A Demonstration of SQUEGE: a CRPG Sub-Quest Generator

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

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

Abstract

Scripting the plot in a computer role-playing game requires a large number of scripts that are difficult to program, track and maintain. Game adventures often include simple plots, called side-quests, that are independent from the main plot. Side-quests are important, as they add value to the open-world appeal of the game (e.g., for acquiring experience or resources), but they still need scripts. We have designed a tool to aid in the rapid creation of side-quests. The game designer provides the game setting and a list of objects in the setting. Our tool uses this information to create an outline for the side-quests. Then we use ScriptEase, a generative design pattern tool, to generate scripts from the side-quest outlines for the Neverwinter Nights game. A game designer can also adapt these outlines after the generation process, to add value such as humour to the side-quests.

Introduction

In the not-so-distant past, Computer Role-Playing Games (CRPGs) were authored by a small team of programmers and consisted of simple graphics, simple sounds, and a simple story. Just as the power of computers has increased dramatically, so has the quality of the CRPGs being produced. This presents a challenging problem for CRPG authors since an immersive and entertaining interactive story is difficult to create. Not only must the game designers be technically creative, they must be capable of crafting better non-player characters (NPCs), plots, dialogue, action, and interaction – in short better stories.

Often a CRPG story is made modular using quests. Each quest is a small story that focuses on a small subset of NPCs, NPC interactions, and settings found in the story. A complex quest will involve the player character (PC) in many different interactions (with NPCs, containers, items, etc.). These interactions are called encounters. Each encounter requires a script to record and control the game state. This makes scripting the quest difficult, since all the scripts must work together to maintain a cohesive story.

Many quests are similar enough that we introduce the notion of a *quest* pattern. *Quest* patterns are a class of design patterns (Gamma *et al.* 1995) that serve to provide a reusable solution in scripting quests (Onuczko 2007). A *quest* pattern contains the encounters that occur in the quest and also describes a structure that determines when the encounters are available for the PC to perform.

In CRPGs, a side-quest's primary purpose is to add breadth to the game by increasing the player's freedom. Since these quests do not need depth, most can be unaltered instances of a *quest* pattern. Instead of having a game designer create hundreds of side-quests for a commercial CRPG, many of these side-quests can be rapidly created through instantiating *quest* patterns whose options are automatically picked by an intelligent system.

This demonstration describes our Sub-QUEst GEnerator (*SQUEGE*), a tool that aids in the rapid development of side-quests in CRPGs by generating *quest* pattern outlines. The goal of this research is to generate sub-quests quickly and conveniently. Given a scene (setting, NPCs, items, containers, etc.), a game designer can "push a button" and a sub-quest is automatically generated. Parameters can be set to make the generated quests as simple or as complex as desired. The designer can generate hundreds of different side-quests quickly and easily (guaranteeing variety and enhancing the game experience). In effect, *SQUEGE* could be used to facilitate the rapid generation of important game content, reducing costs and improving the quality of the product.

Quest Patterns

To understand how *SQUEGE* works, some knowledge about *quest* patterns is necessary. Each quest pattern consists of a list of quest points. Each quest point has a *label*, a list of quest points, called the *enablers*, that enable the quest point, and an encounter that the PC must perfrom to have the quest point become reached. When any of the *enablers* is reached, the quest point becomes enabled. The *enablers* list may also specify that the point is initially enabled at the start of the quest. To reach a quest point, it must be enabled and its encounter must occur.

There are three different types of quest points. When a PC reaches a *normal* point, all points in the quest become disabled except for those that have this point in their *enablers* list. An *optional* quest point is a point that does not need to be reached to complete the quest. Reaching an *optional* point does not eliminate any previously enabled quest points. The final type of point is a *close* point that completes the quest. When a closed point is reached, all points for the

