Procedural Quest and Dialogue Generation in Role Playing Games

BACHELORTHESIS 1

Student Jaqueline Wieland, 1410601025
Supervisior DI Dr. Markus Tatzgern

Puch Urstein, 5th June 2016

Eidesstattliche Erklärung

Hiermit versichere ich, Jaqueline Wieland, geboren am **19.11.1994** in **Salzburg**, dass ich die Grundsätze wissenschaftlichen Arbeitens nach bestem Wissen und Gewissen eingehalten habe und die vorliegende Bachelorarbeit von mir selbstständig verfasst wurde. Zur Erstellung wurden von mir keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Ich versichere, dass ich die Bachelorarbeit weder im In- noch Ausland bisher in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der den BegutachterInnen vorgelegten Arbeit übereinstimmt.

Puch Urstein, am 05. Juni 2016

Unterschrift

Jaqueline Wieland | Personenkennzeichen

Kurzfassung

Prozedural generierter Inhalt wird immer beliebter in der heutigen Spieleindustrie. Zurzeit werden prozedurale Algorithmen hauptsächlich dazu verwendet einzigartige Spielumgebungen zu erzeugen. Der nächste große Schritt in diesen Trend ist es nicht nur Umgebungen zu erzeugen sondern auch Quests, Dialoge und sogar Hintergrundgeschichten.
So können die bereits prozedural generierten Welten automatisch mit Inhalt gefüllt werden. Das ist ein wichtiger Schritt in der Spieleindustrie weil die Verbrauchsrate von Spielinhalten immer stärker steigt und Spieleentwickler immer schneller und schneller neue Spielinhalte erzeugen müssen um ihre Kunden zufrieden zu stellen.
Diese Thesis beschreibt was Quests und Dialoge in ein einem Rollenspiel-Kontext sind und gibt einen kurzen Überblick über den Themenbereich der prozeduralen Generierung. Zweitens behandelt es bereits verwendete prozedurale Methoden bezüglich Quests in bereits erschienen Spielen.
Im Hauptteil werden neuere Methoden zur prozeduralen Erzeugung von Quests und Dialogen vorgestellt. Zum Schluss werden wichtige Faktoren die in prozeduralen Systemen zu beachten sind analysiert und ob die vorgestellten Methoden in der heutigen Spieleindustrie einsetzbar sind.

Schlagwörter:

Prozedurale Generierung, Quest Generation, Dynamische Dialoge, Spieleentwicklung, Rollenspiele

Abstract

Procedural generated content is getting more and more popular in the game industry. At the moment procedural generation is primarily used to create unique game environments. The next step in this development trend is to generate procedural quests, dialogues and even backstories, to fill an automatically created world.
This is an important step as the consumption rate in the game industry increases steadily and the producers have to provide new content faster and faster.
This paper describes what quests and dialogues in a role-playing context are and shortly describes what procedural content generation is. Secondly it takes a look at already used procedural methods to generate quests in current games.
In the main part it gives a short overview on newer methods how to create quests and dialogues in a procedural way. Lastly it analyses which factors have to be monitored in order to evaluate a good procedural quest system and if the proposed methods are useable in the game industry.

Keywords:

procedural generation, quest generation, dynamic dialogues, game development, role-playing games

Contents

1 Introduction

In today's game industry it is essential to release new content for a game in a fast paced period of time to maintain the player's interest into a game or game series and to satisfy their need for new content (Hendrikx et al. 2013). Especially in story-based games such as role-playing-games (RPGs) the release of fresh content is mandatory to keep players satisfied and attracted to the game. Currently there are lots of different approaches to answer this need. For example *World of Warcraft*[1] releases a new Add-on nearly every year, to offer new content for their players. Other RPGs often offer additional downloadable content to their game. This opens the possibility for the players to continue their journey through these game worlds and consume more of its content. Hundreds of artist and developers work many years to create content to satisfy their players wish for fresh and fascinating content. But the consumption rate of video game content increases more and more. Despite the numbers of people working on a game, they are not able to create enough new content by hand in a short time while keeping it affordable for their consumers. This fact leads to a bottleneck in the game production. Static hand - crafted content always has its limits, mostly in time and cost (Ashmore and Nitsche 2007).

Due to this development of a continuous faster consuming community, the usage and importance of procedural generated game content increases steadily. To always create new content for players automatically without dozens of hours of development, procedural content generation (PCG) can be used. Besides it creates unique content for players, it is also a huge competitive advantage against other game studios in cost and time (Trenton et al. 2010), (Grey and Bryson 2011). Currently procedural generation is mostly used for environment creation, and very little for quest and dialogue generation (Nitsche et al. 2006). Especially in RPGs procedural generated quests and dialogues would be a massive advantage to generate unlimited content, especially side quests. For the main story of an RPG a scripted storyline is preferable to maintain excitement and dramatic. Although there are already methods to use procedural generated quests, the fear of not coherent and meaningful quests and not yet usable algorithms is prevailing (Lee and Cho 2012).

First of all this paper will approach topics like what dialogues and quests are and how they are constructed. Which quest types are commonly used and what standard patterns are emerging in RPGs. Furthermore this thesis gives and overview on a few different approaches how to achieve an automatically generation of quests. It investigates the advantages and disadvantages of those and their usability in the game industry. Lastly it will investigate factors which has to be monitored to evaluate the quality of quests and dialogues. I takes a look at which elements are necessary to build up a quest and are important for a conversation between player and Non-Player Character (NPC). The whole context of this thesis refers to the usage of PCG to generate side quests in RPGs. In most cases in RPGs the main story line and thus the main quest line should be well written and thought-out by a game designer. The reason is that the main quest line normally is intended to tell a specific story. This could not be achieved by the proposed methods. Also

1. Activion Blizzard. 2016. "World of Warcraft" accessed May 27. `http://eu.battle.net/wow/`

an advantage of a scripted main story is to keep it dramatic and emotionally attaching for the player.

1.1 Procedural Content Generation

PCG is a way to create game content rather via algorithms on the fly during runtime, instead of creating them statically by hand. (Hendrikx et al. 2013) (Smith and Mateas 2011). This leads to a lot of advantages. It gives the possibility to create infinite amount of game content. Also it reduces the needed memory of the game, because the content is created during the runtime . The advantage to use content which is created by algorithms rather than developers or artist is, that a computer can calculate and construct a game world much faster than any person ever could. Which makes development faster and cheaper. It also adds realism and variety to the game because humans tend to work in specific patterns with limited creativity. This increases the replayability of a game in an enormous way.
Although PCG has huge advantages it is not an easy task to create really meaningful, realistic and believable content. It also needs a lot of testing and iteration to ensure a usable outcome. This procedure is called the generate-and-test process. During that part not only the generated content gets evaluated but also the runtime efficiency of the algorithms is getting optimized.
There are many different fields of study in PCG. Often they get classified by which type of content they provide (Nitsche et al. 2006). A lot of different types of game content can be created by procedural methods. For example graphics, textures, character animations, levels and even whole ecosystems.
This paper will mainly look at the process of generating quests in a procedural way and everything that goes hand in hand with that topic. This includes the creation of story as well as the interaction with the world through dialogues NPCs.
Further the procedural creation of game environments will be briefly discussed. Both fields of PCG are essential in this topic. To create quests there has to be a corresponding level design to make them possible in the first place.
The creation of game worlds, environments and levels is probably the currently most used way of PCG. This field of use is already present in the game industry for a very long time. One of the first games which used PCG to generate different maps was *Rouge*[2]. Although these maps were only shown as ASCII based graphics they already used procedural generated levels (Clarke 2014). *Minecraft*[3] uses PCG to provide an infinite world to explore. One of the most recent and advanced ways to use procedural environment creation is shown by *No Man's Sky*[4]. Hello Games managed to create a whole universe, including planets with unique creatures and ecosystems, only by providing rulesets and algorithms. There are a lot more examples how to use PCG to generate an always new

2. Toy, Michael, Glenn Wichman, and Ken Arnold. 1980. Rogue. University of California, Berkeley.
3. Mojang. 2016. "Minecraft" accessed May 27. `https://minecraft.net/de/`
4. Hello Games. 2016. "No Man's Sky" accessed May 27. `http://www.no-mans-sky.com/`

and interesting world for the player (e.g. *Spelunky*[5]).
Just creating game environments is not enough in the game production. A world lacking of interaction, narrative content, goals, challenges and rewards can lead very fast to a frustrating and boring experience for the player. (Ashmore and Nitsche 2007)
So the next big step in PCG is the automatically generation of quests. This commonly includes the generation of dialogues, story, challenges, and goals. This paper will give an overview over a few approaches how to achieve the automatic generation of quests in the context of RPGs.

