PNQ: Portable Non-Player Characters with Quests

Jafar Al-Gharaibeh and Clinton Jeffery
Computer Science Department
University Of Idaho
Moscow, USA
jafara@vandals.uidaho.edu jeffery@uidaho.edu

Abstract—**There is a growing interest in using game-like virtual environments for education. Massively multiuser online games such as World of Warcraft employ computer controlled non-player characters (NPCs) and quest activities in training or tutoring capacities. This approach is very effective, incorporating active learning, incremental progress, and creative repetition. This paper explores ways to utilize this model in educational virtual environments, using NPCs as anthropocentric keys to organize and deliver educational content. Our educational NPC design includes a knowledge model and a user performance model, in addition to the physical traits, behavior, and dialog model necessary to make them interesting members of the environment. Web-based educational content, exercises, and quizzes are imported into the virtual worlds, reducing the effort needed to create new NPCs with associated educational content. The NPC architecture supports multi-platform NPCs in two virtual environments, our own CVE (Collaborative Virtual Environment) and Second Life.**

Keywords- virtual worlds; CVE; NPC; quest; MMO.

I. INTRODUCTION

The spectacular success of Massively Multi-user Online (MMO) games has led to a large amount of interest in educational multi-user virtual environments. World of Warcraft (WoW) and similar games have demonstrated both the mass appeal and the potential of this genre. Organizations such as the U.S. Army are using custom games for recruiting and training purposes, and NASA has proposed an educational MMO to use a space theme to promote interest in science and engineering education and careers.

As yet, no one has produced an educational MMO with a significant following. MMOs take years and millions of dollars to develop. A few institutions such as Harvard and Indiana have developed small-scale educational virtual environments by hiring game companies to write them atop proprietary game engines or using a custom version of a commercial virtual world such as ActiveWorld, but most educators that wish to work with MMOs find their way to a general-purpose MMO such as Second Life.

Second Life's main focus is on user-created content. The landscape, buildings, virtual objects, and the appearance of the avatars are largely the result of end-user actions. Educational institutions have presences there for marketing purposes and on-line lectures and discussions. They are in the process of exploring the educational potential of Second Life, but the cost of building educational content (beyond virtual buildings and meeting places) is high. Second Life does not directly support the quests and activities that are one of the main reasons to play MMOs; quests resemble the activities that are the mainstay of conventional education software. Similarly, Second Life has no model of experience or skills development which are a major motivator for users to play MMO games.

Second Life's scripting language, called LindenScript, enables virtual objects to incorporate behavior, and access external content via protocols such as HTTP. One can build anything with this framework. However, without support for modeling user activities and experience, the cost of developing substantial educational applications in Second Life is prohibitively high for most educators.

The project described in this design paper began not within Second Life but rather with a custom educational virtual environment for computer science distance education, called CVE. With the goal of enabling distant students to attend lectures and office hours, and do homework assignments and labs within the virtual environment, initial efforts focused on reproducing a local CS education environment, including 3D representations of two physical CS departments, avatars, text and voice chat, and interactive collaboration on common CS tasks of editing, compilation, execution and debugging. The initial project was placed on Source Forge for public access at cve.sourceforge.net, but its authors had to face the question: who cares? Perhaps some developers of Second Life projects have faced a similar dilemma: the virtual environment lacked the compelling aspects of discovery, quest activities, and experience levels and advancement which make MMOs special. It was decided to solve this problem for both the CVE and Second Life virtual environments, in order to learn from the comparison. The central building blocks for the experiment are the computer-controlled non-player characters (NPCs), who serve as tutors and record keepers for users' accomplishments. The design for the NPCs and their quests presented in this paper is called: Portable Non-player character tutors and Quests (PNQ, pronounced "pink").

II. RELATED WORK

This work relates to the design of NPCs and also to education in virtual environments. Redfern and Naughton [5] discuss the use of modern technology and the advances in virtual environments research in distance education. They propose that virtual environments, specifically where users can interact with each other and with NPCs, are suitable tools to improve education.

The Belief-Desire-Intention (BDI) agent architecture enhances users' interaction with NPCs by developing behavior models that resemble human behavior [12]. Merrick and Maher presented a design for NPC behaviors in computer games based on motivated reinforcement learning [6]. They also presented an adaptive NPC model which considers the impact of the changing environment in open-ended worlds on the NPC. The adaptive model evolves and adapts the NPC to the changes in dynamic environments and does not limit its behavior to the pre-programmed rules [7].

CHI Systems, under contract to the U. S. Army Research Institute developed a training system called Virtual Environment Cultural Training for Operational Readiness (VECTOR). In this system they applied experiential scenario-based virtual environments to train solders in cultural familiarization. They incorporated cognitive-model-controlled NPCs that can evaluate and respond to the cultural propriety of trainee's actions [8].

