Narrative world creation with assisted environment filling

Marijn Goedegebure

Master of Science Thesis

TU Delft Delft University of Technology

Computer Graphics and Visualization

Faculty of Electrical Engineering, Mathematics and Computer Science
Computer Science

Narrative world creation with assisted environment filling

Marijn Goedegebure
4013484

Committee members:

Supervisor: Prof. Dr. Elmar Eisenmann
Mentor: Dr. ir. Rafael Bidarra
Member: Dr. ir. Joost Broekens

February 16, 2018

Abstract

Games such as Skyrim use a narrative world, which is a game world that is used to tell stories. Creating narrative worlds is a time-consuming process because of the collaboration of writers, 3D artists and game designers, the story adjusting the world over time and the requirement for diverse and interesting environments. This thesis focuses on solving these problems. First, the manual approach was designed that allows the manual creation of a narrative world, while focusing on filling environments. Secondly, the assisted approach builds on the manual approach and is able to assist a designer in making the decisions necessary to fill environments. The manual approach uses a set of objects with explicit relationships, created following a modular approach. The manual approach incorporates a discrete and sequential timeline of events, with each event connected to a single location, directly into the process. Several actions are used to fill the narrative world for each point in time of the timeline. The assisted approach offers assistance for each decision of the actions. Due to computational complexity, each decision is assisted independently. The discrete decisions are evaluated and ordered through the use of weighted criteria. The continuous decisions, the placement of objects, requires generation of options, which is done through an adaptation of Merrell et al. [2011]'s GPGPU algorithm. A usability study was conducted to test the effectiveness of the assisted approach compared to the manual approach. Results were collected through a creativity support index (CSI), a questionnaire on the assisted approach and quantitative measurements on object usage and time taken for actions and locations. The manual approach scored better in the CSI results than the assisted approach. The final questionnaire showed that this was caused by automation results not matching the user's expectations as these differ, at least, per user and location type. The following conclusions are made. First, explicit relationships allow direct insight in how objects interact. Secondly, the usability study results supports the iterative process employed by, both our, and many other creative process approaches. Finally, in general, it is difficult to provide valuable assistance as expectations differ greatly. The incorporation of uncertainty into the assistance can help lessen this problem.

Preface

This report is the final result of my Master's thesis project. It marks the end of, after many amazing years, my time as a Computer Science student at Delft University of Technology. While the project had its ups and downs and was challenging from start to finish, I am proud of what I have achieved. Before I leave university for the 'real world', I would like to use this opportunity to thank several people. First of all, my thanks go out to Rafael Bidarra, my direct supervisor. Although we did not always agree, he often helped me push the thesis in the right direction. I would also like to thank Elmar Eisenmann and Joost Broekens for being a part of my committee and the efforts that took. A special thank you goes out to Tim Balint, who, from September, was always ready to offer feedback, advice or a proofread. A second special thanks goes out to Jan-Willem van Velzen, whom I started working with after a dip in motivation during the summer. Together, we spent many days working on our own thesis. Sharing the lessons learned and discussing the ins and outs of each thesis. Next to these people there are several other people I would like to thank. First, my girlfriend Lauri, for her never ending support and my cat, Freek, for clinging to my arm while I tried working on my thesis. I would also like to thank all friends I have made during my time at the TU Delft. Many fun and good moments were had, which made these years even more memorable. Lastly, I thank my parents for making this all possible and whom never stopped believing.

Marijn Goedegebure
Delft, the Netherlands

Table of Contents

Chapter 1

Introduction

1-1 Context

Games such as Skyrim™ and Fallout 3™ tell a vast amount of narratives. From short and simple to long and complex, the narratives in these games require a world that is capable of hosting all of these narratives as well as their connections, dependencies, and effects. This world is what we will call a *narrative world*. This narrative world consists of one or more locations, each having an associated space called an environment, where narratives take place and the story that is told adjusts the environments and the objects that are part of the environments. For example, stealing an object makes it disappear from its current position inside an environment and be part of the player's inventory. This elevates the creation of a narrative world from a 3D problem to a 4D problem due to the inclusion of time. For example, objects can be in several different environments during a story, which is prone to errors.

During the narrative world creation process, many creative decisions are made, which may cause requirements to change over time due to new insights. This may cause environments to be redone. Redoing environments is, first of all, time-consuming, but is also prone to errors given the four dimensions of the narrative world. The many creative decisions also cause the process to mainly be manual labor as *automating* such a process is difficult.

Something not yet considered is the player's perspective on narrative worlds. The player wants a narrative world with unique environments that are each interesting to look at and explore. This results in a trade-off between unique assets being created and time saved due to reusing assets. Reusing assets too much can cause art fatigue, the player becoming irritated and/or bored with the repetition of assets, reusing assets too little causes the time consumed for creating art assets to explode and make a game impossible to finish [Burgess, 2013]. The required diversity also influences the depth of the decisions made, as there are many options to consider and choosing between options is influenced by the other environments. To ensure diverse environments, significant time needs to be available to explore the options, which is done manually.

The manual labour, possible errors and possible art fatigue cause the narrative world creation process to be time-consuming and makes it an interesting field of research. Let's look at how the industry approaches the narrative world creation process, this will later be referred to as the *industry approach*.

Game development starts with a preproduction or design phase, which is a planning phase of the project focused on idea and concept development and production of initial design documents [Bates, 2004]. This is followed by a production phase in which the game is created [Chandler, 2009]. The creation of the narrative world occurs during the production phase, with high-level design decisions being made during preproduction. The process to create a narrative world is split into two steps: Environment creation and Environment filling.

During environment creation the designer defines a space, using techniques like a height map and/or a modular kit, on which (or in which) objects are placed during environment filling objects. In some cases the environments are independent of each other, an example of this is Halo which has separate levels, in other cases, the combined environments form an open world, for example Skyrim, Fallout 3 or Witcher 3.

The goal of environment filling is to place objects into a space until it is adequately filled, this means that it facilitates the story and conveys the meaning of that location. This requires the designer to know the meaning of a location and the story that is told there. Writers write descriptions about the location and a story that is annotated with information with locations, objects and characters. Using this, the designer instantiates objects from a list of *master objects*, created by the art team, and places them inside the space created during environment creation. Instantiation is done to reduce the amount of unique objects required. Changing requirements is a source of problems, because when the master object is changed all instances of that object are changed as well. This may cause instances to not fit their purpose in the environment anymore, requiring a fix [Burgess, 2013]. The objects used in an environment are either purely decorative or a part of the story. The story objects are important, otherwise, events of the story are not facilitated in the world and cause the story to not be playable. Objects inside an environment may have implicit relationships. For example, a designer adds a closet and a faucet with the idea that those are meant to be right next to each other. In the current *industry approach*, these relationships are never explicitly defined, which may cause problems when working with multiple people. A different designer may see the two objects and move them away from each other, breaking the implicit relationship.

1-2 Motivation

Games, in general, are an increasingly popular medium for consumers as can be seen from the revenue of the past years [McDonald, 2017]. *Roleplaying Games (RPG's)* are a popular part of the games medium and focus on telling narratives through games, making them adequate examples for our motivation. Next to a general revenue increase, budgets for RPG's are also increasing. Using the Witcher 2 and the Witcher 3 as an example of these increases, the overall budget of Witcher 2 was 25 million zloty, about 7 million US Dollar, and the overall budget of The Witcher 3 was 306 million zloty, about 84 million US Dollar, with the exchange rate being roughly the same [Superannuation, 2014]. This shows a multiplication of

the budget by 10 times. These increasing budgets show that game developers of RPG's are under pressure to develop more, with bigger teams, in a similar time span.

Some areas of game development are under more pressure than others. One of the areas that is influenced significantly is the creation of narrative worlds which host narrative and gameplay elements. Examples of the increase in size and complexity of the narrative worlds are observed from the large difference in accessible areas and quests of Witcher 2 and Witcher 3 [Koch, 2015]. This observation is made across the game industry, with other good examples being the Elder Scrolls and Fallout series.

The increase in budget has several effects, such as larger teams, pressure to produce more content and increased size and complexity of the narrative world. Larger teams increase communication required and a number of people working on same parts. This requires knowledge to be more explicit and readily available to each individual in the team as other teams might have questions about choices made. These effects also affect the problems described in the context. For example, larger teams may cause more small changes to master objects which each have the chance to break unintended usage. The increased complexity of narratives causes more changes in the narrative world to occur and each of these changes has to be made part of the narrative world. With no explicit method of tracking the changes that occur through time, these requirements are prone to an increasing amount of errors when narrative complexity increases. The pressure to produce more content may cause faster production to be required by the designers, which does not scale well due to the amount of manual labor required. This may cause designers cutting corners resulting in bland results and art fatigue due to repetition. This is supported by the article by Burgess [2013], the dangers being art fatigue and redoing lots of work if communication is not on point. In line with this manual labor is the chance of requirements changing over time. With larger teams, requirements may change more frequently due to multiple people working on the same assets and each having their own idea about its purpose. Secondly, some requirements are not explicit and changes, due to changing requirements, may cause these requirements to be removed. These unclear requirements may cause work to be redone which puts more pressure on the designers, and ultimately cause more bland results.

1-3 Proposed solution

We propose a new approach to the narrative world creation process with the goal of assisting the game designer in creating a narrative world. The approach is split into two, with the basis called the *manual approach* and the *assisted approach* which builds on the *manual approach* and aims to provide assistance in the narrative world creation process.

The manual approach allows the creation and filling of a narrative world. Environment creation is kept simple and the focus lies on environment filling as the problems mentioned in the context occur there most frequently. The environment filling part supports several actions that form a toolset that allows a designer to place and arrange objects based on master objects defined using the semantic model of Tutenel [2012]. The semantic model allows the definition of master objects and their explicit relationships, which results in semantic definitions stored in a semantic database. The explicit relationships aim to solve the problem that the implicit relationships cause. As a guideline for the narrative world creation, the story is used. The

story is a structured and computer readable narrative, which defines the characters, objects, locations and events. This is done by describing the story using the PDDL-format that accompanied Ware and Young [2014]. This results in a list of characters and objects that the designer needs to place at a certain time for a certain location of the story, resulting in a narrative world that facilitates the story and the changes that it requires.

Because of the many creative decisions made during the narrative world creation process, the time pressure to deliver and the danger of art fatigue, a trade-off has to be made between quickly deciding and taking the time exploring the options of a decision. The *assisted approach* aims to allow the designer to make a quicker and clearer assessment of the options available, causing him to be both quicker to decide as well as having spent sufficient effort exploring the different options. The *assisted approach* builds on the *manual approach* and incorporates assistance into the actions of the process. This assistance aims to provide guidance to the choices that the designer makes for each action. This is done by ordering the options available for a certain choice and presenting them to the designer. The options are ordered using an *automated* approach that uses criteria based on information extracted from the current objects and relationships of the environment, the narrative and the semantic definitions and combines them with their respective weights. Not all decisions in the actions are discrete, the placement of objects is a continuous problems and thus requires the options to be generated before they are ordered. This is done by adapting the algorithm by Merrell et al. [2011]. The criteria used are evaluated for their usage in the context of narrative worlds as Merrell et al. [2011]'s criteria are focused purely on furniture layouting.

The proposed solution leads to the following main research questions as these cover the major uncertainties of the thesis:

Research question 1.1. How does the usage of the PDDL-format of Ware and Young [2014] as narrative input affect the *industry approach*?

Research question 1.2. How does the usage of types, in the form of master objects, and explicit relationships affect the *industry approach*?

Research question 1.3. How can the algorithm of Merrell et al. [2011] be adapted in order to used in the narrative world creation process?

Research question 1.4. To which extent is the proposed *assisted approach* capable of assisting the designer in making decisions compared to the *manual approach*?

In answering the above research questions, the following main contributions were made:

1. A manual approach to the narrative world creation process that uses the PDDL-format of Ware and Young [2014] and, master objects and explicit relationships through the use of the semantic model of Tutenel [2012].
2. An assisted approach to environment filling using *evaluation of options* for both the discrete and the continuous domains.
3. A usability study of and conclusion on the capability of the two approaches to support the designer in the narrative world creation process.

1-4 Thesis structure

The remaining chapters in this thesis are structured in the following manner. First, Chapter 2 deals with the background and related work in the fields of narrative representations, semantics, mixed-initiative background, mixed-initiative approaches and automation of environment filling. Secondly, Chapter 3 discusses the manual approach to the narrative world creation process. This is followed by Chapter 4 that discusses the assisted approach. Chapter 5 discusses the important details regarding the implementation of a prototype to test the manual and assisted approaches. The method and results of the usability study are discussed in chapter 6. Lastly, the thesis concludes in chapter 7 with the future work being discussed in chapter 8.

Chapter 2

Background & related work

This chapter discusses the background and related work of the manual approach and the assisted approach. For the manual approach, narrative structures and semantics are discussed. The narrative and semantics also include looking at potential useful information for the assisted approach. For the assisted approach, the mixed-initiative background, mixed-initiative approaches, and different assistance techniques are discussed.

2-1 Semantics

Semantics are used to model a modular approach to objects. The goal of a modular approach is to reuse art assets by creating assets that are instantiated and define relationships on how they can be combined [Perry, 2002]. For environment filling there is no known industry standard, but Tutenel [2012] provides an interesting approach that describes the usage of a semantic model to create game worlds.

Tutenel [2012]'s approach states the following aspects that can be expressed by designers using their semantic model:

- What a geometric model actually represents: what type of object it represents, what classes it belongs to, etc.
- The essential characteristics of the objects in the game world.
- The way a player (and other characters) can interact with the game world and its objects.
- How objects relate to each other. For example, how they are placed relative to each other.
- How the objects behave over time, possibly influenced by other objects in the world.

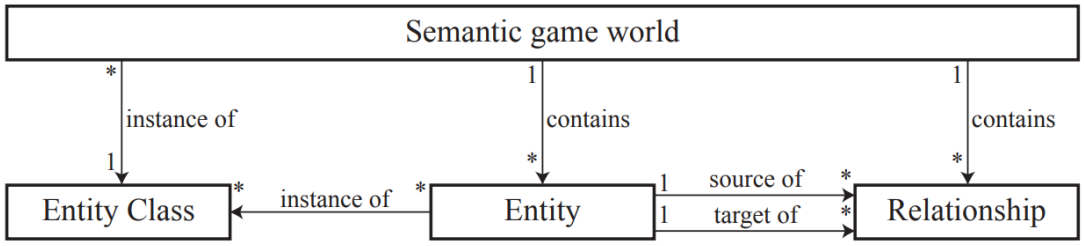

Figure 2-1: The relationships between semantic game world, entity class, entities and relationships. Semantic game world is an instance of an entity class, and contains multiple entity classes and relationships. Entities and relationships are connected with one-to-many relationships. (Source: Tutenel [2012])

Tutenel's semantic model is meant to provide a single, centralized and consistent knowledge base. Secondly, it allows people working on a game, to have a better understanding of the game world and to have a more expressive language at their disposal. Before looking at how the modular art set was created, first, the semantic model is discussed.

The semantic model uses three levels of information: Intra-object semantics, object interrelationships, and world semantics. Intra-object semantics are characteristics to a single object, and that object only. Object interrelationships are relationships between two or more objects. World Semantics is information that is bounded to some objects but holds for the entire world.

The main concept of the semantic model is the Semantic Game World. This contains entities, which are instances of entity classes, and relationships between these entities. Each entity class can have characteristics, which is described using attributes and behavior, which is described using services. An entity class can not only represent objects, but also abstract entities or even intra-object semantics. For example, the semantic game world itself is an instance of an abstract entity class. See Figure 2-1 for an overview.

Entity classes support a parent-child relationship that allows inheritance of characteristics and behavior. Two kinds of entity classes are distinguished: physical object classes and abstract entity classes. The abstract classes are used to describe entities that do not have a physical presence. This can, for example, be used to describe a government being part of the world. Physical object classes represent entities that have a physical presence in the world. Physical object classes can be composed of other physical object classes to create compound objects. The physical object class can be further split up into tangible object classes and spaces. Spaces are regions bounded by a 3-dimensional shape that can contain instances of physical object classes. Tangible object classes are made up of a single type of matter. Comparable to the real world, each tangible object is composed of a particular matter. Each type of matter has particular characteristics, like wood being flammable. To better emulate matter, three subcategories are used: substance, mixture, and material. Substances consist of either pure chemical elements, combinations of multiple elements or other substances. Mixtures are two or more substances blended together and materials are the composition of multiple types of matter. An example of a substance is water with milk being the mixture which water can be part of. An example of materials is a raw material such as cotton or ore.

Entity classes, relationships, and matter can also have attributes. Attributes are either numerical, boolean, string, vector or a formula in terms of other attributes. Services are meant

to simulate particular functions of entities. Services allow an entity class or matter to perform an action based on a context. An action is a process, performed by an entity class, resulting in values of attributes to change or yielding new instances of entity classes. Actions make use of context to define preconditions. The described action pattern has also been seen in narratives in the previous section, see Figure 2-5. Combining semantic actions with narrative events would make it possible for the narrative to influence the resulting environment. Relationships are used to connect entities with each other. A relationship is a connection of a particular type that associates two or more entities. Game-specific classes are used to link content, such as 3D models or textures, to tangible object classes. This makes it possible to render the created worlds. A context is also able to define implicit relationships between entities.

The previous paragraphs introduced semantics in general, the upcoming paragraphs will discuss how to create a semantic database which supports a modular approach to object instancing. Entika [Tutenel, 2012] was developed to test out the semantic model using several applications. Some were described in Tutenel [2012], while others were published separately Lopes et al. [2012]. Several examples of the databases used for these publications are available to us. One is interesting as it showed a filled in semantic model aimed to do layout solving. By looking at how the description was approached, more can be learned about what it entails to describe a world using semantics. Unfortunately, there are only snippets of explanation in Tutenel [2012]. The requirements discussed at the beginning of this section are of value here, as they describe what a semantic model should meet to effectively use it to create a game world. Some of these focus specifically on gameplay elements, which are not of interest to us, while others are useful for narrative world creation. While it is known what the requirements and purpose of different elements of the semantic model are, it is unclear how these are used to formulate a semantic world. To determine this, an analysis of the example is done below.

The one example that stood out was one called 'gamesimple.edp' and it's related publication is unknown. But it does host the spaces and relationship types as described in the layout solving approach by Tutenel [2012]. The semantic database, which is accessed through Entika, hosts both the information for filling the environment as well as the information required for game mechanics. This makes it difficult to differentiate between the two. The most important parts of the semantic database are discussed in regards to layout solving.

The tangible classes describe both the location types as well as the objects that can be placed within. The hierarchical structure of the tangible classes is shown in Figure 2-2.

On the top level, the different location types exist as well as a 'Physical object' class that is the parent of the physical objects. The physical objects are further divided into (partially) penetrable implicitly as a child of the physical object class and fully impenetrable as an explicit class. The physical object entity defines a position, location and rotation attribute, as well as the following spaces:

- back
- bottom
- center
- clearance
- empty when placed

```
office
scene
store rack
Physical object
        Building lot
        Field
        ...
        Fully impenetrable object
                arena
                couch
                crate
                ...
```

Figure 2-2: The hierarchical structure of the tangible objects as defined in the 'gamesimple' Entika database used for layouting.

- front
- left
- middle line X
- middle line Z
- off limits
- off limits except for parts
- right
- top

Each attribute and space defined by a parent is inherited by its children. The 'fully impenetrable object adds' an off-limits space that covers the space that the object will take. There are three types of shapes: normal, off limits or clearance. Off-limit shapes do not allow any other shapes to overlap with them. Clearance shapes allow overlap with other clearance shapes, but otherwise no overlap. Normal shapes do not infer any restrictions on the overlap. The spaces not named clearance or off limits fall under the normal type and their purpose is to describe the different faces that can be used to connect with relationships. Two special kinds of space are also defined: Store shelf space and the world space. The world space is used to describe the encompassing world that holds all other spaces. The store shelf space allows the placement of objects on spaces that are created when the 'store rack' class is created. This pattern can be used for other similar kinds of objects.

The abstractions offered by the semantic database consist of attributes and relationship types. The attributes already mentioned are the ones employed by the physical object class. Other attributes are used by relationships to help quantify their value during layout solving. These attributes are range max, range min, angle offset, height, and radius. These relationship types are all children of the 'placement' relationship type indicating their usage in layout solving. These are used in some of the following relationship types.

- Above, uses height to indicate the height that the tangible class should be above another tangible class.
- Against.
- Around, uses radius to indicate how close around the tangible class the other entities should be placed.
- Facing.
- On, uses angle interval and interval size, but it is unknown what their exact purpose is.
- Parallel.
- Follow on, uses range min, range max and angle offset to dictate the range in which the next tangible class should be placed and the angle offset at which it should be placed.

2-2 Narrative representations

The goal of this section is to compare different representations and figure out the best one with the goal of incorporating narrative information in the narrative world creation process and in the automation described in the proposed solution. Before looking at individual narrative representations, first, some general background is discussed. The field of computational narratology encompasses, among other things, the creation and telling of narratives in all shapes and sizes. The distinction between creating and telling is important as it requires narrative information to be stored before it is told. This distinction has been the source for several narrative models among which are the bipartite model, which is eloquently described by Kybartas and Bidarra [2016] and the tripartite model as first mentioned by Genette [1983]. The aforementioned models dictate that a *narrative* consists of *story*, *discourse* and, in the case of the tripartite model, *narration*. These models look at a *narrative* like they consist of multiple layers with each layer holding information on a specific part. In both models, the story is used to describe the underlying story by listing the characters, objects, and events that occur. Discourse holds information on how the story should be adapted to be told in the specific instance. For example, the story of red cap can be told (also known as *realization*), by only adapting the *discourse*, using text, video, telling the story from a different perspective, etc. In the tripartite model, the discourse layer is split up into medium independent information like perspective and medium dependent information like text or video adaptations. The creation of a narrative world is the realization of a narrative to a 3D digital world. Following the tripartite model, our focus will be on realizing the story part, which provides a first step to realizing all aspects of a narrative, as our time is limited and it provides a first step to realizing all aspects of a narrative.

Story generation is often seen as a planning problem Weld [1994], which are known from their applications in AI. In some planning problems, characters of the story are seen as actors with goals, beliefs and available actions Hindriks et al. [1999]. The planning problems are solved step by step. An action is selected for which the preconditions are met, after which the effects of this action are applied to the world. There are several narrative representations known for the input and output of these algorithms. The high profile ones are: the Partial

Order Causal Link (POCL) [Riedl and Young, 2010], Story Intention Graph [Elson, 2012] and Universe [Lebowitz, 1984, 1985, 1987]. Each of these represents the narrative world using actors, objects, and actions, which are selected to form a series of events that is the resulting story. For clarity, the planning algorithms used are of no interest to us, it is the input and output which we are interested in. This results in the POCL representation being favored as an implementation by Glaive [Ware]. Using PDDL resulted in an easily readable representation, with examples available.

