CHALMERS | UNIVERSITY OF GOTHENBURG

Tabula Imaginarium

- An Ipad application for aiding a spatially separated tabletop role-playing game.

Master of Science Thesis in the Master Degree Programme, Interaction Design

Jonathan Johansson

Peter Lundberg

Examiner: Staffan Björk

Chalmers University of Technology

University of Gothenburg

Department of Computer Science and Engineering

Göteborg, Sweden, December 2011

Tabula Imaginarium

An Ipad application for aiding a spatially separated tabletop role-playing game.

Jonathan Johansson

Peter Lundberg

Examiner: Staffan Björk

Chalmers University of Technology

University of Gothenburg

Department of Computer Science and Engineering

SE-412 96 Göteborg

Sweden

Telephone + 46 (0)31-772 1000

Cover:

The cover picture was taken during a roleplaying where Tabula Imaginarium was used.

Abstract

This report describes the conceptualization and development of an Ipad application, called Tabula Imaginarium, with the purpose of aiding a player of a tabletop role-playing game that is spatially separated from the other participants. Guided by the main design goal of being an application that should be a natural part of the role-playing activity, even if a player is not spatially separated from the other participants, Tabula Imaginarium presents a way to easily and quickly draw maps on the fly during a role-playing session, and live-time synchronization of these maps between applications on different Ipads. In order to do this, it contains different drawing features as well as placement and movement of objects and character tokens.

The report summarizes the features of similar applications, such as Virtual Tabletop applications for the PC and map-drawing applications for Ipad. It also presents theory that describes the activity of tabletop role-playing in depth. In the conceptualization part the report describes brainstorming sessions as well as application of theory in order to pinpoint parts of the tabletop role-playing activity that is important to transfer to the Ipad device. This can be seen to mainly focus on the immersion of the spatially separated player. The iterative nature of development is also depicted in its five cycles of design, implementation, user testing and analysis, and ended by its final test with remote test persons. In the last parts of the report, the features of Tabula Imaginarium are described in detail, while presenting suggestions of future improvements that could further aid in immersing a spatially separated player.

Table of Contents

1. Introduction

This project concerns the creation of software tools in order to aid the experience of people engaged in tabletop role-playing games. A tabletop role-playing game is a social activity that combines storytelling, role-playing and gaming with a division of participants into players and referee. Players take the role of a character in a fictional make-believe world, while the referee takes on the role of everything else that inhabits that world. The referee presents the players with scenarios containing problems for the players to solve. Events are resolved by the players describing what they do, and the referee interpreting and deciding upon the results of these actions. In order to interpret actions in a fair way, tabletop role-playing games present sets of rules that the referee can use, much like in any other type of game.

The idea of using software tools to in some way aid the activity of playing tabletop role-playing games is not new, and the most notable example of this is probably Virtual Tabletop applications (RPG Virtual Tabletops, 2012) developed for the PC. Other examples that have inspired this project are the Undercurrents project (Bergström, Jonsson, Björk, 2010) and map-creation tools for the Microsoft Surface (Microsoft.com, 2011).

With the advent of the tablet device, it seemed like an interesting idea to try to use this type of device in this context, as it is small enough to not obstruct the gaming area. As we are two master students at the Interaction Design program with backgrounds in Computer Science, the development of an actual application seemed like the most fitting way for us to explore this area. Both of us also had previous knowledge and interest in tabletop role-playing games.

Since it is a common problem during tabletop role-playing sessions that one player that is usually present might not have the possibility to attend, we decided to try to aid in solving this problem specifically. We wanted to see what an Ipad application can do to allow the spatially separated player to still participate.

1.1. Purpose

The purpose of this project was to explore ways to aid in creating an experience of togetherness between tabletop role-players when one person in a group cannot physically attend a role-playing session.

If one player is not physically there, it creates a rift between players that we would try to bridge. The intention was to, by the use of cameras, audio and Ipads, create a feeling of togetherness between the players, even though all of them are not at the same location. This focused more on social presence than on long-term effects of the perceived relationship of the group. The hope was to create an atmosphere where even the player that was not physically attending the role-playing session did not feel left out, without hindering the possibility of the others to be engaged in the activity.

1.2. Goal

The main goal of this project was to produce an application for the Ipad device that contained features that made the activity of tabletop role-playing transferable to that medium. It did, however, not only look at transferring pure game elements, but also on how to transfer social activities. The application was designed to be as general as possible, and not be specific for one type of tabletop role-playing games. We assumed that this application would be used by all participants of a role-playing session, each person having their own Ipad.

The second goal was to perform test runs with the application, during which we used video/audio conference systems, to study how the application could be used to improve the social presence of a player who was not physically present.

It was possible that the application would feature functionality such as simple map making and drawing tools, sharing of pictures, die rolling, person to person or global chat, sharing of PDF documents, sharing of background music or a video conference.

1.3. Question

As the goal of the project was the creation of an application for an Ipad device, which was a quite new device that had not been explored that much; our question had to assume the use of an Ipad. Only looking at our presumed application without the context of the device itself would not suffice though, and as such, the question would have to indicate a broader use of the Ipad as a device, with our application as well as other features of the Ipad. The specific purpose of the application was to help in the creation of togetherness in a role-playing group with one player not attending the session in person though, and thus our question became as follows:

How can one use an Ipad to bridge the distance between a group of players and a player that is not physically attending a role-playing session?

1.4. Delimitations

When it came to bridging distance, we only considered the case where there was only one person located elsewhere when a role-playing session took place. We saw this as the most problematic case; as only one person was outside the group, and that player would probably feel alienated from the rest. We therefore focused our efforts on the experience of this player. Also, because of limited resources, we wanted to reduce the number of cases we needed to consider. Another limitation we decided upon was that we would focus on the case where all players had access to their own Ipad.

The project also focused on tabletop role-playing that tried to have a balanced approach between narrative and gameplay. As such, we were not looking into solving this problem for free form role-playing and similar styles or for games that entail a minimum of narrative.

2. Background

The very idea of this project was loosely based on the Undercurrents project (Bergström, Jonsson, Björk, 2010), which also employed a personal computer device for each participant of a role-playing session, although without focusing on aiding spatially separated participants specifically. Undercurrents also focused on providing additional information channels during a session, while our project more concerned the perceived experience of a remote player. As such, it was necessary to look at Virtual Tabletop applications (RPG Virtual Tabletops, 2012), which are actually made with spatial separation in mind.

Since the Ipad utilizes a touch-screen it was also quite natural to look at aids made for tabletop role-play for the Microsoft Surface, such as Tisch (Fröhlander and Hartelius, 2011) and SurfaceScapes (SurfaceScapes, 2010).

2.1. Virtual Tabletop Applications

Although distinct from our problem in that they focus on a situation where *all* participants of a role-playing session are spatially separated rather than just *one player*, Virtual Tabletops still have functionality that is similar to what we wanted to create.

The RPG Virtual Tabletop webpage mentions a quite extensive list of Virtual Tabletop applications (RPG Virtual Tabletops, 2012), of which we have looked at some of the more popular alternatives (Enworld.org Forum, 2010) (Community.wizards.com Forum, 2009), namely: D20Pro, Fantasy Grounds II, and Map Tool. All of these applications are made for PCs.

Similar systems for the Ipad have also been released, such as: Battle Map, RPG Cartographer, and Dungeon Mapp, although these do not focus on spatially separated participants, though the provided functionality is still somewhat similar.

Virtual Tabletop applications generally include features such as map and image sharing with movable character tokens, dice rolling, and chat systems (RPG Virtual Tabletops, 2012), although the Ipad applications focus solely on maps and tokens.

Matrices listing the individual functionality of each of the mentioned applications can be found in Appendix I and II.

2.1.1. D20Pro

D20Pro is an application made for the PC platform to support and enhance tabletop role-playing. D20Pro integrates the common virtual tabletop features: maps, dice, and chat with more game-related features like combat resolution, initiative order, and effect tracking. Because of this D20Pro can also be used as an aid when role-playing even if the players are co-located, although some of these features also gears the application towards use with a specific role-playing game system; more specifically: 3.5 OGL, although it is stated that D20Pro can be used while playing with any D20-based system. (D20pro.com 2011)

2.1.2. Fantasy Grounds II

Fantasy Grounds II focuses on enhancing the narrative and mood when playing tabletop role-playing games while spatially separated. It combines the common features: maps, dice, and chat with more aesthetical functionality, such as customizable portraits, 3D dice, interactive handouts, and screen tinting. The interface itself is also true to the theme of the supported activity. (Fantasygrounds.com 2010)

2.1.3. MapTool

MapTool is a free alternative to D20Pro and Fantasy Grounds II, which focuses on customizability, with the included functionality of custom-created content such as scripts and macros. MapTool is also completely open source, which gives users the possibility to freely modify the application itself, and share their additions with others. MapTool also has the common features: maps, dice, and chat. (RPTools.net 2011)

2.1.4. Battle Map

Battle Map is an application for creating maps, both on the fly and while preparing a future role-playing session. These maps can then either be shown on the Ipad or on an external screen. It supports fog of war, which can be turned on and off separately for the Ipad and the external screen. The referee cannot add his own content, but the content that comes with Battle Map is, according to its developers, plenty, well-made, and regularly updated. (RazeWare.com 2011)

2.1.5. RPG Cartographer

RPG Cartographer is also an application for map creation, with a feature set which focuses completely on this. The user can scale objects, work in different layers, export and import maps, and visibility control. According to its developers, the application contains over 800 unique map tiles and 500 character tokens. (ITunes.apple.com 2011)

2.1.6. Dungeon Mapp

Dungeon Mapp distinguishes itself from its competing Ipad applications by supporting more than just map making. It also includes dice rolling as well as initiative tracking, health tracking and status effects on tokens. The user can also link maps to other maps, so that the transition between them can be done in a fluid manner.

The application is also specifically geared towards on the fly use, as the interface is supposed to allow for fast creation of maps with the hope of creating rooms in about the same time as it takes for the referee to describe them. (Ambitioussoftware.com 2011)

2.2. Tisch

Tisch is an application for the Microsoft Surface developed as a Master Thesis project aimed at supporting tabletop gaming. The primary focus was on tabletop role-playing games by the use of new technology; in this case Microsoft Surface. Tisch features tools for creating maps, and emulating simple board games. Some of these features are drawing tools, grid nets with either squares or hexes, and movable character tokens. Tisch also includes features for increasing immersion, such as line of sight, fog of war, day and night cycles and weather overlays. Tisch uses two ways of input, either touch input or input via tag recognizers. The tag recognizers are mostly used for accessing menus and changing modes, while the touch input was used for drawing. The tokens could be assigned to tags, so that tokens could be moved with figures that had the tags under them, making it more realistic to real tabletop games. (Fröhlander and Hartelius, 2011)

3. Theory

As role-playing games contain elements of role-playing, gaming and storytelling, they can, and have been, described in many different ways. For the purpose of this project, the expression is generally used to reference the activity of playing a tabletop role-playing game, as opposed to live-action role-play or educational role-play. In order to define this activity in a way that is meaningful in the context of this project, we have been looking at how role-playing games describe themselves, and how these games have been described in research.

3.1. Self-descriptions from Rulebooks

By looking at how rulebooks from different genres of role-playing games define themselves, it becomes clear that they stress different areas, although some particular things are generally quite similar. They all separate the participants of the game into a referee (referred to as Game Master, Dungeon Master, Storyteller, Storyguide, Keeper, Narrator, etc.) and players. The players take the role of protagonists in a story - deciding what they do and say, and the referee acts as a narrator that controls the rest of the world (Pathfinder Roleplaying Game 2009, p.8). It is possible for players to each control more than one character, although this is uncommon in most games. This type of acting done by both players and the referee, involves improvisation, since there is no script other than notes and outlines used by the referee. The story is told collaboratively by the players and the referee. (Call of Cthulhu 2005, pp.24-25) The referee presents stories and situations, describes the world, plays other characters and adjudicates the rules (Dark Heresy 2008, p.7), and can be seen as a combination of referee, storyteller, opponent and game designer (Mutant Chronicles 1997, p.49).

Role-playing games involve the telling of stories that take place in the imagination of the participants (Star Wars Roleplaying Game 2007, p.7). It is not a game in the regular sense of the word, but it is a way of experiencing adventures beyond everyday life (Drakar och Demoner 1991, p.4). It is not a passive experience, as all participants have the possibility to change the course of events. It is a creative experience, where the participants develop a story together. (Gemini 1998, p.18) It is founded on dialogue between players and the ability of imagining a made-up world (Eon Deluxe 2002, p.10).

Rules are also an important part of a role-playing game, although exactly how important may differ between games. Rules are used to interpret the actions of the characters in the story (Call of Cthulhu 2005, pp.24-25). They are used by the referee to present a balanced setting, where adventures seem real (Warhammer Fantasy Role-play 1995, p.9). They make the game world understandable, define what is possible, and offer objective determination of success and failure (Call of Cthulhu 2005, pp.24-25). Warhammer Fantasy firmly states that the rules are guidelines only, and that they are subject to change whenever a referee so wills (Warhammer Fantasy Role-play 1995, p.9). The Dungeon Master's Guide however states that all rules are created with great care and changing them should be done only after careful consideration of the consequences (Dungeon Master's Guide 2000, p.11). Vampire: The Masquerade has its own view on rules, stating that they should be used as little as possible in order to tell thrilling stories, and that whenever story and rules conflict, story should always take precedence (Vampire: The Masquerade 1998, p.21). Rules also tend to involve randomness by the use of dice. Dice are rolled to resolve encounters, since it adds drama, promotes surprise and keeps people honest (Call of Cthulhu 2005, pp.24-25).

3.2. Descriptions in Research

MacKay defines role-playing games as "*an episodic and participatory story-creation system that includes a set of quantified rules that assist a group of players and a gamemaster in determining*

how their fictional characters spontaneous interactions are resolved". (2001, pp.4-5) Although it has important points, this definition is broad enough to contain activities that are beyond that of a tabletop role-playing game, such as live-action role-playing games and play by mail-like role-playing activities.

Cover, on the other hand, uses a more fitting definition as she addresses tabletop role-playing games specifically, defining them as "*a type of game/game system that involves collaboration between a small group of players and a gamemaster through face-to-face social activity with the purpose of creating a narrative experience*". A key part is the presence of a gamemaster; i.e. a referee, as well as collaboration. Face-to-face social activity is required for it to fit the definition of a tabletop game. (2010, p.168)

Cover also defines tabletop role-playing games by looking at its antecedent genres: wargaming (game) and fantasy literature (narrative) (2010, p.9), recognizing that it contains parts from both.

She notes that tabletop role-playing games have productive interactivity rather than selective, which essentially means that the participants can add ways where the story could go, rather than just choosing between given alternatives. It is also possible to negotiate the consequences of actions and possible ways where the narrative could go, in a very interactive way, as you simply talk to the referee. Unlike computer games, a referee has a greater capacity for interpreting non-standard actions (2010, p.33). This gives the participants what Cover calls narrative agency (2010. p.47), which she differentiates from interactivity, since agency requires the possibility to shape the game system itself, not only to interact with it. Actual narrative agency is the possibility to control the story, which she also notes is one of the main reasons why people play tabletop role-playing games. (2010, p.52)

This, of course, assumes the existence of such a game system to shape and interact with; a system functioning as a social contract between the participants, resolving conflict in a fair manner. Normally this means some manner of fortune: for instance, rolling dice. (Bowman, 2010, p.107)

Bowman describes role-playing games as having three major functions, the building of group cohesion, learning problem solving through scenario enactment and the creation and performing of alternate identities (2010, p.180). Group cohesion is created through the ritual aspect of role-playing (2010, pp.48-49), problem solving is an inherent feature in many genres of role-playing games, although the type of problem might vary (2010, pp.104-105), and alternate identities are created since the game revolves around the concept of creating and playing a character in the game world.

3.3. Differences between Role-playing Games

Tresca writes about the evolution of role-playing games and mentions a division proposed by Ron Edwards, separating role-playing games based on the experience they tend to provide. The proposed experiences are gamist, narrativist and simulationist. Gamist players "*seek victory and loss conditions*" and play the game like any other game. Narrativists play the game in order to tell a good story, while simulationists try to "*accurately and realistically reflect the imagined world"*. Tresca states that Dungeons and Dragons, the first role-playing game, was largely gamist (Tresca 2011, p.67), although Edwards himself calls Dungeons and Dragons gamist and simulationist.

Bowman suggests that role-playing games are separated in that they present different types of problems for the players to solve. Some may focus on tactical combat scenarios (such as Dungeons and Dragons), while others focus on resolving social or ethical dilemmas (such as Vampire: The Masquerade) (Bowman, 2010, pp.104-105).

