Zombieland

INGI1131 Course project (2014)

April 19, 2014

Zhongmiao Li
(*zhongmiao.li@uclouvain.be*)

Manuel Bravo
(*angel.bravo@uclouvain.be*)

Year 2023. My name is Michael Philip Jagger and I am one of the few survivors of the end of the world. Yes, you heard properly, I survived the end of the world. As Jean Dixon, well-known american astrologer and psychic of the 20th century, predicted, the world as we knew it until then disappeared. She claimed that Armageddon would take place in 2020 and Jesus would return to defeat the unholy. Unfortunately, she was not completely right. The end of the world only brought what we now know as zombies.

An explosion in one of the most-secret government laboratories of the United States during the evening of the second monday of March of 2020 triggered the apocalypse. It is not completely clear what they were investigating; nonetheless, the few information we have makes us think that they were trying to create stronger, faster and fearless species, most probably for military purposes. Apparently, they tested it too early and something went wrong. The "product" as we named the substance they created not only upgrades physical capacities, but it also transforms the living into brainless and cannibal species. In consequence, differently to how the zombies were pictured in the science fiction literature, the zombies from which we escape are stronger and faster than us. The disease was rapidly globally spread. If some of us were able to survive so long was due to luck and the sacrifice of others.

When the disease was not so spread, we were able to gather quite large amount of food and weapons. For the last three years, we have been living in an old school in the east of Cardiff, Wales. We have vigilants 24 hours per day, 7 day per week. Although we are a small group (around 12 people right now), we are starting to run out of food and bullets. We have been forced to leave our safe place and move around the city to find food and bullets. Most of the people who left never came back. Unless we obtain food and bullets soon, this place will not be safe anymore.

We need your help. We know about multiple locations where we could get food, weapons and medicines. We want to create a simulator that would help us to take into account all the possible unexpected events we will find there and considerably reduce the number of human losses. We have been studying the zombies for a while so we can precisely tell you how they move and behave.

1. The simulator

You will have to model the following components:

- The room that contains the food, medicines and bullets.
- "The Brave" is the human that tries to collect food, bullets and medicines.
- The zombies that are trying to eat the brave.

A simulation has four inputs: the room, the number of zombies, the number of objects that "The Brave" has to collect before leaving the room and the number of bullets "the Brave" initially has. The number of bullets left should be displayed at all time. The zombies will be placed randomly before the simulation starts based on the room disposal.

The simulation finishes whenever "The Brave" collects at least the number of objects required and leaves the room (using the door) or if a zombie kills "The Brave".

1.1. The room

The room is represented by a matrix. It is one of the parameters of the simulator. Each cell of the matrix contains an integer that represents what would be in the real room. The possible initial values for a cell are the following:

- 0: Empty space
- 1: Inaccessible space
- 2: A pack of 3 bullets
- 3: Food can
- 4: Medicine
- 5: Door

The room can be of any size with any contents. The only restriction is that it has to contain exactly one door, and this, cannot be in a corner. An example of a room could be the following:

```
map( r(1 1 1 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1)
     r(1 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 1)
     r(1 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1)
```

```
r(1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 1)
r(1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1)
r(1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1)
r(1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1)
r(1 0 3 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 1)
r(1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1)
r(1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1)
r(1 0 4 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1)
r(1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 4 0 1)
r(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
```

Please, do not confuse a cell of the matrix with a cell in Oz. A matrix cell is only an element of the matrix. Remember, you are not allowed to use Oz cells.

1.2. “The Brave”

“The Brave” represents the person that gets into the room and tries to collect as much food, bullets and medicine as possible. He/she can at most move two cells per turn. Nevertheless, if a cell contains an object, collecting it costs one. In other words, if in a turn, “The Brave” want to collect a food can, he would only be able to move one cell.

1.3. The zombies

The zombies are spread across the room. They are dumb but they move fast. They can at most move three cells per turn. Nevertheless, they do not move smartly. They always try to move three cells in the direction they did previous move. Thus, they move straight until an obstacle is found, then they randomly change their direction.