Mateas and Stern developed a Java-like behavior language for story-based agents called ABL. The language has an interface to communicate with a robot's sensory motor system. It supports joint actions, sequential and parallel behavior. The following example from [9] illustrates the feel of code written in ABL.

```
parallel behavior StartTheHandlers() {
  with (persistent, priority 20)
    subgoal handlerDAGreet();
  with (persistent, priority 15)
    subgoal handlerDAReferTo_grace();
  with (priority 10, ignore_failure) subgoal
    handlerPreInviteAptMove();
}
sequential behavior BeatGoals() {
  with (persistent when_fails)
    bgOpenDoorAndGreetPlayer();
  with (persistent when_fails)
    bgYellForGrace();
  with (persistent when_fails)
    bgInviteIntoApt();
}
```

The use of multi-user virtual environments for education has been discussed in the Quest Atlantis project (QuestAtlantis.org). Quest Atlantis is a learning and teaching computer game that leverages commercial gaming environment strategies to develop a 3D multi-user environment to immerse children, ages 9-12, in educational activities. It allows children at participating elementary schools to engage in a virtual world and perform educational activities, chat with other users, and build virtual spaces in some designated plots. [10, 11]

Several interesting experiments have added non-player characters to Second Life. No existing Second Life NPCs deliver tutorial quests from declarative specifications as in the case of PNQ. Art Fossett's blog describes a non-player character created as a virtual object [2]. Making an object look humanoid is a challenge in Second Life, but can be accomplished using sculpted primitives, which are a restricted form of 3D model that graphic artists can produce with commercial grade tools at substantial effort. Fossett couples this humanoid-looking virtual object with an external "chat" program called a PandoraBot, which implements AIML chat and plays a role similar to the PNQ dialogue model. Doron Friedman et al built a Second Life NPC by taking an ordinary user avatar and attaching a virtual object (a ring) to it that turns the avatar into a puppet controlled by an external program. This NPC can move around the environment, albeit with very simple rules for essentially random movement [3].

III. NON-PLAYER CHARACTER TUTORS

NPCs are an important component of role playing games and MMOs, presenting activities and the storyline of the game. Computer-controlled NPCs can be enemies, allies, monsters or pets, but their most important role is that of quest giver or assistant to the user who goes on an adventure. Quest givers are also important because educational content can be delivered to users through quests in educational virtual worlds. This technique can be used to draw users into tutorial quests. Figure 1 shows several avatars from the CVE virtual world including NPCs. Figure 2 shows avatars from Second Life. A quest giver NPC in CVE has a red ball over his head, while in Second Life the NPC has a red ring around his arm. This marks NPCs with available quests as persons in the environment with whom the user has a reason to interact.

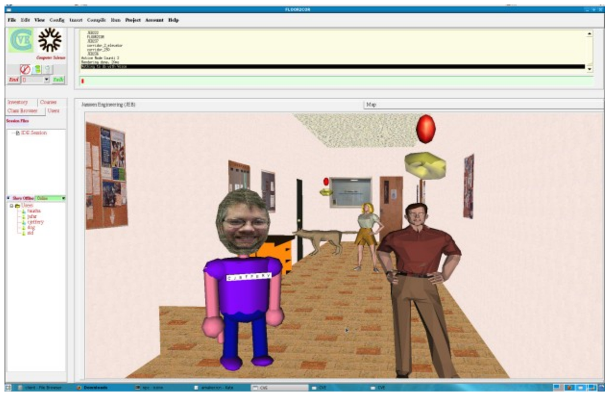
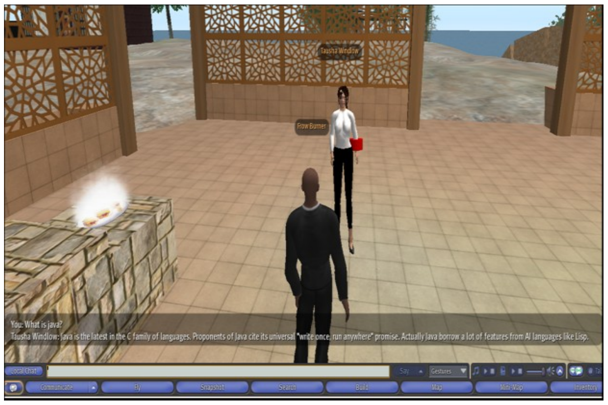


Figure 1. NPCs and other users in CVE. An NPC is marked with red ball above their heads.



You: What is java?
Tausha Windlow: Java is the latest in the C family of languages. Proponents of Java cite its universal "write once, run anywhere" promise. Actually Java borrow a lot of features from AI languages like Lisp.

Figure 2. An NPC and another user in Second Life. The NPC is marked with a red ring around her arm.

A PNQ is created much as a regular (human-controlled) character. End users can utilize their regular account or an auxiliary account to create an NPC character, adding quests and activities and making them available to other users. When the user is logged off, their avatar is still present on the system, controlled from a remote NPC client just like a regular user and functioning as a quest-giving NPC and a virtual secretary, interacting with other users as instructed by the player. Many educators will choose to create multiple characters playing specific educational roles, giving them an artificial personality and a set of activities for other users, and leaving them under the control of the computer full-time.