2 Quests in Role-Playing Games

Before creating procedural quests, it has to be clearly defined what those are and what meaning they have in RPGs. Further each term which is related to quests also has to be clearly outlined. This includes the definition of RPGs, NPCs and the communication between NPCs and the player. This chapter provides all those definitions as well as a closer look at quest types and quest clichés which have emerged in RPGs. Lastly this chapter provides a motivation for creating quests in a procedural way.

2.1 Role-Playing Games

In a RPG the player slips into the role of a character in a virtual game world (Kacmarcik 2005). The player leads the actions and decisions of this character. The main difference to other game genres is the high amount of interaction between the player and the game environment as well as the focus on an exciting and unique story. In most RPGs the goal is to complete a main storyline, represented through a main quest line, and several smaller self-contained side quests. The quality of an RPG depends mainly on the quality and quantity of interaction possibilities with the game world. Another factor is the excitement and dramaturgy of the storyline.

2.2 Non-Player Characters

NPCs are entities in a game which are controlled by an Artificial Intelligence (AI). In today's game they are often implemented as behaviour oriented finite state machines (Kacmarcik 2005). This NPCs are giving the player the opportunity to interact with the game world. They provide a variety of interaction possibilities. NPCs are able to give the player information, quests, narrative information, tutorials, and offer and introduce game mechanics like trading, professions or new skills. The complexity of NPCs in RPGs ranges from the simplest implementation as a NPC which stands only at one place and doing nothing else than provide information and quests for the player like in *World of Warcraft*,

5. Mossmouth. 2016. "Spelunky" accessed May 27. `http://www.spelunkyworld.com/`

to NPCs with their own unique daily routines like in *The Witcher 3*[6] (Grey and Bryson 2011). In the best case a NPC should not be just a soulless entity which humbles through the game world, but rather a representation of a person with personality, emotions and memories. Also it should be aware of its surroundings in the game world and react in a believable way to its local environment. Believability is one the most important factors for a NPC. NPCs from a lot of the current RPGs lack of the ability to react on their immediate environment or only react in a simple, not realistic way (Clarke 2014). This is also an area which has to be improved but is not the main topic of this thesis. Although an later discussed approach in this paper addresses an improvement in that specific field.

2.2.1 Dialogues - The interaction with NPCs

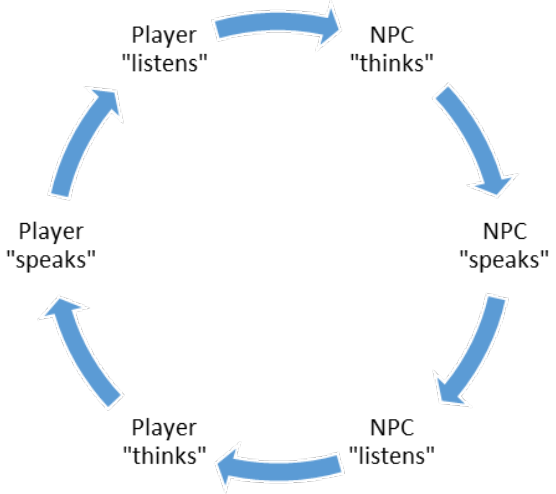


Figure 1: The Interaction Cycle

Dialogues are an essential part of RPGs. Through them the interaction between player and NPCs takes place. In many cases only such dialogues enable the player to get a quest and interact with the game world. Dialogues are the most common way for the player to gain information about the game world and build up an emotional bond with NPCs in the game. Another way for the player to obtain a quest could be through a bulletin board or something similar. But in the most RPGs dialogues are what makes the game a lot more realistic, believable and immersive for the player. A dialogue normally consist of the exchange of a set of lines between two or more parties. Usually between the player and one or more NPCs (Kerr and Szafron 2009).

The processing of a dialogue normally follows a strict order. One of the parties starts the conversation with a line, usually a greeting line, the other party listens to the input, thinks about it, chooses between different answer possibilities and gives a suitable response. This exchange of lines between the player and the NPC continue until the dialogue is fully

6. CD Project Red. 2016. "The Witcher 3" accessed May 27. `http://thewitcher.com/en/witcher3`

processed and ends usually with a farewell line (e.g. “Goodbye, X”). During that process important information are getting passed between all members.

2.3 Quests

Quests are one of the core elements of RPGs and massive multiplayer online role-playing games (MMORPGs).(Smith et al. 2011) Typically the whole progress of the played character, personal and through the world, is achieved through completing quests. These quests lead the player through the game and in most cases also give him essential narrative information as well as rewards to upgrade the character. Quests without narrative content often feel frustrating and trivial to the player (Ashmore and Nitsche 2007). Quests in RPGs consist normally of a given task, a goal, success and lose conditions and narrative context.

The specific type of a RPG quest is characterised by a clear defined start and end point. The player is given a specific task which has to be fulfilled in the given game environment. The motivation to complete those quests is that through them the player is able to develop the in-game character as well as to discover new parts of the game and narrative content. The development of the character is often implemented as a level system, in gaining new skills or gaining better equipment in order to make the character stronger. The progression through narrative content means the discovery of new places, obtaining new information of the game world or progress in the storyline of the game.

2.3.1 Quest Types

According to Jonathon Doran and Ian Parberry (Doran and Parberry, May, 2010) quest types can be defined through the motivation of the NPC who provides this quest. Through a quest NPCs try to satisfy their needs and desires commonly with the help of the player. These motivations result into quests which are leading to different actions the player has to perform in the order to complete those quests. Often more than one of these actions has to be done during a quest. These actions are defined by Doran and Parberry (Doran and Parberry, May, 2010):

1. Damage an NPC or item

   The player has to damage an NPC in the game world or destroy an item. Usually the goal is to kill or destroy that target. In some rare cases it is asked to perform non-lethal actions in order to knock somebody out or capture them.

2. Talk to an NPC

   The goal of this action is that the player talks to a specific NPC with the intent to gain new information about the game world, the NPC or to get another quest.

3. Assemble an item from parts

   The player has to perform an action to create a new item by combining different other items. This leads often to crafting game mechanics like alchemise, forging, and leatherworking.

4. Trade items with an Entity

   The trading with an entity action includes a verity of possible interaction. Commonly it means that the player has to acquire an item and give it to an NPC or drop it at a location. Furthermore the opposite of these actions is included e.g. pick up an item or get it from an NPC. This actions leads to one of the most common quests in RPGs: the delivery Quest. For this quest the player has to get an item (from an NPC, loot from enemies, professions or pick it up at a location) and deliver it to another NPC or location. Such a quest is intended to lead the player through the world and let him explore new areas.

5. Defend an entity or location

   The player has to defend an explicit entity in the game world against attackers. This could be one or a group of NPCs. They could be standing in one place or moving towards another location. The continuous movement would results to an escorting action for the player. Another possibility could be that the player is asked to defend a location against enemies. E.g. defending a building against raiders.

