TRUE STORY: Dynamically Generated, Contextually Linked Quests in Persistent Systems

James Pita
TEAMCORE Research Group
University of Southern California
Los Angeles, CA 90089
307.371.2139

jpita@usc.edu

Brian Magerko
Games for Entertainment and
Learning Lab
417 Communication Arts Bldg
Michigan State University

magerko@msu.edu

Scott Brodie
Microsoft (Carbonated Games)
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

sbrodie@microsoft.com

ABSTRACT

Massively Multiplayer Online Role-Playing Games (MMORPGs) typically use a handful of static conventions for involving players in stories, such as predefined quest or story paths (a quest or story path is one in which the user experiences a sequence of related quests that must be accomplished in a particular order). Beyond the work done in MMORPGs there has been strong research in designing adaptive approaches to interactive fiction/drama that dynamically author content for users of the interactions [10] [18]. The system architecture presented in this paper, TRUE STORY, is designed to address issues concerning dynamically generated quest or story paths in persistent worlds, such as MMORPGs, for users to engage in more enhanced, interactive and personal experiences. TRUE STORY empowers persistent world designers by offering a truly modular approach for dynamically generating and presenting compelling content that results in user experiences worth telling a story about. The current implementation is set in a game model to demonstrate a dynamic quest generation system built to present users with unique and compelling experiences linked by context to past quests and/or experiences. This is achieved by utilizing history and relationships developed through interaction between world objects and actions.

Categories and Subject Descriptors

I.2.1 [**Applications and Expert Systems**]: Games

Keywords

Multiplayer Games, MMORPG, Interactive Narrative, Contextually Linked Goals, Dynamic Quest Generation, Story Generation, Game AI

1. INTRODUCTION

Persistent game worlds, such as World of Warcraft, have inherited a number of conventions established in standalone games that have prevented users from enjoying the unique experience that persistent systems have the potential to offer. The most notable conventions are *leveling*, a reward system that presents quantified status points to players for completing tasks, and *questing*, the act of receiving and completing designer defined goals in return for level rewards. Combined with the desire for a narrative component in persistent game worlds these conventions force designers to offer their users a finite set of scripted quests that all users experience in a similar fashion. Although these conventions allow for an enjoyable experience they prevent the user from truly molding their quest or story paths into a unique and personal one that they can identify with. A potentially better approach then, would be to determine possible quests for a user and then select the most relevant quests to what the user has done in the past [18]. In turn this will help drive a user's unique quest experience over their character's lifeline.

Noting the lack of unique quest experience inherent in persistent game worlds such as World of Warcraft, we have designed an open-ended framework that more naturally utilizes the interactive and persistent properties of MMO systems. This aids in producing an interactive impact similar to dynamically authored interactive systems that most designers of persistent worlds have failed to capture properly in practice.

In an effort to create a framework that is capable of dynamically generating contextually linked and meaningful quests there were some key design principles that had to be addressed. As actions and interactions occur between users and the persistent world they must be tracked and the system must be updated accordingly so the quest generation system can create quests that are unique to the current state of the world. While examining these interactions and actions, the system must be capable of generating quests that are both relevant and meaningful to a user's experience. This must be achieved without any long-term quest paths or end goals as a constraint because the framework is meant for persistent worlds. This does not infer that quest paths cannot end up achieving a meaningful end state, however, they should not be driven by such a goal. It is also important that the information tracked by the system is properly managed so that it does not grow to large to handle, which in turn could slow the system down with meaningless searches for relevant information. Our framework, TRUE STORY, attempts to address these design principles and the related work section below describes how the framework was influenced, formulated, and modified to scope beyond just MMO games.

2. Related Work

The problem of dynamically authoring narrative into an interactive experience is nothing new. Numerous games and academic papers have been dedicated to exploring and discussing the topic [10] [1] [9] [14] [20] [3] [18]. While the seed for the concept of TRUE STORY came directly from analyzing the attempts made by numerous games in the MMORPG genre, the majority of the initial design was informed from research in the Interactive Drama (ID) domain. The recent establishment of

"Interactive Drama as a new medium" has generated a number of academic papers and example implementations related to the combination (or separation) of interactivity and storytelling.

