
CSC 211 – Introductory Programming and Design

Laboratory Assignment 10: *Not Alone Anymore*

Monday, April 07, 2008

Due date — Section 1: Wednesday, April 16, at 1 PM
Section 2: Tuesday, April 15, at 1 PM.

1 About this Assignment

1.1 Objectives of the assignment

Today's lab features again our bottle-picking barbarian. Today we will

- add monsters who wander around in the world
- implement a `GridSquare` class;
- implement the path planning algorithm.

Read very carefully the text of this assignment before rushing to type code. Read it completely through once. Do not stop at the first unclear sentence you encounter; sometimes things are explained and detailed a bit later. Then you should start all over again. This time stop to ask questions when a point remains fuzzy, code your solution, and move on to the next section.

1.2 Handouts

This week's sole handout is the text of this assignment as a `.pdf` file.

2 What to Do, Part I: The `GridSquare` class

2.1 About grid squares

In this revised version of the application, I would like each grid square to maintain its own state: is it occupied by an obstacle or is it a free space square, and then possibly, what type of *terrain* does it correspond to (indoor, field, forest, etc.)? What objects are in the square?

So we are going to implement grid squares as objects of a class `GridSquare`, and our game will store a reference to a 2D array of `GridSquare` objects.

2.2 Methods of this class

It may be a good idea to implement some of your path planning as methods in your `GridSquare` class. For example, a monster (more on that later) who knows its current location on the grid (i.e., the `GridSquare` object it stands on) might want to invoke a method `nextMove` that would take as parameter a reference to a destination `GridSquare` and returns a reference to the first step in that direction (when possible).

3 What to Do, Part II: Monsters

3.1 Monsters vs. barbarians

What makes a monster different from a barbarian? Not much. Both wander around (more on that later) and pickup objects ((more on that later). The only difference is that the barbarian's decision are taken by the player while the monster is driven by an AI. This means that you can recycle a great deal (most, in fact) of your `Barbarian` class into a `Monster` class (until we learn about a better way of doing that next week).

3.2 Different kinds of monsters

When you generate a monster, the constructor should assign to this monster one of three possible kinds:

- aggressive: This monster will always move towards the `Barbarian` if within N squares of him/her (N is a constant of your game).
- cowardly: This monster will always run away from the `Barbarian` if within N squares of him/her.
- wanderer: This monster picks a random destination and tries to reach it. When it gets there, it picks another destination.

Your monster should also be assigned one of three possible behaviors with respect to bottles:

- drinker: The monster picks up any bottle it finds, drinks all of its content, and then drops it;
- collector: The monster can only carry M bottles, so it only keeps the ones that are the most full. If it encounters a new bottle, it compares it to the ones it was carrying.
- absent-minded: This monster has an $n\%$ probability of dropping one of its bottles each turn of the game.

4 What to Hand in

4.1 End-of-session evaluation

You are not expected to complete the assignment by the end of the lab session, but you are definitely expected to have done *some* work during that session. Try to use the lab session to make sure you understand everything about the assignment. Ask questions; try things; ask more questions.

You should *not* leave the lab before your work has been *evaluated*. This first evaluation is worth 10 pts out of 100 for the complete assignment. If you leave before you have been evaluated, these points are lost with no chance of a later evaluation.

4.2 Your project

A cleaned-up and properly named project folder containing all your source files and your report should be uploaded to your CSC211 submit folder. Do *not* include the documentation produced by javadoc to this folder.

5 How You Will Be Evaluated

5.1 Point distribution

The maximum number of points is 100, but extra points could be awarded for excellent aspects of the project or report. The point distribution for this assignment is as follows:

Execution evaluation

End-of-session evaluation	10 pts
Execution of the project handed in	30 pts

Source Code

Identifier names	5 pts
Good indentation and general readability	5 pts
Judicious comments within the code	10 pts
Javadoc comments	10 pts
Follows lab specifications	10 pts

Report

Discussion	10 pts
General quality of the writing and presentation	10 pts

If you submit a project late, then it is your responsibility to notify the TA (with CC. to the instructor) that the project is finally available in your submit folder. If you fail to do so, then the “late penalty clock” will keep ticking until the TA gets around to checking your folder and notices your project. Unless specifically asked to do so, do *not* mail your project folder as an attachment.