ScriptEase – Motivational Behaviors for Interactive Characters in Computer Role-Playing Games

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

Department of Computing Science, University of Alberta
Edmonton, Alberta, Canada T6G 2E8
{meric, duane, jonathan, waugh, onuczko, siegel, schumach}@cs.ualberta.ca

Abstract

ScriptEase is a tool that allows authors with no programming experience to create interactive stories for computer role-playing games. Instead of writing scripting code manually, game authors select design patterns that encapsulate frequent game scenarios, creating stories at a higher level of abstraction and being shielded from the underlying scripting language. ScriptEase has been extended to support *behavior patterns* that generate ambient behaviors for non-player characters. This demonstration shows how ScriptEase creates intricate non-player character scripts to generate compelling and engaging character behaviors. We demonstrate our ScriptEase motivational *ambient* and *PC-interactive* behaviors for a guard character using BioWare Corp.'s Neverwinter Nights game.

Non-Player Character Behaviors

A computer role-playing game (CRPG) is an interactive story where the game player controls an avatar called a player character (PC). A major determinant for the success of an interactive story is the development of the supporting characters, i.e., non-player characters (NPCs). Using AI to create NPCs that exhibit near-realistic ambient behaviors makes the game more entertaining and enhances the player's experience. Characters in CRPGs usually perform simple and predictable behaviors, especially in worlds with hundreds or thousands of NPCs. For example, in games such as Fable and Morrowind, the state-of-the-art character behaviors are repetitive, with characters that rarely interact with each other [1]. NPCs walk predefined paths, make random comments about the PC, stand still or perform a simple animation. In Sims 2, ambient behaviors are more developed, but they are dependent on the design model that is integral to the game. There are several reasons for a generally poor state of interactive characters in games, the most common being the effort necessary to script individual NPCs [3]. The programming resources required to model character behaviors are hard to financially justify, especially when the NPCs are not essential to the main plot. Creating realistic ambient behavior for a single character is challenging: even more challenging is to define realistic interactions between NPCs. Interacting NPC behaviors are rarely seen in CRPGs, since they complicate event synchronization. Halo 2 has support for "joint behaviors" [2] but the game designer still has to write custom code for these behaviors. We developed a concurrency control mechanism that solves the inherent synchronization problems [1]. Our behavior model generates engaging NPC behaviors without writing code. This model is robust, flexible, extendable, and requires minimal CPU resources.

Behavior Patterns

We added *behavior patterns* to ScriptEase [5] to express the *ambient* and *PC-interactive* behaviors of NPCs. Ambient behaviors are NPC behaviors that lack PC involvement, while PC-interactive behaviors are triggered by the NPC's perception of the PC. We constructed a *Guard* behavior pattern to illustrate our behavior model [4]. The guard performs a set of *ambient* behaviors, such as patrolling near the guarded object, resting on a bench, and checking the guarded object. The guard also performs *PC-interactive* behaviors, such as watching for the PC, warning the PC to leave when the PC gets too close, and attacking the PC when the PC gets very close. A game author uses ScriptEase to insert behavior patterns into a story or to create new behaviors without any script writing. ScriptEase generates the scripting code from the patterns automatically. We recently showed how ambient behavior patterns can easily and quickly re-generate and improve the behaviors of all ambient NPCs in the Neverwinter Nights (NWN) Prelude from the official NWN campaign [1]. This demonstration focuses on the behaviors of a guard NPC.

A behavior pattern is defined by a set of behaviors and a control model that selects the most appropriate behavior. We developed a scalable concurrency control model for interacting NPC behaviors to ensure that neither deadlock nor indefinite postponement can occur, and to ensure that interactions are realistic [1]. An *eye-contact* mechanism prevents new ambient behaviors from being initiated while an NPC is still executing behaviors in progress or when a PC-interactive behavior occurs. Each behavior pattern is

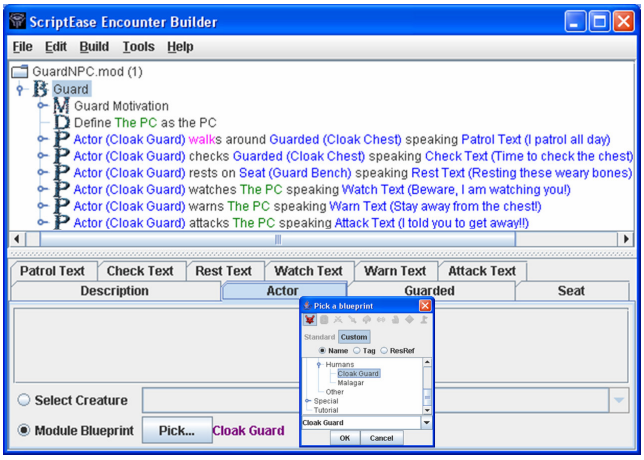

Figure 1. An instance of the ScriptEase Guard behavior pattern.

composed of basic behaviors that are re-usable and easy to assemble together.