6. Visit an NPC, item or location

   Sometime a player is asked to visit a location or NPC in the world. Often the visiting action is used as an implicit part of the actual quest. E.g. to talk or trade with an NPC at a different location than the players location. Nevertheless sometimes the visiting itself is the goal of the quest. E.g. to spy out enemy territory, explore old ruins.

7. Use an item or skill on an item, location or NPC

   This point includes every interaction between the player and an entity in the game world through a skill the player has or an item in which possession the player is. The meaning of used items has usually something to do with the quest context.

8. Wait

   During some quests the player is asked to just wait at a specific location while some other process is ongoing. This could be a NPC casting a spell, crafting items or narrate something.

2.3.2 Quest Clichés

Through that basic structures of quest types emerged an amount of different RPG Quest clichés like Kill Quest, Fetch Quests, Deliver Quests, Lock and Key Quests or Escort Quests.

Kill Quests
Normally in a Kill – Quest, the player gets the order to kill a specific amount of enemies in an area or a special unique enemy. Sometimes brining back a trophy of the killed enemy is also mandatory to ensure that the task has been fulfilled.

Fetch Quests The goal of a Fetch Quest is that the player has to find one or more special items and bring it back to the NPC.

Deliver Quests
One of the most used quest type. The player gets an item from an NPC, who also gives the quest to the player, and the player has to bring it to another NPC in a different place. The motivation behind that quest cliché is to force the player to meet new NPCs and visit new locations.

Escort Quests
The assignment in an escort quests is to protect a NPC or a group of NPCs during their movement from one place to another. Often these NPCs are getting attacked on their way to the target place by one or more waves of enemies.

Lock and Key Quests
The last type of quest clichés which is often used in games are Lock and Key Quests. There the player encounters a barrier which prevents the player to progress further in the world. To solve such a barrier the player has to find a key to overcome the barrier. This does not has to be literally a key but could be somewhat item, information or ability to pass the barrier. For example to gain the ability to swim (key) to overcome a river (barrier). These type of quest exists since the very beginning of RPGs like *Zelda*[7] or in *Yoda Stories*. This type of quests is often found in Puzzle games and RPGs. It can be rather easily implemented procedurally and has already been used in the industry (e.g. Chapter 3.1.2).

2.3.3 Examples

This chapter shows a few quest examples from well-known RPGs. Those examples are getting analysed and broke down in their different action types.

A Frying Pan, Spick and Span - The Witcher 3
"A Frying Pan, Spick and Span" is a *The Witcher 3* Quest where the player is given the task to find the frying pan of an old lady. She borrowed the pan to an stranger and now

7. Nintendo. 2016. "Zelda" accessed May 27. `http://www.zelda.com/`

wants it back. The pan itself is located in a locked house. The player has to get to the house (Visit an NPC, item or location). In order to get into the house the player has to use the skill "Aard Sign" onto the door to blast it open (Use an item or skill on an item, location or NPC). Now the player has to find the frying pan in the house and pick it up (Trade items with an Entity) and return It to the old women (Visit an NPC, item or location) in exchange to get an reward for the finished quest (Trade items with an Entity).

This example shows that even such a simple quest of finding a frying pan includes a lot of different actions. More complex quests are containing even more of such actions.

When Bears Attack - Dragon Age: Origins
The goal of the Quest „When Bears Attack" in *Dragon Age: Origins*[8] is to kill all bears which are threatening villagers. The player has to track down the bears (Visit an NPC, item or location) and kill all of them (Damage an NPC or item). If he has finished his task he can return and get a reward from an NPC (Trade items with an Entity).

In your heart shall burn - Dragon Age: Inquisition
During the Quest "In your heart shall burn" in *Dragon Age: Inquisition*[9] the player has to defend several trebuchets against attacking enemies in order to protect the village (Defend an entity or location).

2.3.4 Quest Requirements

Nevertheless how quests are structured or in which context they appear, quest are one of the most important storytelling mechanisms to give the players an opportunity, to interact with the game world (Doran and Parberry, May, 2010).

In any case there are a few requirements every quest has to meet at any time:

1. Is has to be reachable for the player
2. The quest must be able to get completed through the player
3. It has to make sense in the game context

There are different ways quests can be given to the player:

1. They can trigger when the player enters a specific level or area
2. They can be started when the players get their hands on a special object
3. And the last, most commonly used way to spread quests is through dialogues and interaction with NPCs

8. Bioware. 2016. "Dragon Age: Origins" accessed May 27. `http://dragonage.bioware.com/dao`
9. Bioware. 2016. "Dragon Age: Inquisition" accessed May 27. `https://www.dragonage.com`

2.3.5 Motivation for Procedural Quests

Scripting each quest by hand is a huge effort for the developers and costs much time and money (Lee and Cho 2012),(Hendrikx et al. 2013). Due to a limitation in budget and time, developers often tend to gives side quests in RPGs less attention and invest less effort into them than in other parts of the game. Often they degenerate to trivial tasks with a low amount of narrative context or none at all. This fact can be very frustrating for the player and reduces the level of immersion (Grey and Bryson 2011). In a lot of RPGs the side quests make most of the playtime. They are the content which keeps the player engaged for a long time while the main story line can often be completed within a few hours. So it is in the interest of the developers to create a huge amount of good side quests with a low amount of developing costs. Another factor to consider besides the saving in time and money is that particular in story-based games like RPGs and MMORPGs the consumption rate of game content increases steadily (Lee and Cho 2012), (Hendrikx et al. 2013). For this reason a method needs to be developed to provide content at least as fast as the consumption rate of the players. This is where PCG helps. PCG algorithms opens up the possibility for infinite content.

3 Procedural Quest and Dialog Generation

The first part of this chapter is about two examples from the game industry which are already using algorithms for procedural quests. The next part consists of explanations and analysis of two different approaches on procedural quests. Lastly this chapter provides information about the quality evaluation of quests and dialogues.

3.1 Examples from the Game Industry

This chapter investigates methods for procedural quests which were already used in released games. Firstly the Radiant Quest System from *Skyrim*[10] will be explained. Secondly the algorithm behind the creation of Puzzle Levels in *Yoda Stories*[11] and *Indiana Jones and his Desktop Adventures*[12] gets analysed.

3.1.1 The Radiant Quest System in Skyrim

Skyrim currently uses a procedural quest algorithm named Radiant Quest System[13]. During the whole playing process the player gets watched and recorded by an Agent. This

10. Bethesda Game Studios. 2016. "The Elder Scrolls: Skyrim" accessed May 27. `http://www.elderscrolls.com/skyrim/`

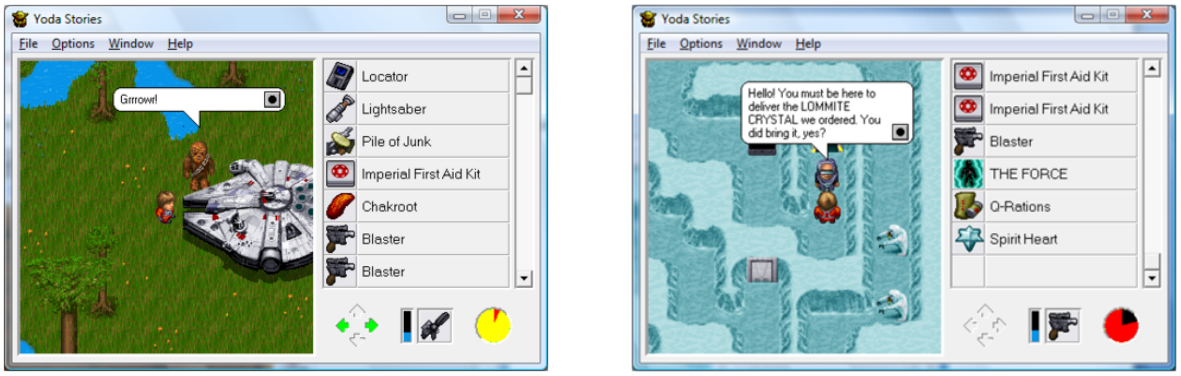
11. Crowley, Mark. 1997. Yoda Stories. LucasArts.

