A Demonstration of ScriptEase Interruptible and Resumable Behaviors for CRPGs

Maria Cutumisu, Duane Szafron, Jonathan Schaeffer, Kevin Waugh, Curtis Onuczko, Jeff Siegel, Allan Schumacher

Department of Computing Science, University of Alberta
Edmonton, Canada
{meric, duane, jonathan, waugh, onuczko, siegel, schumach}@cs.ualberta.ca

Abstract

Intelligent non-player characters that exhibit realistic ambient behaviors produce more captivating and immersive stories for the player. However, the creation of non-repetitive and entertaining behaviors is challenging, since it involves writing complex custom scripting code for thousands of characters in a common game adventure. This demonstration describes the generation of motivational behavior scripts using generative *behavior patterns* with ScriptEase. We demonstrate interruptible and resumable motivational ambient and latent behaviors for a tavern scene in a custom *Neverwinter Nights* game module.

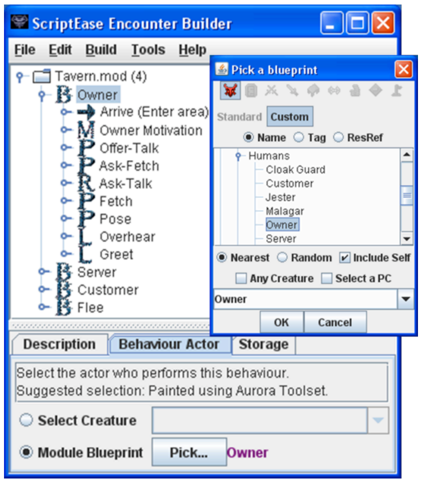
Introduction

Non-player characters (NPCs) are background characters that are controlled by the computer, rather than the player. Their behaviors are increasingly complex, as the believability of the NPCs has a significant impact on the quality of the game story. Game companies are focusing their efforts on creating immersive worlds in which the story takes precedence over other aspects of game production. They need to develop new stories quickly and reliably. However, it is currently difficult to create NPC behaviors that are non-repetitive for each of the thousands of NPCs, since custom manual scripting is required for each unique NPC behavior. Complex behaviors, such as behaviors that synchronize collaborative NPCs, are harder to implement, making manual scripting impractical (Mateas and Stern 2004). In addition, a game author who is not a programmer must rely on programmers for scripting. Therefore, current computer games have simplistic repetitive ambient NPC behaviors. It is rare for the NPCs to interact with each other, and behaviors that are interrupted are not resumable. In short, NPC behaviors are unnatural and boring.

This demonstration features three types of NPCs with behavior scripts generated from ScriptEase behavior patterns (ScriptEase 2007) in a custom tavern module using the *Neverwinter Nights (NWN)* computer role-playing game (CRPG). Frequent NPC actions are captured by design patterns that can be reused and adapted by authors in different game scenarios to produce stories. An author attaches the most appropriate behavior pattern to an NPC, and ScriptEase generates scripting code automatically. In the tavern scene, a server NPC offers drinks to patrons, fetches supplies for the owner and talks to other servers. An owner NPC asks the server to fetch supplies, offers drinks to patrons, and fetches supplies on its own. A patron NPC asks the server and the owner for drinks, walks to the bar, and converses with other patrons. During the day, some patrons enter the tavern and some leave. These behaviors are natural and interesting.

Behavior Patterns

Ambient behaviors are actions that an NPC displays spontaneously (*proactive*) or in response to a collaborative behavior initiated by another NPC (*reactive*). The NPC chooses an ambient behavior to execute at any time based on current motivation values. During the performance of an ambient behavior, the NPC may be interrupted by a game event. *Latent* behaviors are actions that an NPC performs in response to an external game event, such as the presence of the player character (PC). Collaborative behaviors introduce problems regarding synchronization, deadlock and indefinite postponement. We have developed a concurrency control mechanism (Cutumisu 2006) that is embedded in our ambient behavior patterns and it is invisible to the story author, and only partially visible to the pattern designer. The key to our model is the ability to support interruptible and resumable NPC behaviors; when a latent behavior completes, the NPC resumes the interrupted ambient behavior at the appropriate stage. A behavior pattern is composed of a set of roles (e.g. **Customer**, **Flee**) that each comprise basic (proactive, reactive, and latent) behaviors. A basic behavior (e.g., a proactive behavior **Offer-Talk**) triggers a set of tasks composed of actions. Our model ensures synchronization in a single basic behavior by preventing an NPC from starting a task before the previous task is complete, and for a collaborative behavior, until the collaborator's current task is complete. However, when a latent behavior interrupts an ambient collaborative behavior of an NPC, when an ambient behavior with a higher priority (collaborative) interrupts an ambient behavior with a lower priority (independent), or when a collaborator simply takes too long to complete its current task, the waiting NPC may spin for a new ambient behavior or even cancel the collaboration. The task construct and the four queues maintained by each NPC in our model ensure that the appropriate task is performed at any time and that the behaviors are correctly resumed.



