create a story and good game experience with both interesting and meaningful choices.

2.1 Dungeon Master

In table-top RPGs, the Dungeon Master (DM) is the "God" of the game [13]. The player's available actions are ultimately ruled by what the DM will allow in a game. Because of the critical role the DM has in the experience, each game is heavily influenced by the DM's abilities. An unsatisfactory DM is incapable of properly negotiating their overall vision of the story with their players' actions. There are two styles of bad DMing, falling at opposite ends of a DM story control spectrum.

On one side of the spectrum is the "railroading" DM. This style of DM has a story in mind and does not allow the players to deviate from it. Players are given no meaningful choices within the game, as their actions have no effect on the progression of the game. When a player offers an action that does not fit the DM's ideal storyline, it is not allowed. A more skilled DM will allow an action, but adapt the story such that the action ultimately does not have a negative effect on the story.

For instance, in "DM of the Rings," a satire comic based on a railroading DM, one of the players attempts to dismount from a warg. The DM has already decided that the player and warg will fall off a cliff, so he disallows the player to dismount, even given the player's dice roll [31]. A more skilled DM would have allowed the player to dismount, but caused another game action to lead to the same DM-desired outcome without the player feeling as if his actions didn't matter at all.

On the opposite end of the continuous spectrum, a DM that does not exert any control over the game's story ends up with chaos, or worse, boredom instead of a good gaming experience. This is due to a lack of interesting choices; the player cannot make informed decisions because there is not enough information available.

With such a large world and so many options, players often feel lost and unsure of where to go. Without any guidance from the DM, the players will get easily side-tracked or bored with the gaming experience. Experienced players may use their previous gaming sessions to guide them through the game, but without any constraints, the game lacks focus and interesting choices, and can become an exercise in frustration [13].

Bad DMs are of particular interest to us because they are analogous to styles of game play within computer role-playing games. We will discuss this in more detail in the next section.

3. COMPUTER ROLE-PLAYING GAMES

Throughout the late '70s and early '80s, table-top role-playing games found their way into the computer domain. The well-defined combat rules from *Dungeons & Dragons* easily translated to the systematic nature of computers. Computers had an advantage over the table-top counterparts in that the computer could quickly and easily store all the rules for the game and effortlessly calculate turns within the combat. Instead of a single encounter taking possibly hours; it could be completed within minutes or even seconds, allowing for more progress in a shorter time.

However, the playable storytelling aspects of table-top RPGs were more difficult to represent, and as such have long been under-represented in computer role-playing games. CRPG gameplay instead focuses on battle, allowing very few player choices outside of those related to combat.

Due to the lack of storytelling support, CRPGs tend towards the two extremes of bad DMing mentioned in the previous section. Many classic CRPGs, such as the *Final Fantasy* series [27], have finely crafted stories which the player is railroaded into playing. While the stories may be grandiose and well-constructed, the player lacks meaningful choices, and is thus merely a character in someone else's pre-arranged story. In fact, major elements of such CRPG stories are often presented in non-interactive "cut-scenes" (linear animations) driving home the fact that they are a part of the game that cannot be played, only consumed in set form.

At the other extreme are games such as *Oblivion* [2], open worlds with a multitude of options for the player, but the overall player experience lacks cohesion and interesting choices for the player. *Oblivion's* designer, Ken Rolston went so far as to point out while designing the game, he kept in mind, "In games, stories suck, so focus on the other elements of narrative: setting and theme" [23].

These shortcomings lead Chris Crawford to point out, "no games have approached what a good DM could do with players around a table" [11]. How can this be addressed? One option would be to completely eschew CRPG conventions and develop a new game genre. We, however, are interested in extending the CRPG genre. We are encouraged by the fact that stand-out games within the genre such as *Planescape:Torment* [7], *Knights of the Old Republic* [6], and *Baldur's Gate II* [4] all have stories that are more playable than games at the polar extremes of the DMing spectrum. They accomplish this through embedding interesting and meaningful choices within their systems of *quests* – the main mechanism currently used in CRPGs to deliver story to the player, one related to quests in literature and table-top RPGs.

