Statecharts Modelling of Tank Wars

Silvia Mur Blanch

Modelling, Simulation and Design Lab
School of Computer Science
McGill University, Montreal, QC, Canada
silvia.murblanch@gmail.com

Abstract. As the video game industry profits keep increasing faster year by year, there's a growing demand for more realistic games, this leading to a necessity for better AI specification and fast game development which, consequently, calls for easily reusable AIs. By using modelling languages, AIs can be specified at a higher level of abstraction, allowing the modellers to focus on the logic of the model they are building, instead of having to worry about the implementation issues. The present project sets out from this proposal, expounded in the paper *Model-based Design of Computer-Controlled Game Character Behavior*, in which their authors used the Statecharts formalism to demonstrate their approach by modelling a video game character's AI.

1 Introduction

Computer games sales have experienced a huge growth in the past few years. The industry has widely expanded its market targets, nowadays offering variety and quantity enough to fit every taste and age range. At the same time, the development of more powerful computers and consoles has led to an increasing demand for more realistic computer games, which doesn't exclusively mean developing better physics simulations for water or car crashes. In a first person shooter or war simulation game, for example, it seems obvious that the enemies should shelter behind walls or other elements of the environment, whenever bullets are raining down on them. All in all, this means writing better, more complex AIs, which is becoming harder every time.

The authors of the paper *Model-based Design of Computer-Controlled Game Character Behavior*, on which this project is entirely based, state that AI specification should be done at a higher level of abstraction, that AIs should be modeled instead of coded, so that it could be done by people who don't necessarily know any programming languages. And, at the same time, to satisfy a necessity for reusability, AI specification should be made as modular as possible. Which would then be the appropriate formalism to use? Given these requirements, and the fact that these models will define reactions to game events, it seems obvious that we'll be looking for a formalism that's state/event based and that provides autonomous/reactive behaviour. The formalism that we've chosen is Statecharts, since it's well known by us and we have our very own Statecharts

modelling tool, AToM^3, developed at the Modelling, Simulation and Design Lab (MSDL) in the School of Computer Science of McGill University.

2 Modelling an AI

To demonstrate the approach proposed in the previously mentioned paper, the authors used the Tank Wars game by Electronic Arts (EA), thus modelling a tank's AI with the Statecharts formalism. First of all, the tank's architecture is abstracted in different layers, each of which defines a component or group of components of the tank. As well, each layer represents a different level of abstraction (see Fig.1).

The tank's AI will make decisions based upon the information it gets from the environment; this data is perceived by the tank through its sensors, at the lowest level of abstraction, then refined and passed on to the other components of the tank. The tank will react to the input it gets by executing actions; for example, whenever an obstacle is detected in the tank's path, the tank should turn left or right to dodge it. The event flow in Figure 1 describes the order of the transformation process of the input received through the sensors into the output given through the actuators.

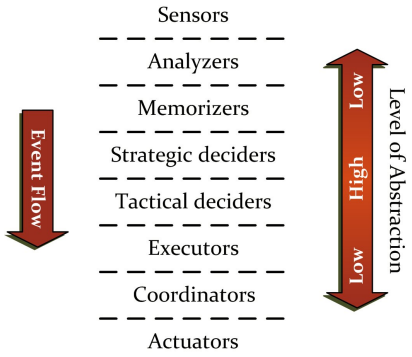

Fig. 1. Tank's AI architecture

Each components structure is modeled by a class, and its behavior by a statechart. Though modelled separately, the components of the tank are orthogonal, which means that the statecharts will run concurrently when the execution starts, and that the events generated in one statechart can (and definitely will) trigger transitions in some of the other components. For example, when the *InRangeDetector* component of the tank detects that the enemy is in the range of the turret, it will generate the event *readyToShoot*, which shall trigger a transition in the *AttackStrategy* component and make it change its state to *Shooting*.

Figure 2 depicts all the tank's components put together, and the dependency between some of them. When a component's functionality depends on data from another component, synchronous communication must occur.

3 Time-slicing to Event-driven

In the EA Tank Wars game, information from the environment is provided to the tank every 50ms. This information includes current tank's fuel level, cur-

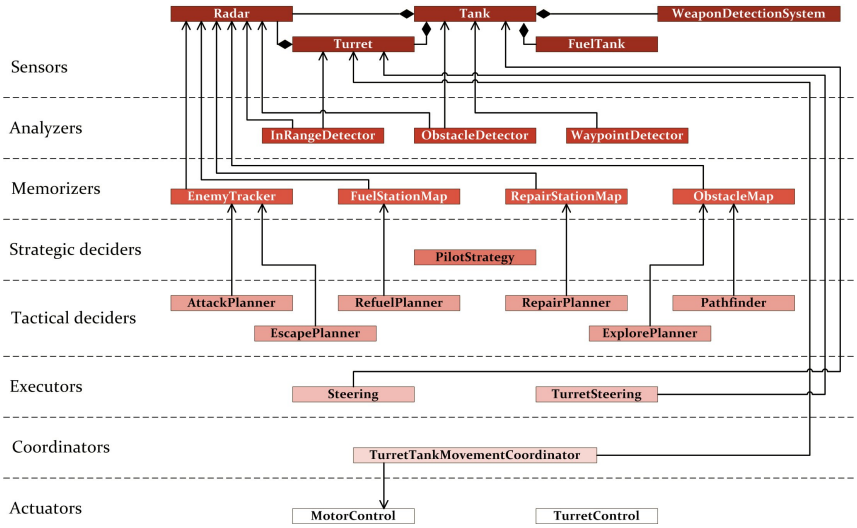

Fig. 2. Tank's AI components

rent tank's health level, etc. Thus, the main difficult of modelling the tank's AI using a state/event formalism resided in bridging the gap between the time-sliced world and the event-based world. So, after compiling the classes and statecharts into C++, some intermediate code writing was still needed.

Everytime the tank receives input from the environment, the function *ai()* is called and, consequently, this is the starting point for the EA Tank Wars competition contestant to write his or her own AI code. In the statechart modelling solution, the input data is processed, causing events to be generated and sent to tank's components (statecharts), in order to trigger the appropriate trasitions, thus obtaining the corresponding reactions (which at the same time may generate more events that will trigger more transitions) and, finally, translating these into the AI's output that will be sent to the game environment in response to the previous input.

4 Conclusion

The approach proposed in [3] was developed and tested by its authors, being the tank's components modelled using the Class Diagrams formalism, where each component's structure is described by a class, and its behaviour by a statechart. The classes and statecharts where implemented using AToM^3, then compiled into C++ code, and finally inserted into the EA Tank Wars main game loop and successfully run.

However, the statecharts implemented weren't as detailed as the ones originally described and, though the authors managed to bridge the event-based

world with the time-sliced world, the whole solution wasn't as event-based as it had been initially planned.

In the practical part of this project I will try to modelate the tank's AI components from scratch, using the more elaborate statechart designs depicted in [3] and a more simple, self-made simulation environment, instead of the EA Tank Wars game environment.

References

1. Electronic Arts. EA Tank Wars. `http://info.ea.com/company/company_tw.php`, 2005.
2. David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.
3. Jörg Kienzle, Alexandre Denault, and Hans Vangheluwe. Model-based design of computer-controlled game character behavior. In *ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems*, pages 650–665. Springer Berlin / Heidelberg, 2007.
4. Hans Vangheluwe and Juan de Lara. AToM^3, A Tool for Multi-formalism and Meta-Modelling. `http://atom3.cs.mcgill.ca`, 2002.