COMP220: Graphics & Simulation

3: Mathematics for graphics

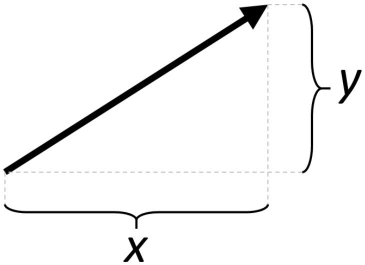
Learning outcomes

- Outcome 1
- Outcome 2
- Outcome 3

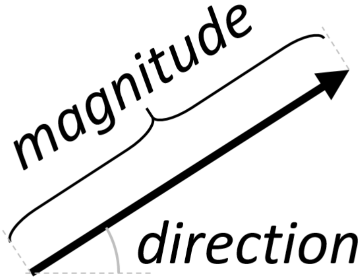
Vectors

Vectors

A vector has **components**



A vector also has **direction** and **magnitude** (or **length**)



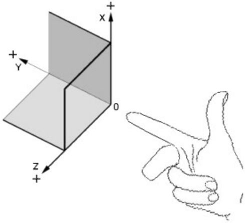
The **origin** is the point represented by the vector $(0, 0, \dots)$

Radians

- We often measure angles in **radians**
- $\pi = 3.14159\ldots$
- π radians $=$ 180 degrees $=$ half a circle
- $\frac{\pi}{2}$ radians $=$ 90 degrees $=$ right angle
- Careful! Some things in OpenGL work in **degrees**, others in **radians** (just to confuse you...)

Right hand rule

OpenGL uses a **right-handed coordinate system**

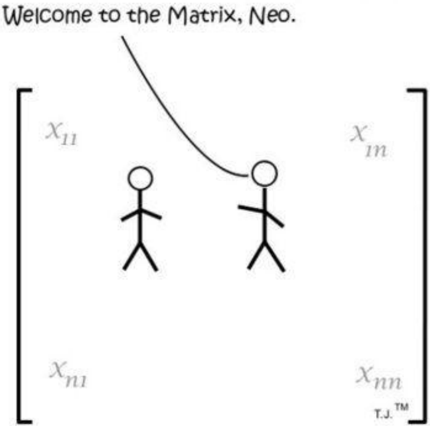


- The *x*-**axis** points towards the **right-hand side** of the screen
- The *y*-**axis** points towards the **top** of the screen
- The *z*-**axis** points **out** of the screen

Homogeneous coordinates

- In 3D graphics, it is useful to represent a **point in 3D space** as a **4-dimensional vector**
- The extra coordinate is called w
- Simple explanation: w should always equal 1 for points in 3D space; having w there makes certain calculations easier
  - (Actually, a point (x, y, z) can be represented as a vector $(x \times w, y \times w, z \times w, w)$ for any $w \neq 0$)
- In homogeneous coordinates, the origin is $(0, 0, 0, 1)$ not $(0, 0, 0, 0)$!

Matrices


Welcome to the Matrix, Neo.
x_{11}
x_{1n}
x_{n1}
x_{nn}
T.J.™

Matrices

- An $m \times n$ **matrix** is a rectangular array of numbers, having m rows and n columns

$$\begin{pmatrix} 3 & 0 & 2.4 \\ 1.7 & -6 & -4.5 \end{pmatrix} \qquad \leftarrow \text{A } 2 \times 3 \text{ matrix}$$

- Note: the plural of **matrix** is **matrices**
- In computer graphics we mostly work with **square** matrices (number of rows = number of columns)

Multiplying vectors and matrices

- Two $n \times n$ matrices can be **multiplied**, giving a new $n \times n$ matrix
- An $n \times n$ matrix and an n-vector can be **multiplied**, giving a new n-vector
- See `https://www.khanacademy.org/math/precalculus/precalc-matrices/multiplying-matrices-by-matrices/v/matrix-multiplication-intro`
- (But you don't really need to know how to calculate these manually...)

Commutativity

- Multiplication of numbers is **commutative**
  - $a \times b = b \times a$
  - e.g. $2 \times 3 = 3 \times 2$
- Multiplication of matrices is **not commutative**
  - In general, $A \times B \neq B \times A$
  - There may be some matrices where $A \times B = B \times A$, but they are the exception

Transformations

Transformations and matrices

- A **transformation** is a **mathematical function** that **changes points in space**
- E.g. shifts them, rotates them, scales them, ...
- Many useful transformations can be **represented** by matrices
- Multiplying these matrices together **combines** the transformations
- Multiplying a vector by the matrix **applies** the transformation

GLM

We will use the **GLM** library to do matrix calculations for us

`http://glm.g-truc.net/`

GLM aims to mirror GLSL data types (`vec4`, `mat4` etc) in C++

Identity

The identity transformation does not change anything

```
// Default constructor for glm::mat4 creates an ↩
    identity matrix
glm::mat4 transform;
```

Translation

Translation shifts all points by the same vector offset

```
transform = glm::translate(transform, glm::vec3(0.3f,  ↩
    0.5f, 0.0f));
```

Scaling

Scaling moves all points closer or further from the origin by the same factor

```
transform = glm::scale(transform, glm::vec3(1.2f, 0.5f ↩
    , 1.0f));
```

Rotation

- How do we represent a rotation in 3 dimensions?
- One way is by specifying the **axis** (as a vector) and the **angle** (in radians)
- Axis always runs through the origin

```
float angle = glm::pi<float>() * 0.5f;
glm::vec3 axis(0, 0, 1);
transform = glm::rotate(transform, angle, axis);
```

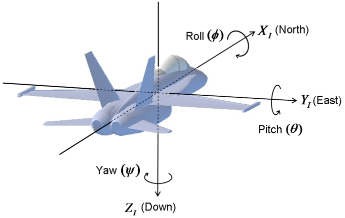
Combining transformations

```
transform = glm::translate(transform, glm::vec3(0.5f, ↩
    0.5f, 0.0f));
transform = glm::rotate(transform, angle, axis);
```

- Transformations **do not commute** in general — changing the order will change the result
- The order they are applied is the **reverse** of what you might think — i.e. the above rotates **then** translates

Euler angles

- Any orientation of an object in 3D space can be described by **three** rotations around:
    - The x-axis $(1, 0, 0)$
    - The y-axis $(0, 1, 0)$
    - The z-axis $(0, 0, 1)$
- These angles are sometimes called **roll**, **pitch** and **yaw**



Gimbal lock

https://youtu.be/rrUCBOlJdt4?t=1m55s