The Quest in a Generated World

Calvin Ashmore
School of Literature, Communication, and Culture
Georgia Institute of Technology
Atlanta, GA 30332-0165
ashmore@gmail.com

Michael Nitsche
School of Literature, Communication, and Culture
Georgia Institute of Technology
Atlanta, GA 30332-0165
michael.nitsche@lcc.gatech.edu

ABSTRACT

As procedural content becomes a more appealing option for game development, procedurally determined context is necessary to structure and make sense of this content. We find that a useful means to structure content in 3D games is the quest. The task of generating necessary context then becomes one of quest generation. This paper describes how we implemented a basic quest generator based on key and lock puzzles into a procedural game world. It uses notion of quest as spatial progression and discusses the design of the game world and how our quest generator connects to it. Its findings are twofold: on the technical level we managed to implement a highly flexible content and context generator into an existing game engine; one the content level we can trace signs for higher player interest in quest-enhanced procedural game worlds in comparison to unstructured spaces.

Author Keywords
Procedural generation, spatial generation, quests, virtual space, video game

PROCEDURAL SPACE AND THE QUEST

The field of procedural content has found a substantial amount of attention recently. This is largely due to Will Wright's brainchild *Spore*. *Spore*'s use of generated content is augmented by the game's open playing style. Procedural content not only makes development cheaper but also offers new design issues and challenges. The player of a procedural world can actively shape its development and customize the result. The game world itself can become a reflection of the player and her intentions. But to do that there must be some method of contextualizing generated content within a game environment. The problem is less one of content generation than one of context building. This paper addresses the *Charbitat* project which uses spatial metaphor to tackle the problem of quest generation within a procedural space.

Procedurally generated space has been used in games since the earliest days of electronic games. The reason was twofold: 1) Early games did not have the necessary memory space to hold the graphical details of designed levels; and 2) generated levels would ensure a different experience on each play. In that way, even technologically limited games could offer seemingly endless game universes and a high replay value - as seen e.g. in *Elite*. The difference in level of detail or other visual cues between designed levels and generated ones was not significant because the graphical level of detail was limited. For example, *Rogue* used basic ASCii symbols but the game world still stands for an innovative and engaging approach. As storage space has become ample through better data media and faster processing of the data, hand-crafted level design has become the norm. But with ever more powerful systems, production costs have soared as game worlds demand ever more content to provide their players. That is one reason why procedurally generated content has seen a revival. It seems to offer an exit strategy out of the spiraling increase of content production costs.

Creating procedural content is not necessarily difficult, but creating *meaningful* content is substantially more challenging. It is relatively easy to create random levels but far more complicated to infuse these levels with some meaningful structures. Yet, without context and goals, the generated behaviors, graphics, and game spaces run the danger of becoming insubstantial and tedious. Even if it is rife with interactivity and content, without context, the space is merely an empty shell instead of a game. As in the game *Myth*, such an environment is a discursive machine [2], having the potential for gameplay but lacking purpose. Countering this lack, we argue that generated spaces have the potential to intrigue and inspire the player and not merely be an open expanse or infinite dungeon. We argue that the necessary context can be provided by procedurally generated quests that assign significance to the game locations.

This paper will suggest a way to generate quests by situating them inside a player-driven procedurally generated 3D world. We implemented a prototype of this system in the experimental game *Charbitat*. In *Charbitat*, players generate an infinite 3D world as they explore and interact with it. On top of this content creation, our system generates quests situated within the space to meaningfully direct the player's experience by creating goals and challenges. We thereby introduce a second tier to procedural generation: that of context on top of content.

Situated Play, Proceedings of DiGRA 2007 Conference

Procedural Game Spaces

In the history of games that use procedural space generation, there have been largely two approaches. The first concentrates on open spaces or terrain. The generation process works by creating maps or heightfields, usually determined by some fractal algorithm [5]. An early example of terrain generation is in Lucasfilm's 1995 *Rescue on Fractalus!*, and this sort of generative method has been extended to form the basis for a lot of default terrain in many games. Terrain generation eventually has been used to form the basis for whole planets in *Spore*, and in non-game programs such as *TerraGen* and *Mojoworld*.

