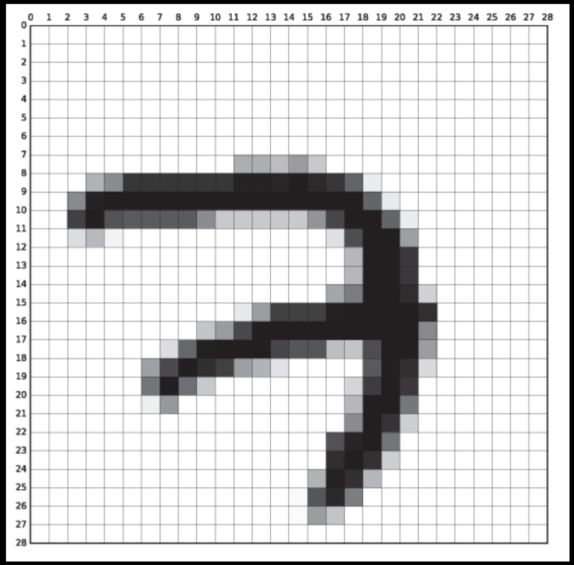
Matrices

$$\mathrm{M} = \begin{array}{cc} & n=3 \\ m=3 & \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ \dots & & \\ & \dots & m_{3,3} \end{bmatrix} \end{array}$$

< Indexing by row and column

$$\mathrm{M} \in \mathbb{R}^{m*n}$$

- We can also consider loaded images as matrices:



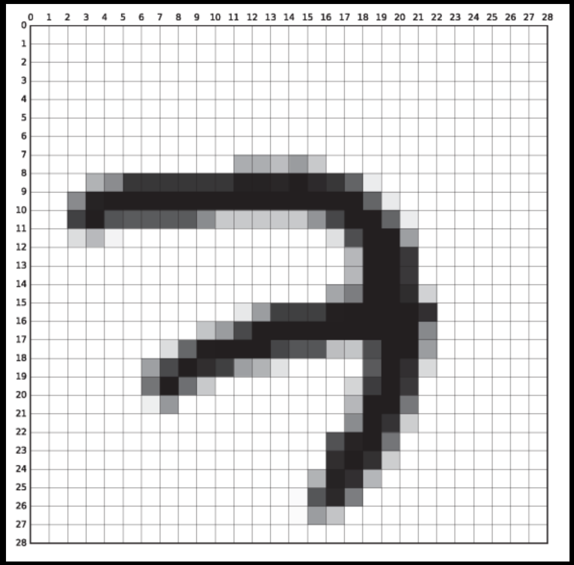
Matrices

$$M = \begin{array}{cc} & n=3 \\ m=3 & \begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & \dots & \\ & \dots & m_9 \end{bmatrix} \end{array}$$

< Indexing as if we flattened it

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

- We can also consider loaded images as matrices:



Operations with matrices

- <u>Multiplication by a real number</u>

$$a * \mathrm{M}$$

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

n=3

m=3

$a * m_1$	$a * m_2$	$a * m_3$
	…	$a * m_9$

Operations with matrices

Having two matrices of the same dimensionality:

- <u>Addition</u>

$$M + N$$

n=3

$m_1 + n_1$	$m_2 + n_2$	$m_3 + n_3$
	…	$m_9 + n_9$

m=3

Operations with matrices

- Transpose M^{T}

$$\mathrm{M} = \begin{array}{cc} & \text{n=3} \\ \text{m=2} & \begin{bmatrix} \boxed{m_1 \quad m_2 \quad m_3} \\ m_4 \quad m_5 \quad m_6 \end{bmatrix} \end{array}$$

Operations with matrices

- <u>Transpose</u> M^T

< Just rotated and flipped

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

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

Matrix-matrix multiply

- Corresponds to a transformation
- Dimensions must match:

$$A * B = C$$

$$m * \mathbf{n} \qquad \mathbf{n} * p \qquad m * p$$

Matrix-matrix multiply

- Corresponds to a transformation
- Dimensions must match:

$$\underset{m*n}{A} * \underset{n*p}{B} = \underset{m*p}{C}$$

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

< We look at separated **rows and columns** ...

