On the Structure of Role Playing Game Quests

António Machado
Universidade de Lisboa, Instituto Superior Técnico, INESC-ID
antonio.machado@tecnico.ulisboa.pt

Pedro Santos
Universidade de Lisboa, Instituto Superior Técnico, INESC-ID
pedro.santos@tecnico.ulisboa.pt

João Dias
Universidade de Lisboa, Instituto Superior Técnico, INESC-ID
joao.dias@inesc-id.pt

Abstract

In this paper, we present a quest structure (in the form of a grammar) obtained from an analysis of the main quests of a single player role playing game (RPG), namely "*The Witcher 3 - The Wild Hunt*". This grammar extends a previously presented analysis by other authors on MMORPG quests. This extended grammar is suitable for application in single player RPGs. The grammar allows the procedural generation of quests in any RPG wanting intricate quests. We believe that this extension makes the previous grammar more expressive, and it will bring us closer to being able to represent and procedurally generate quests that are equal to human-authored ones.

Keywords: procedural generation, interactive storytelling, RPG, quests, story

Título: Sobre a Estrutura das Missões dos Role Playing Games

Resumo

Neste artigo, apresentamos uma estrutura de missões, sob a forma de uma gramática, obtida através de uma análise às missões principais do *role playing game* "*The Witcher 3 - The Wild Hunt*". Esta gramática estende uma análise feita previamente por outros autores a missões de MMORPGs. A gramática alargada é própria para aplicação em RPG modo *single player*, permitindo a geração procedimental de missões para qualquer RPG que deseje ter missões mais intricadas. Acreditamos que esta extensão torna a gramática anterior mais expressiva e que irá ajudar-nos a representar missões geradas procedimentalmente, de forma semelhante a missões escritas por autores humanos.

Palavras-chave: geração procedimental, narrativa interativa, RPG, missão, história

1. Introduction

Computer Role Playing Games (CRPGs), commonly referred to as Role Playing Games (RPGs) are a video game genre where a player embodies a story world character (and/or several, commonly referred to as party) and must overcome a series of linked challenges, ultimately achieving some overarching goal or the conclusion of a central storyline. Furthermore, the player can develop his/her character(s) through consequential decisions. RPG's are known for being content-heavy games, possessing vast worlds with various non-playable characters (NPCs), as well as intricate storylines and side-quests [Hartsook et al 2011]. The process of this content creation takes a considerable amount of time and money [Li and Riedl 2010]. It's consumption is much faster though, and after a player completes every main quest and side-quest, the game's replay value drops off considerably.

Consequently, RPGs are perfect candidates for the application of Procedural Content Generation (PCG), which is the use of computer algorithms for creating content that meets a set of evaluation criteria [Hartsook et al 2011], [Togelius et al 2011]. This becomes quite useful, when trying to produce content for the game industry, that is becoming more demanding [Hendrikx et al 2011]. According to Hartsook [Hartsook et al 2011] there are two broad uses of PCG in games: content creation and adaptation of gameplay. Through the automatic generation of content one could offload the task of content creation. Thereby reducing the amount of work done by humans and making development costs cheaper. Also, by learning players' information that can't be known during design-time, such as their preferences, desires and abilities, one could adapt game content. Having a game with personalized story and world could maximize player pleasure and minimize frustration and boredom [Hartsook et al 2011].

PCG has become quite popular in recent years, currently being used in a variety of different subfields. Examples of generated content include: dungeons and maps (Binding of Isaac: Rebirth, Diablo series); enemies (Left 4 Dead), animation of character behaviour (Spore), weapons (Borderlands 2) and even whole universes (No Man's Sky). Another area where PCG could be used is in generating interactive stories (examples given in the next section). A system based on interactive storytelling must be capable of generating interactive narratives in a coherent and believable fashion. This means that the sequence of events that constitute the overarching story must be causally and temporally coherent and characters that partake in these events must act in a believable manner [Brenner 2010]. In storytelling systems, characters are perceived as being believable when the actions they execute are motivated by their desires and intentions and these are consistent with the knowledge they possess about the current story world [Riedl and Young 2010].

In the context of RPGs, the sequence of events that constitute the overarching story, can also be called quests. Quests are tasks given by NPCs to a player in the form of requests, that ask the player to complete goals often in return for some reward. If quests are seen as the mere movement from one location to another to achieve a goal, which involves character actions and dialogue, one can see that a sequence of these quests could shape a story. In games, story is the progression of the player through space [Ashmore and Nitsche 2007], [Hartsook et al 2011]. The set of quests that are necessary to complete the game form up the main story. Additional side-quests are often offered to the player to extend the gameplay [Li and Riedl 2010]. Having a system that procedurally generates these quests can potentially increase the variability and replayability of games.

This paper will focus solely on quest structure, and presents the results obtained from our structural analysis of the main story quests from "*The Witcher 3 - The Wild Hunt*"(Witcher), an acclaimed single player RPG game. This analysis departs from and further extends the work of Doran and Parberry [2011] that consists of a structural analysis of several MMORPG quests. The paper here presented is an extended version of the paper that originally appeared in the proceedings of Videojogos 2016. We consider a quest's structure to be all actions performed by the player, since the moment the quest is given to him/her, until the quest's goal is achieved. Using this structure, we can procedurally generate a variety of intricate quests and apply them in a RPG. The remainder of this paper is divided into four sections: 1. review of several approaches; 2. presentation of the results of the structural analysis of the main quests from "*The Witcher 3 - The Wild Hunt*"; 3. an example of a quest from Witcher, based on the results obtained; 4. a brief description of our developing project; 5. conclusions and future work.

2. Related Work

To solve the problem of plotline adaptation, Li and Riedl [Li and Riedl 2010] present an offline algorithm that, given a main plotline (consisting of a sequence of quests), a library of quests, and a set of player requirements, produces "*a sound, coherent variation*" of the original plotline, while preserving the human authors' intent and meeting player requirements.