A second approach aims at the generation of dungeons and interior spaces. Unlike open terrain, these spaces have explicit constraints and use them to partition the environment. The assembly of these constraints is used to generate new levels. This method originates in *Rogue* and *Nethack* and has since been extended into more recent titles like *Diablo*, and forms the basis for the generated dungeons in the Nippon Ichi titles such as *Disgaea* and *Phantom Brave*.

Other spatial generation projects use different constraints for generation of space. One example is the CityEngine project [11], which simulates cities by using water, elevation, road patterns, and population density. While the application of this project is not appropriate for a game world, it describes a method for generating spaces based on logical parameters. Also relevant is the Instant Architecture project, whose aim is to create a grammar for architectural form [15].

The space-generation method in *Charbitat* uses a synthesis of all the above. It uses a tile-based system in which every single tile is treated as a terrain and generated through a heightmap. Every tile is populated with virtual flora and fauna, which are positioned following certain spatial conditional rules and filters [10]. As a result, the generated space is not only highly versatile but also structured around certain conditions. This allows for spatially situated quests to be implemented into a unique and infinite procedurally generated landscape. We see the value of such a generator, for example, in MMO worlds or a new breed of RPG and adventure games.

Defining the Quest

As a device, the quest transcends game genres, and can be thought of as a means for structuring play within a virtual environment [7]. Quests consists of several recurrent properties, such as the objective, the task, and success or failure conditions, several of which are explored in Aarseth [1]. Notably, quests vary in their presentation and execution, so developing a comprehensive definition is difficult. Here, we shall examine quests as they are defined in several genres and identify the kind most applicable to the procedurally generated setting.

Quests have been widely applied in numerous games [14] in many different ways. Some titles require quests to be completed in a linear order others allow many concurrent optional quests to take place at once. The simulation described in this paper uses linear quests, but in the future, it could be extended to accommodate others. When searching for useful kinds of quests to adapt in a procedurally generated environment, there are a variety of possible options to choose from.

Quests dominate Role-Playing-Games and many Massive-Multiplayer-Online-Games, where they often have explicit starting and completion conditions. In this type of quest, the player is given a specific task that they may fulfill in the environment. In this case, quests are gradually revealed and form a meta-structure in themselves, as seen in the leveling-up quest structure of *World of Warcraft* that carefully orchestrates spatial progression through questing that references a character's level and ability. These quests could be read as quests of personal growth as well as spatial expansion. In our case we restrict it to the virtual hero and her development that can be quantitatively measured and regulated.

A quest-situation can also be read into mission-based games, such as *Counter Strike,* in which players' goals are encoded in terms of explicitly defined objectives in the space such as "Bring this virtual item to that location and activate it – then defend it against opponents". These goals are known to all parties before the game starts and are often met with opposition from other teams attempting to accomplish their own goals within the space. For instance, players must secure strategic areas, protect other players or non-player-characters, or prevent the other team from reaching their own objectives. Winning the game and delivering on a quest within a game session depends on defeating other players as much as it does on spatial progression.

A third type is the type of quest that is motivated by exploration of a space. In this type of quest, players explore a space, but are restricted by some obstacles (locks) that have to be overcome with the help of some items (keys). Both are presented in the game space itself. We find these kinds of quests in the *Zelda* and *Metroid* game series. Obstacles may not be passed until the player obtains some token (such as an item or skill), yet the quest depends less on the growth of the character and more on the items collected. There are several factors at work here: first the player must recognize the obstacle and understand that they need to find something to get around it, then the player must actually obtain the token, and finally the player has to pass the obstacle. This type of activity is the key and lock puzzle, and shall be explored in detail momentarily.

