Escape from Zombies

Matteo Bezzi

Alma Mater Studiorum – University of Bologna
via Venezia 52, 47023 Cesena, Italy
matteo.bezzi7@studio.unibo.it

1 Introduction

This report describes the developing process of a video game in the context of the Autonomous Systems course. At the end of the lessons students were asked to develop a project taking advantage of the new notions learnt during the course and facing the related applicative problems.
The developed game has been named Escape from Zombiesand involves the player in saving his teammates and his own life avoiding zombies and finding enough food to survive.

2 Vision

At the end of this project we expect not only to have a working video game but, more important, to have acquired a basic know-how to work with autonomous entities and facing new problems with different perspectives.

3 Goals

The main goal of this report is to develop a video game employing concepts learnt during the course. In order to do that, the first thing we have to do is defining the basic elements and rules of the game we want to create, to produce the first set of requirements. After having produced a working prototype, we may be able to extend our project to add more elements, more rules and more complex behaviors to our agents.

4 Game Description

We have chosen to develop a game about zombies and humans as we wanted to two categories of agent with similar characteristics but completely different behavior. The game is played in a two dimensional field where zombies, humans, passive items and obstacles are placed.There are obviously two teams: The ZombieTeam which members are zombies and the HumanTeam which members are the survivors and the user. All the members of both teams can perceive a limited area around them. Humans can also communicate with each other. All the

players have an initial amount of life that gradually decreases. If the life goes down to 0 the player is dead. Humans can restore their life eating food they pick from the field. They are faster than zombies but they can't attack them. Zombies can restore their life attacking (biting) humans. They are slower than humans but they can attack opponents and they are supposed to outnumber the humans. The user plays in the human team. As other humans needs to take food, is faster than zombies and can't attack them but he will have special abilities and the possibility of giving orders to the survivors. The game can work even without a player. Zombies will look for humans, follow them and try to beat them until they are dead. Survivors will look for food and will try to avoid zombies. If given they will also execute orders by the player. The player is supposed to help survivors by guiding them or holding zombies. The winner team is the one whose members survive longer.

5 Requirements

We are now formalizing a set of requirements that our project will have to fulfill. Most of the them can be deduced from the game description of the previous paragraph.

- The game is played in a two dimensional field where Zombies, Humans, passive items and obstacles are situated.
- The field can spawn food. Food can be picked, eaten or dropped by players.
- Two teams: Zombie team vs Human team. Different number of players recommended.
- The user must be able to interact with the game.
- The game must work even without the user.
- All the players have values or states they can't directly control (eg. life, dead/alive).
- Players are able to interact and communicate.
- All players can perceive a limited area around them.
- All players can move in the field but can't pass obstacles.
- Zombies and humans have different movement speed.
- Humans can pick and eat or drop food. Zombies can attack humans.
- The user may be just controlling a human player or a special player with some particular ability.
- The game must take note of score and number of alive player to notify the winner.

Non functional requirements

- Players behavior must show to the observer an autonomous behavior.
- Interaction with human player has to be effective.

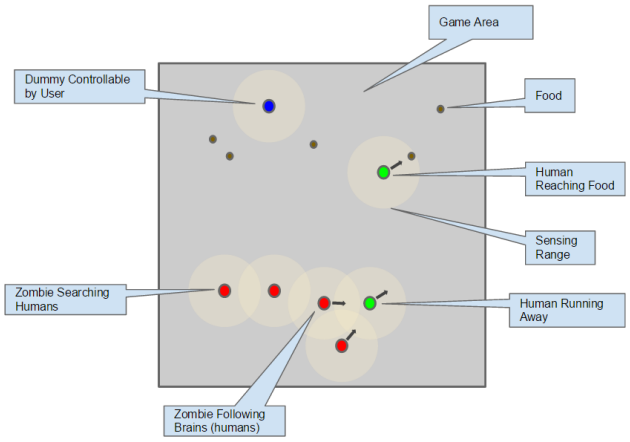


Fig. 1. Concept of the game

6 Analysis

6.1 Glossary

Term	**Description**
PlayingField	A 2D space where Players and passive entities are situated
Player	An autonomous entity that takes part to the game
Team	A set of players
Survivor	A player, member of human team
Zombie	A player, member of zombie team
Obstacle	a passive entity situated in PlayingField that prevents movement or access to players
Food	a passive entity situated in PlayingField that can be spawned, picked, dropped or eaten
Action	what a player can do to change something in the Playing-Field eventually effecting himself or other players.
Move	an action that a player can perform to change his position on the field
Pick	an action that removes the food from a target area and loads it on the player
Drop	an action that spawn on a target area a food unit removing it from player
Eat	an action that uses the food to restore life of the player
Attack	an action that a player can execute to hit an other player

6.2 Domain model

From the requirements we can pinpoint two entities that appears to mainly represent our system: player and playing field.

Player
A player has to be autonomous and situated. Players are never still unless they are blocked or dead. Depending on the team player may have different goals and behaviors. If this game were a table game, a player would be one of the active participants.