12. LeFevre, Paul D. 1996. Indiana Jones And His Desktop Adventures. LucasArts.

13. UESPWiki. 2016. "Radiant Quest System" accessed May 27. `http://www.uesp.net/wiki/Skyrim:Radiant`

Agent stores with whom the player interacts, in which way, and where. With this information the Agent fills predefined quest patterns to create new quests. The problem with this approach is that these quests do not feel interesting or special enough. Besides Bethesda announced it as there would be an infinite amount of quests, they often repeat. So eventually you have to do the same task over and over again.

3.1.2 Yoda Stories or Indiana Jones and His Desktop Adventures



(a) Forest World **(b)** Snow World

Figure 2: Yoda Stories Screenshots (Corner 2016)

Yoda Stories and *Indiana Jones and His Desktop Adventures* are Puzzle Games which are completely build up via a procedural method (Howard 2016). The world gets generated through the placement of predefined rooms. Some of the rooms consist of puzzles which the player has to solve. Both games have different zones in the game world and the puzzles depend on each other. The player needs to solve them in the right order. Most of the puzzles were based on finding and trading items until the player has found the last item which is needed to open the final dungeon or door. The procedural puzzle generation is solved through a Puzzle Tree. This Chapter will now take a closer look at a simplified version of the algorithm which is used in *Yoda Stories* to get the general idea. This algorithm works not with different depended zones but rather one big zone. However the algorithms can be easily modified for a more complex game.

Different Room Types

First of all we will define different types of rooms in the game world. There are **Filler Rooms** which are used to create a little bit of variety in the level. Filler Rooms are having nothing to do with the puzzles itself but are providing more content to explore. Usually these are filled with enemies who the player has to defeat. **Puzzle Rooms** can be associated with Leaf Nodes in the tree or with normal nodes. The first ones can contain items which the player has to find (e.g. hidden under a rock). The second one usually contains an NPC who trades a found item for another which is then necessary for another puzzle.

The **Start Room** and the **Final Room** are preselected and related. The goal of the Start Room is to give the player a motivation and narrative context to get to the Final Room. The Final Room is the end of the level and also represents the Root Node in the puzzle tree. From there the whole structure of the puzzle generation starts. This is necessary to guarantee a solvable level.

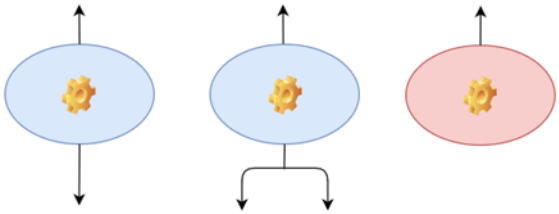
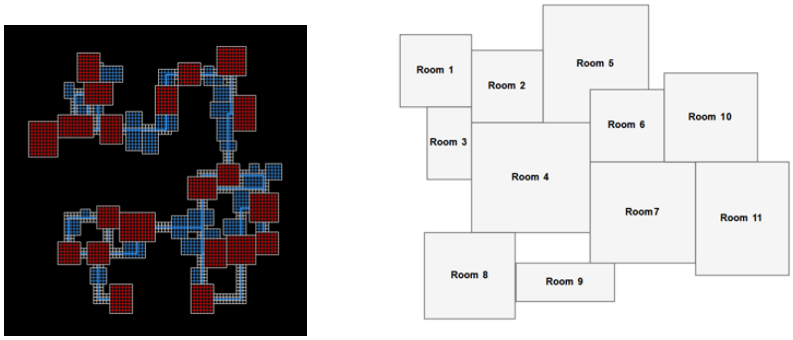


Figure 3: The different Puzzle Node Types

The Algorithm

The Puzzle Tree contains Puzzle Nodes. Each of these node represents a different puzzle. There are Leaf Nodes, like the start room, which do not take any input and producing only one output. (E.g. finding an item X under a rock). These items are needed to solve other puzzles represented by nodes which take one or more inputs and producing one output (e.g. trade X for Y, trade X AND Y for Z). The Root Node, which stands for the Final Room, takes one or more inputs but is producing no output.



(a) TinyKeep Level Layout (PhiGames 2016)

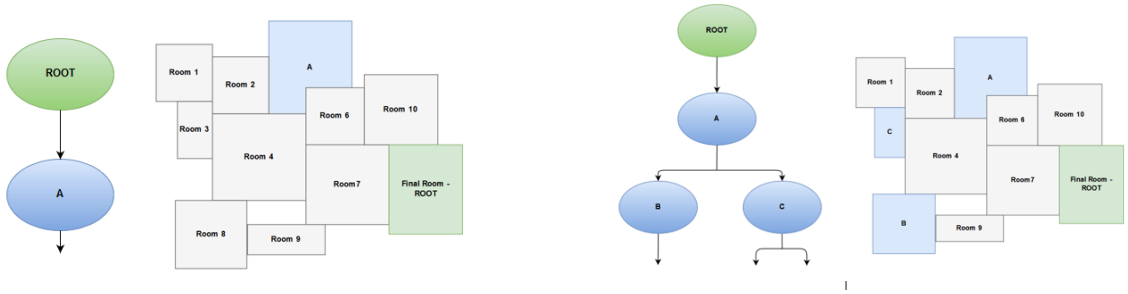
(b) Level Layout used for the example

Figure 4: Procedural Level Layouts

Before the puzzle generation itself can begin there must be a simple layout of the level which can be filled. This basic layout can be produced from another procedural method for level generation. For example the level generation algorithm from *TinyKeep*[14] can be used. The result from this algorithm could be something looking like in Figure 4a. In the following example the level layout from Figure 4b is used.

14. PhiGames. 2016. "TinyKeep" accessed May 27. `http://tinykeep.com/`

This layout can now be filled with the pre-produced rooms mentioned above. First of all the Root Node – the Final Room – has to be selected from the pool of start and final rooms and placed randomly into the empty level layout.



(a) The start of the Puzzle Tree **(b)** Puzzle Tree with placed Puzzles

Figure 5: Puzzle Tree Stages

The next step is to take a puzzle node (A) from the pool of puzzle nodes and place the associated room somewhere into the level. As in this version of the algorithm there are not any locked zones it does not matter where to place the room. This puzzle node will now be added as a child node in the puzzle tree. As it is seen in Figure 5a the puzzle node A takes one input. The output of each puzzle node and the input of the parent node has to match. For example if the Final Room needs a *key card* to open a door, A has to produce this *key card* as an output.
The total number of puzzle nodes placed into the level depends on two factors. The number of rooms the level layout provides and the coverage level the developer wants.

The placement of all puzzle rooms repeats until the number of nodes in the tree plus the number of dangling inputs reaches the total number of rooms that should be placed. At this point in the algorithm, the tree could look something like in Figure 5b.

This tree would mean, in this level will be seven rooms associated with puzzles. Now the nodes with dangling inputs are filled with Leaf Nodes which do not take any input. One of the Leaf Nodes has to be the pre-selected Start Room. The end result of the tree could look like Figure 6.

The rest of the available empty rooms can be filled with the Filler Rooms. Now a complete solvable level with different puzzles and a goal is created.