Different role-playing games provide different game systems and different published material on settings where referees can place their stories. Edwards points out that the most important issue of any role-playing system is how it resolves the events that happen in the game. He proposes the division of different types of resolution methods into drama, karma and fortune. Drama resolves events without reference to any listed attributes or elements. Karma does reference listed attributes or elements, but without adding any randomness. Fortune uses randomness, but usually delimited by some kind of quantitative attribute or score. He also postulates that there is no direct correlation with the type of event resolution used and the type of roleplaying game experience that arises. (Edwards, 2001)

According to Edwards, role-playing games can roughly be separated into categories based on their historical lineage. Some role-playing games are based on the design philosophy of Dungeons and Dragons, and are thus gamist and simulationist. A prime example of this would be newer versions of Dungeons and Dragons, as well as Pathfinder Roleplaying Game. Other games developed more in a simulationist direction, such as RuneQuest, Champions, Basic Roleplaying System and GURPS. More narrativist games did not appear until the mid nineties, as a reaction to older designs. An example of such a game would be Vampire: the Masquerade. (Edwards, 2001)

Another important part of a role-playing game system is how it handles character creation, which is usually done by the players themselves with the help and consultation of the referee. Using characters created by the authors of a role-playing scenario or by the referee is also a possibility. Character creation could essentially be seen as a specific type of event resolution, as it can be dealt with in the same way. Generally, there is some type of limited resource (points) that is spent to make a character good at certain things. There might also be a varying degree of randomness as well. Older versions of Dungeons and Dragons used a lot of randomness in character creation, while in newer editions, the degree of randomness can be chosen, as multiple alternatives are suggested.

3.3.1. Examples

Pathfinder is a fantasy-themed role-playing game that builds upon the rules and theme of Dungeons and Dragons. It focuses heavily on tactical combat scenarios and character creation supports different mechanical builds that allow for different choices in such scenarios. Rules are comprehensive and cover a lot of areas in the game. Character creation can be either fortune-based or karma-based, and it generally tends towards being non-drama based. It is mostly gamist and somewhat simulationist.

Call of Cthulhu is a horror-themed role-playing game that builds upon the books of H. P. Lovecraft. It uses a light-weight generic rule system called the basic role-playing system, which is very simple. The game focuses on investigation scenarios, where players gather clues and try to piece the clues together in order to understand what they are supposed to do to stop whatever evil is afoot. Combat is mechanically covered, and it is present in the game, although it is not the main focus. Character generation is both fortune- and karma-based (but mostly fortune), although drama-based elements do exist.

3.4. Frames

Fine, MacKay and Cover all acknowledge the existence of a multitude of frames of experience in tabletop role-playing games. What constitutes a frame was defined by Goffman as "*a situational definition constructed in accord with organizing principles that govern both the events themselves and participant's experiences in these events.*"; thus, a framework that we can use to interpret and ascribe meaning to events. In tabletop role-playing games, Fine identifies three different frames: the primary framework - that is "*the commonsense understandings that people*

have in the real world", the game rules framework - which consists of the game rules and conventions, and the game world framework - based on the alternate possible reality in which the narrative takes place. (Fine, 1983, p.186) These three levels are further differentiated by both MacKay and Cover.

Cover distinguishes a sub frame of the primary framework, although she calls this the social frame in which planning of what to do in the game takes place. She thus separates this from off-record speech and irrelevant speech, as well as discussions concerning how the game world relates to the actual world. (2010, pp.138-140) This distinction makes clear that in-game decisions are not only taken in the game world frame as a character, but often also in the primary framework, taking in account the social structure of the players, rather than the characters.

She also separates the game rules framework, which she calls the game frame, into dice rolls and narrative suggestions. Narrative suggestions are actions of the characters that can be expressed in game terms. (2010, p.144) These suggestions may lead to dice rolls, in the case when the outcome needs to be determined objectively.

Cover also separates the game world framework, by her called the narrative frame, into narrative speech and in-character speech. Narrative speech consists of the referee's descriptions of what has happened and what happens in the game world, as well as narrative descriptions from players concerning what their characters do (common) and what they have done (less common). (2010, pp.147-149)

MacKay also separates the game world framework into different sub-frames, although his division is somewhat criticized by Cover for being too specific to drama and speech theory (Cover, 2010, p.89). MacKay observes three frames within the game world frame: the narrative, constative and performative frame, where he separates storytelling speech as being part of the narrative frame, speech addressing players in second person while providing description as belonging to the constative frame, and lastly speech which constitutes what a character does in the game world as part of the performative frame. (MacKay 2001, pp.55-56) Cover believes that all of these different types of speech work to form the narrative, and as such the narrative frame. She states that her definition of narrative speech contains MacKay's constative speech, and her definition of in-game speech contains his performative speech (2010, p.89). It seems as if MacKay's differentiations might be useful, even though they might not actually serve to distinguish different frames.

Fine also discusses frame-shifting, which is said to occur often and rapidly in tabletop role-playing games, mainly because the nature of frames is voluntary, and the consequences for breaking frame are few, unless it affects the continuation of the game too much. (1983, pp.196-197) He also stresses that problems may occur if such a frame shift is not understood by the audience, leading to a miskeying of what is said. (1983, p.201) This can often be avoided by the use of markers which denote a certain frame.

An interesting activity that is not mentioned is the intentional masking of game terms with narrative speech, where certain ways of expressing actions are keyed to actions formalized in the rule system (Dungeon Master's Guide 2000, p.71). This might lead to miskeying of what a participant says, as something that could easily be interpreted in the game world framework is actually meant to be interpreted in both the game world and the game rules framework.

3.4.1. Semiotics

Loponen and Montola discuss interpretation in the context of role-playing games from a semiotic perspective, describing three different methods of how meaning is ascribed to symbols:

imaginisation, diagrammatisation and allegorisation. Imaginisation is the process of ascribing meaning through the use of imagination, which is used mainly for in-character discussion and descriptions from the referee. Diagrammatisation ascribes meaning through the use of abstractive, systematic and logical attributes, which among other things is used for the rules of the game, and is thus very common during combat. Allegorisation is ascribing meaning through the use of allegories and analogies, trying to read symbols as metaphors. Background music and non-diegetic lighting is mentioned as commonly used allegoric symbols. (2004, pp.47-49)

They further note that this distinction is important because it clarifies what is important to communicate during a game. During diagrammatisation, it is important to be able to convey factual information, while during imaginisation, mood and feelings take precedence. (2004, pp.47-49)

3.5. Negotiation in Role-playing Games

Fine discusses the activity of negotiation in a role-playing game, and so do Loponen and Montola. Fine simply states that the shared fantasy, which is his name for the collaborative story, is negotiated through a pragmatic lens, and he focuses on the social reasons why a referee has to listen to players, while Loponen and Montola discuss this type of negotiation through the use of semiotics. They call these collaborative stories imaginary frameworks or diegeses (Loponen and Montola, 2004, p.39), which is also similar to what Cover refers to as alternative possible worlds. Diegesis is originally a term used to differentiate narrative description from direct imitation, i.e. mimesis. This distinction can be traced back as early as to Plato and Aristotle. (Elam, 1980, pp.110-111)

Loponen and Montola note that the diegeses of different participants are by definition different, and that the best possible situation is that the diegeses of all participants are equifinal, that is - they are similar enough to produce the same consequences. They also note that negotiation occurs when diegeses are found not to be equifinal, and calls this type of negotiation arbitration, where the diegeses are gradually changed until they appear to be equifinal. (2004, pp.40-41)

Fine points out that these arbitrations tend to occur since the referees do not describe everything in detail, thus creating confusion as to what is actually happening. This may result in a player stating that his character does something that he finds appropriate as far as he understands the scene, while the referee understands the situation differently. (1983, p.107) He also points out that it is seen as a cardinal metarule that rules are guidelines (1983, p.115), which makes the possibility for negotiation quite considerable. A too arbitrary approach to rules and the game system might lead to a lack of immersion though, since they establish a reality which is coherent and thus more plausible (Bowman, 2010, p.105).

Fine also analyzes what approaches participants have concerning leadership in a tabletop role-playing game, describing four different approaches: single leader, task specialization, consensus and anarchy (1983, p.172). These approaches have implications for the communication during a game and for the possibility of narrative agency of the individual players. It should be noted that the notion of leadership could exist on different levels. A character of a player could be a leader based on the part it plays in the narrative, while a player might be a leader based on the social structure of the playing group. Thus, negotiation concerning a course of action for the characters could be done either as characters or as players. When discussing game structure, Fine focuses on player leadership, rather than character leadership.

A single leader may be selected democratically or emerge naturally based on competence or the social order among players (Fine, 1983, p.172), though such leadership rests on its legitimacy in the eyes of the other players. A single leadership can be efficient, since the players decide on what they should do quicker, although it might diminish the involvement of the others (Fine,

1983, p.173), depriving them of narrative agency. Good leadership balances task-orientation with the needs of the individual players (Fine, 1983, p.173), allowing for quick division of labor and decision making when needed, while still allowing other players to influence the narrative.

Task-specialization essentially means that the skills and capabilities of the different players are transparent in such a way so that an understanding is naturally formed on the issue of labor division. (Fine, 1983, pp.174-175) This would only occur in occasions when a specific task is at hand that can be divided based on skill and capability, thus ruling out a majority of narrative decisions, leaving them open for player involvement.

Fine describes consensus as the optimal structure, if it can easily be achieved through informal social control among the players. Anarchy, on the other hand, being the total lack of structure, is less optimal for most types of games. (1983, p.175)

3.6. Rituals and Immersion

Bowman describes rituals as involving "*a separation from daily life, an immersion into a liminal reality, and a reintegration back into the social group after the ritual experiences transpire*" (2010, p.74), and points out how this type of ritual is a central part of forming group cohesion. (2010, p.48) This means that a ritual has three phases: the pre-liminal, liminal and post-liminal. The pre-liminal phase consists of the separation from the previous world, the liminal of rites that contribute to an experience of transition to an alternate reality, and the post-liminal of the return to the previous world, although reincorporated by the experience. (2010, p.48) She suggests that role-playing can be understood as such a ritual, since it contains the same elements, where the transition to the alternate reality could be the transition to the diegesis. (2010, p.49)

The diegesis is described by Bowman as "*the sum of all that is true in the reality of the game*" (2010, p.50) though, which is a bit unclear from a frame analysis perspective. Is it enough to be engaged in the game rules frame, or must there be engagement in the game world frame? Loponen and Montola define diegeses as imaginary frameworks, which is probably not true for the game rules framework, which might mean that what Bowman actually means is a transition, as a group, to a frame other than the primary framework. Irrespective of what framework must be inhabited for the ritual to take place, it is needed that the players are immersed in the activity though.

Cover analyzes immersion in role-playing games by differentiating spatial, temporal and emotional immersion, stating that they all work together to give a narrative experience (2010, p.106), although they all rest on the presence of social immersion (2010, p.116). She also defines immersion as a high degree of interest (2010, p.108), and points out that the degree of immersion of these different types varies depending on gaming style and role-playing game (2010, p.122).

Spatial immersion is the immersion in the game world space in which the narrative takes place. Cover mentions battle maps, miniatures and graph paper as tools for showing spatial relationships, but she also recognize that they do not in themselves serve to visually immerse the players. (2010, p.109) In fact, spatial immersion rests a lot on descriptions made by the referee of what is perceived by the characters, or what they know about the setting. The tools mentioned provide aids for diagrammatic interpretation, while referee description is mostly interpreted through imaginisation. Therefore, it comes as no surprise that these tools are, according to Cover, mostly used during battle sequences, while puzzle-solving and information gathering sequences could do without them (2010, p.109).

Temporal immersion is described as the feeling of suspense of what will happen next, and the suspense is greatest when there are multiple possible outcomes that are reasonably computable. (Cover, 2010, p.110) It does not have to be directly linked to what happens in the

narrative, as a lot of the suspense in a role-playing game may be derived from its actual gameplay elements. Dice rolls that greatly affect what happens to a character create great moments of suspense, and thus temporal immersion. (Cover, 2010, p.111) Temporal immersion is also created when event causality is clear and the players feel that their actions directly influence which events occur in the narrative. An interesting feature of role-playing games is the possibility to alter the time scale. Time in the narrative could pass at any rate, either faster or slower than the actual time in the real world, (Bowman, 2010, pp.111-112), which is interesting to note when it comes to the creation of suspense and clear causality.

Emotional immersion is the engrossment in one's character, and the degree to which a player has embraced his role, and is described as the most intense type of immersion (Cover, 2010, p.112). Cover argues that emotional immersion becomes greater when players create their characters themselves, and get to improve and develop them over time, since the individual investment in the character then becomes clearer, and the group knows each other's characters better, thus immersing them as a group as well. (2010, p.113) The sense of control a player has also increases the degree to which he is emotionally immersed. (2010, p.114) Cover states that one of the times when the control a player has is at its greatest is during character creation, although the exact degree of control varies between role-playing games. She also mentions outfitting characters as something that adds to emotional immersion, as it connects the player to the character.

MacKay also stresses the importance of the referee addressing the players as their character as it fixates them in their roles (2001, p.86). This of course relates to their emotional immersion, since being fixated in one's role means engrossment in character.

Bowman discusses the relation between a character and its player by describing four different stages of the evolution of a character: genesis, development, interaction and realization, (2010, pp.156-157) where the two first stages occur before any actual role-playing sessions, and constitutes what most role-players would call the creation of a character. The third step involves all the interaction between the character and the game world, and given enough interaction, the character might become realized. When a character is realized, the player has a strong understanding of the character as a distinct entity. (2010, pp.156-157)

According to Cover, none of these three types of immersion are possible unless there is social immersion, that is: engrossment in the social interaction. (2010, p.116) She discusses distraction, such as phone calls, as detrimental to this type of immersion, and thus also the other types of immersion as well (2010, p.116).

3.6.1 Dice

Cover noted the temporal immersion dice rolling can provide (Cover, 2010, p.111), which could easily lead to the conclusion that dice rolling should be transferred to the digital medium. Fine discusses dice and the relation players have with their dice in great detail however, pointing out that players are reluctant to use dice that are not their own (1984, p.93) and that there is attachment to dice based on how they look (1984, p.95). He also describes a superstition among players, where some dice may be seen as lucky or unlucky (1984, p.93). The act of rolling dice also seems to give players a feeling of control which does not transfer if a player uses a digital random number generator instead (1984, p.98).

4. Method

As our project focused a lot on the experience of a participant of the tabletop role-playing activity, a literature survey that delved deep into description and analysis of this activity was seen as needed. We also decided that we needed to look at previous work in the form of applications for other devices, as well as other applications for the Ipad, in order to understand what functionality that might be useful, and which pitfalls that should be avoided.

It was agreed that we were to perform design iteratively (Somerville 2004, p.71) rather than using a waterfall model approach (pp.66-68), starting with a prototyping process for the different aspects of our application, followed by coding, user testing and analysis. This choice was mainly because of the exploratory nature of the application that should be developed. It was not exactly clear what features would be useful in the end, and thus, a more dynamic approach seemed more useful. Different parts of the application would be developed in parallel, although we were to focus specifically on one type of functionality at a time, thus developing the application incrementally (pp.71-73), so that there would always be something relevant to test during an iteration cycle. After each cycle we was to analyze the data we received from the user test, in order to decide upon what part of the application that would get prioritized until the next user test.

4.1. Initial Time Plan

Week 8-9:	Literature survey, learning Objective C.
Week 10:	Deciding on what functionality that should be prototyped.
Week 11-12:	Prototyping of potential functionality to see if it fulfills its intended use.
Week 13:	Code structure and architecture.
Week 14-15:	Coding.
Week 16:	User testing and evaluation.
Week 17-18:	Coding and redesign.
Week 19:	User testing and evaluation.
Week 20-21:	Coding and redesign.
Week 22:	Preparing for final user tests.
Week 23:	Final evaluation and user tests.
Week 24:	Main focus on final evaluation and finishing up the report.
Week 25-26:	Finishing up the report.
Week 27:	Everything should be done.

5. Conceptualization

This section describes the work done in deciding what functionality the application should focus on.

5.1. Design Goals

The main goal of the project was to develop and implement an Ipad application which should aid a physically absent player in participating during a tabletop role-playing session. This application should at some point be possible to submit to the Apple App Store. The application should be useful for an as broad a category of role-playing groups as possible, and although it aims to solve the problem with a physically absent player participating during a session, it should not feel like something that is only used during those situations. It should not be a hindrance if used when all players are attending the session, but rather feel as a natural part of the role-playing activity. As such, it had to be designed to be easy to learn and use while providing tools that are useful during normal circumstances as well, as it otherwise might be seen as a distraction by the group. As distractions are detrimental to the social immersion on which the role-playing activity as a whole rests, the application should be part of the role-playing activity, not only a solution to a problem.

5.2. Literature Study

In order to better understand the role-playing activity a literature study was performed. Luckily, we were guided in our efforts by a PhD student that was researching the field, and thus it was quite simple to find relevant literature, as we simply borrowed most of it from his bookshelf. The purpose of the study was to distinguish the activity that was supposed to be transferred to the digital medium, by finding out what the integral parts of the activity are. The main focus was on the perceived role-playing experience from a player perspective, as opposed to that of a referee perspective.

