Intro to Processing + Agile Techniques

Workshop 2

Ruzanna Chitchyan, Jon Bird, Pete Bennett
TAs: Alex Elwood, Alex Cockrean, Casper Wang

Today's Workshop

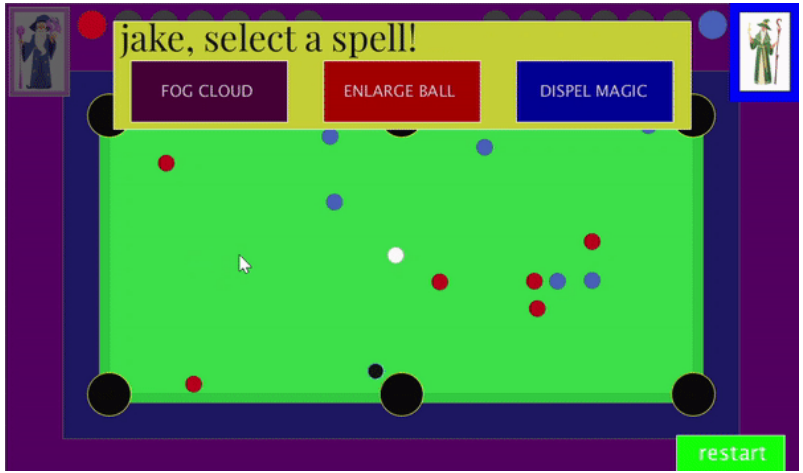
- Project examples from last year (10mins)
- Introduction to Processing (20mins)
- Develop an app (~ 60 mins)
  - Practice Pair Programming
  - Practice Kanban board use
- Try some (very light) evaluation (15mins)

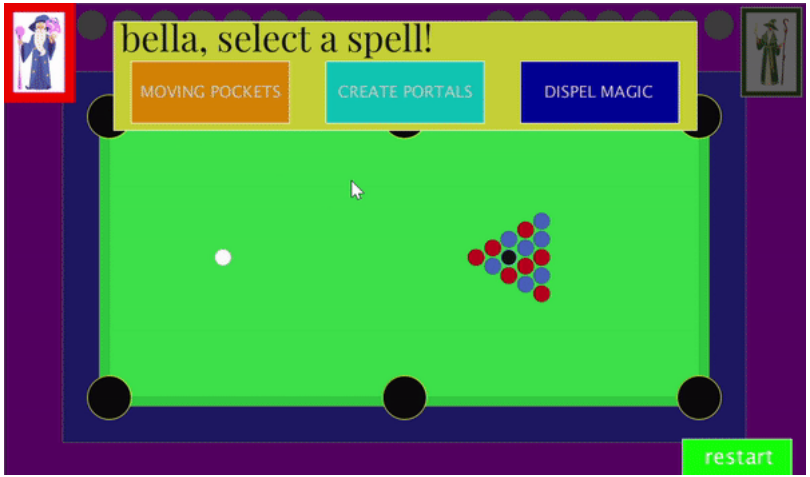
Coursework

- Any questions about the coursework brief?
- Have you made contact with everyone in your team? Let us know now if not!
- Have you made a commit on your repo?
  - Team Photo
  - List of games (inspiration and ideas)

Game Example: Wizard Pool

- "pool with a twist of magic"
- Challenges: Ball collision, spell system, tutorial



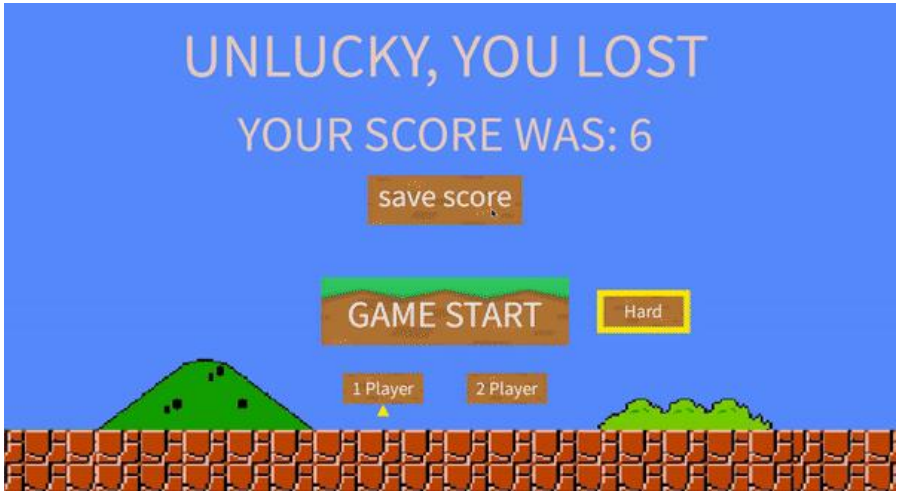


Game Example: JUPITER X Resource War

- Based on the rogue-like game spelunky, but… based in space and has an unkillable ghost enemy [github]
- Challenges: generative map, collision detection, performance optimisation

