Exercise – Steering Behaviours

Exercise 1:

Based on the lecture notes attempt to implement a Behaviour to handle Steering Behaviours.

An example of how this could be done is as follows:

* Extend the **GameObject** class from previous sessions to include a **Velocity** that is applied to the object’s **Position** each update. Also add an **addForce()** method to the **GameObject** that allows us to add to the **Velocity**. The **GameObject** should also contain a variable for **MaxVelocity** and **MaxForce**. Each **update()** the **GameObject** should ensure that the magnitude of its **Velocity** does not exceed **MaxVelocity**.
* Implement a **SteeringForce** class. This class is an abstract base class that has a single pure virtual method, **getForce()**, that takes a **GameObject** pointer as a parameter and returns a force represented by two floats.
* Implement a **SteeringBehaviour** behaviour that derives from the **IBehaviour** class from a previous topic. This class should contain a pointer to a **SteeringForce** object. The **SteeringBehaviour’s execute()** method would calculate a force to add to the **GameObject** it is currently updating by passing the **GameObject** into the **SteeringForce’s** **getForce()** method and using the returned value as the force to add to the **GameObject**.

Exercise 2:

Implement a **SeekForce** that derives from **SteeringForce** by implementing the Seek behaviour from the presentation. Its **getForce()** method should create a force to apply to the passed in **GameObject** that drives it towards a target **GameObject**.

The returned force should be clamped so that its magnitude is equal to the **GameOBject’s** **MaxForce**.

Set up a project that has a user-controlled “player” **GameObject** and an “enemy” **GameObject** that has the **SteeringBehaviour** applied to it with a **SeekForce** driving it towards the player-controlled object.

Exercise 3:

Based on the lecture notes, implement a **FleeForce** that derives from **SteeringForce** that implements the opposite of the **SeekForce** behaviour.

Change the project so that the “enemy” can toggle between using the **SeekForce** and the **FleeForce** for its **SteeringBehaviour**.

Appendix:

As a refresher on Vector maths, you can query a vector’s magnitude with the following code:

float magnitude = sqrt( x \* x + y \* y );

You can normalise a vector with the following code:

x /= magnitude;

y /= magnitude;

ty

You can also clamp a vector to a certain magnitude with the following code:

// after normalising the vector

x \*= desiredMagnitude;

y \*= desiredMagnitude;

ty

References:

* The author of these techniques is Craig Reynolds, see his website [here](http://www.red3d.com/cwr/).
* A great tutorial for revision on the Seek behaviour can be found [here](http://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-seek--gamedev-849).
* A great tutorial for revision on the Flee behaviour can be found [here](http://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-flee-and-arrival--gamedev-1303).
* A great tutorial for revision on the Wander behaviour can be found [here](http://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-wander--gamedev-1624).