BTW: these are vectors!

$$A * B = C$$

$$m * n \qquad n * p \qquad m * p$$

$$\begin{array}{|c|c|c|}\hline a_1 & a_2 & a_3 \\ \hline a_4 & a_5 & a_6 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline b_1 & b_2 & b_3 \\ \hline b_4 & \dots & \\ \hline & \dots & b_9 \\ \hline \end{array} =$$

2 rows x 3 columns

$m = 2, n = 3$

3 rows x 3 columns

$p = 3$

$$A \ * \ B = C$$

$$m * n \qquad n * p \qquad m * p$$

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{bmatrix} * \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & \dots & \\ & \dots & b_9 \end{bmatrix} = \begin{bmatrix} \ & \ & \ \\ \ & \ & \ \end{bmatrix}$$

2 rows x 3 columns

$m = 2, n = 3$

3 rows x 3 columns

$p = 3$

m rows x p columns

$$A * B = C$$

$m * n \qquad n * p \qquad m * p$

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

dot (row 1, column 1)

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{bmatrix} * \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & \dots & \\ & \dots & b_9 \end{bmatrix} = \begin{bmatrix} \boxed{} & & \\ & & \end{bmatrix}$$

2 rows x 3 columns

$m = 2, n = 3$

3 rows x 3 columns

$p = 3$

m rows x p columns

$$A * B = C$$

$m * n \qquad n * p \qquad m * p$

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

dot (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

$$A * B = C$$

$m * n$ $\quad$ $n * p$ $\quad$ $m * p$

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

dot (row 1, column 2) = (1*0 + 0*3 + -3*-1) = 3

$$\begin{bmatrix} 1 & 0 & -3 \\ -2 & 4 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 4 \\ -2 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & \boxed{3} & \\ & & \end{bmatrix}$$

2 rows x 3 columns $\quad$ 3 rows x 3 columns $\quad$ m rows x p columns

$m = 2, n = 3$ $\quad$ $p = 3$

$$A * B = C$$

$$m * n \qquad n * p \qquad m * p$$

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

dot (row 1, column 3) = (1*4 + 0*-1 + -3*2) = -2

$$\begin{bmatrix} 1 & 0 & -3 \\ -2 & 4 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 4 \\ -2 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 3 & \boxed{-2} \\ & & \end{bmatrix}$$

2 rows x 3 columns

$m = 2, n = 3$

3 rows x 3 columns

$p = 3$

m rows x p columns

$$A \ast B = C$$

$m \ast n$ $\quad$ $n \ast p$ $\quad$ $m \ast p$

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

dot (row 2, column 1) = (-2*1 + 4*-2 + 1*0) = -10

$$\begin{bmatrix} 1 & 0 & -3 \\ -2 & 4 & 1 \end{bmatrix} \ast \begin{bmatrix} 1 & 0 & 4 \\ -2 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 3 & -2 \\ -10 & & \end{bmatrix}$$

2 rows x 3 columns

$m = 2, n = 3$

3 rows x 3 columns

$p = 3$

m rows x p columns

$$A * B = C$$

$m * n$ $\quad$ $n * p$ $\quad$ $m * p$

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

dot (row 2, column 1) = (-2*1 + 4*-2 + 1*0) = -10

$$\begin{bmatrix} 1 & 0 & -3 \\ \boxed{-2} & \boxed{4} & \boxed{1} \end{bmatrix} * \begin{bmatrix} \boxed{1} & 0 & 4 \\ \boxed{-2} & 3 & -1 \\ \boxed{0} & -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 3 & -2 \\ \boxed{-10} & & \end{bmatrix}$$

... 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*?