From the study, as presented in section 3., it was possible to draw some conclusions as to what the important parts of a role-playing game are. From section 3.1. and 3.2. we deducted that a role-playing game is collaborative to its nature and contains a division of its participants into players and referee. It can rely more or less on rules, but there are generally at least some rules. Whatever rules a role-playing game might have, some of them are not always enforced, as an important part of the game is that the participants are able to negotiate, affect and control the game system itself. This gives the players narrative agency, and is generally done through negotiation with the referee. This means that a transfer to a different medium cannot limit the possibility to bend or change rules, and can thus not be programmed to rigidly enforce rules. It might also be a good idea to aid in negotiation, as the possibility for effective negotiation will probably be hampered for the spatially separated player. We also noted that the creation and realization of characters inhabiting the narrative are a core part of the player experience.

Dice was described as a common way to introduce randomness, which is often used as a way to resolve events in the game. In section 3.6.1 we found arguments that points towards that any digital transfer of dice rolling should probably replicate the act of rolling dice rather than just transferring the generation of a random number. It should also allow for a digital representation of a die which can be reused over many sessions, allowing for difference in size, color and player ownership.

The concept of frames, as described in section 3.4., was also important, as it helped us understand what requirements one has to put on an alternative medium of communication if it is to be useful in this context. Speech in role-playing games occurs on many levels, and if transferred directly to a different medium, it might not be obvious what frame should be used for

interpretation, as intonation, physical gestures, and context might be lost. We also found that the different methods of ascribing meaning to symbols presented in section 3.4.1. fit quite well into the three different frames, where allegorisation is mostly used in the primary framework, diagrammatisation the the game rules framework and imaginisation in the game world framework, even though exceptions obviously exist. This was also an important realization, as it means that the activity of role-playing may be very different in each of these frames. Transfer to a digital device is therefore also different depending on frame. Diagrammatisation requires the possibility to convey facts, while transferring imaginisation requires the possibility to convey mood and feelings.

From section 3.5, we found that depending on the structure of a particular group of players, inter-player negotiation might be more or less important, while negotiation with the referee is always important, as it is the core of the game. Clearness of descriptions might lessen the need for negotiation though, especially if used in situations where details have dire consequences for characters, as the players would then be more likely to argue for their diegesis. This type of negotiation naturally occurs in the planning sub-frame of the primary framework, which was seen as important to know if any aids were to be created to help participants negotiate. The notion of arbitration and the need to point out non-equifinal states in the diegeses was also seen as an important addition to our knowledge of the subject.

From what is said in section 3.6, we argued to ourselves that spatial, temporal, and emotional immersion depends on the amount and quality of interaction within the game rules framework and the game world framework, while the social immersion depends on the amount and quality of interaction within the primary framework. The interaction would have to be within the group that is playing the game though, or it would otherwise only serve as a distraction. This was also seen to fit quite well with the presented definition of a ritual, since without the interaction in the primary framework, the transition to the liminal state would not be possible, and the reintegration in the post-liminal state would lack meaning. The definitions of different types of immersion was also seen as useful concepts when it comes to test analysis and discussion concerning what constitutes participation in a role-playing session and how immersion could be measured.

5.3. Idea Generation

Idea generation for finding solutions to these problems was executed in four stages. The first stage was brainstorming sessions, followed by a second stage where competitor analysis was performed. The third stage was analysis of what has been found, and narrowing down what features could be interesting, and prioritization of the individual features. The last stage was idea generation as a part of the actual development of the application, as this was done iteratively. Design and implementation was followed by testing by a role-playing group that got to give suggestions and ideas, which were considered and either incorporated in the design, or discarded. This stage is described in the section on Development.

5.4. Stage One: Brainstorming Sessions

The brainstorming sessions were simple and consisted of the continual creation of a feature list containing tools which might be generally useful or that may aid the physically absent player in participating in the activity. The following tools or functions were suggested:

- Activity indicators telling if a player is there or not.
- Activity log that stores all that has happened during the session, possibly utilizing a Wiki solution.
- Application mood lighting.

- Attention polling from referee or other players. Some type of objection alert was suggested, so that a player can point out non-equifinal states in the diegeses. A referee alert was also suggested, so that the referee can get the attention of all players. The possibility of a voting system was also discussed, so that players could be aided in determining the cause of action, thus furthering the possibility of making consensus decisions, as Fine suggested that consensus decisions were preferable (1983, p.175).
- Counting utilities, such as a calculator or a timer with an alarm. Value trackers for initiative order, hit points, and spell durations were also suggested.
- Digital character sheets. A possible solution would be editable PDF-documents.
- Digital dice rolling. Possible features of such dice rolling would be a high score list for each die, loaded dice for referees, personalized dice that can be saved between sessions with statistics for each die. Dice should also be visual entities identifiable by color and size.
- Digital non-player character statistics.
- Digital whiteboard for notes and sketches or simple battlefield drawings. The possibility to add speech bubbles with text notes in them was suggested as well.
- Emotes to convey bodily or facial expressions.
- Integrated PDF reader.
- Integrated web browser.
- Loot-division aid, allowing the absent player a fair chance to get some of the things the players find. It is customary for many role-playing games that the players find interesting or useful items during a role-playing session and since there might sometimes be conflicts as to what character gets what item, this might be a useful feature.
- Negotiation tokens representing some sort of bribe to the referee, adding flavor to situations where the diegeses are arbitrated.
- Public and/or private text chats between Ipads. Chat could be color-coded based on which frame one speaks in: primary framework, game rules framework or game world framework. It would also be logical to have different chat aliases based on the frame spoken in, such as: Non-player character, Referee, Out of game, and In game.
- Sharing of played audio.
- User rights management, allowing referees to stop players from using certain features.
- Video streaming from Ipad to Ipad.
- Visual representation of a map. Such a representation could have a map construction tool as well as a combat grid net. It could have area templates for combat effects, fog of war around player tokens as well as line of sight calculation tools. Some type of movable player tokens should probably be present. A video stream was suggested as a possibility for the token picture.
- Voice over IP from Ipad to Ipad.

5.5. Stage Two: Competitor Analysis

The competitor analysis consisted of browsing online for products with features similar to that of the suggested feature list, and products that aim to in some way digitalize the role-playing game

activity. The latter type of product is generally called virtual tabletop applications, and two matrices showing the individual features of some of these products can be found in Appendix I and II, one for Ipad applications, one for PC applications (see section 3.1. for descriptions of each application).

As can be seen in these matrices, there exist several virtual tabletop applications that all implement some parts of the feature list, yet none of them are completely sufficient for our purpose. The PC applications are more complex and rich on features, and can be used in a networked fashion, while the Ipad applications are simpler, but do not allow for network use.

5.5.1. Virtual Tabletop Applications for Ipad

The main applications found for the Ipad were Cartographer, Dungeon Mapp and Battle Map. By comparing these three one finds that they are all aimed to be more limited than their PC counterparts, yet still carry quite a lot of functionality. In the case of Cartographer, it might actually have too many features, as the interface in our opinion feels very cluttered and counter-intuitive as to what tool does what. Dungeon Mapp and Battle Map, on the other hand, keep their interfaces clean, and their tools seem much more intuitive. This is something that should be considered when designing an Ipad application. When wanting to keep the interface clean and the functionality useful, too many features available to the user at the same time might be a problem as there is such limited space to use.

Even more importantly, these three applications lack network support, which was one of our main concerns, as all of the suggested features assume network communication.

Only one of the three Ipad applications adds features which are not directly connected to the creation and display of maps: Dungeon Mapp. This application has tools which keep track of the individual health, damage, movement rate etc. of the movable tokens which are placed on the map. It also has aids for keeping track of initiative order during combat.

5.5.2. Virtual Tabletop Applications for PC

When it comes to PCs there exist a large amount of different virtual tabletop applications with somewhat different functionality. Since many of them share the main bulk of their features, there was no need to analyze all of them in detail though.

Maptools' main selling point is that it is free, and that it provides great support for external scripts and libraries, while Fantasy Grounds 2 has a well-made interface with mood-creating components such as beautiful menus, character sheets, three-dimensional dice simulations, handouts, lighting and much more. D20Pro is mainly prioritizing the gaming experience by incorporating system specific stats and calculations into the program that lessen the amount of work for referee and players. For this to work they have also released specific packs with updates for different role-playing systems.

All of these applications have network support allowing for distant play. In comparison with the applications for the Ipad platform the features of these applications are much more exhaustive and more suited towards using a prepared map during the session than drawing on the fly. Fog of war is a perfect example of such a feature, as it exists in two of these applications, and, combined with an elaborate line of sight feature, forces the map to be prepared in order for it to function properly, as it requires emplacement of light sources and walls that block line of sight.

All three applications also support text chat that makes a difference between in character and out of character talk, allowing for greater clarity as to the frame in which one speaks. Fantasy

Grounds 2 has broader options that allows for more different types of speech, such as individual NPC speech as well as storytelling speech.

5.5.3. Non-Virtual Tabletop Applications

When it comes to applications with similar features as those listed in the feature list, there were plenty to be found; although the most interesting question was if they exist for the Ipad, and if they could be used in conjunction with our application.

A Skype (Skype.com, 2003) application exists for the Ipad, which allows for voice over IP, and (provided the use of an Ipad 2) also video communication. Video communication can be a problem though, as one would like to have all non-distant participants on one end, and one distant participant on the other end. This might be problematic with Skype if assuming that everyone uses an individual device, as you can only see one other person at a time. It would be possible that the group uses Skype combined with a laptop or stationary computer and a webcam instead, while the distant participant uses the Ipad. Skype does indeed provide reasonable implementation for these two features though, and it can be used in conjunction with another Ipad application.

There is also a Dropbox (Dropbox.com, 2012) application for the Ipad which allows for distribution of pictures, character sheets and handouts in general, as well as multiple PDF readers that can be used in conjunction with our application.

5.6. Stage Three: Analysis and Prioritization

Because of the small screen size of the Ipad, making it prone to clutter if too many functions are fit into the same application, it seemed reasonable that the application should focus on some primary functionality rather than becoming too broad in scope. Thus, some things would have to be prioritized over others. It was therefore decided that things like counting utilities (as found in the Dungeon Mapp application) and loot division aids, although useful, might not be necessary enough for transferring the activity. Voice over IP, sharing of handouts, character sheets, PDF reader and video communication, although nice to have in an integrated solution, would have to fall outside the scope of this project, as other solutions exist that can handle this. Integrated web browser also seemed quite unnecessary, as it is quite easy to switch between applications on the Ipad.

There are also other features which can quite easily be solved without transferring the activity to the digital medium. Dice rolling, can for instance be done manually, and the results simply reported through voice communication. A complete digitalization might have been preferable, but it was not seen as necessary enough for it to be a priority, especially since a good transfer would have to be quite graphical (see section 4.6.1. for reasoning on dice).

Another decision concerning the focus of the application was whether it should be usable on the fly, or if it should require a certain amount of preparation from referee or players. Because of the improvisational nature of a role-playing game, and the fact that the referee is supposed to interpret non-standard actions from the players (as described in section 4.3.), on the fly use seemed like the primary way in which it would be used. Preparing content for a session, such as multiple battle maps and individual player tokens might be a great aid, but being able to do things on the fly is crucial. This gave features like fog of war and line of sight a low priority.

Text chat is a feature that would be very good to have, but which could at least partially be replaced by the use of voice over IP. If one would assume text chat as the only means of communication, it would also be detrimental to the social immersion, and tabletop role-play as a face-to-face activity would not transfer (see Cover's definition in section 4.3). Implementing it could serve as a means for communicating directly with one other player or to communicate

without disrupting the communication of other participants, but the lack of a proper keyboard on the Ipad still made this less tempting then it could be. The Ipad does have a digital keyboard, which uses the touch screen, but when it is used, it obscures half of the screen, and writing on it lacks the tactility of a physical keyboard. Therefore, it seemed as if using this keyboard should be avoided as much as possible.

5.6.1. Immersion

It is possible to roughly categorize most of the remaining features into categories based on what type of immersion they aim to aid: social, spatial, temporal or emotional (see section 4.6.). The features which do not seem to affect immersion can be seen as utility functions.

- Emotional, such as mood lighting and shared player audio. These are functions that transfer symbols that are generally interpreted through allegorisation.
- Social, such as activity indicators, attention polling, emotes, negotiation token aids, and text chat.
- Spatial, such as visual map representation and whiteboard. Although symbols used for creation of spatial immersion are generally to be interpreted through imaginisation, clarity in spatial relationships may minimize the need for arbitration of diegeses. Mood lighting and shared player audio could also add to spatial immersion, if they are interpreted as being part of the diegesis; i.e. playing medieval market music in a game where the characters are currently in a medieval market setting. This is probably not the most common case though.
- Temporal, such as dice rolling, visual map representation, and whiteboard. In tactical combat scenarios, visual map representation and whiteboard may help in discerning the causality of events.
- Utility, such as activity log, counting utilities, digital non-player character statistics, loot division aid, and user rights management.

Since the role-playing activity is in its nature immersive, and the goal was to recreate this immersion as much as possible, features that help doing this were deemed much more important than utility features. Since emotional immersion is mostly dependent on player agency with respect to the played character (Cover p.113), mood lighting and shared played audio was assumed to give fairly small additions to emotional immersion. Therefore the important features would have to be the socially, spatially and temporally immersive ones.

5.6.2. Use Cases

Since one of our design goals was to make the application seem natural to use even in the case when no player was spatially separated from the others, we decided that it would be necessary to also look at use cases other than our main use case.

It was possible to discern some different possible use cases for the application:

- The referee is preparing a session.
- The referee and a group are playing a session using one Ipad, all in one location.
- The referee and a group are playing a session using multiple Ipads, all in one location.
- The referee and a group are playing a session using one Ipad, while one player is in a different location, using one Ipad.

- The referee and a group are playing a session using multiple Ipads, while one player is in a different location, using one Ipad.
- The referee and a group are playing a session using multiple Ipads, while more than one player is in a different location, each using an Ipad.
- The referee and a group are all in different locations, using one Ipad each.

The main use case targeted by the application was the one where multiple Ipads are used, while one player is in a different location. This did not, however, mean that all other scenarios could be ignored. Since one of the design goals was for the application to feel useful during scenarios other than the main use case, the most probable cases were also to be considered very important. Playing without any distant players, while having one or multiple Ipads seemed to be very probable use cases, and thus the following prioritization was decided upon:

- Main priority: The referee and a group are playing a session using multiple Ipads, while one player is in a different location, using one Ipad.
- Secondary priorities: The referee and a group are playing a session using one Ipad, all in one location; the referee and a group are playing a session using multiple Ipads, all in one location; the referee is preparing a session.

Multiple distant players were not considered, since the case where only one player is distant seemed to be the scenario that would be most detrimental to the immersion of that player; all others can fully interact with each other, while the distant player cannot.

5.6.3. Features on a Use Case Basis

The different features suggested would be of different use dependent on the use case. Analysis that distributes the remaining features over the use cases is provided below:

- Referee is preparing a session. This might make use of a visual representation of a map, where maps from adventure modules could be added to the application. Loading images into the application would then be needed. If a grid is used, a snap to grid function as well as tiling of the image would also be needed. Actual manual drawing could also be useful, which would mean a need for effective drawing tools; for instance, circles, ellipses and lines in different sizes and color. It might also be possible to manually draw textures (floor tiles).
- The referee and a group are playing a session using one Ipad, all in one location: All functions needed when preparing a session could be useful in this case, provided that they are efficient enough to use ad-hoc. A combat grid as well as combat tokens to represent characters and monsters might be very useful. For tokens to work well they should either be generic or easily imported. If drawing is present, multiple drawing layers could be useful; probably separating referee layers from player layers. Drawing could be freehand, and multi-touch would be preferable given that there are multiple users for one device.
- The referee and a group are playing a session using multiple Ipads, all in one location: All functions needed when playing a session on one Ipad could be useful, although multi-touch drawing might be unnecessary. Network support is needed, although a local network would suffice. Drawing layers could be used to communicate information secretly if there was a possibility to hide layers from others. Text chat could also be used to pass hidden messages to other participants, although if text chat was to be

implemented, some kind of indication of pending messages would be needed, as focus would generally be on playing the game.

- The referee and a group are playing a session using one Ipad, while one player is in a different location, using one Ipad: All functions needed when playing using multiple Ipads, all in one location would be useful. Network support would have to be non-local though, and some kind of online game service would help in connecting participants. Text chat could also make use of frame indications of messages. Emotes, negotiation token aids and attention polling might be useful for this case. Attention polling might be activity indicators, forcing other applications to display something, nudges, sound effects or a voting system.

Since virtual representation of a map is a feature prevalent in all use cases, this feature was the focus of the first iteration. Since network support is so extremely vital for most use cases, this was to be covered in the second iteration.

5.6.4. Choice of Test Group

When tests started to be performed for real, they were done on a more or less weekly basis, being less formal and involving a specific group that would be easy to get together at short notice. This allowed for more regular testing, although the amount of test participants was lower than optimal. This meant that we got regular and fast response on the features we implemented though, as well as a lot of stress testing for the application. This was a huge advantage at some times, as we found a lot of odd problems that would otherwise have been hard to identify. The group was also accustomed to development and the use of early prototypes, which made it possible for us to showcase new features in an early stage and still get relevant feedback, while also getting in depth advice on how we could try and solve some problems. By testing weekly towards one group, that group grew familiar with our application, which the benefit that the users knew the application well, and could at an instant test new features and give feedback on how these features affected the application as a whole.