Motivational NPC Behaviors

Each behavior pattern includes a proactive model and a reactive model. The *proactive* model selects a *proactive behavior* based on a selection mechanism. The *reactive* model specifies a *reactive chain* for each proactive behavior. The completion of each reactive behavior triggers the next reactive behavior until the end of the chain. In previous work, we presented a guard behavior pattern whose proactive behavior was selected using a static probability vector. In this demonstration, we show for the first time how motivational behaviors can be used instead of a static probability vector. Our guard NPC is motivated by *duty*, *tiredness* and a sense of *threat* to the item being guarded. Our proactive model dynamically generates a probability vector based on motivation levels, before choosing a proactive behavior. Figure 1 shows a *Guard* behavior pattern (**B**) opened to illustrate its motivational model (**M**) and proactive behaviors (**P**). The reactive chains composing the proactive behaviors are not shown. Each proactive behavior has a component that updates the values of the motivations affected by this behavior. For example, when a *Patrol* proactive behavior is executed, tiredness and threat increase, while duty decreases. For a *Check* behavior, tiredness increases, while duty and threat decrease. For a *Rest* behavior, tiredness decreases, while duty and threat increase. A behavior's motivational model can be easily modified. ScriptEase allows the creation of new motivations that can be associated with any relevant behavior. The motivational model ensures that the versatile NPCs select relevant and believable behaviors at any moment, avoiding repetition by increasing variability.

Creating and Using Behavior Patterns

This demonstration shows the process of generating ambient and PC-interactive behaviors for NPCs using ScriptEase. In particular, a custom module that was created using the NWN game illustrates a guard NPC. We also show the ScriptEase library of behavior patterns that an author can use to easily and quickly populate a game world with engaging groups of ambient and PC-interactive NPCs. A game author can compose reusable behaviors to create a new behavior pattern or add behaviors to existing patterns, without writing any scripts. We illustrate the main three steps in creating an interactive story with captivating NPC behaviors.

Step 1: Create the Module. Create the scene containing all the areas, objects and NPCs using the NWN Aurora Toolset, and save it as a game module (GuardNPC.mod).

Step 2: Use ScriptEase. Open this module in ScriptEase. Select the *Guard* pattern from our library of behaviors. Set the pattern options to specific game objects: the guard (`Cloak Guard`), the guarded object (`Cloak Chest`), the resting place (`Guard Bench`), and the text phrases spoken as part of all guard behaviors. Figure 1 illustrates the instantiation of the *Guard* pattern and the setting of the `Actor` option to the `Cloak Guard` NPC. After the options are set, choose the *Save and Compile* menu option that generates NWScript code for the NWN game engine.

Step 3: Play! Finally, play the module in the NWN game and observe the guard performing its motivational ambient and PC-interactive behaviors. The guard patrols for a while, getting more tired and worrying more about a threat. When the threat becomes high, the guard checks the chest, returning to patrol if satisfied that the guarded object is still there. Eventually the guard is tired enough to rest, but after resting must check the chest to satisfy the growing threat and then patrol to satisfy the sense of duty.

Conclusions

This demonstration shows how manual scripting can be avoided by generating NPC behaviors with ScriptEase. It illustrates how motivational ambient and PC-interactive behaviors are easily inserted into BioWare Corp.'s Neverwinter Nights game, improving the NPC behaviors, and consequently the overall game experience.

References

[1] Cutumisu, M., Szafron, D., Schaeffer, J., McNaughton, M., Roy, T., Onuczko, C., Carbonaro, M.: Generating Ambient Behaviors in Computer Role-Playing Games. LNAI 3814, Intetain 05, Italy, 2005, pp. 34-43

[2] Isla, D.: Handling Complexity in the Halo 2 AI. Game Developers Conference, 2005

[3] McNaughton, M., Cutumisu, M., Szafron, D., Schaeffer, J., Redford, J., Parker, D.: ScriptEase: Generative Design Patterns for Computer Role-Playing Games, Proc. 19th ASE '04, Austria, 2004, pp. 88-99

[4] Guard Movies. http://www.cs.ualberta.ca/~script/guard

[5] ScriptEase. http://www.cs.ualberta.ca/~script/scriptease.html