LudoNarrare: A Model for Verb Based Interactive Storytelling

By

Joshua Stark

APPROVED:

Kurt VanLehn_____Director

Jon Wetzel_____Second Committee Member

Executive Summary

Instead of providing the illusion of agency to a reader via a tree or network of prewritten, branching paths, an interactive story should treat the reader as a player who has meaningful influence on the story. An interactive story can accomplish this task by giving the player a large toolset for expression in the plot. *LudoNarrare*, an engine for interactive storytelling, puts "verbs" in this toolset. Verbs are contextual choices of action given to agents in a story that result in narrative events.

This paper begins with an analysis and statement of the problem of creating interactive stories. From here, various attempts to solve this problem, ranging from commercial video games to academic research, are given a brief overview to give context to what paths have already been forged. With the background set, the model of interactive storytelling that the research behind *LudoNarrare* led to is exposed in detail. The section exploring this model contains explanations on what storyworlds are and how they are structured. It then discusses the way these storyworlds can be brought to life. The exposition on the *LudoNarrare* model finally wraps up by considering the way storyworlds created around this model can be designed. After the concepts of *LudoNarrare* are explored in the abstract, the story of the engine's research and development and the specifics of its software implementation are given. With *LudoNarrare* fully explained, the focus then turns to plans for evaluation of its quality in terms of entertainment value, robustness, and performance. To conclude, possible further paths of investigation for *LudoNarrare* and its model of interactive storytelling are proposed to inspire those who wish to continue in the spirit of the project.

The Problem and Research Question

For the past forty-five years, game designers and authors have passionately pursued the creation of interactive stories that could be mediated by a computer. The idea of playing the role of a character in the unfolding drama of a story has a strong appeal. However, even after many attempts, interactive storytelling has remained a difficult dream to achieve with any amount of satisfaction. Solutions such as narrative media between gameplay, branching structures, illusion of choice, and environmental storytelling, while all providing interesting experiences, have failed to capture the true essence of what an interactive story could be. Even today, current commercial interactive storytelling solutions resort to prebaked plotlines which fail to give the player true, satisfying agency and rich play. From the above, the following question arises: What model can be used to better approach the dream of an interactive story?

Implementing this model on a computer presents a multilayered engineering challenge. Conceptually, a balance between designer, computer, and player must be struck. This maps to the creation of systems for interactive storytelling. The designer needs a tool or language to build an individual interactive storyworld. The computer needs an engine that uses many algorithms and artificial intelligences to maintain and bring life to the interactive storyworld the designer created. And finally, the player needs a way of interacting with the computer as it proceeds through the story. They need the ability to act in the story and experience its progression. To make interactive storytelling come to life, and thus be measureable, a further question must be put forth: What does a system and toolset for creating interactive stories that follow a robust and satisfying model look like? This question drives the research documented herein.

Literature Review

At the current moment, there are more talented people than ever before working away to try to solve the interactive storytelling problem. Many of them come from a background of video game development. Others are creators of literature, theatre, music, or film. Each offers up a unique lens through which to look for solutions. Even since I began research in this subject around a year ago a large amount of interactive storytelling works have been released. To mention every work and research project centered on interactive storytelling up until today would be impossible. Therefore, this review will cover only a sufficient selection of the highlights in both creations and research as well as the body of vocabulary built to describe interactive storytelling both outside and inside this paper.

Perhaps the oldest form of interactive storytelling for computers is interactive fiction. Interactive fiction, such as the genre's most well-known title, *Zork* (1977), provides the player with a completely text-based interface which they use to move throughout an environment solving puzzles. These games parse typed commands from players and return text describing the results of these actions. Interactive fiction can focus on having the player explore the network of text describing the storyworld, having the player solve complicated logic puzzles, or anything in-between. With modern tools like Inform 7 and Twine, a community of authors have refined the process of creating these types of works. The rich history of interactive fiction provides a lot of inspiration to draw from, but its text-based nature precludes it from a mainstream audience interested in visual storytelling and its focus on puzzles can make it less dramatic and plot driven.

Many of those who do work in interactive fiction, such as Emily Short, are using their knowledge and experience to try to solve the interactive storytelling problem.

At the same time interactive fiction came onto the scene, video games were quickly gaining ground as the hot new entertainment medium. Most early games only used story to have a premise that gave context to what the player was doing, such as defending Earth from aliens or rescuing a princess from a villain's castle, a practice still common today. With influence from the early 90's multimedia boom, games like *Myst* (1993) started to mix in video between moments of gameplay to make the context of play even more immersive. Role-playing games whose ancestry went back to *Dungeons & Dragons* (an excellent example of interactive storytelling without a computer) used branching story paths, dialog trees, cutscenes, and actions tied to growing character statistics to create some of the richer story experiences be found in video games. Point-and-click adventure games similar to interactive fiction but with visual interfaces also emphasized story. Starting with *Half-Life* (1998), first-person shooters started to use environmental storytelling and scripted events to tell stories. The independent game movement that arose around the year 2007 started to use the same techniques of the past with more personal and complicated subject matter outside the realm of strict fantasy and sci-fi. Some of these games also experimented with storytelling through proceduralism, or storytelling through the rules of the game itself. Jonathan Blow's *Braid* (2008) and Jason Rohrer's *Passage* (2007) exemplify this philosophy. Recent independent endeavors like *80 Days* (2014) and *Sunless Sea* (2015) are starting to make some major strides towards realizing interactive storytelling in a satisfying form.

Chris Crawford, one of the first video game developers, has created some of the more inspiring and fleshed out work on interactive storytelling. Having left the video game industry dissatisfied with its direction, Crawford set out to solve the interactive storytelling problem in the year 1992. In 2005, this work resulted in the book *Chris Crawford on Interactive Storytelling*, an important work for laying out the design philosophy, system models, and algorithms he had developed for over a decade. Crawford emphasized the concepts of verb thinking and human interpersonal drama. In his book, Crawford goes through the specifics of a personality model that can drive artificial intelligence agents to behave in human and dramatic ways. He also offers a thorough critique of previous attempts at interactive storytelling, in particular, those done by video game designers. The technology in the book became *Storytron* (2009), Crawford's interactive storytelling engine. Unfortunately, this project turned out a failure for Crawford. In postmortem writings, he seems to place particular emphasis on how the failings of the tool for creating *Storytron's* interactive stories led to difficulties for potential creators. If it is too frustrating to author storyworlds, none will exist. Crawford was particularly concerned with the ability of traditional authors with no technical skills to create storyworlds for *Storytron*. Despite this, *Storytron* still featured a scripting language called Sappho. Programmers found the language to be too limiting; nonprogrammers found it to be too challenging. He also found that his system for authoring verbs in *Storytron* was too complicated for most creators to handle. He suggested that instead authors should just work with a handful of generic verbs. *Storytron*'s interface, which made almost exclusive use of text, did not offer much flexibility and would often lead to unengaging or awkward results. Crawford would later

suggest and use a system of iconic sentences rather than text sentences to get around some of the issues text feedback presented. To this day, Crawford has continued to work on the interactive storytelling problem through his current project *Siboot*.

