CSC 584/484 Spring 22 Homework 2: Steering Behaviors

Due: 02/22/22 by the start of class

Overview

Your task for this assignment is to explore various movement algorithms. Using `SFML`, you will implement various dynamic movement algorithms and compare their performance. You will submit your code and 2–3 page writeup of your results, including screenshots where appropriate. This assignment is worth 30% of your homework grade.

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.

Variable Matching Steering Behaviors (3pts)

Your task for this part of the assignment is to build a library of four variable matching steering behaviors. To facilitate efficient implementation of these behaviors, you must:

1. Create a pure virtual `SteeringBehavior` class.[1]
2. Create four separate subclasses of this pure virtual class to match each of the four kinematic variables (position, orientation, velocity, and rotation).

You will demonstrate position and orientation matching in the next section. To demonstrate velocity matching, have the character match the velocity of the mouse pointer. You will need to sample the mouse pointer's location periodically and convert it to velocity by dividing the distance by time, put that in a kinematic data structure, and pass it in as a parameter to the velocity matching behavior.

Arrive and Align (5pts)

Implement the arrive and align algorithms we covered in class. Have your shape arrive at the location of mouse clicks and use align to smoothly orient in the direction of motion. Check the documentation of SFML for how to determine mouse click locations. Test out at least two different methods for implementing arriving and aligning (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?

Take a series of screenshots to illustrate your results. To facilitate the screenshots, you must implement breadcrumbs. Keep a fixed-length queue of n locations sampled every i iterations, and draw small circles on those locations. The exact values for n and i will depend on how you set your max velocity/rotation, so experiment to find values that enable you to illustrate motion in your screenshots.

Wander Steering Behaviors (3pts)

Implement a wander algorithm. You may choose to implement one of the algorithms we covered in class, or invent something new—it's up to you. Make sure your shape is oriented in the direction of travel and to

[1] There are ample resources on google describing virtual classes in C++. For our purposes, this class should have a function `public SteeringData calculateAcceleration(Kinematic char, Kinematic goal) = 0;` (or something substantively similar).

handle boundary violations gracefully (*i.e.*, what happens when the shape reaches the edge of the screen). Implement at least two different methods for changing orientation. Which one looks better? Why?

Take a series of screenshots to illustrate your results. To facilitate the screenshots, you must implement breadcrumbs.

Flocking Behavior and Blending/Arbitration (12pts)

Using multiple independent characters, implement flocking behavior using the standard Boids algorithm (Reynolds 87).[2] 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? Note, you should probably run experiments using varying numbers of characters. Don't forget to take screenshots to support your analysis and make sure each character is using breadcrumbs (you may wish to tweak n and i). Take a series of screenshots to illustrate your results.

Writeup (7pts)

Now that you have implemented and evaluated a number of algorithms, write a 2–3 page single-spaced paper summarizing your findings. Note, 2–3 pages means at least ***two 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—and therefore the screenshots—does not count toward your 2–3 page requirement.

What to submit

By the start of class on 02/22/22, please upload a .zip archive to Moodle. This archive should contain all of your code and all Makefiles required to compile in the standard grading environment. Make sure to include a README with explicit instructions for how to compile and demonstrate each part of the assignment. Include your writeup **in .pdf format**. All files should be included in a single .zip file.

[2] Reynolds, Craig W. "Flocks, herds and schools: A distributed behavioral model." In Proceedings of the 14th annual conference on Computer graphics and interactive techniques, pp. 25-34. 1987.