UNIVERSITY OF HULL

Department of Computer Science and Technology

Applicant Experience - Asteroids in Processing Sheet 5: Keeping track of our ship

Having the ship attached to us at all times is nice, but we need it to exist in our world, to have coordinates it can call its own.

Let's create a PVector to store this in. We'll aptly call this ship_pos. Like with our pA, pB, pC that we wanted to persist and keep track of, we will declare this PVector at the very top, outside of setup and draw. We can then use this within draw() to push our ship around.

At the top:

```
PVector ship_pos; // Goes at the top of the file, next to PVector pC;
```

In Setup:

```
ship_pos = new PVector( width/2, height/2 );
```

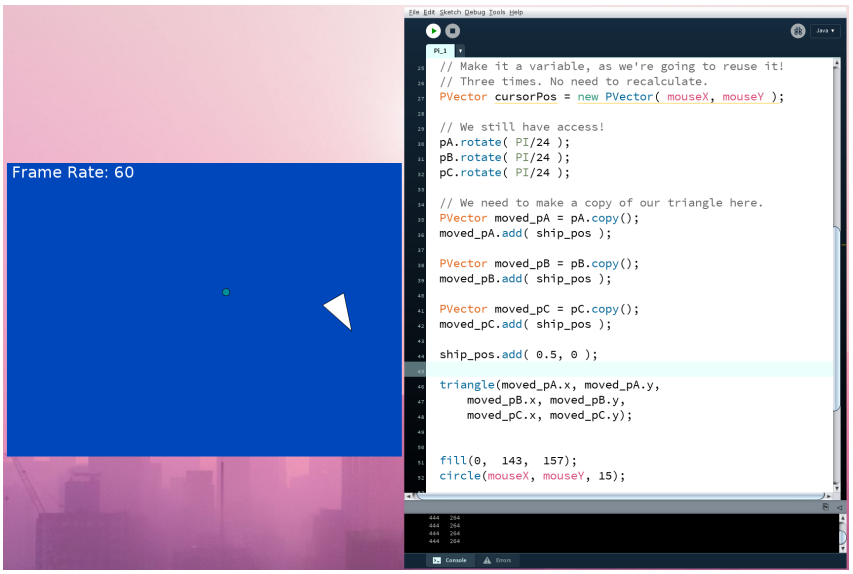
In Draw, after the translation, but before calling triangle():

```
ship_pos.add( 0.5, 0 );
```

Similarly, change all `moved_pA.add( cursorPos );` to use ship_pos instead of cursorPos. The reasoning for this is that it has its own home position now, it's not tied to our cursor. So we want to translate it back from (0,0) to where the ship needs to be. That's the whole reason we now have a PVector to keep track of where to put it.

This should slowly move our ship towards the right-hand side, until it goes off into the void beyond our drawable screen. We can put the coordinates to anything we want, even extreme values such as 54,321 along the X. But we're only able to draw as far as our window goes. 800 pixels. The starting position we want is for our ship to be in the center. This is easy to calculate, as the width of our window is over 2, and likewise the same for height.

UNIVERSITY OF HULL



To prove to yourself that we're modifying the ship_pos PVector, you could print it out at the end of each draw cycle. You should see the numbers incrementally go up, in line with whatever you put in `ship_pos.add()`.

Exploration

We're not done yet, but we've made some good progress towards the game. It's beginning to take… shape.

Part of what makes Computer Science so fun and interesting, and why we chose to use Processing in this series, is to mess around with things and to see their impact. For everything we've covered so far, try experimenting with the values, try changing something and testing your own understanding and limits, this is where the secret to learning programming comes from.

But what if I break it?
So what! It's a piece of code that you wrote. You could undo or make a copy of the file beforehand so you have something to go back to. Or just copy the relevant sections of code from this document.

For inspiration, try playing with some of the values we've been dealing with. These could be rotation amounts, or the amount to move our ship by (E.g negative values of x move it left).

What if we rotated pA, pB, and pC by different amounts each draw frame? Would it still be our triangle?

Can you randomise the colour of our ship?

Can you add text to the screen which prints the angle of our triangle? (you might want to lower the rotation amount for your own checking purposes).

Hint: If we take straight up as our orientation, we can use pA and we're looking for its heading.

As a small parlour trick, I shall provide the magical incantation to turn our Beyblade of a broken spaceship into a miraculous space airplane.

```
ship_pos.add( 10*cos( pA.heading() ), 10*sin( pA.heading() ) );
```

If you're really maths crazy (I know I am!), then you might be able to use some fancy trigonometry to show how what we have above (10cosA, 10sinA) would make a circle as we move the value of A from 0 to 2PI.

That's all for now, congratulations for sticking through until the end. You should be able to find the Asteroids sketchbook available through the Processing window by going to Open, and it will be in the same location as your sketchbooks. This is an example of what we are eventually aiming for on our journey.

Stay tuned for more updates and Processing tutorials. In the meantime, we have included an example of what our final Asteroids game might look like. If you go to open a sketchbook, you might spot the 'Asteroids' folder. Go and play the sketch, we're using circles in our example (to make the maths a bit easier). The code is a little bit more complicated than what we've covered so far, but in-time we will get there.