2.1 Conditions and Contextualization

The unifying characteristic between the examined systems was an underlying driving force, or planner, that progresses the storyline or interactions of the system. It became apparent, however, that there had been little work to incorporate these ideas into a dynamic quest generation system and less yet towards dynamic quests in persistent worlds based on player interaction. By utilizing the principles learned from reviewed work we were able to design an open-ended framework capable of creating unique quest experiences for users in persistent systems.

In order to achieve a generalized and open-ended quest generator it was important that we developed a framework that could incorporate any form of quest experience. Quests are limited to those defined for a particular system, but there is not limit on how many types of quests can be defined in a system utilizing our framework. Incorporating the defined quests and the principles of dynamically authoring narrative the framework is then capable of presenting a player with a unique questing experience and quest paths, one where each consecutive experience relates to previous interaction within the system.

Quests that are generated are not driven by a specific story arc or plot focus, which is different from the work done in interactive drama. Some approaches to interactive drama direct users and story content by evaluating previous actions and deciding what plot line appears logical at the moment [10] [1] [18] [16] [19], however, since questing experiences are very different from dramatic experiences users should be capable of directing their own experience if presented with the proper opportunity space. This opportunity space is created by generating quests that are related to past experiences so a user can pursue a path that expands their knowledge of a previous experience.

2.2 Open-Ended Design

Impro [6] in particular influenced our design through its initial defense of designing stories without knowing what the end goal is, thus speaking to how an interesting experience could arise within an interactive experience. A quote from an excerpted Johnstone conversation explains one of the major dilemmas faced with persistent worlds:

> I have to write a story and I'm supposed to map out everything that's going to happen so that my teacher can mark it. She says it will stop me [from] writing the wrong things.

Although this has been a traditional method of story generation the problem that arises is when a user experiences interactive drama from themselves, especially in real time and in a persistent, digital world, a designer cannot possibly predict what actions or choices that user may make. The only way to make such a prediction is by limiting the user's options and thereby limiting their experience. Some attempts have been made at modeling user preferences and generating story content based on such models [17] [9] [18], however, in a questing system this would limit the questing options being presented to a user. Users may want to change their questing preferences on a regular basis.

TALE-SPIN's [11] focus was also beneficial in realizing how the use of attribute variables for characters could be used to add depth to the contextual links we developed in our implementation. Attributes help define how difficult a quest will be for a user to complete and can help narrow the quest space being presented to users. They also add an extra component for MMORPGs by giving users an incentive to increase attributes in order to achieve more complex quests, which is different than the traditional method of receiving more complex quests to increase attributes.

2.3 Layer Relationships

Interestingly enough, an important trait that can help guide dynamic quest generation and is often overlooked involves neither of the aforementioned topics, but a third involving modeling relationships between characters. David Freeman, in his book *Creating Emotion in Games*, states a screen-writing tool used for mapping relationships between characters from film. He describes how he models basic relationships such as "protective" or "jealous" between the main characters in his game. Depending on which traits other characters are aware of then dictates how a character might react.

Behaviors such as these relate to the overall believability of synthetic characters [15]. Adapting this trait layer technique to model relationships that can dynamically adapt based upon previous and current actions is at the heart of what makes TRUE STORY tick. TRUE STORY achieves this by maintaining relationship states between characters as well as mapping an individual's knowledge of how other characters are related to each other. Over time as a user continues to interact with characters in the system they naturally gain knowledge of these characters' nuances, but in turn within the system the characters gain knowledge of the user's behaviors as well as other characters' behaviors.

Freeman also states clearly what separates good and bad stories "Interesting, but also deep". This quote relates to any object or plot point, and it relates to the TRUE STORY framework as well. TRUE STORY's success comes from checking for possibilities that may present a user with an interesting and deep quest based on interactions that have occurred within the system.

3. TRUE STORY Framework

TRUE STORY is a dynamic goal generation architecture for use in persistent systems where multiple users directly interact. The current implementation takes place within a text-based, medieval, persistent world implemented by us that makes use of traditional medieval roles. The following examples occurred at a state when the world consisted of roughly thirty rooms and twelve characters.