Outside commercial and hobbyist art, interactive storytelling has also appeared as an interest in academic research for a long time. The Oz Project at Carnegie Mellon University focused on utilizing artificial intelligence to drive the characters in an interactive storyworld and make sure that an appropriate level of drama was maintained throughout the story. One of the members on the Oz project, Michael Mateas, went on to do additional interactive storytelling research with Andrew Stern, resulting in *Façade* (2005). *Façade* is a highly influential interactive story that places the player in the role of visiting a married couple for drinks where, as the night continues, drama between the characters grows. The player interfaces with *Façade* through a first-person 3D environment, recorded dialogue audio, and natural language processing. Being incredibly ambitious, the results of *Façade* were mixed. Interfacing with agent characters through natural language sounds like it would open up many possibilities for expression and allow for comfortable social interactions. However, the limitations of the agents' intelligence and the relatively small amount of mappings between sentences and actions ended up creating an awkward experience when the players strayed too far from the author's intentions. Many actions also failed to feel meaningful, having little effect on the overall story. Mateas and Stern spent three years producing *Façade*; developing even small storyworlds with their architecture took too much development time. Just like the problems Crawford's *Storytron* encountered, *Façade*'s engine failed to take off into something more because of an awkward interface and frustrating authoring tools. Since

the release of *Façade*, its technology and methods have not been duplicated or improved. However, the papers written by Mateas and Stern about the development of *Façade* provide a valuable resource for looking at the creation process of a particularly complicated interactive storytelling system.

Before going any further, I will bring clarity to the meaning of some of the more common and important words used in this paper. First, of course, is the term interactive storytelling. By this I mean to speak of any story where the audience, at some point during the story's life, has a say in how the story progresses. When I speak of the ideal interactive storytelling that my research tries to solve for, I mean an interactive story where the interaction has depth and meaning; an interactive story which allows for rich play and drama. By meaningful interaction, I intend to describe interactions where the player has a good idea of what their actions will do, and generally speaking, the results of their actions satisfy their hopes. Players are the humans who play with the storytelling system, exploring it, subverting it, or embracing it. Verbs are the actions an agent chooses from and then executes in the story. Plot and story are distinguished; plot is the sequence of events while the story is the plot filtered through the telling, given style and meaning. Storytelling is the act of transforming the plot into story. Storyworlds are the sets of rules, entities, and verbs which are used to unfold an interactive story and are constant from playthrough to playthrough. An interactive storytelling engine is a piece of software which reads files defining storyworlds and animates them. Agents include both artificial intelligence characters and the player. These terms lay the conceptual foundation for understanding the contents and worldview of this paper.

Technical Approach

The solution to interactive storytelling this research explores is not the only nor the best solution. Throughout research the intention was to develop a general model for interactive storytelling; the solution should avoid favoring one genre or type of story over another. Conceptually, the process can be split between modeling the story and modeling the interaction. Of course, the two eventually must meet and work together. The story is modeled by a series of storyworld states in between which are transitional verbs. The interaction is modeled by allowing any agent interacting with the storyworld to act in the world through these verbs, having influence on which state is arrived to next. With this model, the player and any artificial intelligence characters or objects are of equal status, interacting with the storyworld in the same exact way. Thus, modeling any computer agent in the interactive story is actually a separate but related problem to modeling the interactive story itself and is, for the most part, beyond the scope of this research. The resulting interactive story model serves as a framework for writing interactive stories and needs to be designed as a tool. Not only should the model simply function, but it should also be easy to work with and powerfully expressive for designers of interactive story. All of these ideas together are the basis for an interactive storytelling solution.

To create an interactive story, a computational model of story needs to be designed. The simplest understanding of story divides it into a series of logically and dramatically ordered events that, taken as a string, create the plot. Now, of course, story is not plot, but for the sake of simplicity avoid the issue of modeling the storytelling for the moment. As a plot progresses, the world that the story started as changes. A city that once stood is decimated by war. The dog who once was looking for a bone has found

one. Two people who used to be on opposite ends of the world have met and fallen in love. A man becomes angry, and then he sets a building on fire. The world has a form and that form changes. For the sake of the computer, this change should be thoroughly discrete. This model, in an attempt to generalize, could assume that time always progresses linearly in a story, which is often not the case. Flashbacks and nonlinear telling of a plot exemplify creative moves in storytelling that, while they can still be broken down into a series of events, the events they are made of have a more complex temporal and causal relation to each other. A woman reminisces about her childhood. Time travelers rush to resolve world ending paradoxes. The storyworld's state still transitions, but for drama rather than logic.

What does the state of a storyworld look like? The world can arbitrarily be divided into entities and these entities can be differentiated by their properties. A storyworld can consist solely of an island, a coconut tree, and a man. These are three distinct entities, all of which have their own properties. The island could have the property of being sandy; the coconut tree can have the property of being alive. The man can have the properties of being named Lester, being a good climber, and being terribly hungry. A storyworld transitions state by transitioning properties. A sum of property transitions for different entities changes the nature of the storyworld. Lester could climb the coconut tree and eat a coconut. By doing so, he loses the hunger property. While video games model their virtual worlds almost exclusively in spatial terms, stories often have very little concern for the mathematical specifics of space. Therefore, the exact position of the coconut tree does not matter. This model can run into limitations, however. Once the entities are divided, the divisions stand. By changing properties,

entities can often change into two entities or become part of another entity. While these cases could be added to the storyworld state model, they would add complexity and mostly, again, describe details which are fundamentally spatial or numerical, properties a story does not often concern itself with.

Verbs change the storyworld's state and move the story forward, animating it. While the storyworld state's entities and properties describe what the storyworld *is*, verbs describe what can happen in a storyworld; they describe what the entities can *do*. Verbs are the glue between states, the glue between the story and the agents playing in it, and the glue between the story and the storytelling. Verbs have two ends which feedback into each other; conditions and operators. The conditions look at the storyworld state and determine if the verb is logically possible. If the verb is possible and is executed, the operators determine just what kind of effect the verb is going to have on the storyworld's state. With the change in the storyworld, the conditions are then checked to see what new verbs are possible. Conditions are if-statements that say whether or not the verb is a valid way the story can progress at the given story state. If a woman has legs, she can run. If the robot has a laser, it can shoot down a helicopter. If a monster is tied up in chains, it is unable to move or attack anyone. Conditions keep a story logical. Instead of simulating the mathematical specifics of physical law, conditions compress the patterns of reality (or the reality of a fantasy world) into a more manageable form. Operators define how the state changes when the verb occurs. They state which entities get or lose which properties. Verbs are executed by entities tied to agents. The cat agent, who is tied to a cat object in the world state, could choose to execute the verb "drink milk," making the cat object lose its thirsty property and giving the milk bowl the property of emptiness.