5.6.5. Choice of Test Game

Since we chose to have one test group that was used a lot, it came naturally that the game played would be that which that group wanted to play. This was the role-playing game Pathfinder (see section 4.2.1 for a description), and more specifically, printed official modules for this game called Adventure Paths. These paths contain a lot of detailed tactical combat scenarios, as well as city maps, well-defined political factions, non-player characters, and an advanced storyline. We realize that this affected the goal of having a broad application, as our testing gave results based on the chosen game.

6. Development

Development of the application is covered in this section, divided more or less chronologically into five phases. Each phase consists of one iteration cycle of design, implementation, testing and analysis leading to the development of some select features. It was initially the idea that each iteration cycle would add new features from our list, but it later came to focus more on refining the features already implemented, as they were deemed inadequate.

6.1. Learning the Platform

Prior to any actual development, there was of course the need to learn the platform and programming language which was to be used. As the application was to be developed for the Ipad (Apple.com, 2012b), the choices were either Flash (Adobe.com, 2012) or the IOS platform (Apple.com, 2012a). Since it seemed to be the easiest solution, it was decided to use the IOS platform, which provides an extensive graphical framework as well as simulation software. IOS uses the programming language Objective-C (Apple.com, 2012c), which builds upon a messaging system borrowed from Smalltalk. This programming language had to be learned in order to program for the platform.

In order to program in Objective-C it was also needed to use the only IDE (Integrated Development Environment) that allows for compilation of the IOS framework: X-Code (Apple.com, 2012d). X-Code does not run under Windows or Linux, which were the only operating systems with which we had previous experience. It was therefore also needed to familiarize ourselves with Mac OS (Apple.com, 2012e).

To grasp the platform and delve into one of the areas where problems were most expected, it was decided to start implementing a simple network lobby. While researching options on how to implement network communication for the IOS platform, it was found that the standard framework lacked support for sending data asynchronously. In order to avoid slowing down the application by waiting for synchronously sent messages, we decided to look for third-party solutions. After a while, the AsyncSocket (AsyncSocket, 2012) library was found, which provides the desired functionality, supporting both UDP and TCP/IP.

Initially, UDP seemed to be a preferable choice, as it lessens the strain on the network, lowering latency between clients. However, the AsyncSocket library does not have any example code, documentation or tutorials for its UDP implementation, which led to the need for a tedious trial and error approach, which in the end yielded results.

Another problem that was identified was that of the firewall in the Chalmers network which we were on. This firewall blocked all network traffic on the ports which we tried to use. Solutions such as VPN tunneling software as well as contacting network administrators were fruitless, and thus it was decided to give up on using the open network. Therefore a small local ad-hoc network was created using a laptop, as Ipads can join ad-hoc networks but not create them. Not being able to test the application over a non-local network was of course a setback for the project.

Lastly, there was also the problem of threading, as we wanted to create a waiting loop when a client tries to connect to a server. This was problematic as all things done graphically has to be performed in the main thread, which led to the solution of having a waiting thread which ends by passing a message that is then run in the main thread of the application. Although it was not understood at the time, the AsyncSocket runs in the main thread as well, unless specifically told not to do so, which led to odd behavior if one tried to pass messages to it from another thread.

A network lobby is not only a question of setting up a network connection; there also has to be some manner of player broadcasting when sessions are created, so that others can join them.

For this we decided to use a Bonjour network (Apple.com, 2012f), which automatically solves this as long as devices are on the same local network. This might not always be enough though, so another connect feature was also added so that one can try to connect to any chosen IP address in the hopes that a session is currently hosted there. The idea was that Bonjour would be used by the group, and the connect-to-IP used by the participant not currently present. Because of this, it would also be necessary for the lobby to present the IP address used by your device, so that this can be communicated in some way to the non-present participant. This was problematic though, as the only simple way of finding out the IP address is to use code that is declared private by Apple, and thus not available if one wants to be able to add the application to App Store (Apple.com, 2012g). We therefore looked for a different solution, and found that it was possible to extract the IP address from Bonjour instead, although it was not obvious how to do so.

As a side note, it should also be mentioned that a few weeks into the project, Apple released a new major version of X-Code, adding a lot of important features, but also redesigning the user interface, and making parts of our project corrupt.

6.2. Phase One: Map Drawing

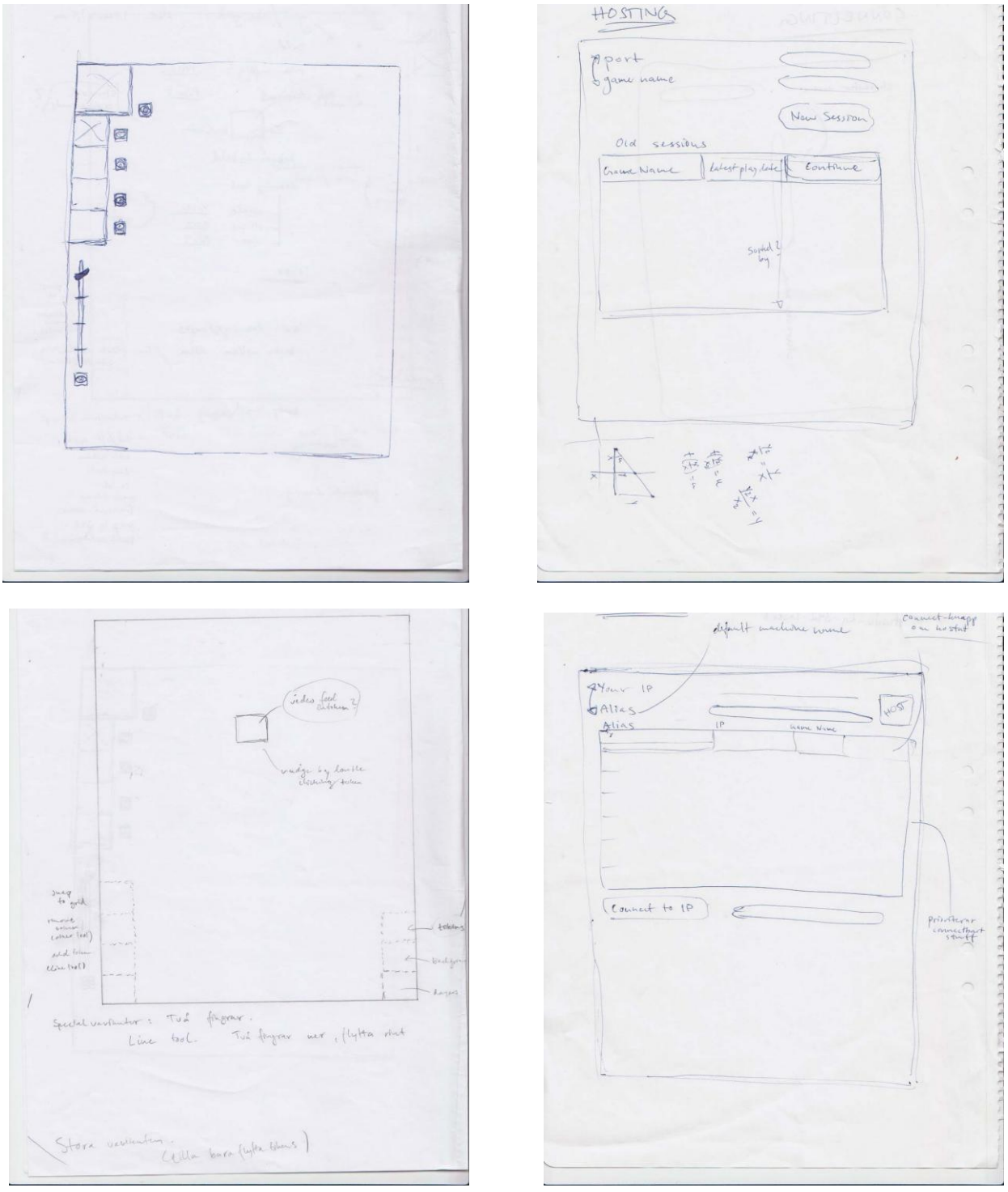
The first phase of development focused on visually representing a map. It was soon realized that the simplest way to draw a map is by having some sort of whiteboard. Therefore, map drawing started out as just that, a whiteboard.

6.2.1. Design and Implementation

As we decided to start out with the whiteboard feature, our first priority was freehand drawing and the drawing of lines, since these types of drawing were assumed to be most versatile, and thus most important.

We started out by making some paper prototypes. Our initial assumption that one would like to have multiple maps available at the same time led to an interface that focused on how one would switch between them. It also contained a simple menu displaying all drawing tools which would be possible to use: lines, circles, freehand etc. This menu was placed in the upper left along the edge of the display, since this is where that type of menu is commonly placed in most drawing software. Since functionality that was vital for testing the application was prioritized, the actual switching between different maps way was not implemented. It might have had a higher priority if development was to focus more on prepared sessions though.

After implementing drawing, it would have been reasonable to perform user tests. This was not done at the time however, as no Ipads were available and the X-Code simulation software would not suffice. Rather than halting the project it was decided to implement movable tokens instead, as this is another suggested feature that builds a visual map representation that is also present in all virtual tabletop applications analyzed.



These pictures depict the initial paper designs for drawing interface and network lobby.

6.2.1.1. Drawing Canvas

Before any drawing could be done, there would have to be something to draw on, which in IOS means a UIView. As we assumed that maps can become quite large, the canvas on which one draws has to be fairly large as well. This combined with the fact that the Ipad screen is quite

small, meant that scrolling would be needed. The first idea was to simply use a UIScrollView, which is the standard component in IOS for scrolling in a view. This did not initially work though, as the large canvas that we wanted made the UIScrollView crash, as it tried to keep everything in memory at once.

Two solutions seemed to exist to this problem: building our application on a game engine instead, such as Cocos2d (Cocos2d.com, 2010), or to use an IOS component called CATiledLayer, which automatically tiles and loads the contents of a UIView. We assumed that learning to use a game engine would take more time, and thus decided on the second solution.

CATiledLayer is a class that divides a UIScrollView into a large number of same-sized tiles that can be cached on demand, rather than letting the device load the whole UIView into memory at the same time. This, however, uses more CPU time, but lessens the strain on memory enormously. The use of a CATiledLayer gave us the possibility of having a drawing canvas with almost unlimited size.

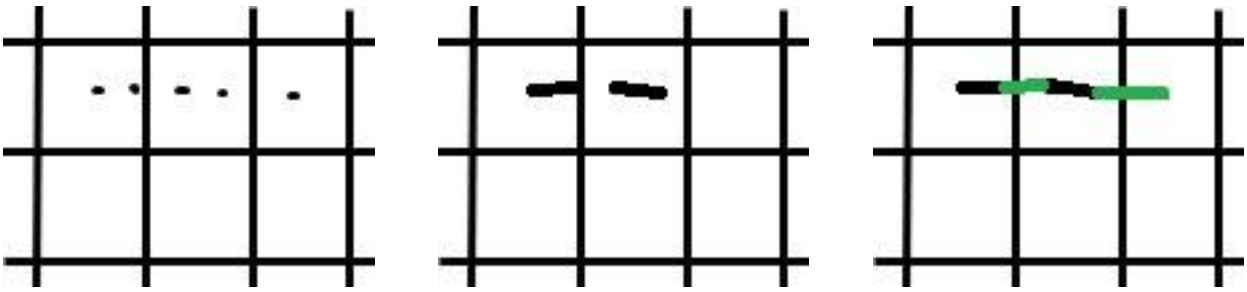
Since CATiledLayer loads the information to display for individual tiles, all information also needed to be stored based on which tile the information belonged to. As CATiledLayer only allows for redrawing of whole tiles, we chose to make the tiles fairly small, so that they could be redrawn individually when something should change on the screen. In essence, tile size was decided by testing different values and optimizing our choice based on application performance.

6.2.1.2. Freehand Draw Functionality

Freehand drawing was implemented by registering points in the touch events of the UIScrollView in order to use them to draw lines. Initially, the actual construction of the lines was placed in the messages which handle touch events, as it was seen as logical to do it there. This approach made the drawing lag a lot though, as the events took longer to finish, and thus lesser of them were allowed to be called. Because of this, the line construction was moved so that it was only performed when the system actually called for a redrawing of the display.

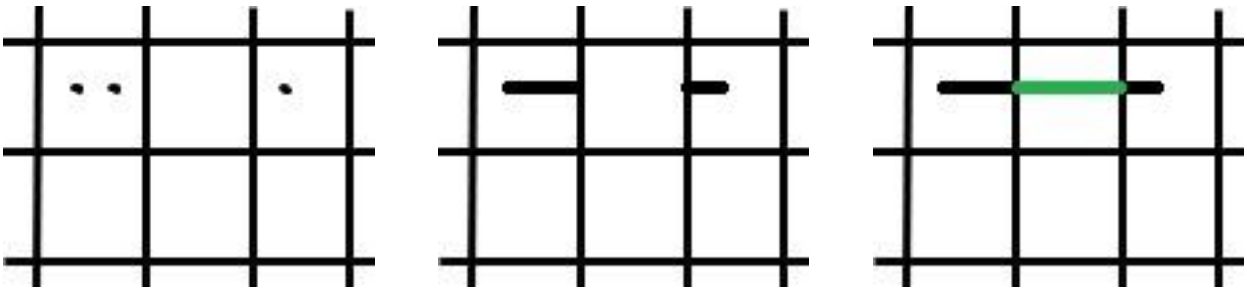
Moving the line construction code created other problems though, as manipulation of the tile data was now done in multiple threads simultaneously. This led to the need for concurrency locks on the data objects, and letting tiles copy objects that would be drawn when requested by the drawing message.

Line drawing now worked, although only partially. As drawing was done per tile by adding lines to the tiles individually as the finger was moved over them, problems occurred when the finger moved too quickly over tile borders. Moving quickly means that the Ipad registers fewer points, which meant that there would be holes around the tile borders, where the previous point of a tile was in a different tile, and thus not registered. This was fixed by always adding the last point that was outside the tile as well.



First picture depicts the registered points; the second picture shows the lines being drawn normally. The third picture adds the lines that are drawn after the fix.

This solved the problem as long as one did not move the finger quickly enough to skip a tile completely. If there were no points at all registered within a tile, there would still be a gap in the line within that tile. Thus we wrote an algorithm that, for each pair of points registered, stepped through all tiles between them and added the point pair to those tiles as well. It might sound as if this would create lines where there should not be any, but in fact it does not. This is because all points are still saved in the global coordinate system for the whole map, and not in tile specific coordinates; so any points that are actually outside a tile but still added to the tile, will not be drawn.



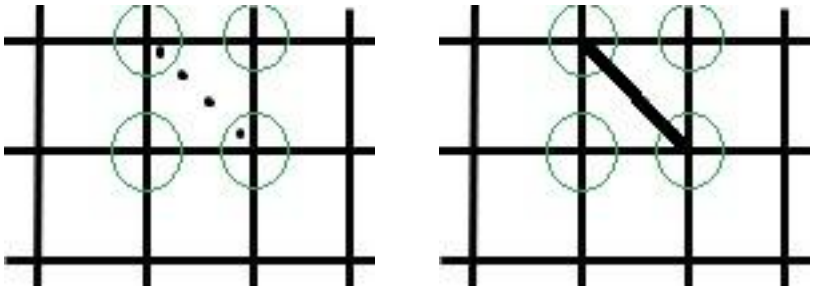
First picture depicts the registered points; the second picture shows the lines being drawn normally. The third picture adds the lines that are drawn after the fix.

Lastly, there was an issue when drawing lines parallel to a tile border while being very close to it. If the line width was larger than the distance to the border, the parts of the line that were outside the tile would not be drawn. This was solved by comparing the distance to each tile border with the line width, and if needed, add the points to the tile on the other side of the border as well.

6.2.1.3. Snap to Grid

When free hand drawing was realized, we moved on to implementing drawing straight lines. Since there was already a grid system in place, that is, the individual tiles, it was decided to implement the drawing of straight lines by snapping points to this grid net. This was pretty simple, as the only thing needed was to detect the closest tile corner of any new point registered in the touch events, and adding a line to that corner in all tiles adjacent to it.

What was actually a matter of discussion was how to implement the drawing of diagonal lines, and whether to allow for diagonal lines within one tile only, or if they should be able to stretch between multiple tiles. Since it would be complex to allow for diagonal lines over multiple tiles, it was decided not to implement it, as the gains were deemed to be low. Diagonal lines within tiles were implemented though, which meant that line snapping would have to only occur if points were very close to the tile corners rather than always snapping to the closest corner. This allowed for a dead zone while drawing, which meant that the finger would have time to cross the tile diagonally without the line snapping to a corner.



The first picture depicts snap-to-corner zones and registered points, the second picture shows the line which would be drawn.

