

ScriptEase - A Demonstration of Ambient Behavior Generation for Computer Role-Playing Games

Maria Cutumisu, Matthew McNaughton, Duane Szafron, Thomas Roy, Curtis Onuczko, Jonathan Schaeffer, Mike Carbonaro^(*)

Department of Computing Science, University of Alberta

^(*) Department of Educational Psychology, University of Alberta

Edmonton, Canada

{meric, mcnaught, duane, troy, onuczko, jonathan}@cs.ualberta.ca, mike.carbonaro@ualberta.ca

Abstract

ScriptEase is a publicly-available visual tool that enables game designers to easily create complex interactive stories for computer role-playing games (CRPGs), without programming. In particular, ScriptEase facilitates the automatic generation of scripting code for ambient behaviors of the numerous non-player characters (NPCs) that populate the CRPG world. ScriptEase generates this scripting code using generative design patterns, responding to the challenge of creating entertaining and non-predictable behaviors for NPCs, without the effort of writing custom complex scripts for each NPC. This demonstration describes the steps of generating complex and non-repetitive ambient behavior scripts using generative behavior patterns with ScriptEase, in the context of *Neverwinter Nights*, a CRPG from BioWare Corp.

Manual Scripting vs. ScriptEase

Computer role-playing games (CRPGs) are coming to rely increasingly on the sophistication and excitement of their stories rather than just the realism of their graphics. *Neverwinter Nights* (NWN) is an award-winning CRPG from BioWare Corp. that uses the NWScript language to expose powerful scripting facilities to professional and amateur game designers. For each scripted object in the game, this text-based scripting language is used to specify behaviors that run in response to game events. The construction of a complex story requires the designer to script a considerable number of interacting game objects, such as props and non-player characters (NPCs). The large number of objects in a CRPG virtual world requires game designers to focus their scripting effort on a privileged set of objects vital to the story line. This situation has a negative effect on the depth and complexity of the game experience. Many NPCs that could potentially enrich the game adventure, instead display repetitive or boring behaviors, due to designers' concentrated efforts on developing NPCs directly involved in the storyline. In addition to being time consuming, manual scripting causes errors due to copying and pasting scripts among similar objects and from mislabeled objects with obscure names such as "M1S04CPATRON". This problem is

exasperated by the growing number of game designers that do not have programming skills, who must rely on programmers to write their scripts. Finally, testing non-linear stories with thousands of scripts introduces additional challenges for the game design process.

ScriptEase offers a fast and easy way of solving these problems through the use of *patterns* (ScriptEase 2003; McNaughton, Schaeffer, *et al.* 2004). Frequently occurring themes in the game, such as *pull a lever – open a door*, are captured using generative design patterns that can be adapted for various game situations, with no programming knowledge required. ScriptEase generates NWScript code automatically which makes scripting more productive, eliminating most common types of errors. With ScriptEase, non-programmer game designers can create more complex and entertaining stories. We have identified four types of patterns (McNaughton, Cutumisu, *et al.* AAAI 2004): *encounter patterns* – for generating scripts attached to inanimate objects, *behavior patterns* – for generating scripts attached to creatures, *dialog patterns* – for generating conversation scripts, and *plot patterns* – for generating scripts that control story plots. This demonstration introduces *behavior patterns* by showing the steps used to create ambient behaviors for NPCs.

Ambient Behavior Patterns

Behavior patterns encapsulate ambient behaviors for NPCs. We constructed three ambient behavior patterns that are responsible for all the interactions between the NPCs in a commonly occurring CRPG bar scene. We illustrate the functionality of ScriptEase behavior patterns using this basic set of ambient behaviors, although other complex behaviors can be produced by combining the component sub-behaviors in alternative ways.

- **Server:** *proactive behaviors*: approach a random customer, approach the bar, offer a drink to the nearest customer, and do nothing (except for basic default ambient behaviors such as yawn, stretch, or look into the distance); *reactive behaviors*: make a trip to the storeroom to fetch supplies when asked by the owner, respond when a drink offer is accepted by a customer,

and make a trip to the bar to get drinks for a positive response from the customer.