The POCL plan, as described by Riedl and Young [2010], is the basis for an expansion that incorporates intent to improve the planner. Next, to describing the inner workings of the planning algorithm, the paper also describes what will be the end result. It describes the usage of a POCL plan during and after planning. The formal description of the POCL plan is as follows:

Definition 1. A POCL plan is a tuple $\langle S, B, O, L \rangle$ such that S is a set of plan steps, B is a set of binding constraints on the parameters of the steps in S, O is a set of temporal orderings of the form $s_1 < s_2$ where $s_1, s_2 \in S$, and L is a set of causal links of the form $\langle s_1, p, q, s_2 \rangle$ where $s_1, s_2 \in S$ and p is an effect of s_1 and q is a precondition of s_2 and p unifies with q.

The paper provides several examples of resulting POCL plans. Although the formal definition makes it quite clear what information is available, the graphical representations as shown in the paper do not offer much help in making the output usable by algorithms. This is also something which other papers overlook when saying that they use a POCL plan as input, for example by Hartsook et al. [2011], as it is unclear how exactly they define the input. One system that implements POCL is Glaive, it also uses PDDL for both input and output Ware. Unfortunately, any publications [Ware, 2012, Ware and Young, 2014] on the matter only discuss the approach and implementation of the planning algorithm used and not discuss how PDDL was adapted to host the result. The download of Glaive does come with examples of resulting stories which can be analyzed to determine how they represent the story using the domain and problem definitions of PDDL, if they adjusted anything and how they approached the storage of the resulting plan.

Before going into the examples, some background on PDDL is required. Common ways to represent the problems for AI planning problems are STRIPS, ADL and later on PDDL. The goal of PDDL was to standardize the way planning problems were described, this representation was first used in the 1998/2000 International Planning Competition (IPC) [ica, 2017]. Over the years, the PDDL standard has seen various changes and updates and is currently at version 3.1, but planners often use variations of this standard or do not support all parts of it. The basis of PDDL is the distinction between the domain definition and the problem definition [Haslum]. The domain definition contains the predicates and actions available to the planner, it also declares types and static facts. Each action may have preconditions and effects. The problem definition is used to declare the starting state and the goal of the problem. AI planners use these two definitions to solve the problem. In AI planning, the purpose of the solver is to reach the end goal. The steps taken to reach that end goal, also known as a plan, are only used for debugging purposes. In narratives, the steps taken are essential to retelling the story, which requires storage of these steps.

The download of Glaive comes with a total of 6 stories, each with a domain, problem and solution description. The problem and domain structures are best described by Haslum as

```
(define (plan aladdin-cave-solution)
  (:problem aladdin-cave)
  (:steps (fall-in-love king jasmine castle)
          (order king hero castle (loves jasmine king))
          (go hero castle mountain)
          (slay hero dragon mountain)
          (pillage hero dragon lamp mountain)
          (summon hero genie lamp mountain)
          (command hero genie lamp (loves jasmine king))
          (love-spell genie jasmine king)
          (marry king jasmine castle)
          (appear-threatening genie hero mountain)
          (slay hero genie mountain)))
```

Figure 2-3: An example of the output of Glaive, which is bundled with the Glaive download. It shows the definition of the plan, associated problem, and list of instantiated actions.

```
(:types character thing place - object
        male female monster - character
        knight king - male
        genie dragon - monster
        magic-lamp - thing)
```

Figure 2-4: An example of the basic types used in the Aladdin story bundled with the Glaive download.

they are not changed in Glaive's examples. Each of the solution descriptions defines the plan, with steps that were the output of the planning algorithm, and the problem it is associated with. Figure 2-3 shows an example of a solution of Glaive. Each step first lists the action instantiated, followed by the arguments of that action. The "fall-in-love" action shows a basic action that uses two characters as arguments combined with the location where this occurs. The "order" action shows a different set of arguments with an instantiated predicate as one of the arguments. Instead of just updating the current beliefs, these can be used to describe intentions.

As described by the PDDL definition, the domain file describes the requirements of the file, the types available, the predicates available and the actions available. The types used in each domain differ among the different examples. This makes sense as Glaive is a narrative planner that does not rely on specific types. Our approach will most likely rely on a set of standard types that allows the differentiation between characters, things, and places. The 'aladdin-domain' shows examples of standard types, which can be seen in Figure 2-4. The 'character' and 'thing' type can be used to describe objects found in the environment while 'place' can be used to describe the locations used in the story.

The predicates declared also vary per story as each story uses different descriptors. Where one story uses an 'alive' predicate, another does not. For all the stories, the most common predicates used are the 'at' and 'has' predicates, which are examples of predicates that describe a spatial relationship between an object and a location and between a character and an object. This spatial relation can be of use for our approach as spatial descriptions can be used as requirements.

The predicates and types are used in the actions, which are described using parameters, preconditions, effects, and agents. Parameters declare the types and names of the type instances used in the action. The preconditions declare the set of predicates should (not) hold. The effects declare which predicates are made true or false. The agents declare the

```
(:action travel
  :parameters   (?creature - creature ?from - place ?to - place)
  :precondition (and (alive ?creature)
                                        (at ?creature ?from))
  :effect       (and (at ?creature ?to)
                     (not (at ?creature ?from)))
  :agents       (?creature))
```

Figure 2-5: An example of the action 'travel', which changes the predicate 'at' from one place to another.

actionable agents of the action, the one that can use this action. The actions that use the spatial predicates can be seen as spatial actions, actions that affect the spatial description of the world, characters or objects. Other actions describe other types of descriptions of the world, characters or objects. An example of a spatial action is the 'travel' action, as can be seen in Figure 2-5. The 'travel' action shows a creature being at one place before the action, to being in another at the end of the action. This is done by changing the 'at' predicate from the original location to a new instance that includes the new location.

The problem definition describes the associated domain, the objects, the initial state and the goal state. The objects describe the instances of the different types. The initial state describes the predicates that are true at the start. The goal describes the predicates that at least have to be true, or false, at the end. Interesting about the initial state description is, that for each example, it describes where each object or character is at. This shows that spatial information is known and maintained throughout the telling of the story because each step triggers an action that influences the initial state description. Another interesting example is the 'heist' problem definition, as it not only declares the locations and the 'at' for each object and character, it also declares the connections between locations. These connections are used in the move action and are also spatial descriptions of the world.

The solution definition defines the actions taken to go from the initial state description to the goal state. Not much interesting can be said about the solution description as it is just a list of instantiated actions. It is time-consuming but possible, to determine whether the listed steps will result in the described goal state. This means that when loading a solution like this, one must trust, to a certain extent, that the solution provided is valid. Several examples of instantiated actions can be seen in Figure 2-6. The actions shown in the figure rely on the actions to update the information known about the world. Some actions may not use a location in their action, which is thus not included in their description. This is not necessary for Glaive, but might be of added benefit to make locations more explicit when regarding the creation of narrative worlds.

In conclusion, although there are quite some variations between the different problem and domain descriptions, the PDDL implementation by Glaive offers interesting possibilities to combine with semantics. This offers possibilities to extract interesting information and incorporate these into the automation. For the automation to effectively rely on the descriptions some standardization is necessary, which has already been touched upon by Glaive. Secondly, the spatial descriptions provided by predicates require exploration to determine the different types and the different ways these can affect the creation of a narrative world.

```
(:steps (fall-in-love king jasmine castle)
        (order king hero castle (loves jasmine king))
        (go hero castle mountain)
        (slay hero dragon mountain)
        (pillage hero dragon lamp mountain)
        (summon hero genie lamp mountain)
        (command hero genie lamp (loves jasmine king))
        (love-spell genie jasmine king)
        (marry king jasmine castle)
        (appear-threatening genie hero mountain)
        (slay hero genie mountain)))
```

Figure 2-6: Shows the list of instantiated actions of the Aladdin solution.

2-3 Mixed-initiative background

Horvitz [1999a], Horvitz [1999b], and Liapis et al. [2016] give excellent insight into the mixed-initiative approach. Horvitz [1999b] introduces the term mixed-initiative which he describes as:

> "Mixed-initiative is not just whether an aspect of part of the process is automated, it is a natural interleaving which allows many different layers on which decisions can be made by either user or service."

Liapis et al. [2016] expands on this definition:

> "The human creator and the computational creator "take the initiative" in mixed-initiative systems. However, there is a sliding scale on the type and impact of each of these creators' initiative."

Liapis et al. [2016] reason that the extremes are not that interesting to look at, as this is either a configurable algorithm, which has barely any human initiative or, as an example, checking spelling, which has barely any system initiative. Their focus is more on two types of mixed-initiative that are commonly used: either computer-aided design or interactive evolution. In computer-aided design, humans have the idea and the computer supports their creative process. In interactive-evolution, the computer creates content and humans guide it to create content they prefer. Computer-aided design is often used whenever the process can be aided by inserting computationally supported, powerful and creative tools. Interactive evolution is used in domains where designing a fitness function is difficult and where a naive measure may be more harmful than helpful.

Mixed-initiative systems must consider the following: when to engage users with a service, how to best contribute to solving a problem, when to pass control of problem-solving back to users, and when to query a user for additional information to minimize uncertainty. Horvitz [1999a] gives excellent insight into the different aspects that should be regarded when deciding upon a mixed-initiative approach. Mixed-initiative has come forth from discussions among human-computer interaction researchers as an approach that lies between developing new kinds of automated services and new kinds of approaches to direct manipulation by users. The following quote by Horvitz [1999a] sums up the goal of a mixed-initiative approach:

"Surely, we should avoid building complex reasoning machinery to patch fundamentally poor designs and metaphors. Likewise, we wish to avoid limiting designs for human-computer

interaction to direct manipulation when significant power and efficiencies can be gained with automated reasoning."

While describing mixed-initiative systems it is important to know that an agent is an algorithm that runs continuously and has the purpose to decide on whether, when and which action to perform. This means that mixed-initiative systems rely on an agent, or agents, to handle the human-computer interaction. The actions available to the agent rely on the available automated systems. A simple example is the invocation of an automated service. The following factors are summed by Horvitz [1999a] as the critical factors for the effective integration of automated services with direct manipulation interfaces.

1. Automation should provide significant added value over direct manipulation solutions.
2. A mixed-initiative system should consider uncertainty about a userâĂŹs goals.
3. Agents should employ models about the attention of users and consider the costs and benefits of deferring action to a time when action will be less distracting.
4. Agents should choose ideal actions using costs, benefits, and uncertainties.
5. Agents should employ dialog to resolve key uncertainties while considering costs.
6. Users should be able to directly invoke or terminate automated services in case of poor decisions by the agent.
7. Agents should minimize the cost of poor guesses, including appropriate timing out and natural gestures for rejection.
8. Agents can be of more value when they can degrade the precision of a service to match the current uncertainty. Doing less but doing it correctly under uncertainty can result in valuable advances while minimizing undoing or backtracking.
9. Agents should be designed such that users can easily complete or refine an analysis provided by an agent.
10. An agent should default to tasteful behaviors and courtesies that match social expectations.
11. Agents should maintain a memory of recent interactions and provide mechanisms to reference objects and services in this short-term memory.
12. Agents should learn more about working with users while interacting with them.

Liapis et al. [2016] lists the following factors when creating a mixed-initiative tool in general:

- Who is your target audience?
- How can the method for control over content be balanced?
- How to resolve conflicts that arise due to the human stating conflicting desires?
- How expressive is the system?

- Can the computer explain itself?

For computer-aided design the question "can the computer explain itself?" is more important than for interactive evolution as the tools given to the user should have direct and explainable effects, while the solutions generated by some interactive evolution approaches are unexplainable and only become explainable when the user adds constraints, which act as an explanation.

When looking specifically at computer-aided design, the core problem to solve is: What novel and useful editing operations can be incorporated? This is the core problem as these editing operations make up for the toolset the user can use to create the solution that he wants. These editing operations must be useful, as otherwise, the process would be cumbersome. The demand for novelty is assumed because the mixed-initiative process is often designed instead of the already existing process which is cumbersome.

For interactive evolution, the core problem to solve is the occurrence of human fatigue. This can occur when users are required to perform a large number of content selections, when feedback from the system is slow, when the users are simultaneously presented with a large amount of content on-screen, or when users are required to provide very specific input. This can be reduced by providing effective ways to sort through or reduce the options provided to the user.

Using these factors there remains a large space of design opportunities as described by Horvitz [1999b]:

- Developing automated services that are performed in line with a user's activity.
- Finding metaphors that provide efficient means for users and computers to communicate information about intended or ongoing contributions.
- Developing automated services that can vary levels of precision or completeness.

2-4 Mixed-initiative approaches

Examples of papers that explore this space of design opportunities are of help to determine what will and what will not work for narrative world creation. The following examples each create and/or fill a world using not just direct user manipulation or complete automation, but by a combination of both.

Sentient Sketchbook [Liapis et al., 2013] iteratively generates multiple options, based on the current input and some input parameters, from which the user can choose from. The selected option becomes the main solution and new options are generated based on that. The user is able to freeze parts of the solution such that these are unmodified in the generated options. This allows the user to steer the current solution towards an end solution.

Interactive furniture layout using interior design guidelines (IFL) [Merrell et al., 2011] employs a similar approach as it generates multiple options for the user to choose from. The difference lies in what is generated, for Sentient Sketchbook this was the entirety of user input, the addition, removal, and positioning, whereas for IFL this was just the positioning, as it requires

a set of classes to be known beforehand. It does, however, handle a continuous space whereas Sentient Sketchbook used a grid to discretize and decrease the complexity. Similar to Sentient Sketchbook, the user can freeze elements of the current solution so that they are unmodified in the generated options.

Sketchaworld [Smelik et al., 2010, 2011] uses abstractions of actions that each applies an algorithm when performed. The user selects one of the abstractions and provides the input required. The automated service for that abstraction adjusts the world according to the input. This simplifies the multiple actions required to a single action for the user which he can directly control. This shows the power of good metaphors for communication of intent.

Tanagra [Smith et al., 2010, 2011] is a mixed-initiative level design tool which allows the creation of 2D platformers. It uses an underlying, reactive level generator to ensure that all created levels are playable. It also allows for the rapid generation of many different levels using the specifications of the user. The level is iteratively refined by the user, which places and moves level geometry, as well as adjusts the pacing of a level. They use an agent to determine when and what adjustments should be made, as well as dictate when new solutions should be generated. The agent can be seen as a subordinate assistant, as it can suggest different potential designs and obeys the designers' commands.

Liapis et al. [2012] describes a system that employs a choice-based interactive evolution approach. The user creates an initial map that is used to generate variants and to estimate the user's intentions. These intentions are updated when the user chooses among the generated variants. This process does not include an agent and is more akin to an interactive evolution approach.

Although a limited sample of mixed-initiative approaches was presented, these all had world creation as a focus. There are much more approaches in different research areas, but these were not discussed, as the discussed approaches serve as samples to spark creative ideas on approach mixed-initiative for narrative world creation and not cover the entire mixed-initiative research area. Tanagra showed, as the only one, an approach that includes an agent that reasons about uncertainty and invokes automated service. All others provide automated services and a simple, hand-off interaction approach. Although the results of these other approaches have been positive, this simplified interaction does not achieve the full potential of a mixed-initiative solution. However, the design and implementation of an agent capable of reasoning under uncertainty is outside the scope of this thesis but would be an interesting future research direction.

To reflect on the vision provided by several papers on mixed-initiative, first and foremost, there are two, general, basic approaches to mixed-initiative: Computer-aided design and interactive evolution. For environment filling, interactive evolution seems a good fit, as the process is mostly about decisions and thus do not offer many possibilities for novel and useful editing operations. Interactive evolution would allow exploration of the space of possibilities while still allowing control for the user. A common interaction used for interactive evolution is a suggestion based interaction. This principle was applied in Sentient Sketchbook Liapis et al. [2013]. Next to these two approaches is the addition of an agent that can, in the case of interactive evolution, further help out the user.

The 12 aspects mentioned in Horvitz [1999a] mostly concern the creation of an agent, with some being generally applicable to the interactive evolution approach. An interesting aspect

is "An agent relies on good and effective automation", which makes us believe that figuring out a good approach to mixed-initiative is as important as determining a good approach to automation. Environment filling will be focused on, which is supported by the reliance of an agent on good automation. The aspects include several demands regarding automation, which result in the following list:

- Automation should provide significant added value.
- Automated services need to be able to be performed in line with the user's activity.
- Automated services should offer varying levels of precision or completeness.
- Automated services need to communicate properly about the information that they generated.

2-5 Automation approaches

As discussed in the mixed-initiative section above, providing significant automation is important to make sure that a mixed-initiative approach can work. Two already discussed papers provide ways of automating a part of the process. Tutenel [2012] provides an automatic solving approach to adding and placing an instance inside a scene and Interactive furniture layout using interior design guidelines (IFL) [Merrell et al., 2011] provides an automation to just the positioning of instances.

The automatic solving by Tutenel [2012] uses procedures to determine which tangible classes are to be placed and in what order. The procedures are created using a semantic description language. This semantic description language allows designers to describe what particular types a scene consist of. The procedure is read by the layout planner which determines the next class to instance. This planner provides the solver with new instances, which the solver then finds an orientation and position for. The planner works using a list of statements and rules that need to be executed in order (the procedure). The following types of statements are available for a procedure:

- Pick statements query the classes using attribute values, services to provide, the matter they should consist of and predicates they should contain as conditions.
- Place statements trigger instancing of the last picked class.
- Conditional and loop statements offer if-then-else and for/while loop functionality.
- Backtrack statements force the algorithm to backtrack a given number of steps. This gives the possibility to redo a part of the placement if no suitable locations could be found.
- Execute procedure statements allow the construction and usage of sub-procedures, which allow hierarchically built scenes.

To create a procedure, a description of the scene is required. This is done using a semantic scene description language. This language allows designers to specify which and how instances should occur in every scene of a given type. This semantic scene description language can be a way to describe the requirements of the narrative to the environment. Description entities are used to define which instances need to be present and how they should be placed. Each description entity consists of two components: an instance component and a placement component. The instance component offers a detailed description of which types of instances, which classes, need to be added. The placement component defines scene-specific relationships that the instances need to adhere to. A collection of description entities makes up a semantic scene description, which can be converted to a procedure. This involves three steps: sort the description entities and add them to the procedure, encompass these procedure statements with loops to handle amounts defined in the description entities and with conditional statements to handle context-specific operations, and perform an optimization to ease the workload of the solver when executing the procedure.

Sorting the description entities is done by creating a dependency graph where tangible classes are the nodes and every placement relationship between two tangible classes as a directed edge. Edges point towards the node on which the other node depends. The next class to order is determined by the following four criteria:

- Class with least outgoing edges to nodes of unordered entities
- Class with most incoming edges
- Class with most outgoing edges to nodes of ordered entities
- Class with the largest size

Circular references cause conflicts during sorting, this is fixed by picking one of the classes that created the circular dependency using the above criteria and remove the outgoing relationship that creates the circular dependency. After sorting, the control statements are added. Loops are added using the amount defined and context-dependent elements are encompassed by conditional statements. As the last step, some extra rules are applied that guide the choices for classes to pick. For example, an object query can return classes of varying sizes. A rule is added that focuses on the lower spectrum of classes to pick from when adding a small scene and from the higher spectrum when adding a large scene.

Although Tutenel [2012] describes the planning part of the process in great detail, the placement of the instance is not discussed much. Tutenel et al. [2009] goes into more detail on how the known relationships and tangible classes, and the relationships of the new instance are used to determine the location of the new instance. The relationships used can either be explicit or implicit. An example of an explicit relationship is the 'facing' relationship, which was already mentioned in the semantics section, and is used to describe that a couch should be facing towards the tv. Implicit relationships are based on features, which, combined, make up the geometric representation of a class and thus all instances of that class. The feature types are the same as the space types. The 'off-limit' feature cannot overlap with any other feature and the 'clearance' feature can overlap with other 'clearance' features.

To find the location for a new instance all possible locations for a new instance are determined based on the ground type of the instance, its features, and the features of the already placed

instances in the current layout. The ground type is the feature type on which the instance can be placed. The ground type feature has a shape which makes up the surface for the new instance to be placed upon. This is trimmed using the 'clearance' and 'off-limit' features. This trimming phase also incorporates the possible orientations of the instance by using a discrete set of angles. The list of possible areas is further refined through the rules connected to the instance. The list of areas is cut into a grid of smaller areas, and for the center point of these areas, a list of geometric constraints are evaluated. The areas for which these constraints do not hold are discarded. To differentiate between the remaining areas, attractors and detractors between classes are used. These allow the possibility to favor one area above the other while not completely invalidating it. As the last step, an area is picked based on the different weights (including attractors and detractors). Subsequently, a random location within that area is picked to place the new instance.

IFL's positional generation algorithm uses two types of criteria: functional and visual. The functional criteria evaluate whether the layout supports the typical human activities. The visual criteria evaluate how the layout scores from the perspective of visual composition. The following functional criteria are supported by IFL:

- **Clearance**. Some furniture classes require clearance so that they can be properly accessed. This criterion allows classes to specify a clearance range and direction.
- **Circulation**. To support traversal through the room and accessibility to all furniture, circulation is required. The free configuration space is calculated using the space a person takes combined with the space that the furniture occupies.
- **Pairwise relationship**. These relationships allow the designer to apply manual constraints to furniture inside the room. For example, a "next to" relationship would allow a constraint for two pieces of furniture such that they will be positioned next to each other.
- **Conversation**. To support conversation in a normal tone of voice, the seats within a conversation area should be roughly four to eight feet apart.

And the following visual criteria are supported:

- **Balance**. Rooms are more visually attractive when the mean of the distribution of visual weight is at the center of the composition. A common assumption, that is applied here, is that larger classes carry more visual weight.
- **Alignment**. Alignment is the equal orientation of furniture instances relative to each other and to the walls of the room.
- **Emphasis**. Emphasis defines a dominant focal point for the interior point. This allows the human eye to rest on that focal point without being distracted by other dominant points.

Each criterion has a weight, that is set empirically, and is aggregated into a cost function. This cost function is highly multimodal and is not amenable to exact optimization techniques.

