CS 330 Artificial Intelligence and Game Development
Spring 2022

Mikel D. Petty, Ph.D., pettym@uah.edu, 256-824-6140

General Instructions

Instructions that apply to all programming assignments in this course are on this page.
Instructions specific to the individual programming assignments are on the following pages.

Programming and Deliverables

1. Write your program(s) in C++, Java, or Python. Any other language requires instructor pre-approval. R is not allowed. (If you really want to program in R, see me offline.)
2. Use the same algorithms and calculations as shown in the course textbook and lecture slides.
3. Write your code with reasonable attention to software engineering quality. Include explanatory comments as needed. See my R code for examples of commenting that I consider to be sufficient.
4. Combine all deliverables into a single zip file before submission.
5. Include the name(s) of the students who contributed to the assignment in both the main program's comments and in the submission zip file's filename.
6. Submit the zip file containing the deliverables via Canvas file upload.
7. Due dates for the programming assignments are given on the Canvas course Assignments page. Submissions are due at 11:59pm (not 12:00 midnight) on the due date.
8. If you submit late, do not submit via Canvas; send the zip file to me as an email attachment. Send from your official UAH email adress, not Canvas; Canvas messages drop attachments.

Teaming

1. You are permitted and encouraged to work in groups of up to two (2) students on the programming assignments. Both students in a group will receive the same grade.
2. Teams are not required to be permanent. You may change partners, or change from teaming to not teaming, or vice versa, for different assignments.
3. Teaming is not required. Individual submissions are accepted with no grade penalty or bonus.

Recommendations

1. Don't wait until the day before the due date to start the assignments. Allocate enough time to do a good job, and allow time for unexpected difficulties.
2. In previous courses, some students have tried to do a line-by-line port of my example R code into C++, Java, or Python. You are welcome to try that, but it may not be the best approach. It may be better to use my code to understand the algorithms, and then implement the algorithms natively in your chosen programming language.
3. The algorithms in the programming assignments are well known and it is relatively easy to find code for them online. Should anyone be tempted to use code found online, please resist the temptation and instead write your own. Not only are the consequences of plagiarism, if discovered, distinctly undesirable for everyone involved, but more importantly, you will learn a lot more from the experience of writing and debugging your own implementations.

CS 330 Artificial Intelligence and Game Development
Spring 2022
Mikel D. Petty, Ph.D., pettym@uah.edu, 256-824-6140

Programming Assignment 1: Dynamic Movement

Objective

Your primary objective is to implement and test the dynamic movement update and three dynamic movement behaviors, Seek, Flee, and Arrive. Your secondary (optional) objective is to learn how to plot your movement trajectories.

Requirements

Implement the dynamic version of the Newton-Euler-1 movement update algorithm and the dynamic Seek, Flee, and Arrive movement behaviors. These are all described in Chapter 3 of the Millington textbook and Lecture 6. Also implement a Continue movement behavior, which does not alter a character's movement at all, but simply uses the initial values. The Continue behavior is not in the textbook, but it is in my R implementation. For this assignment, the Continue character's initial velocity and rotation will all be 0, so the Continue character should not move at all.

Your program should output each character's trajectory as a text file (.txt). The output file should have one record per character per timestep, including a record for the initial conditions (time = 0). For example, for a scenario with 4 characters that runs for 50 timesteps, there should 204 records in the output file, in the following order. (This list shows proper order, but not proper format for the output file.)

```
character 1, timestep 0
character 2, timestep 0
character 3, timestep 0
character 4, timestep 0
character 1, timestep 1
character 2, timestep 1
character 3, timestep 1
character 4, timestep 1
...
character 1, timestep 50
character 2, timestep 50
character 3, timestep 50
character 4, timestep 50
```

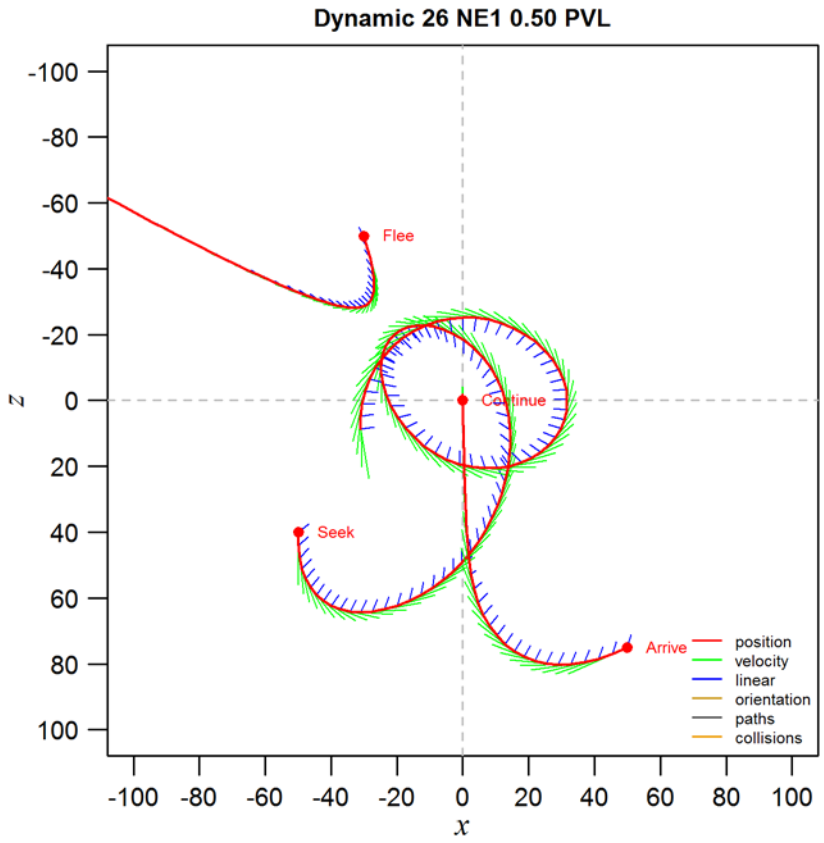
Each record should have the following 10 fields, in the order listed, and separated by commas:

1. simulation time
2. character id (numeric)
3. position *x* (meters)
4. position *z* (meters)
5. velocity *x* (meters per second)
6. velocity *z* (meters per second)
7. linear acceleration *x* (meters per second per second)
8. linear acceleration *z* (meters per second per second)
9. orientation (radians)

10. steering behavior code (1=Continue, 6=Seek, 7=Flee, 8=Arrive)
11. collision status (TRUE if collided, FALSE if not collided; always FALSE for Program 1)

Run your program for 50 simulated seconds with a timestep duration of 0.5 using Newton-Euler-1 integration, i.e., 100 timesteps after initialization. Your scenario should have four characters with the initial conditions in the following table. The movement target of the Flee, Seek, and Arrive characters is the Continue character.

Character number	1	2	3	4
Character id	2601	2602	2603	2604
Steering behavior	Continue	Flee	Seek	Arrive
Initial position	0, 0	-30, -50	-50, -40	50, 75
Initial velocity	0, 0	2, 7	0, 8	-9, 4
Initial orientation	0	π / 4	3π / 2	π
Max velocity	0	8	8	10
Max acceleration	0	1.5	2	2
Target	0	1	1	1
Arrival radius	0	0	0	4
Slowing radius	0	0	0	32
Time to target	0	0	0	1



Replicate, as closely as possible, the character movement trajectories shown in the preceding image. You are not required to implement a trajectory plotting program yourself; I will provide three different plotting programs (see Resources below). In fact, you are not required to plot your trajectories at all (but see recommendation 2 in Assignment-specific Recommendations below).

Deliverables

1. Source code for the dynamic movement program
2. Trajectory data file, with trajectory data for all four characters
3. (Optional) Trajectory plot image, with all four characters

Grading

The assignment is worth a maximum of ten (10) points. It is 10% of your final semester grade, so each point is 1% of your semester grade. The grading rubric is as follows:

Points	Criterion
0	Assignment not submitted or deliverables missing
1	Assignment deliverables all submitted
0-1	Software engineering quality of the program reasonably good
0-2	Movement update present and correct
0-3	Movement behaviors (Seek, Flee, Arrive, Continue) present and correct
0-1	Trajectory data file in proper format and readable by trajectory plotting program
0-2	Trajectories reasonably similar to example trajectories
0-10	Total

The criteria are independent of each other (except, of course, that you cannot get any of the other points if you don't turn the assignment deliverables in). Note that if your trajectory data file is not readable by the trajectory plotting program, the highest score possible may be 7.

Resources

All of the following have been posted to the course Canvas page in Files > Programming Assignments > Program 1: Dynamic Movement:

1. My R code for Dynamic Movement; this consists of 5 code modules, numbered 0-4. Module 3 contains the main timestep loop, the movement update function, and movement behaviors. Module 4 is the trajectory plotting program.
2. Two additional trajectory plotting programs, one written in Python and one written in Matlab. You are welcome to use these trajectory plotting programs if you do not want to learn how to use my R trajectory plotting program. Disclaimer: These programs were written by students, not me. I do not know either Python or Matlab and I cannot support these programs. You may have to modify them to make them work properly.
3. A partial example of the trajectory file for this assignment's scenario generated by my implementation of the assignment movement behaviors, with times 0-4 for all four characters.
4. An example of a complete trajectory file for a different scenario, which you can use while learning to plot trajectories.
5. An image of the complete trajectories for this assignment, i.e., the same image as above.

6. A list of Frequently Asked Questions (and answers to them) for Program 1, collected from previous iterations of this course.

Due date and time

See Assignments page in the course Canvas website.

Assignment-specific recommendations
(See the general instructions for general recommendations.)

1. Implement and test the movement behaviors one at a time. Start with Continue. Once you get Seek working, Flee is a trivial change. Then the code for Seek can be enhanced for Arrive.
2. Test your program by plotting your trajectory data file with one of the trajectory plotting programs that have been posted to Canvas (see Resources above). I may input your deliverable trajectory data file into my plotting program when grading your assignment. If your trajectories do not resemble mine, or if your data file is not properly formatted and thus unreadable by my program, you may not know it unless you attempt to generate the plots.
3. If you decide to team, one team member should immediately start by figuring out how to plot the output trajectory data file, using my example trajectory data file. If you are working alone, you should do that first. Seeing a plot of your trajectories will make testing and debugging the movement behaviors much easier.

End of Programming Assignment 1