The purpose of TRUE STORY then is to systematically assign unique quests, such as a quest to avenge your brother's murderer, to users and objects continually, based upon some specified set of constraints. Constraints are predefined rules or preconditions that are required to be met for a quest to be presented to a user. The constraints used for TRUE STORY include the relationship between two characters, a character's past experiences (quests), performable actions such as attack, proximity to information and attributes such as thievery skill. These help dictate what a character is capable of accomplishing so the users are not assigned impossible tasks and by utilizing a user's memories as a constraint we are able to keep quests within a relevant quest path. As a user completes quests their memories and attributes are updated

according to the task performed, which in turn opens up new quest opportunities. For example, if a user has brought numerous petty criminals to justice then the city guard may send that user on a quest to catch a more serious criminal since the user has proven their merit.

While designers have the ability to define different constraints than those mentioned, constraints generally come in the following forms:

- **Memories**: past quests in which a role has been played (either as protagonist, party member, or passive participant). For instance, in an example taken from our implementation user character A (please note that in our framework both user characters and non-playable characters are identical in function and form so all following examples can be created by a user character interacting with either another user character or non-playable character) has chosen to take the role of a thief and has stolen an object from character B, both characters within the system have now created a different memory of the account. Character A maintains a full memory of how the action took place and what was stolen while B, who in this particular instance did not catch A in the act, creates a memory of the item being stolen, but his memory does not include A as the thief in question. These memories are now stored and will help spark quests in the future. If character C now comes in contact with B, assuming user character C is a reputed lawman and is well known to be so by B, character B will dynamically generate a unique quest for C to find and bring the thief to justice. This quest will not be generated for other characters that are well known to be lawbreakers. It was necessary for A to maintain a memory of the account in order for the system to recognize when C has successfully completed the quest since A is the only character who holds an account of who actually stole the object. Since memories could be created at alarming speeds in a persistent world it is important that an appropriate method for memory management is designed. A persistent world must be capable of eliminating memories that are not useful and tracking those that are.

- **Attributes**: designer defined object properties (ex: affinity, thievery, importance, health, damage, etc). These are used to see if a character has the capability to complete a quest. A character with no fighting history or qualifications will not be asked to take on an evil dragon until they have improved their merits.

- **Actions**: designer specified actions that correspond to the types of quests characters could accept, offer, or earn memories from. Within the context of our system these involve talking with other characters to gain information, stealing, attacking, and examining the world around them. Using the current running example with character A, B, and C taken from the current system, C's actual assigned quest would either be to steal the item back or kill the culprit (character A) depending on the importance of the original item stolen. To recognize that this was a form of justice and not villainy the quest's reward is a positive affinity gain plus whatever B is offering as a reward. In order to accomplish this C would have to examine other characters until he discovered a character with the stolen item, which in our case this character would automatically be labeled as the thief. Given a more robust implementation there may be methods for extracting information to find the real thief if A has since dispersed of the item. Please note in a larger system there may be a different quest type to discover the culprit's identity and turn that character into the law, which would make more sense in this example.

- **Layers**: relationships established between objects and/or their properties based upon context. Influenced by quest roles and/or relationships with other objects [5]. As players interact with others, relationships begin to develop. If character A continually steals from character B and give the items to character D then B will begin to develop a negative relationship with A while D develops a positive one. As an additional clarification, relationships can be established between characters and non-character objects if designers so choose, as they all derive from the same base class (our system implementation does not demonstrate this directly).

- **Proximity**: designer defined area of affect to access memory information from game characters or objects. This is also a constraint that considers the networked environment and the need for search limitations in practice (typically a radius around each object, in our implementation information can only be gained from the objects in the current room). This allows for efficient memory searches in a reasonable time. As a user interacts with objects within the system they will maintain a memory of that interaction for use in future situations, but they cannot draw on information they have not previously learned or are not in proximity of.

