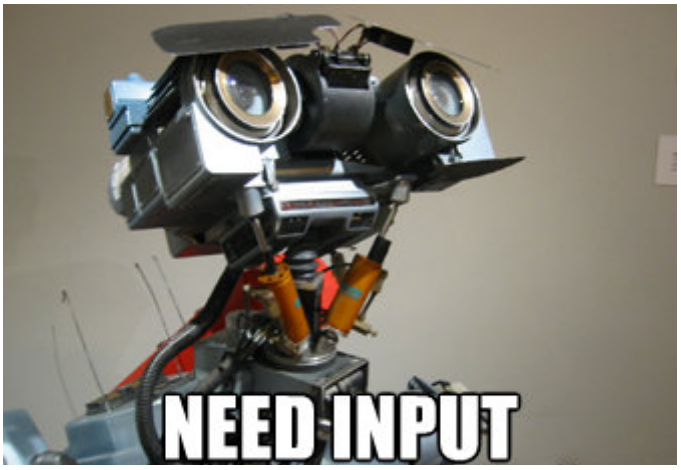
UNIVERSITY OF HULL

Department of Computer Science and Technology

Applicant Experience - Asteroids in Processing Sheet 2: Following the Cursor

In the previous sheet we saw some special words, `width` and `height`, which Processing simply 'knew' about, as if by magic. Just like those special functions we used, `draw` and `setup`, Processing also provides some more variables which we can access from anywhere to know certain things.

Currently our circles are drawing in random locations. However, we want some interactivity in our programs. After all, we want visual feedback, and feedback requires INPUT.

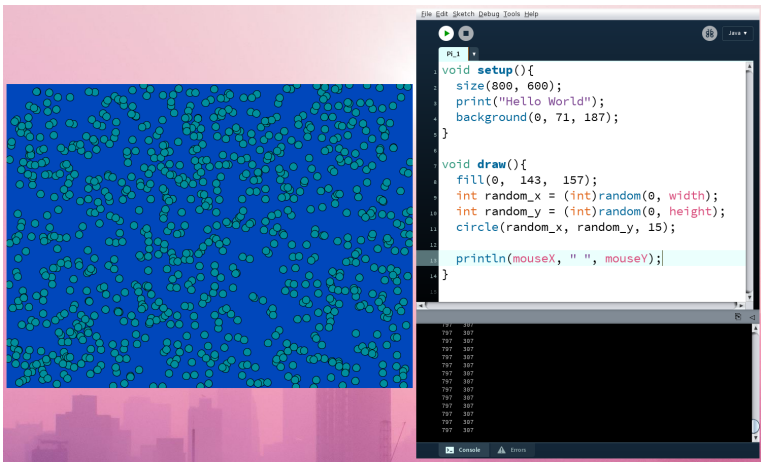


Processing provides us the position of the cursor, in x coordinates and y coordinates, at all times.

Let's use `println` (print line) to see our coordinates in real-time.

1. Remove any print or println statements from your program. (Delete the line)
2. Add `println(mouseX, " ", mouseY);` to the bottom of your draw block. (The gap between " and " is important! It's a space. Otherwise your numbers would run into each other.)

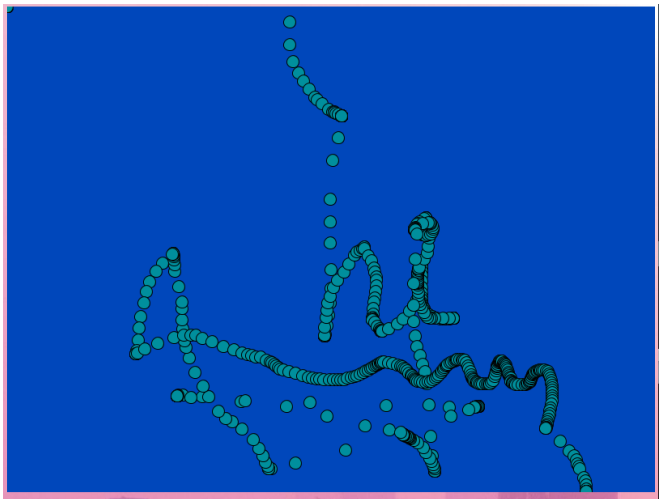
If we run our sketch now, each draw frame it will print the value of mouseX (our cursor x-position), then a space (as we need to separate our numbers), then the mouseY value (our cursor y-position).