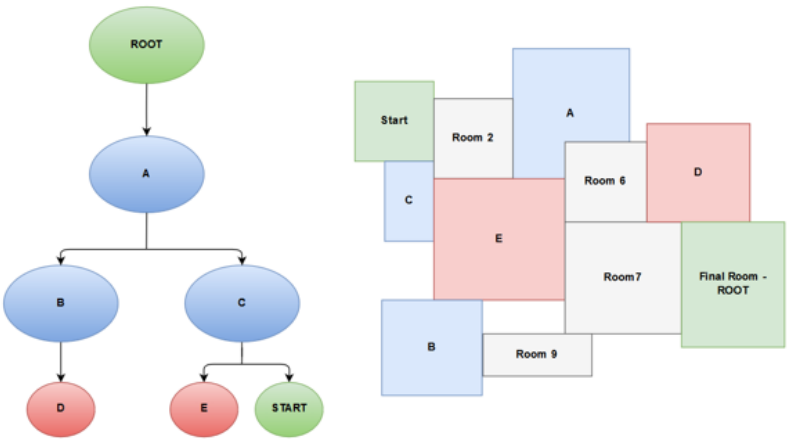


Figure 6: Endresult of the Puzzle Tree

3.2 Algorithms for Procedural Quest Generation

This chapter is about the explanation and analysis of two very different approaches to make procedural quests possible. The first approach is to plan quest with the help of the Petri Net System. Secondly this chapter analyses a completely opposite approach by using a Multi Agent System which deals with the usage of Believable Social Agents (BSA) and the interaction between them.

3.2.1 Planning Quests with Petri Net

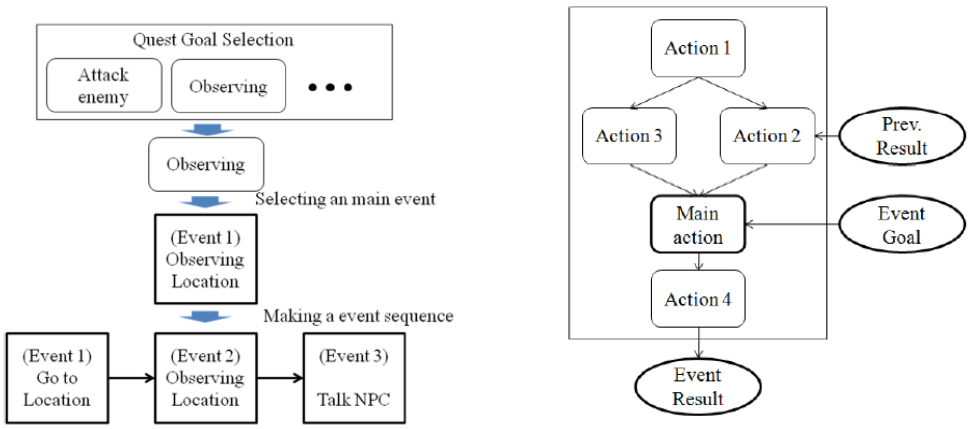
$$Quest = \{Event_1, Event_2, ..., Event_n\}$$

Figure 7: Quest Definition in Planning (Young-Seol Lee and Sung-Bae Cho 2011)

The first proposed approach on procedural quest generation is through planning. In this case with the help of the Petri Net System (Young-Seol Lee and Sung-Bae Cho 2011).

Quest through Planning Planning in the context of quests generation means to pick a sequence of events which leads to a specific goal. The player has a start state and will have a goal state at the end of a quest. The picked event sequence should provide a state transition from the initial state to the goal state. After the completion of each event the player's state has to be closer to the wanted goal state. If the player's state is not the same as the goal state, new events get inserted into the event sequence until this condition is met. The main idea of a quest in this approach is that it consists of different events. The completion of all this events through the player's actions leads to the completion of the quest. It leans closely on the concept of quests and their classification through actions from Ian Parberry and Doran Jonathan (Doran and Parberry, May, 2010).

Every event has some pre- and post-conditions. Every post condition can interpreted as a precondition of another action or as the quest goal itself. As the post conditions influence



(a) Quest Plot Generation **(b)** Conceptual Structure of an Event

Figure 8: Conception of quest generation through planning (Young-Seol Lee and Sung-Bae Cho 2011)

other events they are getting stored in a queue. They never get erased unless they are get consumed as a precondition for a different action. A few examples of an event would be "Go to Location", "Observing Location" or "Talk to NPC". In order to complete an event the player has to perform different actions. [Chapter quest]. One of these actions is picked as the main action, it is the main goal of the event. The main action also produces the post condition of the event. This is shown in Figure 8b. Figure 8a shows how a quest plot is generated through the selection of different related events and the selection of one Goal Event.

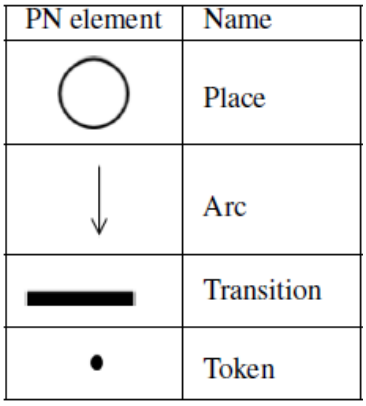
In the approach of Young-Seol Lee and Sung Bae Cho the Petri Net System is used for the planning of procedural quests. Petri Net is basically a network of different elements interacting with each other. The standard Petri Net elements which are used to represent an event are described in Figure 9a. Young-Seol Lee and Sung Bae Cho modified the standard Petri Net to perform well with RPG quests. The modified elements are shown in Figure 9b.

The *Item Node* stores an item name and a number of tokens. This number represents the amount of times the item has to be collected by the player. The *Conversation with NPC Node* stores the result of meeting up with an NPC and is often used as precondition for further events. The token shows how often the player has to have a conversation with the NPC. The *Location Node* is used to display the necessity of the player being in a specific place to receive or complete a quest. It can be used as a precondition for an action or to store it for later use as post condition. The token shows the players current location.

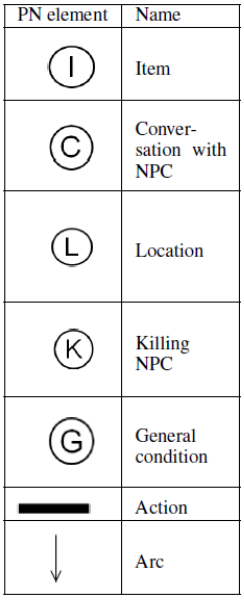
To store that the player killed a NPC the *Killing NPC Node* is used. This can as well be a precondition for another event or a post conditions of one. If it is a unique NPC the token is 0 or 1, for dead or alive.

In the *General Condition Node* every condition is stored which is none of the above.

Example



(a) Basic Petri Net Elements



(b) Modified Petri Net Elements

Figure 9: Petri Net Elements (Young-Seol Lee and Sung-Bae Cho 2011)

The example in Figure 10 shows how the event sequence for a simple collect quest could look like. Also it gives a closer look at Event 3 of the event sequence from the quest. The event "Collect N of Item A" is shown via the extended Petri Net elements. From a previous conversation (precondition) the player knows that he has to collect 10 (N) pieces of the specific Item A. The player performs the action of collecting that item until the token number of ITEM node is the same as N. Then the event is positively completed and generates and stores the post condition of 10 collected Items A. This post condition is stored and probably later used to change the items (as precondition) against a reward.

3.2.2 Believable Social Agents

This chapter explains the usage of BSAs to create believable quests in a procedural way. Secondly it discusses different examples how the basic system could be modified. Lastly the chapter analyses the restrictions of this approach.