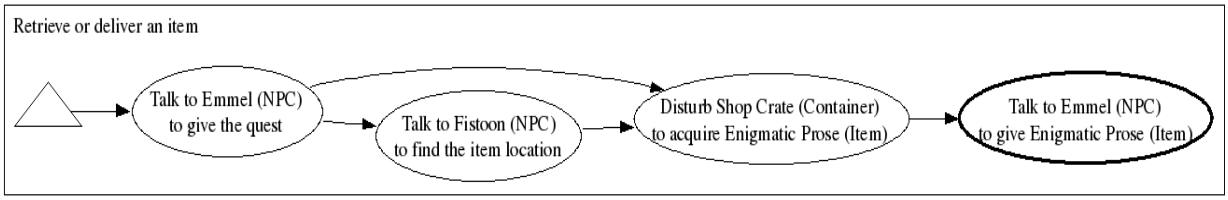

Figure 1: A sub-quest outline generated by *SQUEGE*.

quest become disabled and no new points can become enabled.

Quest patterns are specific enough that creating the scripts for them is a straightforward process. Each *quest* point needs one script. The associated encounter describes which event will trigger the script, while the *enabling* points and type of *quest* point determine how the script alters the game state of the quest. When using a generative design pattern tool, such as ScriptEase (McNaughton *et al.* 2004), the scripting process becomes automated.

Using *SQUEGE*

To show how *SQUEGE* works we give an example that generates a quest in Neverwinter Nights (NWN) by Bioware Corp. While this example only shows the generation of one simple quest, *SQUEGE* can be used to generate multiple quests of arbitrary complexity within the same CRPG adventure.

First, a game designer creates a new game adventure using the NWN Aurora Toolset, a CAD tool for the game. The author creates a city setting with several buildings that the PC can enter. The author then prepares the game adventure for quest generation by creating several NPCs, containers, and items that will populate the game.

SQUEGE is currently external to the Aurora Toolset so the game designer must list all the NPCs, containers, and items that can be used in the side-quest. Next, *SQUEGE* automatically generates a side-quest by first selecting a *quest* pattern from its catalogue of patterns. *SQUEGE* instantiates values for the various options of the quest pattern and produces a graph that acts as an outline for the *quest* instance. This outline contains all the information required to produce scripts for the side-quest.

Figure 1 shows the graph that *SQUEGE* generates for the game designer. This *retrieve/deliver an item* side-quest instance has the PC performing four encounters: beginning the quest by talking to Emmel, finding the location of the item by talking to Fistoon, acquiring the item in a shop crate container, and returning the item by talking to Emmel again. The triangle represents the start of the side-quest and it enables the first quest point in which the quest is given to the PC.

The game designer creates the conversations for Emmel and Fistoon in the Aurora Toolset. *SQUEGE* does not attempt to do this, as writing dialogue is one of the game designer's specialties. The author may add humour and specific story-related references to make the conversations interesting and unique.

The author uses the outline to instantiate the corresponding *quest* pattern in ScriptEase. The process is straightforward as the author only specifies the options for the pattern.

Now the author has the opportunity to adapt the side-quest instance in ScriptEase to add a creative touch to the story. One of these adaptations could be to have a bandit NPC appear and attack the PC when the item is acquired. Such adaptations are quick and easy to do in ScriptEase.

At this point, the game designer has finished generating a side-quest and can now play the adventure in NWN. If the process is repeated, the designer would end up with a completely different side-quest using different NPCs, containers, and items in the same setting.

Demonstration

This demonstration showcases the entire side-quest generation process of *SQUEGE* for NWN. First, the demonstrator creates a game setting in the Aurora toolset. Then, *SQUEGE* generates a sub-quest outline from its pattern catalogue. The conversations required for the sub-quest are created in Aurora. Next, a pattern for the sub-quest outline is instantiated in ScriptEase. Finally, the sub-quest is played in NWN.

References

Gamma, E.; Helm, R.; Johnson, R.; and Vlissides, J. 1995. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley.

McNaughton, M.; Cutumisu, M.; Szafron, D.; Schaeffer, J.; Redford, J.; and Parker, D. 2004. ScriptEase: Generative Design Patterns for Computer Role-Playing Games. In *Automated Software Engineering*, 88–99.

Onuczko, C. 2007. Quest Patterns in Computer Role-Playing Games. Master's thesis, University of Alberta, Edmonton, Alberta, Canada.