Introduction

• Explore how we can visualise simulations
  • Modelling velocity and acceleration
  • Modelling forces and interaction

• Show how we can utilise *Processing* to show these simulations
Where and Why

• Fluid simulations
  • Rolls Royce – ASiMoV project
    • Need to gain more efficiency out of jet engines
    • Increase power to weight ratio
    • Increase thrust
    • Reduce amount of fuel required, allowing for greener flights
    • Aim is to simulate an entire engine with very high levels of fidelity
Where and Why

- Fluid simulations
- Virus spread simulations
  - COVID-19
    - Each particle is a person
    - One person gets infected, moves around and effects other people
    - Slowly, people recover
Where and Why

- Fluid simulations
- Virus spread simulations
- Cellular Automata
  - Game Of Life
    - Each cell looks at its 8 neighbours
    - Applies a rule depending on how many cells are “alive” and “dead”
Particles

• **Particle** → A single point that contains:
  • A position and (very often) a velocity
  • Often a radius/size
  • Often a mass
  • Sometimes a range in which it can view other particles
  • Sometimes a charge will be applied to a particle

• We can’t set the velocity or the position once it has been created
  • We can set its initial state though
Forces

• Forces are how we move particles
  • Gravity $\rightarrow F = m \times 9.81$ (for $\Earth$)
  • Air resistance $\rightarrow F = s^2 \times \rho \times (-1 \times v)$
  • Friction $\rightarrow F = -1 \times \mu \times N \times v$
  • Steering force $\rightarrow$ desired velocity – current velocity

• All of these can be used with Newtons 2\textsuperscript{nd} Law $\rightarrow F = ma$
Acceleration and Velocity

• From Newtons 2\textsuperscript{nd} Law, calculate the acceleration
  • \( F = ma \) therefore \( a = \frac{F}{m} \)

• We can then calculate the velocity
  • \( a = \frac{v}{t} \) therefore \( v = at \)

• Finally, find the new position
  • \( v = \frac{d}{t} \) therefore \( d = vt \)
• In life, time is continuous...
• We often chuck up time as it:
  • Allows us to approximate time
  • Minimises the amount of computation
  • Allows for flexible granularity
• Therefore, we measure the change in a given timestep
• All the previous equations can be used to with this idea...
Acceleration and Velocity - Discrete time remix

• From Newtons 2\textsuperscript{nd} Law, calculate the acceleration
  • \( F = ma \) therefore \( a = \frac{F}{m} \)

• We can then calculate the velocity
  • \( a = \Delta v / \Delta t \) therefore \( \Delta v = a\Delta t \)

• Finally, find the new position
  • \( \Delta v = \Delta d / \Delta t \) therefore \( \Delta d = \Delta v\Delta t \)
Simulation Process

Create starting state

Draw current state

Update current state

Display final state

If we get to the end state (if any exists!) ...
Presenting Simulations

- We need a large amount of control in order to create these simulations...
- In Python, it can be rather complex so we need to use a different tool
- Introducing *Processing*!
  - Based on Java
  - Designed to allow for artists to easily create interactive installations
Simulation Process - Revisit

1. Create starting state
2. Draw current state
3. Update current state
4. Display final state

If we get to the end state (if any exists!) ...
Process in *Processing*

```java
void setup() {
    /* initial state
     * code here */
}

void draw() {
    /* draw state
     * code here */
}

exit();

/* code within draw function */

If we get to the end state (if any exists!) ...
Some key differences between Java and Processing

• To define and store a colour, *Processing* has the type `color`
  • Can be either black value, RGB or HSB
  • RGB and HSV dependant on `colorMode()` (more on this later!)
  • RGB is default...

• We can have global variables, does not need to be in a class
• `setup` is ran first, `draw` is ran each time we generate a frame
• Loads of built-in functions (see [https://processing.org/reference](https://processing.org/reference))
Processing Example

Richard Kirk  R.Kirk.1@warwick.ac.uk

08/07/2022 ● MB0.01 ● CS2D7 Data Visualisation Lecture 8
Positions in Processing

(0, 0)   (width, 0)

size(
    widthSize,
    heightSize
);

(0, height)   (width, height)
Shapes in Processing

rect(
  x, y,
  w, h
);

ellipse(
  x, y,
  w, h
);

triangle(
  x1, y1,
  x2, y2,
  x3, y3,
);

line(
  x1, y1,
  x2, y2
);
Colours, Fills and Outlines in *Processing*

- `fill(color x)` → all shapes from here are filled with this colour
- `stroke(color y)` → all outlines (and lines) from here are this colour
- `noStroke()` → all the outlines (and lines) from here are not visible
- `colorMode(<mode>)` → changes how the colour is represented.
  - `<mode>` can be either
    - RGB (red, green, blue)
    - HSB (hue, saturation, brightness)

```plaintext
colorMode(RGB);
fill(color(100,150,200));
colorMode(HSB);
fill(color(100,150,200));
```
Interactivity in *Processing*

- *Processing* has a library for sliders, text, buttons
- ControlP5 →
  [https://www.sojamo.de/libraries/controlP5/#examples](https://www.sojamo.de/libraries/controlP5/#examples)
- *Processing* has its own package manager, where we can download and manage ControlP5
  - Sketch → Import Library... → Add Library...
  - `import controlP5.*;` at the top of the file
- We will include some of these in the lab
Acceleration and Velocity – *Processing* Remix

• From Newtons 2\textsuperscript{nd} Law, calculate the acceleration
  • acceleration.x = force.x / mass;
  • acceleration.y = force.y / mass;

• We can then calculate the velocity
  • velocity.x = acceleration.x * dt;
  • velocity.y = acceleration.y * dt;

• Finally, find the new position
  • position.x = velocity.x * dt;
  • Position.y = velocity.y * dt;

Need to define the following globally:
• force \(\rightarrow\) the force that needs to be applied to the particle
• mass \(\rightarrow\) mass of the particle
• dt \(\rightarrow\) the change in each timestep
• Define a particle class

• Each particle is often a separate object
  • Try and keep things encapsulated well
  • `public void update()` → Often included, update the position and velocity of the particles
  • `public void display()` → Often included, display the given particle

• Maintain an array of particles
Interesting Reads and Key Links

• Processing Documentation ➔ https://processing.org/reference
• ControlP5 Examples ➔ https://www.sojamo.de/libraries/controlP5/#examples

Next up: Lab 8 – Particles and Particle Systems