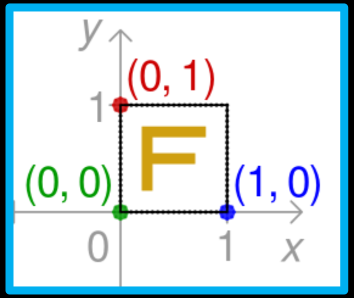
- Matrices can have encoded data (images, points, 3D coordinates, …)
- Multiplication by another matrix can serve as a **transformation operation**:

$$\underset{m*n}{A_{data}} * \underset{n*p}{T} = \underset{m*p}{A_{transformedData}}$$

PS: Video illustration on Khan A.

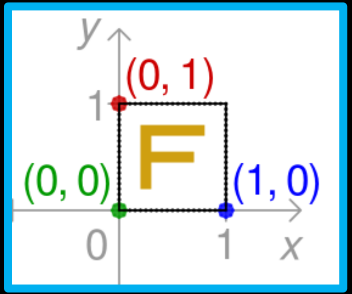
Transformation Matrix

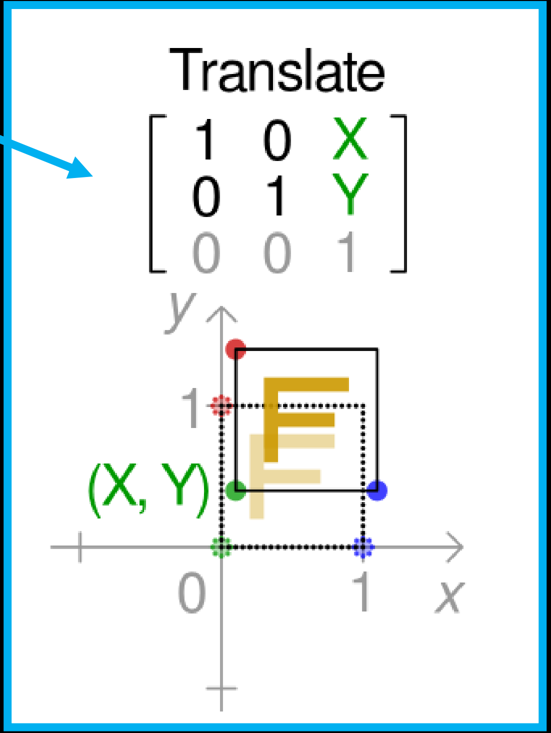
$$A_{\text{data}} * T$$



Transformation Matrix

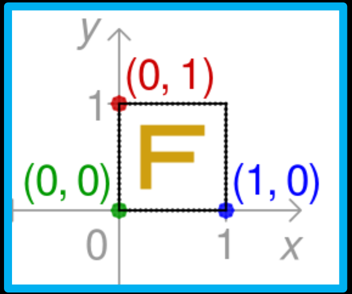
$$A_{\text{data}} \; * \; T$$

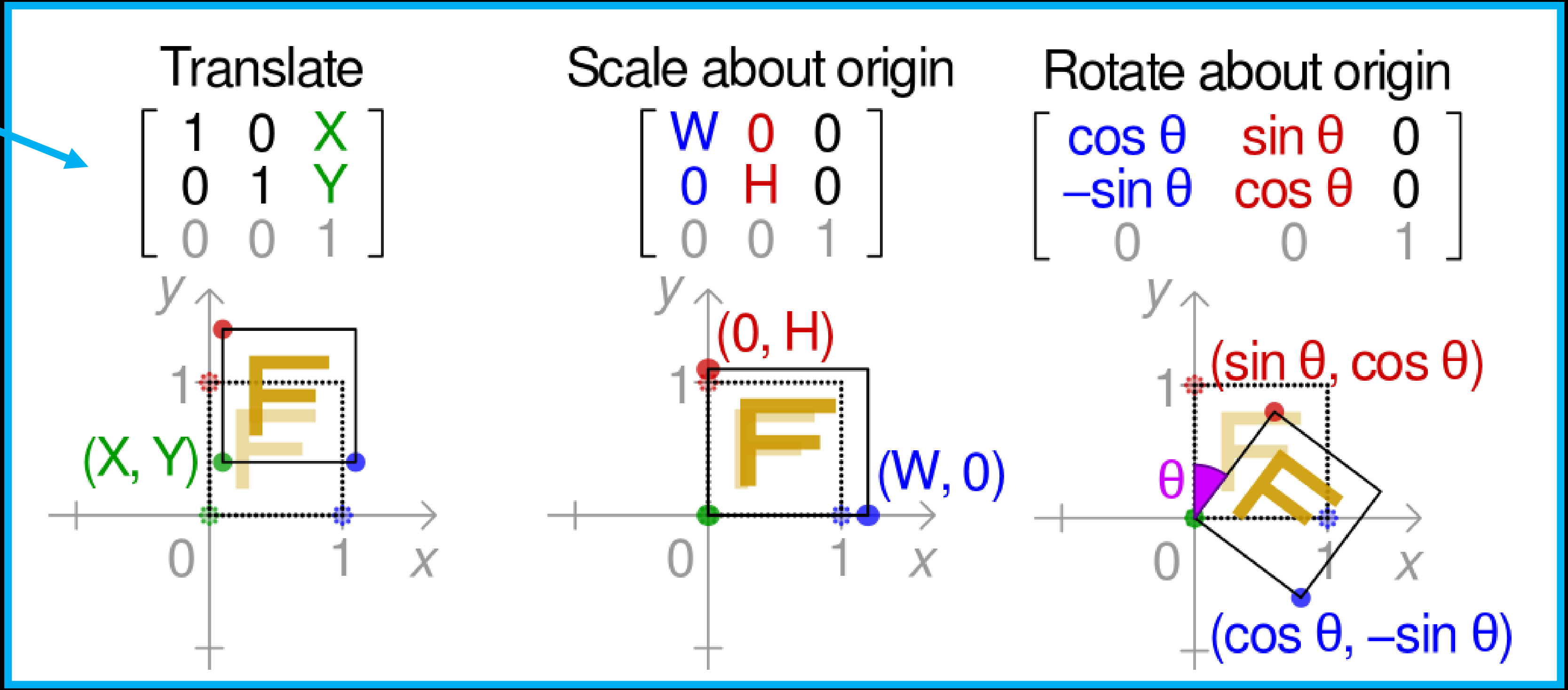




Transformation Matrix

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

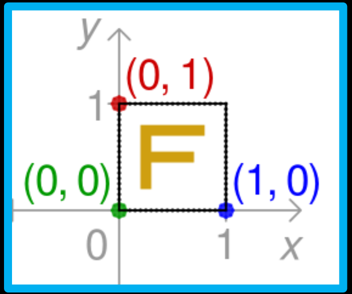


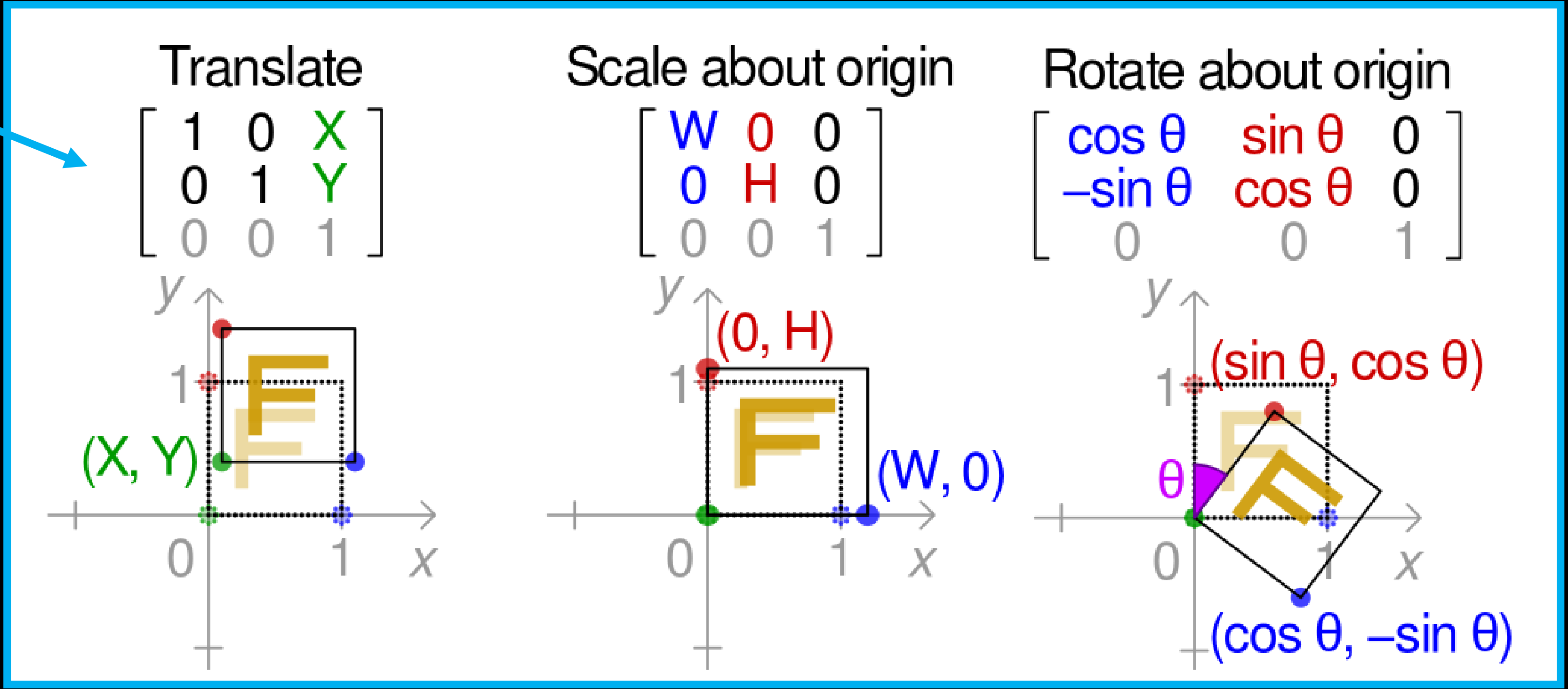


- And many other operations too (*translation, rotation, scale*, …)
- We often use this in Computer Graphics applications (Games)

Transformation Matrix

$A_{data} * T$

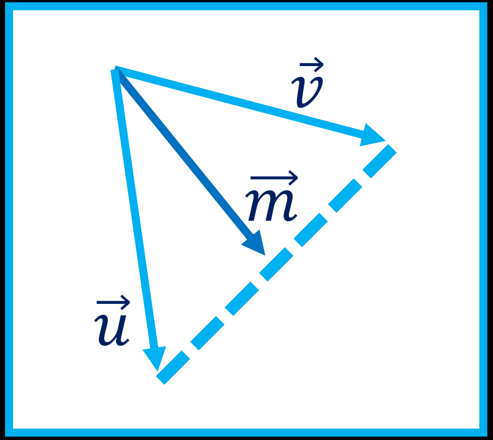




- And many other operations too (*translation, rotation, scale*, …)
- We often use this in Computer Graphics applications (Games)
- **Matrix-Matrix Multiply is very fast on GPUs**