Across the various game outlined above, quests are understood as dramatized searches that can follow certain themes and patterns. Such patterns have been outlined e.g. by Propp [12] and Campbell [3] – both have been applied to game studies [13]. Others have interpreted quests a religious or personal/ psychological journeys (e.g. pointing at the Jungian origins of Campbell's approach). *Charbitat*,

in fact, plays with the notion of a quest as a psychological journey in the narrative setting of an internalized dreamscape. The hero in *Charbitat* has been poisoned and remains trapped in her own dreamworld. The mission is to find certain locations within this dreamworld to heal herself. But although the psychological dimension of quests is important for their understanding and context, we limit the discussion here to the actual performance of the questing. For the purposes of this paper, the quest is realized in a form of spatial progression [6].

We concur with Aarseth [1]: "If we examine a number of adventure games, they all seem quite similar in terms of form: the player-avatar must move through a landscape in order to fulfill a goal while mastering a series of challenges. This phenomenon is called a quest." Aarseth's definition has three elements: the space, the challenges, and the goal. In addition, we suggest a forth element to specify a quest: the setting. The quests in *Charbitat* are framed within a larger dramatic setting that is defined by the fictional game world that must be healed. The player's engagement with the game world changes depending on the way quests are framed. Game quests like these are part of a game's fictional world [8]. Goals and challenges of a quest are situated within the virtual space of a game world, which in turn is situated within this larger fictional dramatic setting. Together these coalesce into four core elements that are the framework for understanding and defining quests in a virtual world: The setting, the space, the challenge, and the goal.

This definition poses certain demands to quest generation. Certain conditions have to be met in order to make a quest recognizable and accessible to the player. First, the player must be made aware of the quest setting, understanding the goal and objective. Next, the quest must be situated in an accessible space, within which the player has the capacity to fulfill the specified goal. The goal must be attainable, and there must be obstacles to challenge the player to overcome.

For a key-lock structured quest in our case this means: the goal is made evident when the player first encounters a lock in the form of an spatial barrier; when the player has found the specific token (key) that allows him or her to overcome the barrier a second key level in the quest has been reached; using the key to unlock the barrier opens up the space and completes the quest. All of these steps have to be clearly implemented in a procedural quest generation.

ON CHARBITAT

The *Charbitat* project is a full modification of the *Unreal Tournament* game. It uses procedural techniques to generate a game world at runtime as the player explores the already existing environment. The system has been described in greater detail in other papers [9, 10]. The goal in *Charbitat* is to generate space according to the player's actions in the game world. The world is partitioned into square tiles, which may be thought of as the basic unit of space within the game. The player navigates the overall world stitched together from infinite tiles that are calculated on demand and then placed in the world.

Figure 1: Different Elements

The tiles in *Charbitat* can contain 3D objects that might ordinarily be placed by a level designer. These include static mesh objects, such as trees and rocks but also dynamic elements such as lights, sounds, creatures, pick-up items or powerups. While the tiles are generated as terrain structures using height maps and malleable surfaces, objects are spawned according to certain rules and conditions. Every single tile is a small world generated in a combination of the two aforementioned generation techniques. *Charbitat* traces player behavior within the world and uses this data as seed values for the tile-generation. At the same time, the overall game world is also weighted. Tiles can have global features that span across a single territory. These features include rivers, walls, cliffs, roads, or coastlines. The generated tiles persist behind the player as she explores, coalescing the empty space into a landscape. The player leaves the world in her trail, complete with rivers, forests, and mountains as she moves through the game. At any moment players can load and save worlds to return to them later.

In our tests, the open nature of the game world in *Charbitat* was appealing to players, but it lacked context. It invited players to explore and create more of the world but did not provide much of an immediate direction or context between different tiles. The game has an overall narrative and dramatic setting with several goals within the space, but the space was not confined or limited in any way. A quest structure was seen as necessary to direct the player's activity.