4. QUESTS

Jeff Howard describes a quest as "a goal-oriented search for something of value" [14]. The quest for the Holy Grail in Arthurian stories is a familiar example of a legendary quest. This style of quest is often used within table-top RPGs by DMs to give a general direction for player actions.

In a table-top RPG, the quest does not need to be followed, or the players can choose to solve the quest in a multitude of ways. For instance, a quest to overthrow a tyrannical king can be solved by combat (kill the king), subterfuge (convince the king's followers to dethrone him), diplomacy (work with nearby kingdoms to remove the king from power), or any other way the players can imagine – and convince the DM to allow.

Taking the lead from table-top RPGs, computer role-playing games also use quests as a staple in gameplay [14]. Quests are often used to direct the player through the game's story, or to give meaning to the player's actions. On the surface, this is very similar to the quest structure in table-top role-playing games, where the DM often provides at least a main quest that the players use to direct their actions. However, in a CRPG, there is often only one way to fulfill a given quest, with a combat-based solution being the most prevalent.

There are games that are exceptions to this, especially the stand-out games mentioned above such as *Planescape: Torment* and *Morrowind*. Both *Planescape* and *Morrowind*, for example, integrated non-combat solutions to a selection of its quests, allowing players to choose whether to take a traditional combat role or to fulfill one of the other supported solutions to complete the quest. But, given current tools, such multiple-solution quests are burdensome to implement and highly bug-prone. For instance, in *Morrowind*, there are quests where you can choose to kill a specific NPC or find another non-violent solution. Regardless of the choice made by the player, they can later receive a quest which requires them to talk to the NPC that they may or may not have killed. This not only demonstrates the bug-prone nature to this approach, but also shows that the choice the player makes does not truly affect the storyline.

In part for these reasons, far more prevalent is the absence of player choice in quest actions, which becomes even more noticeable in Massively Multiplayer Online Role-Playing Games (MMORPGs). In an MMORPG, the game world is persistent, which means that the game continues to run even when a player is not playing. Up to thousands of players are simultaneously playing the game on a central server; therefore the game must support each of these playing experiences. Due to the persistence of the game world and the large number of simultaneous users, there is rarely a central story arc (an exception to this is *A Tale in the Desert* [12] which resets the game world to the beginning state once a year, thereby having an "end" to the story). Quests are used mainly to give thematic meaning to the supported player actions – typically fighting with enemies scattered throughout the world – and to move players towards areas suitable for their level.

Quests are popular in both single-player and MMORPGs; however the online worlds have thousands of quests to examine. While we are going to briefly focus on quests in the extremely popular MMORPG *World of Warcraft (WoW)* [8], the issues we discuss are found in almost all CRPGs. There are a number of quest taxonomies suggested for *WoW*; we combined and adapted the systems available to make the following taxonomy [15, 21, 29]:

- Kill X number of enemies (where X may be 1, and the enemy unique)
- Kill enemies until X number of a specific item drops (where X may be 1+)
- Collect X number of specific items from the environment (where X may be 1+)
- Deliver an item to a specific NPC.
- Talk to someone specific.
- Escort someone.
- Use a special ability.

As this taxonomy begins to highlight, in *WoW* the majority of the quests and experience points received are related to activities that revolve around killing. To illustrate this further, we examine the second expansion of *World of Warcraft: Wrath of the Lich King* [9], which was released in late November 2008. One of the first regions (called zones) that a player can choose to explore is Howling Fjord. In this zone, there are 133 quests available to an Alliance player. Breaking down these quests into the above taxonomy and then weighting the categories based on the experience received for doing each quest illustrates the continued combat-centric game design.

Of the 133 available quests, 55% of the quest experience comes from straight kill-based tasks of the type "Kill X number of enemies," "Kill a specific named enemy," or "Get X number of items from killing enemies (drops)." Furthermore, 22% of the experience is received from collection-style quests, most of which require collecting items from areas that are infested with enemies.

