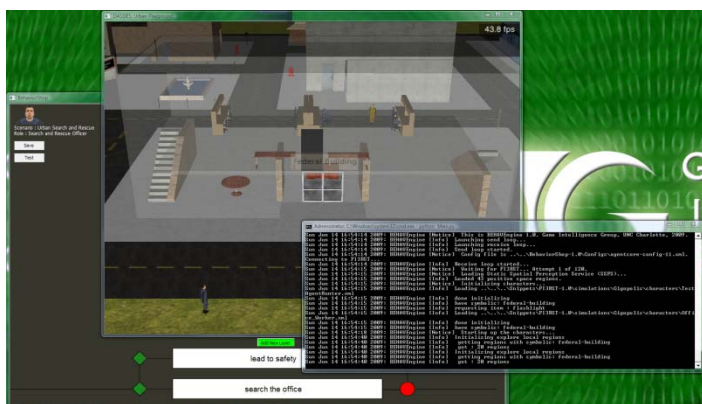


Rapid Development of Intelligent Agents in First/Third-person Training Simulations via Behavior-based Control

HOW BEHAVIOR-BASED CONTROL CAN FACILITATE BUILDING AGENTS WITHOUT PROGRAMMING



Executive Summary

Lately, first/third-person training simulations have played an increasingly important role in facilitating mission rehearsal, environment familiarization, and cultural awareness. The two most important parts of any first/third-person training simulation are the environment (which consists of the terrain, static objects such as buildings, and interactive objects such as vehicles) and the intelligent agents in that environment (for example, bystanders at a car crash or victims in a building on fire). Typically, developing any new scenario requires building a new environment and new intelligent agents. This process is almost always time critical, as the earlier the scenario is developed, the more it can contribute to training. Hence, it is important to develop intuitive tools which facilitate the rapid development of such scenarios. While there has been a lot of progress in developing tools to model environments, the tools to model intelligent agents lags far behind. Often the design of intelligent agents requires programming (scripting), which considerably slows down the process of building scenarios. This paper discusses and demonstrates how behavior-based control can facilitate the rapid development of intelligent agents without programming.

Behavior-based control has its roots in robotics and is an extension to the subsumption architecture (Brooks. 1986). At

an abstract level, agents created with behavior-based control consist of one or more prioritized layers of behaviors, where each layer maps a combination of percepts (what an agent sees) to a combination of actions (what an agent does). At any instant in the simulation, the agent receives one or more percepts, activating one or more behavioral layers, causing the action(s) associated with those layers to be carried out by the agent.

The Game Intelligence Group at the University of North Carolina at Charlotte, under the guidance of Prof. G. Michael Youngblood, has pioneered the use of behavior-based control in first/third-person training simulations. Their flagship research project DASSIES (Dynamic Adaptive Super-Scalable Intelligent Entities) incorporates tools to design agents via behavior-based control (BehaviorShop) and a behavior-based agent engine (BEHAVEngine) which operates on the agent definitions (produced using BehaviorShop) and implements their behavior in any standard first/third-person shooter game engine (simulation environment agnostic).

The Game Intelligence Group has performed extensive human trials to evaluate the effectiveness of behavior-based control. Participants of the trial (mostly people not from a computer science background) were provided a general text description of agent behavior in a scenario and were asked to design an agent using BehaviorShop. The study involved a total of thirteen different agent descriptions, divided into five scenarios, and over a hundred human participants. Results indicate that at least 80% of the participants were able to build agents with at least 80% behavioral accuracy (based on compatibility with text descriptions of each scenario).

These results strongly assert the potential of behavior-based control in designing agents for first/third-person training simulations. Behavior-based control and its implementation in BehaviorShop and the BEHAVEngine represent the state of the art in building agents for first/third-person training simulations, and their adoption will greatly enhance all first/third-person training simulations.