Game Example : Topsy Turvy

- Classic platformer but… with gravity reversal
- Challenges: physics engine, multi-player mode, high-scores log



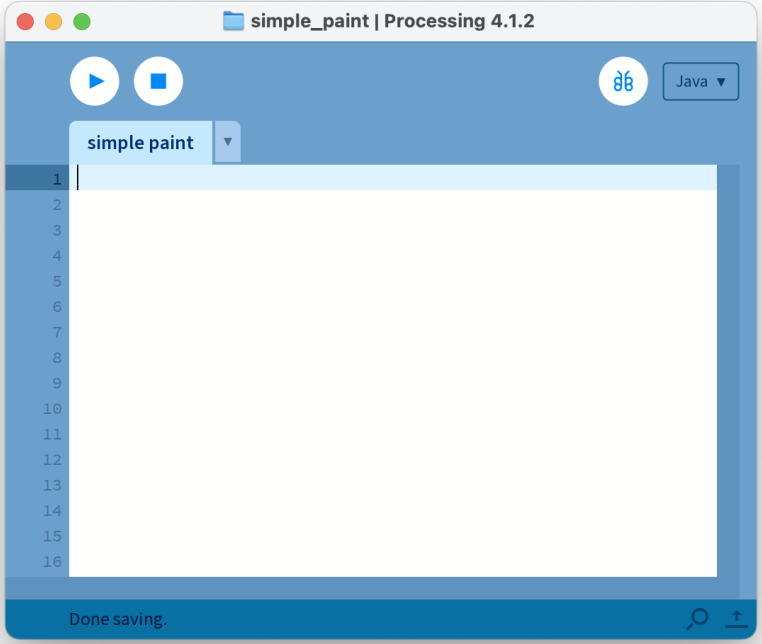
Simple Paint

- Today you are going to develop in pairs a **simple drawing tool** in Processing.
- Inspired by the classic Microsoft Paint, but… this is only a starting point, you will be adding any features you like.
- You will be swapping your paint app with another pair at the end to draw a **portrait**
- *…keep this use in mind before going off and making an abstract generative geometric paint program!*

image

Follow Along Example

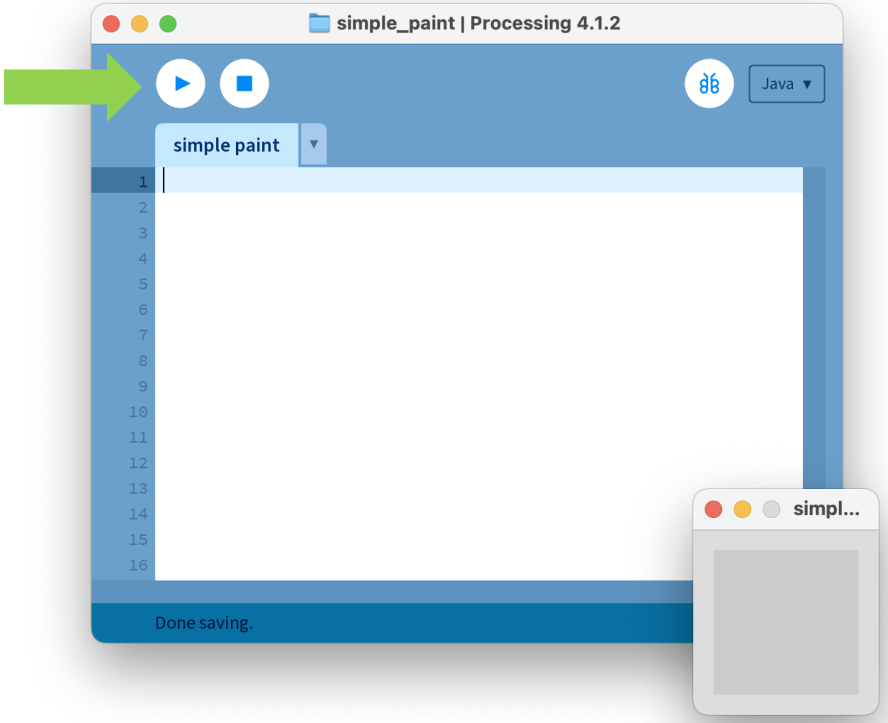
- Follow along with this quick demo.
- This will start you all off with a very bare bones Minimum Viable Product (MVP).
- Start by opening up a new Processing ‘sketch’