A Markov chain Monte Carlo sampler is used to explore the cost function. The Metropolis-Hastings algorithm is employed to explore the density function. The algorithm maintains a current configuration F and iteratively proposes a modified configuration that is either accepted or rejected. If the proposal is accepted, it becomes the current configuration. The algorithm iterates until its computational budget is exhausted. All accepted samples are retained for possible use as suggestions. The acceptance of a proposal move is governed by the Metropolis-Hastings acceptance probability. Rapid exploration, as well as precise modifications, are important, that's why three different proposal moves with equal probability are used. Two moves focus on precise modifications that make local adjustments, the first changing the position and the second changing the orientation of the selected instance. The third move allows a larger change by swapping two randomly chosen instances.

Tutenel [2012] and Tutenel et al. [2009], together, describe an approach that covers both the selection of a class and the placement of the instance. The selection of a class is done through procedures created by the user. Although these procedures offer detailed descriptions of instances and relationships and allow variation per location, they need to be created beforehand requiring the designer to know exactly how he wants the location to look like. Merrell et al. [2011] only addresses the placement of an instance. While the selection of the class to place is an interesting approach, it also relies a lot on the rules described beforehand. This makes both approaches difficult to incorporate into a mixed-initiative approach, but the underlying ideas can definitely be of use when looking at the discrete problems.

Regarding the continuous problem, the placement of an instance, Tutenel et al. [2009] discretized the areas and adds contractors and detractors to figure out the best areas to place the instance, followed by a random placement of the instance in the area. [Merrell et al., 2011] proposes a more advanced approach that uses the different criteria as contractors and detractors, including relationships and features similar to Tutenel et al. [2009]. This advanced approach does not discretize the area but solves it by exploring the continuous space using a Monte Carlo sampler. They are confined to a single area and also do not allow the placement of instances on top of each other.

Chapter 3

Manual approach to the narrative world creation process

This chapter discusses the manual approach to narrative world creation process while focusing on environment filling. First, an overview of the process is discussed, after which environment filling is looked at in more detail.

3-1 Process approach

Figure 3-1 shows the approach to the narrative world creation process. The inputs used by this approach affect how we approach the narrative world creation process. The two inputs used are the narrative input and the semantic database. The process is composed of two main phases, indicated by the thicker borders, called *environment creation* and *environment filling*. These steps are equal to the steps discussed in the introduction. Environment creation is about creating a space that is used to place objects inside. The space defines what is and what is not allowed to be filled. This space is directly connected to a location of the narrative. As to focus on environment filling, environment creation is kept simple. It is limited to relatively small size, it does not support any hierarchical structure, does not allow for height differences or multiple levels and the spaces are not combined into an overarching world. The other two steps are connected to the way locations and time are interpreted from the narrative. The narrative defines locations with each location having a set of events associated with it. The events occur linearly and thus each are a point in time, later referred to as a *time point*. Figure 3-2 shows an example visualization of some events, locations and time points of a narrative that demonstrates these properties. Given this definition of location and time points, the layout of each location has to be known for each location *and* each time point. To facilitate this, two steps are added to the process: *location selection* and *time point selection*. During *location selection* a location is selected from the list of locations and during *time point selection* a time point of a specific location is selected. The time point selection step facilitates each location adapting to the input narrative.

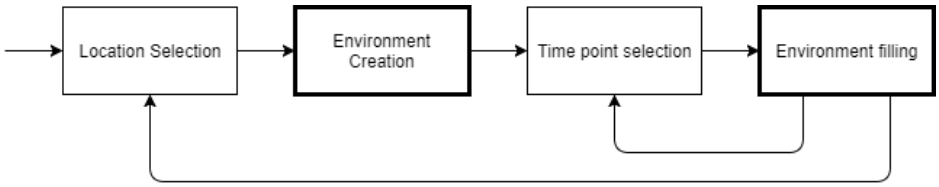

Figure 3-1: Shows an overview of the steps done during the creation process.

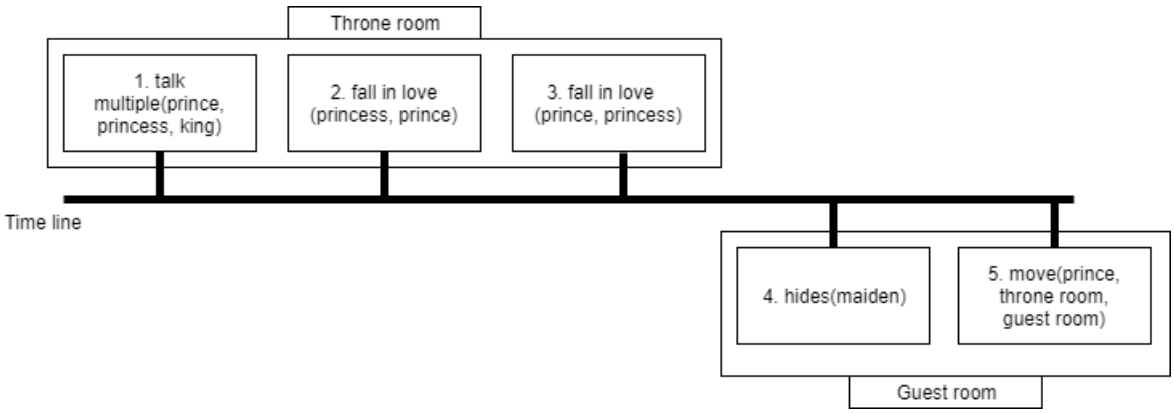

Figure 3-2: Shows an example narrative with several events annotated with time points and location types.

This way of looking at locations, events and time points does not incorporate all aspects as events can vary in duration and can occur in parallel. The approach is sufficient because our goal is to facilitate environment filling and research the assisting of it, and the adaptation of locations through the narrative is still part of the approach.

Figure 3-1 shows, in addition to the steps, several arrows that connect the steps. These arrows visualize the order in which the steps are taken. Starting with *location selection*, then *environment creation* and *time point selection*, and lastly *environment filling*. These follow the work flow of first creating and then filling an environment. When done filling an environment the designer either chooses to fill the same environment for another time point or go back to location selection and start creating and filling a new location. The designer determines when the environment is adequately filled, it is unknown what parameters the designer uses. As can be seen from the order of the steps, the created environment is not adaptable through the narrative as the selection of a time point is done after this.

3-2 Environment filling

As discussed in the introduction, environment filling is about filling an environment with objects, placing them inside the space created during environment creation. The result is a set of objects with a position and orientation, also called a *placement*. As input, the designer is presented with the created environment, which acts as a restriction, and a set of master objects to choose from. The master objects provides all available options.

To fill the environment a set of actions is needed. There are three actions: 'add', 'change' and

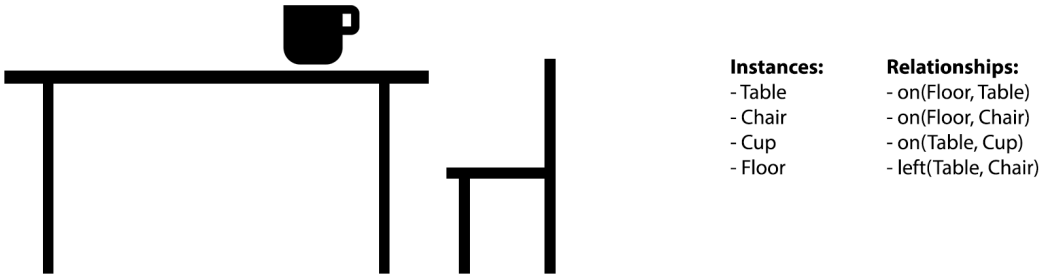

Figure 3-3: An example with multiple relationships and the source and target of each relationship.

'remove'. When filling an environment the designer performs these actions iteratively until the desired result is achieved. Each action consists of several steps that have to be performed. The steps are adjusted due to the usage of semantics to describe the master models, as explicit relationships add extra steps to some actions.

For the rest of this thesis, a relationship is shown as $X(A, B)$, X is the name of the relationship for example 'on', A is the source of the relationship and B the target of the relationship. Figure 3-3 shows an example of this, where the table has two relationships, it being *on* the floor and the chair being *left* of the table. The table object is both source and target of different relationships.

The designer is able to take two types of actions: selecting option(s) from a list, and determining the position and rotation of an object. The first is a discrete problem, as the number of possible options to choose from is limited, the second is a continuous problem, as the position and orientation of an object are continuous domains. Objects, relationships and other criteria decrease the valid positions, but as long as the valid space is not reduced to a single value, the domain will always be continuous.

Following are the specifications of the three actions. The added steps due to explicit relationships are highlighted with italics. Each step is annotated whether it is **D**iscrete or **C**ontinuous. The 'add' action consists of the following steps:

1. **D** Select an master object to instantiate
2. **D** *Select relationships, of selected master object, to instantiate*
3. **D** *Select objects that are to be combined in the selected relationships*
4. **C** Determine placement in environment

The selection of a master object requires the selection of a single item from a list of options, whereas relationship selection and the relationship instantiation require the selection of one or more options. Figure 3-3 shows examples of objects having multiple available relationships. Similarly, relationships may offer one or more objects when determining the other end of a relationship. Figure 3-4 shows the 'on' relationship having multiple objects to choose from.

The 'change' action consists of the following steps:

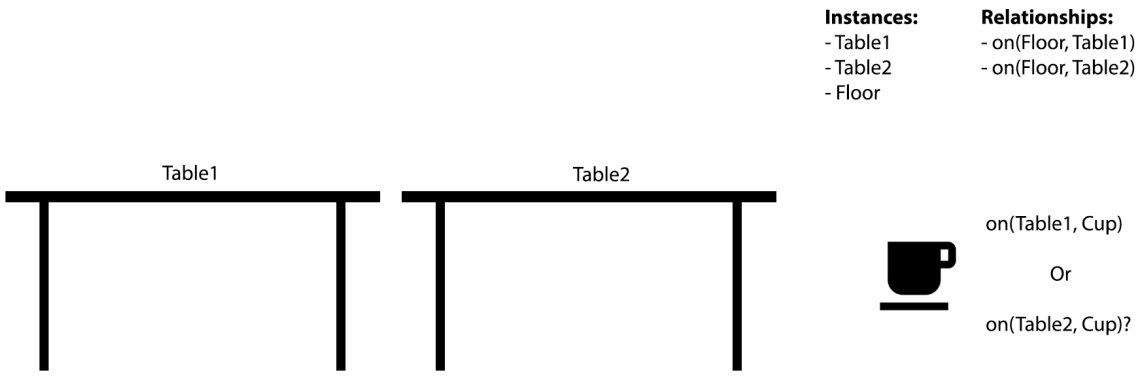

Figure 3-4: An example with the 'on' relationship having multiple objects to choose from.

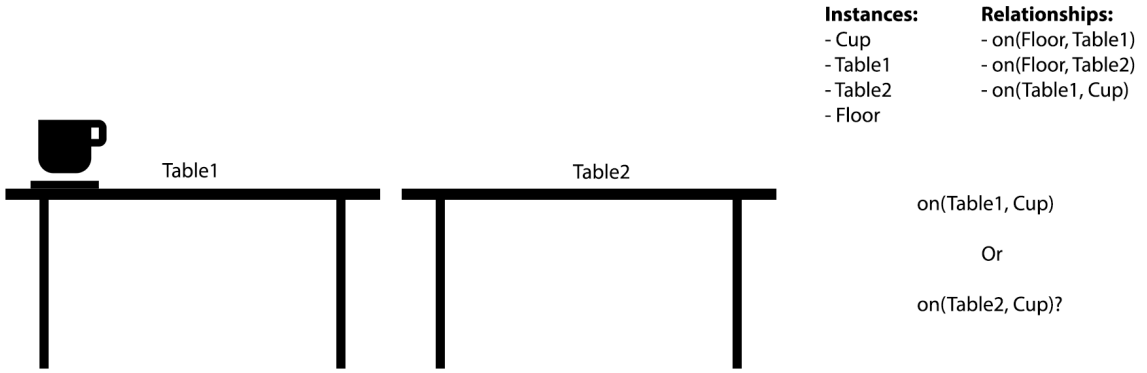

Figure 3-5: An example that shows the options that a single relationship can have when changing.

1. **D** Select an object
2. Change its attributes or *instantiate relationships*
   (a) **D** Change attributes and relationships
   (b) **C** Change placement of object

For the 'change' action, whether the choices are continuous or discrete depends on what is changed. Equivalent to the 'add' action, changing the relationships is a discrete problem, while changing the position or orientation is continuous. When changing relationships, the target and source of the relationship can only be changed to other already placed objects. Figure 3-5 illustrates this. With the cup on the table, changing its 'on' relationship offers two options for the source, the currently placed on table and the other table.

The 'remove' action consists of the following steps:

1. **D** Select an object
2. **D** Remove object from environment
   (a) *Remove all relationships of the objects*

Figure 3-6: An example that shows the result of removing the chair of Figure 3-3.

The 'remove' action offers only a discrete step. Although the relationships do not add steps, they cause the removal of all relationships of that object to be removed as well because each relationship misses either the source or target of the relationship. By removing the chair of Figure 3-3, both the 'on' and 'left' relationship are removed from the environment, of which the result can be seen in Figure 3-6.

Chapter 4

Assisted approach to environment filling

This chapter discusses the assisted approach given the manual approach of the previous chapter. First, a general approach to assistance is discussed in which the reasons behind separating the discrete problems from the continuous problems are explained. The approaches to the discrete and continuous problem are discussed briefly, with the specifics being discussed in separate sections.

4-1 General approach

To determine the best kind of assistance, it is important to consider the designer's view on the problems. The choices made in actions are approached quite differently by a designer compared to an algorithm. A designer looks at addition in its entirety when deciding, with each of the decisions as part of that larger decision. For an algorithm, these smaller decisions are often made individually due to computational complexity or the individual decisions are simplified as to allow the combination of the smaller decisions into the larger decision. This is even more the case when the solutions are to be used in a mixed-initiative system where the time available to solve a problem is quite limited.

An example of how separation may work is the way Semantic Game worlds [Tutenel, 2012] approaches layout solving. In this approach, the determination to add an object from and the placement of that object are solved separately. An example of the simplification is the discretization of the space as done by Sentient Sketchbook [Liapis et al., 2013]. By limiting the x- and y-axis to a grid, the possible placement positions are reduced from a continuous space to a discrete set.

For us, this comes down to either discretizing the continuous space as to make that, and thus the total, problem easier or to solve each problem separately. As to not limit the possible positions and orientations, each step of an action was assisted individually. Figure 4-1 shows

Figure 4-1: Discretized space used in Sentient Sketchbook. (Source: Liapis et al. [2013]).

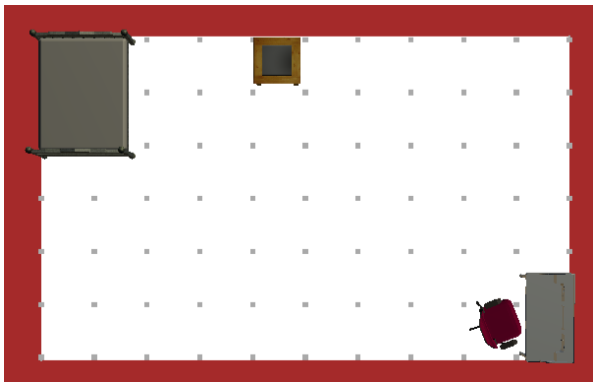

Figure 4-2: The implemented continuous narrative world creator.

the discretized space by Sentient Sketchbook, and Figure 4-2 shows the continuous space as was implemented for our approach.

Because of the individual assistance, each step is limited to the information provided for that step, it does not use the results of a follow-up step to aid its decision. For example, the selection of a master object does not look forward to the next step, relationship selection, and the possible results that that offers. It uses the possibilities that it offers, but not the results of that step. The possibilities are readily available through the semantic database, but the solutions require determination of the step. Assisting with each step individually requires multiple distinct approaches because there are multiple different types of steps. The types of steps that occur in the actions, are categorized as either D(iscrete) or C(ontinuous):

- Master object selection: selecting a single master object to instantiate. ('add', D)
- Relationship selection: selecting relationships that a new or existing object will have. ('add'/'change', D)
- Relationship instantiation: selecting the objects to fill in the gaps of the selected relationships. ('add'/'change', D)
- Object placement: figuring out the position and orientation of one or more objects. ('add'/'change', C)
- Object selection: selecting an object either to change its properties or to remove it. ('change'/'remove', D)

Two automation approaches were designed, the first being an approach for the discrete step types and the second for the continuous step type (object placement). Although a general approach is described for all of the discrete types, they each require a more specialized approach due to each having different information available and different information that is important.

The discrete steps are, in general, approached using *evaluation of options*. Evaluation of options is about quantizing the value that options have and, based on that, order the options. These aim to assist the designer in making a quicker and clearer assessment of the options available. The options are evaluated using weighted criteria, that quantize the value a certain option provides. Multiple criteria are used and combined in a total score which is used to order the list. The weights and criteria can easily be adjusted to fit with each step type and can be adjusted to steer the evaluation towards a specific goal. The ordering helps the designer choose between the options. Although the designer might disagree with the current ordering, it provides a basis to reason from.

For the continuous step type a single approach is used, the *generation and evaluation of options*. Due to the continuous domain, options have to be generated. During generation, the options are evaluated using weighted criteria. The generation algorithm is run multiple times and the resulting options are shown, ordered using the criteria, to the designer.

The following two sections provide details on the automation of the discrete problems, The discrete evaluation of options and the generation and evaluation of options for the placement.

4-2 Discrete evaluation of options

The criteria used for the *evaluation or options* are based on information derived from the narrative, the semantic database, and the current environment filling process. This section discusses the methods for extracting information from the different sources and the types of information that are available. The specific criteria and what information is used are discussed in Chapter 5.

The master objects and relationships of the semantic database form the basis of a dependency graph, with each relationship indicating the dependency of a particular master object on another. For each master object a node is created and for each relationship a directional edge. Figure 4-3 shows a basic dependency graph based on a semantic database.

Using this dependency graph, criteria calculate their result by checking each node and/or edge, fetching the data needed and in the end calculate its value based on the combined data. For example, for each master object, each edge in the graph is checked whether it uses this master object as a source or as a target and a counter is kept up to date. In the end the master objects can be ordered according to the amount of edges that use them. The dependency graph is enriched with other types of information, from the current environment and the narrative.

The narrative information will not, nor will the current environment, provide information on what can be used best as decoration. To fill this gap, an extra relationship was added between master objects and location types that defines a decoration weight. The decoration weight indicates, on a scale of 0 to 1, the importance of adding an master object to a location type as decoration. The use of this decoration weight is discussed in Chapter 5.

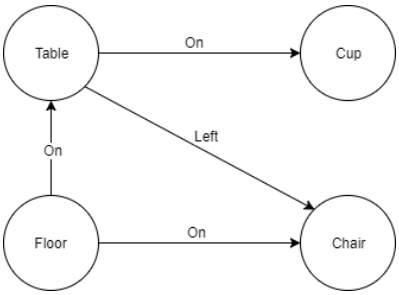

Figure 4-3: A dependency graph created from objects and their relationships as encoded in a semantic database.

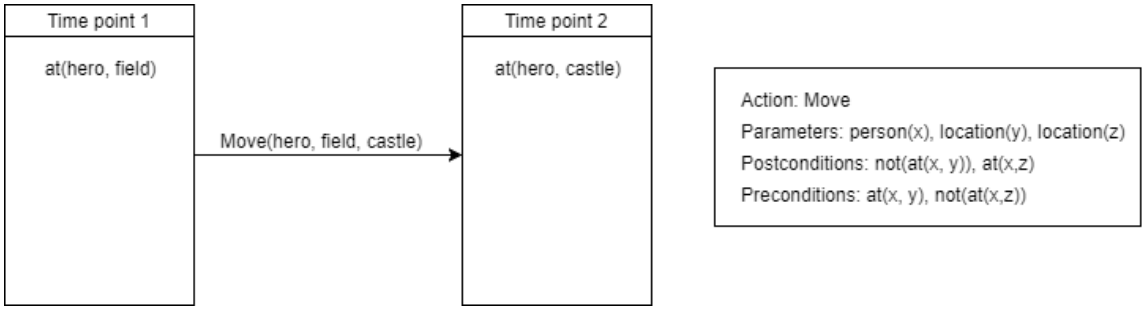

Figure 4-4: Two time points and their beliefs that are adapted through events that occur.

4-2-1 Narrative information

The events used in the narrative use preconditions and postconditions, these conditions put requirements on the environment which the event takes place in. These conditions are called beliefs and are formatted as predicates. The narrative includes an initial set of beliefs which are adjusted by the post conditions of events. This results in a set of beliefs after each event, and thus time point. Since events are connected to locations, the set of beliefs is also connected to a location for each time point. Figure 4-4 shows an example of beliefs changing because of an occurring event. The beliefs act as requirements on master objects or relationships for the locations' environment. For example, the $at(X, Y)$ predicate dictates that, given a certain time point, master object X should be placed in the environment of location Y.

The following types of beliefs can be defined in the narrative:

- A belief that requires/disallows one or more master objects to be instantiated
- A belief that requires/disallows one or more relationships to be instantiated

4-2-2 Current environment information

For the current environment, the main information is the objects and relationships that are placed in the current environment. Additional information is, given a certain object, what relationships have not be instantiated yet. A subset of those relationships are available relationships. Available relationships are relationships that have not been instantiated, but

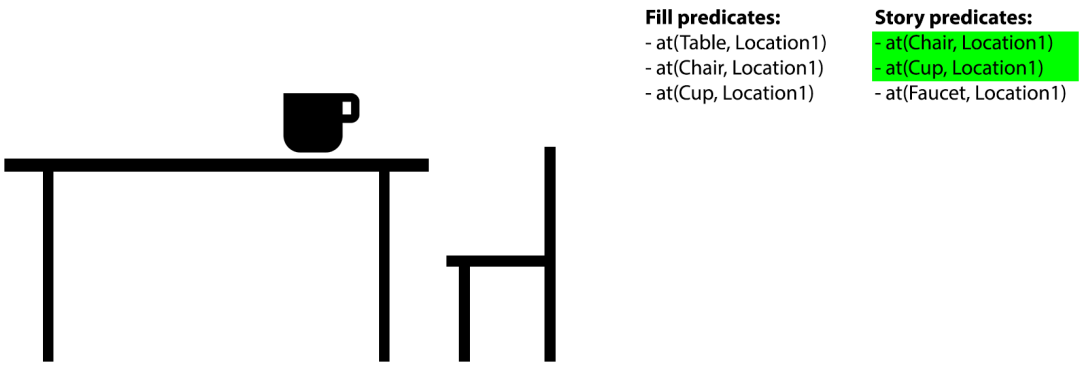

Figure 4-5: The predicates a narrative requires and the predicates caused by instantiating master objects. In green the predicates that overlap.

that could be instantiated. For an available relationship, either the source or the target, is already instantiated and the current looked at object/master object is the other end of that relationship. This information can be translated into predicates that can be compared to the believes of the current time point. Which believes have and which believes have not been fulfilled yet.