This model of verbs allows for the player and the artificial intelligences to interface with the storyworld as equals. Every agent has a turn to act. All agents in the story go through the same interaction cycle. First, they observe the state of the storyworld and the events that have occurred in it. After doing this, they make a choice of one of the possible verbs they can execute. This choice creates an event in the story which changes the storyworld. The next agent then observes the recent event and the new storyworld, going through the same cycle. This continues indefinitely until the story reaches a defined ending. Sometimes an agent might reach a point where they cannot take any action. If this becomes the case, a null "wait" verb needs to be available that lets the agent forfeit their turn. This opens up an interesting limitation to this system. If the storyworld is able to reach a state where every agent is incapable of action and they all have to wait, the interaction cycle gets stuck in an infinite loop. Under this model, designers of storyworlds would need to design around this limitation.

Events and verbs are different sides of the same coin. When an agent chooses to execute a verb, they are choosing to take action. Depending on conditions in the storyworld, that action could lead to one of any number of events tied to the verb. The series of all the different agents' actions can be viewed as a series of events, otherwise known as a plot. All the possible plots an interactive story can have are to be found in all the possible series of verbs. Verbs need to operate in sequence and not in parallel. Parallel verbs create overlapping events which might contradict each other. A dry readout of a list of verbs would be a very uninteresting way of telling a story. Storytelling works because of execution. If execution was optional, reading a Wikipedia summary of story would be just as entertaining as reading or watching the actual work. The barebones plot

reduction of a story can often times make a great story seem ridiculous or uninteresting. Therefore, a system needs to exist which tells the computer how to embellish the telling of the events a verb causes. While an advanced artificial intelligence with artistic reasoning that tells the plot with style would be the dream solution, a more reasonable method is the simple mapping of events to storytelling. Events could map to dynamic 3D cutscenes, paragraphs of text, or the pages of a picture book. The player will choose an action, an in return, get an interesting telling of what happens because of that choice. The problem with this method is if an event is repeated, it unfolds in exactly the same way as before. Heavy repetition of media in video games, for instance, is widely criticized (namely, repeated audio dialogue). A partial solution is to give the designer the ability to define how the telling of an event can be permutated based on the storyworld's state.

This model that connects actions, events, and storytelling together is a way to join the interaction to the plot and the plot to the storytelling. This creates a mutual bridge between interaction and storytelling. The result, if the storyworld is well built, should be interactive storytelling realized. Verbs are the key, fundamental element to making this all come together. Actions in the storyworld are defined by verbs. Events in the storyworld are defined by verbs. Storytelling in the storyworld is defined by verbs. The storyworld's state serves the verbs, determining where they appear. Designers of storyworlds need to create interesting, effective verbs and then design the storyworld state space to allow for a dramatic, logical flow of those verbs. By emphasizing verb design, creators emphasize the importance of connecting interaction to storytelling.

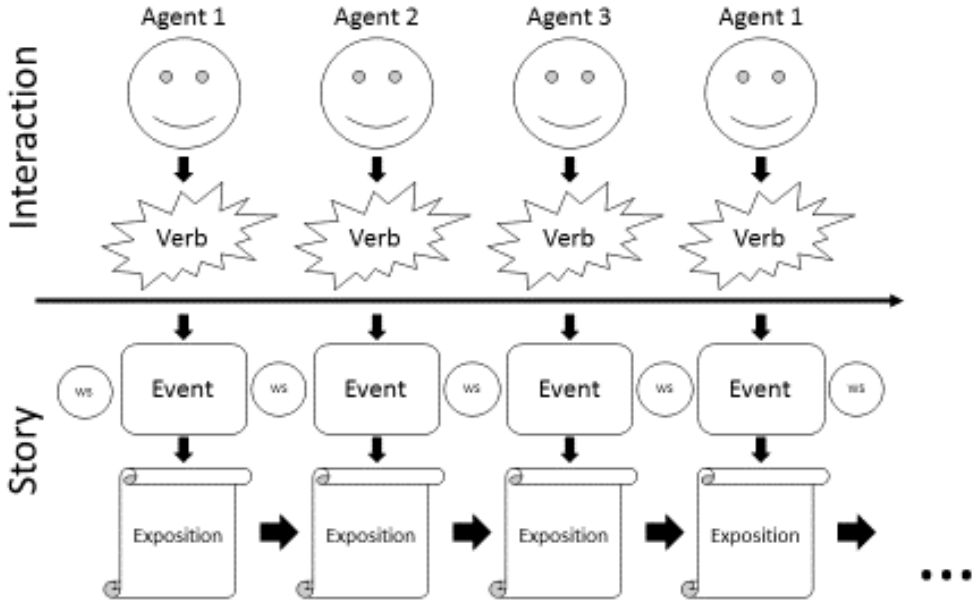

Figure 1. Verbs and events bridge interaction and story.

These designers need tools to create storyworlds. A model for interactive storytelling must be able to act as a framework for numerous interactive stories. To work within this framework, tools are needed. The storyworld model of entities and verbs is a strange mix of data definition and semi-procedural instructions. Designers need the ability to state what entities exist and what properties those entities have. This can be tricky, as the entities cannot only be designed with their current state in mind, but their eventual permutations also need to be considered. Entities need to be assigned agents if they represent characters or objects which act through verbs. Entity definition is data definition. Creation of verbs, on the other hand, is a far more complex process. Verbs need to be able to query the state of the storyworld thoroughly enough to check conditions. A language for describing these queries needs to exist for the designer. Operators of verbs also need a language that describes what they do to the storyworld state. Once a verb is executed, it becomes an event that needs to be told to the player. A

method for determining what to show for this event needs to exist; most likely, this would be a reference to some piece of media or set of instructions that, based on the storyworld's state, tell the system how to combine multiple pieces of media into a single experience. Beyond entities and verbs, the designer needs to define what events might be shown at the beginning of the story to set the stage as well as the points when the story might end and what events happen during those endings. The most advanced parts of the interactive story to design are the externally scripted artificial intelligences which act as agents. They need to be designed for the system's interface. Designers could share general purpose artificial intelligences with each other, assisting creators who are not programmers. This should suffice as the requirements for tools that allow for the production of interactive storyworlds that follow this research's model.