Playing field
The Playing Field is a passive item that include all the non-autonomous entities of the game as food. If this game were a table game, PlayingField would be the game itself with board, tokens and other stuff.

6.3 Logic architecture

In the previous two entities we have seen what in a table game would be players, board, token and other stuff. From a deeper analysis we can notice that there is still missing something to interface those two entities, describing not players nor other elements of the game, but the game it self: in other words, the rules of the game. Those rules may be embedded in players and in the PlayingField but, in order to keep the system more manageable and structured the project, it appears to be more strategic to add an other entity between those two: the game arbitrator.

Game arbitrator
The game arbitrator has to manage interactions between players and with the other elements of the game. It's the interface of players with the PlayingField and tells them only what they are supposed to know allowing them to do only what they are allowed to do. It could be both passive or active in the sense that it could also be calling players to play instead of only waiting for them. If this game were a table game, the game arbitrator would be someone external that knows rules and manages other participants requests.

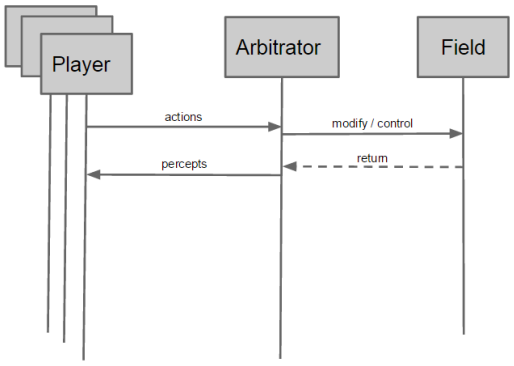


Fig. 2. Interaction of the tree entities identified during the analysis

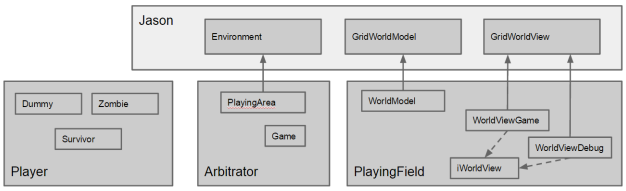
6.4 Abstraction gap

From the analysis we have seen that players will have to be autonomous entities, we will need an environment where they will move and interact and we will need a graphical interface for interaction with the user. If possible we would use some platform to rise our technological hypothesis and reduce the abstraction gap. During the lab hours of the Autonomous Systems course we could get in touch with different tools such as Jason, NetLogo. In order to develop the may prefer to use Jason as it provided an extendible platform developed in Javaand so easily integrable.

7 Project

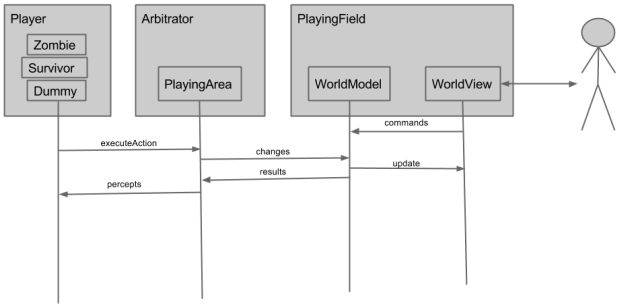
Since we have never developed a system on Jason we use want to learn by this example some common strategy to better develop our project. Looking at the examples provided with Jason we can find in the case study "Gold Miners" more than one similarity with our problem. Our players will have to collect food instead of gold, food will be eaten instead of being taken to the depot and we will need different sized and structured maps. On the other hand we won't use a leader to coordinate other agents, we won't use external algorithms to choose a path and, most of all, we will have opponents to be hit or evaded.

7.1 Structure



The project of this system is guided by the architecture of Jason. The entities we got from the analysis are now mapped into Jason classes.
All the players are obviously agents. We have tree different kind of agents: Zombie, Survivor and Dummy. Dummy agent is supposed to be -in some way- controlled by the user.

7.2 Interaction

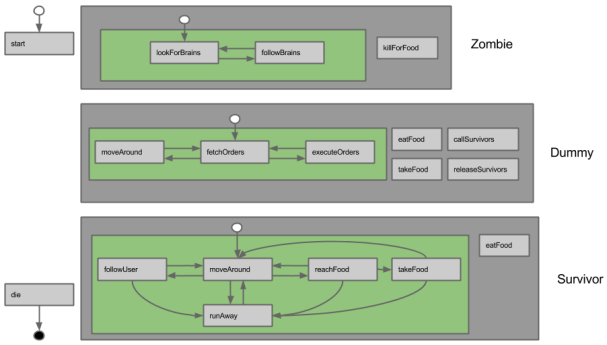


User interaction
The interaction of the user may happen in many different ways. As we have chosen to use Jason, the I/O interface for the user will be the WorldView. Assumed that, user interaction with the system could be implemented in several ways. In order to keep agents as autonomous as possible user will only leave

his commands in the model as writing in a black board and the agent will perceive those commands by figuratively reading them in the border of the arna. An other possible interaction was sending them as a message but the previous method seemed to be more realistic and effective.

