StoryWorld: Procedural Quest Generation Rooted in Variety & Believability

Vincent L. Prins
v.l.prins@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, The Netherlands

Jelmer Prins
j.prins.2@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, The Netherlands

Mike Preuss
m.preuss@liacs.leidenuniv.nl
LIACS, Leiden University
Leiden, The Netherlands

Marcello A. Gómez-Maureira
m.a.gomezmaureira@utwente.nl
HMI, University of Twente
Enschede, The Netherlands

ABSTRACT

Procedural Content Generation (PCG) in games has become more common in recent years. Procedural narrative, however, is still tricky, as designers fear that applying PCG to narrative means that they lose control over the system. In this paper, we attempt to tackle procedural generation of quests, a specific type of narrative, like in the Role Playing Games genre. To this end, we present StoryWorld, an *Event*-driven simulation of *Items*, *Locations*, *Characters*, and their *Memories* and *Desires*. Its modularity makes it easy to add new items and events. Future work could see its memory system made more realistic.

CCS CONCEPTS

• **Applied computing** → **Computer games**; • **Software and its engineering** → *Interactive games.*

KEYWORDS

procedural content generation, procedural quests, narrative, RPG, game AI

ACM Reference Format:
Vincent L. Prins, Jelmer Prins, Mike Preuss, and Marcello A. Gómez-Maureira. 2023. StoryWorld: Procedural Quest Generation Rooted in Variety & Believability. In *Foundations of Digital Games 2023 (FDG 2023), April 12–14, 2023, Lisbon, Portugal*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3582437.3587181

1 INTRODUCTION

Procedural Content Generation (PCG) for games provides a way to generate game content programmatically and in a semi-randomised fashion, often guided by a designer [9]. This allows for the creation of a large amount of content that can take a multitude of variables into consideration. Games like Dwarf Fortress [1] and Ultima Ratio Regum [6] show that it is possible to generate complex, interactive game systems and environments with PCG. These games feature procedurally created terrains, histories, civilisations, characters, languages, poetry, dance forms and more. Narrative is also among the target areas of PCG [11]. In games, narratives are often presented as a quest: a task, or series of tasks, to complete a goal.

While PCG is capable of producing a large amount of content, players might not consider the variety in the content sufficiently novel or interesting. Compton calls this the "10,000 bowls of oatmeal" problem [4]: technically, the bowls might have a slightly different amount of oat, different sizes and shapes, but to the user it is all oatmeal. Another challenge for PCG in narrative-based games is that control over a game's narrative is given away to a rule-based system that may fail to match the intended overall experience [10]. On the other hand, procedural quests can add meaning and context to a procedurally generated world, as well as provide flexibility, because they can be dynamically generated during gameplay [3].

Here, we present such a system called StoryWorld. It is capable of generating procedural quests and acts as a logical foundation for why these quests are important to the world and how completing or failing them will permanently affect the world. This leads to logical interconnectedness between quests and the game world which the player might experience as a more dynamic and "alive" world. We argue that this leads to a more believable and thus more immersive world. StoryWorld combines various parts of previous work in the field. We specifically add the concept of *Desires*, as we believe it is crucial for the sake of a *believable* narrative system, which we define as responding sensibly to players and maintaining coherency.

2 RELATED WORK

Several approaches towards procedural narrative, and by extension, quests have been suggested. Here, we cover some of them.

The CONAN system uses a world state to generate quests [2]. The world state is designer-specified, consisting of locations, items, and non-player characters (NPCs). NPCs have personality traits which are taken into account when CONAN generates a quest for an NPC, who then proposes it to the player. This ensures that peace-loving NPCs do not ask players to kill others. The quests themselves come from a designer-specified grammar, where literals can be replaced by NPCs, locations or items.