Details and Methods

The *LudoNarrare* engine implements the interactive storytelling model from this research. Using the Unity game engine as an interface, *LudoNarrare* focuses on creating interactive picture book experiences. The picture book genre was chosen for its relative simplicity and its emphasis on visual storytelling, both of which I believe are important to selling the interactive storytelling idea. Conceptual thinking fails to reveal many of the finer problems that a real system will encounter. So while the conceptual musing above lays a foundation for understanding the research, the true lessons are found in the real manifestation of the system, beyond what foresight provides. The process of creating *LudoNarrare* required a large amount of faith in the design since a large amount of interconnecting parts had to come together for any meaningful results to happen. This engine underwent a heavy process of evolution. Many features were cut; many features had to be added. Ideas about the storyworld model required further fleshing out beyond what was originally planned. Several tools for designing storyworlds where created and thrown out. Once the engine reached a point that storyworlds were ready to be designed, the research began to shift towards thinking about the principles that would lead to interesting interactive stories. With the given time limitations, I could only begin to explore the possibilities of what types of storyworlds could be created.

The first few months of developing *LudoNarrare* where completely rooted in design work and brainstorming. Reading the research of others interested in interactive storytelling provided some helpful thoughts, but the most important sources of inspiration were the already existing tools for interactive storytelling and the products created with them. Whenever an interesting idea came to me, I made sure to add it to a bulleted list of

notes. From these notes came the *LudoNarrare* Design Bible, a twenty page design document detailing a plan for the initial version. Many handwritten documents, mostly flowcharts and graphical depictions of systems, also contributed to the growing knowledge around what *LudoNarrare* was going to become. As part of another external research project, I had to prepare a ten minute presentation regarding the architecture and design of the engine. This task proved incredibly useful; challenging myself to explain the system plainly to an audience forced me to clarify some otherwise fuzzy concepts. Presenting the ideas made them vulnerable to an audience who could reveal issues which I was blind to. With the design in place, the hard work programming the engine and getting it running began.

Development on *LudoNarrare* began with C# in Visual Studio. First, before an engine that animated storyworlds could be created and tested, a method for creating and storing storyworlds had to be devised. Originally, due to the large push in the interactive storytelling community for tools that are easy for nonprogrammer authors to use, the plan was to create a GUI toolkit that allowed intuitive defining of the storyworld's entities and verbs. A large amount of time was put into building this GUI tool; however, it quickly became apparent that it would take too much time to finish and debug. An alternative was sought after. Although I was initially not fond of the idea of writing the storyworlds in a structured file format like XML, I had no other choice but to implement it.

Alongside building the GUI, I was writing the classes for the different components of the storyworld data structure, figuring out how it would appear in memory and what functionality it needed. The very first storyworld model featured entities composed of a name, a set of attributes, and a set of relationships. Attributes where string

tags like “tall” or “blue”; an entity either had an attribute or it did not. Relationships were named pointers to other entities; for example, Lester “loves” Julie. Entities also featured three additional properties, obligations, goals, and behaviors, which were the original solution to artificial intelligence agents. Obligations said that if an entity could do the verb the obligation pointed to, they would. Goals said that if an entity had no obligations and could do a verb whose operator matched the operator the goal described, it should. Behaviors made entities without obligations or goals chose a verb based on the verb’s odds. This system for controlling agents would survive several iterations of *LudoNarrare*.

Verbs originally consisted of a set of conditions, a set of operators, and a text description that described the verb’s event. This would prove to be way too simple a model from the start. The first element added to verbs were arguments, which allowed entity references to be passed to the verb by the agent doing it in order to change the nature of its operations. For instance, the “walk” verb needed an entity with the attribute “place” that the agent would walk to. This argument system would prove to be a powerful way to compress a large amount of verbs into one for both the designer and the player, adding way more possibilities without the cost. Conditions checked named or argument entities for attributes and relationships while operators added or removed attributes and relationships from the given entities.

The first interface for *LudoNarrare* was completely text based, even though the intention from the start was to eventually build a picture book interface in Unity. A scrolling window of text descriptions of the events told the story to the player while drop down combo boxes where used for the process of selecting verbs and verb arguments. Each storyworld had a beginning piece of text which set the stage and several endings

which were defined by a set of conditions and story text. With the storyworld class models, the storyworld XML parser, and the player interface built, the last task before having a working interactive storytelling engine was to engineer the loop that breathed life into the flow of events as conceived in Figure 1. The player would be given an initial choice of verb. Then, all other agents were cycled through randomly, being asked to make a move. The obligation, goal, and behavior agent model was used to determine if they would. In order for a verb to be chosen, all possible verbs had to be enumerated. This required functions for checking conditions and building a tree of all possible versions of the verb based upon the arguments it could accept. Once a verb was chosen, its results were displayed and its operators applied. This process would continue until the scheduled check for an ending had been arrived to.

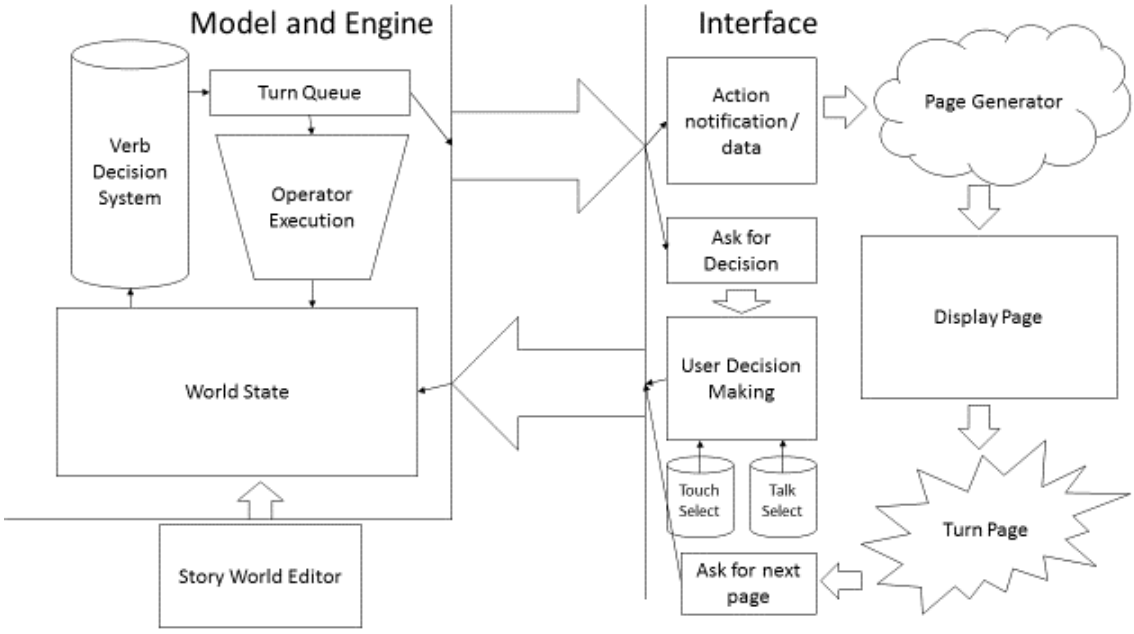

Figure 2. An early flowchart showing the story and interface loop.