For example, the instantiation of a master object, e.g. 'cup' in a location called 'study room' results in an 'at' predicate being generated that looks like this $at(cup, studyroom)$. This predicate is then compared to the beliefs of the time point and when a belief is equal to the created predicate, mark the narrative predicate as fulfilled. Figure 4-5 shows a visualization of this example.

Predicates allow many aspects of the narrative and the semantic database to interact but this requires standardization of predicates. This standardization is to make sure that the beliefs of the narrative and the instantiation of master objects and relationships result in the same predicate types. For the criteria, the information gained is combined with the dependency graph. For example, an unfulfilled 'at' predicate results in the master object of that predicate to be tagged as 'required by the narrative' in the dependency graph.

4-3 Generation and evaluation of options in the continuous domain

We adopt the approach of IFL [Merrell et al., 2011], discussed in Chapter 2, to generate options. It fits well with the placement step as it offers a solution to determine the placement, both position and orientation, of each object in an environment. IFL uses several criteria to evaluate the solution during generation. Not all of these criteria are applicable in our approach as the focus is not on interior design layouting and the design guidelines that were incorporated cause some criteria to be too specific. Environments can take many shapes, sizes and themes, not just interior design. To support this wider variety of environment, the criteria were adjusted to be more general. Some adjustments were also in order to make the criteria work with the semantic database. The used criteria are discussed in the Chapter 5.

Each location type has its own characteristics. In IFL's approach, the weights that accompany these criteria were set empirically, in our approach these weights may vary between location

types and are affected by the unknown requirements set by the designer. To support placing objects on top of each other, something not done by IFL, each horizontal surface is solved separately. This also means that the weights can be adjusted for each horizontal surface separately.

The approach by IFL is able to figure out a new position for each object in a space, not just the newly added one. This has two effects. First, during addition, the placement of an object incorporates the movement of all other objects. This means that there needs to be a way to select objects and designate them to not be changed. This is done by incorporating a freeze action equal to the IFL's freeze action. When an object is frozen it can not be moved or rotated during the generation of options.

Chapter 5

Implementation details

This chapter discusses the implementation details of the prototype developed to test the difference between the assisted and manual approach. It starts with discussing the general requirements of the system, after which an overview of the system is given, followed by the implementation of the different parts.

5-1 System overview

The prototype's main goal is to allow the testing of the manual filling of an environment, the *manual approach*, versus the assisted filling of an environment using the proposed automation to order options, the *assisted approach*. The focus is on environment filling and allows us to leave out parts of the narrative world creation process as they would distract from testing these two methods and due to the scope of the narrative world creation process, some functionality is simplified. Time was spent on providing the other working parts of the narrative world creation process, which is able to accept different narratives, can work with different semantic databases and provides the UI necessary to create a narrative world. This is to the detriment of the implementation of the automation, which affected the quality of the results that the automation is able to provide. The reasoning behind such simplifications is discussed in this chapter. The resulting effect of these simplifications will be discussed with the results of the usability study in Chapter 6.

The system consists of three different components, shown in Figure 5-1. All components were implemented during this thesis. The semantic database implementation already existed.

- The *Narrative parser*, which reads in and parses the information of the narrative
- *Automation*, which facilitates the automation during environment filling
- The *Narrative world creator*, which is the main application that combines the semantic database with the other components to make a narrative world from scratch.

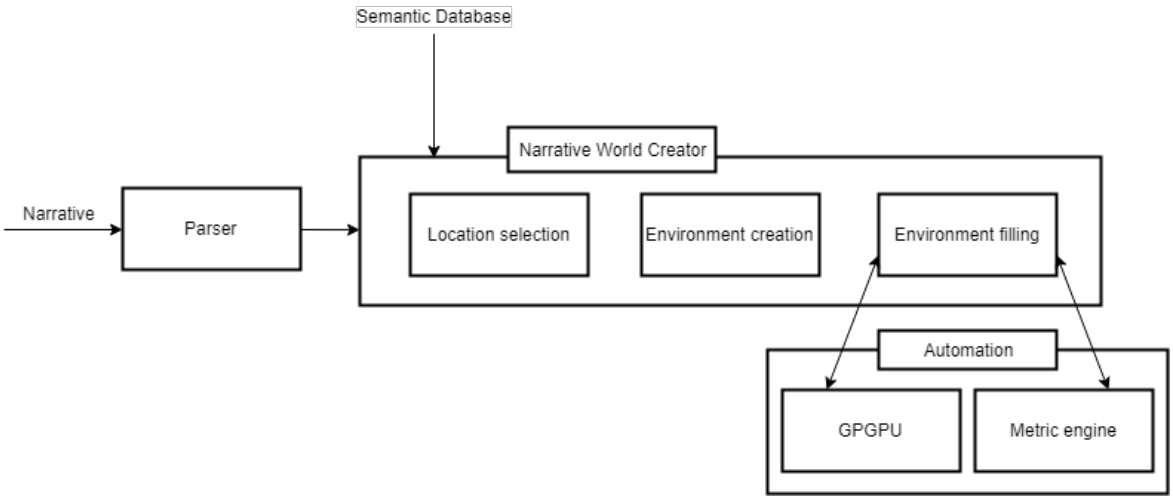

Figure 5-1: The different components of the implemented system.

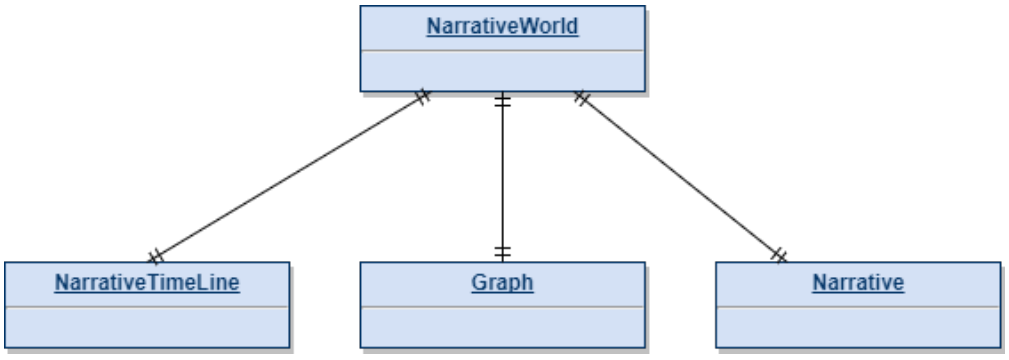

Figure 5-2: The narrative world model represents a narrative world. It includes the *narrative*, the *graph*, consisting of locations and connections, and the *narrative time line* defining all time points and events.

An overview of the data model used by the entire system, shown in Figure 5-2. Parsing the narrative results in filling the 'NarrativeWorld' model with information. The 'Narrative' model holds all information directly parsed from the input narrative, the 'Graph' model stores the locations and their connections and the 'NarrativeTimeLine' model stores, for each location, the events, time points and the belief state.

5-1-1 Semantic database specifics

Entika provides an implementation of Tutenel [2012]'s semantic model, and it was used to create the semantic database used in the usability study. Unfortunately, there are only small and limited examples available of semantic databases. These were explored in the Chapter 2, with some suggested adaptations as conclusions. The definition of master objects and relationships was adjusted, but spaces remain the same. Master objects and relationships are each discussed in a separate section. How the semantic database connects with the narrative world creator component is discussed in section 5-2.

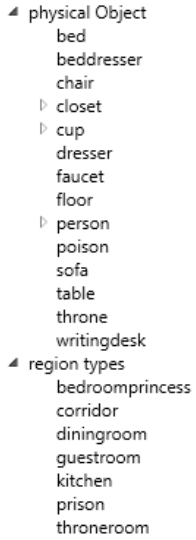

Figure 5-3: Hierarchical parent-child structure of the master objects.

Master objects

A new hierarchy was implemented to better split the different types of master objects defined in the semantic database. See Figure 5-3, on the top level there are two master objects which are the parents of all other master objects: 'physical Object' and 'region types'. 'region types' are the different location types which are connected to the narrative. 'physical Object' is the parent to all master objects in the semantic database and defines all base attributes and spaces for the master objects. Further inheritance is available to be used for inheriting specific attributes, spaces or relationships.

Relationships

We choose to implement a new set of basic relationships with clear and exact meaning because the meaning of the relationships used in the 'simplegame' example were not clear. The goal was to provide a set of basic relationships that can be expanded upon and to form a set of relationships that can be combined into more advanced relationships.

The usage of relationships as defined by Semantic game worlds was also slightly adjusted. As a simplification, relationships only exist between master objects, instead of between master objects and spaces, and between master objects. Secondly, the relationships only define a single source and a single target, instead of allowing multiple targets per relationship. These simplifications allow less expressive relationships but were good enough to provide a basic set to test the difference between assisted approach and manual approach.

We define relationships as being either evaluations or restrictions of either *position* (X,Y,Z) or *rotation* (X,Y,Z). Restrictions restrict one or more values of an attribute and evaluations provide scores for a range of possible values. For clarity, restrictions are called 'boolean' relationships and evaluations are called 'valued' relationships. The following relationships are supported by the implementation:

- **On**, describes whether object A is on top of object B. This defaults to the defined top space.
- **Distance**, describes whether object A is at a certain distance from object B.

There exist some restrictions on the amount of relationships each object may have. Each object must have a single 'on' relationship that defines on which other object it is placed. Additionally an object may have multiple other relationships based on the ones known by it's master object. This does not mean that a master object only defines a single 'on' relationship, but that the instantiated object

Although very limited, these relationships offer a basic set to express relationships between master objects. The types can be used to expand this set of relationships to a true basic set that can be combined into more advanced relationships.

5-1-2 Parser specifics

The parser is responsible for reading out the input narrative and deriving all necessary information from it. The parser supports a PDDL-format based on the pddl format discussed in Chapter 2. The parser goes through the following steps:

1. Parses the narrative as three separate files: the domain, problem and plan file. This results in a list of locations, the events associated with them and the predicates that hold for each of these locations on any given time point. This is the 'Narrative' model, Figure 5-4.
2. Takes the locations and derives their connections using the 'move' events. This is stored in the 'Graph' model, Figure 5-5.
3. Derives the belief state of each location and time point and stores these in the 'NarrativeTimeLine' model, seen in Figure 5-6.

During creation of each location, a floor object is added which will eventually store the surface of the location when it is created during environment creation. The narrative timeline is, through the time points, directly connected to the locations that are part of the graph. It is also directly connected to the narrative, because the time points store the predicates that are believed at that time point. In the end, the narrative world model is ready to be used in the narrative world creation process.

5-2 Narrative world creator manual approach

5-2-1 General overview

The narrative world creator component, facilitates the *actions* available to the designer and connects the other components to the *actions*, see Figure 5-1. The *actions* are performed through a Graphical User Interface (GUI) that covers each of the steps of the narrative

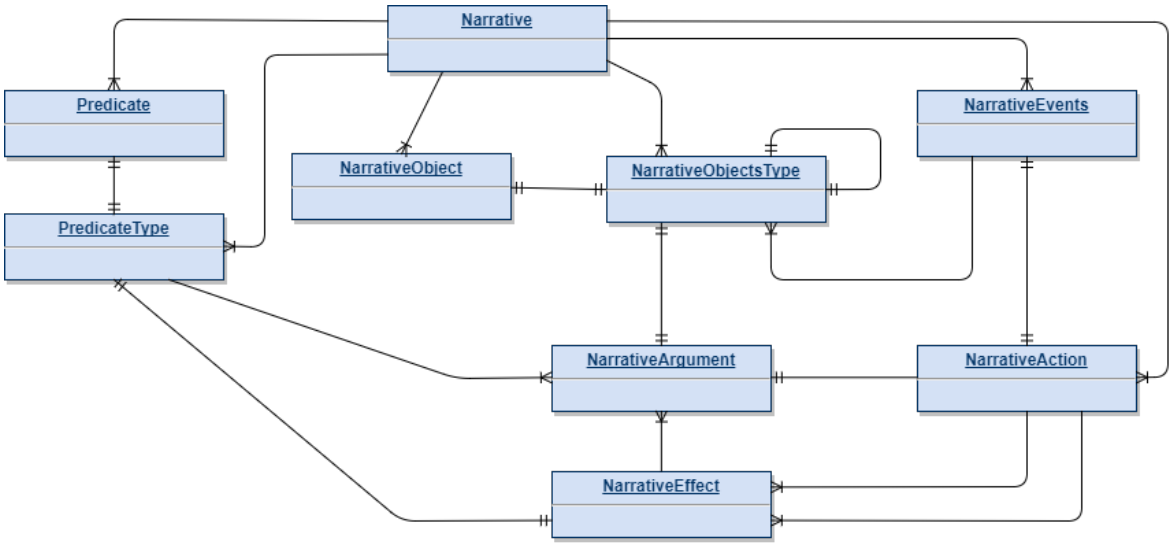

Figure 5-4: The different data types found in the pddl formatted narrative.

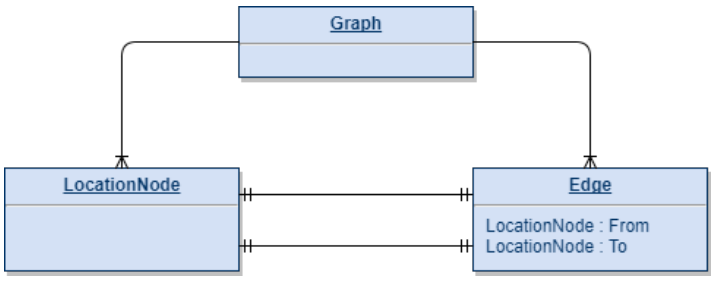

Figure 5-5: A simple graph implementation that uses nodes and directed edges.

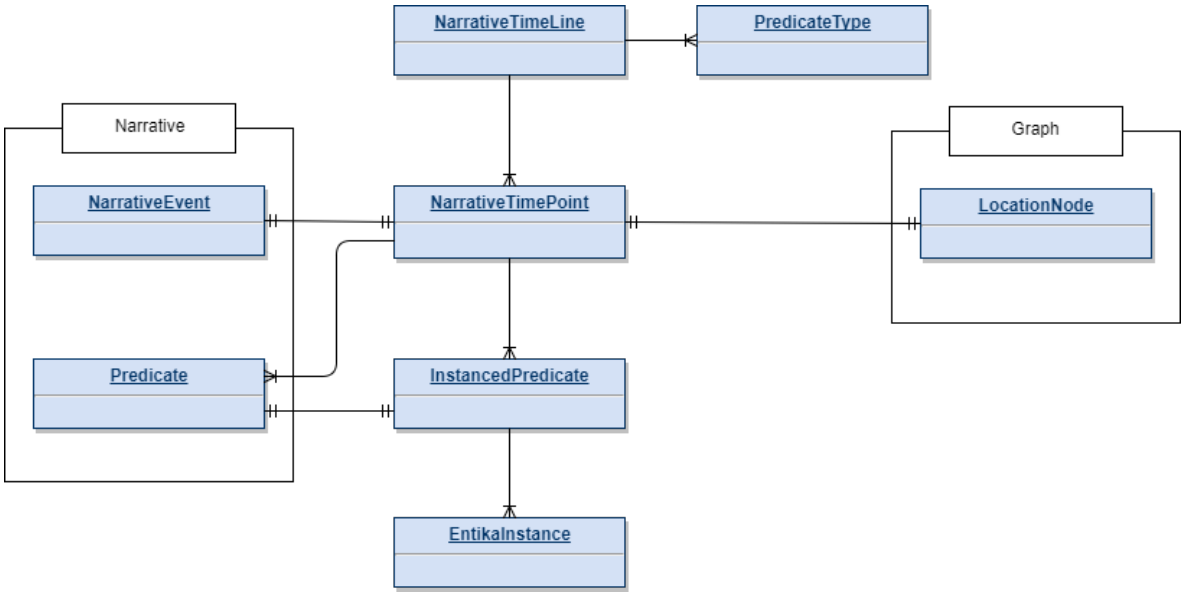

Figure 5-6: A narrative timeline that consists of one or more time points with each time point connected to the appropriate event, and location. Each time point has a list of predicates that define the belief state at that point in time. 'EntikaInstance' is the data representation of the instantiated objects.

world creation process. The GUI is implemented using Windows Presentation Foundation (WPF). It includes basic UI elements, but is able to incorporate a 3D renderer powered by the Monogame framework. The 3D renderer allows visualization and direct interaction with the environments as it is being created and filled. The Model-View-ViewModel (MVVM) pattern is embedded into WPF, and we implemented a Model-View-Controller (MVC) on top of that to make sure that data is consistent through the application, whether it is shown through the 3D renderer or through other UI elements.

The process discussed in Chapter 3 was fully implemented, except for the time point selection, which was left out of the final version because of the focus of the usability study on the difference between the manual approach and assisted approach. The final implemented process is shown in Figure 5-7.

The semantic database is connected to the narrative world creator component by directly accessing the database files created through Entika. This is done by using functions of imported semantic projects. The old and stale code base of Entika made it difficult to determine what functions to call. Connecting the narrative parser was simpler due to it being our own written functions. The parser is written using a static class and can be easily transferred to a separate project.

5-2-2 Location selection

The first step of the process is the selection of a location. This requires the designer to be able to view the different locations that are part of the narrative. The graph is layed out through an energy-based graph visualization algorithm, the end result is shown in Figure 5-8.

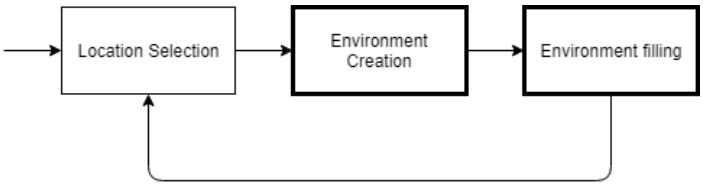

Figure 5-7: The implemented narrative world creation process, full process is shown in Figure 3-1. The selection of time points was left out.

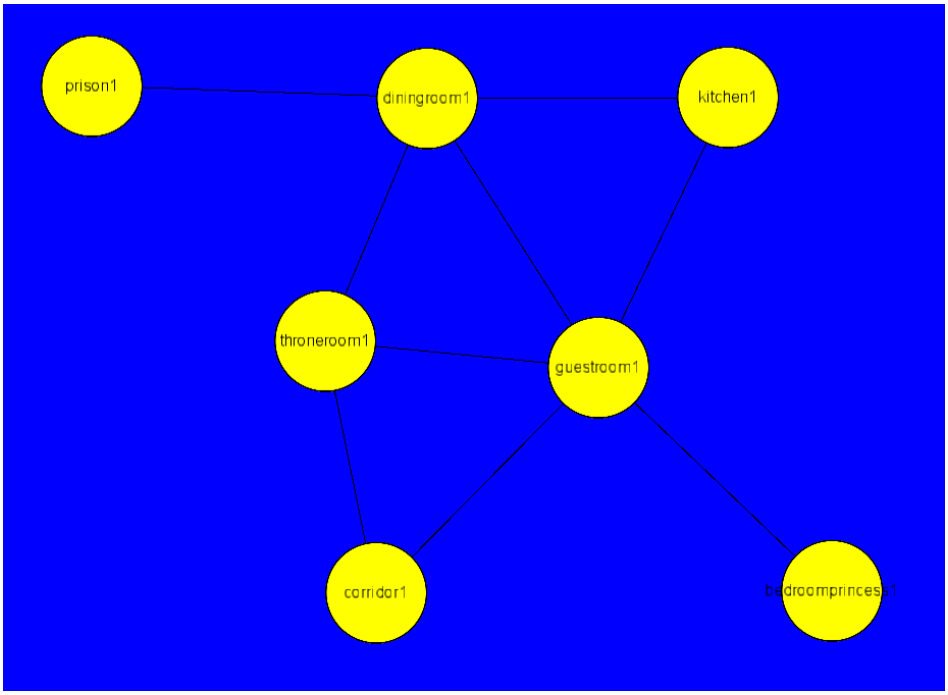

Figure 5-8: Visualization of the graph, allowing the designer to view the different locations and their connections.

The algorithm is an implementation of the algorithm implemented by Satsuma [2016]. The visualization allows a more intuitive way of looking at the locations and their connections than a list of locations which isn't able to show any connective context. The visualization was done by rendering 2D circles for the nodes of the graph and lines for the edges of the graph through Monogame.

5-2-3 Environment creation

The creation of the environment is done by clicking and dragging to create a rectangular shape. Figure 5-9 shows an example of a created environment. The dots in the figure are placed at every unit, which translates to 1 meter. This offers the designer a scale to base the environment on. As an additional aid, a 3D model of a human is drawn. This is also the first Monogame view that uses a 3D camera, and it is later used to show the filled environment. The top-down camera was used to simplify the controls available to the designer while still allowing an easy way to inspect the current state of the environment. The camera allows

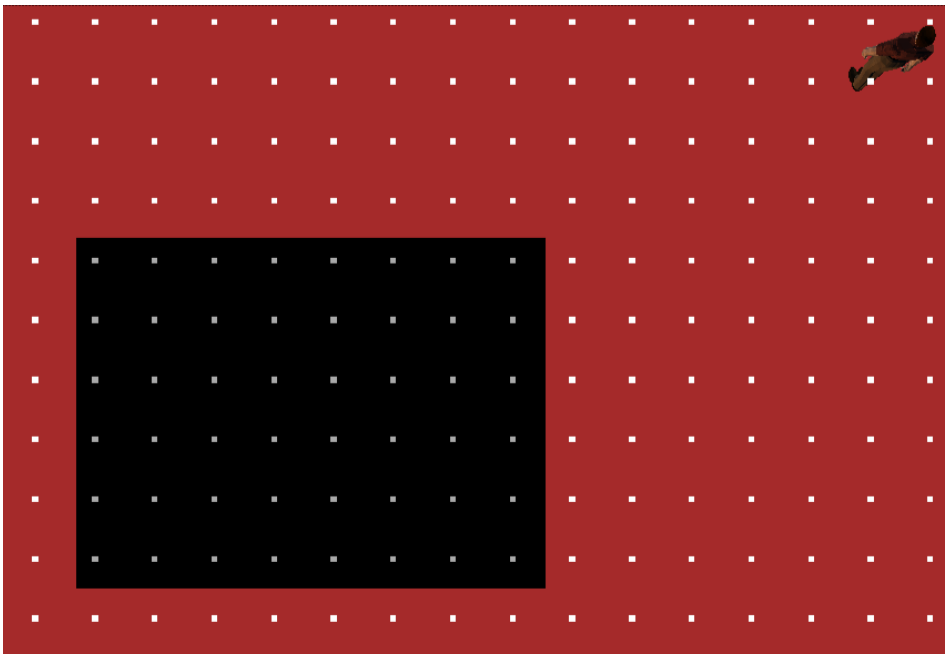

Figure 5-9: Shows the result of the implemented environment creation.

panning and zooming to navigate the environment.