The complete plotline is represented by a partially ordered, hierarchical plan composed of events that will unfold in a virtual world. These events occur within and outside quests, and are represented by actions performed by the player, non-player characters in the virtual world. Events possess preconditions that must be satisfied, and effects that become true. Causal relationship between two events is established through links via some condition that needs to be satisfied. They allow abstraction hierarchies, through decomposition of abstract events into less abstract ones.

In their approach quests are represented as *top-level* abstractions. Quests have only one effect, the acknowledgement of its completion, and they may or may not have preconditions. Quests are then decomposed into two abstract events: a task and a reward, which are further decomposed into basic actions. A quest that requires the player to hunt down a witch, can be decomposed in the following way: the player would first have to get a water bucket to pour on the witch, ultimately killing her (the task event), in order to acquire the trust of the king (the reward event). In between, events like the witch dropping her shoes, the player picking them up and showing them to the king as proof of the achievement, would complete the plotline. Narrative soundness and coherence, are then guaranteed, through the satisfaction of all preconditions of an event, connecting each event through causal links, thus creating a path that leads to a significant outcome.

The game plot adaptation algorithm takes the partial-order plan described, as well as the set of player preferences. The search is conducted by adding and removing events until success criteria are met. Once complete, the resulting story structure is converted and sent to GAME FORGE system [Hartsook et al 2011] that renders a world that supports the story and executes the game. Indeed, although Li and Riedl [Li and Riedl 2010] are capable of producing quests with a certain amount of control, based on players' requirements, their approach is still dependent on human authoring for the sequence of quests that constitute the plot line. Furthermore, the quests are customized and generated at the start of the game, as opposed to being generated while the player is in play.

Doran and Parberry [Doran and Parberry 2011] did a structural analysis of almost 3000 human-authored quests from several Massive Multiplayer Online Role Playing Games (MMORPG). The analysis showed a common structure shared by human-authored quests, "*changing only details such as settings, but preserving the relationship between actions*". They observed structural patterns in quests, which occurred in predictable situations, each with its own implicit preconditions and effects.

They first observed that quests can be categorized into 9 distinct NPC motivations: Knowledge, Comfort, Reputation, Serenity, Protection, Conquest, Wealth, Ability and Equipment. They believe the use of motivations to be essential for ensuring intentionality in the generation quests. Quests are thus intended to represent a NPC's prime concern. Each of these motivations contains 2-7 motivation-specific strategies. In turn, each of these strategies is composed of a sequence of 1-6 actions, that the player must perform. Each action is further defined as either an atomic action performed by the player, or a recursive sequence of other actions or action variants [Doran and Parberry 2011].

The quest structure is represented in the form of a grammar, in which terminal symbols are atomic actions and non-terminal symbols are action rules, that extend to further actions or action rules. Here, atomic actions are viewed as concrete actions performed by the player during the game. The sequence of actions, that the player is required to perform to complete the quest, can be viewed as the leaves in a tree, with the root representing the entire quest [Doran and Parberry 2011]. Actions can also be replaced by sub-quests, that use the same structure.

With this structure, made from the extracted rules and commonalities of the analysed quests, the authors are able to demonstrate a prototype system that procedurally generates quests, which in their view are appropriate for use in RPGs. The generator starts with an NPC motivation, from that, the generator consults the list of specific strategies, selects one and creates a quest that addresses the motivation. The generator was written in Prolog, due to its "*ability to backtrack and try alternative solutions*" [Doran and Parberry 2011].

Although this approach makes the characters feel a little bit more believable through the linking of a motivation behind their offered quests, we have yet to talk about coherence between the sequence of generated quests. A solution for this problem was proposed by Paolo Burelli and his peers [Burelli et al 2015]. They developed a quality system also based on the structure defined by Doran and Parberry [Doran and Parberry 2011]. Their system provides a way to increase cohesiveness and connectivity between quests, by adding a progressive tier system. They introduce 2 concepts: NPC tier, which helps rank the quest in terms of influence and level of the NPC, and NPC profession, which influences the type of quests that are generated. Tiers are divided into ascending qualities. Each tier has categories containing professions motivations and items, fitting the quality of the tier. In their system, the first quest is always of minor quality. Once a quest is completed, the system performs a calculation to determine if it should advance to next tier, which would have a different NPC providing the quest. This change would be highlighted using a connection text. The tier of the next quest is determined randomly with an increasing chance of progress dependent on the number of quests completed in the current tier.

Quests generated using the structure previously described, which was used by Doran and Parberry on their prototype generator [Doran and Parberry 2011a], are solely based on MMORPGs. As previously stated, according to Doran and Parberry's structural analysis, human authored quests have a shared structure. Having this in mind, we tried to analyse quests from single player RPGs, which tend to have a strong focus on the story component of the game, using the rules defined by Doran and Parberry. We chose "*The Witcher 3 - The Wild Hunt*" since it is a contemporary game with complex structure of quests and was received with critical acclaim.

3. Structural Analysis

For the structural analysis, 58 main story quests from "The Witcher 3 - The Wild Hunt" (Witcher) were examined, to determine whether they also shared the structure extracted by Doran and Parberry from their own analysis. Quest descriptions were obtained from spoiler sites, and from watched walkthroughs. Quest descriptions consisted of the sequence of actions the player had to perform, NPC dialogue and causal and temporal relationships between the different quests.