To test this first version of the *LudoNarrare* engine, several tiny, trivial storyworlds were written. One storyworld simply had the player and other agents moving

from room to room. Another had the player painting a room different colors. In one of the storyworlds, the player could pick up items, use them, or put them down. All of these very basic storyworlds were used to debug the many features that had been built up until this point. Most of these features, in order to work, had to be rethought and rewritten. Once these issues had seemed to be ironed out, work on a more substantial storyworld to show off interactive storytelling in *LudoNarrare* began. This storyworld was called *Desert Delvers* and it was supposed to tell the story of adventurers heading out into desert mountains in search of gold. A demo that featured many elements of the test storyworlds was built where the player and other agents could walk around a small town picking up items, buying supplies, hurling insults, and punching each other. This storyworld began to burst the engine at the seams, as more ugly glitches started to appear with the larger possibility space. Problems around the believability of the agents also began to rise, as their stochastic behaviors seemed to be very uninteresting. However, it did run. This was a start, though perhaps a bit ambitious.

After the text based version of *LudoNarrare* had reached enough of a functional level, it was time to move it over to Unity. Since Unity used C# for its scripting, the porting process offered no major issues. An interface for an interactive picture book was to be made for Android tablet. Input would be both gestural and spoken. Since this version of *LudoNarrare* was to be presented as a live demo, part of the focus was on making the visual interface attractive. A large amount of work was put into a 3D model for turning pages and a pleasant system for selecting verbs and arguments involving colorful icons and selection arrows. Instead of just text for storytelling, each page featured a series of sprites with a text overlay. This required the addition of a page class

which contained instructions for drawing each page instance. Every storyworld would need to contain these page definitions for the beginning, verbs, and endings.

With the first version of *LudoNarrare*, the inefficiency of XML for writing storyworlds became more and more clear. JSON was considered as an alternative, but ultimately, the XML parser was replaced with a custom parser. With a custom parser of storyworld files came a custom language for writing storyworlds, a language called LNScript. While LNScript took a while to implement, with a lexer and parser written from scratch, it opened the door for new features to be added to the engine more efficiently.

The second version of *LudoNarrare* added a plethora of new features to both entities and verbs. Attributes became tags. Entities gained two new properties; integer numbers and strings. These would allow for conditions and operations based around mathematics and text. Entities also had a set of sprites which could be used to draw them on a page. Verbs gained variables, preconditions, cases, and discriminators. Variables allowed the verbs to work with lists of entities and check their properties as a whole. Cases allowed a verb to have varying results depending on the storyworld's state. Punching a friend has different outcomes from punching an enemy. Both a good artist and a bad artist can paint a painting. However, the resulting painting will vary based on the properties of the creator. Preconditions were conditions universal to all cases, while each case had its own conditions it had to check. Every case also had a set of operators and a page describing the event that the case correlated with. Adding cases required a reworking of the system that generated all possible verbs, since a verb would only be possible if at least one of its cases could be executed. Finally, discriminators were added

to limit what verbs would be turned into pages in the storybook. If a discriminator existed, only when its associated set of conditions were satisfied would the page describing the verb appear. This allowed for the player to have limited knowledge of what was happening in a story, unaware, for instance, of what agents outside their current area were doing.

The storyworld that was created to demonstrate this new Unity version of *LudoNarrare* focused around a Rock, Paper, Scissors tournament where the player competed against a simple agent. The verb set was simple, consisting of being able to play the game, giving praise or insults, and querying for details about the objects in the story. This storyworld made heavy use of variables, cases, pages, strings, and integer numbers. While this storyworld worked flawlessly, other attempts to make a storyworld with this second version of the *LudoNarrare* engine often ended with a large amount of technical problems. The parser was not completely correct, nor did the logic behind conditions, operators, and agent behavior always work. The Rock, Paper, Scissors demo storyworld played well with an audience, but it was extremely simple and was only loosely an interactive story. The backend of *LudoNarrare* would have to be rewritten a third time for the lingering problems to be resolved.

To begin development on this third version, the LNScript language was heavy modified to be more consistent and more powerful. Conditions could now be more complicated expressions involving “and” and “or”. Numeric expressions gained added complexity via the use of parenthesis to combine arithmetic operations. Cases could render more pages and discriminators could completely hide certain verbs. A proper