Grey et al. base their quest engine on *believable social agents* [5]. They consider these as having emotions, being able to perform actions, perceiving them being performed, having memories of these

FDG 2023, April 12–14, 2023, Lisbon, Portugal

ACM ISBN 978-1-4503-9855-8/23/04.
https://doi.org/10.1145/3582437.3587181

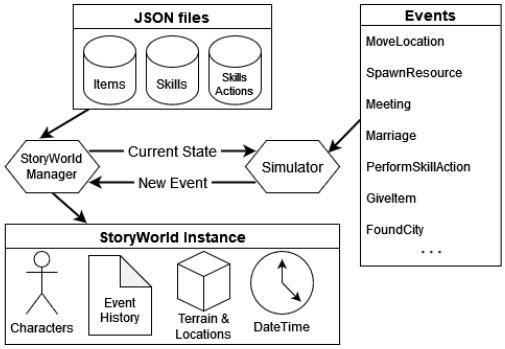

Figure 1: Overview of the architecture of StoryWorld.

actions, and spreading information through conversations. The memories and emotions of agents are there to provide believable reasoning for giving a quest. Memories may also leave a mark on the emotions of an agent, which stays even after the memory has been forgotten.

TRUE STORY is a goal-generating architecture that uses history and relationships developed through interaction between world objects and actions [7]. NPCs in TRUE STORY have memories, inter-NPC relationships, and attributes such as thievery skill. These are used as constraints for quest generation. Memories of an event may also differ between two agents participating in it (e.g. a robbery that went undetected; only the perpetrator knows who did it).

Warpefelt has presented the Relic game series [12]. It looks at the minimum amount of generated content that still produces believable narrative. It generates a story world in time steps, where at each step an event is generated. These events may include locations and characters. For the game, special items are scattered throughout a dungeon. These items make reference to the generated story world, which provides meaning and coherence. The generated content is produced with naive string substitutions.

Talk of the Town is a system driven by knowledge and memories that NPCs have and share [8]. Knowledge originates (e.g. observation), propagates (e.g. a statement in conversation), deteriorates (mutation) and terminates (is forgotten). NPCs can also lie or misremember things. Furthermore, NPCs have relationships with other NPCs, for example, liking or disliking someone. The result is a complex emergent social network.

3 STORYWORLD DESIGN

StoryWorld is a system for procedural narrative, that is driven by the occurrence of *Events* in an open-ended environment. In general, StoryWorld consists of *Characters* and *Items*, and systems to facilitate interaction between these two. Characters have *Memories*, *Skills*, *Relationships*, *Desires* and *Tasks*. All changes to the StoryWorld are done through *Events*. Every time a character moves or creates an item this is done through an event, and the involved characters may establish memories of said event. This allows for partial or incorrect information which is valuable for creating emergent narratives. *Events* are very open-ended and can be as simple as observing an item, to being more impactful such as the death of a character. Figure 1 summarises the system.

StoryWorld works in two phases. First, a 'StoryWorld instance' is generated, consisting of procedural terrain and history generation in which events are fired sequentially (e.g., characters being born, marrying, dying). In the second phase, a player can interact with the system and invoke their own events, with the system only creating events when an in-game day has passed. Thus, the system and players take turns in performing actions.

On top of StoryWorld we implement a visualisation layer in order to turn it into a playable game and to make the information in StoryWorld understandable for players. In the game, the player can interact with the world as a character who has the same abilities as the NPCs. The main focus for the player is accepting Tasks from the NPCs and completing them. The game most closely resembles a game from the fantasy-based RPG genre where characters have items and skills in an open world that allows for resource collection. The different parts of StoryWorld will be defined and explained in the following subsections.

3.1 Events

Events are a representation of everything that happens in StoryWorld. Anytime any change is made to anything, it happens through an event. *Characters* are then able to receive *Memories* from these events. All events are fired by a central system called the *Simulator*. Events have a definition for their conditions that need to be met before they can be fired. Marriage, for instance, requires two people to have a good *Relationship* before the event can occur. Even then, when the event occurs is determined probabilistically; certain events occur more often than others, as they have been programmed with a higher or lower chance of firing.