Using the descriptions obtained and the resulting quest structure from the analysis described by Doran and Parberry, we undertook an attempt to represent Witcher quests. Alas, representing these main quests proved to be more difficult than expected. Indeed, the NPCs that gave out the quests, shared the same or similar motivations (previously described), but since the action rules Doran and Parberry defined had some limitations, it was impossible to fully describe Witcher quests. In order to come up with the necessary changes to counter these limitations (both described in the upcoming paragraphs), each quest was analysed in the following way: first, it was necessary to record every player action in the respective quest; second, a tree was built for each quest, in the manner described in the related work section, using the rules defined by Doran and Parberry, while adding the changes required to represent each quest; third, the changes made were extracted and added to the new set of rules, the results of which can be seen in Tables 1, 2 and 3. Changes made are highlighted in bold. The sequence of actions and rules are written in Backus Normal Form, a notation technique for grammars, same as in [Doran and Parberry 2011a].

Table 1. Strategies for each NPC's motivation.

Motivation	Strategy	Sequence of Actions
Knowledge	Deliver item for study	<get> **<give>**
	Spy	<goto> spy **<report>**
	Interview NPC	<goto> listen **<report>**
	Use item on field	<get> <goto> use **<give>**
Comfort	Obtain luxuries	<get> **<give>**
	Kill Pests	<goto> **<defeat> <report>**
Reputation	Obtain rare items	<get> **<give>**
	Kill enemies	<goto> **<defeat> <report>**
	Visit dangerous place	<goto> **<report>**
Serenity	Revenge, Justice	<goto> **<defeat> <report>**
	Capture Criminal	<goto> **<capture> <report>**
	Check on NPC (1)	<goto> listen **<report>**
	Check on NPC (2)	<goto> take **<give>**
	Recover lost/ stolen item	<get> **<give>**
	Rescue NPC	<goto> **<rescue> <report>**
Protection	Attack threatening entities	<goto> <defeat> **<report>**
	Capture Criminal	<goto> **<capture> <report>**
	Treat or Repair (1)	<get> <goto> use **<report>**
	Treat or Repair (2)	<goto> repair **<report>**
	Create Diversion (1)	<get> <goto> use **<report>**
	Create Diversion (2)	<goto> damage **<report>**
	Assemble fortification	<goto> repair **<report>**
	Guard Entity	<goto> defend **<report>**
	Recruit	**<goto> listen <report>**
Conquest	Attack enemy	<goto> **<defeat> <report>**
	Steal stuff	<goto> <steal> **<give>**
	Recruit	**<goto> listen <report>**
Wealth	Gather raw materials	<goto> <get> **<give>**
	Steal valuables for resale	<goto> <steal> **<give>**
	Make valuables for resale	<goto> repair **<give>**
Ability	Assemble tool for new skill	**<goto>** repair use
	Obtain training materials	<get> use
	Use existing tools	**<goto>** use
	Practice Combat	**<goto>** damage
	Practice skill	**<goto>** use
	Research a skill (1)	<get> use
	Research a skill (2)	<get> experiment
Equipment	Assemble	**<goto>** repair **<give>**
	Deliver supplies	<get> **<give>**
	Steal supplies	<steal> **<give>**
	Trade for supplies	<get> <goto> exchange

When comparing with the structure of Doran and Parberry [Doran and Parberry 2011], the first change worthy of noting, regards the last action of a quest's strategy, specifically "give" and "report" actions. During our analysis, we observed that some Witcher quests required the player to give or report something, as a last step before completing a quest. But some of the strategies that they defined don't allow this. The opposite also happened, where strategies defined in [Doran and Parberry 2011], have a last action give/report, but in the game the player isn't required to do any of those. To give a few examples: in the short main story quest "Disturbance", the player must explore the castle, to find and remove an object ("repair"), that is messing with one of Yennefer's spells. After its removal, the player is required to report back to Yennefer. This sequence isn't fully represented in [Doran 2011a], because it is missing the report action. A second example would be the main story quest "The Sunstone", which requires the player to find and gather the lost item Sunstone. Using Doran and Parberry's structure, the player would be obliged to give the Sunstone after it was gathered. However, this quest finishes as soon as the Sunstone is acquired. This was a recurrent problem, so to counter this obstacle, we decided to put in almost every strategy, one of two action rules, namely a <give> or a <report>. This way, it is now possible to decide during generation, whether a quest's final action should require the player to either report a quest's completion, to give an item back, or neither.

The second change resides in the <goto> set of rules (see Table 3). According to Doran and Parberry's rules, it wasn't possible to be given a sub-quest without having to learn something (see Table 3, rule 9 and rules 12-14), which is something that happened often in The Witcher 3. So, it was necessary to add a <prepare> rule, that offered this possibility (see Table 3, rules 10 and 16). Also, the rule that required learning (rule 10 from Table 3) no longer has a mandatory "goto" action, which limited the order in which certain events could occur. Now, the new <goto> action rule can be expanded to offer more possibilities to the quest. It was also added the action rule <rescue> (see Table 3, rules 33-36), to allow the player to simply free a character, and not having to escort it every single time.

Table 2. Atomic actions.

#	Action	Pre-condition	Post-condition
1.	ε	None.	None.
2.	capture	Somebody is there.	They are your prisoner.
3.	damage	Somebody or something is there.	It is more damaged.
4.	defend	Somebody or something is there.	Attempts to damage it have failed.
5.	escort	Somebody is there.	They will now accompany you.
6.	**examine**	**Somebody or something is there.**	**You have information about it.**
7.	exchange	Somebody is there, they and you have something.	You have theirs and they have yours.
8.	experiment	Something is there.	Perhaps you have learned what it is for.
9.	explore	None.	Wander around at random.
10.	**follow**	**Somebody or something is there.**	**You will now accompany they.**
11.	**free**	**Somebody is there.**	**They are no longer prisoner.**
12.	gather	Something is there.	You have it.
13.	give	Somebody is there, you have something.	They have it, and you don't.
14.	goto	You know where to go and how to get there.	You are there.
15.	kill	Somebody is there.	They are dead.
16.	listen	Somebody is there.	You have some of their information.
17.	read	Something is there.	You have information from it.
18.	repair	Something is there.	**It is fixed, built or resolved.**
19.	report	Somebody is there.	They have information that you have.
20.	spy	Somebody or something is there.	You have information from it.
21.	stealth	Somebody is there.	Sneak up on them.
22.	take	Somebody is there, they have something.	You have it and they don't.
23.	use	Somebody or something is there.	It has affected characters or environment.
24.	**wait**	**None.**	**Wait for something to happen.**