grammar was written for the new LNScript so that a rewritten parser could be more easily checked for issues bringing in storyworld data. That grammar appears in Figure 2 below.

```
<STORYWORLD> is starting non-terminal.

<STORYWORLD> ==> storyworld WORD <STORYWORLDBLOCK>

<STORYWORLDBLOCK>n==> { <DEFLIST> }

<DEFLIST> ==> <DEF> <DEFLIST> || <DEF>

<DEF> ==> <BEGINDEF> || <ENTITYDEFLIST> || <VERBDEFLIST> || <ENDDEFLIST>

<DRAWDEFLIST> ==> <DRAWDEF> <DRAWDEFLIST> || <DRAWDEF>

<DRAWDEF> ==> draw: <REFERENCEOP> image WORD, <EXPRESSION>, <EXPRESSION>; || <TEXTDEF>

<TEXTDEF> ==> text: STRING;

<BEGINDEF> ==> beginning { <PAGEDEFLIST> }

<PAGEDEFLIST> ==> <PAGEDEF> <PAGEDEFLIST> || <PAGEDEF>

<PAGEDEF> ==> page WORD { <DRAWDEFLIST> }

<ENTITYDEFLIST> ==> <ENTITYDEF> <ENTITYDEFLIST> || <ENTITYDEF>

<ENTITYDEF> ==> entity WORD { <ATTRIBUTELIST> }

<ATTRIBUTELIST> ==> <ATTRIBUTE> <ATTRIBUTELIST> || <ATTRIBUTE>

<ATTRIBUTE> ==> <AGENTDEF> || <ICONDEF> || <TAGATTR> || <RELATEATTR> || <STRINGATTR> ||
<NUMBERATTR> || <IMAGEATTR>

<AGENTDEF> ==> agent: WORD;

<ICONDEF> ==> icon: STRING, <EXPRESSION>, <EXPRESSION>, <EXPRESSION>;

<TAGLIST>    ==> <TAGATTR> <TAGLIST> || EMPTY

<TAGATTR> ==> tag: WORD;

<RELATEATTR> ==> relate: WORD, WORD;

<STRINGATTR> ==> string: WORD, STRING;

<NUMBERATTR> ==> num: WORD, NUMBER;

<IMAGEATTR> ==> image: WORD, WORD;

<EXPRESSION> ==> NUMBER || (<EXPRESSION> <NUMOPERATOR> <EXPRESSION>) || <REFERENCEOP> WORD

<NUMOPERATOR> ==> + || - || * || /

<VERBDEFLIST> ==> <VERBDEF> <VERBDEFLIST> || <VERBDEF>

<VERBDEF> ==> verb WORD { <ICONDEF> <TAGLIST> <VARIABLELIST> <ARGUMENTLIST>
<PRECONDITIONLIST> <CASELIST> <DISCRIMINATORLIST> }

<VARIABLELIST> ==> <VARIABLE> <VARIABLELIST> || EMPTY

<VARIABLE> ==> variable VARIABLE { <CONDITIONLIST> }

<ARGUMENTLIST> ==> <ARGUMENT> <ARGUMENTLIST> || EMPTY

<ARGUMENT> ==> argument VARIABLE { <TEXTDEF> <CONDITIONLIST> }
```

```
<PRECONDITIONLIST> ==> <PRECONDITION> <PRECONDITIONLIST> || EMPTY

<PRECONDITION> ==> preconditions { <CONDITIONLIST> }

<CASELIST> ==> <CASE> <CASELIST> || <CASE>

<CASE> ==> case WORD { <CONDITIONLIST> <OPERATORLIST> <PAGEDEFLIST> }

<DISCRIMINATORLIST> ==> <DISCRIMINATOR> <DISCRIMINATORLIST> || EMPTY

<DISCRIMINATOR> ==> discriminator never; || discriminator WORD { <CONDITIONLIST> }

<CONDITIONLIST> ==> <CONDITION> <CONDITIONLIST> || EMPTY

<CONDITION> ==> where: <CONDITIONDEF>;

<CONDITIONDEF> ==> ATOMICCONDITION || not <CONDITIONDEF> || (<CONDITIONDEF> and
<CONDITIONDEF>) || (<CONDITIONDEF> or <CONDITIONDEF>)

<ATOMICCONDITION> ==> <REFERENCE> has tag WORD || <REFERENCE> has relate WORD ||
<REFERENCE> has string WORD || <REFERENCE> has num WORD

<ATOMICCONDITION> ==> <REFERENCE> WORD <REFERENCE>

<ATOMICCONDITION> ==> <REFERENCE> WORD <COMPARISON> <EXPRESSION>

<ATOMICCONDITION> ==> <REFERENCE> WORD matches STRING || <REFERENCE> WORD matches
<REFERENCE> WORD

<ATOMICCONDITION> ==> VARIABLE empty || VARIABLE same VARIABLE

<COMPARISON> ==> = || != || < || > || <= || >=

<REFERENCE> ==> WORD || VARIABLE || one VARIABLE || all VARIABLE

<REFERENCEOP> ==> WORD || VARIABLE

<OPERATORLIST> ==> <OPERATOR> <OPERATORLIST> || EMPTY

<OPERATOR> ==> do: <OPERATORDEF>; || do: <AGENTOPERATORDEF>;

<AGENTOPERATORDEF> ==> for WORD <OPERATORDEF>

<OPERATORDEF> ==> <REFERENCEOP> add tag WORD || <REFERENCEOP> add relate WORD,
<REFERENCEOP> || <REFERENCEOP> add string WORD, STRING || <REFERENCEOP> add num WORD,
<EXPRESSION>

<OPERATORDEF> ==> <REFERENCEOP> change image WORD to WORD

<OPERATORDEF> ==> <REFERENCEOP> remove tag WORD || <REFERENCEOP> remove relate WORD,
<REFERENCEOP> || <REFERENCEOP> remove string WORD || <REFERENCEOP> remove num WORD

<ENDDEFLIST> ==> <ENDDEF> <ENDDEFLIST> || <ENDDEF>

<ENDDEF> ==> ending WORD { <CONDITIONLIST> <PAGEDEFLIST> }
```

Figure 3. LNScript Grammar Definition

Alongside a new grammar and parser for LNScript, *LudoNarrare* needed a total rewrite of conditions and operators. Instead of handling these procedures in the engine loop, the

functions where moved to the condition and operators classes. The refactoring of these classes proved highly successful with the code being cut to half the size and losing most of its original bugs.

The final change in the third version of *LudoNarrare* was a total rethinking of how agents were to be implemented. The obligation, goal, and behavior system, while simple in concept, showed itself to, in practice, be confusing to design for and it did not produce many interesting results. Keeping with the philosophy of placing the player and the other agents on as equal standing as possible, the new system would make the agents' thinking entirely separate from *LudoNarrare*. The engine would give outside, custom scripted agents the list of possible verbs, the story so far, and the state of the storyworld and look for a decision in response, much like how it does for the player. The algorithm for decision making could be anything; it could be a traditional artificial intelligence, a mapping of decisions from a database of prebaked performances, or simply a random number generator. These and other possibilities could open doors for far more complicated and interesting models of personality, memory, planning, and learning than the old system provided. The modifications to the general storytelling loop these changes required brought an additional feature: the ability for an entirely agent controlled storyworld. This means an agent can be put in control of the player's character. All the agents together would then use the model for interactive storytelling to procedurally generate a story based on the storyworld's rules. The player could also then be placed in control of entities which were previously only controllable by agents. Unfortunately, time did not permit for much experimentation with these external decision making agents.

Further investigations could explore the wide array of systems that could be plugged in as agents. The current storyworlds created with *LudoNarrare* use agents with basic random behavior and support for probabilistic decisions regarding which arguments to choose. Furthermore, the agents may be forced to make a specific choice of verb given a certain set of circumstances. While this system is about as uninteresting and barebones a solution as possible, it still provides reasonably believable and entertaining behavior given a small storyworld with well-designed verbs that work around the limitations of the agents that choose them.

To display the heightened stability and extra features of the third implementation of *LudoNarrare*, a storyworld loosely based on the *Three Little Pigs* was designed. In it the player takes on the role of the wolf trying to eat the three pigs, all of whom are computer controlled agents. It follows the narrative flow of the children's story, with the pigs building homes of different materials which are then destroyed by the wolf. The pigs and the wolf go back and forth with each other in their conflict; one ending has the wolf eating all the pigs while another has all the pigs surviving the wolf's attempts to get them. A large amount of the variation in the story comes from the interactions between the wolf's tools and the pigs' homes. For example, the wolf can destroy a house made of brick with a sledgehammer, but cannot destroy a house made of iron with it. This storyworld features many verbs which would break the previous version of the engine. The importance of stability when experimenting with storyworld design cannot be understated; a broken engine means limited features and unpredictable behavior. The *Three Little Pigs* storyworld, while still not of any true complexity, does a better job of

showing off the type of experience interactive storytelling can create compared to previous demos.

