D-Quester

Capstone Proposal

Jeremy Jones

2014

Project Description

D-Quester is a dynamic quest management system. This system is designed to be fully implementable in existing and in-development games. This system uses nodes to handle quest progression along with events and states to handle player rewards, changes to the environment, and other user-defined behaviors. D-Quester is implemented in Unity as a custom component that extends Unity’s editor. Along with a custom menu this quest creation medium provides an easy way to create and manage quest objects and the associations between them. In order to demonstrate the features and usability of D-Quester, I will build two separate games in Unity using the quest management tool.

Target Users

D-Quester is designed to be used by game developers and designers. Upon completion, this project will be fully implemented in Unity allowing developers familiar with this game engine to easily make dynamic quests for their games. Additionally, the questing manager and API will be coded in C#. This separation allows any programmer to implement D-Quester into other game engines with some additional effort. In conclusion, any game developer, designer, or programmer making a quest or task based game would be the target user of this product.

Features List

* Allow the user to create quest objects and add them to existing game objects.
* Allow the user to connect quest objects in a node-based relationship to simulate quest progression. Each node may have as many children and parent nodes as desired.
* Allow the user to manipulate built-in states of questing objects and define their own states. The default states would include “not started”, “in-progress”, “hidden but in-progress”, “completed”, and “failed.”
* Allow the user to subscribe methods to events fired by quest objects in order to add custom logic.
* Allow the user to define rewards for completing individual quest nodes. The definable rewards include any numeric and Boolean data such as experience points, gold, reputation, and access to areas, actions, and tools.
* Allow the user to subscribe to an event triggered when a quest node is completed. This event would normally be used for rewarding a player with items or increasing bag space.
* Allow the user to define a player’s inventory along with the number of free spaces required in their inventory when a quest is completed to insure there is room for any items received.
* Allow the user to save the setup of a node as a preset accessible from a custom context menu.
* Allow the user to modify the default states that new quest objects are assigned when created.
* Allow the user to add a quest object’s states to a list of presets on the custom context menu. When one of these presets is clicked, the currently selected quest object will have its list of states changed to match the selected list.

Stretch Goals

* Have another student implement D-Quester in a small game.
* Add custom user-defined events to be added to quest objects.
* Add default support for timed quest nodes. These quests would expire and be automatically failed if a set amount of time passed while a quest object was in certain states defined by the user.
* Add random event support. These quests would have a chance to trigger every so many ticks or would only be triggered when certain user-set requirements were met.
* Add explicit achievement support. These objects would use the quest object system but have their own standalone set of nodes. Additionally, achievements would have separate convenience methods related to adding achievement points to a player’s total and displaying notifications when an achievement is completed.
* Add quest objects that have their availability to be started or completed change based on the time of day or with some other form of user-defined time range.

Technical Specifications

I will be using Visual Studio 2012 to code the base quest manager with C#. Additionally, I will be using Unity to showcase D-Quester. With Unity’s scripting API, I will have a custom component and custom context menu for adding different quest objects. All Unity-related programming will be coded in C# (.NET 2.0) using the Visual Studio 2012 Tools for Unity plug-in. D-Quester will be designed to run on the Windows operating system and will be compatible with Windows 7 or later versions through Unity’s client with version 4.5 or later.

Phase 1

Week 1 Goals

I plan on finishing the first iteration of the base quest system and API. This includes quest objects, events, and reward system handling.

Week 2 Goals

I will add the custom unity component for quest objects and extend Unity’s editor. Additionally, I will begin work on the context menu to allow quest objects to be created from it with both a default option as well as from custom preset options the user will be able to set.  
Justification

I chose this Capstone because of my passion for role-playing games. I have played many different games with varying types of story and game play. While some offer variation in the form of alternate endings and limited choices at key moments, very few allowed for a fully dynamic choice system. I feel this project is Capstone worthy because of the technical challenge in extending Unity’s editor as well as the iterative process of making a tool to be used for development. Unity is optimal for making games with the existing tools but can be very challenging when attempting to extend its functionality. Additionally, this will be my first experience creating a developer tool for use by other people. I plan to get feedback from other students once my tool is user-ready and to make changes based on the feedback I receive. This project will help me become a better developer because it involves elements of game development I have not had experience working with before and will likely be expected to perform in industry. Starting out at most companies will require me to become familiar with in-house technologies as well as extend the functionality of such resources as the need arises. In addition, if I do make tools, they will need to be user-friendly and be well documented for the other developers that use them. Working in Unity I will learn its existing API. When finishing iterations of my tool, I will be able to learn about the areas I need to improve upon in order to make my tools more functional and user-friendly in the future.