A. *NPC Profiles*

Creating a PNQ NPC consists of specifying its knowledge, dialogue, and behavior models in a profile file. The profile is read at start up via HTTP by an NPC agent, a program that "plays" the NPC as a client on the virtual world server. An NPC profile is a file containing NPC details in simple HTML (for handwritten profiles) or XML (for machine generated profiles) format. An NPC profile can be created and maintained as a webpage that contains the following sections. In HTML they are each given in a named anchor tag. Although a graphical wizard for creating profiles is available, many NPCs can be created manually by copying a template and changing the content details.

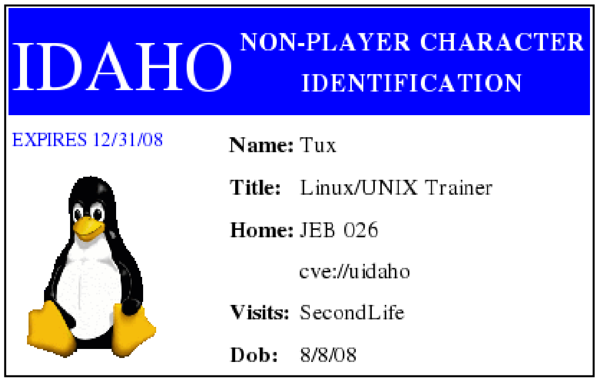
id	An "ID card" presentation of the NPC, suitable for an "inspect details" operation in a game. The id provides an image, name, and basic attributes. Figure 3 shows an example ID card.
knowledge	A specification of the NPC's knowledge model consists of a teaching section with a bulleted list of named links to quests. NPC knowledge also includes a more dynamic experience (user model) database that is not part of the profile.
dialogue	A specification of this NPC's verbal capability.
behavior	A specification of this NPC's active (e.g. mobile) behavior. The four kinds of PNQ behavior specification include stationary, wanderer, routine, and companion.
avatar	A specification of this NPC's avatar (link to 3d model file, dimensions, and textures).

B. *Knowledge Model*

PNQ NPCs use two types of knowledge: the quests they offer, and what they remember about users from past quests. The former is almost static, refined occasionally by the NPC's creator. The latter is dynamic and accrues during game play. The management of NPC quests is discussed in section IV.A below. The NPC's user experience model is discussed in section IV.B.

C. *Dialogue Model*

In World of Warcraft, players interact with NPCs using popup menus. This contrasts egregiously with human player interactions. Some MMOs such as Everquest enhance the realism of NPCs by forcing NPC interaction through the chat channel. NPCs that chat feel more natural but are frustrating when they fail to respond usefully to a player's conversation.


IDAHO NON-PLAYER CHARACTER IDENTIFICATION

EXPIRES 12/31/08

Name: Tux

Title: Linux/UNIX Trainer

Home: JEB 026

cve://uidaho

Visits: SecondLife

Dob: 8/8/08

Figure 3. An NPC ID card.

Chat-based interaction is more "portable" across virtual worlds. The PNQ NPC's dialogue model consists of chat rules written in AIML [4] augmented with offers to undertake available quests. The AIML rules determine how the NPC replies to player utterances.

D. *Behavior Model*

The PNQ NPC behavior model provides rules for NPC movements and responses to external stimuli. Four NPC behavior types are supported: stationary, routine, wanderer, and companion. A stationary NPC does not move from a specified home location. A routine NPC regularly does specific tasks including movements at predetermined times. A wanderer is an NPC that moves randomly within a prescribed domain. A companion NPC accompanies a player on a destination-based quest.

A behavior language is needed to write the rules that describe how an NPC acts in the virtual world. As discussed in section II, ABL is an example of such a language [9]. ABL and other scripting languages were not chosen for this project because they were viewed as too low-level. Many of the expected NPC authors are educators with little or no programming background. Thus, a new high level declarative behavior model language was invented for PNQ. The new language captures simple behavior scenarios that an NPC should be able to do in an educational virtual world, including movements and offering quests. Figure 4 shows a fragment of the grammar.

```
BehaviorModel : Type "{" States "}";
Type      : Wanderer | Stationary | Routine | Companion ;
States    : State States | State ;
State     : StateName optSchedule "{" Actions "}" ;
Actions  : Action Actions | /* empty */ ;
Action   : Login | WalkTo | SetState | Offer| Redirect | Logout |
             Teleport;
Login      : LOGIN TIME IN Place ;
WalkTo    : WALKTO Place optAT ;
SetState : SET StateName;
Offer      : OFFER Topics;
Topics     : Topic "," Topics | Topic;
```

Figure 4. Part of the grammar for the PNQ NPC behavior model.