Additionally, the only ways to gain experience in *World of Warcraft* are to complete quests, kill enemies, or discover new areas. The experience received from discovering new areas only accounts for a minute portion of the overall experience received by a player.

5. TASK-BASED vs. GOAL-BASED QUESTS

Most striking about the style of quest described above is how different these quests are compared to the definition provided by Jeff Howard. In his description, the key concept is that of a "goal-oriented search." In contrast, most contemporary CRPG quests present the player with a checklist of actions to take to complete the quest. The goal is to complete the list; there is no player choice involved other than whether they choose to complete the quest or not.

In such quests, which we refer to as task-based quests, this list of tasks provides the player with the collection of non-optional actions that must each be completed in order to complete the quest. In contrast, a goal-based quest presents an end point, or goal, for the player to achieve. The player chooses, within the constraints of the game world and mechanics, how to reach the given goal.

An example of a task-based quest that could be found within many RPGs is the quest to save a farm from wolves. A farmer will give a player a task of killing the wolves to save the farm. The player must kill the wolves in order to receive the reward for completing the quest, regardless of class or role-playing preferences. In contrast, a goal-based quest would simply explain to the player that there are wolves killing the local livestock. The player would then be allowed options for completing the quest. They could still kill the wolves if they choose, but other options would be available such as creating better fencing or helping restore the local deer population so the wolves no longer have to hunt livestock. In this way, the player is able to make an interesting choice (no choice is clearly better than the other) based on class, race or personal preference. With the inclusion of these choices having a meaningful effect on the game (e.g. different quest rewards, game world evolution, or even affecting future quest and solution availability), quests become playable.

The existence of player choice in how quests are completed is the key difference between goal-based and task-based quests. Early adventure games such as *The Secret of Monkey Island* [17] and *King's Quest* [26] games often presented the puzzles within the game as a quest for the player to fulfill: "Become a pirate," "Save the mayor," "Save the land". While on the surface these seem like goal-based quests, there was in fact no player choice involved in quest completion. This could easily lead to frustration where the player would be searching for the solution the designers had chosen for each quest, even though there were other options that made sense to the player but were not supported.

For example, in *Monkey Island* there exists the simple goal of getting past some deadly piranha poodles. At this point in the game, the player has become a swordmaster, but they are not allowed to fight the poodles, they must go find the exact item required to pass the dogs. The player has a chunk of meat, but this alone is not exactly what is required. Using the meat with grog (potent alcohol) does not work, but instead the player must eventually realize that they need to use the meat with a small flower they (hopefully) found while walking through the woods to drug the meat. Until the player stumbles upon this solution, they cannot progress further in the game.

This frustration is at heart caused by the fact that these quests were presented as goal-based quests, but were actually task-based quests with opaque specifications. A true goal-based quest allows for interesting player choice, where there are multiple ways to fulfill the quest, and one solution is not obviously better than the others.

5.1 Agency

Giving the player interesting choices with which to solve their quests relates strongly to the theory of agency within game design literature. In particular, Wardrip-Fruin, et. al. describe agency as “a phenomenon involving both player and game, one that occurs when the actions players desire are among those they can take (and vice versa) as supported by an underlying computational model.” [30]

Achieving agency, in this account, does not require enabling players to "do anything." In fact, it is in many ways the opposite. It requires crafting the dramatic probabilities of the fictional world and developing player understanding of the underlying computational system so that the two are in concert with one another.

We see a version of this in traditional tabletop role-playing games. A set of dramatic probabilities are established for characters that connect to the underlying game system. For example, the expectations for classes like Wizards and Clerics – and races like Dwarves and Elves – are in part derived from literary sources (e.g., Tolkien) and in part formed by player knowledge of the game system (e.g., Clerics have a wide range of healing spells, Elves are generally better at agility-heavy tasks than Dwarves). These then come into action during gameplay, as players use their characters to attempt actions that are appropriate both to the dramatic probabilities of the situation and the specifics of their character – one doesn't role-play Aragorn and Gandalf the same way, even if they find themselves in the same situation. The DM facilitates play that is appropriate to the different dramatic probabilities of each character, using the rule system, enabling an experience of agency.