Designing a Scenario in a First/third-person Training Simulation

The key goal of designing a scenario in a first/third-person training simulation is to facilitate the training of operatives for a particular situation in a test bed that closely resembles the real world and allows them to develop expertise with respect to the particular situation (for example, training rescue workers to systematically search a building for victims of a fire). Training scenarios can be extremely diverse in nature, but at an abstract level consists of three key elements, namely, the environment, the intelligent agents, and the human agents. A training simulation in a sense emerges from the interaction of these three elements, and designing a particular scenario involves modeling the environment and building the intelligent agents to mimic a real world situation. This process is best described by considering a specific scenario. Consider our example of a scenario which focuses on training rescue workers to systematically search a building for victims of a fire. In this case, the terrain and the particular building (with static elements such as walls and interactive elements such as doors and elevators) form the environment. The victims of the fire in different parts of the building are the intelligent agents. Designing this scenario would thus involve modeling the environment and building the agents, after which this scenario would be ready to be used for training rescue workers given an appropriate interaction interface for those workers.

The current state of the art in building such scenarios includes a diverse array of tools for environment design. Intuitive 3D modeling tools, as well as existing libraries of static and interactive objects, can be leveraged to construct a realistic environment. An important fact to note about building environments is that this process is fundamentally a 3D modeling exercise requiring no programming expertise and, given intuitive tools, can be completed relatively easily. Furthermore, existing libraries of environments and objects can be easily leveraged. For example, objects such as vehicles need to be designed only once and can be reused for multiple scenarios.

When it comes to building intelligent agents, the situation is far more complicated. There are hardly any tools which match the intuitiveness of 3D modeling tools, and often, agents need to be built by programming or scripting. Achieving the most trivial behaviors takes a significant amount of time, and agent building and tuning remains the most time consuming step of scenario design. Furthermore, it is relatively hard to reuse intelligent agents. For example, in the rescue worker scenario described earlier, we should not use a single agent to model all victims in the building because human beings do not behave in an identical manner—current simulations however

often do include single dimensional characters that reduces scenario believability and training effectiveness. Typically, we would want a range of human-consistent behaviors randomly assigned to different agents in such a scenario. The important fact to note about intelligent agents in training simulations is that, in contrast to building the environment, this has fundamentally been a programming exercise requiring a certain amount of expertise in logic development.

Another important point to note is that, unlike environments, which can be designed by 3D modelers based on descriptions, building intelligent agents requires a subtle understanding of the scenario and needs to involve domain experts who often lack the programming skills to achieve this task. For instance, building intelligent agents mimicking bystanders in a foreign country would require a nuanced understanding of the culture, which is hard to describe, and should be built by a domain expert, while the buildings and vehicles can be easily described via standard technical specifications. The lack of intuitive tools for building intelligent agents often requires the domain expert to collaborate with a software developer, complicating and delaying the process of scenario development.

It is thus critical to develop a theoretically sound framework for building intelligent agents to serve as a foundation for designing intuitive tools to address the problem of creating interesting agents in first/third-person training simulations. While this overarching objective is clear, achieving it requires incorporating ideas from two seemingly diverse fields, artificial intelligence (AI) and human-computer interfaces (HCI). The field of AI, to a large extent, has focused on building intelligent agents achieving concrete goals in an optimal manner with little regard to the complexity of defining such agents. HCI, on the other hand, studies the design and implementation of intuitive interfaces that allow the human user to achieve the task at hand with relative ease. The problem at hand requires formulating a framework that balances what the agent can achieve against the complexity of the agent design interface. Behavior-based control achieves this balance, as discussed in the next section.

Behavior-based Control

Behavior-based control (Mataric 1992) is an extension to the subsumption architecture (introduced by Brooks in 1986) and has its roots in robotics. Intuitively, agents created with behavior-based control consist of one or more prioritized layers of behavior, where each layer maps a combination of percepts to a combination of actions. At any instant in the simulation, the agent receives one or more percepts, activating one or more layers and causing the action(s) associated with that/those layer(s) to be carried out by the agent. Behavior-based control is

inherently parallel in the sense that multiple percepts can be received at a single instant of time, which can lead to multiple actions also being performed at a single instant of time. There are two key aspects to behavior-based control, the first being the mapping of percepts to actions (or combinations of percepts to combinations of actions) represented using one or more behavior layers, and the second being the prioritization of the layers, which specifies which layers override other layers in the case where multiple percepts are received. We illustrate these key ideas in behavior-based control using a simple example.