The creation of a rectangular instead of a polygon is a simplification of the approach discussed in Chapter 3. A polygon would distract too much from the focus on environment filling. Environment creation was included in the final implementation as to give the designer context on what he will be filling and to reinforce the idea that he is creating a narrative world of his own.

5-2-4 Environment filling

The actions discussed in chapter 3 have been implemented. The 'add', 'change' and 'remove' actions each follow the defined steps. For the 'add' action, master object selection and placement are implemented as separate tabs, but relationship selection and instantiation are combined into a single tab. Figures showing the manual approach UI are in appendix A. Figure A-1 shows an overview of the UI, Figure A-2 shows the main menu, which is static for each approach and Figures A-3, A-4 and A-5 show the UI for the different step types.

The 'change' action allows the designer to change the placement of one or more objects, but not their relationships or the instantiation of those relationships. Removal and re-adding is the only way to adjust these. Selection of objects for changing and removal is done through the Monogame view, by directly selecting the objects, see Figure 5-10.

5-3 Environment filling assisted approach

The implementations of actions for the manual approach are adjusted to incorporate the automation provided by the automation component. For the 'add' action, the manual steps

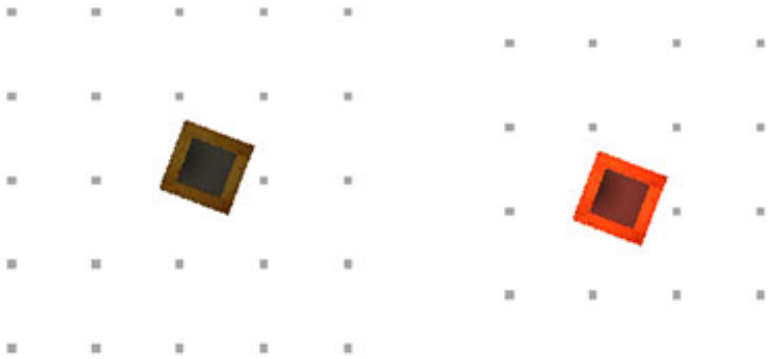

Figure 5-10: An object before and after selection.

are replaced by assisted versions. Figures of the resulting UI are in appendix A. Figure A-6 shows an overview of the UI, with Figure A-7, Figure A-8 and Figure A-9 show the UI for the different step types.

The different steps of the 'add' action were automated. The following sections go into more detail, with the section 5-3-1 discussing master object selection, relationship selection and relationship instantiation, and section 5-3-2 discussing the placement step. The 'change' and 'remove' actions were not automated, but the automated placement step of the 'add' action was re-used for the 'change' action. This allows the generation of new placements for all current objects which is equal to the placement offered during the assisted 'add' placement. To be able to prevent objects from being moved or rotated during the assisted placement the freezing of objects was implemented and is always available in a tab in the UI. A figure showing this is in appendix A, Figure A-10.

5-3-1 Discrete

In the automated version of 'add', for master object selection quantized options are provided and three goals, selectable through radio buttons, that affect the weights associated with the criteria. Relationship selection and relationship instantiation remain combined and the automation lists subsets of relationships that are available to the selected master object. From these discrete steps, only the master object selection implements a set of criteria, the relationship selection and instantiation options are randomly generated. This was done to save time and should not affect the end result greatly as the relationships offered are very basic and the random generation is able to generate most combinations.

As discussed in the approach, the master object selection step makes use of criteria to quantize the value a master object has to add it to the environment. The values calculated for each criteria are, after being normalized, combined with weights to result in a total score on which the list is ordered and is displayed to the designer. The combining of the weighted criteria is done through summation of all weighted criteria. Three goals are defined upfront with

Table 5-1: Shows the weight for each criteria for each goal.

	Required	Decorative	Neutral
Incoming Edges	1	1	1
Outgoing Edges	1	1	1
Incoming Decorative Edges	0.1	3	1
Outgoing Decorative Edges	0.1	3	1
Incoming Required Edges	2	0.1	1
Outgoing Required Edges	2	0.1	1
Incoming Available Edges	1	1	1
Outgoing Available Edges	1	1	1
Required	3	0.1	1
Required Dependency	3	0.1	1
Decoration Weight	0.1	3	1
Area	3	3	3

each goal defining the set of weights statically. The value of each weight was determined empirically and shown in Table 5-1. The goals are: *neutral*, *required* and *decorative*. *Required* means that the value indicates master objects that are required by the narrative or master objects on which required master objects depend. An master objects is dependent on another when it is the target of an 'on' relationship. *Decorative* master objects are all master objects that are not required or are a required dependency. *Neutral* focuses on neither decorative or required master objects. The three goals allow the designer to focus their search instead of having a single value that incorporates all of these goals.

Criteria are based on the information provided by the current environment, the narrative, and the semantic database. The discussed dependency graph is implemented and is annotated with information from each source. Each criterion extracts data from the dependency graph. Following is a list of criteria that are implemented:

- Incoming edges of an master object, the number of relationships a master object is a target of.
- Outgoing edges of an master object, the number of relationships a master object is a source of.
- Incoming/outgoing decorative edges, the number of relationships that are between decorative master objects and the master object is the target of.
- Outgoing decorative edges, the number of relationships of a master object that are not required by the narrative.
- Incoming/outgoing required edges, the number of relationships of a master object that are required by the narrative.
- Incoming/outgoing available edges, the number of relationships of a master object that are available in the current environment.
- Required, whether the master object is required by the narrative.

- Required dependency, whether a required master object depends on the current master object.
- Decoration weight, decorative value of the master object based on the current location type.
- Area, the surface area that the model of the master object takes up.

The set of criteria by Tutenel [2012] was expanded to include narrative information, through the required and decorative criteria, and from the current environment, through the incoming/outgoing available edges criterion. The criteria use both integers and booleans as result, which are combined by using the 0 and 1 representation of a boolean and combining this with its weight.

The addition of weights allowed the manipulation of the values to make some values more important than others. The required goal uses increased weights for the required edges, required and required dependency values and uses decreased weights for the decorative edges and decoration weight criteria. This makes the end value be higher for master objects that score higher for these criteria, while not completely discarding the meaning of the other criteria. The influence was flipped when concerning the decorative goal and equalized for the neutral goal.

5-3-2 Continuous

Placement is done through the generation of new placements of all objects, including the newly added object, by the adjusted Interactive Furniture Layout (IFL) algorithm [Merrell et al., 2011].

A version of IFL was provided by Merrell et al. [2011] and was used as a basis. It was re-implemented in the newest version of C++ and CUDA. A wrapper was implemented that made it callable from the *Narrative world creator* component that was written in C#. Weights and other configurations were made inputs of the wrapper instead of the static weights used by IFL. Users can not influence weights, the weights were set empirically to balance the effect each criterion had on the final solution of the algorithm. Lastly, the displacement of an object was restricted to the area defined by the surface it is placed on.

As discussed in Chapter 4, not all criteria provided by IFL were usable by us. The following criteria are used and follow the implementation by IFL:

- Pairwise relationships
- Visual balance
- Focal point
- Symmetry

Other criteria were part of IFL but used a different implementation. During implementation of these criteria, a different approach was taken as to make the implementation fit better with the spaces defined by semantics. Semantics defines spaces as rectangles and the implemented

criteria calculate overlap between these spaces to determine the costs. This resulted in the following criteria:

- Offlimits costs
- Clearance costs
- Surface area costs

In the end, only a subset of criteria was used for the usability study due to performance issues caused by scalability issues. The usage of multiple surfaces and limited time to implement criteria resulted in the performance issues. This was solved in two ways: the surface area cost criterion was deemed unnecessary, because repositioning an object was adjusted to not allow an object to be placed outside of the target surface area; and the clearance costs criterion was not included in the usability study as it did not scale well enough in its current implementation. The resulting criteria, although not very expansive or configurable, provided some highly variable results that the designer could use as starting ideas for placing objects.

Chapter 6

Experimental evaluation

This chapter discusses the experimental evaluation, and the results of the assisted approach, compared to the manual approach. First, the design is discussed, followed by the implementation of the usability study and finally the results. The design discusses what the goal of the usability design is and how this is approached using different metrics. The implementation section discusses the input of the system and the instructions document. In the results section, the results of the different measurements are analyzed and visualized.

6-1 Design

This usability study is designed around the following research question, which was stated in the introduction:

Research question 1.4. To which extent is the proposed *assisted approach* capable of assisting the designer in making decisions compared to the *manual approach*?

This makes the goal of the usability study to compare the two approaches and see whether the assistance provided in the assisted approach is of help. The usability study should determine how much better one approach is to another and it should measure the cause of this difference. Due to the usability study having to also measure the cause of the difference between the two approaches, the best way to measure this is by having each participant use both approaches and comparing them. This allows the participant to directly compare the two modes and more information to be extracted concerning the causes of the effects that each approach has on environment filling.

Both qualitative and quantitative metrics are used to measure the different measurement types. To measure which approach is better and how much it is better, a questionnaire called Creativity Support Index (CSI) is used, discussed in Carroll and Latulipe [2009], together with quantitative measurements on semantic usage and time spent on specific parts of the system. The CSI questionnaire is designed to evaluate creativity support tools. It does this

by incorporating creativity theory into the NASA Task Load Index questionnaire. The CSI questionnaire is a great fit for measuring the performance of the two approaches as it is designed to measure the performance of systems for creative processes. The CSI questionnaire is filled in after the participant has filled environments with a particular approach. In theory, the results of the CSI questionnaire should measure the effectiveness of the tested approach, which should be also reflected in the quantitative measurements on usage and time spent. The CSI questionnaire works with six main factors: 'Collaboration', 'Enjoyment', 'Exploration', 'Expressiveness', 'Immersion' and 'ResultsWorthEffort'. First, the questionnaire determines what a participant thinks of each factor. Secondly the relative importance is measured and results in average factor counts for each factor. The highest possible factor count is 5, indicating that participants chose it as more important than every other factor. These are combined in to a result for each factor and an overall result of the questionnaire. Interpretation of the CSI questionnaire results give insight into the causes for difference in performance through the factors, but only offer a basic insight. A final questionnaire, at the end of the usability study, is used to look at the exact causes of the difference in performance.

The following hypotheses incorporate the defined metrics into measurable hypotheses that should provide an answer to research question 1.4. Hypothesis 6.1 directly measures the goal of this usability study, while hypotheses 6.2, 6.3 and 6.4 provide context to the CSI result through quantitative results.

Hypothesis 6.1. *The assisted approach will result in a higher overall CSI questionnaire result than the manual approach.*

Hypothesis 6.2 compares the semantic usage with the CSI results. We think that the lowest scoring approach is also the least capable of exploring the possibilities available and thus result in the smallest set of master objects and relationships used. Hypothesis 6.3 compares the time spent on individual actions with the CSI results as both measure the ease of the application. Hypothesis 6.4 is based on the notion that a good supporting system will result in the user making fewer errors and thus requires less usage of the 'change' and 'removal' actions.

Hypothesis 6.2. *The approach which has the lowest overall CSI questionnaire result also has the smallest set of master objects and relationships used.*

Hypothesis 6.3. *The approach which has the lowest overall CSI questionnaire result also has the longest time spent on individual actions.*

Hypothesis 6.4. *The approach which has the lowest overall CSI questionnaire result also has the most 'change' and 'remove' actions.*

The usability study is done by students and research employees at the TU Delft. In an optimal case, the usability study should be conducted by professional level designers, but this is outside the scope of this thesis. The students and research employees at the TU Delft are adequate for testing the system because their experience with computers lowers the time required to get used to the system. Experience with narratives and environment filling are added benefits and can provide more insight into the effect of the application on more knowledgeable participants. There are enough students and research employees to conduct

the usability study among but finding proved to be difficult. Each participant participates completely voluntary, which makes it difficult to find participants as they do not benefit from participating in any way. To decrease the chance of people declining, the usability study is limited to 30 minutes. We aimed to have 10 to 20 participants which would provide sufficient information to look at comparing the two modes and the causes of potential differences.

6-2 Implementation

The usability study was conducted in the master thesis student room in the high-rise building of EEMCS (HB12.320). This room offers a quiet working space in which the participants can perform the necessary tasks. The tasks were performed on a desktop PC on a 24-inch monitor with mouse and keyboard. The desktop PC has been recently bought and uses a Nvidia 1070 Graphics card which the system uses to run the IFL algorithm on. Participants performed the tasks one at a time at a scheduled time.

The usability study makes use of several resources, namely: a narrative, a semantic database, an instruction document, a starting questionnaire, a final questionnaire, and the CSI questionnaire. The narrative and semantic database are inputs to the implemented system and the instruction document is a step-by-step procedure for the tasks. The start and final questionnaire are designed and created by us, whereas the CSI questionnaire is conducted through a java program available at http://www.erincherry.net/csi.html. The starting questionnaire collects data on the subject's experience with computers, narratives, and environment filling. It is in appendix D. The final questionnaire collects data on what the participant thought of the assisted approach. Its questions focused on measuring the causes for the difference in performance between the assisted and manual approaches. It is in appendix E. The instruction documents follows the following procedure. The participant will test both approaches of environment filling. The participant will be filling two environments using the manual approach and two using the assisted approach. Because the assisted approach only partially replaces the manual approach, the manual approach is done first as it provides a good way to explain the basics. After filling two environments using the manual approach the participant is asked to fill in a CSI questionnaire about the manual approach. This is followed by an explanation of the differences for the assisted approach. The participant fills another two environments, but this time using the assisted approach. When done, a CSI questionnaire is filled in about the assisted approach. The written manual is in appendix C.

In the next section the details of the inputs are discussed. In the last section, the adjustments to the implementation to include the quantitative measurements are discussed.

6-2-1 System input

The narrative used is a small story with four characters moving between locations and interacting with each other and the environment. The narrative is called castle and the pddl definition is in appendix B. The characters of the narrative are a king, a prince, a princess, the daughter of the king, and a maiden of the king. The prince and princess try to poison the king, but the maiden saves the king by switching the cup of the prince and the king. The prince dies, the princess is sentenced to prison and the king marries the maiden.

The semantic database holds 19 master objects, 29 relationships, and 7 location types. Among the master objects are the ones used by the narrative, but also additional master objects to decorate the rooms. The relationships connect the different master objects allowing placement through 'on' relationships and evaluation through distance relationships. The location types match the ones used by the narrative.

6-2-2 Quantitative measurements

The quantitative measurements measure three things: usage metrics of semantics, usage metrics of actions and time spent on specific parts of the system. For semantic usage the following metrics were implemented:

- In general
  - What master objects were used, and how much
  - What relationships were used, and how much
- For each location
  - What master objects were used, and how much
  - What relationships were used, and how much

The usage numbers and time spent both use the same type of actions, namely the following:

- Addition
- Both manual and assisted change
- Manual change
- Assisted change
- Removal

For usage numbers of actions, the following metrics were implemented.

- In general, amount of actions done per type

For time spent on specific parts of the system, the following metrics were implemented for each location:

- Total time spent
- Time spent per location
- Time spent on action per type and location

Table 6-1: Collected data

Number of participants	13
Starting questionnaires	13
CSI questionnaires on manual approach	13
CSI questionnaires on assisted approach	13
Final questionnaires	12
Complete quantitative metrics	11

6-3 Results

Thirteen participants participated in the usability study in a time period of 2 weeks. Each participant executed all tasks. The results of each metric are summarized in a separate section. For one person the final questionnaire was not submitted and their input was lost. For two persons something went wrong with the collection of the quantitative results due to system crashes, these were fixed after they occurred but resulted in lost data. This has been taken into consideration for the different metrics. Following is an overview of the available data:

6-3-1 Starting questionnaire

Figure 6-1, Figure 6-2 and Figure 6-3 display the results of the starting questionnaire. The starting questionnaire asked the participant to rate their experience with computers, narratives, and environment filling. Figure 6-1 shows the reported experience with computers. All except one participant reported their experience with computers to be high. This may indicate two things, first, that understanding and learning the interface is no problem. Being able to quickly understand and learn the interface should reduce the learning bias introduced by testing the two approaches back to back. Secondly, their feedback on the interface should be considered quite valuable as they are quite experienced with different types of interfaces. Figure 6-2 shows the reported experience with narratives. Although generally rated lower than the experience with computers, most participants assess their experience to be average or above. Understanding the usage of a narrative and understanding the used narrative should be no problem. Figure 6-3 shows the reported experience with environment filling. These results are more uniformly distributed indicating a large variance. Some participants rated their experience with narratives to be very low, which may influence their opinion on the use of narrative information as they might not understand the need.

6-3-2 CSI results

In general, the CSI reflects tool, task, and expertise level of the participant. To be able to compare the CSI scores between two treatments, only one factor may be varied between the two. In our case, the expertise and task level are the same and the tool varies. Expertise is barely different as can be seen from the starting questionnaire results. The task is equal as the participant is required to fill environments in both modes. The types of locations are not significantly different from each other. The tool is different as two significantly different approaches are provided to the participant.

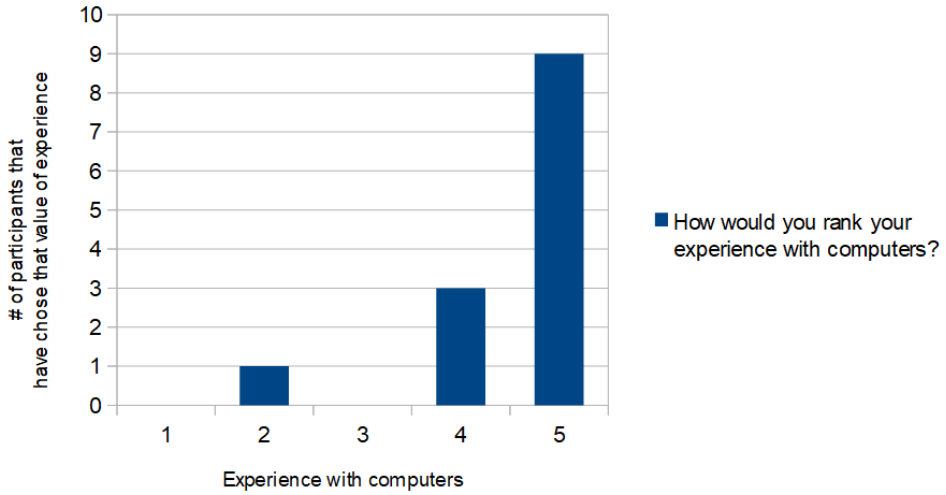

Figure 6-1: Barchart showing the experience with computers.

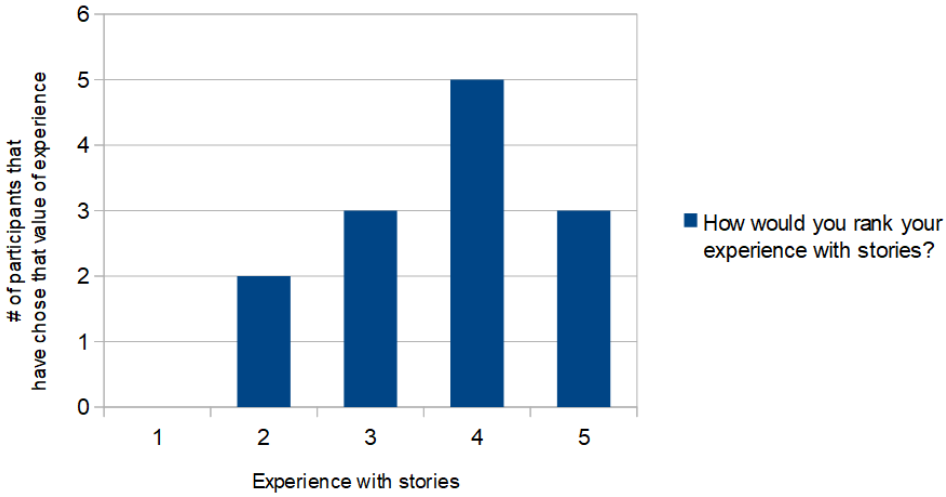

Figure 6-2: Barchart showing the experience with narratives.

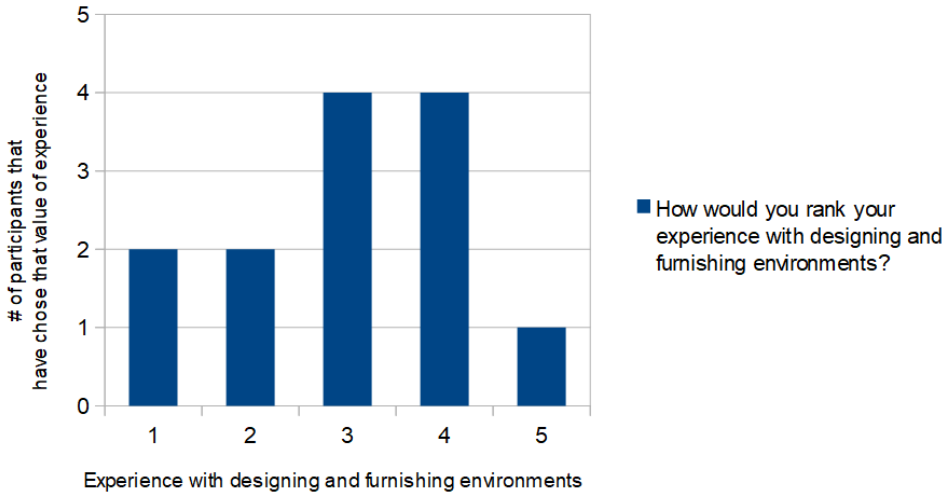

Figure 6-3: Barchart showing the experience with environment filling

Table 6-2: Shows the CSI results of the manual approach.