The following is a stationary NPC example. The NPC logs in at 8 AM in room jeb 228 and offers quests from its profile in the "help" topic until 5 PM.

```
Stationary {
  default : {
   login   8:00 in place(jeb 228)
   offer help
   logout 17:00
   }
  }
```

If no "offer" statements appear in a behavior model, the default behavior is to offer all eligible quests listed in the NPC profile. The following is a more complex example that features a Routine NPC. The NPC logs in to the virtual world and walks to different places, provides office hours and "teaches" by offering quests. In this model, the NPC spends different parts of the day in different states. The state named MWF is scheduled (by a string notation inspired by the UNIX cron(1) tool) to commence at 10 AM every Monday, Wednesday, and Friday.

```
Routine {
    default : { login in place(jeb 226) }
    MWF "0 10 * * MWF": {
        login in place(jeb 005 corridor)
        walkTo place(jeb 226)
        state(officehours)
        walkTo place(jeb121) at 10:30
        state(teach)
        walkTo place(jeb 226) at 11:30
        logout
        }
    officehours {
       offer debug, question
       }
    teach {
       offer CS210, programming languages, Lisp, Flex, Bison
       redirect debug : Dog
       }
    }
```

IV. QUEST ACTIVITIES

In games like WoW, quest activities are used to teach the game itself as well as to entertain. WoW features about 10 types of quests, requiring the user to perform various tasks, such as combat, exploration, delivery, and gathering or manufacturing of virtual items. Other MMOs provide additional kinds of quests. Many of these quest types involve activities that could be utilized in educational quests. Educational MMOs will define many new quest types for learning purposes. In a geology class, users might get rewards for exploring a virtual world that reflects real places and collecting information about those places: minerals they have, climate, water resources, etc. By analogy to Second Life's user-defined virtual objects, educational MMOs will need end user teacher-defined quest activities. The NPCs and quests described in this paper are intended to fill that role.

Quests are a primary mechanism for tutorial learning. Quest specifications resemble UML use case descriptions [1]. The kinds of steps are limited to those observable by NPCs interacting with the user in the supported virtual environments. The main differences between a quest and a use case description are that a quest may contain auxiliary content (such as quizzes and demonstrations) that are used to measure completion of the quest steps, and a quest lists rewards for completion, if any. Quizzes and demonstrations will often need to be external references to pools of questions. The difference between quizzes and demonstrations is that a quiz is delivered and answers interpreted by an NPC agent directly, while a demonstration involves an in-world interaction (in this case, a session with a tutorial UNIX command-line shell) that is monitored by an NPC agent. Evaluation of deeper understanding may require offline human evaluation, or fall outside the realm of what an NPC Tutor can reasonably perform.

A. *Quest Repository*

PNQ Quests are maintained in the same way as NPC profiles: they are HTML files linked from the knowledge section in the NPC's webpage. The quest webpage includes several sections. A quest builder tool facilitates the process of creating new quests.

Figure 5 shows an example quest from the domain of computing. Figure 6 is a screenshot of a quest being offered in a response to a user clicking a quest red sphere above an NPC. Figure 7 shows a quest activity example from Second Life where the NPC is informing the user about an error in a Java quest that the user has submitted.

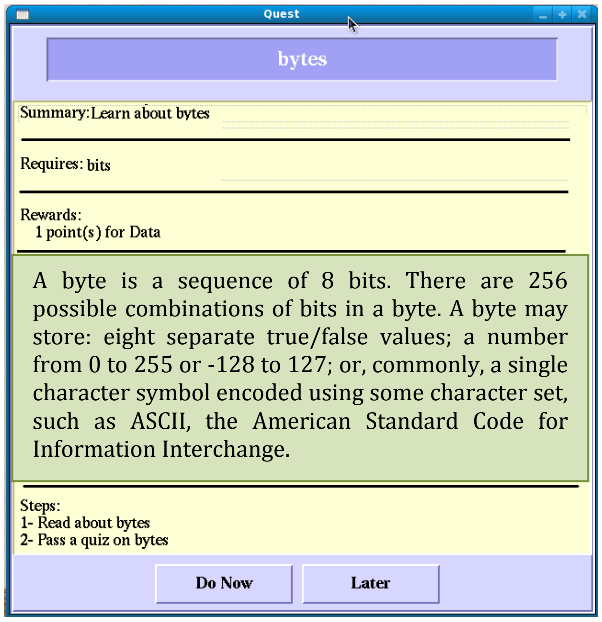
```
Name     :  ls
Summary : Learn the basics of the ls command.
Requires  : Files, Directories
Steps       :Read the UNIX manual page for ls.
Pass a quiz on ls command line options.
Demonstrate "ls" for Tux.
Rewards  : UNIX: 1
Quiz (2/2 to pass)
How can you get a long listing that shows file permissions and
          size?
> ls -l
How can you list all files in all subdirectories?
> ls -R
Demo (2/2 to pass)
Show me a simple listing of the root directory.
> ls /
Show me a listing of the current directory, sorted by the time
          each file was last accessed?
> ls -t
```

Figure 5. An example quest as it appears in a web page.

B. *Quest Rating and User Reward via Peer Review*

Players need to be motivated to perform quests, and their accomplishments need to be recognized. Each quest completed from any virtual world needs to be rewarded and remembered. Rewards in the form of virtual objects or clothing might be duplicated on multiple virtual worlds, while other rewards may be specific to a particular world, such as Linden dollars, or a new character ability.