Agents

AI and Agents are used for a wide variety of things in games. For example the behaviour of single enemies or the organisation and management of whole teams of enemies. Or the behaviour of friendly NPCs, managing their whole actions, daily life cycles and more.
In this case a special form of Agents is used, named Believable Social Agents (Grey and Bryson 2011). This Agents are producing procedural side quests by reacting upon local events and local knowledge of historic events. This restriction of knowledge does not only increase believability but also works against the so called "The Guy in the Street Rule". This cliché is about that is does not matter where in the world you did something,

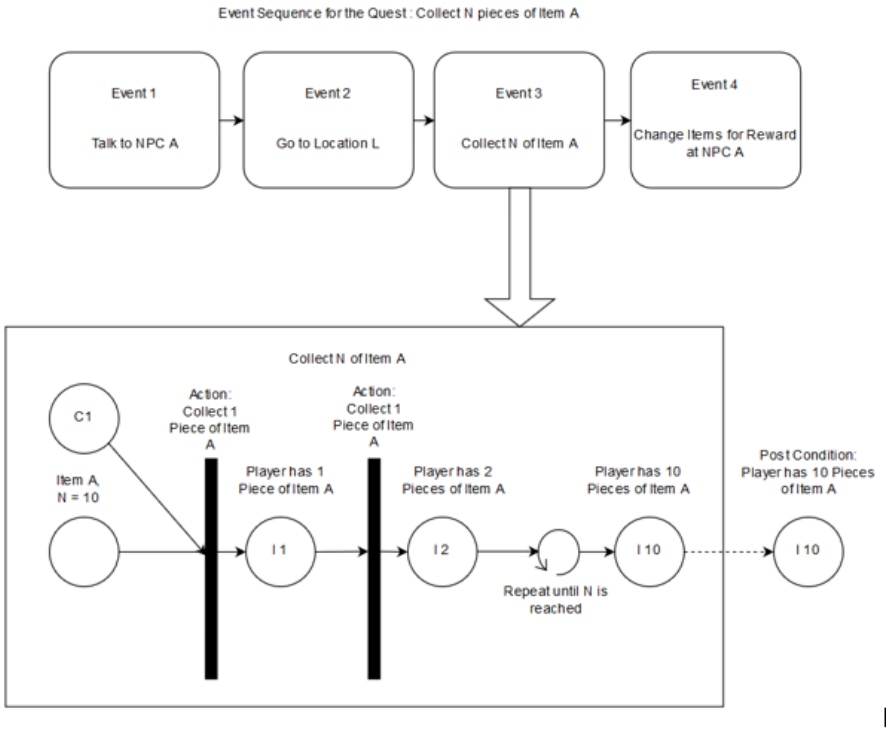


Figure 10: Petri Net Example: Collect Quest

immediately all NPCs knows about that. This behaviour is, in the most cases, unrealistic not very believable.
The main concern about the use of Agents in the area of narrative content is to ensure a coherent story and a reaction to the player's behaviour in a believable way (Riedl and Stern 2006).

Procedural Quests through Believable Social Agents
The approach of John Grey and Joanna Bryson to generate and implement procedural quests is to create whole believable NPC social networks through Social Agents and their interaction with each other (Grey and Bryson 2011). This emerges a society which the player can explore and interact with. Each NPC in this society represents an Agent. This Agent has an individual perception, memory and conversational ability. It stores its own set of priorities, memories and feelings towards other NPCs and the player.
Also the NPC gets influenced by local events and historic knowledge. This is a part of things which are stored as memories.
Based on the feelings, experienced events and actions of the NPC, the Agent creates individual and believable quests.

The Agent uses a hierarchical behaviour oriented design (BOD) to implement its behaviour. Under BOD Prioritised, Ordered, Slip-stack Hierarchical (POSH) dynamic plans are used. POSH is very similar to behaviour trees and provides a clean and understandable structure for the Agents behaviour.

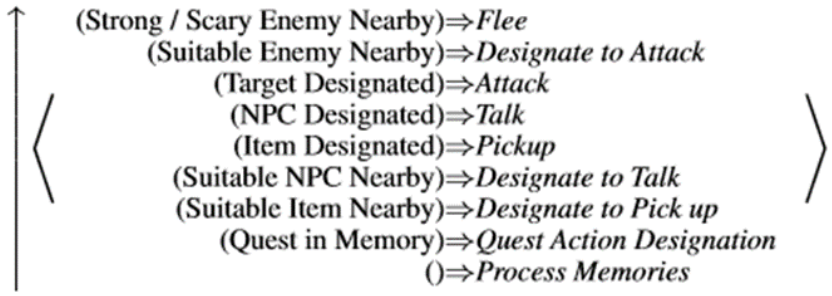


Figure 11: Agent Behaviour Drive Collection (Grey and Bryson 2011)

The Drive Collection (Figure 11) gives an overview of which behaviour the Agent is capable of. On the left side in the brackets the prerequisite for the action on the right side is defined.
Each Agent has the same Drive Collection. To make unique characters rather the emotions, perceptions and memories of the Agents gets altered than the basic set of behaviours.

Every NPC or Agent respectively stores **Emotions** towards other NPCs. In the current paper these feelings are Fear, Like and Hate. These are very simple feelings. The approach with BSAs could get developed further to also include more complex feelings like envy or love.
The base value of these feelings could be randomly set during the creation of the game world or get tweaked by a developer. To give the NPCs a base value of feelings at the beginning of the game leads to the fact that the Agents can already produce quests based on those feelings without the need of any experienced events and memories of them.
Through events or memories respectively the feelings of one NPC to another gets influenced. This leads to believable quests as they are based on the memory and feelings of the NPC. As already mentioned in Chapter 2.3.1, Quests are a way for NPCs to fulfil their needs and desires.

Each Agents stores its **Memory** in Memory Primitives (MP). These primitives are storing information about events the NPC has encountered. Each MP stores information about the actions the Agent has seen and the NPCs and items which were involved in this action.
To simulate an episodic memory, each memory is first stored in a short term memory. Every piece of memory in the short term memory is stored in a stack. This stack gets processed through the iteration over the behaviour tree. A stack is a useful data structure so recent events get processed sooner.
If a MP from the STM gets processed, the first thing to happen is that the feeling values this memory influences are getting adjusted. In which way the influence takes place is based on a general ruleset. For example if Bob sees that Eve kills Alice, and Bob liked Alice, the hate value towards Eve increases. Further if it is an important memory, which is evaluated by the type of action that happened it gets processed and added to the long

term memory.
Each MP in the long term memory has also a Decay Value. Over time this value gets decreased and when it has reached zero the memory gets erased. But if the memory gets accessed before that, for example to generate a quest or in the context of a dialogue, the decay value gets increased. This behaviour leads to the fact that important memories, like if someone killed another, are stored longer than trivial memories, like someone picked up an apple.

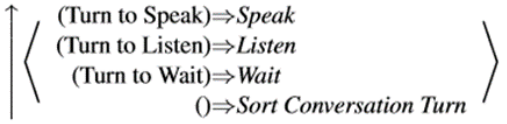
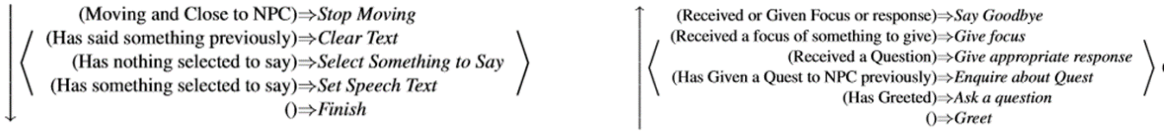


Figure 12: Conversation Cycle (Grey and Bryson 2011)



(a) Speak Drive Collection **(b)** Set Speech Text Drive Collection

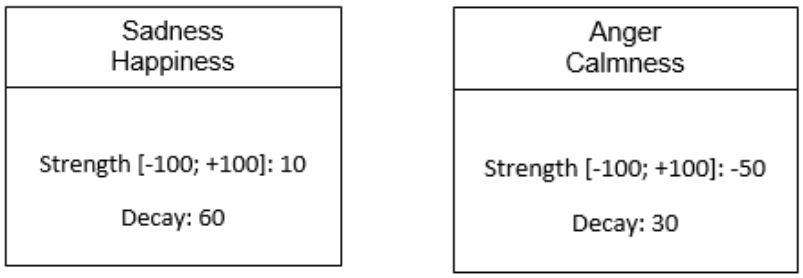
Figure 13: Speak Behaviour (Grey and Bryson 2011)

As each NPC has to be able to tell the player about his quests, so the NPC needs the ability to communicate. For this reason the Agent gets an own **Conversation Ability**. The process of the conversation is shown in Figure 12.
The conversation takes place through the passing of conversation primitives (CP) and translating the content of the CPs into human understandable text. The translation from CPs into normal text depends on the type of the CP (e.g. Greeting, Farewell).
Based on this type a suitable string from a pool of predefined template strings gets received. This string can contain variables as placeholder for runtime dependent information, like the NPC name or the player name. During the conversion this variables are getting replaced by their proper value.
An example for such a template would be: "Greetings, *Playername*!"
CPs are handled as a subclass of MPs. They can access MPs and other CPs to be able to refer to them. This gives them the ability to use other Agents, items, and events that happened earlier in their conversation.
During the Listen action the CPs are passed via a pull method from the speaking Agent to the listening Agent. The Speak action implements then its own Drive Collection (Figure 13a).
And in the Set Speech Text action in Figure 13a the CPs are translated into understandable text. What type of conversation possibilities the Agent is capable of to use is determined in a separate Drive Collection (Figure 13b). In the current state these possibilities are rather limited.