This connects directly to our critique of quests in current computer role-playing games. Because they are fixed lists of tasks, they assume that the dramatic probabilities are the same for a fighting character, a healing character, and a stealthy character. They assume the probabilities are the same for a character no matter what quests they have completed in the past, what special abilities they have, and so on. This, in turn, requires that combat be the center of almost every quest – because this is one of the few things that every character can do, even if it doesn't always make dramatic sense if we take characters seriously.

Our research is motivated in part by this mismatch. We want to enable quest authors to create quests, with a tractable amount of effort, that provide dramatically appropriate paths for different sorts of characters – both in their fictional and game system representations. Beyond our near-term research, we are also interested in directions that will make it possible to dynamically alter and generate portions of quest structures to support dramatic probability for a wide range of characters with no more authoring effort than is required with today's relatively clumsy quest authoring tools.

6. PROPOSED EVOLUTION WITHIN CRPG QUESTS

We are not the first to note the need for a system that better models the collaborative nature of the DM and players. In *Hamlet on the Holodeck*, Murray mentions that in the future, the challenge in games will be to “capture a wider range of human behavior than treasure hunting and troll slaughtering” [20]. Many games are limited to these actions because more complex actions require a more complex gaming system. Adding to this in *Character Development and Storytelling for Games*, Sheldon mentions that video games “need to be able to adjust to the improvisation of players” [25].

More specifically, Susana Tosca states, “In computer games, we need to prepare the quest events in a much more fixed way [than table-top RPGs.] The result is less lively and immersive […] To my mind, the success of pen&paper games is precisely in the common creation of a story […] that depends on all participants being human and changing the script constantly” [28].

Spurred by this, we are proposing a system called The Grail Framework that supports playable quests; quests with interesting and meaningful choices. This framework allows authorial control over the world via designer goals similar to a Dungeon Master presiding over a table-top game. We create player choice by supporting an RPG system that allows for goal-based quests with multiple solutions. The quests and solutions available to the player are shaped by the player’s own history such as where the player has traveled, who they have talked to, and how they have solved previous quests.

To create enough content for the player to be able to make these choices requires much more work from the designer. Not only is there the issue of content generation, but it is also difficult for the designer to keep track of the inter-dependencies between quest lines and quest solutions. Because of this, it is necessary for the Grail Framework to also contain author support tools such that designers are realistically able to create the content required for such a system.

7. QUESTBROWSER

To truly achieve playable quests, the designer will need to be able to create a large number of possible solutions for each quest. We have created the QuestBrowser brainstorming tool as part of the Grail Framework to help designers with this challenge as well as help alleviate the difficulties in thinking up multiple interesting solutions for each quest.

There are other quest generation and design tool systems currently being developed. ScriptEase [18] is a designer tool created to work with the *NeverWinter Nights* [5] Aurora toolset [3]. ScriptEase follows a pattern-based approach to authoring, with many of the common designing tasks available as a pre-scripted selectable component in the tool. Many of the standard quests regularly found in CRPGs are available as patterns in the quest library. These patterns are extensible so that a designer is not restricted to just the quests available in the library. Once created, these quests are available within a *NeverWinter Nights* module, but unlike the Grail Framework, the quests are statically placed, do not change based on the player's actions, and are unlikely to be goal-based (but are instead task-based).

Charbitat is another example of a world that uses a quest generation system. Specifically, it is a procedurally generated world which includes a lock-and-key style quest creation system that works in conjunction with the terrain generation. *Charbitat* implements quests based on spatial progression through the world [1]. As the level is generated, a quest can be created on a new tile which uses the world state as context for the goal of the quest. Unlike the Grail Framework, quests within *Charbitat* are generated without author input, but are based instead on the tiles the system is generating.

8. SYSTEM DETAILS

QuestBrowser is a GUI interface that leverages the common-sense database ConceptNet3 [16] to find links between quest-related ideas. One of the benefits of using ConceptNet is that the database is able to easily supply relationships between objects that are unusual or surprising in some way as it does not have a personal bias or preference for a specific outcome.