The action rule <defeat> (see Table 3, rules 27 and 28) was added, so it could be possible to decide during the quest generation whether a strategy would require the player to either kill or merely damage an enemy. Before, strategies had one of these actions imposed, which conflicted with the representation of some of the Witcher quests, that had one of those strategies. In some Witcher quests, where it was only required for the player to damage someone or something, the rules defined by Doran and Parberry would instead have the player kill someone or something, and vice-versa. An example of this is the quest "Bald Mountain". After the death of their mentor Vesemir, Geralt (the player's character) and Ciri bloody for vengeance, track down Vesemir's killer Imlerith and ultimately kill him. Using Doran and Parberry's set, it wouldn't be possible to represent this. In their set, the strategy "Revenge, Justice" from the "Serenity" motivation has a mandatory "damage" atomic action, while what we seek is a "kill" atomic action.

Finally, for the actions (see Table 2), we added four new atomic actions that can be performed by the player. During the analysis, we encountered some actions that were not represented in their set of actions. The most significant ones being "examine" and "follow". Doran and Parberry's rules only considered learning either through listening to a character, after doing a sub-quest for said character, or by reading a book. After analysing Witcher quests it was clear that one can also learn by examining clues or objects. One example is the quest "Novigrad Dreaming", where a ghost leaves drawings that, after being examined, show the player what he/she must do next. The action "follow" also appears a lot, practically in every quest. While following a character, the player can learn a bit of the character's backstory and of the possible relationship with the main character Geralt. Actions "wait" and "free" were also added, the first rarely appearing, while the second is an alternative to simply escorting a character, after being rescued. This action appeared more often than the "escort" action in the Witcher's main story quests. A good example for the use of the action "free", is the quest "A Poet Under Pressure". Here the player must rescue the Halfling Dandelion. After following a Witch Hunter that fled from a failed ambush, the player enters the house where Dandelion is being held captive. After you defeat the Witch Hunter, the player can free Dandelion and report to Irina. This quest is also a good example for the addition of the action rule <defeat>. In Doran and Parberry's strategies, the "Rescue NPC" strategy (see Table 1, motivation Serenity) requires the player to "damage" the captor, whilst in this quest the player is required to "kill" the captor.

Table 3. Action rules in Backus Normal Form (BNF).

#	Rules	Explanation
0.	<QUEST> ::= <Knowledge> │ <Comfort> │ <Reputation> │ <Serenity> │ <Protection> │ <Conquest> │ <Wealth> │ <Ability> │ <Equipment>	This is the root of a quest, which expands into one of the 9 motivations. Which will eventually be expanded into one of the strategies, specific to said motivation.
1.	<subquest> ::= ε	Go someplace.
2.	<subquest> ::= <QUEST> **<goto>**	Go perform a quest and return.
3.	<goto> ::= ε	You are already there.
4.	**<goto> ::= goto**	Go to a known location.
5.	**<goto> ::= wait**	Wait at a location for someone or something.
6.	<goto> ::= explore	Just wander around and look.
7.	**<goto> ::= follow**	Follow somebody or something.
8.	**<goto> ::= stealth**	Sneak by.
9.	<goto> ::= <learn> **<goto>**	Find out where to go to and go there.
10.	**<goto> ::= <prepare> <goto>**	Prepare yourself before going somewhere.
11.	<learn> ::= ε	You already know it.
12.	<learn> ::= <goto> <subquest> listen	Go someplace, perform a subquest, get info from NPC.
13.	<learn> ::= <get> read	Go someplace, get something, and read what is written in it.
14.	<learn> ::= <get> <give> listen	Get something, give to NPC in return for info.
15.	**<learn> ::= <goto> <subquest> examine**	Go someplace, perform a subquest, examine something.
16.	**<prepare> ::= <goto> <subquest>**	Go someplace and perform a subquest.
17.	<get> ::= ε	You already have it.
18.	<get> ::= <steal>	Steal it from somebody.
19.	<get> ::= <goto> gather	Go someplace and pick something up that's lying around there.
20.	**<get> ::= <goto> take**	Go to someone and take something.
21.	<get> ::= <get> <goto> exchange	Get something, go to someone and exchange.
22.	<get> ::= <get> <subquest>	Get something and do a subquest.
23.	<steal> ::= <goto> stealth take	Go someplace, sneak up on somebody and take something.
24.	<steal> ::= <goto> <kill> take	Go someplace, kill somebody and take something.

25.	<capture> ::= <goto> use capture	Go someplace, use something to capture somebody.
26.	<capture> ::= <goto> damage capture	Go someplace, damage to capture somebody.
27.	<capture> ::= <goto> capture	Go someplace and capture somebody.
28.	**<defeat> ::= <goto> damage**	Go someplace and damage somebody.
29.	**<defeat> ::= <goto> kill**	Go someplace and kill someone.
30.	**<report> ::= ε**	There is nothing to report.
31.	**<report> ::= <goto> report**	Go someplace and report to somebody.
32.	**<give> ::= ε**	There is nothing to give.
33.	**<give> ::= <goto> give**	Go to somebody and give something.
34.	**<rescue> ::= free**	Free somebody from imprisonment.
35.	**<rescue> ::= <defeat> free**	Defeat somebody, and free somebody from imprisonment.
36.	**<rescue> ::= escort**	Escort somebody to someplace.
37.	**<rescue> ::= <defeat> escort**	Defeat somebody, and escort a different somebody to someplace.