In many cases quests cannot be generated until specific types of memories, attributes, or layers have been established. Using the running example we'll assume there is a non-playable character E who was designed as a thief lord who gives thieving quests to characters who are known thieves. Before E will actually generate a quest for A, A must have proven to be a strong thief. The current implementation recognizes A's capability based not only on their thievery attribute but also by A's memories of their successful thieving attempts. If A has performed numerous thieveries than E will search their memories of known important items and create a quest for A to steal one of these items.

Related to the concepts of constraints is the concept of *relevancy*. While constraints serve to limit the amount of information the system has to search, relevancy performs a qualitative check on the data that falls within that constraint space. What determines the "relevancy" of a quest is strictly up to the system designers. In TRUE STORY, a quest is relevant if it has some connection of memory producing capacity that ties into the users current memories and layers. Currently this check is primitive and does not actually score which quests are more relevant, however, within the context of the framework it can be developed to score relevancy based on designer specified constraints. For instance given our implementation, lets assume character A and E have

formed a strong affinity with each other and character E is assassinated by an unknown assailant. As character A wanders the world looking for quests they may come across a character F who is able to generate a quest for A that involves the unknown assailant. Although this information is not conveyed directly, by involving A in a quest that includes the assailant, A has a higher probability of fulfilling their quest to kill E's murderer. This makes what may otherwise be a trivial quest far more relevant than others may be. By scoring relevancy to past actions it helps drive users to meaningful quest paths. Although other quests will be offered to a user at the same time, the most relevant quests can be offered first. The user does not see first hand the relevance of the quest, but is will begin to understand that these quests will help them discover important information.

The approach to generate quests is twofold: first *determine possible quests to generate*, then *select the most relevant (i.e. the most in context) quests to what the user has done in the past*. The hypothesis, then, is that because the relevancy check ensures a user receives quests that are within context of previous quests/actions their character has taken, the resulting chain of events is interesting at worst, but compelling at best. The justification is that as a user progresses their character's lifeline they should naturally follow story lines that interest them by choosing quests with relevance to a particular story arc. This returns to the concept of allowing users to create their own dramatic situations through interaction. Additionally, the fact that the quests offered are unique to a character's situation, along with the character's actions having lasting effects on the persistent world, might suggest that the user will feel a greater sense of agency within the world they are essentially helping to craft (or at least by contrast to the previously scripted quest chains users were accustomed to).

4. User Interaction

TRUE STORY can be thought of as a set of small systems interacting to produce emergent results (unique quests). In practice, quests are presented in a traditional manner, much like that of World of Warcraft. A user chooses to speak with other characters and those characters will in turn offer up information and quests (see Figure 1). The bulk of the underlying calculations of TRUE STORY are purposely made transparent to the user. The difference between TRUE STORY and a system like World of Warcraft is that the quests being presented to each individual character are unique, with a constantly changing world creating an avenue for new quests as the user seeds a character with new quest experiences and interactions. Each time a user completes a quest for another character in our system that quest memory is stored for all concerned parties. If character A is charged with stealing an item from character B by character E and successfully completes the quest then all characters will maintain a unique memory of the action. A will become more proficient in his thievery skill, B will now have a memory of having lost an item dynamically creating a quest to discover where the item has gone, and E will create a memory that A has helped them in the past thus granting A higher affinity with E. Although in the system's current state the information tracked is minimalistic with all changes to the world being similar to the one above in nature, the framework itself can provide for more comprehensive information tracking if the system is designed properly.
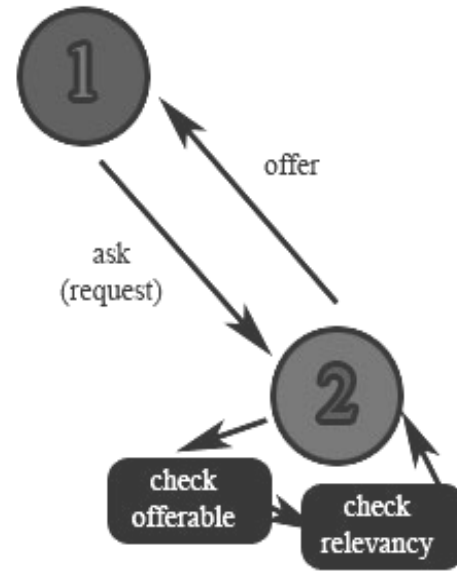