ConceptNet3 structures its data by storing concepts as nodes which are connected to other nodes based on their relationships. There are currently twenty discrete relationship types, such as *PartOf*, *ConceptuallyRelatedTo*, *UsedFor*, *CapableOf*, and *LocationOf.*

ConceptNet3 is mined from a crowdsourced corpus of natural language sentences about the world, and consequently contains a fair amount of noise. Through experimentation we determined that it is necessary to limit which link types we use, eliminating the most abstract link types, such as *ConceptuallyRelatedTo*, in order to not swamp generally useful quest suggestions in a sea of bizarre results.

We use ConceptNet3 in particular because each concept / relation / concept pairing is scored by other users. This score is stored in the database and available for use in filtering responses returned by the system. We also limit the length of the paths returned by ConceptNet to 5 nodes. This was chosen after experimental runs showed a higher number of noisy results when paths became too long. Requests to the database also began to noticeably slow down when paths became 6 nodes or longer.

In our interviews with Sony Online Entertainment quest designers, it became clear that the user interface was an overlooked portion of quest design tools. While we are not primarily working on the human-computer interaction elements of the quest-building problem, it is still important that the designers be able to successfully use what we create. We increase the chances by presenting a basic graphical user interface (GUI), rather than asking quest builders to perform actions such as hand-specifying complex database queries or entering the same variable name in multiple text files.

To interact with the system, a designer supplies the GUI tool with a quest concept, and, optionally, a possible quest objective. The system returns a list of linked paths through the knowledge space that connect the starting concept and quest objective specified. This gives the designer possible ways reason about the relationships between these objects.

If a designer does not specify a goal or objective, QuestBrowser will show all nodes directly connected to the concept node chosen. In this way, if the designer has a concept in mind, the QuestBrowser tool can help the author think of quest goals.

As an example, the designer may choose to create a quest that is located in a church, forest, cave, alley, or city. The designer can be as specific or general as they would like. When the designer enters one of these locations into QuestBrowser, a list of concepts associated with the location specified is created. If the designer decides they would like to create a quest for a cave, QuestBrowser will generate a list of concepts including *underground*, *dark*, and *mine* among others.

Given these concepts, the designer may decide that they would like to make a quest involving a darkened underground room with the objective of finding light so that a player may cross the room. Certain ideas may initially come to mind, such as finding a torch or building a fire. However, the brainstorming tool will return a table displaying all paths with 5 or fewer concepts between *underground* and *light,* showing ideas that also make sense but may not come to a designer's mind.

The following are some examples returned by QuestBrowser for the situation above:

Underground ←(*At Location*)– **Rock** –(*Part Of*)→ **Moon** –(*Capable Of*)→ **Light**

Underground ←(*At Location*)– **Fungus** –(*Capable Of*)→ **Bioluminescence** –(*Capable Of*)→ **Light**

Underground ←(*At Location*)– **Coal** –(*Used For*)→ **Burn** –(*Capable Of*)→ **Light**

Using the UI, the designer is able to manipulate the quest paths found by requiring or excluding nodes, constraining the types of links between the nodes and requiring results to be above a certain reliability score. This allows the author to narrow down the field of results in a way that suits their needs.

In unassisted quest authoring, the designer has to imagine properties of a location that the player might change (such as dark), and imagine ways to change them. Using a brainstorming tool such as QuestBrowser, designers discover interesting and unusual quest possibilities, lowering thes creativity burden of creating interesting and meaningful choices for the player when presented with a quest.

8.1 Nodes

Each concept is stored as a node within ConceptNet. Nodes are represented as "Subjects" within our GUI. Because of the nature of the corpus of the ConceptNet data, we felt it was important to give the designer the ability to restrict the results in various ways. For some designers, results returned might be funny, while for others, it would be inappropriate. An example of this is the following solution returned for the Start State: Underground, and the Goal State: Light.