	Avg.Factor Counts (SD)	Avg. Factor Score (SD)	Avg. Weighted Factor Score (SD)
Collaboration	0.38 (0.87)	0 (0)	0 (0)
Enjoyment	2.38 (1.71)	10.30 (1.75)	21.90 (15.36)
Exploration	3.38 (1.61)	11.15 (3.11)	35.32 (20.40)
Expressiveness	3.46 (1.31)	10.15 (2.12)	33.17 (15.11)
Immersion	2.08 (1.61)	11.15 (4.14)	22.92 (23.13)
ResultsWorthEffort	3.31 (1.03)	13.08 (2.72)	41.14 (20.46)

Following the example provided by Carroll and Latulipe [2009], table 6-2 and table 6-3, respectively, show the results of the manual and assisted approach. Although any form of collaboration was not part of the tasks and they were told it was unnecessary, some participants selected the collaborative factor nonetheless.

Due to both modes being about the same task, in such a within-subjects study, the factor comparison is done for the environment filling task in general and not for a particular approach. The factor counts accumulated during the last CSI questionnaire, after filling environments using both approaches, are used as the factor counts for both manual and assisted CSI results. Avg. Factor Score shows the score that the participant gave each factor. It is the sum of the two measurements, which each use a scale of 0 to 10. This allows a minimum score of 0 and a maximum score of 20, with higher numbers indicating better support by the tool. Weighted Factor Scores are calculated by multiplying a participant's factor score by the factor count. This makes the weighted factor scores more sensitive to the factors that are the most important Carroll and Latulipe [2009].

The overall CSI result for the assisted approach, 42.46, is lower than the manual result, 53.51. This indicates that the participants thought the manual approach supported the creative

Table 6-3: Shows the CSI results of the assisted approach.

	Avg. Factor Counts (SD)	Avg. Factor Score (SD)	Avg. Weighted Factor Score (SD)
Collaboration	0.38 (0.87)	0 (0)	0 (0)
Enjoyment	2.38 (1.71)	8.54 (4.22)	17.50 (15.84)
Exploration	3.38 (1.61)	8.38 (3.88)	30.64 (23.34)
Expressiveness	3.46 (1.31)	9.15 (3.93)	30.21 (18.32)
Immersion	2.08 (1.61)	8.00 (4.81)	17.76 (13.12)
ResultsWorthEffort	3.31 (1.03)	9.08 (4.50)	28.78 (20.62)

process better than the assisted approach. Secondly, the assisted result has a high amount of variance, with a standard deviation of 18.28, when compared to the standard deviation of the manual result, 7.66. This indicates that the participants were more divided on their opinion regarding the assisted approach than the manual approach.

The factor counts show the factors 'Expressiveness', 'Exploration', and 'ResultsWorthEffort' being scored the highest. This makes sense given the task of filling an environment, a complex task that is heavy on the creative influence a designer may have. The average factor scores are lower across all factors. The highest, and most interesting, losses are 'ResultsWorthEffort' (-4), 'Immersion' (-3.15) and 'Exploration' (-2.77). These factors were the highest scoring for the manual approach. The decrease for 'ResultsWorthEffort' shows that the participants felt like either or both the approaches did not offer satisfactory results given the effort. The decrease in 'Immersion' would indicate that the interface provided was distracting the participants from performing their tasks. Perhaps this is caused by the importance value and total costs providing distracting information. A decrease in 'Exploration' is strange as the assisted approach provides suggestions for the placements and attempts to help explore the options. Perhaps the quality was not high enough to provide, in the participant's eyes, helpful suggestions which caused the participants to score this factor lower. Another interesting change is 'Enjoyment', which did not decrease by a lot but saw a large increase in variance. This combined with the increase in variance across the board and the lower factor scores support the overall CSI score that the assisted approach performed worse than the manual approach.

6-3-3 Quantitative results

The quantitative results consist of system usage, semantic usage and time spent. Unfortunately, the system usage was only stored for general action types per location and not for each action individually. Instead of directly looking at the actions done, ratios were used to look at the semantic usage and the time spent.

Semantic usage

The goal was to provide insight into how the semantic database was used. This is done by comparing the ratios between total master objects and relationships and the used master objects and relationships. The ratios are calculated for each approach and for each location. In total there are 20 master objects and 29 relationships in the semantic database. Table 6-4 and table 6-5 show the calculated ratios. When looking at the two approaches both the

Table 6-4: Master object ratios for each approach and ratios for each location

	Ratio for each approach		Ratio for each location
Manual	0.31	prison1	0.22
		bedroomprincess1	0.40
Assisted	0.28	corridor1	0.21
		throneroom1	0.35

Table 6-5: Relationship ratios for each approach and ratios for each location

	Ratio for each approach		Ratio for each location
Manual	0.26	prison1	0.18
		bedroomprincess1	0.35
Assisted	0.20	corridor1	0.15
		throneroom1	0.25

ratios for master objects as well as relationships do not lie far apart. There is a difference between locations, which can be observed from both relationships as well as master objects. The common aspect of these locations is not the associated approach, but the lower ratio locations being, typically, less populated with objects.

Time

The time spent on a particular action was recorded to compare the time spent for actions. This was done by tracking the total time used for a particular action given a certain location. To compare three ratios were used, *addition/total*, *change/total* and *remove/total*. Figure 6-4 shows these ratios for the four different locations.

The only location to include a significant amount of time spent on removal is the prison1 location. This is explained by prison1 being the first location to be filled. The 'remove' action is discussed and the participant is instructed to perform this action by the instructions document. This is the only time this is instructed, which means that for the other locations the participants saw no need to use the 'remove' action. This may be caused by the usability study not being long enough for the participant to make errors that require removal.

Figure 6-5 shows each ratio visualized as a box plot. This allows the inspection of the spread of each ratio. 'Throneroom1' and 'bedroomprincess1' have more and higher variance and slightly more time spent on the 'add' acction compared to the 'change' action. This indicates that the time spent on adding versus changing is wildly different between participants, while the variance is much smaller for the 'prison1' and 'corridor1' locations. Each approach has both a high variance and low variance location, which indicates that this problem has other causes than the difference between approaches.

The average total time spent on each location is:

- Prison1, 277 seconds
- Bedroomprincess1, 210 seconds

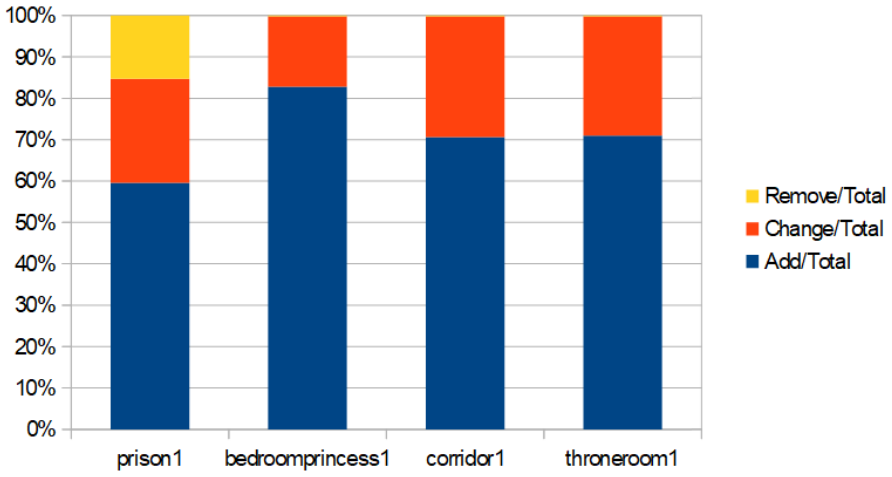

Figure 6-4: Bar chart showing the add/total, change/total and remove/total time percentages for each location.

- Corridor1, 354 seconds
- Throneroom1, 252 seconds

The 'prison1' and 'corridor1' locations having the most spent time on is most likely caused by them being the first environment to fill for the respective approaches and thus include an explanation from the instruction document which increased the time spent. If the actions were tracked individually one would be able to say more about the usage of the system as one could filter out the first action that was done as it would incorporate the explanation.

6-3-4 Final questionnaire

The final questionnaire focuses on the reasons why participants preferred one approach above another. This is done by enquiring about the opinion on different parts of the assisted approach. Subjects were extracted from the answers because not all comments directly answer the question but offer interesting information. This does require some interpretation of what was meant. Each subject is briefly discussed in its own section.

Importance value

The importance value was used during master object selection and was used to indicate the importance of a master object given a certain goal. Overall, the participants liked the way it allowed the participants to focus on a particular goal. They did not like how the goals were not just filters, but boosted some aspects while keeping others in consideration, as this sometimes resulted in non required master objects to be high in the list although the goal was set to required, as well as required master objects being further down the list. From their perspective these results were inaccurate as the 'required' goal was active. Another

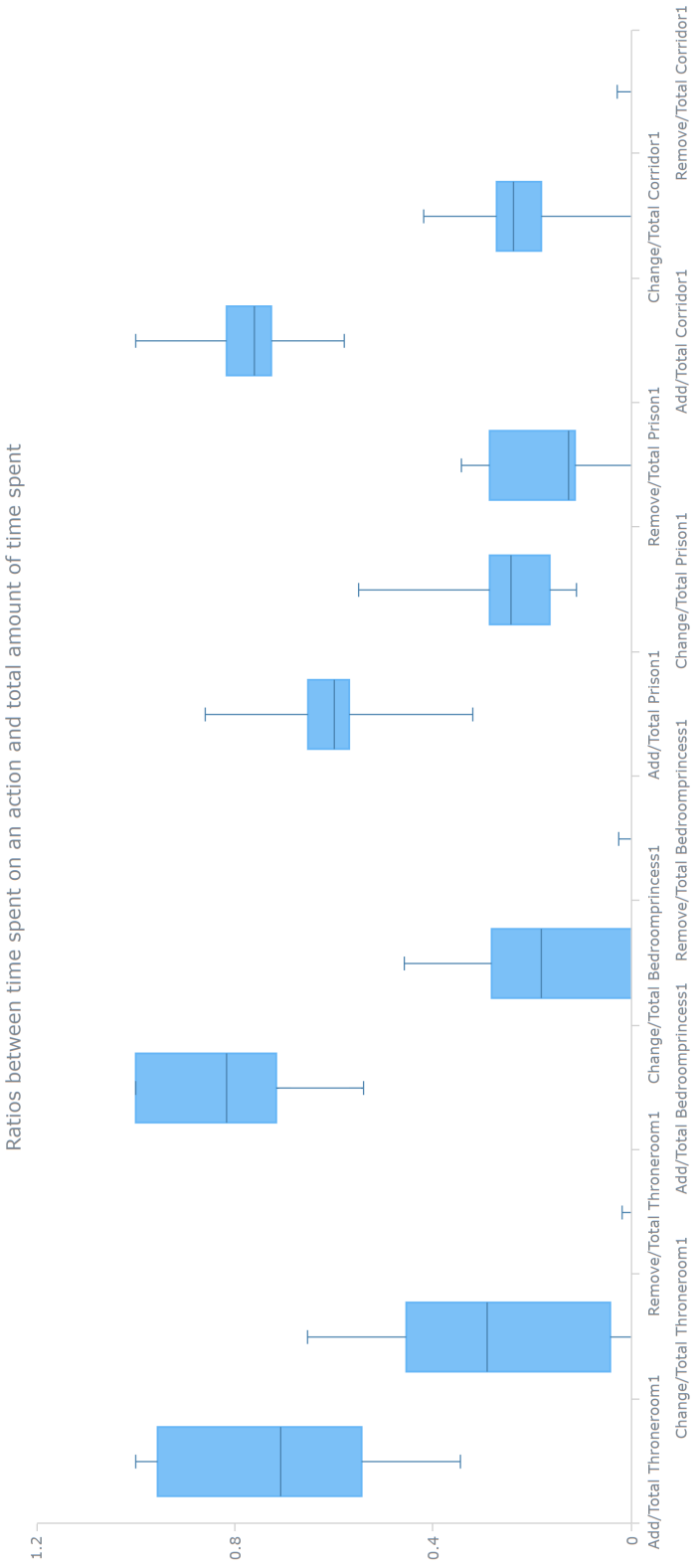

Figure 6-5: A box plot for each of the ratio for each of the locations.

point of criticism was the importance value giving high values to, from the participants view, wrong master objects. This was caused by master objects being rated as important for weird location types. This point of criticism indicates a more fundamental problem, namely the fact that each participant's perspective on what is important is different.

Using and combining relationships

During relationship selection and instantiation, the assisted approach provided suggestions for the participant to choose from. The suggestions were regarded as helpful, clear and understandable. A point of criticism was that it was impossible to see the possible relationships available and which master objects can work with those relationships. The participants felt the available relationships types to be limiting, as they wanted to have more precision, for example by allowing a relationship that places something directly next to another one.

Placement suggestion

While one participant liked the suggestion of placements for the new combination, others did not like it at all. Most criticisms focused on the quality of the suggestions, as participants felt like they still wanted to adjust the position and rotation of most, if not all objects. This caused most participants to manually change the placement afterwards.

Most participants liked comparing the different options through the four screens but did not like the use of a costs function to indicate the costs. They felt like the costs were out of context and very vague as it was unknown what caused some suggestions to be scored worse than others. This shows that the total cost function, when presented, should be sensible, but this is, equal to the importance value, different for every participant. Some negative points on the comparison of options were that saving an option by showing that option through the top left screen and not being able to tweak the placement of objects directly when comparing suggestions was cumbersome.

6-3-5 Discussion

During this discussion, we will be answering the hypotheses one by one using the collected results. First, we start with the main hypothesis, followed by the contextual hypotheses. Lastly, the results of the final questionnaire are discussed and connected to the results of the other metrics.

Hypothesis 6.1. *The assisted approach will result in a higher overall CSI questionnaire result than the manual approach.*

Before making conclusions on the results it is important to note that the CSI questionnaire was used correctly as the task and expertise level were the same for each participant while varying the tool. This is supported by the starting questionnaire as it shows the expertise level is about equal for the participants. The answer to the hypothesis is: no, the assisted approach did not result in a higher overall CSI result than the manual approach. It also resulted in a higher variance, which means that some participants liked it equally or a bit more and some

participants liked it much less than the manual approach. The factors gave some insight into which factors were rated worse, but each of them increased in their variance making the results difficult to conclude upon. The heaviest affected factors are: 'ResultsWorthEffort' (-4), 'Immersion' (-3.15) and 'Exploration' (-2.77).

Hypothesis 6.2. *The approach which has the lowest overall CSI questionnaire result also has the smallest set of master objects and relationships used.*

The results of the semantic usage are not able to answer this hypothesis, the ratios for each approach lie too close together. The ratios for each location type show that typically less populated location types also have a lower ratio of master object and relationship usage. This seems to be the cause of the difference in ratios and overshadows any influence the difference in approaches might cause. To potentially better show the difference in approach through this measurement, the usability study would have to include significantly more types of locations and would require each participant to fill a lot more of them as to spread out the effect that dictates the current results. This would also require a significant increase in distinct master objects and relationships as to make sure there is enough variety for the participant to choose from. The current scope of the thesis did not allow for this.

Hypothesis 6.3. *The approach which has the lowest overall CSI questionnaire result also has the longest time spent on individual actions.*

Unfortunately, as the quantitative results did not include data on individual actions or amount of actions, this causes the hypothesis to not be answerable directly. The presented ratios are also not able to answer the hypothesis as it is affected by other variables than just the approaches used to fill them. The same counts for the total time spent on each location as the most time is spent on the first location that is filled for each approach, which is probably caused by participants pausing during the action to read the instructions provided. Filling more environments and tracking each action individually would allow the measurement of the difference between the two approaches.

Hypothesis 6.4. *The approach which has the lowest overall CSI questionnaire result also has the most 'change' and 'remove' actions.*

This hypothesis could not be answered, but not just because the number of actions was not tracked. The only time the 'remove' action was used, was during the location for which it was instructed to use it. No other time did a participant feel the need to remove an object from the environment. This is a positive outcome as the participants were happy with the objects they added, but this does not provide context for the CSI results.

In general, it is difficult to answer the verification hypotheses due to the unclear results. The solution to all three of them is to have longer and more detailed usability studies, which were out of scope and unfeasible given the set of possible participants. The last to discuss results, that do provide interesting points, are the final questionnaire results.

The final questionnaire asked opinions from the participants on the assisted approach, asking them what they liked about parts of it. Overall there were three major points of concern to the participants: ease of use, quality of the automation results and the results of the automation not matching their expectations. Each of these points is connected to one of the

major decreasing factors of the CSI result, with the connections between the factors and the final questionnaire supporting the results of the CSI questionnaire. Ease of use is connected to the decrease in 'ResultsWorthEffort' factor of the CSI results, the quality of the automation results is connected to the decrease in 'Exploration', participants found the quality of the suggestions not high enough, and the results not matching their expectations is connected with the 'Immersion' factor.

The ease of use can be improved by improving the assisted actions to require fewer clicks, the pointers from the final questionnaire gave good ideas. However, the quality of the automation and the results not matching expectations indicate a more substantial, fundamental, problem. Namely, how the automation can know what the designer expects. This is observed from the participants approving the ability to focus the importance value towards a certain goal, but the disapproval of the result of this focus as it strayed from their expectations. This problem lies at the core of providing mixed-initiative solutions for creative processes.

Chapter 7

Conclusion

At the start of the thesis several research questions were stated, for which the thesis would look for answers. Each research question is reiterated and the answer is concluded upon. First research question 1.1 and research question 1.2 are answered.

Research question 1.1. How does the usage of the PDDL-format of Ware and Young [2014] as narrative input affect the *industry approach*?

Research question 1.2. How does the usage of types, in the form of master objects, and explicit relationships affect the *industry approach*?

The structured narrative allows the *manual approach* to use the locations, time points, events and beliefs of the narrative during the narrative world creation process. This makes the steps needed to facilitate the narrative and its changes explicit, which allows for a structured process that has explicit steps for the selection of time points. The beliefs are re-used in the *assisted approach* as information with which criteria were made. Semantic definitions allow for a modular approach used in the *industry approach*, but most importantly, explicit relationships. The explicit relationships allow clear definitions of relationships between objects and can be connected to the narrative through beliefs. The decision about adding relationships is not an explicit step in the *industry approach*, but is made explicit by additional steps for most actions in the *manual approach*.

The addition of narrative and semantic definitions to the *industry approach* resulted in the *manual approach*, which was described as one of the main contributions of this thesis. Although the *manual approach* is capable of producing a narrative world that facilitates the narrative and its changes over time, it does this under several simplifications such as, a discrete timeline and sequential events. Next to that, most time was spend on environment filling, resulting in simplified versions of the other parts of the process.

Research question 1.3. How can the algorithm of Merrell et al. [2011] be adapted in order to used in the narrative world creation process?

The Interactive Furniture Layout algorithm by Merrell et al. [2011] was a decent fit with the automation problem offered by the placement of an object. It required adjustments to fit the broader possibilities of environments, which, ultimately, were not used as the environment creation was restricted to rectangular boxes. The criteria were also adjusted as to make them better fit with a wider range of object types, as not only furniture, but any kind of object could be defined in the semantic database.

Next to the IFL algorithm, was the discrete automation, which, combined, formed the *assisted approach*. The *evaluation of options* was the general approach to ordering options for the different problems. This was not implemented for the relationship selection and instantiation, the criteria put together for master object selection showed that the approach has its benefits such as being able to steer criteria towards a certain goal by adjustments of weights. Exposing these weights to the designer would be a good next step to allow more control by the designer.

Research question 1.4. To which extent is the proposed *assisted approach* capable of assisting the designer in making decisions compared to the *manual approach*?

The answer to this research question was researched through the use of a prototype and user study. The CSI results show that the implemented assisted approach did not perform better than the implemented manual approach. The assisted approach did not provide an improved approach to the environment filling process. It did offer great insights on what kind of problems arise when trying to design an approach for this kind of problem. The factor counts of the CSI results support the problems discussed in the introduction. The factor counts show that environment filling relies heavily on 'Exploration', 'Expression', and 'Efficiency'. Although the approach was designed to improve these factors, the end result did not do this. The following causes are derived from the experiment results. First, it is difficult to create automations for creative processes which are able to match the participant's expectations. Each participant has a different idea of what he thinks is part of a certain location and providing any suggestions that lie outside of these expectations are quickly seen as wastes of time and irritating. This comes down to the fundamental problem of what is a "good" suggestion and this relies on aspects that differ per designer, which the current implementation was unable to provide. Secondly, whatever the quality of the suggestions, it is important to provide an easy to use interface. The provided interface heavily influences the assistance value to the participant. When assistance results in too much hassle, the benefit of that assistance quickly diminishes. Thirdly, it is difficult to quantitatively measure the results of a mixed-initiative approach, due to the size of the input and required duration of the usability study. The usability study could be adjusted to be longer or have larger input sets, but this was in our case unfeasible and in a general case unlikely due to the requirements on input and duration. Lastly, the incorporation of narrative to provide extra information to the environment filling process is appreciated. Participants thought the classification of master objects as required to be useful for filling an environment.

There are several general takeaways from this thesis. Regarding the narrative world creation process, explicitness resulted in a cleared process. The explicit relationships also allowed a clear assessment of what was going on inside an environment. An important point to keep in mind is that explicitness should not stand in the way of creative freedom, the current implementation offers not enough relationships to have much freedom and the approach offers no way of easily adding relationships. Mixed-initiative, of which the *assisted approach* is

a first step towards, shows potential for approaching the narrative world creation process. Firstly because participants indicated they saw potential, although it is very difficult to make suggestions match expectations. The same problem exist for generation solutions, but mixed-initiative offers the chance to steer the solution iteratively instead of parameter tweaking before execution in the case of generation solutions. The iterative process is a great benefit of mixed-initiative and the narrative world creation process. The main point of improvement is the quality and certainty of the advice given by the mixed-initiative approach, further research is required and could be of great benefit to the quality of mixed-initiative approaches in general.

Chapter 8

Future work

This chapter discusses the potential minor and major future work in line with the work in this thesis.

8-1 Environment creation

Since environment creation was kept so simplistic, it allows many avenues of improvement. While minor improvements are allowing other shapes of floors, multiple levels, and surfaces at an angle, a major improvement would be to expand the semantic database to facilitate a modular approach for environment creation. With both environment creation and environment filling use the same semantic database offers new and interesting information that can be used during environment filling. For example, the objects used during environment creation could directly influence the possibilities offered during environment filling.

