Programming Assignment #1 Overview

* Module – Access Assignment and Discussion
* General Instructions / Rubric
* Specific Instructions (PDF)
* Files
* R Code
* Plotter
* Flow (example):
  + Setup / Initialization
  + Define Classes: e.g., SteeringOutput, Character
  + Define Helper Functions: e.g., length of vector, normalization of vector
  + Define Movement Functions: update, getSteeringContinue, getSteeringSeek, etc.
  + Set up 4 characters w/initial conditions, print initial conditions @ time=0
  + Main loop

Common Trends from last Semester:

* Every submission I have looked at had the right "bones" to start with
* Syntax, global vs. local scope, use of classes and other data structures
* Several **used the simulation time** (e.g., the i counter in the loop) instead of the ΔT
  + If you had velocities that kept getting bigger and bigger, this was the likely culprit. Remember the time step!
* **Related: using a loop of 1-100**, and dividing that by 2 to print out the time step
  + Not entirely correct, but I didn't deduct anything unless it affected the execution
* **Mixing up 'target' and 'targetSpeed'**
  + Remember the confusing language in the Millington pseudocode?
* **Over-complication**
  + Each steering behavior and movement update could just about be copied from the pseudocode in the text
  + Separate library / class files are good for future expansion / enhancement, but could introduce complexity
* Not printing **initial conditions**
* Forgetting to check if velocity and acceleration are above Max, **then clipping** if so
  + This is another cause of "runaway" trajectories
* **Splitting vectors** into X and Z components is ok, but you have to be careful in how you handle it
  + e.g., when checking the length of the velocity vector, you can't compare each component to the max Velocity for the character
* Linear accelerations not being printed to output
  + Results in no blue lines on plot, but trajectory still works
  + This is because linear acceleration is **only** used to update velocity, which is used to update position
  + If you aren't storing that somewhere it gets lost

**Some tips for tackling projects like these**

* Plan out the larger structure / flow, but don't implement all at once
* Break the project into smaller, easier pieces
* Build helper functions and classes to reuse code and make things easier
* Unit test - make sure functions are doing what you want them to do, and are doing the right thing
* Look at what's given to you, what libraries are available, etc.
* KISS principle
* Don't fall in love with a bad plan
* If something isn't working right, isolate functionalities to determine the issue(s)