4. Quest Example

In this section, we analyse the quest "The Beast of White Orchard", one of "The Witcher 3" main story quests, using the new set of rules (see Figure 6 for full quest). The quest is given out by a "Nilfgaardian" Commander with the promise of information about the witch Yennefer, which Geralt (the player) is currently looking for. The Nilfgaardian army is having trouble with a Griffin, that has been randomly attacking its soldiers, and the Commander wants Geralt to kill it. Looks simple, but this quest requires the player several preparatory steps. First the player must find information by talking to a hunter about the griffin. Information like where its current location is or why it is on a rampage. He/she also has to gather a plant with a strong scent that attracts it, by talking with a herbalist. In a way, the player needs to prepare before the encounter with the griffin. This quest can be seen as having the motivation "Comfort", using the strategy "Kill Pests", which starts with the sequence of actions "<goto> <defeat> <report>" (see Figure 2 and Table 1).

The first action rule <goto>, implies that the player must go to the griffin's location. In Doran and Parberry's rules, the player would either have to learn the location, in case it was unknown, or explore. In this quest, instead, the player was required to prepare before facing the griffin, namely to learn about the griffin and gather something to lure it. So, in this case we use rule number 10 (from Table 3) "<prepare> <goto>". The action <prepare> will be expanded to "<goto> <subquest>" (rule number 16) (see Figure 3).

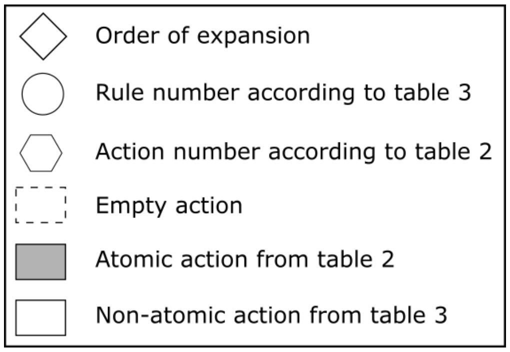
Order of expansion
Rule number according to table 3
Action number according to table 2
Empty action
Atomic action from table 2
Non-atomic action from table 3

Figure 1. Key to all other Figures.

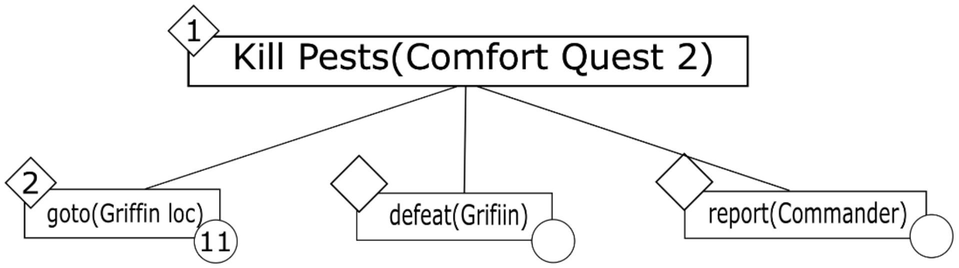

Figure 2. Initial strategy of example quest.

The next <goto> (4th expanded action), requires the player to go to the site where Nilfgaardian soldiers were attacked, but first he/she must talk to the hunter, for guidance. So, the <goto> is expanded to "<learn> <goto>" using rule number 0. The <learn> is then expanded using rule number 12 "<goto> <subquest> listen". The player must first go to the Hunter's house to find that he isn't there, which requires the player to explore and examine clues that direct him/her to the Hunter's location (see Figure 4). Here is the first instance where the use of Doran and Parberry's rules, is unable to represent the quest. It wouldn't be possible for the player to find the hunter, without the atomic action "examine" and the newly added rule number 15, which isn't represented in their set of rules.

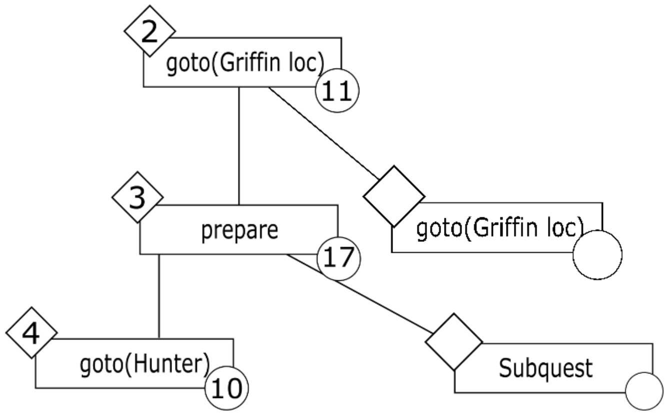

Figure 3. Expansions of <goto> (rule number 11) and <prepare> (rule number 17), both newly added.

After finally reaching the Hunter (12th expansion), the <QUEST> action rule is then expanded to Comfort motivation strategy "Kill Pests". Here the hunter asks the player to kill some wild dogs that are troubling him (13th-20th expanded actions). After reporting to the hunter, the player must follow him to the site where the soldiers were attacked. Again, this atomic action isn't present in the rules defined in [Doran and Parberry 2011a]. Once at the site, the player has again to examine some clues, that lead to tracks that must be followed. Finally leading to the griffin's nest, where the player learns the final details about the griffin (21st-29th expanded actions) (see Figure 5). Notice that if we were using the set of rules defined by Doran and Parberry, this sequence of actions wouldn't be possible. Instead we would have an atomic action "goto", that would make the player go directly to the nest, alternatively to finding his/her way over there.

Thus, closes the <goto> action rule expanded after <prepare> (3rd expanded action) (see Figure 3). Now the <subquest> is expanded using rule number 2 (from Table 3), "<goto> <QUEST>". Before facing the griffin, the player must still gather buckhorn to be used as lure, but first he/she must learn its location. The player must first talk to a herbalist to get this information. This is all represented through expanded actions 32-36 (see Figure 3). Although it wasn't given by any specific NPC, the gathering of the buckhorn is represented through the expansion of <QUEST> using the strategy "Gather Raw Materials" from motivation Wealth, which can be resumed to a simple atomic action "gather", since the player is already at the location (see Table 1).