6.2.1.4. Eraser

Erasing lines was implemented by adding an eraser button to the menu that allows for the drawing of transparent lines. It was discussed if it would be possible to erase in a snap to grid mode, but as this would introduce multiple eraser modes, it was seen as unnecessarily complicated. We wanted the menu to be as simple as possible, so the fewer modes the better.

6.2.1.5. Background Tile Images

As we already worked with a tiled canvas, it was natural that the background color or texture of the canvas would have to be tiled as well. This was simple to implement; however, it was soon realized that whenever a tile was to be redrawn, the IOS component cleared the whole tile first, which meant that whenever drawing would be done, any texture used in the background would have to be redrawn as well. This was quite slow and made the application lag.

When trying to solve this problem some odd solutions found online were tried, such as adding all new drawing in a new sublayer of the original CATiledLayer component; i.e. adding new instances of the same component type to the original instance for each draw operation. The solutions did not work well however, so instead it was decided to rework the code so that background and drawing was separated into different views altogether, with completely separated on-screen drawing. This was a successful solution that would be useful later when implementing other features.

6.2.1.6. Scrolling

Now that we had a canvas as well as background and drawing, it was also necessary to allow the user to move around the canvas as the canvas was larger than the display. The UIScrollView normally does this by moving one finger around on the screen. As this type of movement was already used for drawing, it would have to be done differently though. We came up with two solutions to this: adding some way to switch between screen moving mode and drawing mode, or change so that scrolling is done with multiple fingers only. Since we thought it would be annoying to switch between modes it was decided to scroll using two fingers.

Soon it was realized that there was no way of making the UIScrollView do this. We therefore made some futile attempts at implementing our own scrolling, but it never really got close to the smoothness of the UIScrollView. We therefore instead decided to do all that we could to force the UIScrollView to do what we wanted. This was an arduous endeavor based on a tedious trial and error approach, yet in the end a solution was found: taking one UIScrollView that passes touch events on to a UIView which then sends messages to another UIScrollView that executes our code, which determines whether scrolling should be done, or whether touch events should

be used for drawing. This combines the smooth event detection of the first UIScrollView with the actual scrolling of the second UIScrollView. The middle UIView does nothing but forward the events, as this has to be done since a UIScrollView for some reason cannot forward events to other UIScrollViews.

Another problem was the accidental drawing that occurred when putting two fingers on the screen to scroll. If the fingers were not perfectly in sync, one of them would sometimes trigger drawing events as it was in effect a one finger touch. This was remedied by backtracking when a second finger was detected, removing the line that was drawn. This only works within the confines of a tile though, but this was deemed to be sufficient.

We also spent quite some time trying to fix a problem pertaining to how touch events and standard gestures interact in IOS. When a gesture is activated, only begin events are triggered for the touches, and never end events. This combined with the fact that some standard gestures suspend the whole application creates problems with the finger counting code used in the second UIScrollView to allow for two-finger scrolling. This code increments a count when begin events are triggered, and decrements it on end events, which does not work well if end events are lost. The logical way to solve this would be for the application to clear the count whenever it returns from suspension, but the code which was supposed to run when this happens did not run. We therefore implemented a partial solution that clears the count after the screen has been touched, thus allowing the application to recover rather than becoming permanently locked in scrolling. This still means that the application might not allow drawing the first time it is touched after returning from a suspended state, but it was seen as a solution that was good enough.

6.2.1.7. Zooming

After implementing scrolling, it was realized that the standard zooming functionality of the UIScrollView did not work anymore, which was quite logical as events were intercepted before they reached the component that was supposed to do the zooming. A few solutions to this problem was tried: implementation of our own zoom, trying to detect the zoom pinch gesture before actually sending events to any of the UIScrollViews, and lastly redirecting the zoom gesture of the first UIScrollView in our chain to instead affect the second UIScrollView. The only satisfactory solution was the last one, even though it was not flawless.

This approach did not allow the event chain to separate zoom and scroll functionality, which meant that both could trigger simultaneously. This led to zooming being accompanied with a twitching, irregular scrolling. After some research into the matter we managed to solve this problem by cancelling the zoom gesture when it should not be run: i.e. after the start of a scrolling movement. There was no obvious way of doing this though, and the way that was found was to completely remove the UIGestureRecognizer that handles the zoom event, and then adding a new one to the component.

6.2.1.8. Mode-switching menu

With snap to grid, free-hand drawing and an eraser, it was necessary to in some way indicate which of these modes the application was currently in. This was done by adding a small round icon next to each button, which either was crossed out red or green with a check symbol. This was seen as a clear way of indicating selection without annoying the user with animations or obstructing the button icon.

6.2.1.9. Tokens

Tokens, i.e.: movable images representing characters on a map, were implemented more or less the same way as background tile images, using a completely new view. The minimum size of a token was set to be the same size as a layer tile, while larger tokens would be tiled and cover multiple layer tiles. This is done with an algorithm that automatically divides an image into multiple smaller images. The tiling had consequences when moving large tokens though. As tiles are drawn independently of each other, this results in tearing at the edge between the different tiles containing the token.

As IOS at the time only allowed Ipad applications to access files from its application directory, we decided only to use pre-generated tokens for the time being. This meant a quite limited selection of tokens, with some of them being generic (e.g. a circular image with a metal border and a red background).

It was then necessary to make these tokens available for placement in the application as well. What we found to be most reasonable was to implement a token placement mode. It would of course have been possible to have all tokens always available for placement, but we wanted to avoid clutter when moving tokens around. The tokens possible to place were displayed in a one-row-sized menu at the bottom of the screen. This placement was chosen in order to minimize screen obstruction and to organize the activity flow chronologically from top to bottom. That is: one chooses a mode in the upper left, and a token can then be selected for placement from the bottom menu. Since we had more tokens than could be properly displayed in a one-row selection menu, it was decided to make it a UIScrollView that only scrolled sideways. This worked well for our limited set of tokens, but it was realized that this would be insufficient if larger sets would later be used. Having the UIScrollView link its start and endpoints so that its content would rotate was considered, but it was decided that this would only add unnecessary spatial confusion to the user.

For actual placement two different versions was discussed. One was implementing drag and drop from the selection menu, the other to selecting tokens in the menu and placing by pressing somewhere on the screen. The second was deemed to involve less excise, but somewhat more confusing, as one might forget which token one has selected, especially when switching back and forth between different modes. In the end we went for less excise, as speed seemed important for ad-hoc map creation.

This selection menu was implemented as a chained double UIScrollView, just as the one used to display the whole map, yet we did not want to scroll with two fingers this time, as it would be just as good to use a timer to separate what was a selection and what was scrolling. If a user would touch an area of the menu long enough without moving the finger too much, the menu should interpret it as a selection, while scrolling should be done by default.

Given a select and place-approach, some way was needed to display which token was currently selected. As the selection menu, and thus the selected token, would not be visible all the time, it was decided to add a very clear animation to the token: growing and shrinking. This could have been annoying for the user if it would always be visible, but as it was not, the clarity of this solution was seen as preferable.

Another important issue that was discussed was whether the movement of a token should be grid-based, or free with a snap-to-grid function when dropped. The former would mean that a moving token would move between the tiles discreetly; i.e. a token at any given point would only be in one tile (unless the token itself had a size of multiple tiles: then it would cover as many tiles as its size). We thought that it would be unnecessary to remove freedom if we did not have to do so, and thus went with the second choice.

It was also questionable in what modes it would be possible to move tokens, and if tokens should be visible in all modes. It was assumed that tokens would obstruct the view while drawing, and therefore it was decided that tokens should not show when in a drawing mode. This meant that they could also only be moved in the token placement mode, as we did not want yet another mode for token movement only. The fewer modes the better, was still our design philosophy.

Lastly, deletion of a token was implemented as a double tap on the individual token. It would have been possible to add token deletion to the eraser mode, but this was seen as confusing and possibly problematic, as one might want to erase what is drawn below a token rather than the token itself.

6.2.1.10. Grid Net

As token placement and movement was now implemented, it became necessary to show the user where tiles are situated on the map. This was initially done by indicating tile borders with lines, but changed to only indicate the tile corners with cross-shapes. This way of tile indication looks much less cluttered, while still being very clear. This grid-net was created in a view of its own, separate from drawing and tokens.

6.2.1.11. Drawing Optimization

Before launching the first user tests it was seen as necessary to optimize drawing, as the application still ran a bit sluggish. This was mainly because the whole tile was redrawn from scratch each time any drawing was done. This meant reconstructing all lines drawn in the tile, which became slower for each line drawn. The solution was to store what was drawn before in a buffer, rather than redrawing everything. Further optimization was done by changing the buffer from being a UIImage to a CGLayer. Both can be drawn to a graphical context with ease, but the CGLayer is optimized for quick redrawing, which makes it way more optimized for this purpose. As soon as this was found it was also used to optimize background tile images as well as token images, as these are also redrawn often.

This created another issue with all textures becoming very blurry and loading slower. After some analysis of our code, it was found that we had constructed the CGLayers in the wrong way. We had used the same graphical context for all CGLayers, when what we should have done is to create a new context for each CGLayer.

The pictures above depict two instances of the application being used during the same session. One is currently in the freehand drawing mode, while the other is in the token placement mode.

6.2.2. Testing and Analysis

The first user test thus involved an early version of the application with rudimentary functionality. It had an upper left menu with four buttons: freehand drawing mode, snap to grid drawing mode, eraser mode, and token mode. When token mode was selected, a selection menu containing all different tokens appeared at the bottom. We wanted to test this with a group with some experience of testing early versions of products, so that there would be an understanding and acceptance for bugs and the lacking of some features. Since network support was not yet implemented, the test would have to simulate a use case that we had already deemed secondary: the referee and a group are playing a session using one Ipad, all in one location.

Our setup for this was one referee and a group of three players, all playing an official printed module for Pathfinder. This meant that there already existed maps in the module for many locations the players could go to. As Pathfinder, and especially official modules for Pathfinder, focus on tactical combat scenarios, this would of course also affect the use of the application. No map drawing was done in the application previous to the testing though, so the use was by our definition ad-hoc. We participated in the activity as referee and player, and as such the test would begin by letting the other two participants interact freely with the application without our interference. Later on, we used the application during the role-playing session as a group though, with informal discussion on its performance done during and after the session.

The first problem noticed during testing was that the application could not at all handle multiple users interacting with it at the same time. Even though the device of course supports multi-touch, the application itself does not. It was soon realized that in order to make it support multiple users, the two-finger scroll would have to be scrapped and a lot of work be put into separating touch events into different lines to be drawn; although since this use case was only secondary it seemed better to just ignore this instead.

There were also complaints about the tile edge tearing when moving the tokens. It was suggested that changing token movement to grid based movement would make this look less bad. Another suggestion was to, as long as the token was moving, show the original position (i.e. the position at the start of the current movement) of the token as well as the path it had

been moved, possibly also displaying the distance moved. This feature can be found in the Maptool application, so it was not new to us. We deemed this feature to be a utility rather than something that builds immersion in some way, so we decided against implementing it. It was also suggested that we implemented token stacking in a better way; as it was, only one token would be displayed in a tile at a time, which made it look very strange when a smaller token was on top of a large one.

It was also realized during this test that drawing and tokens could interact in two different ways. Either tokens are already in place, and drawing is performed around them, or drawing is performed first, and the tokens added later. During this particular test, the former was more prevalent, which made it very annoying not to be able to see tokens when in the drawing mode. This would have to be remedied in later versions. When this was suggested, it was also suggested to allow for movement of tokens in drawing mode, as it would be cumbersome to switch between the drawing mode and the token mode continuously.

We also discussed the possibility of using drawing to communicate with each other in a later version with network support, and whether different layers with different visibility options would be useful. We did not get any conclusive opinions on this, so we decided to wait with implementing this, while keeping it in mind so that it could be implemented easily later.

All users thought that the application would work much better if all participants had one Ipad each.

6.3. Phase Two: Network Support

The second phase focused on implementing network functionality for the features already in place. This was the main priority since it was needed in order to test our main use cases.

6.3.1. Design and Implementation

The main question when designing the network was whether an external host should be needed or if one Ipad should act as host. As, at the time, no reason why an Ipad should not be able to act as a host could be found, we decided that the implementation of a server running on a different machine would be unnecessary work. Thus, it was decided that the Ipad that initially starts a session will be the host of that session, and forward messages sent to it to all other participants.

6.3.1.1. Network Connection Setup

As we already had a network lobby implemented from our learning phase, it seemed reasonable to use this as a basis for network setup in the application. We started by revising our previous decision to use UDP, and decided to use TCP instead. Since the features that we wanted to implement network communication for did not need to send continuous information, and losing packages would be problematic to handle, this seemed as a better protocol to use. Thankfully there was an AsyncSocket version for this protocol as well.

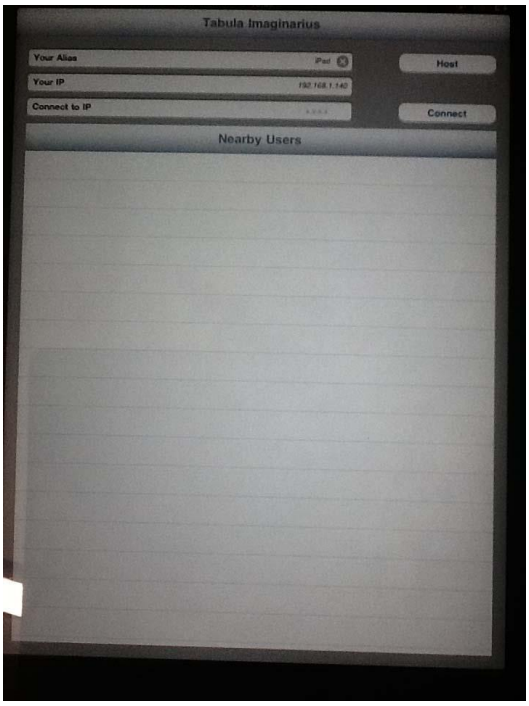
Again we had trouble getting things to work because of thread issues, as using send operations in a thread other than the main thread did not work at all. At first it was very unclear that this was a thread issue though, so it took some time to find out.

As soon as send and receive functionality was up and running we decided to implement all network handling as separate protocols for each separate functionality that existed in the application; i.e. a drawing protocol, a token protocol and a background protocol. The idea was that this would allow for some modularity of the application, making it easier to later add new functionality without conflicting with older ones.

A few interesting things were noticed when implementing this concerning Objective C and object-orientated programming. First of all, it is hard to make objects available where they are needed, since the compiler does not allow circular dependencies. The solutions to this is either to use singletons for important classes that should be readily available, or to utilize the notification system of IOS, which is essentially sending messages to other instances with a notification object as a proxy. Both solutions create code that is harder to follow, but singletons seemed to us as preferable, since it was simpler. The circular dependency issue could in some cases also be solved by defining a class with the same name as you need in the header file rather than importing that class. This can be seen more or less as a hack, but it works.

Each protocol was written so that it supported multiple send operations that were to be called whenever a change was made, as well as a receive operation that was called by the Network Controller whenever a package was received for that protocol. Messages were built as strings with "|" as separators and "≤" as an end of message marker.

Implementation of the actual receive and send operations was simply a lot of tedious work without any serious problems.



The picture above depicts the network lobby interface.

6.3.2. Testing and Analysis

The testing sessions that followed simulated the use case where the referee and a group are playing a session using multiple Ipads, all in one location. As it was still seen as quite unnecessary to try the application with a distant participant when we primarily wanted to test network functionality, this seemed as a reasonable and simple choice. In all other respects, the setup was similar to the test in the first phase, although with one Ipad for each participant and five participants instead of four. Multiple tests was performed though, rather than just one, as small fixes of no real design consequence had to be made for anything to work at all.

One of the main issues that were found was a problematic de-synchronization of tokens that occurred when two users tried to move the same token at the same time. This would have to be fixed by some kind of roll-back and prioritization system.

Another problem was the fact that the network connection goes into an odd hibernation state when the application is suspended; such as when the user starts a different application or does something in the web browser. This is not that problematic for client Ipads, as the connection handles the received packages when the application returns from its suspended mode. It does not work well for the host though, as it is supposed to forward messages to other devices. This means that the whole application becomes de-synchronized.

Lastly, there were huge problems with Ipads simply dropping their connection randomly. The suggested solution for this was to implement decent reconnection algorithms.

An important suggestion from users was the possibility to draw background textures in the same way as one draws lines. This would allow for the creation of quite good-looking maps, while still being quite easy to implement, as we already had background images for each tile. This type of drawing can be found in Virtual Tabletop applications for Ipad, such as Dungeon Mapp. It was also suggested that it should be possible to change the main background for the map in some simple way.

Testers also voiced the opinion that it should be possible to let tokens display some kind of status or other information. How to do this exactly was discussed a bit, but we decided that this was still not vital for the project, as it was seen as utility functionality.

6.4. Phase Three: Map and Network Refinement

Since development started later than expected, and the first two iteration cycles took more time than expected, this would have been the time when we were supposed to start preparing for our final tests. Since the application was in no way ready to be used in such a test, we decided to add more weeks for further iteration cycles.