Even though most of the research and development time was spent getting the *LudoNarrare* engine into a working state, a small amount of time was used to design storyworlds for this system and learn a bit about their nature. These are some of the design heuristics discovered for creating interactive stories with *LudoNarrare*. First, minimizing the entities, properties, and verbs used for only very specific purposes maximizes the dynamic interactions that can happen in a storyworld. For example, instead of having verbs called "burn person" and "burn wood," simply have a verb titled "burn" that can then apply the "burning" attribute to any burnable object it is used on. Second, it could often be the case that the player or another agent has multiple verbs at their disposal which can accomplish their goals. To make this situation more interesting, these verbs should have differing side effects. The agent then has to choose which side effects they want to live with, or even, if they want to live with any of the side effects at all. Finally, the more verbs that can be compressed via arguments the better. This creates more options, makes the job of writing verbs easier for the designer, makes the player feel more control over their actions, and decreases the clumsiness of the verb selection interface. If *LudoNarrare* has a future, most of it lies in the creation of interesting storyworlds that explore the opportunities of interactive storytelling. For interactive storytelling systems like *LudoNarrare* to reach a mainstream audience, they need high quality and well-designed storyworlds that capture players' imaginations.

Evaluation

LudoNarrare has not yet been evaluated formally, but a plan for evaluation can be laid out. The interactive storytelling engine consists of several aspects, all reaching for related but distinct goals. The first aspect to evaluate is whether or not *LudoNarrare* does interactive storytelling at all. Does *LudoNarrare* distinguish itself from video games, virtual worlds, and other interactive forms to specifically create narrative experiences? LNScript and the pipeline for building storyworlds in Unity are the tools creators use to build content for *LudoNarrare*. The efficiency and power of these tools needs to be tested so that problems preventing designers from writing storyworlds can be reduced, increasing both the amount and quality of interactive stories available. The *LudoNarrare* engine itself can be tested in a couple of ways. First, the quality of the interactive storytelling it provides should be evaluated using combinatorial analysis of how many meaningful possibilities it can create for storyworlds. Second, the technical limitations of the engine need to be explored to know just how complex storyworlds can become before they start taking up too much memory or processing time. This knowledge not only determines what devices are capable of using *LudoNarrare* for interactive storytelling, but if the engine can run sufficiently large enough storyworlds at all. Creators would have little interest in small storyworlds similar in size to the demos created during the development process. The final test of whether or not *LudoNarrare* can find some success solving the interactive storytelling problem is to put storyworlds running on the engine in the hands of players and observe their experiences and impressions. These are the methods I propose for evaluating the *LudoNarrare* project to determine its quality and worth.

LudoNarrare's entire purpose for existing is to create and bring to life interactive stories. If it does not create interactive stories, but rather something else, it has fallen short of its aim. Clearly, as discussed earlier, a story can be understood as a series of events well told. *LudoNarrare* meaningfully strings together verbs which are then turned into events and then formed into well told picture book pages. Players have some say as to which verbs are strung together, adding interactivity. Technically, under these definitions, *LudoNarrare* creates interactive stories. But storytelling does not stop at describing events well. It also entails the ordering of events to also be well told. Most stories build to a climax, give exposition in the right spots, and create clear arcs of plot and character. In *LudoNarrare,* there exists no mechanisms for such things. The same logic used to claim *LudoNarrare* creates interactive stories could be used to also claim that all video games and virtual worlds create interactive stories. In fact, *LudoNarrare* can be used to create games (the Rock, Paper, Scissors storyworld is proof of this). Does this unravel the entire endeavor?

No. Interactive stories, by their very nature, are going to have different structures and storytelling methods from traditional, linear stories. The existing body of work already shows that interactive stories tend to be longer than their linear counterparts. They also show that interactive stories follow different rhythms than traditional stories. These known differences and those yet to be discovered are fantastic; they open up entirely new experiences and ways of thinking that would never come from linear storytelling. Earlier thoughts about what *LudoNarrare* was going to be included ideas of "story grammars" that helped guide the types of verbs that could be executed at certain moments of the story. Verbs early on in a story, for instance, might be limited to the less

dramatic, while as the story progressed, only dramatic, climatic verbs could be used. Furthermore, verbs executed earlier in the story could make a note requiring for a particular type of follow-up verb to occur before the story's end. These concepts did not gain ground because outside a storyworld with a vast amount of verbs, they would severely constrain what the agents, particularly the player, could do at any given moment. The player will always need to take some responsibility for the quality of the drama in a story that is interactive, especially if they also desire a satisfying amount of agency. Despite this, a form of drama management which guides the computer agents or provides a structure compatible with interactive storytelling could still be valuable. *LudoNarrare* lacks this; it has the challenge of crafting storyworlds that are inherently dramatic due simply to the logic which connects the possible events and the agents' behavior.

While *LudoNarrare* is a close cousin of video games and virtual worlds, they are not the exact same. Most video games and virtual worlds are concerned with spatial understandings of environment and it shows in the actions they make available (walking, looking, jumping, and shooting are all geometric in nature). *LudoNarrare* does away with thinking this way; dramatic stories have little interest in spatial details. As said previously, by certain definitions games and virtual worlds do tell stories, but they often are not concerned with this. They are trying to accomplish other goals; their creation of stories is accidental. *LudoNarrare* was designed to concentrate designers on intentionally creating stories. If these stories are good as stories, then the goal has been achieved.

Without storyworlds, *LudoNarrare* is barren. And without good tools, storyworlds will be hard to come by. Therefore, *LudoNarrare* is useless if the toolset it provides makes it difficult for creators to craft storyworlds. Designers will primarily use LNScript

for storyworld creation. Like other programming languages, LNScript should be expressive, readable, and writable. To test these qualities, analysis could be done on a sample of designers using LNScript to write storyworlds. The average length of time designers need to write an entity, a verb, or an entire storyworld and the average amount of times designers need to debug a problem could serve as possible quantitative measurements of LNScript's quality. Listening to the feedback and intuitions of designers who already understand another form of interactive storytelling could reveal qualitative data on whether or not LNScript allows for easy expression of ideas. If the majority of designers give up on LNScript due to it being unwieldy, there is a problem. Beyond LNScript, the Unity game engine gives designers of storyworlds a pipeline to load in graphic resources and write artificial intelligence agents. While adding graphic resources is trivial and needs little to no validation, the system for writing agents does need validation. The agent system succeeds if one agent model can be applied to many different entities once it has been written. Quantitatively, this goal can be checked by counting the amount of times each agent gets reused by the designer. If qualitative reports show that designers are struggling to write agents that interact with the LNScript storyworld, there is a problem with the interface between the two. *LudoNarrare* depends on both LNScript and the Unity resource pipeline to deliver designers an effective and comfortable environment for the creation of interactive storyworlds.

Combinatorial analysis could be a potential quantitative tool for learning how dynamic individual storyworlds are and what variety of possibilities they can offer. A large space of possibilities points to the emergent quality of an interactive storytelling system for creating situations that surprise even the creator, an ideal goal. An interactive

story with ten different routes does not compare to an interactive story with 1,000,000 possible routes. Enumerating the paths through brute force would be the easiest implementation. This type of analysis would be able to determine the space of possibility for individual storyworlds and speak to their own quality. However, a large average number of story paths for all created storyworlds would bode well for *LudoNarrare* itself and its ability to create stories that are highly interactive. A possible issue using combinatorial analysis on storyworlds arises when possible story routes are infinite, which could very well be the case. This issue can be solved by limiting the depth of the enumeration to a certain point; although the maximum number of paths is infinite, all meaningful paths can be captured after a certain depth into the story. The possibility of easily creating a storyworld with infinite paths in *LudoNarrare* shows some limitations of considering high path counts as a metric for quality interaction. For more nuance, consideration of event variety and the qualitative measurement of a player's sense of agency are also needed.