Players interaction
The graphical description of interaction we have given before is missing the actual interactions between agents as we are only describing the system in which agents are moving. We could enter more in deep in players interaction but the fact is that, at least for now, the only direct interaction between agents is the communication between Dummy and Survivors.

7.3 Behavior



The most relevant part of behavior description in this system is obviously the one concerning agents.
As we are using Jason, our agents will be described using the BDI model. In particular we are now pointing out a high level view of the goals. We have a Start and a Die goal common to all the agents. The other goals are specific for each agent and can be added by event triggered plans or renewed by a previous goal.

8 Implementation

The full source code is provided with this report. Here we are discussing some technical solution we used to better fit the requirements or to improve the project.

Sensing Area
Each agent can perceive the area around him. Except for the 8 cells adjacent the agent that are obviously the range 1 set, we have defined the range 2 and range 3 as sketched in figure 4.

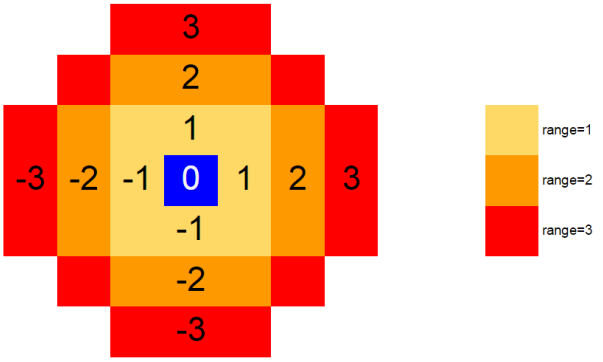


Fig. 3. Perceiving range of a player

Goal renewing

In our system, percepts are updated at the end of the execution of an action. If an agent is not doing anything he won't perceive anything. If an agent is slow in doing something, he will take more time before perceiving the area. This phenomena finds an explanation in the game mechanic: as zombies and humans have both different reflexes and different speed of movement we are decoupling the speed of each agent from environment and other agents. Humans will be faster and more reactive. In order to keep agent always "alive", we created a set of recursive goals that produces a basic behavior for each agent. Even without any interaction with other players, each agent will keep renewing or updating goals.

Internal Actions

Runaway and Follow are the internal action we have designed to let agents choose the next direction when there's a target to be reached or an enemy to be evaded. Direction can have 5 different values: up, down, left, right, skip. Agents can't move diagonally but only long the two axis. If they "decide" not to move they just use a skip move. In order to avoid (or at lest limit) situation in which the agent is stuck, if the agent finds the first chosen cell occupied, then we will try to move in an other admissible one. For example if an agent is moving down and in the next cell below there's an other agent, left or right cell will be then chosen on a random base. Up won't be chosen because is in the opposite direction of movement.

Gui

In order to make easier the debug and the tuning of parameter of the game, we have created two different GUI. The main is WorldViewGame which is supposed to be used by the user who wants to play with the standard rules. The second

interface is WorldViewDebug which allows user to select the scenario, spawn food in the area and change game speed.

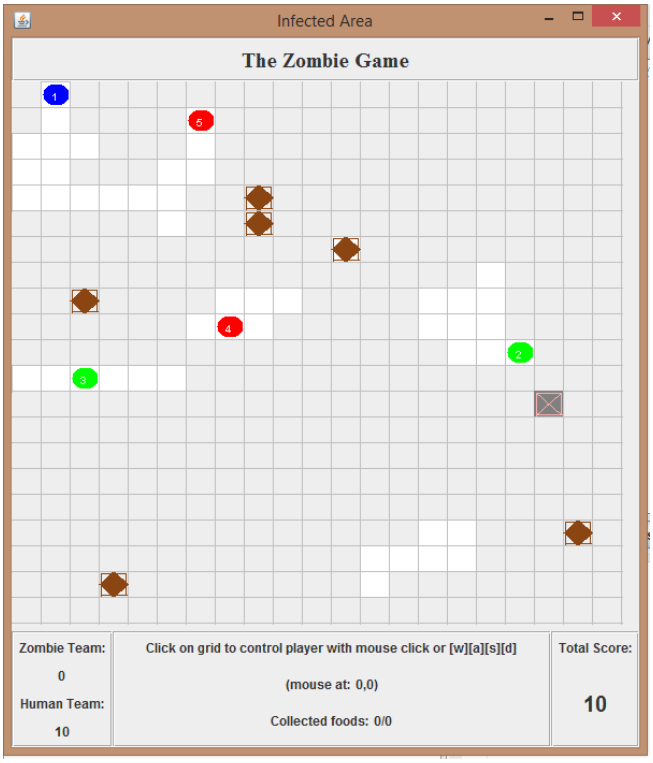


Fig. 4. Resulting graphic of the game