Run + Stop

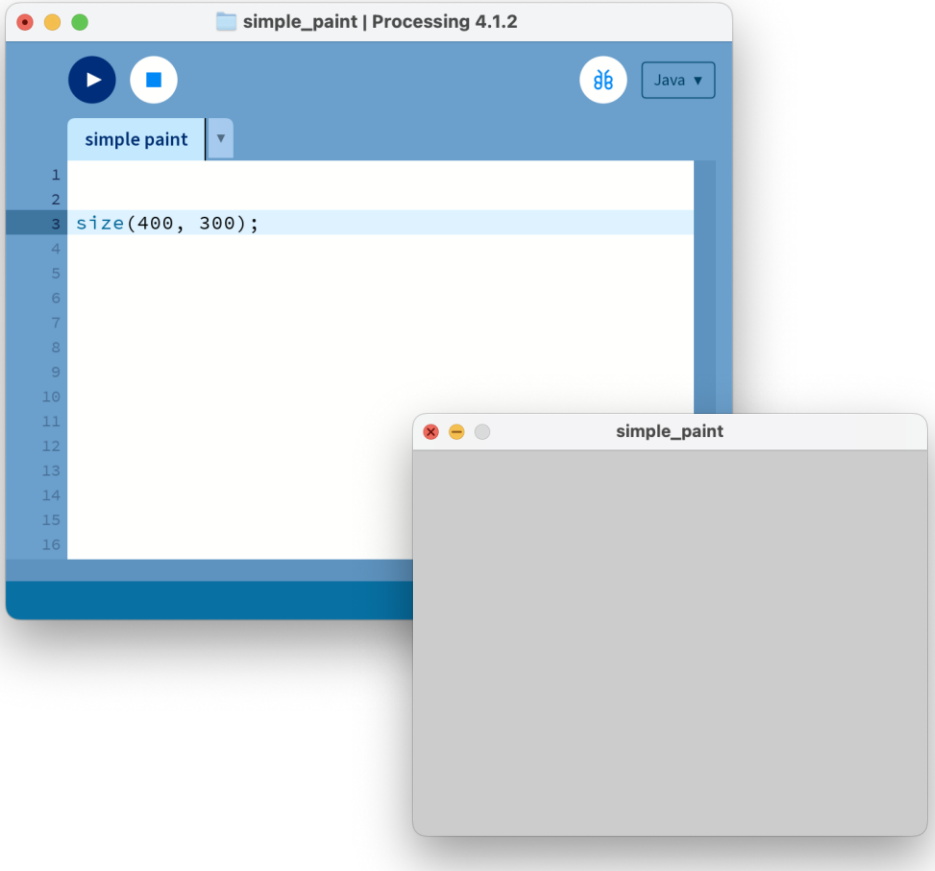
- Use the start and stop buttons to start and stop your sketch from running.

- If you run and empty sketch, it will still work!



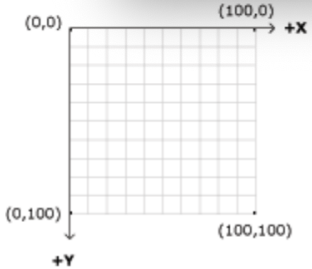
Canvas Size

- First task is to make the canvas a bit larger with the size function:
  - size(*width*, *height*);
  - size(400, 300);
- the size is set in pixels
- Don't forget the semi-colon



Draw a Circle

- Now draw a circle using the ellipse function that takes four arguments:
  - **ellipse**(*x, y, width, height*);
  - **ellipse**(200, 150, 20, 20);
- To draw a rectangle use:
  - **rect**(*x, y, width, height);*
- Coordinate system: **top left is 0, 0**