Consider an intelligent agent which mimics a simple organism in an environment with the following two predefined percepts: a) perceive food and b) perceive predator. Furthermore, the agent has the following three predefined actions: a) explore new regions, b) consume food, and c) flee from predator. Given these basic percepts and actions, an agent design using behavior-based control is illustrated in Figure 1. The key points to note are as follows. First, note that each of the layers maps percepts to actions. For example, layer L2 maps the percept food onto the action eat. Furthermore, note that the layers are prioritized. Layer L1 is overridden by layer L2 which in turn is overridden by layer L3. Also note that layer L1 does not have a percept and corresponds to a default action which is performed when no percepts are received by the agent. This simple agent designed using behavior-based control has the following overall behavior. When there are no percepts available, the L1 layer is triggered, and the agent explores new regions. In the case where the agent perceives food, the L1 layer is overridden by the L2 layer, and the agent consumes the food. In the case where the agent perceives a predator, the L3 layer is triggered, all the layers below it (L1 and L2) are overridden, and the agent flees from the predator. Note that the prioritization of the layers is the most important part of the agent definition and that higher layers can selectively override lower layers (combinations of layers producing blended actions are permissible).

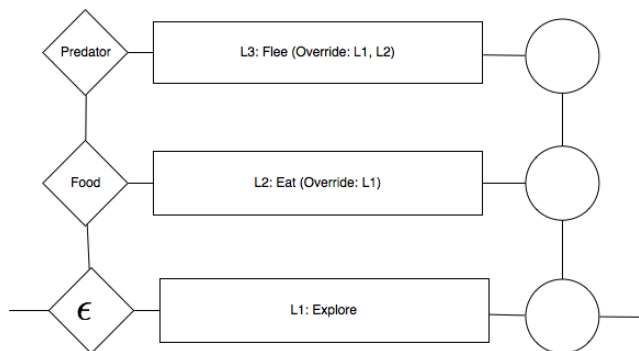


Figure 1. Behavior-based control for a toy agent

While the example discussed here is only for the purposes of illustration, it does demonstrate the key aspects of behavior based control, namely, the mapping of percepts to actions via layers, the prioritization of layers, and the ability to override or combine actions from multiple layers. Given any large set of percepts and actions, any reactive agent of arbitrary complexity can be constructed using behavior-based control. It is now essential to demonstrate how behavior-based control fits in with first/third-person training simulations from a systems perspective, which we discuss in the next section.

