CSC 584/484 Spring 17 Homework 1: Movement

Due: 2/15/17 by the start of class

Overview

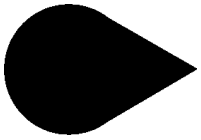
Your task for this assignment is to explore various movement algorithms. Using *Processing*, you will implement various dynamic movement algorithms and compare their performance. You will submit your code and 3–5 page writeup of your results, including screenshots where appropriate.

This is an ***individual assignment***, you are to work alone. As always, you are expected to abide by the University's Academic Integrity Policy (`http://policies.ncsu.edu/policy/pol-11-35-01`), which includes providing appropriate attribution for all external sources of information consulted while working on this assignment.

First Steps

You should begin by familiarizing yourself with Processing (`http://www.processing.org`). "Processing is an open source programming language and environment for people who want to create images, animations, and interactions." Processing contains an IDE and runtime execution environment that is free to download. As of the time this assignment was given, Processing version 3.2.3 is current. Please upgrade to 3.2.3 if you are currently using an earlier version from previous semester. If you're more comfortable, you can create Java classes that extend the base Processing class to work in an IDE you are more comfortable with. Keep in mind that we will need to compile and run your code to grade it, so make sure there are no special requirements that might prevent us from doing so. Specific instructions on how to build a processing `PApplet` with *Eclipse* can be found here: `http://processing.org/learning/eclipse/`.

Start by drawing a circle and triangle to the screen to make the following shape:



Make sure your shape is sufficiently small relative to the canvas size so that when you begin moving it around the screen it has a place to go.

Kinematic Motion (5pts)

Create a data structure to hold your steering and kinematic variables like the one we discussed in class. Make sure you include methods for getting the orientation from direction of motion, getting the direction of motion from orientation, and for applying the standard update.

Using kinematic motion, have your shape start at the bottom left corner of the screen, and traverse around the edge until it returns to its starting location. *Make sure to leave breadcrumbs as your shape traverses the screen.* Save your code and label it "basic-motion". It is probably a good idea to take a screenshot of your shape and the breadcrumbs after it completes its tour for inclusion in your writeup.

Arrive Steering Behaviors (15pts)

Implement the arrive algorithm we covered in class. Have your shape arrive at the location of mouse clicks. You can implement this by using Processing's `mousePressed()` method. Make sure your shape is oriented in the direction of travel. Recall that you can accomplish this by implementing an orientation matching steering behavior. Test out at least two different methods for implementing arriving (this may include, but is not limited two, two different sets of parameters for the arrive algorithm). Which looks better? Why? Which is more successful? Why? It is probably a good idea to take a series of screenshots to illustrate your results. Save and label this file as "arrive-steering".

Wander Steering Behaviors (15pts)

Implement the wander algorithm we covered in class. Make sure your shape is oriented in the direction of travel and to handle boundary violations gracefully. Implement at least two different methods for changing orientation. Which one looks better? Why? It is probably a good idea to take a series of screenshots to illustrate your results. Save and label this file as "wander-steering".

Flocking Behavior and Blending/Arbitration (25pts)

Using multiple independent shapes, implement flocking behavior using the standard Boids algorithm (Reynolds 87). The Boids algorithm includes *separation* (avoid collisions with nearby neighbors), *alignment* (velocity match center of mass of nearby neighbors), and *cohesion* (position match center of mass of nearby neighbors). You will have to blend these behaviors intelligently, using parameters of your choosing. You may implement other steering behaviors of your choosing, or invent your own. Just make it look as good as you can.

What tweaks did you make to the algorithms to make flocking work? Are things easier or harder with more followers? (Note, you should probably run experiments using varying numbers of followers.) What happens if you have two wanders and followers follow the closest wanderer? Don't forget to take screenshots to support your analysis. It is probably a good idea to take a series of screenshots to illustrate your results. Save and label this file as "flocking-behavior".

Writeup (40pts)

Now that you have implemented and evaluated a number of algorithms, write a 3–5 page single-spaced paper summarizing your findings. Note, 3–5 pages means at least ***three full pages***. It is ***strongly*** recommended that you do not limit yourself to only answering those questions posed in this assignment. Think creatively about what you have done. What other parameters can you tweak and what effect do they have on the results? The most successful writeups will contain evidence that you have thought deeply about these algorithms and what they can produce and have gone beyond what is written in the assignment.

As an appendix to your paper, please include all relevant screenshots to support your analysis. The appendix does not count toward your 3–5 page requirement.

What to submit

By the start of class on 2/15/17, please upload a .zip archive to Moodle. This archive should contain all of your appropriately labeled files from each part of the assignment, plus your writeup in .pdf format. You

should also include a README text file containing a description of each of your files, and any special compilation instructions.