Future Possibilities
There are a lot of different possibilities to expand the complexity of this approach of procedural quest generation.

The most obvious way to improve the system is to consider all **quest types** mentioned in Chapter 2.3.1. This opens a wide variety for the Agents to interact with each other and the player. To only use kill and fetch quests as in the current version of the algorithm could be, on long sight, be really boring for the player.
But beside to include more quests types the Agent itself could be improved in many possible ways.

Furthermore by now only a very basic set of emotions are considered. These could be expanded to include a variety of **different emotions** like love, envy, anxiety or pity.



(a) Mood Primitive 1 **(b)** Mood Primitive 2

Figure 14: Mood Primitives

An other option would be that **moods** could be implemented. So every Agent could have a mood which is based upon recent events that happened in his life. For example if somebody who they liked died recently, they would get into a sorrowful mood. Or if they achieved something great, like Alice gets Jack killed, her Agent could get into a very happy mood. Each event indicates which mood is triggered and how strong that one is e.g. to lose a necklace is not so bad as if a family member dies.
The concept is that the data structure of moods are very similar to MPs, in fact can be derived from it, building a data structure of mood primitives. Each mood primitive stands for two counterparts of moods. For example, Sadness and Happiness or Anger and Calmness like in Figure 14. The reason behind this decision is that one Agent must never have two opposite parts of moods, he cannot be sad and happy at the same time. For this reason each mood primitive has a strength value between -100 and + 100. The minus value stands for the negative mood, like sadness, and the positive value for the positive mood, like happiness.
If a lot of events happen which are causing the same mood, the strength of that mood gets increased. For example if the Agent is already sad, which means that the strength value of the appropriate mood primitive has a minus value, and something terrible for the Agent happens again and again it gets increase with each event till it reaches the -100

limit.
If the strength of a mood reaches one of the limit values, it could cause extreme behaviours. To take the previous example, at sadness level of -100 the NPC could get suicidal. This extreme behaviour could also lead to new quests.
If one NPC gets suicidal, another who likes that NPC could give the player the quest to cheer him up or save him somehow. Another example would be that a psychotic NPC, who hates another wants him to get suicidal and hires the player to achieve this.
The mood primitive itself would have a Decay Value similar to the Decay of the memories and would be erased after some time, except if an event happens which would also cause that mood then the decay value could again be increased.

Beside moods and emotions, also different kinds of **personalities** could play an important role. Based on the personality of an Agent the encountered events could have different influences on the emotions and moods of the NPC. For example a very friendly, caring NPC could get a plus in the increase of friendliness towards the player if he does something good to someone the NPC cares about.
Also the selection of quests types could differ based on the personality. A very friendly and calm person would probably not give the player any task where he has to harm other NPCs.

Restrictions
The biggest restriction of the approach that each NPC is represented through an Agent is that no overall, coherent storyline is manageable. Because there is no manager above all other Agents who take care of them and spins a bigger storyline between them.
So this approach is really only usable for side quests or an open world without any bigger storyline at all, which is only made to explore the world and do some tasks during that time.

3.3 Procedural Dialogues

Another important part which has to get attention in the procedural domain beside the quest itself is the creation of dialogues. Good dialogues are an essential part in RPGs as they define the interaction between the player and the NPC and so with the game itself. Get each dialogue scripted by a game designer is not really an option in the procedural domain. A dynamic approach on dialogue generation is proposed by Gary Kacmarcik (Kacmarcik 2005). The proposed Question Answering (QA) System is not explicit made for procedural quests, but with a little adaptation this system could fit very well into it.

Question Answering in Role-playing Games
A different approach to dialogue generation than the already proposed ones in the previous sections is the Question Answering approach by Gary Kacmarcik (Kacmarcik 2005). It is a more dynamic, further developed and evaluable approach than the already used ones.
The Question Answering (QA) System fits surprisingly well in the previous mentioned approaches of procedural quest generation, especially in the Agent-based System.

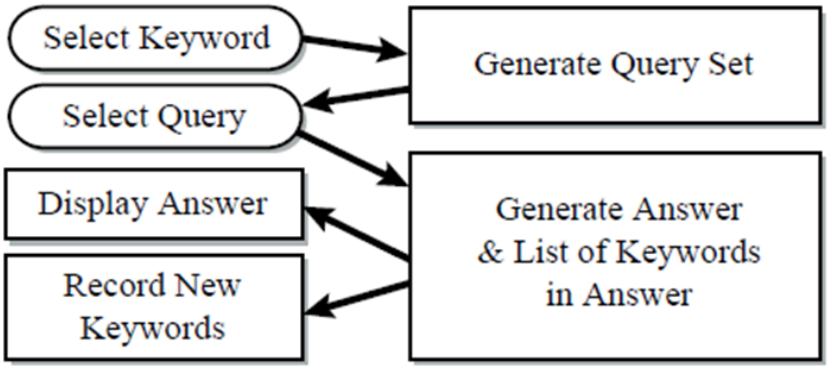


Figure 15: Question Answering System in RPGs (Kacmarcik 2005)

The core of this QA system is the use of keyword, key queries and non-key queries. Key-queries always producing new keywords and the non-key queries are used for the exchange of trivial information (e.g. greetings).
There is no keyboard input the player can give, as this would be too error prone, instead the player chooses his dialogue options based on the keywords. With this keywords, interesting queries based on the NPCs knowledge-base will be automatically generated.
The dialogue starts with the player selecting a keyword about he wants to know more. For this keyword questions are generated based on the NPCs knowledge-base. So the NPC can only give information on things he knows about and produced only queries he is able to answer. From this queries the player selects one he wants the NPC to answer. This queries can reach from "What/Who is X", where x is the keyword, to more complex queries.
Now the NPC answers the query which contains further keywords, e.g. a location or item, and displays the answer in the UI. The player, who has now also knowledge of those keywords, can select them in the next cycles of the conversation or in the interaction with other NPCs.

As the QA works with a knowledge-base each NPC has, this works very well with the Agent-based approach where every NPC has a set of MPs. We can use them to define their knowledge-base which is accessed during a conversation.
It would be possible to modify the MPs of the Agent to store not only involved NPCs and items, but also associated keywords. This opens up the possibility that a NPC also can forget about things during the runtime of the game.
In the planning approach with Petri Net this knowledge-base has to be predefined by a game designer and adjusted during the runtime of the game.
It is also possible for the NPC to withhold information or lie if that is consistent with his personality. Again at this point this QA System works perfectly well with the Agent-based approach as one of the proposed modifications there is the use of different personalities.

4 Evaluation of Quality

The evaluation of a procedural method is important to ensure usable content (Smith and Mateas 2011). Based on the evaluation it can be decided if the algorithms already provide good content or if it has to be adjusted. The following chapter gives an overview on a few criteria for quest and dialogue evaluation.