The main kind of reward that matters to PNQ is the experience points in specific topic areas that enable a character to undertake more advanced quest activities. In the ls activity described in Figure 5, two specific previous quests (tutorials on Files and Directories) had to be completed before the NPC offered this user the ls quest. Completing the ls quest enables any quest that depends on it specifically, and also awards a point of general UNIX experience.



bytes

Summary: Learn about bytes

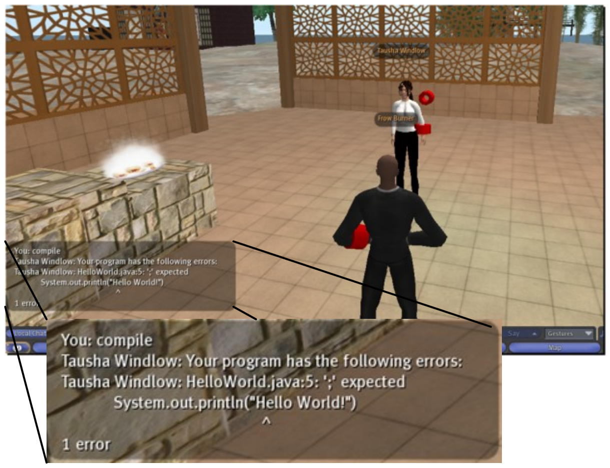
Requires: bits

Rewards:
1 point(s) for Data

A byte is a sequence of 8 bits. There are 256 possible combinations of bits in a byte. A byte may store: eight separate true/false values; a number from 0 to 255 or -128 to 127; or, commonly, a single character symbol encoded using some character set, such as ASCII, the American Standard Code for Information Interchange.

Steps:
1- Read about bytes
2- Pass a quiz on bytes

Figure 6. A sample CVE quest invitation dialog.



```
You: compile
Tausha Windlow: Your program has the following errors:
Tausha Windlow: HelloWorld.java:5: ';' expected
        System.out.println("Hello World!")
                                          ^
1 error
```

Figure 7. Quest activities in Second Life. The user has submitted a "hello world" Java quest. The NPC informs the user that there is an error in the program.

When teachers create new tutorial quests, they vary in terms of how fun they are, and their educational value. In PNQ, the fun is rated by user upon quest completion. Additionally, educational content is evaluated by domain experts. When an activity is created, its author suggests its category and experience point value, but those values are honored only after the activity has been rated.

V. Design and Implementation

The PNQ NPC client is designed as a standalone entity, separate from the server. An NPC client connects to the server like a regular user, with an NPC indicator flag. This design frees the server from NPC management, makes the NPC more flexible by allowing the NPC to run from any machine and allows a human to "play" an NPC. Moving the NPC to a different machine entails moving the NPC's dynamic knowledge about users and their quests. This problem is partially solved by logging dynamic data on the server, if it supports that, which updates the NPC upon its launch. Since Second Life does not have a user model, quests completed there are logged to the CVE server.

A. *NPC's Architecture*

The NPC client design mirrors the NPC profile. The following is the list of the NPC's major components:

1) A knowledge engine: composed of two parts. The quest knowledge is almost static and gets updated irregularly. The NPC also maintains dynamic knowledge of the users' information and their quest activities such as their current position and their active and completed quests.

2) A behavior engine: dictates how the NPC will behave and move around in the world based on the NPC profile.

3) A chat engine: analyzes the incoming chat messages and generates proper response if possible. Chat messages are categorized either as general chat or messages that involve questions or answers about the quests the NPC is providing.

At the top level there is an I/O interface that manages data transfer between the NPC, virtual world server, HTTP servers and also the disk. Figure 8 shows the different NPC components.

B. *Implementation Discussion*

The PNQ NPC client is implemented as part of the CVE project. The source code for the PNQ NPC client is available at cve.sourceforge.net. CVE is written in Unicon, a very high level object-oriented and goal-directed programming language [13, 14]. The NPC client is independent from the regular user client in CVE and could be implemented using any programming language. However, Unicon was chosen to implement the NPC client because it provides:

1) a very simple interface to the standard internet protocols such as TCP and HTTP [13].
2) high level built-in string parsing features that make it easy to parse the NPC profile and quest data.

The NPC client starts by downloading the NPC home page and quest pages that define the profile of the NPC. It then initiates a TCP connection with the virtual world server. Upon a successful login, the new NPC is available in the virtual world accepting interactions from other users.

C. *Network Protocol*

The PNQ NPC network protocol uses string messages consisting of a command name followed by arguments. The NPC client recognizes the following messages coming from the virtual world server and other users: chat messages, messages about other users' locations in the virtual world, and messages about quest requests and activities. All commands must begin with two forward slashes (//). Some commands have arguments to pass information to/from the NPC. This protocol is for the CVE and not used for Second Life, which has its own protocol, except that NPC's in Second Life report quest completion to the CVE server using this protocol. A summary of the commands are listed below.

User presence commands:

users -brings up a list of all the other users who are currently online.

avatar -informs about a new user who just logged in.

move -informs about a specific user movement (x,y,z and direction).

Chat commands:

say -public chat message sent to all users.

tell -private chat message sent to the intended user only.

