# Programming Challenges – Week Two

Attempt each of the programming challenges below in Processing (or your preferred language). You will most likely encounter many important and useful ideas and tools along the way that you can use in your own projects.

**PLEASE** deviate from these instructions if you have some ideas you want to try out! Creativity is encouraged and a good way to learn new things.

# The initial program

Refer to the following link:

* <http://www.learningprocessing.com/examples/chapter-8/example-8-1/>

Copy the program there into a new sketch. Make small adjustments to the program until you understand vaguely what the various parts of the program do.

You might find it useful to look at a slightly extended version of the same program:

* <http://www.learningprocessing.com/examples/chapter-8/example-8-2/>

…and if you’re interested in an explanation of what is going on, the following link is could be helpful: <http://processing.org/learning/objects/>

# The improved program

* Add two more cars of different colours, moving in the opposite direction to the existing cars
* Give the cars the ability to have speed in the y-direction as well as in the x-direction

# Bending the program

* Replace the cars with circles (**ellipses**) – make the program include two ellipses of different colours, moving in different directions.
* Make the ellipses ‘bounce’ off of the edges of the window, instead of reappearing at the opposite edge (wrapping).
* Simulate a gravitational attraction towards the bottom of the screen (hint: continually add a constant amount to the vertical speed of the ellipses)
* Introduce user interaction – for example:
  + Make it so that clicking the mouse generates a new ellipse with random properties at the position of the pointer
  + Make the ellipses follow the pointer around the screen
  + devise a control scheme so that two ellipses can be controlled independently with the keyboard

# Further challenges

* Make it possible for two circles to bounce off of one another (simple **collision detection**)
* Turn this program into a game for two players, where the aim is to push an opponent’s ball into a scoring region