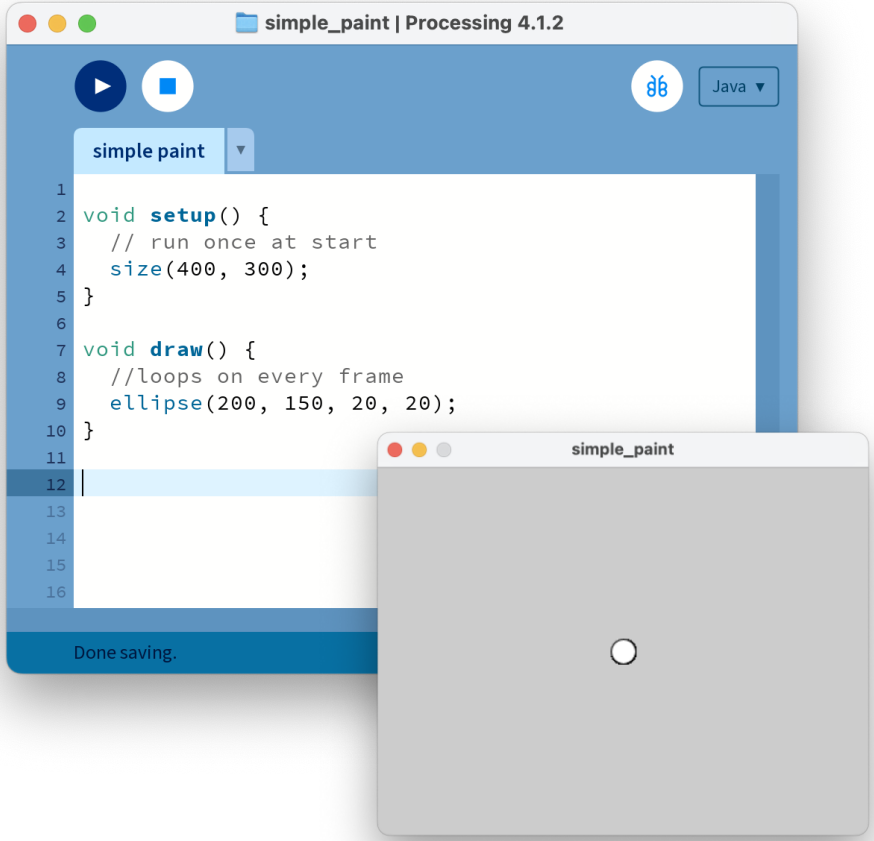


setup & draw

- The last sketch ran through only once then stops. This can work for some non-interactive applications, but we want to be able to animate! So we need to use **setup** and **draw** functions:

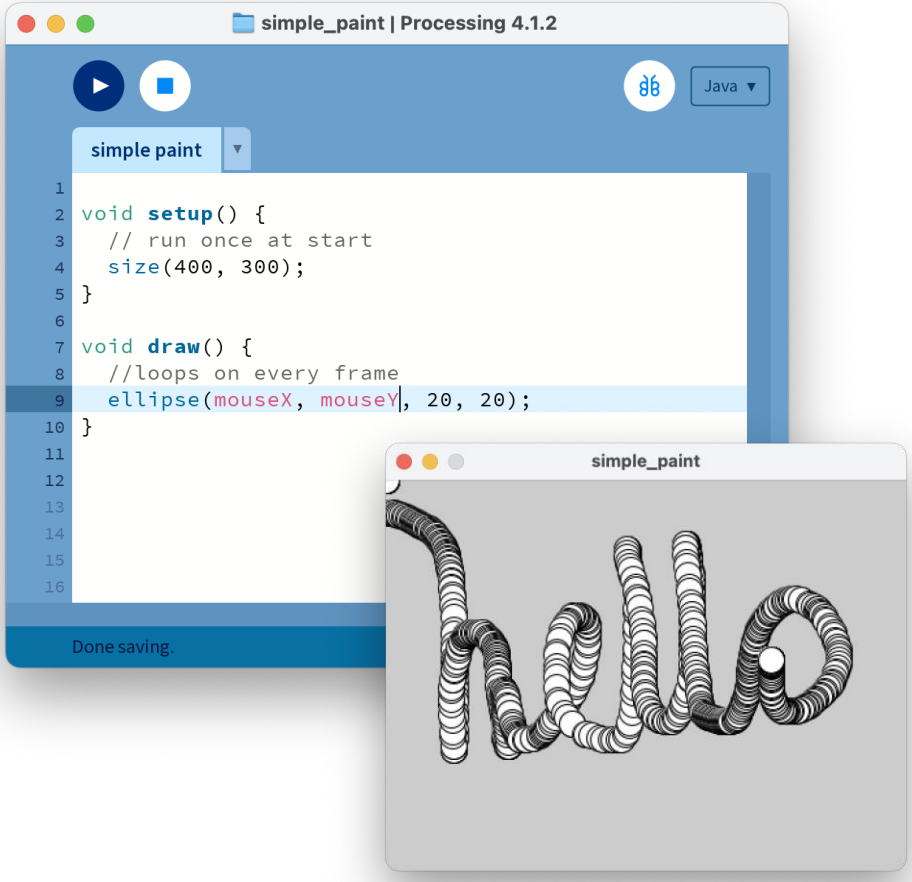
- void **setup**() {
  // runs once at the start
  }