3.2 Terrain & Location Generation

StoryWorld procedurally creates terrain surrounded by water with different biome regions. Afterwards, during history generation, the map is also populated with *Locations*. These locations can be resources, cities or houses. Resources like trees take the biome of a region into account such that, for instance, they only spawn in regions designated as forests. During history generation, a region may be designated as a city, allowing NPCs to build houses.

The *MoveLocationEvent* causes *Characters* to move between locations. It is, for example, necessary for a character to first move to an iron ore resource before they can perform a mining action. This limitation makes it necessary for characters to have *Knowledge* of the location of a resource.

3.3 Characters

Characters are the main focus of StoryWorld. They form *Knowledge* of the world around them based on *Memories* that are created in *Events*. They will then act based on these memories and their *Desires*. The most relevant properties are:

- *Personal details:* Name, gender, birthday, favourite skill, and family connections (e.g. parents, spouses).
- *Inventory:* Characters are able to hold arbitrary amounts of *Items* and use these for various different *Skills*.
- *Memories:* Memories are generated through events and drive many systems like skill XP, *Relationships* and *Knowledge*.
- *Skills:* List of skills and their current level and amount of XP.

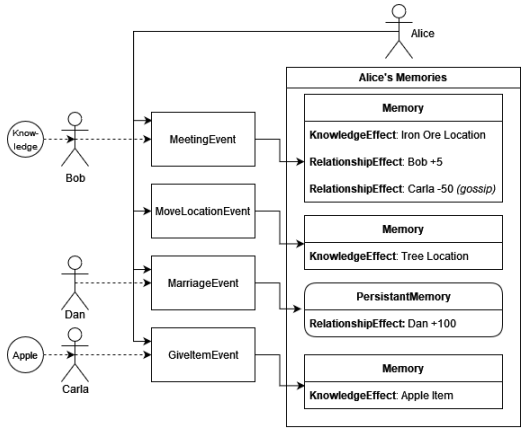

Figure 2: Example of experiencing Events and resulting Memories applying Effects.

- *Relationships:* List of relationships, where each refers to another character. It contains the relationship value (a signed integer) and each *RelationshipEffect* that affects this, allowing characters to recall why they like or dislike someone.
- *Desires:* Something a character wants to achieve, like levelling up a *Skill*, obtaining an item, or building a house.
- *Knowledge:* This is split in *ItemKnowledge* and *LocationKnowledge*. These are contained in memories based on the events where these items or locations were observed.

3.4 Memories & Knowledge

Each *Character* has a list of *Memories*. All memories are based on the *Events* a character participated in. An example scenario is shown in Figure 2. Note that these memories do not need to contain information about everything that happened in the event, and can contain partial or incorrect information. Memories can be seen as the carriers of *Knowledge*, *Skill* experience points (XP), and *Relationships*. *Knowledge* can be information about the *Location* of an object (a house or resource node), or information about the location and attributes of an *Item*.

Memories are either transient or persistent. Transient memories can be forgotten when the list of memories reaches a predefined threshold, starting with the oldest memory. Persistent memories focus on remembering information that should not be forgotten, such as marriage and the *Relationship* bonus that comes with marriage. Note that, even though the memory of how XP is gained can be forgotten, XP itself is not lost.

During a *MeetingEvent*, characters in the same location share knowledge between themselves: e.g. the location of a resource. *MeetingEvents* also create memories about relationships. In most cases, this is strengthening the relationship. Furthermore, the system implements a chance that gossip is spread about a random character, which negatively impacts how characters in the meeting view the targeted character. This is temporary, due to memory loss.

3.5 Items

Items are part of interactions and are used to train *Skills* or build houses. They are defined by a location, name, stackability and an arbitrary number of attributes such as monetary value and weight.

3.6 Skills & SkillActions

Characters have *Skills* which are needed for many actions in StoryWorld. Skills mostly focus on resource gathering and processing, such as fishing or smithing. XP, and thus levels, can be gained by performing *SkillActions*. These actions have skill level, item and tool requirements. All these requirements can contain multiple objects, so actions can take multiple different skills, items and tools to complete. When all requirements are met and a *SkillAction* is performed, the required items are removed, but tools are kept. *SkillActions* further define how much XP is gained in skills and which items are received upon completion. These can only be performed at specific locations (e.g. at an anvil for smithing).