The third phase focused on adding background image drawing, main background selection and trying to fix the de-synchronization issues found during testing. This phase contained a lot of testing sessions entwined between implementation. These tests had a very relaxed and unplanned nature, as they occurred spontaneously whenever the logistical conditions were favorable.

6.4.1. Design and Implementation

The primary design decisions that was to be taken during this phase was how to add background image drawing and switching the main background in a way that did not clutter the interface too much, and without taking too much time to implement. As we had realized that getting IOS components to do what we wanted was never an easy task, we reasoned that reusing components that we knew already worked well should be done if possible.

6.4.1.1 Background Image Drawing

Drawing images would essentially be the same thing as drawing lines, although where the finger moved, instead of lines forming, the background texture in the tile touched should be set to some selected image. The user thus needed a way to select which texture should be drawn when doing this, and a new application mode for doing it. Therefore, a new button was added to the menu, and the selection menu for tokens reused for this mode, although instead of containing tokens, it would now contain tile textures. For testing, a few different textures were added depicting different types of floors and undergrowth.

It was soon realized that the texture in the selection menu would be hard to discern from the background if it was displayed over an area where that texture had already been drawn. Therefore, a border was added to the texture in the selection menu, making it more visible.

There would also have to be an eraser function that resets the background image of a tile to the main background. It was decided to add a transparent texture with a border surrounding it to the selection menu, allowing for this draw operation.

We also discussed whether it would be necessary to implement some kind of area drawing functionality, but decided that the image drawing already was quite fast as it was, and that it would have to suffice.

6.4.1.2. Main Background Switching

When it came to changing the main background there were a few issues discussed. Would it be done in a separate mode, or would the choice be hidden away in a deeper menu system for changes that one does not make very often? We assumed that changing the main background is done quite seldom, and as such, it was not equal to the already existing modes. Yet, the easiest solution would still be to use the same selection menu as was used for tokens and background image drawing. If this was to be used, which we really wanted to do, it would seem inconsistent if it was not a separate mode among the others.

Because of this we decided to implement it as a mode anyway, with all different main backgrounds in a selection menu. Whenever an image in the selection menu was pressed, it would change the main background image, and thus the texture displayed in all tiles that had not been drawn to with the background image drawing feature.

6.4.1.3. Tokens

Firstly, token stacking was re-implemented so that all tokens present in a tile would be draw in back to front order, rather than only the top token being drawn.

In order to solve the token de-synchronization problem identified during testing though, the application that is hosting would, in the token network protocol, need to use a first-come, first-serve policy where the first network message for picking up a token would be the right one, and only that message would have to be forwarded to all other devices. The hosting device itself would of course always have priority over all other devices. Since the correct message would always be forwarded, the state would also repair itself automatically. For safety, some synchronization locks were also added around token objects to hinder multiple threads from modifying them at the same time.

6.4.1.4. Wi-Fi Problem Solving

We also made some research into the Wi-Fi problems that occurred during testing, and found that applications normally suspend their wireless connection on inactivity in order to save battery power. We also found a build flag that one could set to tell an application that it needs a Wi-Fi connection. This should supposedly hinder IOS from doing this. Applications were also found to drop Wi-Fi whenever it enters screen saver mode, which was solved by not allowing the application to do so either.

With all these changes, it still occurred sometimes that an Ipad suddenly, with no apparent reason, lost its Wi-Fi connection. No real solution for this was found, and we instead opted to use another Ipad when this happened. We did however implement a reconnect algorithm that

tries to reconnect and resynchronize the application whenever the connection is lost. This would work in most cases, but not all too well if it would be the hosting application that drops out.

6.4.2. Testing and Analysis

Testing proceeded in the same fashion as earlier, although we were now back to having a group of four participants, still playing Pathfinder.

First of all, the background image drawing was much appreciated, as it looked nice and made it possible to quickly draw maps. During testing, it was soon realized that doors had to be drawn with the freehand drawing tool, which did not look good and took a lot of time to do. It was suggested that it should be able to add movable objects, such as doors, that would work like tokens but only be movable in their own mode.

Because of a lot of accidental modifications to the map, especially by players, it was also suggested that the application should have a specific mode for playing rather than creating the map. In this mode, only tokens would be movable. This would make accidental modification of the map less of a problem.

Another suggestion concerned the snap to grid line drawing, which was mostly used to draw walls of buildings. If it could be possible to allow this drawing to draw wall textures rather than just a black line, it would look much more appealing. It would also be easier to discern the lines from the background tile images, as these sometimes already had a black border. These snap to grid lines also combined badly with background tile images when diagonally drawn, since there was no way to only display the drawn background tile in the correct half of the tile. This was seen as a somewhat minor issue though, since it was purely aesthetic.

Since the map was so large, it was also sometimes problematic to navigate it properly. Some suggestions for helping with this was a smaller version of the whole map being visible somewhere, with a square indicating where on the map one currently is. Another suggestion was a function that centers the map: i.e. brings it back to its default position when starting the application. Lastly, it was suggested to add a function that brings a client to the same map position as the host is currently at, as it was assumed that the referee would host the session, and where the referee is would probably be where the players would like to be.

Finally, it was also stressed that for further testing to be effective, a save/load function would have to be implemented, so that it would not be such tedious work to resume role-playing each time. Drawing the same map over and over was not seen as very productive.

6.5. Phase Four: Further Map Refinement

The fourth phase of development focused on adding functionality suggested during testing, such as save/load and aids for navigating the map. Testing was essentially done in the same way as in phase three.

6.5.1. Design and Implementation

As the functionality of objects would be so similar to that of tokens, no complex design decisions where to be taken regarding that. Map navigation on the other hand was problematic, and a lot of thought was put into how to solve that issue well, but no good solution was found. Implementation of load and save was partially decided based on how the network lobby was constructed, since it was already prepared with a table where the hosting user should be able to select to continue a previous session.

6.5.1.1. Objects

The suggested object placement and movement mode was implemented in exactly the same way as the token mode. This was done by adding a new view similar to that used for tokens, as well as a network protocol exactly the same as the one for tokens. The same type of selection menu was used for selecting which object to add, and two door images were added to it: one depicting a door between two vertically adjacent tiles, and one between two horizontally adjacent tiles. This meant that each of these objects actually covered two tiles.

6.5.1.2. Play Only Mode

The play only mode was very simple to implement, as all that was done was to add a new button to the menu, which when clicked changed a few variables in the application, making it so that only tokens would be movable and no drawing would be possible.

6.5.1.3. Wall Texture Drawing

Although wall texture drawing was not seen as an extremely vital feature, we did decide to look into it, just to see if there was a simple way of doing it, and luckily there was. No real change in the draw implementation had to be made, as the only thing needed was to set a specific texture to draw. We thus added the selection menu to the snap to grid mode as well, and displayed a set of textures in it to choose from.

6.5.1.4. Map Navigation Aids

When discussing what to be done with map navigation we only saw two different solutions: a mini-version of the map situated somewhere on the screen, or a navigation mode that replaces the map with an approximation of the whole map. Allowing for zooming out further would not be an option, as this would mean loading too many tiles into memory at once, which was what we wanted to avoid from the start. The mini-version was deemed as cluttering the screen too much, and approximating how the screen would look when completely zoomed out would be a quite taxing operation that we did not know how to implement. Since we found no obvious solution, it was decided to only implement the other two suggested functions for map navigation: go to center position and go to host position, and move on to testing to see if these would be enough. These functions were implemented so that they were called when pressing buttons in the menu.

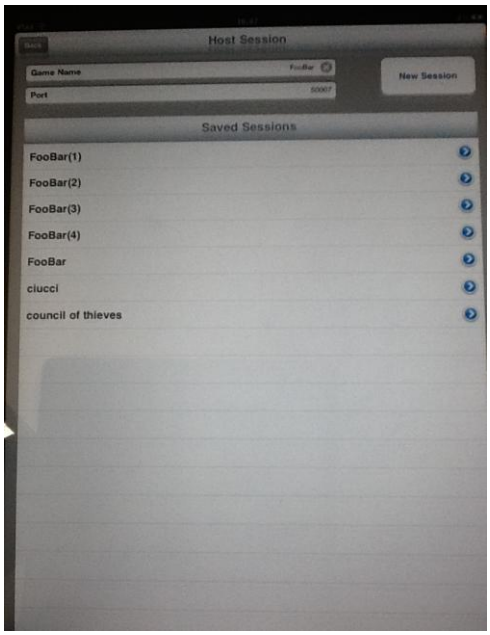
On a side-note, it was also decided to add a button for turning the grid net on and off at this point.

6.5.1.5. Load and Save

As the user tests were suffering a bit from the lack of a load/save function, this was also a priority to implement in this iteration. Implementation was done by using the IOS version of serialization; i.e. archiving the application's map object with an NSKeyedArchiver. For this to be possible, all classes contained in the map object had to be prepared for doing so, by implementing encoding and decoding messages.

An annoying problem that was found when doing this was the fact that it was not possible to archive instances of NSArray or NSDictionary that contained NSValue, which was something we used a lot in the application in order to store point values. There was no explanation anywhere on why this was the case, and no error messages were given. Thus, we had to work around this by writing tedious and more error prone code to archive these objects.

The save function was added to the upper left menu as a button, and the load function added to the network lobby.



This picture depicts the loading function in the network lobby.

6.5.2. Testing and Analysis

Testing proceeded in the same fashion as in phase three, still with four participants and playing Pathfinder.

Although still hampered by random Wi-Fi loss, tests seemed to indicate that the application was improving, and the added functionality was appreciated. The menu now had a bit too many buttons, and the general interaction with the interface was a bit clumsy. Rethinking the interface was suggested, both for aesthetic and interaction design reasons.

After a few testing sessions it was also realized that a way to delete saved sessions in the application would have to be added, as well as a proper way of handling different sessions being saved with the same name. Automatically overwriting any existing session with that name was not seen as being adequate.

When it comes to the play only mode, it was appreciated, but it was suggested that this mode would not have any menus displayed at all, and that it should be possible to scroll with one finger as well, as no drawing was to be performed in that mode. This seemed as sound ideas that would improve interaction for the players a lot, as this would be the mode in which they would be most of the time.

As saved files were reloaded, and many lines drawn and erased over time, it was noticed that sessions took longer and longer to reload, the more lines were drawn and erased. This was of course because of the way redrawing of lines was done, as well as the fact that erasing was the drawing of transparent lines. A suggested solution was to clear all lines from a tile that was essentially empty.

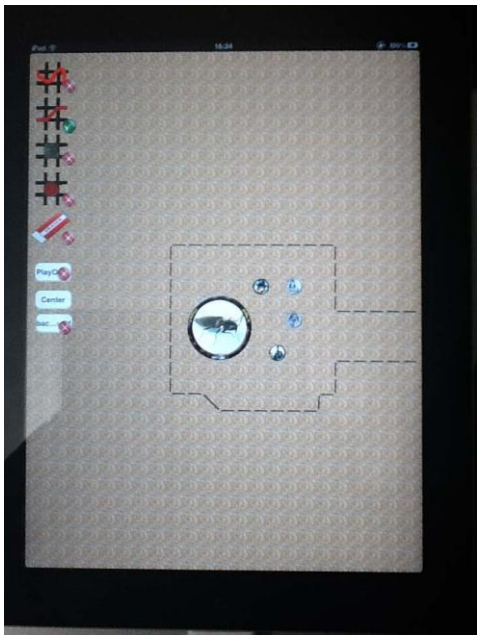
Lastly, we also realized that object movement and adding new objects to the map was problematic, because doors covered such large areas even though the actual door was small; i.e. two whole tiles. When trying to add doors in the vicinity of other doors, one would very often instead touch the transparent area of a tile containing a door, and thus initiate movement for that door. This could of course be solved by changing the add operation to the drag and drop version that was discussed when implementing the selection menu for tokens in phase one. This would

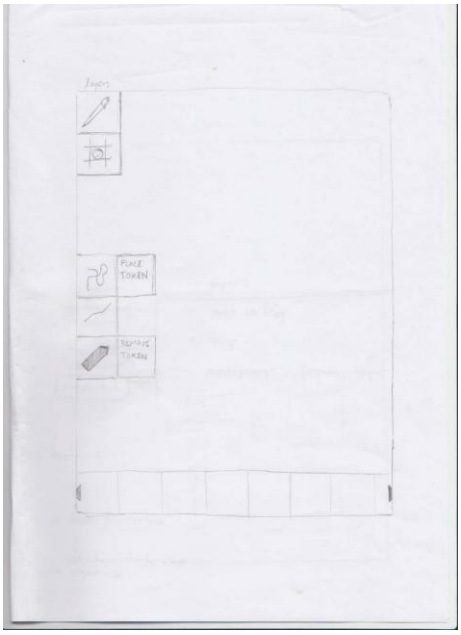
make the interaction in the object mode less coherent with the interaction in other modes though. We therefore decided that it would be kept as it was for the time being.

It was also suggested that, even though we did not want to implement additional information on tokens, we should at least display a unique number on each token, so that generic tokens could be separated in some way. We decided that this was easy enough to implement, and that it actually would make the application more usable.

6.6. Phase Five: Final Improvements

The fifth phase of development mostly focused on the redesign on the interface, but also added some suggested features such as a tile clearing algorithm and displaying a unique number on each token. Testing proceeded as before.





The first picture depicts the interface before redesign; the second picture depicts a paper mockup made during redesign.

6.6.1. Design and Implementation

The most important design decisions in this phase was of course how the interface should look, and if interaction could be streamlined in some way. We wanted the menu to become cleaner, and that it would be possible to remove all visible menus and buttons, and thus enter the play only mode.

6.6.1.1. Improved Devices

During this phase we were also able to use Ipad 2 devices rather than just Ipad. After noticing that the performance difference between them was quite large we decided to change the target device for the application to Ipad 2, as it would mean less time needed for optimizing code, and more time to actually polish the application. This also gave us the possibility to greatly increase the level of detail on textures without a decrease in performance. Before this change, the background textures looked quite pixelated when zoomed in, and because of the improved memory of the Ipad 2, we could afford to swap to a texture with higher resolution when the user zoomed in.

6.6.1.2. Interface Redesign

We started the interface redesign by putting all menu items in a drop down menu on the top left, and added an animation and buttons for showing and hiding this menu. We also made it so that the application interprets a hidden menu as being in the play only mode. It was not hard to add one finger scroll to this mode either, so this was quickly done.

Now, we wanted for the top left menu to be a menu for switching between modes, and nothing else. We thus created another menu in the upper right, with its own show and hide buttons, and added all functions that were not mode-based to this menu: i.e. turning the grid net on and off, move to center, move to host and save. We also added a clear map button as well as an exit button which takes the user back to the network lobby, as we realized that this was functionality that definitely should be in that type of menu. The top left menu was left with main background switching, background image drawing, free-hand drawing, snap to grid drawing, objects, and tokens. We also had an eraser button that we did not know exactly what to do with, as it felt a bit out of place as a mode of its own, as it was specific to erasing drawn lines.

We also realized that all modes except freehand drawing had a selection menu at the bottom. Since we had never come around to implement the selection of which color to draw in, we decided to add colors to a selection menu for this mode. This would make all modes more coherent with each other from an interaction flow perspective.

Since all modes now had a selection menu, we decided to put the eraser button to the left and above of it in all modes except main background switching, allowing it to be selected instead of something in the menu. This would allow for a mode-specific eraser, yet with the button still being at the same place. Since it was no longer needed, the transparent background in the selection menu for the background image drawing was removed, as well as the double tap deletion of tokens. To indicate whether the eraser button or something in the selection menu was selected, the item not selected was set as semi-transparent.

Lastly, we wanted to get rid of the small round icons that indicated which mode was currently selected, as frankly, they were very ugly, and the animation that changed them was annoying to look at. We therefore decided to use semi-transparency on the buttons for the non selected modes instead, removing the selection indicator icons completely.

We also realized that we were not giving appropriate feedback to the user when the application did things that would make it non responsive, such as loading, saving or reconnecting and re-synchronizing the application. This was remedied by added a dimming layer and a loading animations whenever these cumbersome operations were in progress.

Displaying numbers on tokens was also added, as well as the network synchronization of a counter that decided the number for the next token created.

The above pictures depict the redesigned interface.

6.6.1.3. Tile Clearing

In order to clear tiles that were empty, i.e. their displayed content was transparent, an algorithm had to be implemented that determined this. This was not trivial, but after some research and trial and error, it was possible to determine the color of each pixel in a tile, and thus checking them individually for transparency. We allowed the application to perform this check each time a line was added, and if all pixels were transparent after drawing all lines to the tile, all lines where removed.

6.6.1.4. Memory Management

During this phase we also found a neat function in X-Code called the analyzer, which checks the application for improper memory management. This essentially led to an overhaul of most of our allocated objects, as we had previously not made much of an effort to keep track of allocation and de-allocation. We therefore went through all classes, adding proper de-allocation on everything, as well as setting a lot of objects to automatically release themselves when they were no longer in use. After these changes, the analyzer could no longer detect any memory leaks, which we hoped would be a huge stability upgrade for the application.