With the tile system, the space in *Charbitat* is an open, literally infinite, world. Once generated the world is a contiguous environment. Thus, our quests had to span beyond single tiles and operate on the level of the overall world instead. The terrain generation offered already affordances for impeding and blocking player progress: rivers, walls, and cliffs. Games that use key and lock puzzles, such as the *Zelda*, *Metroid*, and *Castlevania* series, use features like these to block the player's progress within the space. Instead of partitioning the game experience via

levels and stages, they use the environment itself to limit the player. In addition, we included virtual walls and bridges as artificial barriers, that can be positioned anywhere in the generated landscape. With this collection of procedurally generated elements, *Charbitat* provides a useful platform for quest generation because it fulfills the demands for setting and blocking player progress in a highly flexible way.

KEY AND LOCK PUZZLES

Key and lock puzzles are a widespread convention in games, but tend to be most effective when the keys do not just open doors but add an extra dimension to the gameplay. Keys enable the player to perform new actions within the game world.

In *The Legend of Zelda: A Link to the Past*, the magic hammer can dispose of posts that block the player's progress but it also is powerful against certain enemies. The bombs that are found early in the game can be used to open sealed walls and they can also be picked up and thrown to fight enemies. In *Super Metroid*, the player can find a grappling beam that allows movement to not only previously inaccessible areas, but allows shortcuts through some areas that have already been previously visited. The engagement with the game space, quests, and objects situated therein are closely intertwined on multiple, but not always directly connected, levels. They carry characteristics of puzzles.

Game designer and theorist Chris Crawford has argued against puzzles as too static game elements [4]. But here the key and lock puzzles are realized in a generative space, which provides for very flexible structure. The puzzle is finding out what is an obstacle, what and where is a key to overcome it, and finally using the key to master the challenge. In pre-designed environments this part is static for most games but due to the player-driven and procedurally generated worlds in *Charbitat* the conditions in our system are ever changing.

The key and lock puzzle may be considered solved when the location of the key is revealed, and the player is free to move on to the next area. However, the appeal of the key and lock puzzle is not only in determining the location of the key and navigating to the next obstacle, but it is in the thrill of meeting challenges along the way and the interaction with the space and the key itself, which extends the player's ability to interact with the game world.

Each key that needs to be found is its own quest, and the path to the key may be fraught with challenges and obstacles, reinforcing the power of freedom that the newfound key gives to the player. The puzzles tie the quest to the space of the game world. The lock itself is a property of the space, embedded in the game environment. When the nature of the lock in space is realized, the player's goal becomes to apply the key so that they can overcome the lock in the space. This introduces the quest: find the key. The challenge is to find and apply the key item itself. To keep this search engaging, the generation method must place obstacles and challenges along the way to obtain the key and finally use it to overcome the challenge. The search is dramatized and not just a matter of mere retrieval. With that we achieve our initial goal: to contextualize the procedurally generated game world and increase player involvement with it. The key and lock puzzle is the bridge between the generated space and the quest.

IMPLEMENTATION

World and quest generation in *Charbitat* happen during the expansion of the game world. Whenever a player reaches the end of the current world and steps up to an edge of a tile in *Charbitat*, a message is sent to the Java backend to create a new tile based on the current player status and the surrounding world. The backend will go through all of the possible allowable configurations for that spot, and choose the best one. It does this by scoring each possibility according to rules that characterize the qualities quests should have in the space. These rules are programmatically defined and shall be explored momentarily.

In order to provide a useful extension to the current world, the backend has to analyze the current condition and select from the countless possible additions. For the key and lock generation it specifically has to be aware of what keys and locks exist in the world and how they are arranged. Based on that knowledge it creates tiles that manifest the appropriate quest structure in the new game space using structures such as rivers or walls that can block player progress and spawning keys as pick up objects in other tiles,

Locks are a property of the tile configuration. The matter of choosing configurations and determining where to place keys requires a thorough analysis of the game world. This analysis is done using a graph. Using the graph network, the procedural quest generator applies the necessary conditioning to structure the player’s progress. It is here that the generator makes sure that all locks remain unlockable and every key is spawned in the proper section. At the same time it takes care such that not all keys will appear too fast and too close.