An example of a *SkillAction* would be catching a shrimp which requires level 1 fishing and a fishing net. When performed, the character acquires a raw shrimp and 10 fishing XP. This can only be performed at a fishing dock. As all skills and *SkillActions* are defined in a JSON format, they can easily be expanded.

3.7 Desires & Tasks

Events can induce *Desires* in a *Character*. *Desires* are taken into account when the *Simulator* chooses a new event. For instance, for a character who has the desire to obtain iron ore, the simulator attempts to choose an event to let this character harvest an iron ore resource. The characters still need to meet the *SkillAction* conditions, and have knowledge of the location of such a resource.

Figure 3 shows events that induce desires. For instance, a *MarriageEvent* induces an *ItemDesire* for a diamond ring. The other desires are *HouseDesire* and *SkillDesire*. These are more complex, as they themselves can spawn new desires. The implication is that the parent desire (e.g. building a house) can only be fulfilled when the child desire (an *ItemDesire* for wood as building material) is fulfilled. *ItemDesires* may also spawn child desires. For instance, gathering iron requires a pickaxe, so this desire is also created.

A character can also give out the *Task* of fulfilling its own desires to another character. A task is completed once the associated desire is fulfilled. Upon completion, task-completer is rewarded an amount of gold coins based on the task difficulty. This system hinges on characters generating desires and being unable to fulfil these themselves. Two characters will, however, only give and complete tasks for each other if they have a friendly *Relationship*.

The main example of such a desire is the *SkillDesire*: the desire to increase the level of a *Skill*. During simulation, characters have a chance to receive it for any skill, but with an increased chance for their favourite skill. This desire calculates which *SkillActions* are needed to reach the desired level. If the character is missing *Items* for these *SkillActions*, there will be an opportunity for another character to take on the gathering of the missing items as a task. This will be completed once the original character receives the items, which then enables them to fulfil their *SkillDesire* by performing the *SkillActions* that were needed to level up.

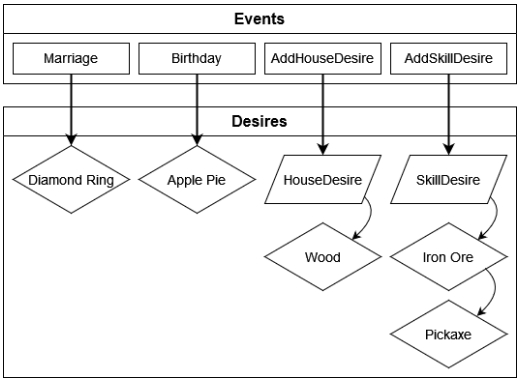

Figure 3: Example of Events inducing Desires. ItemDesires are diamond-shaped, other desires are parallelograms, Events are rectangles.

The system will be able to generate more diverse quests simply by defining more (interesting) skills and *SkillActions*. These will automatically be chosen as subject for a desire. The completion of these desires also has a lasting effect on the world, since the character will have higher skills. This means that next time such a desire is generated, it will require more and higher-level resources to complete. The increased skill level can also have more profound consequences, like this character now being able to complete tasks for other characters using their new skill level.

3.8 Implementation

The visualisation of StoryWorld relies on the Unity game engine and is written in C#. StoryWorld itself is also written in C#, but does not make use of Unity code and compiles to its own assembly. Thus, it can be used in other frameworks.

A Unity project containing both StoryWorld and the visualisation is available as an OSF repository[1]. As an early prototype, it does not showcase the full capabilities of StoryWorld, as player-NPC interactions are not part of it.

4 CONCLUSION

In this paper, we present StoryWorld, a system which implements procedural content generation (PCG), in order to create a dynamic world of non-player characters (NPCs) for players to interact with via procedural quests. It implements RPG-like systems for skills and items and uses these systems when generating quests.