- void **draw**() {
  // runs every frame in a loop
  }



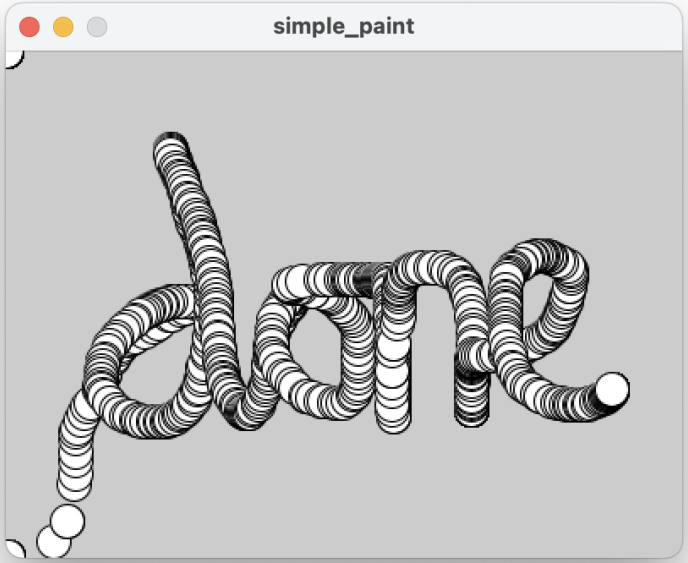
mouseX & mouseY

- Processing has some handy global variables predefined that you can access
  - **mouseX** returns the current mouse x pos
  - **mouseY** returns the current mouse y pos
- The IDE highlights the variable pink to show that it's a predefined variable.
- Other useful ones include **width**, **height** of the sketch window. And previous mouse positions **pmouseX** and **pmouseY**.



MVP achieved!

- You all now have the starting point of a very basic paint program.
    - Try painting the person next to you.
- Your job for the remainder of the workshop is to work in pairs using pair programming and a kanban board to improve on this.
- You will be testing your creation out on another person (with no instructions!) to make a short life drawing portrait at the end of the workshop. But… here are some starting points:



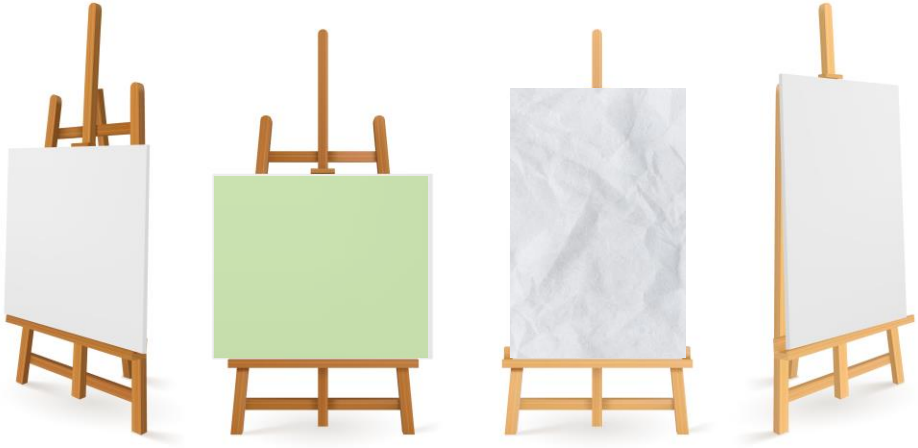
Changing Colour

- Colours values are between 0-255
- **fill**(*red*, *green blue*)
  - Changes the fill colour for the next drawing command.
- **stroke**(*red*, *green*, *blue*)
  - Change the colour for the outline of the next shape

image

Canvas / Background

- Change the background colour of the sketch by calling background(*r,g,b*)
- Try loading an image as a background using image() … note, do this in setup, rather than the draw loop!
- Explore using paper, canvas textures, a sketchbook or use a photograph (say of sky + clouds for a cloud drawing app)



image

Random

- Try randomising some elements with:

  random(*max*)
  random(*min, max*)

- For instance filling a shape with a random colour:

  - **fill**(**random**(255, **random**(255), **random**(255));

image

simple_paint | Processing 4.1.2

Java

simple paint

```
void setup() {
  // run once at start
  size(400, 300);
  background(240, 210, 200); // light pink
}

void draw() {
  //loops on every frame
  if (mousePressed) {
    noStroke();
    fill(100, 100, 200); // light blue
      ellipse(mouseX, mouseY, 10, 10);
  }
}
```

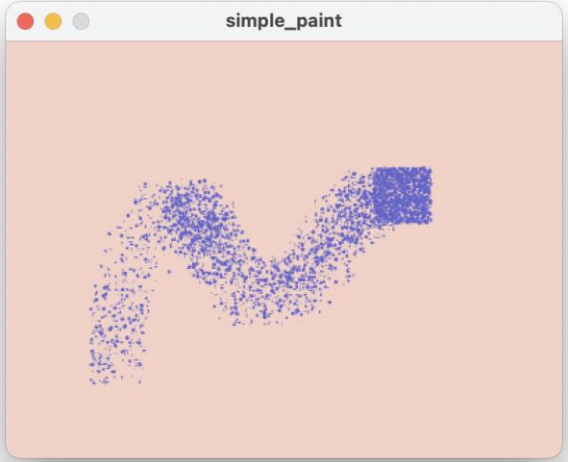
simple_paint

simple_paint | Processing 4.1.2

simple paint

```
void setup() {
  // run once at start
  size(400, 300);
  background(240, 210, 200); // light pink
}

void draw() {
  //loops on every frame
  if (mousePressed) {
    noStroke();
    fill(100, 100, 200); // light blue
    for (int n = 0; n < 10; n++) {
      ellipse(mouseX+random(-20, 20), mouseY+random(-20, 20), random(3), random(3));
    }
  }
}
```



Spraypaint – could be better, perhaps use **cos**() and **sin**() to create a circular area rather than a square?

simple_paint | Processing 4.1.2

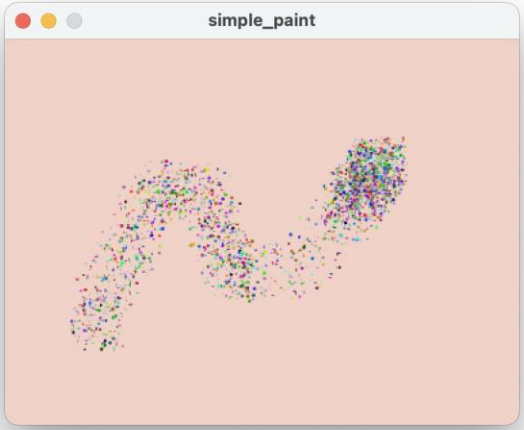
Java

simple paint

```
void setup() {
  // run once at start
  size(400, 300);
  background(240, 210, 200); // light pink
}

void draw() {
  //loops on every frame
  if (mousePressed) {
    noStroke();
    for (int n = 0; n < 10; n++) {
      fill(random(250), random(250), random(250)); // RANDOM COLOUR DROPS
      ellipse(mouseX+random(-20, 20), mouseY+random(-20, 20), random(3), random(3));
    }
  }
}
```

Done saving.



Spraypaint – move the **fill**() command and change the colour for every drop? Try passing two arguments to **random**(*min*, *max*) for a more realistic palette

simple_paint | Processing 4.1.2

Java

simple paint

```
void setup() {
  // run once at start
  size(400, 300);
  background(220); // light grey (single argument is greyscale)
  textSize(20);
  fill(0); // black
  text("Simple Paint v0.1 \nclick to draw a continuous rainbow line", 10, 30);
}

void draw() {
  //loops on every frame
  if (mousePressed) {
   strokeWeight(6);
   stroke(random(100, 255), random(100, 255), random(100, 255));
   line(pmouseX, pmouseY, mouseX, mouseY);
  }
}
```

Documentation + **Commenting** - make sure that a new user can work out how to use your app, and please remember to use comments, either // or /**multiline**/ so that your code is legible

Challenges

These are potential challenges that you could start to fill your Kanban board up with:

- Add a way to change brush colour
- Change the brush size
- Add key commands **keyPressed**()
- Save an image with save("example.jpg")
- Multiple brush types, select with a key
- Use an image() as a brush, rotate randomly to vary the stroke
- Add an eraser. Can this be on right click?
- Add a way to reset/clear the whole canvas
- Auto-scribbler, automatically nudge the pen around the drawing position
- Symmetry – draw a second point on the opposite side of the canvas
- Create smooth lines by drawing between last and current mouse position: line(*pmouseX, pmouseY, mouseX, mouseY*)
- Add opacity to your brush by passing a fourth parameter (0 = transparent, 255 = opaque): fill(red, green, blue, alpha)
- Create lined or grid paper using a for loop.
- Change the **strokeWeight**() of your line based on the speed of mouse movement. Use pmouseX and pmouseY along with mouseX and mouseY to determine the distance moved

- **NOTE**: please add **documentation** – someone's going to be using your system without you being able to explain it. Use **println**() to send text to console or even better **text**() to write to screen.