Figure 1: General Quest Request

As Figure 1 illustrates, there are two similar ways a character becomes associated with a generated quest. The most typical interaction is for the character to "ask" (a.k.a. request) for quests from another character. Characters that have been requested for quests will run an "offer" routine that searches through the constraint space involving the two characters. For instance, character A has just asked character E for any quests they may have to offer. The system is now granted full access to A and E's memories, attributes, and layers. We'll assume A and E are as described before and that A and E have formed a trusting relationship because A has already done numerous thefts for E. Recently E has learned about an item of low importance and an item of high importance. Due to the nature of the items, E can now offer two separate quests to A. The first quest might be to obtain the item of low importance by any means possible and the second quest may be to obtain the item of greater importance while also eliminating any evidence of the crime. This may include assassinating any witnesses. The offering character will then present the asking character with this finite set of the most relevant quests that can be accepted. Quests that may not be offered may include stealing an item of no importance of simply gathering information about other items of low importance.

The second method of becoming associated with generated quests is through generic quests that are automatically offered to other characters. For example, in the implementation we developed to demonstrate TRUE STORY, a generic "Steal" quest for an item of arbitrary value can be offered to any player that has a high enough theft attribute regardless of their relationship with the offering character. This allows characters to continually work on their skills by accepting generic quests to help build them up for more interesting assignments.

Beyond the user-based characters, non-playable characters (NPCs) are also treated within the system with the same logic that user characters are. They are introduced into the world as normal users with heavily pre-seeded memories and as the world adjusts

their memories and attributes in turn adjust along with the world. Their adjustment along with the world is critical to the system so that users may carve a name for themselves in their new interactive realm. As character A begins to accomplish larger feats, NPCs in the realm will begin to gain memories of the user's character and may even recognize them on sight if they gain enough fame. This may also help NPCs to create quests against or for a user's character. If a user has become a well-known justice seeker the villains in the interactive realm may want the user eliminated while other justice seekers may actively search for the user for aid. With enough space allocated for memories, this underlying "social network" could produce some impressive quests based upon long-term memories. Character A may find that the NPC they saved at the beginning of their career has grown into a king or perhaps even a thief lord. Character A may then be called on during a much later date in their interactive career to perform a new deed or perhaps be saved from certain death by such an NPC.

Finally, it is worth noting again that the key component of TRUE STORY revolves around the fact that *the system does not use any long-term quest paths as a constraint*. As rationalized in many of the above sections, a long-term goal would add an "end state" that is undesirable inside of a persistent game system. The purpose of TRUE STORY is to subvert the need for an end state entirely by keeping the player consistently engaged by discovering the unique path they are choosing to follow. A user could begin a quest path to discover the murderer of a well-known friend and, once the assailant is found, bring them to justice. All the while the user has completed numerous quests along their journey, which in turn has opened up new quest paths for them to explore. Although no dialogue has been implemented into our system a user is still capable of following meaningful quest paths that have a strong story element to them. Infusing meaningful dialogue into the system could only make it stronger, but for the sake of dynamic quest generation and quest path generation is not necessary. Although such a quest path as defined above is possible within our current implementation, one place our implementation currently falls short is rating how relevant quests are to a specific quest path. Currently the system only decides whether a quest is relevant to the character seeking quests. In the future it will be necessary to explore assigning quests relevancy weights. As relevancy builds the character can pursue a meaningful story that models a real life experience and although the story may come to a conclusion or the user may feel the story they are following is complete, there was no real objective in mind outside of a meaningful interactive experience. However, the "Future Work" section describes some potential uses of this system in conjunction with end-state allowable implementations.

5. Implementation
For the implementation of TRUE STORY a simple text based persistent world is used. The environment consists of a series of rooms interconnected in a two dimensional plane. All other objects in the world reside either in a room or within another object including characters. Rooms are themselves an individual class. Rooms are the boundaries of the world and can contain any number of objects. A room can also be linked to up to four other rooms in the directions north, south, east, and/or west. Within the boundaries of the world, all interactions occur between characters and equipment. Both of these were grouped higher into an all-encompassing class called 'object', which is the base class for all interactions. Categorizing all physical things in the world as objects makes it easy for all interactions to occur through objects and allows the model to be easily extendable. Even if new 'object' types were created such as food they would be subject to the same interactions as characters and equipment with a few new interactions specific for food. Methods have been created to add rooms, items, characters, and features to the world as the programmer sees fit. In order to add something to the world it was necessary that the object be seeded with the correct information (armor has specific defense values, characters have to be given attributes and characteristics, etc.).