Thinking only of the key and lock puzzles, the world may be decomposed into a graph of nodes. Nodes are identifiers for game spaces and describe to which region this space belongs as well as the condition of this game space. Because any tile might be separated by spatial barriers any single time can contain different nodes belonging to different regions that, in turn, depend on certain keys. Nodes within a tile are connected to each other, as well as to nodes in adjacent tiles. Each connection may have a lock that defines what type of barrier exists between the areas represented by the nodes. A tile with a river running through it will have two areas which are connected by a “swim lock”, indicating that the player must have an item that permits swimming in order to pass from one area to the other. The area nodes correspond to the two banks on either

side of the river. If the player is in one of the areas, there is no way for him or her to move to the other side of the graph without the appropriate key.

Figure 2: A tile split by a river.

The locks in *Charbitat* are of a simple one to one mapping. Locks are often represented as gates in the game world, and the key is an item that will destroy the barrier. We have implemented a number of unique keys: a swim capability to cross rivers, bombs as weapons as well as key to destroy crumbling walls; a water weapon to put out a gate of flames. This logic represents generic key and lock situations using color coded walls and keys: a red key for a red door, a green key for a green door, and so on.

Charbitat also spawns the inhabitants of any tile. Thus, although it is not implemented in the current version of *Charbitat*, it would be possible to add enemies, specifically boss enemies, to "guard" keys. This would provide a growing level of dramatic tension in finding key items in the game world and add to the existing challenge to solve the quest.

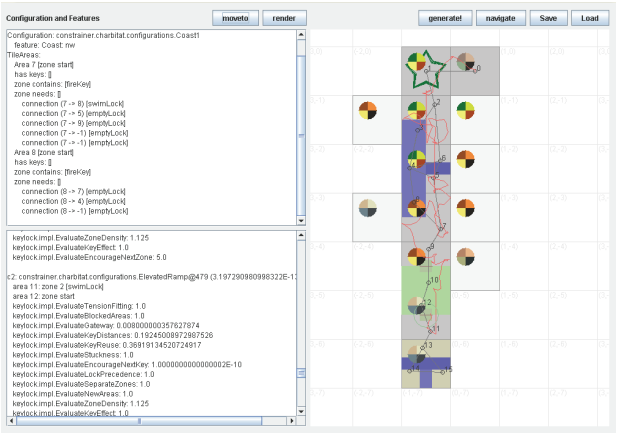

Figure 3: Java backend in action

The actual quest structure works by using rules that dictate what kinds of keys and locks should be placed in the world. These rules are defined using snippets of code called *evaluators*. These evaluators take an area network, with all of its nodes and locks, and produce a score, which ranks that particular world. There are usually several of these evaluators at work, which rank networks based on several different guidelines for how the key and lock puzzles are supposed to be implemented.

When the backend is going through the different configurations, it will compare the full area networks that would be created given the configuration in question, and score the configuration based on the evaluators assessment of the network.

Evaluators work addressing the arrangement of locks, the grouping of accessible areas, and the placement of keys. There are around a dozen evaluators in total that work in Charbitat. Each evaluator aims to fix some particular rule about key and lock placement in the world. Some evaluators encourage the placement of keys under certain circumstances; others restrict the placement of keys in other circumstances. These evaluators must aim to select configurations that will lead to a working whole. The tiles are only parts of the game world, but the goal is to structure the overall world, thus the evaluators must not choose configurations that are the best at the moment, but those which will lead to the best overall results for the game world.

Each evaluator serves a specific purpose, representing some property of quests that we have determined for the world of *Charbitat*. One of the rules used in *Charbitat* is that there must be at least three locks of a given type that precede the appearance of a key. The evaluator that enforces this ranks poorly any network in which fewer than three locks appear before the corresponding key is placed. Another rule is that the player should see many types of locks in the beginning of the game, so that when these areas are revisited, the player will be able to access areas that were previously

inaccessible. This rule's evaluator ranks highly networks in which there is a great diversity of locks visible in the currently generated world. *Charbitat* uses a total of about 13 simple rules to define its space, but the choice and tuning of these rules is a matter of game design rather than the formulation of the quest itself.