6.6.1.5. Improving Saved Sessions Management

In order to support the deletion of sessions, we started out by trying to implement a button in each row of the table displaying all sessions that can be loaded in the network lobby. However, this did not work properly, and as research was performed in order to find a solution, we found out that the component used for the table already had a standard way of deleting rows. We thus gave up on implementing our own solution, and instead used the standard solution. The problem with this solution however, is that it is quite hidden, as the user needs to swipe over a row in order to bring up the delete button. This may seem unintuitive, but as it is also seems to be the

standard for this component in IOS, we decided to use it anyway. At least, there would be a very low risk of users deleting saved sessions by mistake.

The issue with old sessions sharing the same name as a new one and the new session overwriting the old one was solved by checking for old sessions before saving, and appending a number to the new session name if an old session was found; thus distinguishing between them in a quite conventional way.

6.6.2. Testing and Analysis

Testing proceeded in the same fashion as in phase four, still with four participants and playing Pathfinder.

The new interface was seen as a huge improvement, as it felt much cleaner and logically consistent. Being able to remove menus made it possible to utilize the whole screen when using a map rather than creating it. With some main backgrounds selected, the use of transparency on components was a bit problematic, but not so much as to make it unusable.

What was really still lacking, from an interface perspective, was undo functionality, as it happened quite often that a line was drawn erroneously. This would be complex to implement though, as the application would have to keep track of each line drawn, and remove lines by redrawing the whole buffer in each tile. This would be a heavy operation to perform, and thus it would probably lag.

The most prevalent problem was still the Ipad's lack of Wi-Fi stability. This made some testing sessions unbearable; as devices became out of sync and had to reconnect many times. It is of course possible that this could have been solved with a stronger access point, as we were using the router of a local network in order to resolve firewall issues. Having an external server that all Ipads connect to would probably solve this problem, but one cannot know for sure.

An interesting observation made over the whole of the development cycle was that the Ipad was initially prone to distracting the user. Many features other than the application itself were readily available for exploration, and therefore lured the user away from social immersion. This became less of a problem over time, as the users familiarized themselves with the device and its other applications. However, it seems likely that the Ipad might still distract users that are not in the same room as the others, as the social immersion by default can be assumed to be less strong for that user. It might therefore be a good idea to implement some kind of indication, visible to all participants, that a user has suspended the application and is doing something else, so that other participants can react to this.

In general, the voiced view was that the application was good and useful when it worked. We thus decided that the design goal of having tools useful during normal circumstances was now reasonably met. We also believed that our application did adhere to the standards for the App Store, so the goal of possible future submission had not been ignored, although the application itself was not polished enough yet.

6.7. Final Testing

Since the application was seen as good enough with the respect to being a natural part of the activity, it was now necessary to test it in the main use case: i.e. the referee and a group are playing a session using one Ipad, while one player is in a different location, using one Ipad. This would have to be a much more prepared test, with interview questions formulated so that we can in some way measure how well the application actually works; i.e. if it serves to aid in the immersion in the role-playing activity.

6.7.1. Interview Questions

As the interview questions were to measure the immersion of a participant, they relied heavily on the definition of different types of immersion in role-playing games (see section 4.7.). The interviews focused on the perceived experience from the remote player's point of view, as this was always the focus of the project as a whole. Normal in the context of the interview refers to a normal role-playing session without a remote player.

6.7.1.1. Emotional Immersion

- Did you often feel that you were thinking as your character, rather than as yourself?
    - Was this different from how you normally think?
- Did you often speak as your character, rather than as yourself?
    - Was this different from how you normally speak?
- Were you often addressed as your character, rather than as yourself?
    - Was this different from how you are normally addressed?

6.7.1.2. Social Immersion

- Did you feel that you were socially a part of the group while playing?
    - Was this different from how you normally feel?
- Did you feel that you were a part of discussions that were not about what happened in the game?
    - Was this different from how you normally feel?
- Did you at any point feel that your view of how things happened in the narrative was different from the view of the referee?
    - Did this happen more often than normal?
    - Did you discuss this with the referee?
        - If yes, was the discussion fruitful?
        - If no, why not?

6.7.1.3. Spatial Immersion

- Did you, outside of combat scenes, have a good picture of where your character was situated in the world?
    - Did you feel that the application aided you in this respect?
- Did you, in combat scenes, have a good picture of where your character was situated in the world?
    - Did you feel that the application aided you in this respect?

6.7.1.4. Temporal Immersion

- Did you at any point feel suspense about the result of one of your die rolls?

    - Was this in some way different from how you usually feel about your die rolls?
- Did you at some point feel suspense about the result of another person's die rolls?
    - Was this in some way different from how you usually feel about your die rolls?
- Did you feel that you had a good picture of what decisions or events brought the story forward and how events and decisions were related to each other?
    - Did you feel that you had any control over these decisions or events?
    - Was this in some way different from how you usually feel?

6.7.1.5. General

- Is there anything lacking in the application that you think would improve your immersion?

6.7.2. Testing Setup

The testing setups used two large TV screens, two web cameras, as well as two computers running Skype for video and audio communication. The distant participant used a headset, while the group used speakers and a microphone. Distance was simulated by having the participants in different rooms, while still being in the same building. This made it possible to run the applications on a local network rather than through an Internet connection. During the tests the same test subjects as before were used: three players and one referee. At this point, only three Ipads were available though, so the two players that was in the same room as the referee had to share one. This was of course, less than optimal, but there was not much to be done about it. Two tests were performed during two separate sessions, with two different persons trying out being the remote participant.

6.7.3. Results

Testing sessions went quite well, with little technical problems once the setup was up and running. Some delay in the application as well as video delay on Skype occurred from time to time, but only in short periods. Everybody was quite familiar with the application, and there was not any problem in using it at all, and it seemed to fit into the sessions quite naturally. At first we placed the camera in a way that meant that the group would never look into it while talking to the remote test person. This was remedied after complaints about not being able to make eye contact with anyone. Having the camera on top of the screen seemed optimal.

6.7.3.1. Emotional Immersion

The first remote test person felt that it was more difficult than usual to think as his character, as he was not feeling as much a part of the group. The experience was described as not really being there, and that it was hard to become engrossed in the fantasy. The second remote test person disagreed with this though, as he felt that he was thinking as his character just as much as usual. It was suggested that the difference could at least in part be explained by the difference in how much they liked their characters, as the first remote test subject did not care as much for his character as the second.

The same difference between remote test subjects was voiced when it comes to speaking in character, with the same possible explanation. The second liked his character and did not have any problems; the first did not like his character as much and did experience problems.

Both felt that they were addressed as their character just as much as usual. The second remote test subject had the general feel that there was more in character speech compared to out of character speech though, but only because out of character speech was lacking.

6.7.3.2. Social Immersion

Both remote test subjects felt that they were not socially as much a part of the group as usual.

When it comes to discussions concerning what happened in the game, the second remote test person felt that he could participate just as well as usual, while the first remote test person felt that the group had a clear leader structure that made discussions generally unnecessary. The group agreed on this, as the characters did actually have a clear leader structure based on the social status of one character in the game.

No remote test person felt that there was a difference in how they viewed the narrative compared to the referee, so none of them engaged in any arbitration. The second remote test person did however point out that he used the application to poll the attention of the referee at one point, by moving his character's token somewhere while stating what he wanted to do. He also pointed out that this did not work as well as he had hoped, as it was hard for anyone to notice that he was doing anything on the application.

6.7.3.3. Spatial Immersion

The first remote test subject did not feel that he had a good understanding of where he was situated in the world outside of combat scenes, but he did not think that this had anything to do with being non-present, as he generally did not feel comfortable with the theme of the scenario being played, and therefore had trouble visualizing the places where his character went. The second remote test subject felt that his understanding was as good as usual, and that the application helped with this during site exploration; that is when a certain building was being explored, and different rooms were being drawn.

Both remote test subjects felt that they had a good picture of where their character was situated during combat scenes, which they attributed to the use of the application.

6.7.3.4. Temporal Immersion

The first remote test person felt normal suspense when it came to rolling his dice, but the second complained that, since no one could see his dice being rolled, there was not any suspense in doing so.

Both remote test persons complained that there was no suspense when it came to other players rolling dice, as they could not really see the results, only hear them being stated.

Both remote test persons felt that they could clearly see how decisions and events brought the narrative forward, just as when playing normally.

6.7.3.5. General

During the general discussion on what features could possibly improve immersion, it was suggested that the main problem was participation in low-intensity activity that was not directly forwarding the game, i.e. off-record irrelevant speech. It was also said that there was a huge difference between "doing nothing" when actually being in the same room, and "doing nothing" when being distant, as the latter did not make you feel part of the group, while the former still

did. Thus, the more focused the group was on playing the game and forwarding the narrative, the more immersed did the remote test person become.

One feature that was suggested to remedy this was a chat feature that in some way allows the distant player to use a medium of communication that does not immediately bring all attention of the group to that person. Being able to talk to one person at a time was missed, so the chat feature should be both global and to a specific device. This of course would mean that the interface must show the different connected players in some way as well. When we showed the keyboard feature of the Ipad, it was suggested that we did not use it at all though, as the test participants would rather draw text with their finger than use it. During discussion, it was suggested to have a different draw layer for each person, with controllable visibility and the possibility to easily erase it. This is very similar to a feature that we previously had suggested to the group when testing the application in the use case with no remote participant, but at that point no one could answer if it was a good idea or not.

Being able to poll attention was also discussed, as the remote test persons felt that this would have been good to have. Sound was suggested, although it was also assumed that it could be very annoying. A more subtle suggestion was to make it clearer that a person is moving a token so that this will in itself draw attention to that something happens in the application. A suggestion was some type of pulsation animation, or a light halo with a color indicating which player is currently moving it. It was also suggested that there could be a trail after the token for a certain time, possibly in the same color as the halo, so that one could still see what someone had done even if one glanced away from the application for a few seconds.

Implementing a die simulator in the application was seen as an unnecessary idea, as it could still not compete with real dice. It was seen as preferable to have multiple hand-held cameras instead, that could be used to show the result of die rolls.

The importance of a delay free application, as well as delay free video and audio communication was stressed. The possibility of having a sound library in the application, so that sound effects could be sent throughout all connected devices, and possibly even placed in areas of the map was suggested, although the test persons disagreed upon if this was good or not. One thought that it would be a good feature; the other thought that people actually performing sound effects were part of the game.

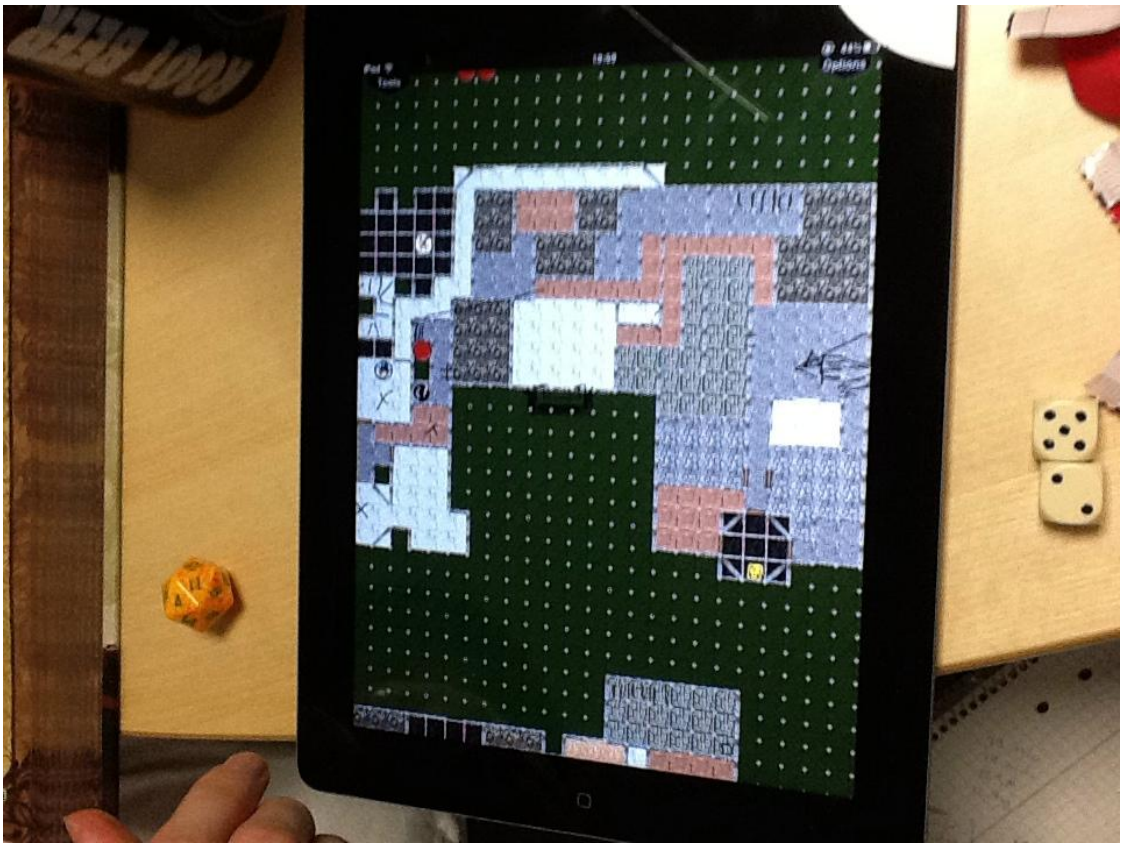
Some utility functions were suggested as well: markers for describing the state of tokens, such as dead or lying down, as well as their current health level. This was seen as helpful in making meaningful decisions during combat scenes. One could of course always poll this information from the referee, but since this would bring all attention to the distant person it would assumedly be avoided. Another suggestion was that it should be possible to indicate a token that one interacts with during combat scenes, as this would also help the distant person's understanding of the tactical situation. Visual effects indicating the outcome of this interaction was also suggested as a fun feature for a more highly developed application.

A suggested way for a person to indicate things was to simply have a tiny light effect that traces the movement of the finger on the application, color-coded for each participant. Another idea was that it would always be possible to in some way see the area of focus of the application of each other participant. The suggestion was to have a visible color-coded frame indicating this, either on the main screen or on a mini map.

It was also firmly stated, that it would have been better if everyone was in a different room, so that the group composition was more equal.

7. Tabula Imaginarium

The application that has been developed during this project has been named Tabula Imaginarium, and it can be described as a virtual tabletop application for the Ipad. It presents tools that can be used to create tabletop role-playing maps on the fly, although preparing maps in advance is also possible. The application automatically synchronizes the content of the currently used map between connected applications. Maps have grid nets and movable tokens that players and referee can use to indicate characters or monsters. Maps can be created by drawing textures into grid squares, or by free-hand finger drawing.



Picture of Tabula Imaginarium

7.1. Map Layers

A map in the Tabula Imaginarium application consists of four different drawing layers, all divided into square tiles by a grid net. The first layer contains the background that is the texture drawn to a grid tile. The second layer contains finger drawings, either texture-based in the case of the snap-to grid wall-drawing tool, or color-based in the case of free-hand drawing. The third layer contains movable objects that can cover one or more grid tiles in a rectangular area. Currently, the only objects available are doors though. The fourth layer contains movable tokens that can be used to represent characters or monsters. These can also cover one or more grid tiles in a rectangular area. What differentiates objects from tokens on a functional rather than conceptual level is that objects cannot be moved in the play-only mode that is supposed to be used by players, while tokens can. The four layers are drawn on top of each other in the order mentioned. A tool which clears all layers of a map is provided in a drop-down menu, as manually erasing everything would be very tedious.

7.1.1. Background Layer

The background layer draws the current background texture of all modified tiles, as well as the standard background texture, to all textures that are unmodified. Tabula Imaginarium contains two tools to modify the textures: one that changes the standard background texture for all unmodified tiles, and one that allows the user to select a texture and draw it to any tiles touched. It is also possible to select the eraser in the texture drawing mode, which will allow for the removal of the current background texture for any modified tiles touched, thus making the background layer again draw the standard background texture to that tile. The grid net itself is drawn on top of all other layers as tiny cross-shapes in the corners

7.1.2. Finger Drawing Layer

The finger drawing layer draws lines that have been added by free-hand finger drawing. Tabula Imaginarium contains two tools for finger drawing on a map: one that allows for freehand drawing in a selected color, and one that allows for a snap-to grid line drawing of a chosen wall texture. It is possible to select the eraser in any of these modes, which will allow for free-hand erasing in the form of a drawn line of transparency, which removes any type of line drawing below it.

7.1.3. Object Layer

The object layer draws objects to the grid tiles that they are currently placed in. Objects can cover one or more tiles in a rectangular area, as determined by the file name of each object. It is thus not possible for the user to scale or rotate an object manually. Tabula Imaginarium contains one tool for placement and movement of objects, which currently only contains door objects. Since movement take precedence over placement, it is impossible to place a new object on top of an old object, as the application will interpret this as an attempt to move the old object. Objects can be placed next to each other and then moved on top of each other though. It is possible to select the eraser in this mode, which allows for deletion of any object touched.