Furthermore, they do not normally grab the objects they find. Nevertheless, from time to time, they could become lucid and destroy the object they found on their way (some probability, let's say 20% of chance), making the task of “The Brave” even more difficult if possible. The way a zombie destroys an object is the same as how “The Brave” collects an object, i.e. destroying an object would cost a zombie one move.

1.3. Interaction between components

Cells can only be occupied by either “The Brave” or a zombie. "The Brave" should not overlap with a zombie and neither can zombies overlap with each other. The component room should take care of this. The food can, the bullets and the medicines do not occupy the cell. These can only be consumed/destroyed once. Thus, “The Brave” does not only need to collect the objects, but it also has to do it before the zombies destroy them all.

It is important to mention that both “The Brave” and the zombies cannot move diagonally.

“The Brave” enters and leaves the room through the door. That’s why it is important that the room always has a door.

The turns should be synchronized. For instance, a zombie cannot use two turns while “The Brave” (or any other zombie) has not use any. There is no need to use timers for representing the turns.

If a zombie and “The Brave” happen to be in contiguous cells, multiple things might happen:

- If “The Brave” does not have any bullet, he dies.
- If “The Brave” has bullets, it depends on the facing direction of “The Brave” and the zombie. If “The Brave” is facing the side or back of the monster, “The Brave” kills the monster. In case of both facing each other, “The Brave” wins. It is not possible that both of them are back to each other. If “The Brave” wins, he will have one less bullets (minimum 0) and the simulation continues.

Note: This resolution rules might not be complete. Feel free to add you own in order to coherently solve the interaction between components. You will not be evaluated by this as soon as your rules are precise and coherent.

1.4. AI and control

As you can see, the zombies are pretty dumb, but they will not know how to move or destroy objects unless you tell them. You need to write simple AI to control them. Remember, there is no need to make it complex, simple is the best!

For “The Brave”, we should be able to control manually it either with WASD keys or arrow keys.

2. Project description

Your project consists of modeling and implementing a complete simulation for the description above. For this, you have to use the abstractions introduced during the course until week 10, and to provide a graphical user interface that allows to appreciate the interaction between the different components. You can only use the declarative model abstractions extended with ports. It is not allowed to use things like NewCell, IsDet, IsFree, *classes*, or whatever language abstraction out of the declarative model and ports.

2.1. Deliverables

The project has to be done in groups of two. We strongly discourage you to work alone because of the load of the project, and also because working with somebody else will help you to see things from a different point of view, and, it will develop your social skills too.

The project must be submitted on iCampus. The submission must contain as attachement an archive file in zip or tgz format whose name is lastname1-lastname2.(zip|tgz). Of course, replace lastname with your last name. This archive must contain the following items:

• Full source code of your program. You have to submit a functor (or a set of functors, but there should be a main one that internally loads all other functors) containing the program. The program receives certain parameters, including the map of the room, the number of zombies, the percentage of objects that “The Brave” has to collect before leaving the room and the number of bullets “the Brave” initially has.

• A README file tells some important and concise information, for example which is the main functor to execute.

• A small report of about 2 pages, explaining the architecture and design of your project, along with the concurrency issues you faced and how you solved them. The full names of the authors must appear on the first page of the report. The report must be in .pdf format. We will not accept reports in .doc or .odt formats.

Submission deadline: Friday, May 9th at 23:58

2.3. Resources

Remember that you have several resources available to get some advice during the time for the project. You have the forum on iCampus (http://www.icampus.ucl.ac.be/), ask by mail to the assistants of the course and during the lab sessions. We will also give a small sample code for QTk, and a functor template to get the arguments of the program. We will also provide some files with maps for you to test. QTk will be used for the graphical user interface (GUI). This interface is quite important because it allows us to judge and grade your project. For more information about QTK, you can refer to http://mozart.github.io/mozart-v1/doc-1.4.0/mozart-stdlib/wp/qtk/html/index.html.
We will give more documentation on QTk and a sample code. For information about functors, you can refer to here:
http://www.ps.uni-saarland.de/~niehren/index.html/vorlesung/node106.html

We encourage you to develop the project by yourselves. Do not copy code from other students. Talk to other people and discuss ideas, but do not copy the code. If you do, you won’t learn that much, and you won’t deserve your points.