Going Further

- Many more functions documented in the Processing Reference:
  - https://processing.org/reference
- Look through the reference and see what functions could be interesting to try out
- More info on the Processing environment here:
  - https://processing.org/environment/

`mouseButton`	Shows which mouse button is pressed
`mouseClicked()`	Called once after a mouse button has been pressed and then released
`mouseDragged()`	Called once every time the mouse moves and a mouse button is pressed
`mouseMoved()`	Called every time the mouse moves and a mouse button is not pressed
`mousePressed`	Variable storing if a mouse button is pressed
`mousePressed()`	Called once after every time a mouse button is pressed
`mouseReleased()`	Called every time a mouse button is released
`mouseWheel()`	The code within the `mouseWheel()` event function is run when the mouse wheel is moved
`mouseX`	The system variable that always contains the current horizontal coordinate of the mouse
`mouseY`	The system variable that always contains the current vertical coordinate of the mouse
`pmouseX`	The system variable that always contains the horizontal position of the mouse in the frame previous to the current frame
`pmouseY`	The system variable that always contains the vertical position of the mouse in the frame previous to the current frame

Pair Programming + Kanban

~60 mins

- Partner up. Swap roles regularly.
- Use a Kanban board to plan and track which features you will be working on. Three columns, move features across:
  - *Not Started - > In Progress -> Done*
- Start simple! Keep features small.
- Consider adding a ‘shelved’ or ‘parked’ column to put features in that aren’t working out (given 90min timescale). Don’t get stuck!

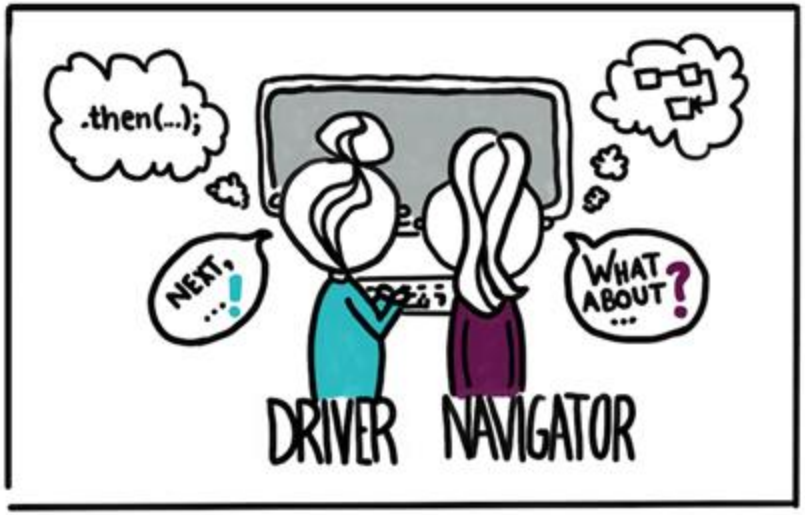
Pair Programming Roles

Driver (Helm)

- The person typing
- Focused on the short-term goal
- Places longer-term goals on backburner
- Talks through what they are doing

Navigator (Tactician)

- Observes the drivers work
- Reviews code in real-time
- Notes suggestions
- Scans horizon for longer-term issues