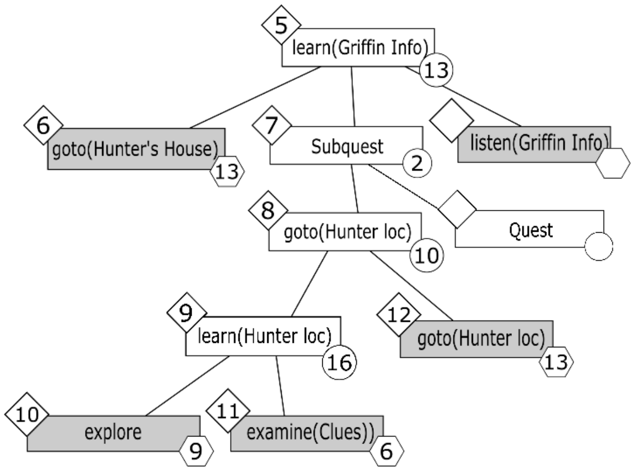

Figure 4. Expansions 5 to 12, use of newly added action "examine".

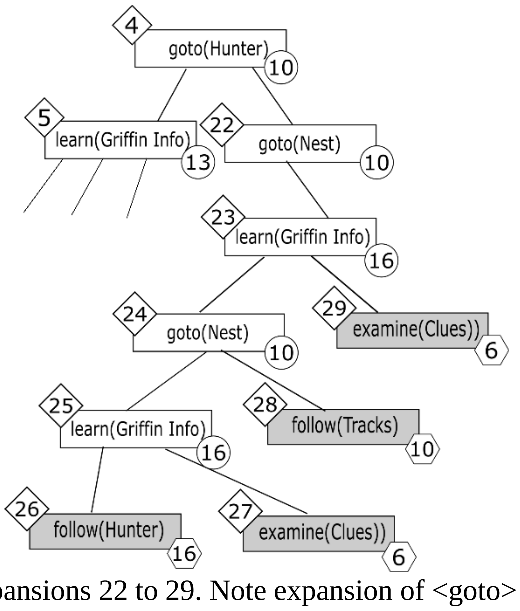

Figure 5. Expansions 22 to 29. Note expansion of <goto> after <learn> is closed and the use of newly added action "follow".

Having gathered every bit of information and the necessary lure, it is now time to move to the griffin's location and ultimately defeat it. The <goto> from the original strategy ends with an atomic "goto" (see Figure 3), continuing with the expansion of the action rule <defeat> (see Figure 2). As stated before, the action rule <defeat> was added, so it would be possible for a generator to decide whether a strategy would have the player kill or damage an enemy. In the end, both actions can be summarized as defeating an opponent. The choice of which action to perform, could then be given either to the player or to the NPC giving out the quest. In this case, the <defeat> rule is expanded using rule 28 "<goto> kill" (see Table 3).

The next step for the player, is to use the previously gathered buckhorn to lure the griffin out. Since it isn't required to learn anything, we expand the <goto> action using rule 10, <prepare> <goto>. The <prepare> action rule is then expanded using rule 16. Now we have "<goto> <QUEST> <goto>", both <goto> are empty since the player is already where he/she needs to be, and <QUEST> is expanded using the strategy "Use existing tools" from Ability motivation, to an atomic action "use"(see Table 1 and Figure 6). Since the player isn't required to move, we are left only with the decisive action of killing the griffin (51st expanded action). Finishing with the <report> action rule, which is expanded using rule 30, "<goto> report"(see Table 3). The player most now return to the Nilfgaardian camp and report to the Commander (52nd-54th expanded action).

As can be observed, in figure 6, all required actions were successfully represented using this new set of rules. The addition of 4 new atomic actions, as well as the addition of action rules <report> and <give> to the strategies, was simply to help represent certain sequences of actions executed by the player in "The Witcher 3". The biggest difference, and probably the most influential, was the removal of the atomic actions "goto", after the <learn> and <QUEST> actions rules (see rules 2 and 9 from Table 3). These atomic "goto"s were restricting greatly the order in which actions could be performed. Their removal, allows for more variability, but we lose control over the size of the generated quest. Nonetheless, it is possible to restrict the expansion of these action rules. This can be done by means of limiting the depth of the tree, or simply by giving a probability for expanding a rule, making some rules less or more likely in certain depths.