Another major expansion, that is required to move environment creation toward creating a completely open world, is the positioning and combining of environments with each other. Stitching together environments allows the placement and connection of an environment to the rest of the world. A part of combining environments is placing environments inside each other. This hierarchical structure allows the designation of provinces, cities and towns, but also designating the stairs to a basement and the division of a floor in separate rooms, which are each filled individually. Hartsook et al. [2011] has already shown an approach to stitching together environments, but this approach does not use a modular approach nor allows a hierarchical structure.

A last major improvement is to allow the narrative to adjust environments similarly as the narrative is able to adjust the contents of those environments. Changes in environment layout do not occur often in games, perhaps because it is difficult to facilitate, which makes it an interesting subject.

8-2 Narrative and semantic input and combining them

There are four improvements possible for the input which the narrative world creation process can benefit from. First, fleshing out the relationships of the semantic database. The current set of relationships is quite simplified and can be extended to make an interesting basic set of relationships. A start would be to add relationships that cover the different axes of position and orientation. Inspiration for these relationships can come from furniture layouting such as Merrell et al. [2011] or GLUNet [Graphics and Data Visualization group, 2017]. Using this basic set, combining the relationships can be explored further and offers an interesting area of research as it seeks out the maximum capabilities of the semantic model and might require changes to function completely. The current approach also mandates the use of an 'on' relationship for each object, this could be extended to allow objects to be placed on walls and hang from ceilings instead of just on other objects.

Secondly, making use of services. Services are part of the semantic model and can be used to describe the actions that a master object provides to the world. An example of a service is a lamp with the light service that, when added, it includes a 3D space that is tagged as 'lit'. This allows a 'lit' descriptor in the narrative to tell the narrative world creator to place certain objects inside the lit space. This opens up a whole new set of possibilities and offers more influence to the writer.

Thirdly, the used approach to time is limited. A complete timeline would be continuous, use events with durations and allow parallel events to occur. This would complicate the adjustment of environments based on the narrative, but the basic principle remains the same.

Lastly, GLUNet [Graphics and Data Visualization group, 2017] could be used to populate the semantic database and provide a significant set of master objects and relationships to test with. Creating a large number of master objects and relationships from scratch is a lot of work, which is of limited quantity available in research.

8-3 Environment filling

A major improvement to environment filling is the expansion of the assisted approach to a mixed-initiative approach that includes an agent. The agent would be able to determine the certainty of knowing the designer's preference and is able to choose and adjust the provided assistance accordingly. The current approach lacks this knowledge and suffers from that because it makes assumptions on the designer's preferences without certainty. Determining the designer's preferences can be difficult as these can change for each location type. An example of an interaction that can help determine preference is by approving or disapproving of suggestions and immediately generating new suggestions based on this feedback.

There are also several possible minor improvements to the current approach, these are divided into interaction improvements and assistance improvements.

8-3-1 Interaction improvements

The manual actions offered to the designer were quite sober and can be expanded upon. For one, more information should be shown when selecting an object or using mouseovers. Such

as information on what master objects can be attached to the relationships that the object offers. Secondly, snapping objects to each other might be a more intuitive way of instantiating master objects and relationships. The actions offered during the assisted approach also need to be improved, each action could use more designer influence by allowing adjustment of weights and could use filters to help reduce the options quickly. The adjusted weights could also be used by the agent to determine the designer's preferences.

8-3-2 Automation improvements

The assisted approach allows for some interesting future work. The criteria used for the discrete step types can be expanded upon in two ways. Firstly, more interesting criteria can be extracted from the information that is already available. Secondly, additional information can be added to allow more criteria. Designer preferences could be a potential source. The criteria used for the placement algorithm can also be expanded upon. By expanding the criteria the designer has a larger toolkit from which he can choose a subset to apply to the current environment.

8-4 Experimentation

The last potential future work is to verify the narrative world creation process with professional game designers. Connecting research to a real narrative world creation process of a large game developer could result in several benefits. First, it would allow larger input sets as these are already created for the development of the game. Secondly, a study can incorporate exploration of more situations of the process more often, due to the process already being done by the designers. The insights could also be significantly greater as the designers know a lot more about the different aspects of environment creation and filling.

Appendix A

User Interface screenshots

A-1 Manual mode

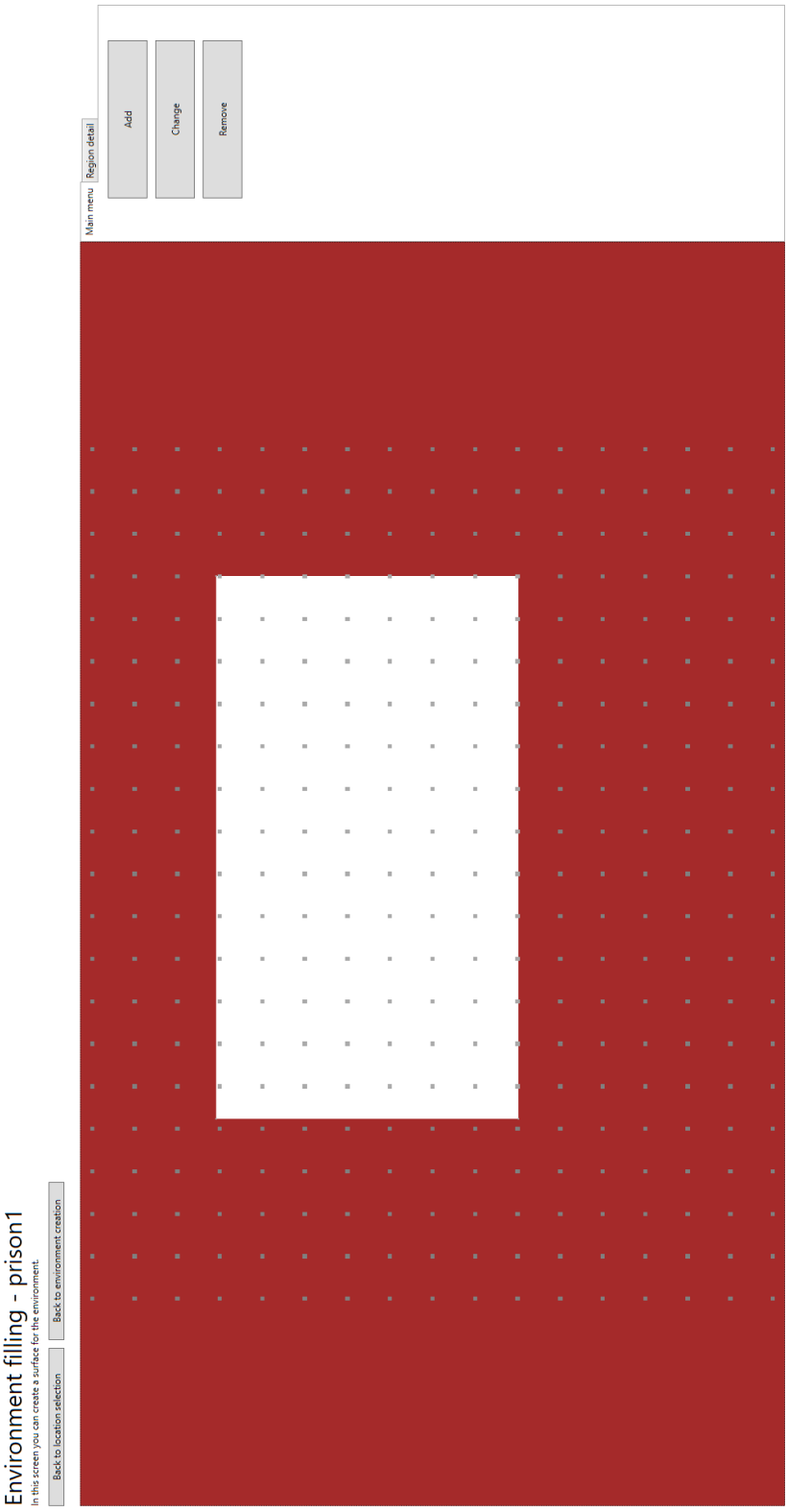

Figure A-1: Overview of the manual mode application.

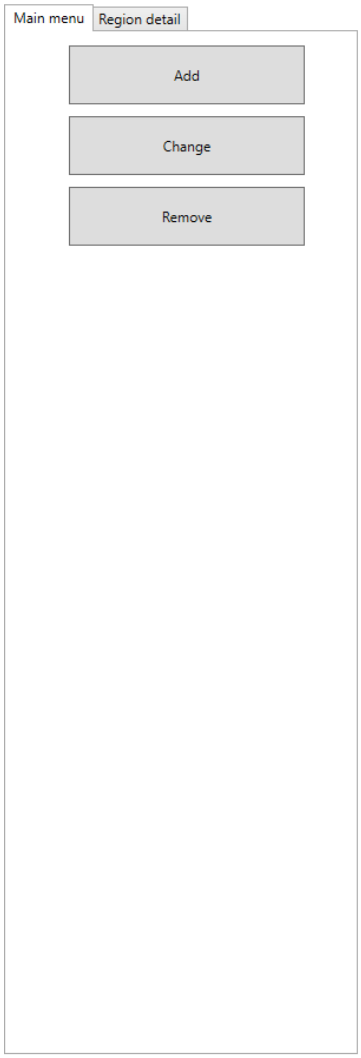

Figure A-2: Main menu of the tabs.

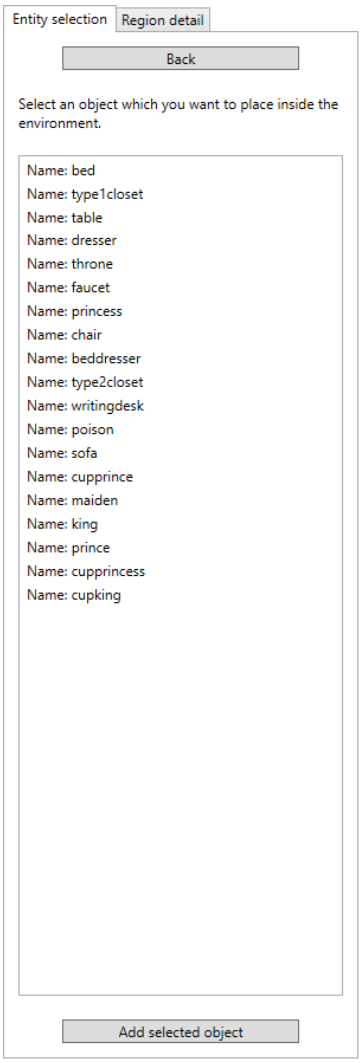

Figure A-3: UI for manually selecting an instance.

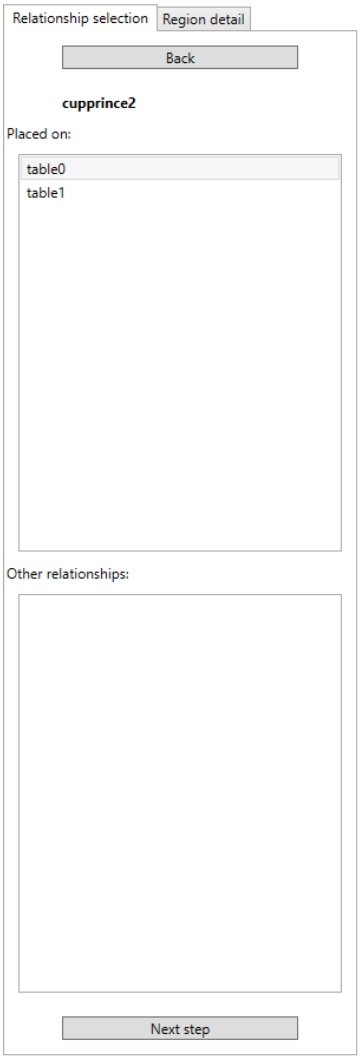

Figure A-4: UI for manually selecting relationships and relationship instances.

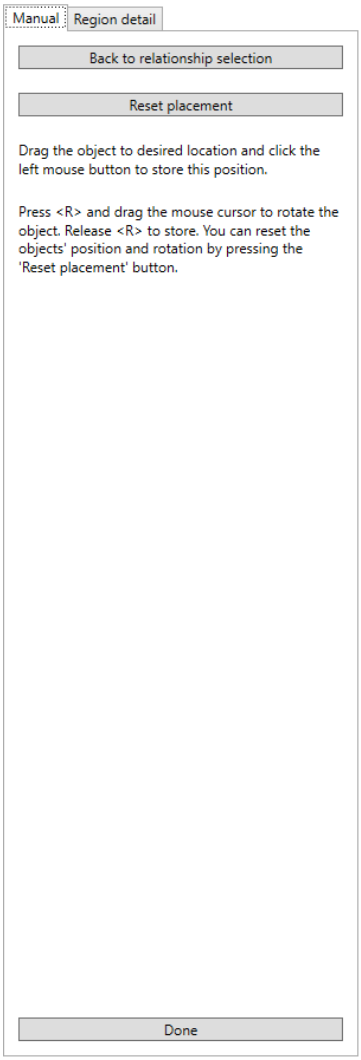

Figure A-5: UI for manually determining the placement.

A-2 Assisted mode

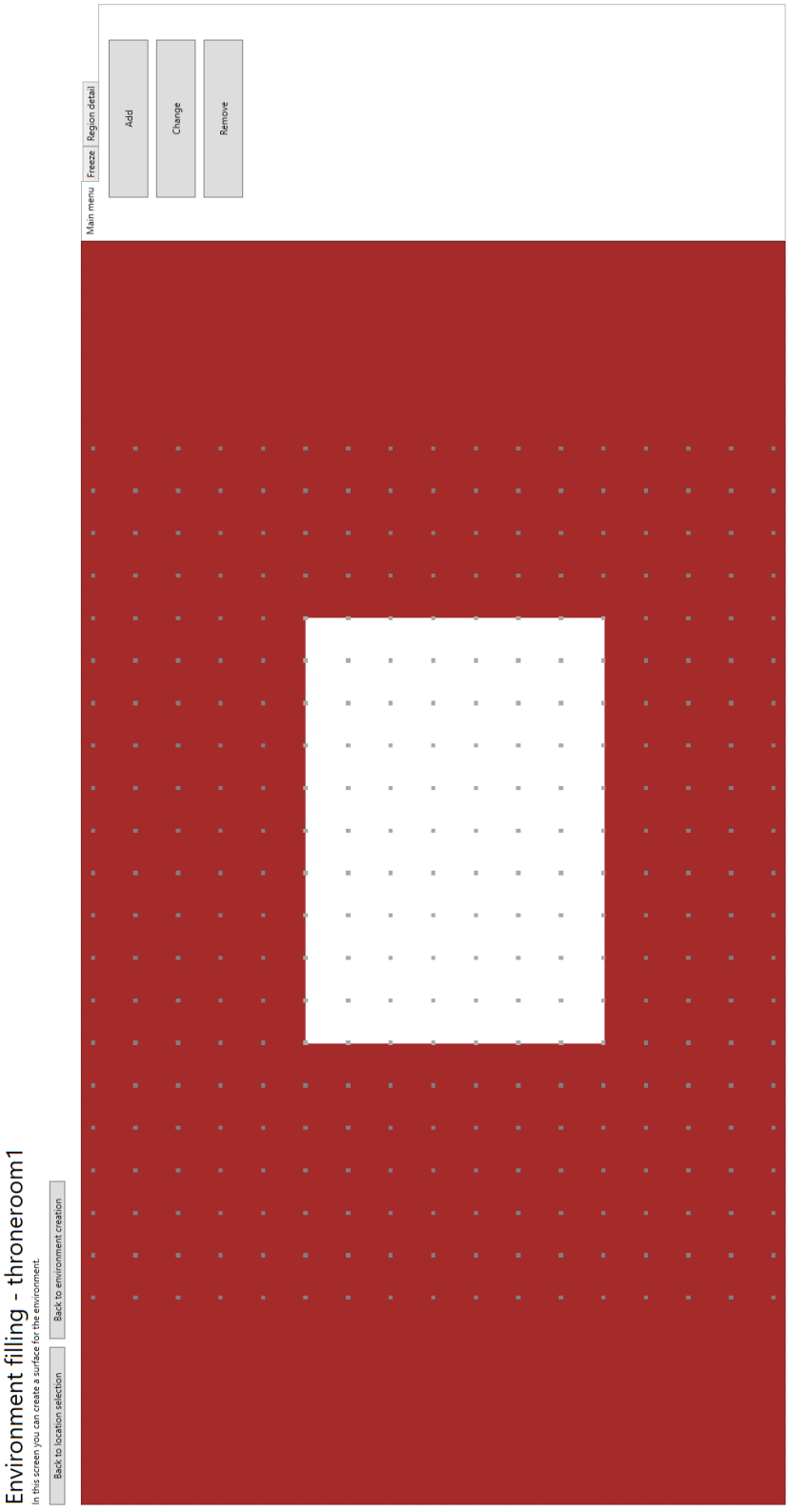

Figure A-6: UI for the assisted placement.

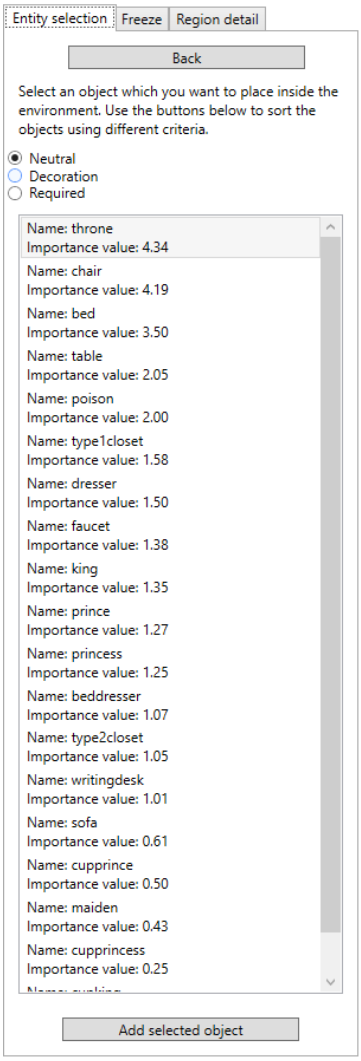

Figure A-7: UI for selecting an instance assisted.

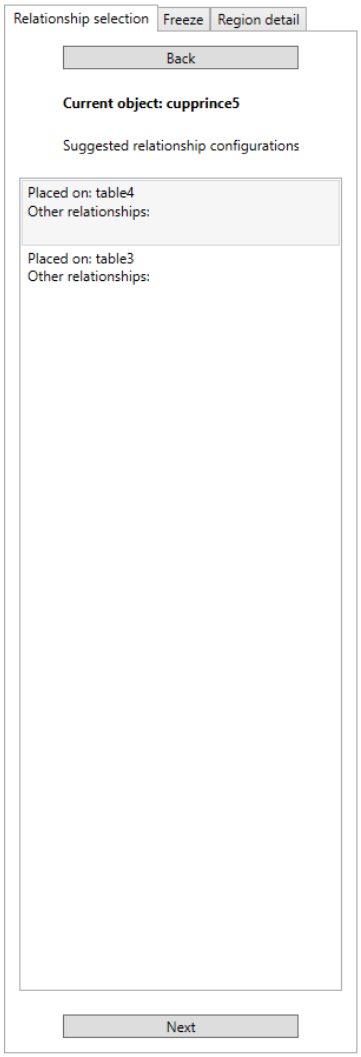

Figure A-8: UI for selecting an instance assisted.

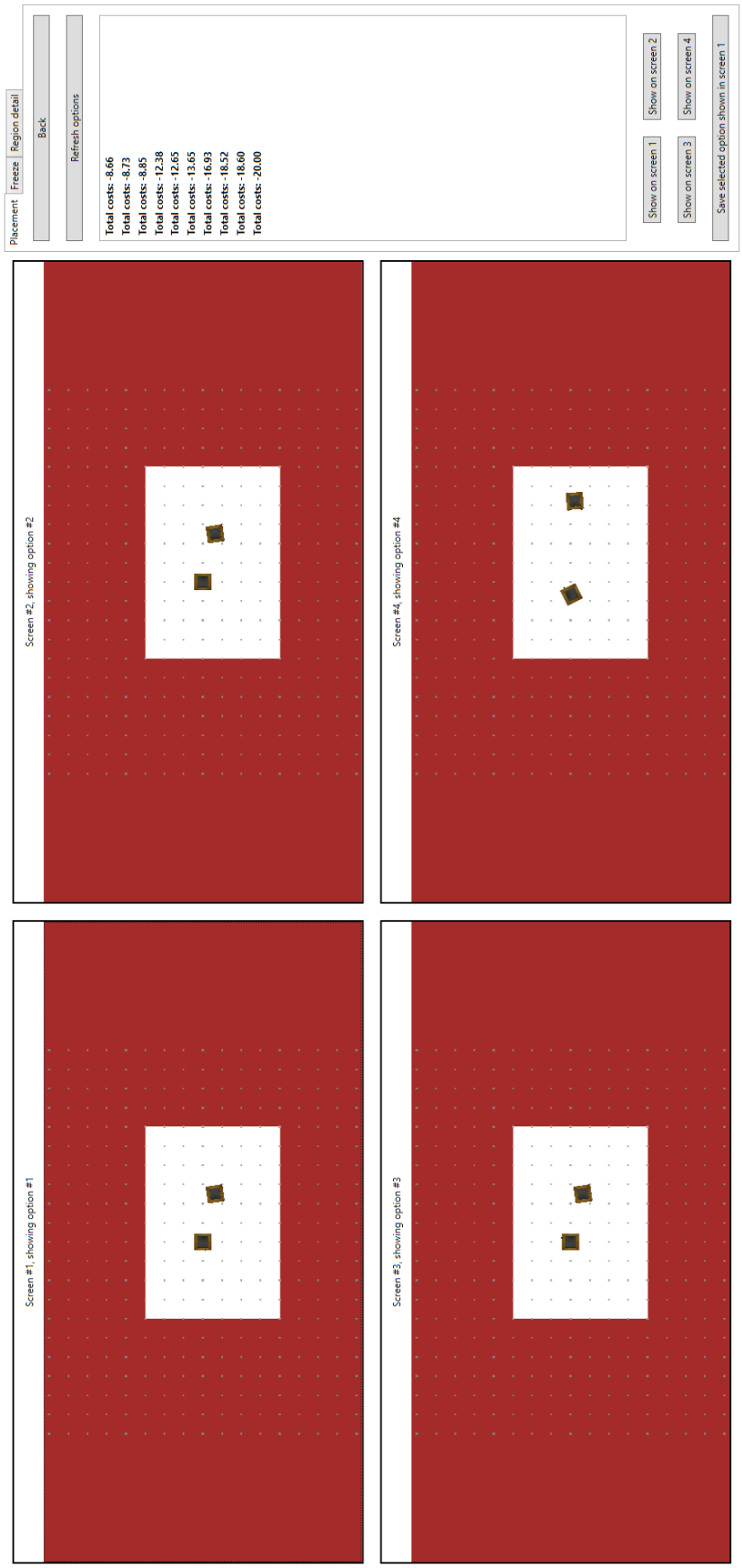

Figure A-9: UI for the assisted placement.