Underground ←(*At Location*)– **Oil** –(*Used For*)→ **Eating** –(*Creates*)→ **Gas** –(*Used For*)→ **Light**

For these cases, the designer is able to exclude the Eating node from any further responses if they would like to avoid these types of solutions. The designer is also able to require nodes. For instance, if the designer wished to see more solutions using the idea of Oil, they could require it in further solutions, and only paths including the node Oil would be shown.

8.2 RelationTypes

ConceptNet links nodes based on the type of relation between them. These are unidirectional links, so while "oil" has the link "At Location" to the node "underground", there is not necessarily a link from underground to oil. For this reason, it is important to allow the tool to traverse links in both directions to find the relations between nodes. Otherwise, the results returned are unnecessarily limited by the system.

To increase the authorial power of the tool, we have given the designer control over which relation types are used in the results. In this way, the designer is able to narrow down the results returned. For instance, some designers may wish to know the temporal relation of subjects, and will want the results to include relation types such as Has Subevent, Has Prerequisite, etc., while another designer may wish to exclude those types of relationships from their results.

These constraints are not always necessary, and often the default relationship links return usable quests. For instance, choosing church as the original concept and heal as the quest objective generates the following non-obvious idea as one of its solutions:

Church –(*LocationOf*)→ **Music** –(*CapableOf*)→ **Heal**

8.3 Example

To illustrate how a designer might use the QuestBrowser tool, we have created the following example. In our fantasy setting, there is a wooded town named "Forest Song." At this point, the designer wishes to create a set of quests involving the town. To begin this process, the author would need to identify a problem within the town that needs solving. Entering Forest and Song into the QuestBrowser system, the designer gets a set of results including the following:

Forest –*(LocationOf)*→ **Bird**–*(CapableOf)*→ **Music** –*(PartOf)*→ **Song**

This result helps spark an idea for the designer, who then decides that the town of Forest Song was named for the sound of the birds in the forest, but the birds have now ceased singing. This is used as the initial background issue within Forest Song for the player to solve. Therefore, the quest within Forest Song is to find out why the birds have quit singing, and to find a way to make the birds sing again.

The next step for the designer is to identify possible solutions for the player. The first step is to find a solution to "Why are the birds not singing?" For this step, the designer may choose to only have one reason. Using the QuestBrowser system, the designer asks for the list of results for the input "Bird". Looking through the results, the designer notices:

Bird –*(Has Property)*→ **Fearing Cat**

Following up on this idea, the designer decides that there are wild cats in the area that are scaring the birds, which in turn is causing them to no longer sing. For the sake of simplicity in this example, this is the sole reason behind the birds not singing.

Now that the designer has decided why the birds are no longer singing, a goal for the player to achieve has also been decided. The goal of the quest is for the player to get the birds to sing again by dealing with the wild cats.

The next step is for the designer to find solutions for this quest. While a typical role-playing game would have the player merely kill the wild cats, this designer would like to support non-violent solutions. Once again, returning to QuestBrowser, the designer starts with the concept "cat" to see what is returned. Some of the responses are:

Cat –*(Desires)*→ **Milk**

Cat –*(Desires)*→ **Not Be Wet**

Cat –*(Desires)*→ **Catnip**

Cat –*(HasProperty)*→ **Curious**

Cat –*(LocationOf)*→ **Out Of Bag**

Given these responses, the designer makes a list of possible solutions. Using the results above, in our example, the designer creates four solutions which are based on the class of the player.

The first solution is to allow the player to simply kill the wild cats to rebalance the population. This is an appropriate solution for a fighting style of character that prefers combat. As stated above, this is the type of quest that would be most common in current RPGs. However, the designer is able to create other quests which alter the dramatic probabilities for other play styles.

For characters that are stealth-based, a second solution uses the idea that cats do not like to be wet. The player is able to stealth past the cats to the area that the birds are nesting. Once there, the player can create moats around the nesting trees such that the cats will not cross. This allows the birds to return to their nesting grounds and the forest to be filled with bird song once again.

