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

Demonstration Instructions

University of Alberta
Edmonton, Canada

Abstract

ScriptEase is a publicly available visual scripting tool (http://www.cs.ualberta.ca/~script/scriptease.html) that enables game designers to create complex interactive stories for computer role-playing games (CRPGs), without programming. In particular, ScriptEase facilitates the generation of ambient behavior for the numerous non-player characters (NPCs) that populate the CRPG world. Instead of manually writing scripting code for the NPCs, game designers can use ScriptEase behavior patterns to produce ambient behaviors for characters in their stories. ScriptEase generates scripting code automatically using generative design patterns. Our tool responds 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.

Demonstration Instructions

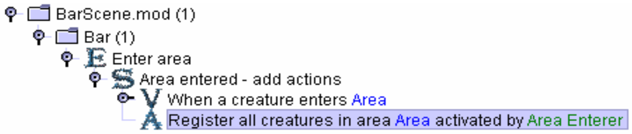
The story illustrated in the ScriptEase demonstration is set in a bar scene environment that is typical in CRPGs. The creatures that interact in the story are servers, customers, and the bar owner, and they represent a common set of NPCs that can be used to demonstrate ambient behavior patterns. With a reduced set of behavior patterns and the exploitation of their underlying character interaction concurrency mechanisms, rich and non-repetitive behaviors can be easily set up and generated with ScriptEase. The patterns used to create the bar scene story can be easily generalized to other types of character interactions. The game designer can easily and quickly populate the scenery with an engaging group of NPCs and then create other parts of the story using the time saved.

Step 1: Module Creation

The demonstration shows the bar scene module together with an owner, a server, and a few customer NPCs that the game designers creates using BioWare's Aurora toolset provided with the NWN game.

Step 2: Scripting with ScriptEase

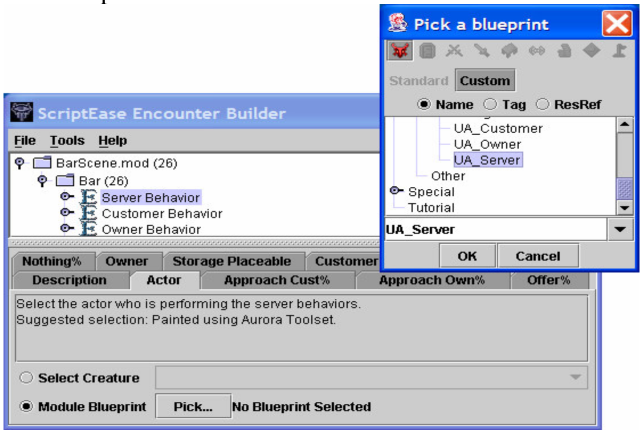
After the creation of the module with the Aurora toolset, the game designer uses ScriptEase as an interface for behavior patterns. The module is opened in ScriptEase and first, the pattern that represents the controller is instantiated. Figure 1 illustrates the Enter Area pattern that corresponds to the situation when the PC enters the area., and the registration of all the NPCs in the area.



BarScene.mod (1)
- Bar (1)
  - Enter area
    - Area entered - add actions
      - When a creature enters Area
      - Register all creatures in area Area activated by Area Enterer

Figure 1 – The NPCs are registered when the PC enters their area.

Three other pattern instances 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. The next step is binding the pattern parameters to the game objects that we want to interact in the bar scene. For example, figure 1 shows how the Actor parameter for the server is set. Finally, the module is saved and compiled with ScriptEase.



UA_Server

Select the actor who is performing the server behaviors.
Suggested selection: Painted using Aurora Toolset.

No Blueprint Selected

Figure 2 – Instantiating the *server behavior pattern* using ScriptEase.

Step 3: Play!

The module created with the Aurora toolset and scripted with ScriptEase is played in the NWN game.

The videos show only selected behaviors displayed as a result of the interaction between the owner, server, and customer. As part of the owner's proactive behaviors, the owner is *sending* the server to the supply room to fetch supplies. The owner's *send* event generates a reactive event on the server's side, *trip*, that determines the server to take the following actions:

- Speaks a random one-liner appropriate for the trip event from a dialog file with the automatically generated name trip-start
- Goes to the supply room and, as she approaches a storage chest (a placeable object in the game), she faces the object, she speaks a random one-liner for the trip event from a dialog file with the automatically generated name trip-middle, and she performs an animation that mimics the server's getting supplies.
- Returns to the location of the partner (the owner in this case), she speaks a random one-liner for the trip event from a dialog file with the automatically generated name trip-end, and she performs an animation that mimics the server's putting the supplies next to the keg.

The server's actions reflect on the partner (owner) as well, because each reactive event stores the partner in an implied definition with the event. Synchronization is achieved in this manner, as the server acknowledges the owner when she completed the trip. Consequently, the owner performs the following actions:

- Faces the partner (the server in this case)
- Performs an animation to mimic the action of taking the supplies brought by the server
- Speaks a one-liner ("Thank you for the supplies", which may vary, as it is randomly selected from a dialog file with one-liners on the same topic).

In the end of the owner/server interaction, the server acknowledges the owner and replies accordingly. At this point, both NPCs are re-registered by the controller and their behaviors continue as before their interaction.

Similar interactions are illustrated for the server-customer behaviors.

References

AIIDE ScriptEase Demonstration Materials, http://www.cs.ualberta.ca/~script