Geometry Friends Competition (CIG 2015)
OPU-SCOM

Benoit Vallade, Alexandre David, Tomoharu Nakashima
Department of Computer Science and Intelligent Systems
Osaka Prefecture University
Osaka Japan
valladeben@cs.osakafu-u.ac.jp

Benoit vallade[illegible]

I. INTRODUCTION

This document describes the particularity of our team's (OPU-SCOM) submission to the geometry friends competition held during the CIG conference. Our submission is targeting the "Rectangle Track" of the competition. Its purpose is to control the behavior of a rectangular shaped agent within a 2 dimensional game. The agent has to collect the objective points spread on the map. To reach its objective it will be able to move on the left and the right as well as modify its height and width (while its area remains the same). Our solution submits an artificial intelligence (AI) composed of two layers. The first layer's purpose is to determine the AI's global strategy. A strategy is sequence of sub-objectives. A sub-objective is not an elementary action. It is more of a step in the progression to the victory. The second layer's objective is to find the best sequence of elementary actions in order to accomplish the first sub-objective of the strategy computed by the first layer. The first action of this sequence is sent to the system which will update the game's state. *During the development of this AI we tried to create a system able to be adaptable to any kind of game. Even if in the case of the geometry friends competition some processing seems unadapted or simple enough to be done manually we tried to prove the validity of a system still in development on this single game.*

II. OPU-SCOM'S ARTIFICIAL INTELLIGENCE GLOBAL CONCEPT

A. *Strategy, sub-objectives and elementary actions*

As outlined above, a sub-objective is not an elementary action but more of a step in the progression to the victory.

For example, in the context of a Role Playing Game (RPG), the agent wins the game by defeating the final boss. In order to kill this enemy, the agent will have to go through a sequence of tasks. First, to become stronger, the agent needs to gain some experience by fighting weaker enemies. Then he has to acquire powerful weapons and items by exploring dungeons or by crafting them. These tasks are sub-objectives and their accomplishment requires a large amount of elementary actions.

Using the same example, the first layer could generate a strategy whose first sub-objective is to gain experience by fighting against a particular enemy. Then in order to accomplish this sub-objective, the second layer will have to determine the best sequence of elementary actions. These actions could be "use punch A", "use punch B", "move backward", "jump" or again "drink life potion" and are directly computable by the system .

B. *Using this concept within the geometry friends*

In the case of the geometry friends competition, the sub-objectives are very basic. Increase or decrease X, increase or decrease Y, increase or decrease the velocity of X, increase or decrease the velocity of Y, increase or decrease the height of the square, and go to a specific point (X and Y). As we are still developing our AI we decided to don't include the last one. The elementary actions available in this game are go right, go left, morph up and morph down.

III. FIRST LAYER

As explained above, the first layer's purpose is to determine the best sequence of sub-objectives in order to win the game. Those sub-objectives are chosen within a defined list. Therefore in order to play to the game the AI needs to determine the specific set of sub-objectives for the game. We developed a procedure able to identify these sub-objectives by analyzing the game's state. The first part of this section describes this identification procedure. Then the second part details the module in charge of generating the strategy.

A. *Sub-objectives identification*

In each iteration of the game, the system sends a copy of the game 'state to the AI. The AI uses these data in order to compute the agent's next action. To be able to use our AI on various games, these data need to be formatted in an identical way.

In order to do so we divided the game's state data in a list of block. Each block represents a separate object of the game; for example, the square, an obstacle or a collectible. Each block is identified by its own identifier (id). This id is composed of the object's type and its number. The number allows us to uniquely identify the objects whereas the type allows us to pool them by family. In addition of its id, each object is linked to a variety of different data representing its characteristics. For example, a collectible would be linked to its position (X and Y).

Nevertheless, some information can't be linked to any object of the game. For example, the time, the game's winner and so forth do not define any object in particular but the game itself. In order to fit these information in the same way as the others, we create an artificial object called "game statistic". This object contains all the neglected data.

Before to be sent to the AI, each object, including the game statistic, is formatted. The format used by our AI is a list of all the data linked to the object preceded by the data related to its id. The Fig 2 represents this format.

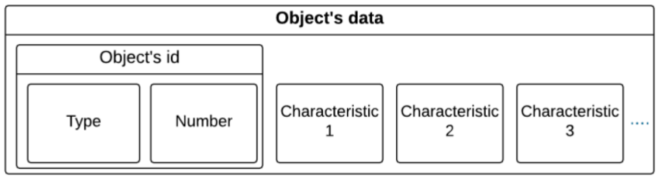

Fig. 1. Representation of the object's data formatting

Our concept of sub-objective is that a particular sub-objective consists of a particular data and an order. Firstly, the data is represented by the id of the object containing the data and by its number within the object's data (numbers 0 and 1 are attributed to the object's id). Then the order can come from two categories, the directions and the targets.

A direction is an order contained in the following list:

- Equal (keeps the value / all data types)
- Different (changes the value / all data types)
- Increase (increases the value / only for numbers)
- Decrease (decreases the value / only for numbers)

The target can take a particular value. The type of the target value depends on the data's type. Fig 3 is a logical representation of a sub-objective as explained above.

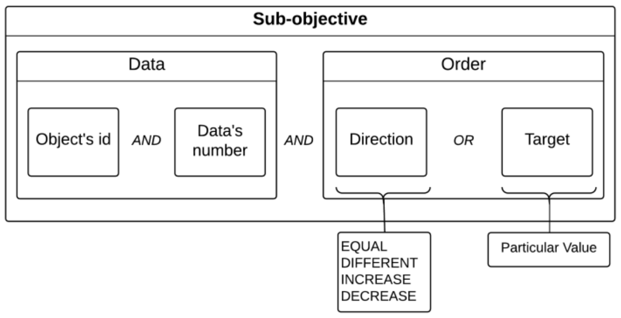

Fig. 2. Sub-objective's format

In order to identify the various sub-objectives available in the geometry friends competition we first needed to collect a lots of game's state data representing various possible evolution of the game. We accomplished this step by using the AI we developed for the competition last year. After collecting these data we processed them in order to evaluate the sub-objectives which would be usable by our new AI. We also purposely focused only on the data relative to the square agent.

B. *Strategy generation*

The second module of the first layer is used to generate the strategy of the AI. While the identification process explained above is done offline. The strategies are generated online depending on the previously identified sub-objectives and the evolution within the game.

The strategy generator is based on a neural network. The neural network take the game state as input and give a single sub-objective as output. This sub-objective is then sent to the second layer.

The reason why we use a neural network is to be adaptable to any kind of game. By using a NEAT based training we created a neural network able to understand the geometry friends game-play and to determine what strategy is good in what condition.

IV. SECOND LAYER

Because of the simplicity of the geometry friends competition's sub-objectives it is very simple to link a sub-objective to a single elementary action (in more complex game a sub-objective may require more than one action). Due to this particularity and because our system is still in development we decided to use a hard coded second layer which translate each sub-objective to its corresponding elementary action. For example the sub-objective increasing X would correspond to the elementary action Go right .

V. CONCLUSION

To conclude, our solution proposes an agent implementation for the "Rectangle Track" of the Geometry Friends' competition which is composed of two layers, the first one is used to compute the global strategy by first identifying (offline) the various sub-objective of the game and then generating the best strategy (sequence of sub-objectives) using a composition of genetic algorithm and neural network. The second layer then generates orders to perform the computed strategy.