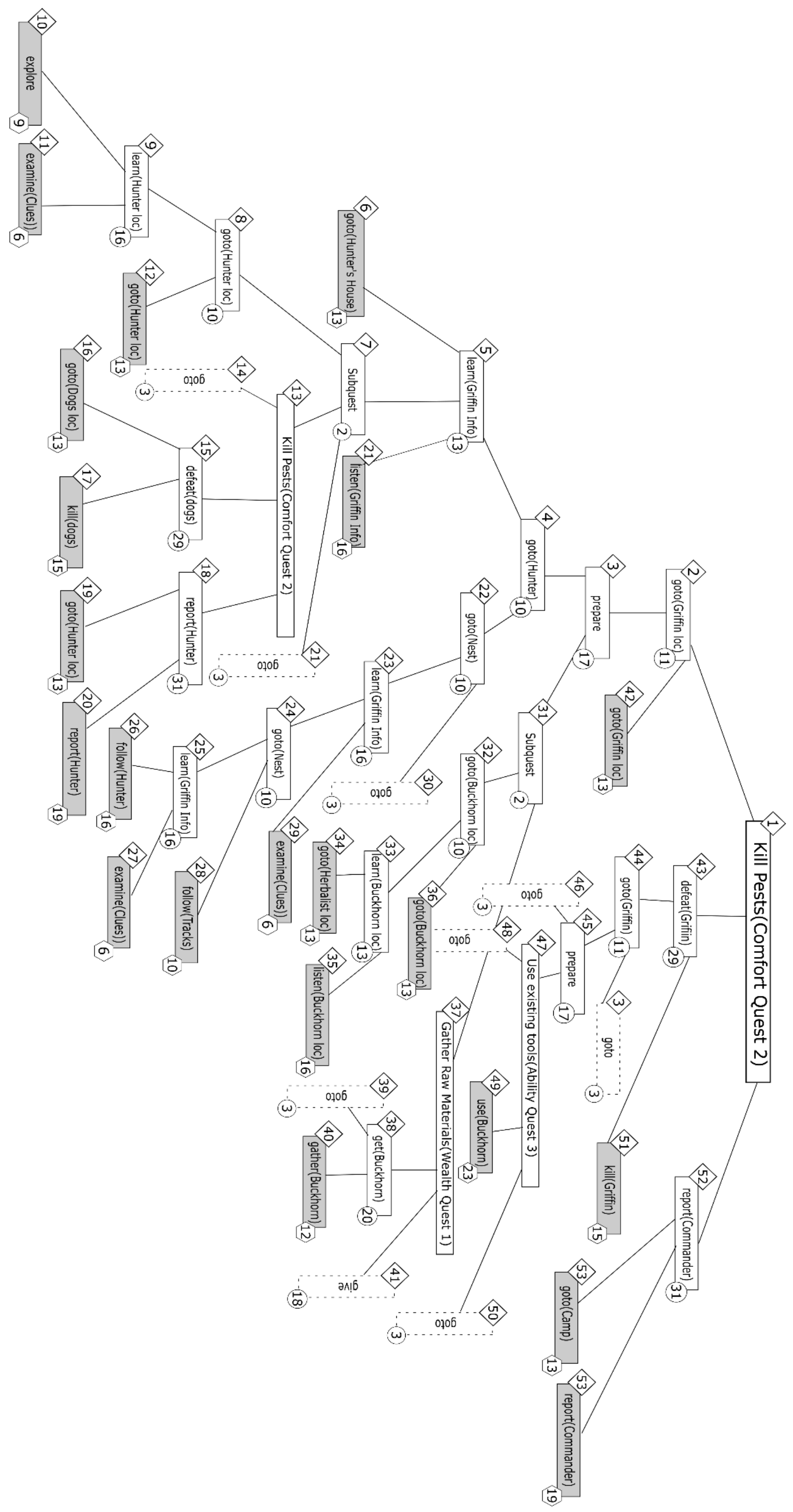

Figure 6. The Beast of White Orchard quest using the new set of rules. The order in which actions performed by the player should be read depth first then left to right.

5. Our Approach

Our goal is to implement a system that generates coherent and believable interactive stories as a series of quests in a RPG type game. The task of generating quests will be distributed between a set of NPCs that will have access to the generator. Each NPC will generate a quest starting with a specific predefined motivation strategy. During quest generation, details from the actions will be filled with characters, items, and locations using the game's world database, until a concrete quest is generated. NPCs could be able to communicate with a predefined set of NPCs to help generate subquests. After a quest is created it will be added to a set of available quests for the player to choose from.

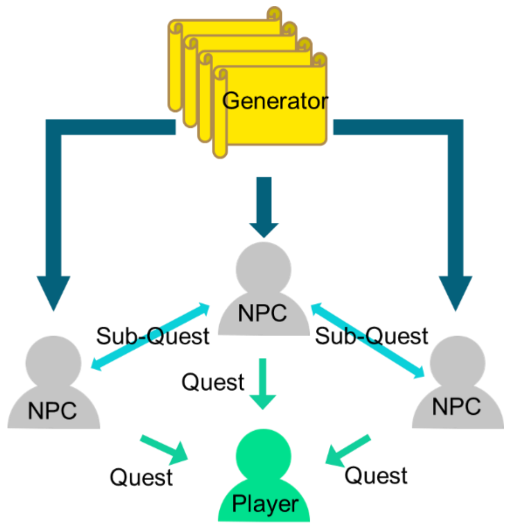

Figure 7. Proposed architecture.

Currently, we have only developed a Quest Editor, that allows the generation and editing of multiple quests based on the structure that resulted from our analysis. The editor generates a quest starting from a randomly generated root node, representing one of the 9 motivations from the structure. For each action in the starting motivation's strategy, the generator selects a sequence of actions in the set of rules (see Table 3), of the current node being expanded. A sequence of action is selected based on weighted choice. A random number between 0 and 100 is generated and compared to a set of ranges, related to the appropriate set of rules, between 0 and 1(both included).

Each sequence of actions has an associated range. If the random number falls into a sequence's range, that sequence gets selected and the weights, related to the set of rules of the node currently being expanded, get updated. To update the weights, we currently use a convergence function of 1/d^2 (although we are looking to test faster convergence functions), where d is the current depth in the quest tree. After this each action in the selected sequence gets expanded, following the same process. The set of weights gets passed to each child node generated. We use these weights and the convergence function to avoid having an infinite growing tree.

As for the specific details of the quest, we currently only enumerate the characters, items or locations that are supposed to appear on it, as the quest is being expanded. Of course, the goal is for the system to use the actual game objects, while generating. The output of our generator would then be a sequence of atomic actions that the player must perform and the game objects associated to each action. Eventually some dialogue and narrative will be added to give a greater sense of what a real game using this system would look like. Additionally, to ensure coherence between quests given to the player, we will use an approach like the one described in the related work section by Paolo Burelli [Burelli et al 2015].