Spell-casting characters are able to solve the quest using the knowledge that cats desire catnip. They can create a poultice using catnip which allows the villagers to befriend the cats. Once befriended, the town villagers can take care of their new feline friends, which will no longer hunt the birds as they will be properly fed.

Finally, the last solution is a bit tongue-in-cheek and uses the concept of "letting the cat out of the bag." The player can use milk and cats' natural curiosity to tempt the cats into bags. Once bagged, the cats can be moved to a new location with plenty of mice for the cats to eat. This solution is one that any style of player can complete.

In this example, it is important to note that the designer is still a key part of the design process. QuestBrowser is a tool to help the designer think of possibilities that use common sense, but the designer is necessary in putting it all together. Additionally, QuestBrowser is not domain specific, so it is up to the designer to decide if or when quests should be class or player specific.

9. DISCUSSION

Quests in computer role-playing games (CRPGs) are generally not playable in their current form, given that some form of choice is necessary for gameplay. Even more is required for good gameplay. But there are no interesting nor meaningful choices for a player to make in most CRPG quests. Instead they are handed a to-do list to check off as they follow the instructions precisely.

Rolston quips, "I hate getting quests. I hate the toil of completing quests. I hate the formal and predictable resolution of quests. At best, I feel a Puritan sense of rectitude for laboring dutifully, of doing my duty to uncover the fog of narrative war" [23].

If quests are to become a goal-based search for something of meaning as Howard suggests, we must move away from the "formal and predictable resolutions," and the "toil" associated with completing a quest.

The quests that Rolston hates are in fact those implemented in the task-based system so prevalent in today's CRPGs. There are no interesting or meaningful choices, so instead they require "toil" or are considered a grind. In many recent MMORPGs, quests are used merely as a thin cover for the very grinding behavior that players complained about in earlier MMORPGs.

Our proposed system, the Grail Framework, supports goal-based quests, granting multiple choices to the player that will impact future quests. Quests and solutions are made available based on player history. This more closely follows the model adopted by many good DMs within table-top RPGs.

Making the move from task-based quests to a goal-based adaptive quest style also allows for interesting and meaningful player choices – their choices now influence the future game, and different solutions are supported for each quest.

In order for the designer to realistically be able to create the number and variety of solutions required for playable quests, we created QuestBrowser to aid the designer in quest creation. QuestBrowser is built upon a common-sense knowledgebase created by thousands of people, therefore the knowledgebase does not have the biases or preferences that just one person would.

Consequently, using this system allows the designer to see past their own quest preferences and routines, finding new and innovative solutions for each quest. This keeps the player engaged as they are offered interesting choices at each quest, instead of the same single solution that is common in many modern-day CRPGs.

Further, the overall Grail Framework aims to create playable variety in how quests are introduced and situated, while simultaneously taking care of much of the "grunt work" that today's quest designers must shoulder (which particularly makes quests with multiple solutions burdensome to implement).

10. FUTURE WORK AND CONCLUSION

As we stated in the beginning of this paper, the Grail Framework is an ambitious project, and there is much work left to be done. Our next focus will be on creating an initial implementation of the back-end system to support quests with multiple solutions.

To test the feasibility of the system, we will be prototyping throughout the development process by creating an online game available to many players. It is necessary for the system to be play-tested by many players so that we can investigate the depth which the game supports different playstyles. It will also be important to receive player feedback on whether the players feel that they have real choices within the game.

Tosca states that "…the success of pen&paper games is precisely in the common creation of a story. […] Paradoxically, this is what cannot be reproduced by computer games" [28]. We feel that while the success of pen&paper games might not be able to be reproduced exactly without the creation of an AI-complete DM, our system will move CRPGs one step closer towards the depth of story table-top games have enjoyed, while retaining the strengths that have made the CRPG a successful game genre.

11. REFERENCES

[1] Ashmore, C. and Nitsche, M. 2007. The Quest in a Generated World. In Situated Play: International Conference of the Digital Games Research Association. (Tokyo 2007).

[2] Bethesda Softworks. 2006 Elder Scrolls IV: Oblivion.