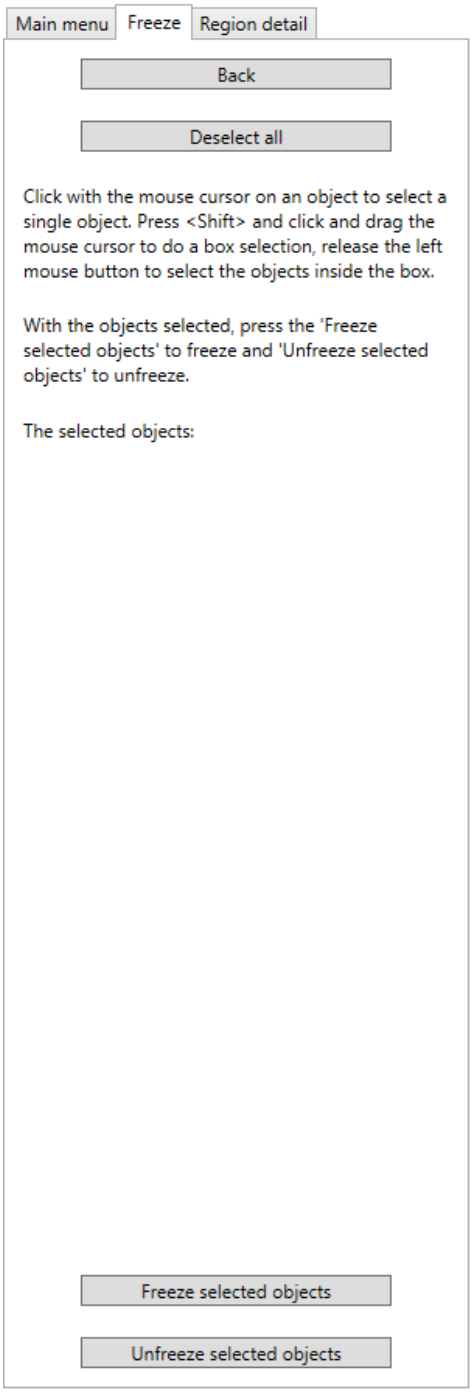

Figure A-10: UI for the freezing of instances.

Appendix B

Input narrative of test

B-1 Domain file

```
(define (domain castle)
    (:requirements :strips :typing)
    (:types
        character thing place - object
        throneroom - place
        bedroomprincess - place
        guestroom - place
        diningroom - place
        kitchen - place
        prison - place
        corridor - place)
    )
    (:predicates
        (dead ?x - character)
        (has ?character - character ?thing - thing)
        (poisoned ?object - thing)
        (poisoned ?subject - character)
        (loves ?character - character ?subject - character)
        (hidden ?character - character)
        (married ?character1 - character ?character2 - character)
        (at ?object - object ?place - place)
        (on ?object1 - thing ?object - thing ?location - place)
        (dark ?location - place)
        (light ?location - place)
        (narrow ?location - place)
        (small ?location - place)
        (sparse ?location - place)
```

```
    (scary ?location - place)
    (around ?object1 - thing ?object2 - thing ?location - place)
    (left ?object1 - thing ?object2 - thing ?location - place)
)
(:action move
 :parameters (?character - character ?from ?to - place)
 :preconditions (and (at ?character ?from) (not (at ?character ?to)))
 :effect (and (at ?character ?to) (not (at ?character ?from)))
)
(:action pickup
 :parameters (?character - character ?object - thing ?place - place)
 :preconditions (and (at ?object ?place) (at ?character ?place))
 :effect (and (not (at ?object ?place) (has ?character ?object)))
)
(:action talk
 :parameters (?character - character ?subject - character ?place - place)
 :preconditions (and (at ?character ?place) (at ?subject ?place))
 :effect
)
(:action talkmultiple
 :parameters (?character - character ?subject - character ?subject2 - character ?place
 :preconditions (and (and ((at ?character ?place) (at ?subject ?place))) (at ?subject2
 :effect
)
(:action fallinlove
 :parameters (?character - character ?subject - character ?place - place)
 :preconditions (and (at ?character ?place) (at ?subject ?place))
 :effect (loves ?character ?subject)
)
(:action hides
 :parameters (?character - character ?place - place)
 :preconditions (at ?character ?place)
 :effect (hidden ?character)
)
(:action poisons
 :parameters (?character - character ?object - thing ?place - place)
 :preconditions (and (at ?character ?place) (at ?object ?place))
 :effect (poisoned ?object)
)
(:action swaps
 :parameters (?character - character ?object1 - thing ?object2 - thing2 ?place - place
 :preconditions (and (and (and ((at ?character ?place) (at ?object1 ?place))) (and (at
 :effect (and (not (poisoned (?object1))) (poisoned (?object2)))
)
(:action placeon
 :parameters (?character - character ?object1 - thing ?object2 - thing ?place - place)
 :preconditions (and (and ((at ?character ?place) (at ?object1 ?place))) (at ?object2
```

```
     :effect (on ?object1 ?object2 ?place)
    )
    (:action drink
     :parameters (?character - character ?object - thing ?place - place)
     :preconditions (and (and ((at ?character ?place) (at ?subject ?place))) poisoned(?obj
     :effect (poisoned ?character - character)
    )
    (:action dies
     :parameters (?character - character ?place - place)
     :preconditions (and (at ?character ?place) (poisoned ?character))
     :effect (dead ?character)
    )
    (:action marry
     :parameters (?character1 - character ?character1 - character ?place - place)
     :preconditions (and (at ?character1 ?place) (at ?character1 ?place))
     :effect (married ?character1 ?character2)
    )
    (:action give
     :parameters (?character1 - character ?character2 - character ?object - thing ?place -
     :preconditions (and (and ((at ?character1 ?place) (at ?character2 ?place))) (and (at
     :effect (and (not (on ?character1 ?object ?place)) (has ?character2 ?object))
    )
)
```

B-2 Problem file

```
(define (problem castle)
    (:domain castle)
    (:objects
        poison cupking cupprince cupprincess table chair throne - thing
        prince princess king maiden - character
        throneroom1 - throneroom
        bedroomprincess1 - bedroomprincess
        guestroom1 - guestroom
        diningroom1 - diningroom
        kitchen1 - kitchen
        prison1 - prison
        corridor1 - corridor)
    (:init
            (at prince throneroom1)
            (at princess throneroom1)
            (at king throneroom1)
            (at maiden guestroom1)
            (at poison bedroomprincess1)
            (at cupking kitchen1)
            (at cupprince kitchen1)
```

```
            (at cupprincess kitchen1)
            (dark guestroom1)
            (light diningroom1)
            (narrow corridor1)
            (scary corridor1)
            (sparse corridor1)
            (small guestroom1)
            (at table diningroom1)
            (at chair diningroom1)
            (at chair diningroom1)
            (at chair diningroom1)
            (at throne throneroom1)
            (at chair throneroom1)
            (left table chair diningroom1)
            (around chair table diningroom1)
    )
)
```

B-3 Plan file

```
(define (plan castle)
    (:problem castle)
    (:steps
        (talkmultiple prince princess king throneroom1)
        (fallinlove princess prince throneroom1)
        (fallinlove prince princess throneroom1)
        (hides maiden guestroom1)
        (move prince throneroom1 guestroom1)
        (talk princess king throneroom1)
        (move princess throneroom1 corridor1)
        (move princess corridor1 guestroom1)
        (talk princess prince guestroom1)
        (move princess guestroom1 bedroomprincess1)
        (pickup princess poison bedroomprincess1)
        (move princess bedroomprincess1 guestroom1)
        (talk prince princess guestroom1)
        (give princess prince poison guestroom1)
        (move prince guestroom1 diningroom1)
        (move princess guestroom1 diningroom1)
        (move king throneroom1 diningroom1)
        (move maiden guestroom1 kitchen1)
        (hides maiden kitchen1)
        (talkmultiple prince princess king diningroom1)
        (move prince diningroom1 kitchen1)
        (poisons cupking poison kitchen1)
        (move prince kitchen1 diningroom1)
```

```
        (swaps maiden cupking cupprince kitchen1)
        (pickup maiden cupprince kitchen1)
        (move maiden kitchen1 diningroom1)
        (placeon maiden cupprince table diningroom1)
        (drink prince cupprince diningroom1)
        (dies prince diningroom1)
        (talk princess king diningroom1)
        (move princess diningroom1 prison1)
        (marry king maiden diningroom1)
    )
)
```

Appendix C

Test document

Test introduction

This test is designed to test the interaction between a user and a computer program when creating and filling a 3D world. The purpose of this 3D world is to tell a story and, when finished, to be used in a game. The computer program offers the tools needed to create a floor surface, add 3D objects to to this surface and, in that way, fill an environment with objects. The world is split into separate locations that will be individually filled.

Following is a summary of the story that is used as input:

A prince has arrived to a castle and is greeted by the princess and her father, the king, in the throneroom. Here the prince and princess fall in love with each other. The maiden distrusts the prince and hides herself in the guest room. Later, the prince and princess meet in the guest room and discuss their plan to kill the king. The maiden hears this and prevents the assassination by pouring over the poison from the cup of the king into the cup of the prince. The cups are served during dinner and the prince drinks the poison and dies. The maiden explains the situation, the king throws the princess into the prison and marries the maiden.

The 3D objects that are available are connected with each other through relationships. These relationships allow descriptions on whether one object is “on” another and whether one object “left” or “right” from another object.

The computer program provides several ways to fill an environment: Adding, changing and removing objects. As well as two modes, one that includes assistance and one without. The without assistance is straightforward, it allows you to perform the actions manually. The assisted mode will include assistance during addition and provide an additional way of changing the environment. This will incorporate information to help you make decisions on what object to add next and how to position objects.

Test execution

Please fill in the Starting Questionnaire

A first introduction

The interface shown shows the locations that are available. The screen that shows this will also show the environment that is created, and later, filled. In the top there are buttons to navigate through the different pages of the application. On the right there is a column that will be filled with information and options concerning the current screen. The different functions will first be walked through once.

Read the following directions carefully.

Selecting an environment
First fill a location manually:

- Select “Prison1” from the graph and click on “Create/fill this location”

Creating an environment
The environment doesn’t have a floor on which objects can be placed. The horizontal and vertical space between two dots is one meter. You can click and drag in the colored window to create a floor for your environment.

- Press “Next” on the right side when you have finished creating the environment.

Filling an environment
Before looking at how you can fill the environment, you need to know how to navigate the 3D world that is rendered. You can change the camera’s position by using the arrow keys on the keyboard. You can zoom in and out using the “+” and “-” keys on the keyboard.

You now have three options, “Add”, “Change” and “Remove”.

- Let’s add an object, click on “Add”.

- Select “table” and click on “Add selected object” in the bottom right corner.

You can now select the relationships this object has with other objects that were already placed. This should currently show a single option, namely the “on floor” relationship. This will place the selected object on the floor.

- Click on “Next step”

Now you can place the object in the environment by clicking wherever you see fit. You can rotate the object by holding “R” and dragging the mouse. Make sure to place the object inside the environment. If you are not happy, you can always reset the placement.

- When you are done, press on “Done” in the bottom right corner. This will return you to the main menu.

With your first object added, let’s look at the other options available in the main menu. Let’s change the table object that we added.

- Click on “Change”.

You are now able to select one or more objects and change their position and orientation. Select the table that we added by clicking on it. The table is now highlighted. Another way of selecting objects is by pressing “Shift”, clicking the left mouse button and dragging the mouse. This will create a box that will select every object part of this box.

- Adjust its position by first pressing “M”, then clicking and holding the left mouse button while dragging.

- Adjust its orientation by pressing “R” and dragging the mouse.

You can highlight more than one object and adjust their position and orientation simultaneously.

- When ready, press on “Back” to return to the main menu.

Lastly, let’s look at removing the table we have just added.

- Select “Remove” from the main menu

The same as for “Change” you can now select one or more objects by clicking on them.

- Click on the table.

With one or more objects selected, you can now press on “Remove selected objects” to remove them. This will remove the object and any objects placed on top of it.

- Click on the “Remove selected objects” button

Each location has a couple of required objects, please fill the environment with those objects and add additional objects as you see fit. The required objects can be found under the **region**

detail tab next to main menu and will be marked green when they have been added to the environment.

You should now be able to complete the ‘prison1’ environment. Return to this document when you have finished filling your first location.

When done with ‘prison1’, fill the “Bedroomprincess1”. The “Bedroomprincess1” is an environment that will also be filled manually.

When done with both of these environments please ask for the **Creative support index questionnaire** on manual filling.

Assisted filling

Now it is time to fill an environment with assistance.

- First return to location selection screen by pressing “back to location selection” in the top left corner.

- Now select “Corridor1” from the graph and click on “Create/fill this location”

Same as without assistance, you must now create a floor for the environment on which you can place objects.

- Press “Next” on the right side when you have finished creating the environment.

Now it is again time to fill the environment. Two things are different for the assisted mode in comparison to the manual mode. The addition of an object is done completely with assistance, which we will zoom in more later. Secondly, changing includes a second assisted mode.

Let’s add an object to try out the assisted mode.

- Click on “Add”

The list of objects is now annotated with an importance value. This shows how important the object is based on a number of criteria which are combined into the importance value. There are three modes which each prefer different criteria. The “Neutral” option does not prefer either objects that are mentioned by the story, also known as required, or objects that are purely meant as decoration. The other two options prefer different criteria to focus more on required and decorative objects respectively.

- Use the different modes and look through the options available. Select an option that you prefer and click on “Add selected object”

After this, you again must select the relationships for the object you wish to add. Instead of selecting from all available options, you can now select from a list of suggested options. It should only include a single option now as there is only a single floor.

- Click on “Next” when you have chosen an option.

With the object and relationships selected, it is time to place the object. It shows the four best options that it came up with in the four screens on the left. On the right you can find more options which you can compare. The options are ordered using ‘total costs’. ‘Total costs’ consists of several cost functions that deduct points for errors made. The fewer errors the closer

to 0 and the better the program thinks the option is. Comparison can be done by selecting an option from the list and selecting one of the “Show on screen …” buttons. This will show that option on the specified screen. When pressing “Save …” in the bottom right, you save the option in screen 1.

- Select an option and show it on screen 1
- Press the “Save selected option shown on screen 1” button to save it

With the object added we can now look at the assisted changing mode.

- Select “Change” from the main menu

Instead of going straight to manual change, you can now select from either “Manual changing” or “Assisted changing”.

- Select “Assisted changing”

This works the same as the placement of a new object. This will find new positions and orientations for each object that is not frozen. To adjust the frozen objects select the “Frozen” tab next to the “Placement” tab on the right of the screen.

- Select the “Freeze” tab

The freeze tab is always available, allowing you to select objects in the same way as removal and manual changing. You can change whether the selected objects are frozen or not by pressing the “Freeze selected objects” and the “Unfreeze selected objects” buttons in the bottom right.

- Press on the “Placement” tab

With the objects frozen or not, you can refresh the options and assess them again.

- Select an option, show on screen 1 and save or click on “Back” to go back to the mainmenu

Remove is still the same.

You now have all the tools to create and fill environments. Please finish filling this environment.

When done with “Corridor1”, please fill the “Throneroom1” environment.

With all of these environments filled, please ask for the **Creative support index questionnaire** on assisted filling and **final questionnaire**.

Appendix D

Start questionnaire

Starting Questionnaire

1. **What is your date of birth?**

   Example: December 15, 2012

2. **How would you rank your experience with computers?**
   Mark only one oval.

	1	2	3	4	5	
Very little experience	◯	◯	◯	◯	◯	A lot of experience

3. **How would you rank your experience with stories?**
   Are you an avid reader, Dungeons and dragons player, game, etc.?
   Mark only one oval.

	1	2	3	4	5	
Very little experience	◯	◯	◯	◯	◯	A lot of experience

4. **How would you rank your experience with designing and furnishing environments?**
   Mark only one oval.

	1	2	3	4	5	
Very little experience	◯	◯	◯	◯	◯	A lot of experience

Appendix E

Final questionnaire

Final Questionnaire

* Required

Selection of objects to add

1. **Did the 'importance value' associated with the options offer more clarity or cause more vagueness?** *

2. **Did focusing the importance value on a particular goal, such as 'required' or 'decoration', allow you to better find what you were looking for?** *

Relationship selections

3. **Did you feel like the offered suggestions were restricting you or were they offering aid on how to combine the different relationships available?** *

Untitled Section

4. **Did the 'total costs' associated with each option offer more clarity or cause more vagueness?** *

5. **Did you find the comparison of options through multiple screens a good way to inspect the options?** *

Powered by Google Forms

Bibliography

Icaps competitions 1998 till present, 2017. URL http://icaps-conference.org/index.php/Main/Competitions.

Bob Bates. Game design second edtion. *Boston: Thomson Course Technology*, 2004.

Joel Burgess. Skyrim's modular approach to level design, 2013. URL http://www.gamasutra.com/blogs/JoelBurgess/20130501/191514/Skyrims_Modular_Approach_to_Level_Design.php.

Erin A Carroll and Celine Latulipe. The creativity support index. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, pages 4009–4014. ACM, 2009.

Heather Maxwell Chandler. *The game production handbook*. Jones & Bartlett Publishers, 2009.

David Elson. Dramabank: Annotating agency in narrative discourse. In *LREC*, pages 2813–2819, 2012.

Gérard Genette. *Narrative discourse: An essay in method*. Cornell University Press, 1983.

Computer Graphics and TU Delft Data Visualization group. Glunet, 2017. URL https://graphics.tudelft.nl/glunet/.

Ken Hartsook, Alexander Zook, Sauvik Das, and Mark O Riedl. Toward supporting stories with procedurally generated game worlds. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 297–304. IEEE, 2011.

Patrik Haslum. Pddl basics. URL http://users.cecs.anu.edu.au/~patrik/pddlman/writing.html.

Koen V Hindriks, Frank S De Boer, Wiebe Van der Hoek, and John-Jules Ch Meyer. Agent programming in 3apl. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 159–166. ACM, 1999a.

Eric Horvitz. Uncertainty, action, and interaction: In pursuit of mixed-initiative computing. *IEEE Intelligent Systems*, 14(5):17–20, 1999b.

Cameron Koch. How big is 'the witcher 3'? we break down the map size, game length and install requirements, 2015. URL http://www.techtimes.com/articles/50240/20150502/the-witcher-3-is-how-big-we-break-down-the-size-of-cd-projekt-reds-ambitious-rpg.htm.

Ben Kybartas and Rafael Bidarra. A survey on story generation techniques for authoring computational narratives. *IEEE Transactions on Computational Intelligence and AI in Games*, 2016.

Michael Lebowitz. Creating characters in a story-telling universe. *Poetics*, 13(3):171–194, 1984.

Michael Lebowitz. Story-telling as planning and learning. *Poetics*, 14(6):483–502, 1985.

Michael Lebowitz. Planning stories. In *Proceedings of the cognitive science society, Hillsdale*, pages 234–242, 1987.

Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. Limitations of choice-based interactive evolution for game level design. In *Proceedings of AIIDE Workshop on Human Computation in Digital Entertainment*, pages 213–220, 2012.

Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. Sentient sketchbook: Computer-aided game level authoring. In *FDG*, pages 213–220, 2013.

Antonios Liapis, Gillian Smith, and Noor Shaker. Mixed-initiative content creation. In *Procedural Content Generation in Games*, pages 195–214. Springer, 2016.

Ricardo Lopes, Tim Tutenel, and Rafael Bidarra. Using gameplay semantics to procedurally generate player-matching game worlds. In *Proceedings of the The third workshop on Procedural Content Generation in Games*, page 3. ACM, 2012.

Emma McDonald. The global games market will reach 108.9 billion in 2017, 2017. URL https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/.

Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. In *ACM Transactions on Graphics (TOG)*, volume 30, page 87. ACM, 2011.

Lee Perry. Modular level and component design, 2002. URL https://docs.unrealengine.com/udk/Three/rsrc/Three/ModularLevelDesign/ModularLevelDesign.pdf.

Mark O Riedl and Robert Michael Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268, 2010.

Satsuma. Satsuma energy based graph visualization algorithm, 2016. URL http://satsumagraph.sourceforge.net/doc/html/class_satsuma_1_1_drawing_1_1_force_directed_layout.html.

Ruben Michaël Smelik, Tim Tutenel, Klaas Jan de Kraker, and Rafael Bidarra. Interactive creation of virtual worlds using procedural sketching. In *Eurographics (Short papers)*, pages 29–32, 2010.

Ruben Michaël Smelik, Tim Tutenel, Klaas Jan de Kraker, and Rafael Bidarra. A declarative approach to procedural modeling of virtual worlds. *Computers & Graphics*, 35(2):352–363, 2011.

Gillian Smith, Jim Whitehead, and Michael Mateas. Tanagra: A mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pages 209–216. ACM, 2010.

Gillian Smith, Jim Whitehead, and Michael Mateas. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):201–215, 2011.

Superannuation. How much does it cost to make a big video game?, 2014. URL http://kotaku.com/how-much-does-it-cost-to-make-a-big-video-game-1501413649.

Tim Tutenel. Semantic game worlds. 2012.

Tim Tutenel, Rafael Bidarra, Ruben M Smelik, and Klaas Jan De Kraker. Rule-based layout solving and its application to procedural interior generation. In *CASA Workshop on 3D Advanced Media In Gaming And Simulation*, 2009.

Stephen G. Ware. Pddl basics. URL https://nil.cs.uno.edu/projects/glaive/.

Stephen G. Ware. The Intentional Fast-Forward narrative planner. In *Proceedings of the 5th Intelligent Narrative Technologies Workshop at the 8th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 57–62, 2012.

Stephen G. Ware and R. Michael Young. Glaive: a state-space narrative planner supporting intentionality and conflict. In *Proceedings of the 10th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 80–86, 2014. (awarded Best Student Paper).

Daniel S Weld. An introduction to least commitment planning. *AI magazine*, 15(4):27, 1994.