# Programming Challenges – Weeks Three and Four

Attempt the programming challenge 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.

This time, instead of a series of small programs, the focus is on developing a working bit of software that other people can use. It’s not an easy task, but you stand to learn a lot about programming – and it can be very rewarding to see your ideas take shape in software!

That said, **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 challenge

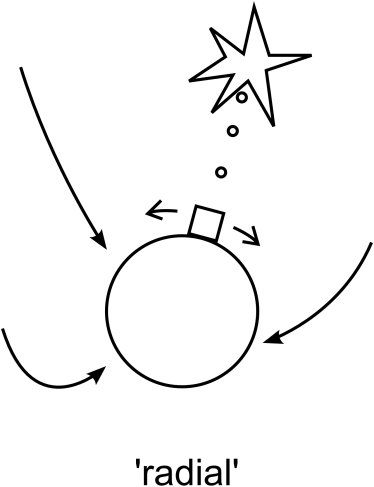
Following the success of the Space Design Competition team, we are one step closer to establishing colonies in the inner Solar System. However, to adequately protect the colonists from hazards such as space rocks, crashing spacecraft and hostile attackers, the need for a defensive weapon system is clear.

Your job is to create a game or simulation that will allow people to practice using such a weapon system to destroy incoming threats to a space colony. It may be important to make the game fun, or addictive somehow.

Note that the game itself doesn’t have to be about space war! Feel free to pick a different setting or a different type of game altogether.

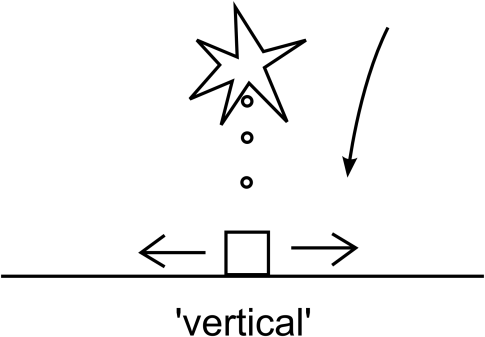
This project will take a while to complete – at least two sessions, and some time in between! However, it’s likely that your game will be ‘playable’ for a large part of the development as you gradually add more features to it.

# Thinking about the program

First of all, take a little while to think about how the game will look and be played.

Here are some basic elements that you will probably want to include:

* A player-controlled cannon (will you use the mouse or keyboard to move/aim/shoot?)
* How you want the player to move: confined to a flat surface (see ‘vertical’, right, and games like *Space Invaders*), to a planet (see ‘radial’, right) or floating in space (see the game *Asteroids*)?
* Missiles/rocks/hazards that appear at the edges of the window (do they track towards the player, the ground, or continue in straight lines? Affected by gravity?
* Something to defend - the ground? Cities (see *Missile Command*)?

Note your ideas down along with sketches of how the game is likely to work. It might help to look at some similar games to get some ideas.

There are many aspects to the game (the programming, user control, graphics and VFX, sound effects, scoring, a story) that might become important. Keep these things in mind – you may want to include these as the game takes shape. If you’re working alone you have to avoid fixating on just one of these things! Working with others can be helpful, as different people will be interested in working on different aspects of the project. This kind of teamwork can be both fun and productive.

But for now, it’s probably a good idea to start PROTOTYPING…please see the next page!

# Prototyping

Once you have sorted out some of the basics of the game, it’s time to start solving the problem of ‘how do I get the computer to do it?’

There are a couple of ways to do this. One is to do a series of experiments, writing small programs that deal with only a couple of parts of the game at a time. For instance, you might want to write a program that handles the player movement and shooting. Then you can write another program that handles the space rock behaviour. Once you are ready you can combine these programs into what will become the game.

The other way is to dive right in and start building up the game. You might still begin with ‘getting the player movement working’, but you will then add new features directly on top of this. This will likely result in some rather tangled code and the occasional terrifying moment when you think you have ‘broken’ the entire game because of a little change you have made! However, you’ll be able to see how the game plays as you make it, which can give you lots of good ideas as you go.

Really, either way is fine – you will learn a lot through the process. Even as you finish the game you will probably think of a much better way to do it - the newest bits of the code will likely be the ‘nicest!’

# Possibly useful concepts/skills

Here are some key ideas and words that might be worth looking into as you proceed:

* **COMMENTING:** get in the habit of adding comments to lines in your code, so you and others can understand what your code does.
* **Object-oriented code** (see Week 2 challenges) – this will be especially useful for creating lots of random hazards.
* **ArrayLists** – create a list of variables or objects, useful when you don’t know exactly how many of an object is going to be on screen at the same time
* Using **pushMatrix() and popMatrix()** to change the location and direction of the coordinate system
* Creating graphics in Paint, Photoshop or the like and using these in a program – see **PImage**.
* **for** loops and **if** statements
* simulating gravity
* simple **collision detection** (circle vs circle, or point vs circle)
* handling **strings** and displaying **text**