The interactions between objects used in TRUE STORY were earlier defined as quests. The quests chosen for this implementation were:

- **Kill**: an interaction where an object in the world is trying to eliminate another object within the world.

- **Steal**: an interaction where an object (in our model a derived character object), is trying to take a different object from a third object, another character.

- **Discover**: occurs when an object is dropped or has been stolen in the world and the initial owner would like to discover where that object is now.

- **Retrieve**: occurs when an object would like to retrieve a separate object in the world, but does not demand that the desired object is necessarily stolen or retrieved in a particular fashion.

The system will have more quest options to choose from as more quest types are created and a user will be given a wider variety of quest paths. A designer can limit the dilemma of the system creating too many quest paths by only allowing the user to choose from a smaller subset of the most relevant quests or by limiting the number of quests that can be given. Although the interactions themselves are important to the system as a whole, the main feature of our implementation is how these interactions are chosen and executed between objects. Although a user may engage in any action at any time of their own accord, each object also contains a set of memories, which in this case are actions that a character has either participated in actively, or been a part of passively. The driving example as been if character A steals from character B, but B does not catch A in the act. These memories are important for establishing the context in which future events happen. At any time a character may perform a steal or kill action, which is then translated into a self-given quest, but doing so will directly affect their interactions with other characters.

Each object also contains a set of attributes. For this implementation the attributes chosen were health, damage, thievery skill and affinity. The affinity variable chosen is essentially a relationship value (negative meaning dislike, positive meaning like) between a user an all other characters in the world as well as a user to the world around (just because character B strongly dislikes character A does not mean A is negatively perceived in the world because A may have performed far more positive deeds than negative ones). Based on the values of these attributes interactions are further conditioned to react accordingly.

It is important that a method for obtaining quests is defined beyond the interactions and attributes of characters. For TRUE STORY each character is capable of asking any other character

within the same room (the proximity limitation) for quest options. After a request has been made, the character asked will scan through their memories and the asking character's memories to find any relevant ties. Each character has two sets of memory, which includes a long-term memory and short-term memory. This was designed for the future implementation of memory decay. Long-term memories are events that will always remain important to the world. An example of a long-term memory would be if character A assassinated the current king of the land. Character A would maintain a long-term memory of accomplishing the act and everyone who heard of the assassination would maintain a long-term memory that someone had assassinated the king. Short-term memories on the other hand are trivial memories that should only matter for the moment. Such a memory might include if character A stole a loaf of bread from character B. Both characters should only maintain the memory for a short period of time.

Once memory has been properly searched, the character asked (B) would have knowledge of any interactions they have had with the requesting character (A) or any interactions they have learned that the requesting character has taken over time. If A makes a request to B, even if B has not personally interacted with A they may have prior knowledge of A ever trying to steal something or trying to kill someone that B is close to then B may be inclined to end all voluntary interaction with A depending on the circumstances. Vice versa, if A tried to steal or murder from a mutual enemy of B then B may be more inclined to help A out.

Now that memory has been addressed, if A has made no prior transgressions against B, the attribute values would be checked. For instance, if B has any steal quests that could be created or that were already created they would only be offered if A has attained a sufficient level in their thievery attribute. B might also offer specific kill quests on a third character if A had enough health and a weapon that was capable of doing sufficient damage. This shapes interaction because B may offer a kill quest to character A, but not to character C based on their current attributes. This also helps in driving game play much like World of Warcraft uses levels to decide whether a player is capable of completing a quest.