4.1 Quest Evaluation

The core requirement of a quest is that it has to be completable by the player at some point in the game. If the player has never the chance to complete the quest then it is just wasted time and frustrating. It is acceptable and also desired that the player may have to overcome obstacles in order to complete a quests, or that it is only possible to complete in a later stage of the game.
Further a quest has to make sense in the current context and has to be believable. Believability is probably the most important property of a good quest. If a quest seems not reasonable to the player it lowers the immersion of the game extremely (Grey and Bryson 2011).
The only way to test this requirements is through play-testing at different stages in the game. Testing each quest is not a possible option because those are procedural generated. The probability that two players encounter the same quests is rather low.
By testing different quests at different points in the game a statistical value can be calculated which determines the probability of unsolvable quests. Each game developer has to choose by himself when this value is low enough. Zero percent would be the best case but if this can be reached is questionable.
As mentioned in Chapter 1.1 every procedural algorithm has to be tested and adjusted over and over again until an acceptable outcome has be reached. Another possibility in testing the system would be if it uses the proposed QA System with keywords. With that, the system could be evaluated at different stages of the game by proving if every keyword needed to solve the game is obtainable at some point in the game.

4.2 Dialogue Evaluation

The two core factors in evaluating dialogues are the speed and fluency of the conversation (Kacmarcik 2005). The speed factor is essential as if the response of the NPC takes too long it can be very frustrating for the player. Testing the speed is a simple task by measuring the time the NPC needs to generate an answer. This can be done in the same time quests are getting tested.
The fluency has to be maintained so that the immersion level of the game does not take any harm through edgy, incomprehensible and unnatural sounding conversations. To evaluate that factor the comparison with an n-gram language model can be used.

5 Conclusion

This thesis has given an overview over two different approaches on how to achieve procedural quest generation in RPGs. Further it took a look at one possible way to produce dialogues in the procedural domain. But before that it gave an extensive overview on what quest in a role-playing context are and through which actions they are built up. Furthermore it described a few quest clichés which have emerged in the role-playing genre over time. Lastly this paper described which factors in this field of procedural generation has to be monitored and evaluated.

After the comparison of two proposed ways to generate quests, the Agent-based approach seems as the more promising one. It bares more possibilities of getting developed further and modified in many different ways. It can be used from a very simple lightweight system to a very complex one. Also a big factor which speaks for this approach is that AI gets developed very fast and especially Agent-based algorithms are getting a lot of attention in the recent time. The planning approach provides not so many ways to modify it and it also does not take different kinds of NPCs into account. Further the quests in this approach are not based on some kind of reasonable motivation for the NPC. This makes it hard for the player to comprehend those quests and probably reduces the level of immersion.
Regarding the evaluation of this procedural systems, play-testing seems to be the best option but is very time consuming. Better evaluation systems still need to be developed.
Finally it seems that procedural quest generations system are ready to be used in the game industry to produce a lot of side quests. If the system is tweaked and adjusted well, the saving in time of generating quests is huge.
Especially as nearly nobody produces only one game any more but rather a whole series of it and the system has only to be adjusted once and can then produce nearly infinite content. Still the main quest line, if there is any, should be designed by game designer.

Acronyms

RPG Role-Playing Game

PCG Procedural Content Generation

NPC Non-Player Character

AI Artificial Intelligence

MMORPG Massive Multiplayer Online Role-playing Game

BSA Believable Social Agent

BOD Behaviour Oriented Design

POSH Prioritised, Ordered, Slip-stack Hierarchical

MP Memory Primtive

STM Short Term Memory

CP Conversation Primitve

QA Question Answering

List of Figures

References

Ashmore, Calvin, and Michael Nitsche, editors. 2007. *The Quest in a Generated World.* DiGRA - Proceedings of the 2007 DiGRA International Conference: Situated Play. The University of Tokyo. ISBN: ISSN 2342-9666. http://www.digra.org/wp-content/uploads/digital-library/07311.20228.pdf.

Clarke, Fabrice, editor. 2014. *Creating Dynamic Role-Playing Game Experiences: Getting off the Railroad.* Southampton, England: Interactive Multimedia Conference 2014. http://mms.ecs.soton.ac.uk/2014/papers/21.pdf.

Corner, The Desktop. 2016. *Mission to Halm: Spelunking the Jedi Way.* http://yodastories.tumblr.com/. Accessed: May 29 2016.

Doran, Jonathan, and Ian Parberry. May, 2010. "Towards Procedural Quest Generation: A Structural Analysis of RPG Quests". Technical Report LARC-2010-02, University of North Texas. https://larc.unt.edu/techreports/LARC-2010-02.pdf.

Grey, John, and Joanna Bryson, editors. 2011. *Procedural Quests: A Focus for Agent Interaction in Role-Playing-Games.* University of York, UK. https://www.cs.bath.ac.uk/~jjb/ftp/GreyAISB11.pdf.

Hendrikx, Mark, Sebastiaan Meijer, Joeri van der Velden, and Alexandru Iosup. 2013. "Procedural content generation for games: A survey". *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9 (1): 1.

Howard, Sean. 2016. *PCG: Puzzle Tree Building.* http://www.squidi.net/three/entry.php?id=159. Accessed: May 29 2016.

Kacmarcik, Gary, editor. 2005. *Question-Answering in Role-Playing Games.* Question Answering in Restricted Domain: Papers from the AAAI Workshop. Menlo Park, California: AAAI Press. http://aaaipress.org/Papers/Workshops/2005/WS-05-10/WS05-10-009.pdf.

Kerr, Christopher, and Duane Szafron, editors. 2009. *Supporting Dialogue Generation for Story-Based Games.* Menlo Park, California: AAAI Press. ISBN: 978-1-57735-431-4. http://www.aaai.org/ocs/index.php/AIIDE/AIIDE09/paper/viewFile/783/1059.

Lee, Young-Seol, and Sung-Bae Cho. 2012. "Dynamic quest plot generation using Petri net planning". In *Proceedings of the Workshop at SIGGRAPH Asia,* 47–52.

Nitsche, Michael, Calvin Ashmore, Will Hankinson, Rob Fitzpatrick, John Kelly, and Kurt Margenau. 2006. "Designing procedural game spaces: A case study". *Proceedings of FuturePlay* 2006.

PhiGames. 2016. *TinyKeep Dungeon Generation Demo.* http://tinykeep.com/dungen/. Accessed: May 29 2016.

Riedl, Mark O., and Andrew Stern. 2006. "Believable agents and intelligent story adaptation for interactive storytelling". In *Technologies for Interactive Digital Storytelling and Entertainment,* 1–12. Springer.

Smith, Adam M., and Michael Mateas. 2011. "Answer set programming for procedural content generation: A design space approach". *Computational Intelligence and AI in Games, IEEE Transactions on* 3 (3): 187–200. ISSN: 1943-068X.

Smith, Gillian, Ryan Anderson, Brian Kopleck, Zach Lindblad, Lauren Scott, Adam Wardell, Jim Whitehead, and Michael Mateas. 2011. "Situating quests: Design patterns for quest and level design in role-playing games". In *Interactive Storytelling,* 326–329. Springer.

Trenton, Marcus, Duane Szafron, John Friesen, and Curtis Onuczko, editors. 2010. *Quest Patterns for Story-Based Computer Games.* Menlo Park, California: AAAI Press. `https://webdocs.cs.ualberta.ca/~duane/publications/pdf/2010aiideMT.pdf`.

Young-Seol Lee and Sung-Bae Cho. 2011. "Context-Aware Petri Net for Dynamic Procedural Content Generation in Role-Playing Game". *Computational Intelligence Magazine, IEEE* 6 (2): 16–25. ISSN: 1556-603X. doi:`10.1109/MCI.2011.940618`.