Paul Holmes and Mitchell Aharon

Project Team #22

Programming Languages and Translators

Language White Paper

**The Geppetto Programming Language**

The ability to simulate intelligent behavior has been a dream of computer scientists ever since computers were invented, and the need for this capability is ever increasing. One need look no further than the game console in the living room to see that modeling intelligent behavior is an everyday need even in something as commonplace and trivial as video game software: the bad guys need to behave intelligently or the game becomes easy – and boring – very quickly.

Recently, applications like Apple’s Siri and IBM’s Watson have put intelligent devices at the forefront of public attention. Advances in speech recognition, natural language processing and data search and retrieval have combined to at least give the impression that in some cases, computers are beginning to approach human levels of intelligence. But the reality is that despite progress in a few narrow domains, we still have a very long way to go before true artificial intelligence is a reality. In the meantime, the best we can do is to simulate and model intelligent behavior.

However, the challenge of coding behavior that is credibly intelligent is a daunting one. Even experienced programmers[[1]](#footnote-1) find the problem difficult, and in any case not every developer can be an expert in AI. Indeed, the people who have the domain knowledge about how a system should behave aren’t necessarily even programmers at all.

Currently, there are few other programming languages that address these needs – at least not mature, widely used languages. Perhaps the most prominent example is generally not regarded as a programming language at all: the Unified Modeling Language (UML), which is widely used to create visual models for use in object-oriented software design. With the proper tools these models may be “compiled” into object hierarchies in target languages like Java. However, few people would attempt to write an application using UML.

A more apt example is the Reactive Model-based Programming Language (RMPL), a language designed at MIT to control the behavior of spacecraft and robotic explorers like the Mars Rover, which can’t rely on direct human guidance. This language is model-based, meaning that a program uses a model of its environment to deduce the environment’s state from observable factors, and determine the proper behavior to exhibit in response. Of course, RMPL is highly domain-specific, being written for embedded systems on spacecraft, so it is not suitable as a general-purpose programming language.

That’s where Geppetto comes in. Geppetto is a computer language designed to bridge the gap between the programmers who know how to write code, and the domain experts who understand how a system should behave. Its main design goal is to simplify the process of developing an intelligent system, both for the programmer and for the domain expert.

For example, let’s say that a psychology researcher wants to conduct an experiment in fear responses.[[2]](#footnote-2) A human test subject controls an avatar in a virtual environment – say, a 2-D maze – and the computer controls “predators” that wander the maze searching for the subject. When the subject encounters the predator, the researcher measures the subject’s fear response. This experiment would be far more compelling if the predator behaved in a realistic manner, rather than, say, simply making a beeline for the test subject immediately upon encountering her. The researcher knows how a real predator would behave, but isn’t an expert programmer; the programmer knows how to write code, but isn’t an expert in behavioral modeling or AI. What’s needed is a way for the researcher to functionally describe the desired behavior without having to learn how to program, and for the programmer to be able to write the code to implement the desired behaviors without having to learn the intricacies of AI programming.

Geppetto addresses both of these issues. It has a ***simple declarative notation*** for describing the properties of intelligent “agents” and their environment which can be used by domain experts who are not necessarily programmers. Furthermore, the language ***encapsulates the details*** of its underlying behavior modeling algorithms, so a programmer who uses it need not be familiar with the intricacies of AI programming. Thus the ***code is simpler and more concise*** than if the programmer had to start from scratch without using Geppetto. Geppetto is ***model-based***, in that the descriptions of the intelligent agents and their environment constitute models which may interact with each other in unforeseen ways to produce potentially novel behaviors. This allows systems written in Geppetto to operate in environments in which not all the details are known in advance. Finally, Geppetto is a ***general-purpose programming language***, so its use is not limited to specific embedded systems like space probes – it compiles into C, so any system that can use the C programming language can use Geppetto.

Perhaps a simple code example would be illustrative at this point (keeping in mind this is only pseudocode and is subject to change or omission in the final product). Let’s revisit the example of the psychology experiment above, and that we want to model an encounter between two intelligent agents: a predator and its prey. We first describe the agents and their initial conditions, and then we wind up the simulation and let it go:

predator.setTrait(AGGRESSIVE);

predator.setLocation(x,y);

predator.setBehavior(SEARCH);

prey.setTrait(CAUTIOUS);

prey.setLocation(m,n);

prey.setBehavior(MOVE\_RANDOMLY);

environment.setVisibility(CLEAR);

environment.addAgent(predator);

environment.addAgent(prey);

until (predator.getStatus() == CHASE)

interact();

predator.moveToward(PREY);

…

The idea is that the next states of the predator and prey are automatically determined by the system based on their given initial states until some stated condition is met, and then the next action is coded.

Using declarative statements, Geppetto allows domain experts such as psychologists and economists to create complex behavioral models without needing to learn the intricacies of AI programming. The language hides the low-level implementation of model construction from the programmer and executes simulations with its integrated engine. By abstracting away the details from the programmer, Geppetto helps to close the gap between domain experts who are interested in simulations and the technical experts who are able to code them.

1. Such as, for instance, the author! [↑](#footnote-ref-1)
2. This actually happened – it’s where the inspiration for this language came from in the first place. [↑](#footnote-ref-2)