You can see in our console window at the bottom, that everytime draw is executed, and a new circle is added to our canvas, we get two numbers: the x-coordinate and y-coordinate of our mouse. Try moving your mouse around and see how the values change each time circles are drawn.

Task 3 - Can you change the circle draw location to be our mouseX and mouseY instead of random_x and random_y?

Re-run your sketch, and move your cursor around on the canvas. The circles should draw underneath your cursor.



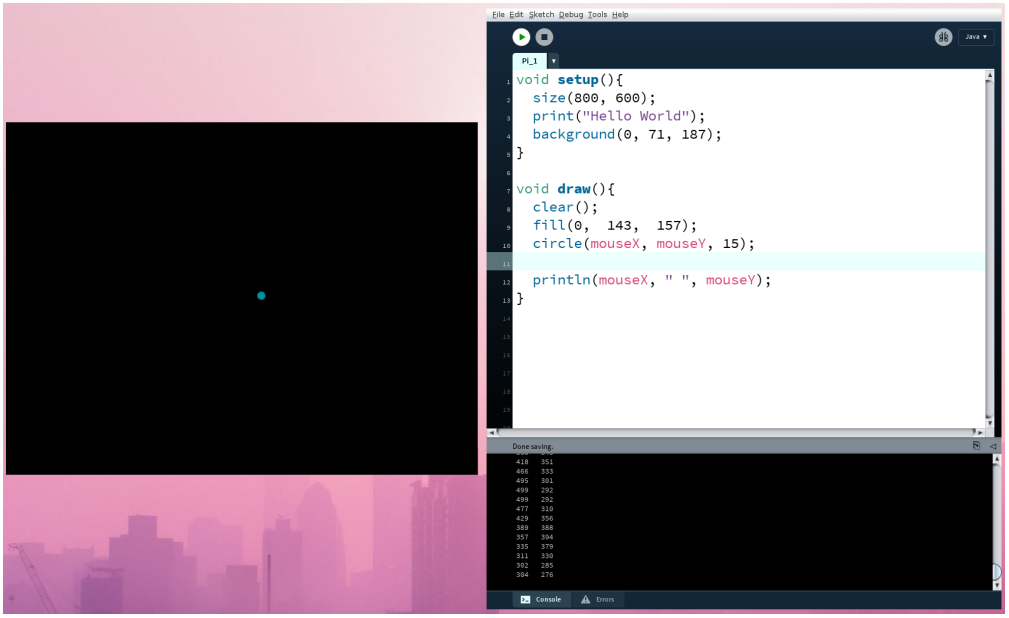
UNIVERSITY OF HULL

Clear the board

Task 4 - This is good, now the circles are drawn underneath our cursor; however, after a while things get a bit cluttered.
Wouldn't it be great if we could wipe our canvas clean?

Well, we can. We can use the `clear()` function at the beginning of each draw call to wipe what came before. Additionally, you may have noticed our random_x and random_y variables are underlined with yellow squiggles. That's because we're not using them anywhere. Processing is helping us out with these messages, and effectively telling us we can remove them as they are unused.

Let's change our draw function to look like the following:



Our random_x and random_y lines are gone and we have added `clear();` to be the very first thing executed in our draw block.

When we run this we can move our mouse around and the circle will continuously draw under our cursor. This is because any previous draw commands are wiped clean. This is what your display (and videogames) do multiple times a second.
As we start our draw stage, we:

1. Clear the screen
2. Select the fill colour of 'something' to draw (in our case a circle)
3. Draw the actual circle, at our mouse coordinates, with diameter 15.
4. Followed by printing our coordinates to the console.