Quest activities commands:

Npcmsg -this command communicates information about quest activities between the NPC, server and clients. This command takes several arguments containing all of the information necessary for each quest activity. The following is a short summary of these arguments. Some arguments have extra information (quest title/score) omitted below for simplicity:

Quest LookFor: Asks the NPC if he has available quests.

Quest Halo: Informs the client that the NPC has available quests.

Quest GiveMe: Asks the NPC to send the next available quest.

Quest URL: Sends the client a quest URL.

Quest Accept: Informs the NPC that the user has accepted the quest.

Q*uest Cancel*: Informs the NPC that the user has cancelled or abandoned the quest.

Quest Done: Informs the NPC that the user has finished the quest.

Figure 9 shows a scenario where the quest messages are used to start a new quest. The process starts when a user client clicks the red ball marking a quest over an NPC. The click sends the message "Quest GiveMe" to the server and then it gets forwarded to the specific NPC. The NPC finds the next available quest for that user based on the knowledge it has, then sends back a quest title along with its URL. The server gets the message and checks whether the user has already completed the quest or is currently undertaking the quest. Of course this is not typical because the NPC maintains a list of completed and active quests for every user, but if the NPC process gets restarted, the quest protocol allows it to reload user dynamic knowledge on demand. The scenario in Figure 9 shows the case when the NPC needs to get updated. After a quest request message, the server replies to the NPC informing it that this user has completed the quest. The NPC then adds this piece of information to its database and finds another available quest for the user and sends it back to the server, which in turn checks again whether the user has already completed the quest. This time the server approves the new quest and forwards it to the client. The client gets the message and downloads the quest from the specified URL or loads it from the disk if it is stored on the local disk. If the user accepts the new quest, which is the case in the scenario we have here, it sends the message "Quest Accept Title(T)" back to the server. The server adds the specified quest to the active quest list for the user and forwards the message to the NPC which in turn also adds the quest to the active quests list for that user. In some virtual worlds, where the server cannot be involved in quest

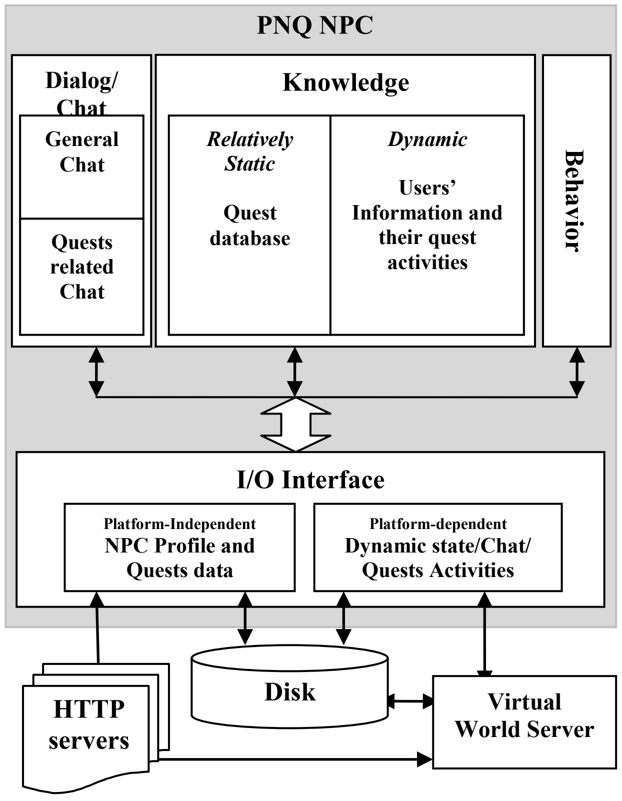


Figure 8. PNQ NPC Architecture.

activities, the NPC relies on its knowledge to determine if the user qualifies to take a specific quest. In this case, if the NPC has to be moved to a new machine, all of its knowledge has to be moved along with it.

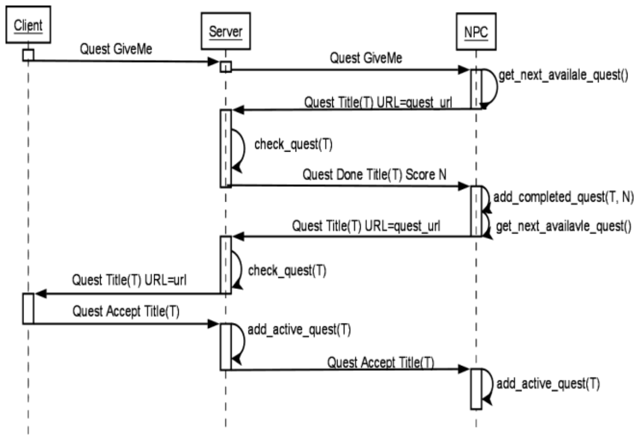


Figure 9. PNQ NPC quest messages get transferred between the NPC, server and the client to start new quests.

