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 acceptable commenting.
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 address, 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.

General 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. You will learn a lot more by writing and debugging your own implementations.
4. I've provided my R implementations of the programming assignments as an aide to learning. Should anyone be tempted to run my code and turn in the resulting output as his/her own work, please resist the temptation. There is a way to recognize the output files produced by my programs. You will learn a lot more by 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 2: Path Following

Objective

Your objective is to implement and test the dynamic Follow path behavior.

Requirements

Implement and test the dynamic Follow path behavior using the "chase the rabbit" algorithm, as presented in the textbook (section 3.3.12) and the slides (Lecture 8), and as implemented in my example R code. Note that implementing the Follow path behavior will require also implementing several supporting geometry functions, as described in Lecture 8 and also implemented in my example R code in program `CS 330, Dynamic movement, 1 Support v8.r`.

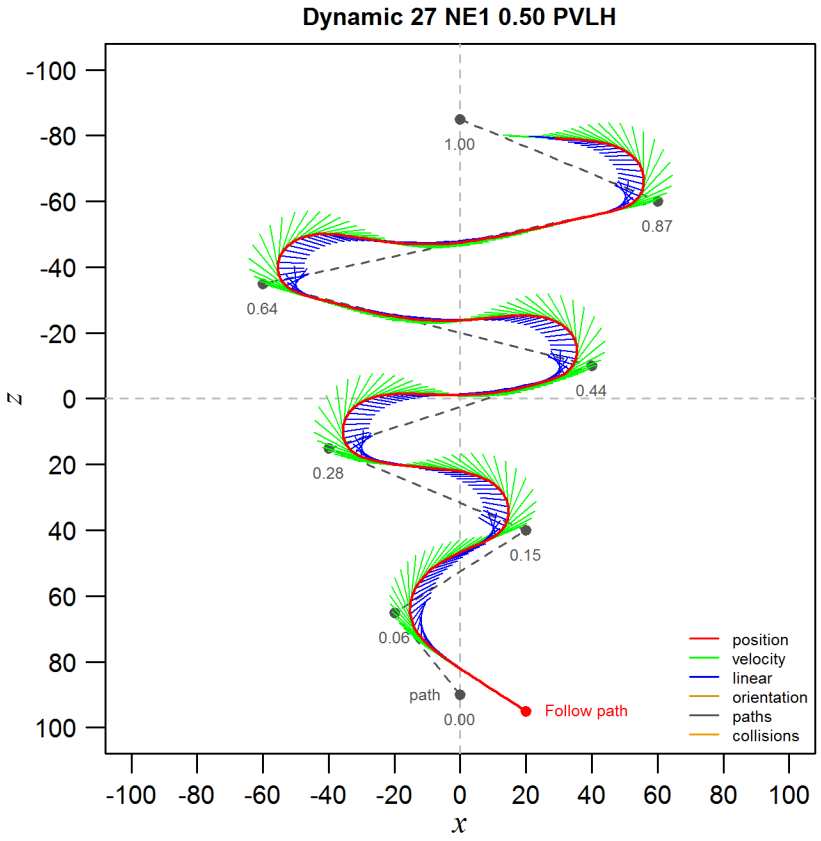
Your implementation of Follow path should be added to your code for Program 1. The Follow path behavior depends on and reuses the dynamic Seek behavior, movement update, and other functionality you implemented in Program 1.

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). The format of the output text file is the same as Program 1.

Run your program for 125 simulated seconds with a time step duration of 0.5 using Newton-Euler-1 integration. Your scenario should have one character with these initial conditions:

Character number	1
Character id	2701
Steering behavior	Follow path
Initial position	20, 95
Initial velocity	0, 0
Initial orientation	0
Max velocity	4
Max acceleration	2
Path to follow	1
Path offset	0.04

The path your character should follow has eight vertices. From start to end, they are:
(0, 90) (-20, 65), (20, 40), (-40, 15), (40, -10), (-60, -35), (60, -60), (0, -85)



Replicate, as closely as possible, the character movement trajectory 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, including Path following
2. Trajectory data file, with trajectory data for the one character
3. (Optional) Trajectory plot image, with the one character

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
0-1	Deliverables all submitted
0-2	Software engineering quality of the program reasonably good
0-1	Movement update present and correct
0-2	Movement behavior (Follow path) present and correct
0-1	Supporting geometry and path functions present and correct
0-1	Trajectory data file in proper format and readable by trajectory plotting program

0-2	Trajectory reasonably similar to example trajectory
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 2: Path Following:

1. My R code for Dynamic Movement. I have made some small changes to the code since Program 1, so this code differs slightly from what I posted for Program 1.
2. Two additional trajectory plotting programs, one written in Python and one written in Matlab. Both of these programs were recently enhanced by students who used them for Program 1, and the new versions are in the Program 2 folder. 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.
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 the one character.
4. An image of the complete trajectories for this assignment, i.e., the same image as above.

Due date

See Assignments page in the course Canvas website.

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

1. If your Program 1 was not working properly, begin this assignment by getting it working. This programming assignment depends on the Seek movement behavior and the movement update function from Program 1 both working correctly.
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 trajectory does 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 implementing the supporting geometry functions and testing them separately. Until they are correct, the Follow path behavior will not work correctly.

End of Programming Assignment 2