[3] BioWare. 2002 Aurora Engine Toolset.

[4] BioWare. 2000 Baldur's Gate II: Shadows of Amn.

[5] BioWare. 2002 NeverWinter Nights.

[6] BioWare. 2003 Star Wars: Knights of the Old Republic.

[7] Black Isle Studios. 1999 Planescape: Torment.

[8] Blizzard Entertainment. 2004 World of Warcraft.

[9] Blizzard Entertainment. 2008 World of Warcraft: Wrath of the Lich King.

[10] Costikyan, G. 2007 Games, Storytelling, and Breaking the String. In Second Person: Role-Playing and Story in Games and Playable Media, P. Harrigan and N. Wardrip-Fruin, Eds. Cambridge: The MIT Press.

[11] Crawford C. 2003 Chris Crawford on Game Design. Indianapolis: New Riders Press.

[12] eGenesis. 2003 A Tale in the Desert.

[13] Fine, G.A. 1983 Shared Fantasy: Role-Playing Games as Social Worlds. Chicago: The University of Chicago Press.

[14] Howard, J. 2008 Quests: Design, Theory, and History in Games and Narratives. Wellesley: A K Peters, Ltd.

[15] [Karlsen, F. 2008 Quests in Context: A Comparative Analysis of Discworld and World of Warcraft. Game Studies Journal, vol. 8, no. 1 (September 2008).

[16] Liu, H. and Singh, P. 2004 ConceptNet: A Practical Commonsense Reasoning Toolkit. BT Technology Journal, vol. 22, no. 4, pp. 211-226 (October 2004).

[17] Lucasfilm Games. 1990 The Secret of Monkey Island.

[18] McNaughton, M. and et al. 2004 ScriptEase: Generative Design Patterns for Computer Role-Playing Games. In International Conference on Automated Software Engineering (Linz 2004).

[19] Mona, E. 2007 From the Basement to the Basic Set: The Early Years of Dungeons & Dragons. In Second Person: Role-Playing and Story in Games and Playable Media, P.

Harrigan and N. Wardrip-Fruin, Eds. Cambridge: The MIT Press.

[20] Murray, J.H. 1997 Hamlet on the Holodeck: The Future of Narrative in Cyberspace. New York: The Free Press.

[21] Oh, G. and Kim, J. Y. 2005 Effective Quest Design in MMORPG Environment. In Game Developers Conference. (San Francisco 2005).

[22] Rollings, A. and Morris, D. 2003 Game Architecture and Design. Berkeley, CA, USA: New Riders Publishing.

[23] Rolston, K. 2009 My Story Never Ends. In Third Person: Authoring and Exploring Vast Narratives, P Harrigan and N. Wardrip-Fruin, Eds. Cambridge: The MIT Press.

[24] Salen, K. and Zimmerman, E. 2005 Game Design and Meaningful Play. In Handbook of Computer Game Studies, J. Raessens and J. Goldstein, Eds. Cambridge, MA: MIT Press.

[25] Sheldon, L. 2004 Character Development and Storytelling for Games. Boston: Thomson Course Technology PTR.

[26] Sierra Entertainment. 1984-1998 King's Quest.

[27] Square Enix. 1987-current Final Fantasy.

[28] Tosca, S. 2003 The Quest Problem in Computer Games. In Technologies for Interactive Digital Storytelling and Entertainment (TIDSE). (Darmstadt 2003).

[29] Walker, J. 2007 A Network of Quests in World of Warcraft. In Second Person: Role-Playing and Story in Games and Playable Media, P. Harrigan and N. Wardrip-Fruin, Eds. Cambridge: The MIT Press.

[30] Wardrip-Fruin, N. Mateas, M., Dow, and Sali, S. 2009 Agency Reconsidered. In Breaking New Ground: Innovation in Games, Play, Practice and Theory: International Conference of the Digital Games Research Association. (London 2009).

[31] Young, S. 2007 The DM of the Rings. [Online]. http://www.shamusyoung.com/twentysidedtale/?p=1017