It is clearly shown that all interactions done within the system could have a direct result on future interactions. If a character A steals from a character B they may soon become a victim of another steal interaction that character B requests of character C. In order to clearly demonstrate these results the main user is allowed to control all characters within the system. This is to simulate a multi-user environment within a single user environment as no networking code has been implemented. The user can switch between characters using a simple command (switch) and then observe the memories and attributes of that specific character. Certain characters are also seeded with pre-scripted memories as to simulate non-playable characters that might exist within a multi-user environment. These seeded characters are used to drive the initial interactions before actual users have created more memories. As stated previously, these characters are treated like normal characters because they will interact with the world in similar ways through memories and interactions. This is why the user is also allowed to control these characters. A full scale MMORPG would play out the exact same way with the caveat that it would have much more information to handle. Characters would still be able to interact in the exact same ways. A full scale MMORPG may also have implemented more meaningful interactions, which would still come in the form

of a 'quest' class since all interactions are required to be of a quest form, and have better defined social roles. It may also include more attribute values.

5.1 Program
The program itself was implemented in a Linux environment using only C++ and a text-based world. All actions are performed through a text prompt where commands are input. Only the room the user is currently in is displayed to the user with its current exits. The user can also see what other objects exist in the room beside themselves. All the commands a user can issue are documented and can be seen using a help command. After beginning the simulation it is up to the user to create interactions and see how they develop the world around them. The user is also capable of creating specific interactions at any point in time so they can manipulate the world as desired and view the affects.

6. Discussion
TRUE STORY has been a positive step towards the dynamic generation of contextually linked quests. Its implementation demonstrates that designers of persistent worlds can provide unique content to all of its users by utilizing the constraints set forth by our framework, however there remains a lot more to be explored, implemented and expanded upon.

The current design of the TRUE STORY framework incorporates some important design principles that have proved invaluable to the framework. The first principle addressed is that the framework is providing a method to track information in persistent worlds to generate unique and interesting quests. This is accomplished by utilizing history, relationships and other constraints set forth by the implementer of the framework. Although the current implementation only accomplishes this on a rudimentary level it does successfully show that unique quests can be generated solely based on user interactions with the persistent world they are taking part in. The second principle addressed is that the framework does not utilize any long-term goals or conclusions to generate contextually linked quest paths. Rather it utilizes previous experiences to try and drive future experiences in a meaningful way. In this way the user can create an endless chain of experiences and quests that they are able to accomplish over time. The current implementation drives unique experiences based on memories and relationships; however, it does not yet drive quest paths in a meaningful way. Currently it falls on the user to choose which quests they feel are most relevant to their character's past experiences. Regardless, the current implementation does provide for unique questing experiences even if they are not yet tied together through relevancy. Having a quest generation tool like this is the next step to creating unique persistent worlds that utilize a questing system.

Based on the components of the framework (memories, attributes, actions, layers, and proximity) it may be possible to adjust the framework to suit an interactive drama setting as well. One example would be to adjust constraints so that the quests presented were forced to model a story arc like the Hero's Journey [2]. This has not been tested; however, the design is done in such a fashion that if constraints are properly implemented it can accommodate almost any form of quest path generation. By expanding the actions possible for characters and modifying the constraint space we will be able to test how our framework adjusts to more structured settings.

Although the current implementation demonstrates some important aspect to dynamic quest generation there are some important aspects that need to addressed for a larger setting. It is clear that a multi-user environment will not affect the performance of the system since the current implementation can be seen as a multi-user environment. Although only one character is able to act at a time they are all treated by the system in the exact same manner so with multiple users the system would simply be handling more actions at a time, not new actions however. This is already accomplished in MMORPGs and therefore should not pose a problem.

The biggest scalability issue comes from memory management. As more users are infused into the system more memories and quests will be created at a rapid rate. It is important that the designer of the system chooses a device for deciding whether an action or quest should be stored in memory or simply discarded. By choosing only more important memories this memory space can be kept manageable. Different forms of memory decay could also be implemented to discard old memories or memories that are no longer applicable to the world. Although this may shrink the possibility space for dynamically generated quests with high relevancy it would not prevent the system from generating random quests for the sake of continuity. These random quests could help produce attributes and possibly provide important interactions to help generate future quests.