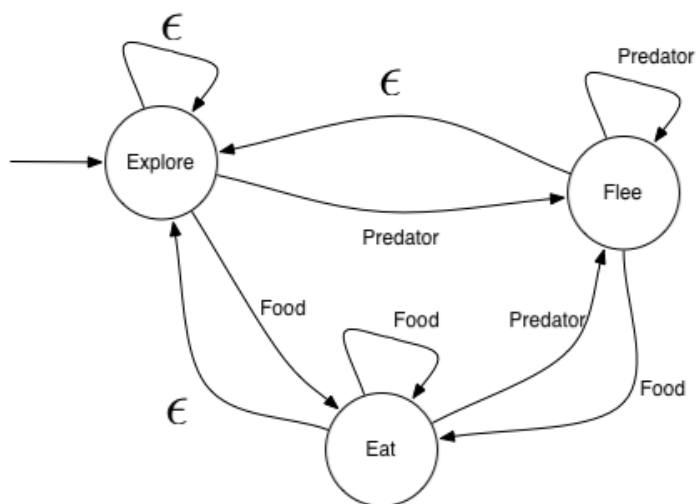


Figure 2. Equivalent Finite State Machine for the toy example

Behavior-based control has a number of advantages over the use of other architectures for game agents, such as finite state machines (FSMs), hierarchical finite state machines (HFSMs), and behavior trees. Behavior-based control is inherently parallel, as multiple active behaviors can be run at once. The representational complexity of a behavior-based control agent, which is an important consideration for the agent authoring process, is far lower than other architectures. As shown in Figure 2, the corresponding finite state machine for the previous example from Figure 1 requires many more transitions. If multiple behaviors are allowed to be active at once, the finite state machine becomes increasingly more complex, as each allowable combination of behaviors requires an additional state. HFSMs and behavior trees reduce this complexity over simple FSMs, but still require more complex models to represent the same agent. The reduced complexity of behavior-based control makes it simpler and faster to create intelligent agents.

Implementation of Behavior-based Control using BehaviorShop and BEHAVEngine

DASSIES (Dynamic Adaptive Super-Scalable Intelligent Entities) is the flagship research project of the Game Intelligence Group at the University of North Carolina at Charlotte, and it includes an industry strength implementation of behavior-based control targeting first/third-person training simulations. The architecture of DASSIES is illustrated in Figure 3.

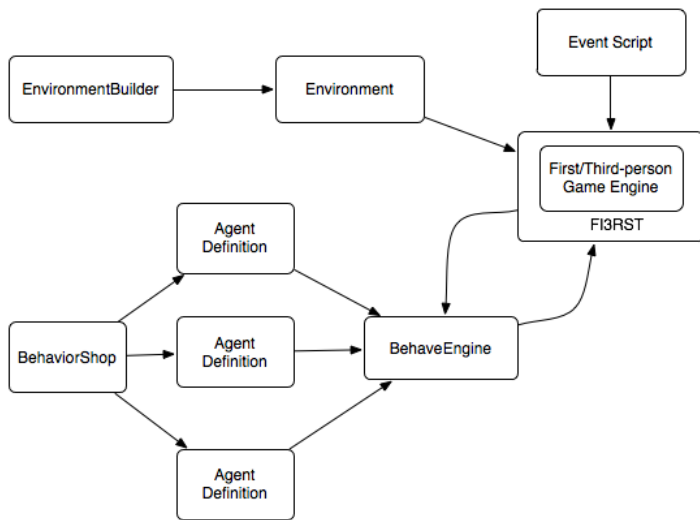


Figure 3. DASSIES System: Integration of Behavior-based Control in First/third-person Training Simulations

The key components implementing behavior-based control are BehaviorShop, which is a user-friendly tool to design intelligent agents; BEHAVEngine, which operates on one or more agent specifications scalable to a large number of agents; and any standard game engine to produce a specific simulation. An additional support component is F3RST (First and 3rd-person Realtime Simulation Testbed), which is a wrapper around game/simulation engines to provide a standard interface for BEHAVEngine (Currently F3RST supports the Panda3d and Irrlicht game engines, but could be extended to support any game/simulation engine).

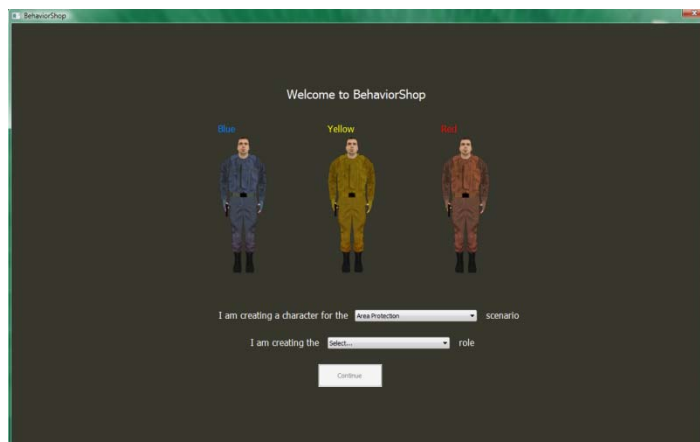


Figure 4. BehaviorShop Startup Screen



Figure 5. BehaviorShop: Defining the layers

BehaviorShop

BehaviorShop is the component that allows users to build intelligent agents using behavior-based control. It has an intuitive user interface based around using sentence-like constructions to define agents. Screenshots of the startup screen, the layers window (where the user defines the behavior layers), and the trigger-action editor are presented in Figures 4, 5, and 6 respectively.