Pause 2

Programming task



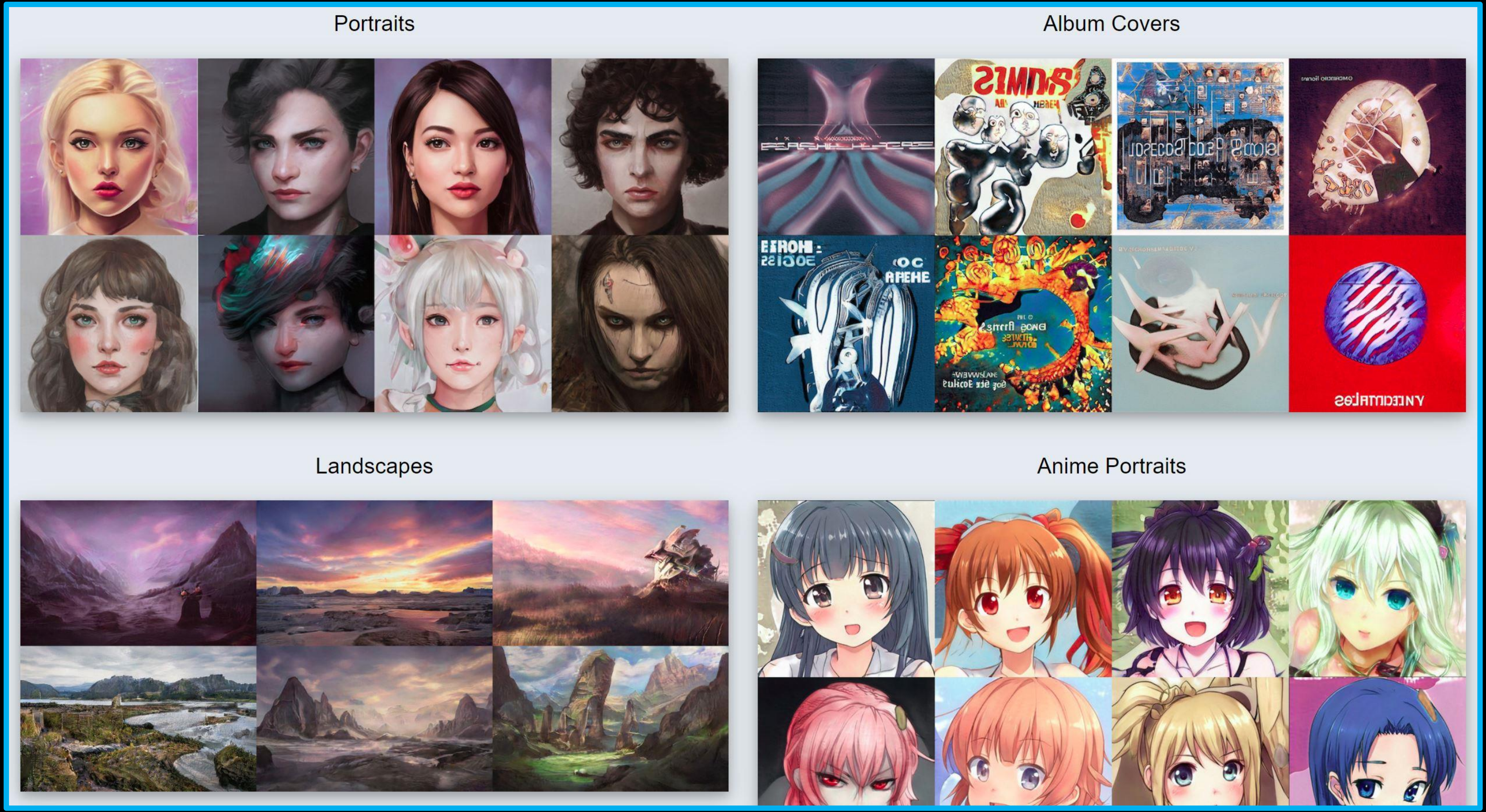
- **Vector manipulation in Python**

- 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
    - $\vec{v} = (v_1, v_2, \dots, v_{512})$ -> Generate an image #1
    - $\vec{u} = (u_1, u_2, \dots, u_{512})$ -> Generate an image #2
  - Using linear interpolation between vectors, they can mix them and **generate images somewhere in between**!

Project ArtBreeder – artbreeder.com/



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