# Ludic Computing coursework 2011-12

# Experiments with Steering Behaviours in Processing

## Overview

This coursework should be done in pairs. It will get you to implement and experiment with the steering behaviours that we introduced in the lectures, using the Processing language and development environment. You will be required to implement four separate Processing programs (or 'sketches') based on the example program provided. There are also three short experimental investigations to perform. A concise write up of each experiment (method, results, conclusions) should be included in a report. Using Processing requires some fairly basic Java programming.

## Submission

Both paper AND electronic submissions are required in CATE. Please ensure the following are submitted before or on **Thursday 8th March**:

- A report containing a write up of your experiments (**paper only**).
- A ZIP archive containing your four Processing sketches (**electronic only**).

## Feedback

Feedback will be given to the class during the lectures on Thursday March 15th.

## Getting Started with Processing

Processing is a programming language and development environment built on top of Java. It aims to make graphics programming accessible to visual artists and designers. You download for Linux, OS X or Windows from processing.org/download/ (it should run fine on the lab computers) and read how to write your first sketch at processing.org/learning/gettingstarted/.

An excellent range of tutorials can be found on the website, but they're mostly aimed at those from a non-programming background, and mostly cover topics unrelated to the coursework. As you're familiar with Java you'll probably find it more rewarding to maybe skim a few tutorials, take a look at some examples (see File > Examples) and then move on to the reference manual (processing.org/reference/) while completing the coursework. There are various existing implementations of steering behaviours in Processing flocking is especially popular. Daniel Shiffman's work is particularly worth looking at (http://www.shiffman.net/teaching/nature/steering/). You may (of course) have a look at his code, but remember that (of course) your submitted code must be your own.

Now download the coursework sketchbook from the course webpage and place the folder in your local sketchbook (which you can locate through Preferences). The Steering folder should contain four subfolders: three empty and one containing the SeekDemo sketch, a demonstration of the Seek steering behaviour. Note that Processing sketches consist of a folder Name containing a main file Name.pde. You may optionally write supporting inner classes in Name.pde or other .pde files. Processing basically wraps all this code into an applet when you press ‘Play’.

## Seek and Arrive

*Experiment 1.1:* Start SeekDemo. Click in the display window to move the agent’s target. Investigate the effects on Seek of varying the mass, maximum force and maximum speed. For each parameter sample a small number of values between 1 and 25.

*Sketch 1:* Edit the SeekDemo sketch so that another agent simultaneously steers toward at the same target using the Arrive steering behaviour. The two agents should not physically interact, so they may occupy the same space. The user should be able to control the stopping distance via the keyboard.

Important: any new keyboard controls you introduce should be documented on the onscreen information display.

*Experiment 1.2:* Investigate the effects on Arrive of varying the agent’s parameters, including the stopping distance.

## Wander

*Sketch 2:* Create a new sketch in the WanderDemo folder (it’s easiest to copy over SeekDemo PDE files and use these as a starting point). Implement a Wander steering behaviour and have the agent use it. The user should be able to control the target jitter, wander radius and distance via the keyboard.

*Experiment 2:* Investigate the effects on Wander of varying the jitter, radius and distance over a range of values.

## Pursue and Evade

*Sketch 3:* Create a new sketch in the HuntDemo folder which has a hunter agent trying to catch a prey agent. The user should be able to control and view information on the following:

- whether the hunter agent uses Seek or Pursue;
- whether the prey agent uses Flee or Evade;
- the relevant parameters for both agents.

Pursue and Evade should use the "Time to Target" formula for the look-ahead time.

*Experiment 3:* Set the parameters so that both agents are identical in mass etc., except with the hunter having a slightly higher maximum speed. It should be possible for the hunter to catch the prey, but not always straight away. Now investigate how well the steering behaviours perform against each other. You should quantify your answer, e.g. in terms of mean capture times. (Hint: have the demo record useful information and write it to a file.)

## A Game of Tag

*Sketch 4:* Create a new sketch in the TagDemo folder in which a number of agents (at least 4) play a game of tag. You may use any steering behaviours you wish to implement the agents. The aim is make the agents collective behaviour look as realistic as possible. In tag, one player is designated 'it' and attempts to catch the other players. When an player is touched by the 'it', the 'it' status moves to that player. Some things to keep in mind:

- When a player is caught they normally give the other players the chance to move away before they chase them.
- Players don't constantly chase or run away: when they're a good distance from threats/ targets they move around more slowly, in a less directed fashion.
- The 'it' usually chooses and chases a single target, but may switch when another target presents itself.
- It will look more realistic if you prevent agents from overlapping.

Describe your TagDemo agents and their behaviour in the written report.