The layers window, illustrated in Figure 4, can be used to add, delete, and move layers. The layer editing window depicted in Figure 6 can be used to select triggers and behaviors for the layers and define levels of priority. Often, actions performed by the agents require positional parameters. These can be specified by selecting locations on a preloaded map through the map window, illustrated in Figure 7 (these are generated in concert with the game/simulation engine). In general, BehaviorShop can employ unique dialog boxes for

soliciting information that does not conform to sentence descriptions such as the spatial information just illustrated.

At any point while designing the agent, the user can test and debug the agent by watching the simulation in the output window, depicted in Figure 8. BehaviorShop is a stand-alone application that integrates well to any networked or shared-memory game/simulation engine.

The trigger conditions (percepts), targetable objects (objects in the game/simulation), and actions presented for user configuration are dynamically specified to BehaviorShop at runtime through coordination with BEHAVEngine and F13RST. This reflects what is possible to perceive and act with and upon in the target game/simulation.

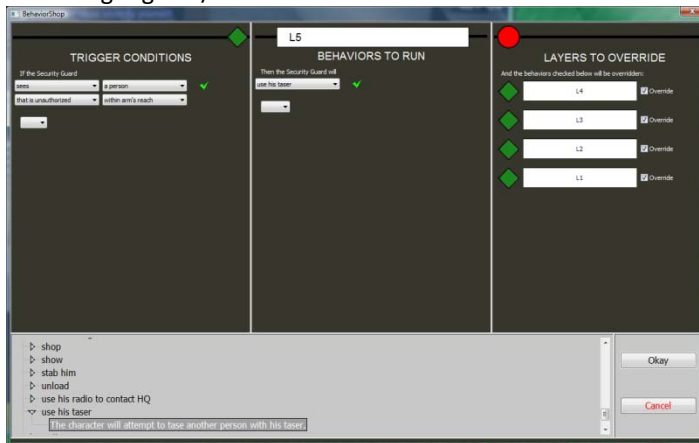


Figure 6. BehaviorShop: Specifying triggers and actions

BEHAVEngine

Agents defined by BehaviorShop are executed by BEHAVEngine in conjunction with F13RST and the game/simulation engine. BEHAVEngine constantly receives percepts for the agents in the simulation, interprets the agent design, computes the appropriate actions based on the agent design, and passes the action messages on to F13RST. These action messages are interpreted by F13RST and appropriate action animations (for example walking, jumping, and shooting a target) are chosen from a library of basic actions and played in the game/simulation engine.