D. Source Code Organization

At the top level view, the NPC source code is organized into two major components: the NPC class and the client application that uses it. The NPC class, called ExternalNPC, features a public interface consisting of login() and mainloop() methods and a few other methods that control the NPC activities. The NPC class also holds most of the common features shared between different kinds of NPCs such as quest activities and basic chat capabilities. NPC client applications can customize the NPCs and give them distinguishing characteristics beyond what is modeled in the profile web page to any more advanced dialogue capabilities and behavior that are required for that specific NPC.

Given the ExternalNPC class, what does the Tux NPC client look like? Tux has a profile on a web home page. Figure 10 shows a minimal Tux NPC client to instantiate Tux in the CVE virtual world. Tux's profile is downloaded from the specified homepage. The methods handle_msg() and idlefunc() are called to handle the received messages from the server and to set what to do when Tux is idle. Although Tux is not doing anything except standing in a fixed position and logging whatever messages he gets from the server without responding to any of them, this is a building block for an NPC client that may form the basis for interesting NPC dialogue, knowledge, and behavior models.

```
class ClientNPC:ExternalNPC()
method handle_msg(s)
  write("tux received ", s)
end
method idlefunc()
  write("tux is snoozing")
  delay(1000)
end
method run(password)
  srvport := "virtual.cs.uidaho.edu:4500"
  homepage := "www.tux_homepage.com"
  login(password)
  mainloop()
  write("NPC loop finished, good bye")
end
procedure main(arg)
   tux := ClientNPC()
   tux.run(arg[1])   # passing the password
end
```

Figure 10. A very simple and compact NPC client example.

The source code for the NPC client itself is organized into several classes. The following is a list of the major classes along with a summary about each class:

ExternalNPC: Holds all the information about the NPC. It has methods for downloading and parsing the NPC profile, managing the connection with the CVE server, receiving server messages and giving proper responses and taking proper actions when asked to chat or to give quest activities.

AvatarData: Holds the NPC's knowledge of other users' avatars and their quests activities. The NPC uses this to be aware of other users' locations and take proper actions if any. Some NPCs for example might interact with other users if they come closer or go farther, and also avoid them if the NPC is set to move. The NPC uses this also to know what quests each specific user has already completed, quests that they can take and quests that they are currently working on.

Knowledge: A collection of knowledge categories.

KnowledgeCategory: A collection of quests that belongs to the same category such as Linux quest.

Quest: Holds all of the information about a quest, such as readings, prerequisites, and steps. It has methods to read and parse a quest from a webpage or a local file and save it if necessary to a local file. The Quest class also keeps track of the quest activities like the current question the user is answering, quest score, and user's answers for different questions and so on.

VI. Multi-Platform Architecture Issues

The most interesting design consideration for NPCs in educational virtual environments is: how much of the NPC appearance or behavior is coded inside the virtual environment using its normal programming APIs and scripting mechanisms. Flexibility is gained by coding externally via a separate program that communicates with the virtual environment. In this project, it was expected that NPCs would be coded largely using the virtual environment's normal programming APIs. Portability needs overrode this intuition. The more logic that can be coded outside the virtual world, in the PNQ NPC client process, the more of that logic can be shared across virtual worlds.

A. Second Life NPCs

The objective of the PNQ experiment in Second Life was to replicate the local Computer Science environment at the University of Idaho (UI) and provide NPCs and quest activities for that environment.

The PNQ's Second Life implementation borrows ideas from [2, 3], namely the AIML and the use of a user avatar rather than constructing a crude avatar from virtual objects. As is the likely case for many other Second Life NPC projects, Second Life affords low construction costs, but maintenance costs paid to Linden Labs can be substantial. In UI's case, a campus-wide island is owned but controlled elsewhere and incurs unacceptable monthly costs, so for this experiment a separate plot was purchased. Land sufficient to represent the UI's Janssen Engineering Building cost US$60, with a monthly fee of US$18. This meets the low-cost requirement. Students built a virtual UI CS department, with unsatisfactory results. The students constructed a facade with a few floors but were unable to build a useful interior given the 347 graphics primitives that Linden allowed on the plot.

Second Life offers no domain support. An NPC that requires specific domain tools such as a Java compiler, as shown in Figure 7, must use external tools. In the example shown in Figure 7, the student's homework (computer programs) was uploaded by clicking a virtual object and specifying a file. The file was transmitted to an external server where it was evaluated by compiling, running, and comparing it against expected results.

Second Life non-player characters were implemented by a C# program using libsecondlife, which has been renamed to libopenmetaverse (openmetaverse.org), to login to a regular user account. Scripted virtual clothing enabled the NPC to be informed when a user clicked on it; in other respects the PNQ Second Life NPCs interacted with the user primarily via chat commands.

B. *NPCs in the CVE Environment*

The CVE virtual environment is primitive compared with Second Life, but its simplicity allows easy experimentation. CVE avatars can be created from 3D models produced by tools such as 3D Studio Max and exported in Microsoft .x format. Figure 1 shows some example avatar models in the CVE environment a long with quest activities. The dog NPC is loaded from a .x file. The other avatar is hardwired model.