Select the actor who performs this behaviour.
Suggested selection: Painted using Aurora Toolset.

(●) Nearest ( ) Random [✓] Include Self

Owner

(●) Module Blueprint Owner

Figure 1. The behavior patterns for the tavern module.

Tavern scenes are common in CRPGs. In a stock *NWN* tavern, patrons walk pre-determined waypoints or stand and perform a basic animation – they do not interact with each other. Sometimes, NPC interaction is simulated by groups of NPCs facing each other and performing talk animations. A tavern scene in *Morrowind* has similar NPC behaviors. In an *Oblivion* tavern, the NPCs converse with each other, but if their behaviors are interrupted, they restart rather than resume at the interrupted stage. The reason for the simplicity of NPC behaviors is that, in order to produce such scenes, a game author would have to write script code for each NPC.

Demonstration Overview

In this demonstration module, the PC observes eighteen customers, two servers and an owner performing their behaviors in a tavern scene. The PC also interacts with some of the NPCs, triggering their latent behaviors. The patterns illustrated in Figure 1 are instantiated by setting their options (e.g., the *Actor* and the *Storage* for the **Owner** behavior) to generate the behaviors displayed in the tavern. After the author instantiates the patterns, ScriptEase automatically generates the scripting code for them and the module is played in the *NWN* game. The **Owner** pattern has four *proactive* ambient behaviors. The owner fetches supplies from the storeroom, it asks one of the servers to fetch supplies from the storeroom, it offers drinks to nearby customers, and it poses by performing an animation. To collaborate, an NPC looks for any NPC with a reactive behavior on the same topic who can make *eye-contact* and who satisfies any conditions included in the initiator's collaborative behavior. The owner also performs a reactive behavior: it responds to customers' drink requests. The owner performs two latent (independent and collaborative) behaviors. When the PC walks very close to the owner, the owner greets the PC and then it returns to its ambient behaviors. When the PC walks within a range from the owner, the owner initiates a conversation (**Overhear** with topic "clue") with a customer to reveal an important story clue to the PC. In our model, an inattentive *Oblivion* merchant (Spronck 2006) can respond to a PC/NPC and behave in a more realistic manner by using appropriate latent behaviors. The server performs ambient behaviors: it offers a drink to a random customer, walks to the bar, and converses with another server. It also performs reactive behaviors: it responds to the owner by fetching supplies from the storeroom, to the customer by fetching a drink from the bar, and to a server's conversation. The server performs a latent behavior: it greets the nearby PC. The customer performs proactive behaviors: it asks a server and the owner for a drink, walks to the bar, returns to its original position, and converses with a customer. It also performs reactive behaviors: it responds to the server's and owner's offers and to a customer's conversation.

This demonstration focuses on the process of generating complex interruptible and resumable behaviors with ScriptEase. The generated tavern behaviors can be viewed in a series of movies (Movies 2007).

Conclusion

We developed a model for representing NPC motivational behaviors using generative patterns that solves the difficult problem of interacting NPCs and of interruptible and resumable behaviors. This demonstration illustrates how this model is applied in a commercial game, *NWN*, to generate complex behaviors that create more engaging interactive stories. The ScriptEase behavior patterns generate versatile NPC behaviors, since they allow the selection of behaviors dynamically, based on the NPC's current motivations and the world state. Our model provides a mechanism that enables an author to generate complex behaviors for thousands of NPCs in a CRPG, without writing code. Our solution can be applied to other game genres, since behavior patterns are created and used at a higher level of abstraction than scripting code.

References

Cutumisu, M., Szafron, D., Schaeffer, J., McNaughton, M., Roy, T., Onuczko, C., Carbonaro, M. 2006. Generating Ambient Behaviors in Computer Role-Playing Games. *IEEE Journal of Intelligent Systems* 21(5) 19-27.

Mateas, M. and Stern, A. 2004. Beyond State Machines: Managing Complex, Intermixing Behavior Hierarchies. *GDC*.

Movies. 2007. http://www.cs.ualberta.ca/~script/movies/aaai07.

ScriptEase. 2007. http://www.cs.ualberta.ca/~script.

Spronck, P. 2006. Adaptive Entertainment. *AIIDE*, Marina del Rey, USA. http://www.aiide.org/ aiide2006/speakers/spronck.ppt.