We developed our editor for the Unity Engine and wrote it in C#. The choice for doing an editor came initially as a suggestion from people working in the industry. The editor allows to have both procedurally generated quest and human authored ones. Allied to this editor, we could have an API to help create game objects on the fly and use it as details for the quest's actions. Now we will focus on applying this generator to an actual RPG. We are looking to develop a mod for a RPG game that uses a procedural quest generation system with all the properties here described.

6. Conclusion

In this paper, we introduce adjustments to the quest structure, defined by Doran and Parberry [Doran 2011a], in their analysis of MMORPG quests. These adjustments were based on our study of the 58 main quests from the prize-winning single player RPG game "*The Witcher 3 - The Wild Hunt*". In their paper, Doran and Parberry concluded that further work was necessary to prove the capabilities of their generator in producing quests equal in quality as human-authored ones. We believe that our adjustments to the original structure, make it more expressive. Since no tests were conducted with human players, it is impossible to determine the quality of quests based on this structure. Nonetheless, we hope that this structure gives a significant contribution in making procedurally generated quests closer to the ones written by humans. Note, however, that in our analysis we found some differences between Witcher quests and the MMORPG quests. Thus, we suggest the study of other single player RPGs to further improve this quest structure, hoping it will help close the remaining gap.

The quest structure defined in this paper is to be implemented in a game experience, currently in development. Our goal is to create a system, that uses a procedural approach to story generation. The system will simulate a story world, where a structured narrative can emerge through the interactions between NPCs and the player in the form of quests, based on the structure presented here. NPCs will be responsible for reasoning about which actions are more appropriate for achieving a goal, select them and create a quest that is motivated by their own intentions, thus providing a greater sense of realism. Additionally, the system will also use a learned player model to adapt the subquests to the player's preferences. We hope this system will be able to improve both the chances of replayability and player enjoyment, by offering the player more customized content variety.

Acknowledgements
This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

References

Ashmore, Calvin and Nitsche , Michael (2007), The Quest in a Generated World. Proceedings of DiGRA 2007 Conference

Brenner, Michael (2010), Creating Dynamic Story Plots with Continual Multiagent Planning. Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence. 1517-1522.

Burelli, Paolo and Buss, Daniel and Eland, Morten and Lystlund, Rasmus (2015), The Quality System – An Attempt to Increase Cohesiveness Between Quest Givers and Quest Types. International Conference on Interactive Digital Storytelling 2015. 381-384

Doran, Jonathan and Parberry, Ian (2011), A prototype quest generator based on a structural analysis of quests from four MMORPGs. Proceedings of the 2nd International Workshop on Procedural Content Generation in Games – PCGames '11.

Hartsook, Ken and Zook, Alexander and Das, Sauvik and Riedl, Mark O. (2011), Towards Supporting Stories with Procedurally Generated Game Worlds. 2011 IEEE Conference on Computational Intelligence and Games, CIG 2011, 297-304.

Hendrikx, Mark and Meijer, Sebastiaan and Velden, Joeri Vander and Iosup, Alexandru (2011), Procedural Content Generation for Games - A Survey. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)

Li, Boyang and Riedl (2010), An Offline Planning Approach to Game Plotline Adaptation. Association for the Advancement of Artificial Intelligence-Aiide, 45-50.

Riedl, Mark O. and Young, R. Michael (2010), Narrative planning: Balancing plot and character. Journal of Artificial Intelligence Research, 217-268.

Togelius, Julian and Yannakakis, Georgios N and Stanley, Kenneth O and Browne, Cameron (2011), Search-based Procedural Content Generation: A Taxonomy and Survey. IEEE Transactions on Computational Intelligence and AI in Games.

António Machado is a Masters Student on Computer Science Engineering in the fields of Intelligent Systems and Games at Instituto Superior Técnico (IST). He is currently finishing his Thesis Project with the Research Group GAIPS, INESC-ID. He obtained his Graduate degree in Computer Science Engineering in 2014, also at IST.

Pedro A. Santos is a Senior Researcher at GAIPS, INESC-ID and an Assistant Professor at the Mathematics Department of Instituto Superior Técnico (IST), Universidade de Lisboa. He obtained his Doctoral degree in Mathematics at the Technische Universitaet Chemnitz, Germany, in 1998. His research interests include Operator Theory, Artificial Intelligence, Game Theory, Game Design and Gamification. He co-authored two books, including a book on Game Design and Development, and published more than 40 papers in international journals, and peer-reviewed conferences and workshops, with subjects ranging from Banach Algebras to Multi-Agent Systems simulations, Teaching of Mathematics, Game Design and Gamification. He was a visiting researcher at the Center for Games and Playable Media at the University of California at Santa Cruz, USA, for two months in 2014. He teaches Mathematics and Game Design and Development at IST. He participated in several research projects, the more recent being INVITE (2009-2013) and RAGE (H2020, 2015-2019). He founded three game companies, designed several released commercial games, gamified the course of Linear Algebra at IST and was advisor in the development of several Serious Games.

João Dias is an assistant professor at the Computer Science Department of Instituto Superior Técnico - University of Lisbon, where he teaches courses on Introduction to Programming, Artificial Intelligence, Logic Programing, and Autonomous Agents and Multi-Agent Systems. He is also currently a senior researcher at INESC-ID. In his research, he is interested in developing and studying cognitive, emotional and social agents, and is currently exploring the use of emotional intelligence skills in agents to establish social relations with others in interactive scenarios. He obtained his Doctoral degree in Computer Science in January 2013, and participated in the EU-funded projects VICTEC, eCircus, LIREC and eCute. He has (co)authored about 30 peer-reviewed publications in international journals, conferences and books. He was local organizer for the International Conference on Affective Computing and Intelligent Interaction (ACII) in 2007 and local organizer for AAMAS 2008. He was member of the organizing committee of workshop on Emotional and Empathic Agents held at AAMAS in 2012, and also member of the organizing committee of the workshop on Affective Agents held at IVA in 2014.

(esta página par está propositadamente em branco)