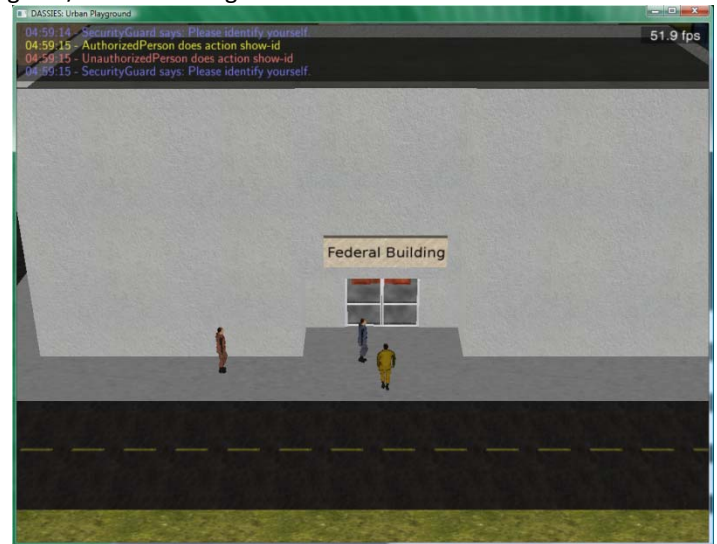


Figure 8. Testing and Debugging the Scenario



Figure 7. BehaviorShop: Selecting Locations in a Preloaded Environment

BEHAVEngine is a multi-threaded behavior-based control engine for game agents. In addition to the core intelligence architecture, it integrates navigation using navigation meshes and a modular perception and action system. Navigation meshes are decompositions of navigable space in the world formed into convex regions. These enable efficient path planning, information compartmentalization, and network partitioning as well as can be used for improved collision detection. The perception and action systems make it simple to adapt the engine to different game/simulation environments.

The engine also uses a novel reactive teaming approach to coordinate multiple agents in a team. This method allows agents designed as individuals to be easily adapted to team-based scenarios with minimal additional design complexity. The reactive teaming technique is also more appropriate for highly dynamic game/simulation environments in which traditional planning or auction-based coordination methods are infeasible due to the heavy computation required.

Human Trials with Behavior-based Control

The effectiveness of behavior-based control and its implementation in BehaviorShop and BEHAVEngine have been extensively validated using a large-scale human trial. Participants were asked to construct an agent for one of thirteen possible characters based on a scenario vignette. Characters ranged in complexity from a simple shopper in a market to a bomb squad technician requiring several complex behaviors. After watching a short instructional video, participants built an agent using BehaviorShop. A total of 102 participants submitted an agent they had constructed to be evaluated. These participants were drawn from a random sampling of individuals, the vast majority of whom had no previous experience creating agents. Agents were evaluated on a ten-point scale by a panel of experts based on how closely they adhered to baseline agents developed for each character. Any borderline agents were loaded into the simulation environment, and their performance was evaluated in the simulation. A score of eight or higher indicated that the agent could successfully perform the assigned task. Based on this scoring metric, 82 successful agents were created (scoring 8 or higher) for a success rate of 80.39%. Among the successful agents, the average score was 9.4, which indicates a high degree of convergence with one of the baseline agents for a given task.

Feedback from the participants about their experiences with BehaviorShop was recorded using five-point Likert scales,

where a rating of one indicates the user strongly disagreed with the statement and a rating of five indicates they strongly agreed. Participants were asked to rate the statement “Creating simulation characters is easy with the DASSIEs Creation Tool”; overall, the users averaged a 3.8, indicating they agreed with the statement and found agents easy to create. Additionally, participants were asked to rate the statement “I understood how to use the tools”; this statement averaged a 3.9 on our Likert scale, which indicates that most of the users did in fact understand BehaviorShop.

Conclusions

First/third-person simulations are an important part of modern training regimens for complex situations, facilitating mission rehearsal, environment familiarization, and cultural awareness. However, until now, creating complex intelligent agents for these simulations has required similarly complex authoring tools or computer programming knowledge. Behavior-based control is a new paradigm for modeling these agents in an intuitive manner, without sacrificing the expressive power of more cumbersome formalisms such as finite state machines. Employing BehaviorShop and BEHAVEngine to leverage the power of behavior-based control, users can easily create interesting intelligent agents with complex behaviors, overcoming a major hurdle in the development of first/third-person training simulations.

Contact Information

G. Michael Youngblood, Ph.D.
Director, Game Intelligence Group
Co-Director, Games + Learning Lab
Assistant Professor of Computer Science
College of Computing and Informatics
University of North Carolina at Charlotte
9201 University City Blvd
435A Woodward Hall
Charlotte, NC 28226-0001
youngbld@uncc.edu
M 704.737.6703
O 704.687.7989