7. Future Work

The first component we would like to address in the future is the lack of an online or networked version of our implementation. Proving that this program could run efficiently on a networked setting would be a strong step. Additionally, in the future it would benefit the usability to add a graphical component to the system. A graphical component is not crucial though.

On the simulation side, the main goal is to expand the current implementation with more actions/quest types, to demonstrate the richness of emergence that can occur through the interactions between a simple set of quests. Also, we need to look into creating a memory decay function for eliminating short-term memories. We've already implemented an importance attribute that will help decide whether events are important enough to move to long-term memory and how long a memory will remain in short-term memory before decay. We need to incorporate this to see how it affects the believability of characters that forget past actions.

8. REFERENCES

[1] Barber, H. 2006. Adaptive Generation of Dilemma-based Interactive Narratives. The Ninth International Conference on the Simulation of Adaptive Behavior.

[2] Campbell, J. September 2003. The Hero's Journey: Joseph Campbell on His Life and Work. New World Library; 1st New Wo edition.

[3] Crawford, C. 2004. Chris Crawford on Interactive Storytelling. New Riders Games.

[4] Fairclough, C. and P. Cunningham. 2004. A Multiplayer O.P.I.A.T.E. Int. J. Intell. Games & Simulation 3(2): 54-61.

[5] Freeman, D. September 15, 2003. Creating Emotion in Games: The Craft and Art of Emotioneering. New Riders Games; 1st edition.

[6] Johnstone, K. June 1999. Impro for Storytellers. A Theatre Arts Book; 1st edition.

[7] Koster, R. November 6, 2004. Theory of Fun for Game Design. Paraglyph; 1st edition.

[8] Lebowitz, M. 1984. Creating Characters in a Story-Telling Universe. Poetics 13; 171-194

[9] Magerko, B. 2007. A Comparative Analysis of Story Representations for Interactive Narrative Systems. Artificial Intelligence and Interactive Digital Entertainment Conference.

[10] Mateas, M. and A. Stern. 2003. Façade: An Experiment in Building a Fully-Realized Interactive Drama. Game Developers Conference.

[11] Meehan, J. 1981. Tale Spin. In Inside Computer Understanding, edited by R. Schank and CK Riesbeck. New Jersey: Lawrence Erlbaum Associates.

[12] Perlin, K. and A. Goldberg. 1996. Improv: A System for Scripting Interactive Actors in Virtual Worlds. The Improv System Technical Report NYU Department of Computer Science.

[13] Propp, V. 1969. Morphology of the Folktale. Trans. Laurence Scott. Ed. Louis A. Wagner. 2nd edition. Univ. of Texas Press.

[14] Riedl, M. and A. Stern. 2006. Believable Agents and Intelligent Story Adaptation for Interactive Storytelling. 3rd International Conference on Technologies for Interactive Digital Storytelling and Entertainment, Darmstadt, DE.

[15] Rizzo, P., M. V. Veloso, M. Miceli, and A. Cesta. 1999. Goal-Based Personalities and Social Behaviors in Believable Agents. Applied Artificial Intelligence, 13:239-271.

[16] Roberts, D. L., A. S. Cantino, C. L. Isbell. 2007. Player Autonomy versus Designer Intent: A Case Study of Interactive Tour Guides. Artificial Intelligence and Interactive Digital Entertainment Conference.

[17] Sharma, M., S. Ontañón, C. Strong, M. Mehta, and A. Ram. 2007. Towards Player Preference Modeling for Drama Management in Interactive Stories. Twentieth International Florida Artificial Intelligence Research Society Conference.

[18] Thue, D., V. Bulitko, M. Spetch, E. Wasylishen. 2007. Interactive Storytelling: A Player Modeling Approach. Artificial Intelligence and Interactive Digital Entertainment Conference.

[19] Weyhrauch, P. 1997. Guiding Interactive Drama. Ph.D. thesis, Tech report CMU-CS-97-109, Carnegie Mellon University.

[20] Young, R. M., M. Riedl, M. Branly, A. Jhala, R. J. Martin and C. J. Saretto. 2004. An Architecture for Integrating Plan-based Behavior Generation with Interactive Game Environments. Journal of Game Development 1(1): 51-70.