Compared with Second Life, CVE features an integrated collaborative IDE that allows tutorial activities for a range of computing topics. Besides shell commands illustrated in the ls example earlier, these include editing, compilation, execution, debugging, and testing activities for C, C++, Java, and Unicon. Different NPCs teach different subjects and have different personalities embodied in their AIML scripts. In order to offer these CVE-native tutorial activities in Second Life, the NPC agents in Second Life must provide the required interactive demonstration facilities themselves via chat, or accept file submissions of captured sessions conducted outside the virtual world.

VII. Conclusions

This paper introduces PNQ, a notation for portable NPC tutors and quest activities built on top of multi-platform non-player character architecture. PNQ provides a quest system for the purpose of encoding educational content. The architecture provides rudimentary NPCs that offer educational quest activities to users across virtual worlds. Creating a new NPC consists of writing a new web profile for that NPC. Creating new quests follows the same approach, enabling the virtual world to be populated with NPCs and educational content by regular users. Some of the quests that are the most fun no doubt will involve interactions that are specific to a particular virtual environment. However, judging from World of Warcraft's quest mix, a large percentage of quests can be world-independent. For tutorial NPCs they will likely be domain-dependent for a learning domain which may or may not have a virtual world embodiment. Without such direct virtual world embodiment of the material being taught, the success of tutoring is defined mainly by the NPC's ability to convey the material via its scripts.

In a preliminary evaluation of PNQ NPC tutors and their educational quests, part of a short survey at the end of a virtual summer camp held in summer 2009 asked users to evaluate their experience in virtual worlds. A few questions specifically targeted users' experience with NPC tutors. Based on users' experience, walking around in a virtual world looking for NPCs and asking for quests, which often finish with a set of questions to answer, is more enjoyable than answering web-based questions. Users also indicated that NPC quest activities motivate them to look for more quests to do. The users show a great interest in interacting with NPCs to get their quests, preferring this method over getting the same quests through a menu. Users think that NPCs are an important part of any virtual world as long as the NPC can do something, such as offer quests and/or interesting chat. Otherwise NPCs are no more than decorations.

Given the capability to create interesting NPCs that live in multiple worlds, significant challenges remain, such as rewards that provide increased character capabilities that function across multiple worlds. Since most users will interact with NPCs through a single virtual world interface, the primary function of these tutorial NPCs is to enable content to be used in multiple environments.

Acknowledgments

Brandon Nutter provided the C# portion of the NPQ NPC's for Second Life. The research was supported in part by NSF ATE grant number 0405072.

References

[1] G. Booch, J. Rumbaugh, and I. Jacobson. The Unified Modeling Language User Guide, Addison-Wesley, 1998.

[2] A. Fossett, PandoraBot NPC. http://artfossett.blogspot.com/2007/07/pandorabot-npc.html.

[3] D. Friedman, A. Steed, and M. Slater. Spatial Social Behavior in Second Life, Proc. Intelligent Virtual Agents LNAI 4722, pp. 252-363, Paris France, September 2007.

[4] R. Wallace, ed., Artificial Intelligence Markup Language (AIML) Version 1.0.1. Working draft obtained from http://www.alicebot.org/TR/2001/WD-aiml/

[5] S. Redfern, and N. Naughton. Collaborative Virtual Environments to Support Communication and Community in Internet-Based Distance Education. Journal of Information Technology Education, volume 1 No. 3, 2002.

[6] K. Merrick, and M.-L. Maher. Motivated Reinforcement Learning for Non-player Characters in Persistent Computer Game worlds, Proc. ACM International Conference on Advances in Computer Entertainment Technology. 2006

[7] K. Merrick, and M.-L. Maher. Motivated reinforcement learning for adaptive characters in open-ended simulation games. Proc. ACM International Conference on Advances in Computer Entertainment Technology 203, pp. 127-134, 2007.

[8] C. McCollum, C. Barba, and T. Santarelli. Applying a Cognitive Architecture to Control of Virtual Non-Player Characters. Proc. The 2004 Winter Simulation Conference.

[9] M. Mateas, A. Stern. A Behavior Language for Story-Based Believable Agents. IEEE Intelligent Systems, volume 17, issue 4, pp. 39 – 47, July 2002.

[10] H. Tuzun. Quest Atlantis: A Computer Game That Transcends the Computer. http://www.e-mentor.edu.pl/_xml/wydania/5/64.pdf

[11] S. Barab, M. Thomas, T. Dodge, R. Carteaux, H. Tuzun. Making Learning Fun: Quest Atlantis, A Game Without Guns. Educational Technology Research and Development, volume 53, No. 1, pp. 86-107, 2005.

[12] K.-S. Yoo, and W.-H. Lee. An Intelligent Non Player Character Based on BDI Agent. Proc. Fourth International Conference on Networked Computing and Advanced Information Management NCM '08, pp. 214-219. 2008.

[13] Jeffery, C., Mohamed, S., Parlett, R., Pereda, R.: Unicon book "Programming with Unicon". (1999-2003),. Available at http://unicon.org/book/ub.pdf.

[14] Jeffery, C., Jeffery, S.: An IVIB Primer. (February 21, 2006), Unicon Technical Report #6b. http://www.cs.nmsu.edu/~jeffery/unicon/utr/utr6b.pdf.