Universite Catholique de Louvain

Projet : Programming language concepts

Pokemoz

Auteurs :
Charles Jacquet (27811200)
Jérôme Lemaire (69601000)

Cours :
LINGI1131

Titulaire :
Peter Van Roy

Assistant :
Jérémie Melchior

FSA13BA
3 mai 2015

Introduction

The game Pokemoz was designed for the class LINGI1131. We have followed the project instructions but also added a little personal touch. In order to be closer to the reality, the wild Pokemoz evolve through the game and can acquire experience during combats but also over time. After a combat, if his Pokemoz is wounded or dead, the head trainer has to go back to his starting point, his house, to cure it or bring it back to life.

The artificial intelligence provided to our head trainer consists of making his Pokemoz evolve to the tenth level through various fights, while curing it when it's been hurt or lost a fight. Once the tenth level completed, the trainer takes the shortest way to reach the exit.

Structure of the code and implementation choices

First, the structure of the three main elements will be presented : the Pokemoz, the trainers and the maps.

We have represented the Pokemoz with a record that has the following structure
Pokemoz = p(type : name : hp : lx : xp :)
Where

- type : atom that can take 3 different values : grass/fire/water
- name : string representative of the name of the pokemoz
- hp : int representative the health status of the pokemoz
- lx : int representative the level of the pokemoz
- xp : int representative the experience of the pokemoz

We have represented the trainers with a record that has the following structure
Trainer = t(p : x :X y :Y handle : type :wild/persoPrincipal name :Name)
Where

- p : PortObject of the trainer's pokemoz
- x : int representative of the abscisse of the position of the trainer on the map
- y : int representative of the ordinate of the position of the trainer on the map
- handle : graphics components of the trainer
- type : atom that can take 2 different values : wild or persoPrincipal
- name : String representative of the name of the trainer

Maps are represented as matrices built with a tuple of tuples. The road map's matrices are filled with zeroes or ones. A value of 1 represents the fact that there is tall grass at that position while a value of 0 means that a road passes through that position.

The trainers map contains values of either N or 1000. A value of N nonzero represents that a wild trainer N stands on the position, N is the 'atom' which represent the wild trainer in the record used to stock them. A value of 1000 means that the head trainer is there and finally a value of 0 represents the fact that the position is free of trainer.

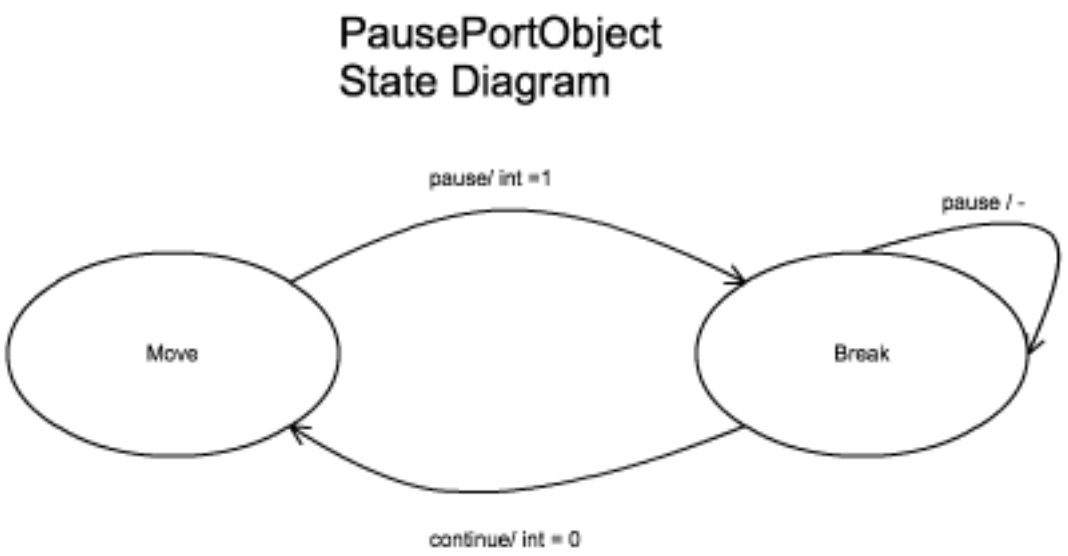
In order to design the game, 5 types of PortObject were used. We have included the Pokemoz and the trainers in PortObjects to be able to update them and accede to their information during the events (fights, displacements, treatments ...) that they live throughout the game.

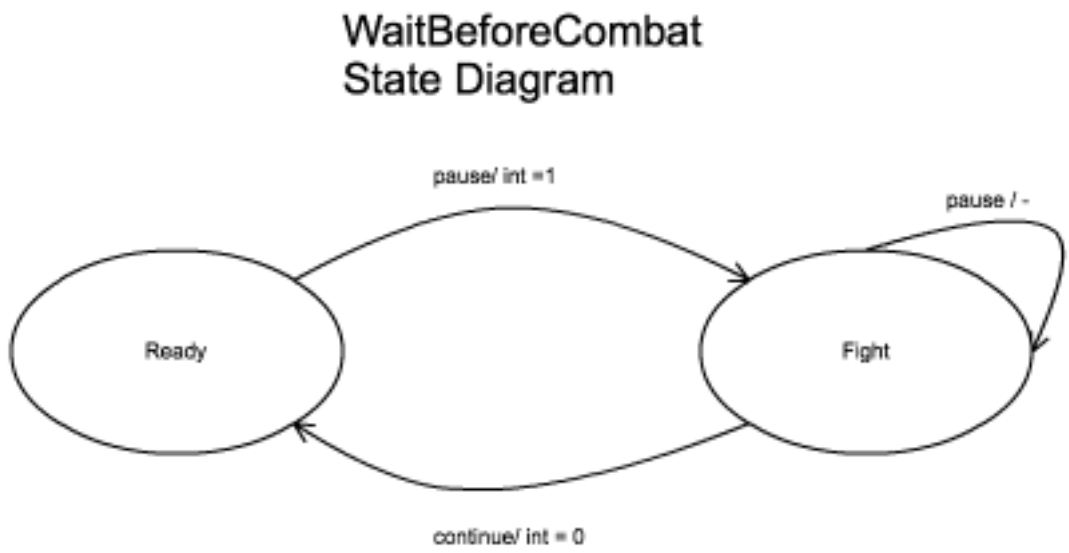
We have decided to use three maps. The first one is the visual representation of the trainers and their situation : it's with this map that the player interacts. The two others maps are not see by the player, they are used as support for the visual representation :

- The first map is used to represent the grass areas on the roads.
- The second one virtually represents the position and displacements of the trainers and is included in a PortObject because it is frequently updated and questioned about its state.

Finally, the two lasts PortObjects enabling us to stop the displacements of the trainers while a combat is running and to start one combat at a time. But unfortunately it doesn't work as well as we'd imagined it. Actually at that time we didn't know PortObjects with states and we tried to create that with these two PortObjects. But some problems still encounter due to the fact that unlike state, our functions aren't atomic ones. That's (with all what we added to try to counter that), the only reason why some times, more than one combat can happen. Or also those trainers can move a little in some case while a combat is already underway (If the movement had start before the 'pause' event or also if more than one combat are started and one stops). When we notice that, we've seen that our architecture doesn't allow use to add state and we had not enough time left to change all our code.

We invite you to take a look at the component and state diagrams to understand the interactions between the PortObjects.





Component Diagram