CS 330 Artificial Intelligence and Game Development
Spring 2022

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

Programming Assignment Frequently Asked Questions

Program 1: Dynamic Movement

General questions

Q1. The textbook and lectures explain the details of the movement algorithms, but I'm not clear on of the overall structure of the program. Can you clarify?

A1. This informal pseudocode is a high-level view of the program's overall structure:

```
 1   initialize data structures for characters 1, 2, ..., 4 with scenario data
 2   for each character i
 3       write initial movement data for character i to output trajectory file
 4   endfor
 5   for each timestep t
 6       for each character i
 7           call get.steering function for character i to produce steering output
 8           call movement update function to update position, velocity, ...
                 for character i using steering output produced by get.steering
 9           write updated movement data for character i to output trajectory file
10       endfor
11   endfor
```

For line 7 in the above pseudocode, the get.steering function appropriate to the character should be called, i.e., call Seek for the Seek character, Flee for the Flee character, etc. The Seek, Flee, and Arrive get.steering functions are explained in Lecture 6. All of the get.steering functions produce an output in the same format; that output format is also in Lecture 6. For line 8 in the above pseudocode, the movement update routine is common to all of the get.steering functions, i.e., there is only one movement update routine, not one per get.steering; it is explained in both Lectures 4 and 6.

This structure corresponds precisely to my R implementation; for example, the timestep loop at line 5 in the pseudocode above starts at line 322 in code module 3 Move v16.r.

Q2. Does my program need to read the initial data for the characters from a file?

A2. You can do that if you wish, but it is not required. There are only four characters in the Program 1 scenario so in your program you can simply initialize your character data structure with hard coded values if you wish. For example, in C you could create four structs, one per character, with the same variables in each structure but different values for those variables. In

C++ you could initialize four instances of a character class with the initial values. (FYI: You will have to read initialization data from a file in Program 3.)

Q3. How do I represent vectors in C++?

A3. I believe that there may be a C++ library class for vectors, but I don't know anything about it. As an alternative, some students in prior iterations of CS 330 have adopted a simple but effective expedient. Because nearly all of the vectors we will use are 2D, i.e., 2-element vectors, you can represent each vector with two separate variables. For example, position is a 2-element vector, [*x*, *z*]. You can represent position with two floats, `position.x` and `position.z`.

A slightly more sophisticated approach is to use a 1-dimensional array with 2 elements. The latter is what my R code does, even though it may not be obvious from the code; in R, all variables are vectors (1-dimensional arrays) by default.

Both of these approaches require you to implement vector operations, such as scalar multiplication or dot product that handle both variables correctly.

In Program 1, when writing vectors, e.g., position, which is a 2-element vector, to the output trajectory file, you don't need to write a vector as a vector. Rather simply write each element of the vector individually as one of the fields in the output record.

Questions about the R code

Q4. The R code has many movement behaviors and features not mentioned in the Program 1 assignment document. Are those required?

A4. You are only required to implement the behaviors and features needed to meet the requirements for Program 1. For example, you will need to implement the Newton-Euler-1 version of the movement update function, but you are not required to implement the High School Physics version.

Q5. I don't know R and I am confused by the R code. Do I have to use it?

A5. **You do not have to use or even look at my R code.** You will not be tested on it. It is intended only to assist you if you want to use it. If you prefer, you can ignore the R code and write the program from the pseudocode in the book, which is what I did.

Q6. In the `dynamic.get.steering.x` functions, what are variables `mover` and `target`?

A6. Variable `mover` and `target` are the moving character and movement target respectively. They are input parameters to the `dynamic.get.steering.x` functions. They are assigned values when the function is called, just like a function in other programming languages. They contain R lists, which are similar to C structs; each contains several fields for the character. (See Q8 for more details on the latter point.)

For example, at line 87 in code module 3 Movement v16.r, this is the first line of `dynamic.get.steering.seek`:

```
dynamic.get.steering.seek <- function(mover, target) {
```

Function `dynamic.get.steering.seek` is called at line 355:

```
steering <- dynamic.get.steering.seek(Character[[i]],
Character[[target]])
```

When line 355 calls `dynamic.get.steering.seek`, `Character[[i]]` is assigned to `mover` and `Character[[target]]` is assigned to `target`. R is call-by-value, so the values of `Character[[i]]` and `Character[[target]]` are copied into `mover` and `target`.

Q7. What is variable `Character`?

A7. Variable `Character` is a list data structure that contains one element for each character in the scenario. For program 1 there are four characters, executing Continue, Seek, Flee, and Arrive. Thus in Program 1, `Character` will have four elements, one for each character, i.e., `Character[[1]]`, `Character[[2]]`, `Character[[3]]`, and `Character[[4]]`.
Each character in the `Character` list can be accessed by index, e.g.,
in `Character[[i]]` and `Character[[target]]`, `i` and `target` are indexes, i.e., integers.
`Character[[i]]` is the character at index `i` in the `Character` list and
`Character[[target]]` is the character at index `target` in the Character list.

Q8. What are the elements of variable `Character`?

A8. Each character, i.e., element, in the `Character` list is a list data structure that contains all of the individual variables for one character, e.g., position, orientation, steering behavior, ...
Those individual variables with a character's list can be accessed with the `$` operator.
For example, in my code `Character[[i]]$steer` is the `steer` variable for character `i` in the `Character` list. The data structure for one character in my R code is itself a list, i.e., `Character` is a list of lists in my program. However, in other programming languages each character could be an instance of a character class (C++) or a struct (C).

End of document