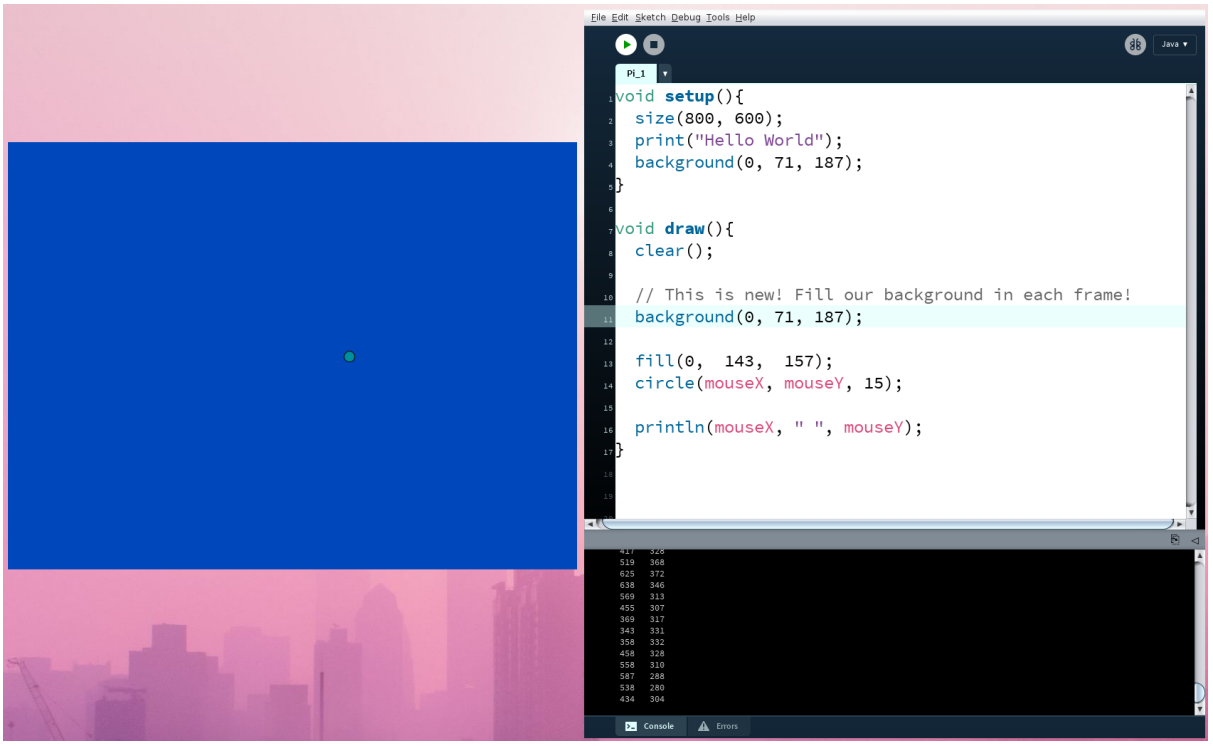
But we have a problem… WHAT HAPPENED TO OUR BACKGROUND?!!?!?!??

UNIVERSITY OF HULL

Task 5 - If we dig a little deeper, we can see that this behaviour makes sense. Where do we tell Processing to paint the background our (0, 71, 187)?

That's right, in the `setup()` block. Remember, this is only ever executed at the very start of our program. This means that for a single frame - blink and you'll miss it! - our screen turns blue. Any frames coming after that all start with a clear, which erases anything drawn previously… including our background!

Let's fix this by ensuring our background colour is always set back to our nice blue colour after that `clear()` step. Just like before, we'll add a line of code after the clear line, which will set our background again. Let's give it a try:



You may notice I snuck in something new whilst we were adding our background line. This is a line which Processing highlights in grey. The line begins with `//` which means that anything after it, on the same line, is ignored by Processing.

As programmers, we call these comments. They are our way of writing little notes to remind ourselves, or other people looking at our mess of spaghetti code, what things do; or rather, the more important question to answer is why we are doing them. As we're going along, maybe you can write some comments when we add things which seem a bit magical, so that you can remind future you what it does.

Experiment with the new things we've covered in this worksheet. What would I need to do to make the circle follow just the left-right movement of my cursor, but stay at a fixed height? (Similar to a block breaker paddle, or space invaders player)