The rules employed by the evaluators are flexible, and can be modified to change the style of world and the resulting quests and experience. Different types of rules may be chosen to change the relation between the player's interaction with the space and the keys and locks. Rules can be defined to adjust the curve of dramatic tension, by placing enemies and bosses near the keys. The evaluators could be tuned to encourage backtracking through previously explored space, so the player can find parts that they missed without the new keys they've found. Alternately, they could be tuned to eliminate backtracking entirely so the space is entirely linear. The construction of the evaluators thus gives a tremendous amount of design control over the possible resulting worlds.

Since each evaluator encodes a specific rule, by changing the programming of the evaluators, the world that they create would be changed. However, in keeping with our definition of quests, the tiles automatically encode the space into the world, while the evaluators introduce the setting, challenges, and the goals by placing obstacles, opponents, and keys. The underlying system is flexible enough to accommodate many variations while still maintaining the quests' fundamental characteristics. Without these constraints, tiles would merely be random collections of meaningless areas. It is in this way that the problem of building a game world changes from raw construction to a more manageable problem of selection, and the context of the quest is infused into the procedural game world.

CONCLUSION

In this paper we have analyzed quests in video games with a focus on their spatial situation and conditioning. We consider both as key elements of any player situation within a game universe. Based on the notion of quests as consisting of setting, space, challenge, and goal, we introduced a quest generator that was implemented and tested in a procedural game world prototype. This generator uses tiles and configurations to partition and organize space. By embedding obstacles in the space of the generated game world itself the generator can create quests around key and lock puzzles that are situated in the game space. In return, we argue that players become more engaged in the game space thanks to the engaging context.

One of the first challenges to overcome was adapting *Unreal Tournament 2004* to work with the procedurally generated environment. *Charbitat* was designed as a full modification with the Java backend running in parallel. Most real time game platforms – including *Unreal* – are still designed for static levels or rigidly defined spaces. We had to coax the generative behavior into the otherwise static world. This problem comes in two parts: the first is in actually creating the environment within the space, and the second is in overcoming the lack of scene optimization that is available for static scenes.

A second challenge occurs in defining the evaluators. These are of tremendous importance in channeling the gameplay but are difficult to write. Ultimately, these evaluators must be able to evaluate a whole game world and make a decision for which new tile will best fit into the whole picture. The quality of any quest depends on them. In *Charbitat*, this decision making is incremental as opposed to holistic, which allows the world to be built up freely. But this also means that occasionally the backend will have a hard time getting the world to fit together so that it flows correctly.

Charbitat has been demoed and played by visitors and other researchers on numerous occasions. Yet, using this as a basis to evaluate the quest generation in a traditional usability way remains difficult. First, because the notion of the quest might be understood very differently by different players; second, because every player of *Charbitat* creates a unique game world with different conditions and quest set ups. No quest world is ever repeated and any direct comparison between player performance in the game world becomes dubious. Because our system delivers a player-driven and completely unique game environment it becomes difficult to compare two player performances next to each other.

What became clear in the testing was that the prototype supported better orientation and higher engagement with the quest generator in place. *Charbitat* always featured enemy encounters and an appealing visual environment, but with the quest system at work players felt most intrigued by this structuring of spatial progress. Any virtual barrier inevitably triggered the desire to circumvent or overcome it. Providing means for that through our system was an effective answer to that call. We interpret this as a first indication for a successful referencing of existing game play mechanics in a generative environment. Basic as the key and lock puzzle set up might be it activated the player to engage in a quest and this activation added to player engagement. To optimize the evaluators and fine-tune this quest-generation more detailed evaluation is still needed. Developing an evaluation framework for quests alone would seem a valid future research endeavor.