- **Customer:** *proactive behaviors*: approach a random customer, approach the bar, go to their initial location within the bar, and do nothing (except for the basic background set of behaviors); *reactive behaviors*: decide whether to accept or decline when a drink offer is made by the server or the owner, and speak to the server when the server is done making a trip to fetch a drink.
- **Owner:** *proactive behaviors*: offer a drink to the nearest customer, send the server to the supply room to fetch supplies, fetch supplies by making a trip to the supply room, and do nothing; *reactive behaviors*: respond when a drink offer is accepted or declined, and speak to the server after the server is done making a trip for supplies.

Execution Architecture

Behavior patterns use an event-based control model that has two components, *proactive* and *reactive*. *Proactive behaviors* are spontaneously initiated on a random basis when the PC is near enough to observe them. Currently, proactive behaviors are chosen by static probabilities specified by the designer. This selection method can be replaced with a contextual or motivational approach. Proactive behaviors fire events that stimulate reactive behaviors to run in response. *Reactive behaviors* are only triggered by proactive behaviors or by other reactive behaviors. For example, a proactive behavior may cause a server to approach a customer and offer a drink. If the customer reacts by ordering a drink, the server will react by making a trip to the bar to fetch the drink and returning to the customer.

Each object in the game executes actions from its personal action queue in order. ScriptEase prevents new ambient behaviors from being initiated while an NPC is still executing behaviors in progress. However, without proper synchronization between action queues, the actions of several actors may interleave. Therefore, after each reactive action is completed, an acknowledgement is sent to its partner, ensuring behavior synchronization.

The set of behavior actions that implement a behavior pattern are reusable from one NPC to another and even within the behaviors of the same NPC. Several events and the behavior actions that realize them are parameterized for better reuse. For example, the *approach a random customer* and *approach the bar* events can be combined into a single *approach target* event realized by the same behavior action. Moreover, for the same event, the designer may use a parameter to select from different behavior actions that realize the event. For example, the designer may provide different behavior actions for the *decide("drink")*, event depending on who initiated the event – the server or the owner.

Demonstration Overview

The story illustrated in the ScriptEase demonstration is set in a bar scene environment typical of CRPGs. The characters are servers, customers, and the bar owner. With a few behavior patterns and the exploitation of their underlying character interaction concurrency mechanisms, rich and non-repetitive behaviors can be easily created with ScriptEase. The patterns used to enrich the bar scene story can be generalized to other types of character interactions. The game designer can easily and quickly populate the scenery with an engaging group of NPCs. All conversation text is taken from conversation files, where one of many alternate text lines is selected randomly to reduce the repetitive nature of ambient NPC utterances.

After creating the module with the Aurora toolset provided with the NWN game, the game designer uses ScriptEase to specify the behavior patterns. The module is opened in ScriptEase and three patterns are instantiated: one for the owner, one for the server, and one for all the customers in the bar. Only one instance of the customer ambient behavior pattern is necessary, since all the customers in the bar have the same tag and the ScriptEase code generator attaches the ambient scripting code to each of the customer NPCs with this common tag. The pattern parameters are bound to the game objects that we want to interact in the bar scene. The module is saved and compiled with ScriptEase and it is played in the NWN game.

The synchronization model used by ScriptEase behavior patterns is scalable to more complex interactions and the bar scene example can be generalized to cover the ambient behaviors of NPCs in other settings. For example, the same patterns, but with different parameters, can be used to generate ambient behaviors for NPCs in a mansion, where customers are replaced by inhabitants, the server is replaced by a butler, and the owner is replaced by a cook.

References

ScriptEase. 2003.

<http://www.cs.ualberta.ca/~script/scriptease.html>.

McNaughton, M., Cutumisu, M., Szafron, D., Schaeffer, J., Redford, J., and Parker, D., *ScriptEase: Generative Design Patterns for Computer Role-Playing Games*, 19th IEEE International Conference on Automated Software Engineering, September 2004, Linz, Austria, pp. 88-99.

McNaughton, M., Schaeffer, J., Szafron, D., Parker D., and Redford, J., *Code Generation for AI Scripting in Computer Role-Playing Games*, Challenges in Game AI Workshop at AAAI-04, San Jose, USA, July 2004, pp. 129-133.