Task 11 Lab: Steering 1- Seek, Arrive, Flee

Summary

The aim of this lab is to use and extend a basic model of steering movement for autonomously moving agents. The behaviours of ***Seek***, ***Arrive*** and ***Flee*** will be used and will form the basis of other *compound* steering behaviours used in later work.

By the end of this lab you should be familiar with, able to describe, and use the Seek, Arrive and Flee steering behaviours, and ready to extend the model presented in later tasks.

Install pyglet and Download Lab Code

Make sure python is installed. Make sure the pyglet module is installed.
Download the code for this lab (zip file) from the unit website and extract to your task folder.

Explore

If all is working ok, you should be able to execute the script and see an agent "steering" through a simple toroid game space. You can change steering mode by pressing keys 1 to 5. Not all keys work at the moment (see later steps). The agent code initially has only `seek()` and `arrive()` for one speed setting and you'll be adding more to this.

Step 1

Have a look at the code:

- Identify the classes/objects used, and in particular understand "who contains what".
- Draw a simple UML style model on paper (or digital) to represent this for yourself and check this with your tutor to make sure your understanding is correct.
  This will save you tears - honest!

Note: Two hashes ## have been added to the code to help identify some of the areas you need to modify or change in Step 2. Your editor "Search" feature is your friend!

Step 2 - Flee!

Add the following features to the code:

- Be able to add additional agents (or multiples) to the world by pressing a key. (See main.py and how the first agent is added to the world. Then look at `on_key_press`.)
- Complete the `flee()` behaviour. It is very similar to the seek() code - copy and adapt.
- Add a "panic distance" to the `flee()` code so that it only "kicks in" when the agent is close to the flee location.
- Add additional arrive deceleration speeds. (There should be three. You could add more.)
- Change the physical properties of the agents. For example, `mass`, `max_speed`… friction?

Extensions

- Try adding "`pursuit`()" so that one moving agent will pursue another moving agent. The world already has some variables ready to help with this. You will need to nominate the agent your other agent(s) will pursue.
- Just for fun, you could update the shape of the agent.
- Consider and implement different limits on forward, side and reverse steering force (acceleration) instead of a single uniform limit.