What does *LudoNarrare* have if computers cannot run it? Both space and time complexity need to be considered to know the hardware options the engine has available. Space complexity would be the simpler of the two to analyze. *LudoNarrare* needs to store all the data for the storyworld model as well as duplicate storage of every verb during the verb decision process. Beyond this, it keeps in memory all the previously rendered pages in the storybook and the graphical sprites used to render entities. It would be unlikely for the data beyond the images to take up a modern memory space. However, to confirm that it does not would simply involve measuring the average memory usage of a wide sampling of storyworlds. Since a storyworld's information can grow with entities

gaining properties, the measurements should be done on multiple storyworlds to show how they grow and shrink over the lifetime of their different paths. If the trend shows that the size of entities grows too large, there may be problem.

Checking the bounds of *LudoNarrare*'s time complexity requires more thought. Almost all processing occurs between the player choosing a verb to execute and the storybook showing the results of that action. Since the interface is turn based, performance is less of an issue than it normally would be for an interactive system. However, even though a ten second or even one minute delay could be acceptable, players should not have to wait twenty minutes or an hour for the results to come in. While there will always be a storyworld sufficiently complex enough to slow processing to a halt on any hardware, the loose limits of what the engine can and cannot do can still be analyzed. Say a massive storyworld that took two years of work to create consisted of 3,000 entities, 1,000 verbs, and 100 agents. A simple time complexity can be found by multiplying the number of agents by the total number of verbs, since each step of the story all agents need to consider the verbs they can choose from. This needs to be multiplied by the time complexity of choosing a verb. The equation in Figure 3 below can be used to get a rough, informal estimate of *LudoNarrare*'s story loop time complexity.

$$Agent * Verbs * (Variables * Entities * Conditions_{Variable} + Arguments * Entities * Conditions_{Argument} + Preconditions * \sum_{i=1}^{4} ArgumentValues\ i * Cases * CaseConditions)$$

Figure 4. Equation for estimating the time to render the next part of a story

With low estimates of the other values (assuming four variables, four arguments, five argument values, four cases, and six conditions for each, all numbers taken from my experience building storyworlds) a storyworld of this size would take on average

$2.34*10^{10}$ or 23 billion operations. While it would be too simplistic to definitively compare this number to the amount of operations a modern CPU can perform, my intuition says that a storyworld this large is pushing the limits of high end hardware, but still running without pausing to process. Since each iteration of determining whether or not verbs are available to an agent are mutually exclusive processes, CPUs that support multithreading would be at an advantage if *LudoNarrare* was modified to support them. All of this is an informal estimate and does not consider all the details, especially the decision process of potentially complicated artificial intelligences driving the agents. More thorough, formal complexity analysis with properly sampled averages would be needed to better understand the engine's limits. While *LudoNarrare* might have no issue with memory space, large storyworlds can potentially push even high end consumer hardware to its limits.

The final test of *LudoNarrare*'s validity as a solution to the interactive storytelling design problem is incredibly simple; put the tools in the hands of creators and the marketplace to evaluate the engine's ability to run interactive stories that are compelling to the population as a whole. If a community of designers forms around the *LudoNarrare* toolkit and develops a prolific selection of engaging, unique storyworlds that capture their audience's imaginations, the system's success proves itself in practice. Furthermore, if commercial interactive stories can be produced with *LudoNarrare* and sell enough to support small businesses, economics vindicates the concept and technology. I do not believe any of this will happen, but perhaps similar engines that build upon the ideas behind *LudoNarrare* succeed someday in the future. If *LudoNarrare* finds no success on its own but inspires a success, it will have still found validation, albeit indirectly.

Conclusion

Interactive storytelling presents a heavy challenge; the concept, while comprehensible, has not yet been fully realized. Even after years of work, some of the best designers in interactive entertainment have still not created an interactive story that truly satisfies the spirit of the endeavor. *LudoNarrare* implements a possible solution to this interactive storytelling problem. It uses a model of interactive stories constructed of a storyworld state that mutates via verbs. These verbs are chosen by the agents interacting with the story; in turn, the verbs are turned into events which are used to tell the story as it flows along. Storyworlds that utilize this model can be written for *LudoNarrare* using the LNScript language. The *LudoNarrare* engine itself then reads these storyworlds and animates them, bringing them to life. Agents, including the player, are then able to interact with the active story. The quality of the interactive stories that can be created with this system has yet to be formally evaluated. Proof of whether or not *LudoNarrare* succeeds in any way lies with the storyworlds that will be created with it. If the system proves to be capable of running a compelling interactive story experience, it has found some success. *LudoNarrare* could serve as a step forward for interactive storytelling philosophically, implementing some key new concepts for the very first time that, once refined by further technology, could be part of the solution to fully capturing the interactive storytelling dream.

How could the *LudoNarrare* project be expanded from here? Larger, more interesting storyworlds could be made to better sell the idea of interactive storytelling to potential investors who could speed up the process of solving the system's design issues. More interesting, complicated artificial intelligence agents could be created to add more

personality to storyworlds. Research could focus on how these agents learn and store information about what they experience in the storyworld. More complicated agents that pursue multilevel goals could also be implemented. The challenge is to not make agents who reason really well, a computationally heavy task, but to make agents that simulate humans dramatically. Where can future technology similar to *LudoNarrare* improve upon and expand its ideas? With the philosophy of keeping the player and the other agents on relatively equal footing, it would be quite easy to implement multiplayer interactive storytelling using the *LudoNarrare* model. Instead of one human player and many artificial intelligences, a storyworld could be a mix of many human players and many artificial intelligences. Further research can be done on improving the system that turns events into storytelling. For example, a potential storyteller agent could be designed to add more dramatic flair to how the story is presented. LNScript and the Unity implementation of *LudoNarrare* present a storyworld creation toolkit that is not the most user friendly or immediately understandable. Better visual tools for creating storyworlds could be experimented with so that more designers would feel invited to contribute to the growing knowledge on interactive storytelling. Debug tools that allow the designer to find issues with their verbs' logic or gain statistical information about queried enumerations on story paths could also be developed to allow for the quality of created storyworlds to improve. Finally, *LudoNarrare*'s interactive storytelling model could be applied to mediums outside of picture books. Imagine the integration of interactive storytelling with music, film, theater, or traditional video games. The possibilities are endless and exciting. *LudoNarrare* provides inspiration for a potential upcoming age of interactive storytelling.