In StoryWorld, a procedural event history is generated for every world. Events range from birth, marriage and death to giving items to others, building houses and more. These events may induce desires, and even desires can induce other desires. In levelling up skills, new items come available which adds to the variety. Human players can, therefore, expect different tasks from high and low-level NPCs. Another aspect of variety is procedural character personalities. Currently, this is limited to characters having a favourite skill. This could be expanded upon in future work. Other future work includes gossip based on character knowledge, memory loss more realistic than first-in-first-out, and support for external terrain.

The design of StoryWorld is motivated by believability. This is accomplished by having tasks bound to desires, which in turn are induced by events. Tasks are further dependent on preceding events between characters that establish a relationship in which a task is contextualised.

The memory and knowledge systems also add to the believability. If a character wants to collect iron ore, they need an active memory containing knowledge of the resource location. Knowledge can be spread between characters during meetings and is lost when the associated memory is forgotten. Memory loss also accounts for forgetting about interactions that influence relationships. It is, for example, possible for a character to hear gossip about someone, temporarily have a bad relationship with them, and then forget about it again, improving the relationship. Memories can also be marked as long-term so as to not be forgotten, such as marriage.

Lastly, having the human player and the computer simulation take turns, makes for a reactive system that can make a player feel like they contributed meaningfully. For instance, if a player completes a task delivering wood to an NPC, the next day, a house may have appeared, as the NPC has used the given items to build it. The house, then, is a permanent reminder that the player contributed to the world. Additionally, the player may notice that an NPC who used to ask for low-level items, starts asking for higher-level items because they were able to level up a skill thanks to the player providing items through tasks.

In conclusion, StoryWorld presents an event-based framework for quest generation that can provide variety, and is rooted in believability.

[1] https://osf.io/r7dhv/?view_only=67157f8349f74480819652e99621d93d

ACKNOWLEDGMENTS

We would like to thank Wouter Ebing and Tijn Huiskens for their input during design meetings.

REFERENCES

[1] Tarn Adams and Zach Adams. 2023. Dwarf Fortress. https://www.bay12games.com/dwarves/. Bay 12 Games. Accessed: 2023-01-09.
[2] Vincent Breault, Sebastien Ouellet, and Jim Davies. 2021. Let CONAN tell you a story: Procedural quest generation. *Entertainment Computing* 38 (2021), 100422.
[3] Yun-Gyung Cheong, Mark O Riedl, Byung-Chull Bae, and Mark J Nelson. 2016. Planning with applications to quests and story. See [9], p. 124.
[4] Kate Compton. 2019. Getting Started with Generators. See [11], p. 15.
[5] John Grey and Joanna J. Bryson. 2011. Procedural quests: A focus for agent interaction in role-playing-games. In *Proceedings of the AISB 2011 Symposium: AI & Games*. 3–10.
[6] Mark R. Johnson. 2023. Ultima Ratio Regum. https://www.markrjohnsongames.com/games/ultima-ratio-regum/. Accessed: 2023-01-09.
[7] James Pita, Brian Magerko, and Scott Brodie. 2007. True story: dynamically generated, contextually linked quests in persistent systems. (11 2007), 145–151. https://doi.org/10.1145/1328202.1328228
[8] James Ryan, Adam Summerville, Michael Mateas, and Noah Wardrip-Fruin. 2015. Toward characters who observe, tell, misremember, and lie. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 11. 56–62.
[9] Noor Shaker, Julian Togelius, and Mark J Nelson. 2016. *Procedural content generation in games*. Springer.
[10] Tanya Short and Tarn Adams. 2017. *Procedural Generation in Game Design*. CRC Press.
[11] Tanya X. Short and Tarn Adams. 2019. *Procedural Storytelling in Game Design*. CRC Press.
[12] Henrik Warpefelt. 2022. Developing the Relic Series: Exploring the Effects of Remediating Naively Generated Narrative. In *Proceedings of the 17th International Conference on the Foundations of Digital Games*. 1–5.