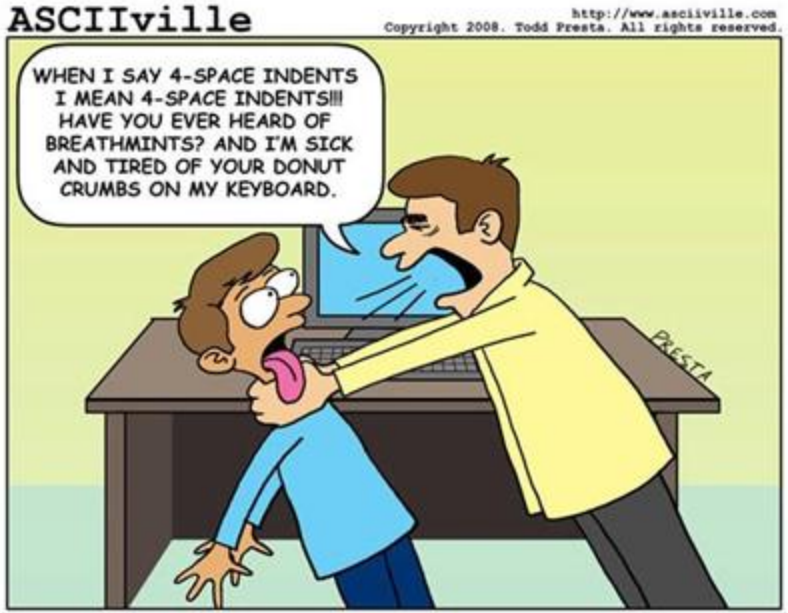
Pair Programming Tips

- **Distractions**. Don't check your phone/mail. Stay focused. Build in more individual time if you need it.

- **Micro-management.** Stick to higher level comments, and avoid saying things such as "now type..."

- **Impatience.** Don't jump right in when the driver makes a typo. They may have seen it and just haven't gone back to correct. Avoid breaking their flow.

- **Keyboard hogging.** Make sure to stick to a rotation schedule and avoid sticking to one role.



The dark side of pair programming.

User Testing

15 mins

- Find another pair and swap over.
- **Without any instruction**, use their paint app to draw a portrait of your pair programming partner (take turns)
- Show the creators of the app how you used it, and your artistic results!
- What worked? What didn't? Did you enjoy it? What would you improve? What was similar with your own? How was the resulting portrait?

image

Week	Date	**Workshop** Monday 09:00-11:00 *MVB 2.11 PC*	**Lecture** Monday 12:00-13:00 *QUEENS BUILDING, 1.40 PUGSLEY*	**Groupwork**
1	22/01/24	Teams, Project Brief [slides] [project brief] [github intro slides]	Introduction and Process [slides] [materials]	Research games, create list on team repo. Install Processing
2	29/01/24	Intro to Processing [slides]	Agile Software Development [slides]	Decide on two game ideas
3	05/02/24	Paper Prototyping, Agile Techniques, Ideas Clinic [slides]	Requirements Engineering [slides] [materials]	Collect requirements. Decide on final idea
4	12/02/24	Requirements [slides]	Object Orientated Design [slides] [materials]	Add requirements section to report
5	19/02/24	Classes Activity, Mock test, Game jam, Summer project prep [slides]	Software Quality and Testing [slides] [materials]	Develop a working prototype over reading week!
6	26/02/24	GAMES JAM	READING WEEK	
7	04/03/24	IN CLASS TEST (assessing lectures 1-4); Testing [slides]	Project Management [slides] [materials]	Define team roles, Estimate sprint effort with planning poker
8	11/03/24	Planning Poker [slides] [heuristic evaluation sheet]	HCI Evaluation Part One [slides]	Write up evaluations from workshop. Plan qualitative assessment (of your choice). Add two difficulty levels to your game.
9	18/03/24	Think Aloud and Heuristic Evaluation[slides]	HCI Evaluation Part Two [slides]	Add quantitative assessment (of your choice) to report
	25/03/24	SPRINT 1	EASTER week 1	
	01/04/24	SPRINT 2	EASTER week 2	
	08/04/24	SPRINT 3	EASTER week 3	
10	15/04/24	HCI Quantitative Task	Software Engineering Extended - Sustainability [slides] [materials]	Develop Game
11	22/04/24	IN CLASS TEST (assessing lectures 5-9)	Coursework Feedback Discussion of marking scheme [slides]	Finish Report
12	29/04/24	Game Demo Day **Monday 29th April 9am-11am**, MVB 2.11 Demonstrate your game. Markers will have a strict 5min window to assess your game, be ready to allow 1-2mins of gameplay and 2-3mins of answering questions.	Feedback on in-class tests (tbc)	Submit Report + Video to Blackboard **Thursday 2nd May 1pm** Submit entire repo as a single zip file to Blackboard. Make sure link to video is clearly displayed at top of repo. We prefer that you have the video on a streaming service of your choice and not contained within the repo so that we're not having to download video files.

homework / groupwork

- Continue brainstorming game ideas.
- Decide on **TWO IDEAS** to bring along to next weeks workshop.
- **Add these two ideas to your repo**! One paragraph each. Images welcome!