The system as implemented in *Charbitat* provides a fundament for further development of procedural context generation. It invites players to experiment with the generation and alter the evaluators. If quests are reflections of and occasions for personal growth, then these evaluators could be individualized to the extreme. Then players indeed can engage with their very own personalized unique quests.

REFERENCES

1. Aarseth, E. "Beyond the Frontier: Quest Games as Post-Narrative Discourse" in Ryan, M.L. *Narrative Across Media*, University of Nebraska Press, 2004.

2. Aarseth, E. "Allegories of Space: The Question of Spatiality in Computer Games" Cybertext Yearbook 2000. Ed. Markuu Eskelinen, Raine Koskimaa. Jyväskylä, Finland: University of Jyväskylä, 2000, pp. 152-171.

3. Cambell, J. *The Hero with a Thousand Faces, 2nd Ed*, PUP Press, Princeton NJ, 1968.

4. Crawford, C. *The Art of Computer Game Design*. Osborne/McGraw-Hill, Berkeley, CA, 1984.

5. Ebert, D. et al, *Texturing and Modelling: A Procedural Approach*, 3rd ed, Morgan Kaufmann, San Francisco, CA 2003.

6. Fuller, M., and Jenkins, H. "Nintendo and New World Travel Writing: A Dialogue" in Jones, S.G. (ed.) *Cybersociety: Computer-Mediated Communication and Community*, Sage Publ: Thousand Oaks, 1995, 57-72.

7. Jenkins, H. "Game Design as Narrative Architecture" in Harrington, P. and Wardrip-Fruin, N. *First Person: New Media as Story, Performance, and Game*, MIT Press, Cambridge, MA, 2004, pp. 118-131.

8. Juul, J. *Half-Real: Video Games between Real Rules and Fictional Worlds*, MIT Press, 2005.

9. Nitsche, M. et al. "Designing Procedural Game Spaces: A Case Study", in *FuturePlay 2006*, 2006.

10. Nitsche, M. et al. "The Many Worlds of Charbitat", in *Game Set and Match II*, 2006.

11. Parish, Y. and Müller, P. "Procedural modeling of cities", in *SIGGRAPH '01*, ACM Press, 2001.

12. Propp, V. *Morphology of the Folktale 2nd Ed*, University of Texas Press, 1968.

13. Szilas, N., "Interactive Drama on Computer: Beyond Linear
Narrative" *AAAI Fall Symposium on Narrative Intelligence* Tech. Rep. FS-99-01 1999 pp. 150-156.

14. Tosca, S. "The Quest Problem in Computer Games" in *Proceedings of Technologies for Interactive Digital Storytelling and Entertainment conference*, Darmstadt, 2003.

15. Wonka, P. et al, "Instant Architecture" in *ACM Transactions in Graphics* 22, 3, 2003.

REFERENCES (GAMES)

Spore, Wright, W. for Maxis/Electronic Arts, 2007

Elite, Braben, D. and Bell, I. for Firebird Software, 1984

Myth, Bungie, 1997

Rogue, Toy, M. and Arnold, K. for Artificial Intelligence Design, 1983

Nethack, open source game, 1992

Rescue on Fractalus!, Fox, D. for Lucasfilm Games/Activision, 1986

Disgaea: Hour of Darkness, Nippon Ichi/Atlus, 2003

Phantom Brave, Nippon Ichi/NIS America, 2004

Diablo, Blizzard Entertainment, 1996

World of Warcraft, Blizzard/Vivendi Universal, 2004

Counter Strike, Le, M. and Cliffe, J. for Valve Software/Vivendi Universal, 1999

Return to Castle Wolfenstein, Id Software/Activision, 2002

The Legend of Zelda (series), Miyamoto, S for Nintendo 1986-2006

Metroid (series), Retro Studios/Nintendo, 1986-2007

Castlevania (series), Konami, 1997-2006

Unreal Tournament, 2004, Bleszinski, C. for Epic/Atari, 2004