7.1.4. Token Layer

The token layer draws character or monster tokens to the grid tiles that they are currently placed in. Tokens can cover one or more tiles in a rectangular area, as determined by the file name of each token. It is thus not possible for the user to scale or rotate a token manually. Each new token receives a number which is displayed in its bottom right corner, which can be used to distinguish similar tokens from each other. Tabula Imaginarium contains one tool for placement and movement of tokens and one play-only mode which allows for movement of tokens.

The placement and movement-mode currently contains a few generic tokens and some tokens depicting monsters common for the Pathfinder role-playing game. Since movement take precedence over placement, it is impossible to place a new token on top of an old token, as the application will interpret this as an attempt to move the old token. Tokens can be placed next to each other and then moved on top of each other though. It is possible to select the eraser in the token placement and movement mode, which allows for deletion of any token touched.

The play-only mode minimizes the user interface to two small buttons in the upper left and right, while only allowing for movement of tokens.

7.2. Map Synchronization

The core part of Tabula Imaginarium, aside from the creation of a map, is its ability to automatically synchronize the content of the currently displayed map with all other connected applications. This is done through the use of a server-client model, where one instance of the application is hosting a session for all other applications, thus allowing them to synchronize with the map of the hosting device. All modifications done to the map are sent to the host, and then distributed to other connected devices. In order to connect devices to each other, Tabula Imaginarium features a network lobby, in which users can see other users of Tabula Imaginarium as well as their current session status: i.e. if they are currently in a session, or are currently hosting a session.

7.3. Map Orientation

Tabula Imaginarium allows for very large maps. The standard way of orienting the screen with respect to the map is the use of a two-finger scroll gesture. It is also possible to zoom in and out to a certain degree by the use of the pinch gesture. All textures used in the map are stored with different levels of detail, so that a high resolution texture will be displayed when a tile is zoomed in. As the maps can become quite large, the application contains a feature for moving to the center of the map, as well as to the part of the map that is currently in the view of the hosting device.

7.4. Saving and Loading of Maps

Tabula Imaginarium also allows the user to save maps by pressing a save button, which makes the map accessible the next time a session is started. The map can then be selected in the network lobby when hosting a new session. If so, the map will be automatically sent to any connecting devices.

8. Discussion

How well were our design goals fulfilled? Did the application allow for immersion for a remote player? Is it an application that is perceived as a natural part of the tabletop role-playing activity? Could Tabula Imaginarium be released to the Apple App Store in a close future, and would it be usable in any type of tabletop role-playing game? What is good with the application, and what could be done better? These are the questions that are discussed in this section.

8.1. Activity Immersion

As our main goal was for the application to help in solving the problem with a physically absent player attending a session, it is of course important to ask if it fulfills this goal. Our measurement for this is the perceived immersion in the role-playing activity of the remote test persons during the final tests.

8.1.1. Spatial Immersion

From these tests it seemed clear that the application did indeed help the spatial immersion of the distant player, especially when enacting scenes part of a tactical combat scenario, but also during exploration scenes when the characters were looking through different rooms in a building. Cover states that tools such as miniatures and battle maps, which is essentially what is implemented digitally in the application - although the maps do not necessarily need to be used for combat, exist more for showing spatial relationships than for immersing players visually (2010, p.109). Our results are in no way contradicting this, but they stress that the possibility to show spatial relationships when it is needed is important for spatial immersion.

Cover also state that scenarios that focus on puzzle-solving and information gathering might not need such tools at all (2010, p.109). It seems to us as if the important question is how much need there is in the scenario for diagrammatical interpretation though, not if it involves combat or not. As stated in the literature study, diagrammatisation is often needed during tactical combat scenarios, but it could just as well be needed during other types of scenarios as well. It is not hard to imagine an investigative scenario or a puzzle where the spatial relationship between objects would be important to convey, and as such, the implemented feature of our application must be seen as an important one.

8.1.2. Emotional Immersion

When it comes to emotional immersion, no effort was made to enhance this with the application, but it seemed like a good video and audio conference setup was in itself at least good enough. Our tests indicated that it might rest on the relationship the player has to its character though, as the test person that did not like his character had problems, while the one who liked his character had not. One could draw the conclusion that this means that having a distant participant may not be a good idea if the players have not already played with their characters before. This correlates quite well with Cover's observations that emotional immersion increases when players continually improve their characters during a longer campaign (2010, p.113).

8.1.3. Temporal Immersion

Temporal immersion was also a type of immersion that our application was not directly built to improve. We decided not to convert die rolls to a digital form, as we believed that players would still prefer rolling physical dice. This seemed to be true for our test persons, but problems with temporal immersion was still identified, as the test persons did not care as much about their die

rolls as during normal play. The suggested solution was not to replace dice though, but rather to make it possible to see the dice of each participant with the use of cameras.

Even though die rolls are mentioned as an important contributor to temporal immersion, it generally rests on the possibility to see the causality of events and on situations that have diverging but computable outcomes. (Cover, 2010, pp.110-111) During the general suggestion part of our interviews, it was proposed that a lot of functionality that we had previously seen as unimportant when it comes to immersion might actually contribute to temporal immersion. In order to be able to compute outcomes in a scenario, any functionality that helps to visualize information can help, provided that the information is needed. During tactical combat scenarios there is obviously a lot of information that could be visually represented, but as already stated, that might be true for other scenarios as well. Maybe temporal immersion rests mainly on the presence of important diagrammatical signs to interpret, with spatial relationships being one type of such a sign?

8.1.4. Social Immersion

Lastly, social immersion was a type of immersion that we wanted the application to aid players in, but which we did not really know how to implement well, since writing is so problematic on an Ipad. We did suggest different drawing layers during our regular tests, but did not get any positive feedback concerning it, and therefore focused on other features instead. During our final tests these drawing layers were suggested as preferable to text chat, and they were proposed as a means to improve social immersion. However, social immersion was not as lacking as we had previously expected, as the only type of speech that the distant participant did not engage in was off-record speech. This made passivity problematic, and did impact group cohesion, but not to an extent that made the participant feel left out of the role-playing activity itself.

Participation in all frames seemed present, although the shift to the primary framework was perceived to occur less. Test persons did however voice the opinion that more communication channels would be good to have, especially if they could be used to communicate with a specific person without receiving the attention of the whole group. It could as a matter of fact be said, that communication itself was not the problem, but the possibility to control the amount of attention drawn to the distant participant. A setup with only an audio and video conference was very binary when it comes to attention, as either you get it or not. It was also very immediate from a time perspective, as no sublime cues that one wanted to receive attention could be given.

Because of this, we believe that further development that focus on improving social immersion should focus on the control of attention with respect to communication. However, it should be noted that it is unclear as to how well inter-player communication would work in a group composition with a different leader structure, as this heavily influences such communication.

8.2. Natural Part of the Activity

Another statement in the design goals was for the application to feel as a natural part of the role-playing activity. Since our testing during development more or less used the same test group, we know for a fact that at least this group started to see the application in this way, although only at the times when it worked as intended. During some sessions when Wi-Fi connections were continuously lost or crashes occurred, it was of course seen as obtrusive rather than natural. To actually know for sure how natural our application feels to use, we would need to test it on a lot more groups though, probably by actually releasing a version for public use, and listening to feedback from users.

It is also not as sure that the application feels natural when used in our main use case, as it has not been thoroughly tested even by one group, though the initial reactions were favorable.

One observation that was quite clear was that the device itself easily became a natural part of the role-playing environment. It is small enough to not obstruct the table too much, and it can easily be held or placed in one's lap for an extended time. During our many tests no complaints were ever made about the Ipad being in the way, and although we initially feared that this would happen a lot, no Ipad was ever accidentally dropped. The web browser and PDF reader of the Ipad were often used in conjunction with the application, which we had anticipated and hoped would happen. As mentioned in the development section, this was a problem for the hosting device though, as a suspended application could not answer to network calls. Therefore, future development should focus on changing the network structure of the application.

When role-playing sessions were played without any access to Ipads, the players said that they missed them, as they did not have access to game rules or a shared map. We see this as a good indication that the application as well as the device worked well as a natural part of the activity.

It was also mentioned as a design goal that the application should be easy to use, although this was essentially a sub-goal to that of being a natural part of the activity. We spent the main part of a complete iteration on restructuring our interface in order to achieve this, and the response in our later tests has been favorable. It seems as if we have managed to create an interface that is reasonably intuitive while not cluttering the screen. To actually be sure if this goal is achieved, we would again need a lot more testing groups, which is a logistically complex question. As there have not been any direct complaints during later testing, we do not think that this should be the main focus of further development though.

8.3. Submission to App Store

Another design goal was to be able to submit the application to the Apple App Store at some point, so that it could be used by others. We are yet not there though, as we still think there are features and improvements that need to be made before reaching a quality standard that we see fit for a public release. To our knowledge we have followed all guidelines and restrictions for submitting an application though, even when they was a hindrance to the project, just so that we maintained the possibility of a future submission.

During the project, content for testing the application have been used that could not be provided in a public version for legal reasons, since we do not own the rights to do so. This leads to the need for one of two things if the application were to be released: create our own content, i.e. textures for tokens and background images, or make it easy for the user to in some way provide their own content. The latter seems to be the easier and most versatile way of to go, and would thus be an important part in any future development.

8.4. System and Genre Neutrality

Lastly, it is also stated in the design goals, that the application should be useful for an as broad a category of role-playing groups as possible. This of course means that it should not be geared towards a specific system or genre of role-playing games; more importantly: role-playing games that present different types of problems for the players to solve should all be viable for the application to be a natural part of. It is very questionable if the application can be said to fulfill this goal, as testing has been majorly done with a game that focuses on one type of problem.

Testing using different systems might be assumed to lead to other suggestions from the testing groups. In order to be useful in all systems it would, as discussed in the section on activity immersion, have to have a broad category of tools that can be used to visually convey information. So, the question is, how broad is the tools that it has? It seems quite reasonable to say that a whiteboard is a very broad and versatile tool that can be used to convey a lot of visual

information. The grid net, tokens, objects, and background images seem to be versatile although possibly less broad, as they focus specifically on map presentation.

What is clear is that future development should test the application in different types of systems and genres in order to further this goal, and that at its present state, one can neither say that the application fulfills this goal nor that it does not.

8.5. Future of the Application

There are obviously a lot of things that could be improved in the application in order for it to actually reach a state where it could be released. There are also additional features that could be added in order to improve immersion. Further testing of the application as it is might also be needed, as the testing already made has in no way reached diminishing returns.

8.5.1. Satisfactory Features

What seems to work well with the application is the interface, as testing after the interface redesign has not yielded any complaints on it. All features provided are used and useful, so none of them should be removed. Texture drawing seemed to be the most liked type of drawing, although this might of course be dependent on our choice of test game. Still, the free-hand drawing is useful as it can be used to draw anything, although it might be tedious and the result not very pretty. As such, it can be seen as a needed catch-all feature for drawing things that is not possible to draw using other features.

8.5.2. Further Testing

First and foremost, more testing of the main use case should be performed, preferably while using the same game system, as well as having a player and character setup that allows for a consensus leadership structure, as this would probably promote more planning speech in the primary framework. Such tests would then allow for more conclusive results as to whether planning speech is hampered or not, and if tools to aid in planning speech would be needed in the future.

Testing of the application using different systems from different genres should also be performed in order to see if the application would be used naturally when playing scenarios that focus on other types of problems, and to generate suggestions for features that would help in dealing with other scenario types. Such testing should probably begin with testing with the whole group being present in the same room.

Finally, testing the application using different audio/video setups would also be a good, as it is hard to say exactly how much the setup used during our testing actually affected the immersion of the remote test persons.

8.5.3. Player Avatars and Indicators

We have always assumed that for the application to be geared towards providing social immersion, it would at some point have to have a visible representation for each player that keys presented information *from* that player *to* that player. During the final testing sessions, this was also suggested by the test persons.

One suggestion on how to do this was to assign different colors to different players. This could be beneficial in several ways, such as adding borders to moving tokens that glow in the color of the player that is currently moving them, adding colored frames that indicate on the map or a future mini map where on the screen a certain person is currently looking. It would also be

possible to add an indication feature that would allow players to point at things in the application by adding a colored glow effect to the place where they put their finger.

For a color to actually be keyed to a player, it would be needed to have some kind of representation of that player as well. Preferably this could be a selectable avatar that would be the same as the character token used by that character with a permanent color border. This avatar could also be used to indicate to users that other users are actually connected. As it currently stands, the application does in no way indicate this. A simpler solution for this would of course be to indicate in text somewhere on the screen when a person connects or disconnects, but as a visual avatar would have other uses as well, it seems preferable.

With visible avatars all players could easily see what other players are connected to the game, and if a player leaves, as that player's avatar would then disappear. The avatar could also be used as a button for performing player-specific operations. It might also be used as a quick way of adding the token of that character to the map, or, if one would only allow one such token at a time, allow for quick navigation to that token by clicking on the avatar.

The only real problem with having avatars is that they would have to be displayed on the screen somewhere, and we already feel that our interface use as much of the screen as it could do without obstructing the view too much. This would mean that the interface would have to be restructured again.

Another interesting idea would be to have the avatar always display a stream of the Ipad camera for each player, provided that they are all using Ipad 2 devices. This could potentially be used to stream die rolls. It would be interesting to look into this, although it should probably not be a priority.

8.5.4. Attention Control

Where users in the same location can easily point out things on their Ipad to another user in the same room, that option did not exist for users in other locations without calling out to all other participants with a verbal description of where they should look. Where users in the same location can subtly hint to the referee that they want attention, a distant user can only call out to all participants, possibly interrupting the flow of the game if attention is needed. This leads to problems, as attention polling is partially avoided. For the application to further aid in social immersion, it should probably be equipped with functions that allow for such attention control. Some of the features suggested in the section on player avatars and indicators would be usable in solving this, such as the pointing feature and the token movement indicator. This might not be enough though, so a simple solution would be to add a button that calls for attention with the use of a visual component that is presented to other users, bringing their attention to the Ipad.

Another suggested way of improving communication that does not draw all attention to the distant participant, was to implement separate drawing layers that can be used for written communication to a specific person. It would take some interface design to administer these layers and to allow users to control the visibility of each layer. But an easy solution would be to add this functionality to the avatar icon in some way.

8.5.5. User-Provided Content

There are many reasons to allow users to provide their own content, but the primary would be that it allows for great versatility with minimum of work. For the application to be useful in all possible cases, it would have to provide a lot of content. As one of the design goals are to allow for a broad category of role-playing games, genre should not be an issue. Yet, when it comes to textures for tokens, objects and background images, there is quite a difference between a

fantasy theme and a science fiction theme. As such, it is easier for everyone if the user can provide textures themselves to the extent that they feel that it is needed. Thus, adding an abundance of images in the hopes that it would satisfy enough users was deemed unnecessary.

This would of course lead to other problems, such as the synchronization of added images between applications, and the potential storing of these images on each device. It might be good to send images on a need basis, and not synchronize whole image libraries continuously between Ipads, as this might seriously affect the performance of the application. This would lead to loading times during play though, which might annoy some users. Thus, there are different ways in which this image distribution could be done, depending on the way the network would be structured. Images could be uploaded to an external web server, sent directly between devices or synchronized through the use of a file synchronization service such as Dropbox, which is available on the Ipad. Dropbox would have the advantage of making it easy to manage image libraries from any computer rather than only being able to do it on an Ipad. It would also allow for image synchronization when not actually running the application, which is also preferable.

One of these solutions would have to be implemented for the application to be released, as well as additions to the interface that allows for the selection and placement of user-provided content.

8.5.6. A Better Network Solution

There are two important issues with the current network solution.

Firstly, network operations are currently handled in the main thread of the application, which means that the whole application is locked if time-consuming network operations are made; such as during a complete resynchronization of the application state. This is especially notable when another device connects to a running session, since it locks the hosting device, and therefore also all other devices. This problem could be solved by running all network operations in a different thread, although this would require some work with thread safety and code restructuring.

Secondly, our tests clearly indicated that the Ipad devices are not optimal as a device for hosting a service. The lack of a stable Wi-Fi connection as well as network suspension when the application is suspended leads to the conclusion that an external hosting device would have to be provided. This would also solve firewall issues when it comes to finding sessions, and make game setup much easier. The only real problem is that someone would have to provide users with this external hosting device.

Lesser problems of course exist as well, and these should probably be solved before releasing the application. Mainly, these problems are questions of optimization when it comes to sending data. For instance, as it stands now, during a state synchronization, the image name is sent on a per tile basis, which is very ineffective. It would be much quicker to send an image name and append all tiles which should have that image as its background image.

8.5.7. Utilities for Temporal Immersion

As mentioned in the section on activity immersion, it was realized during the final user tests that some features that we had previously seen as merely utilities would actually improve temporal immersion in the sense that the understanding of causality might improve. The suggested additions mostly concerned adding information to tokens, such as being able to display information pertaining to its current status. This would potentially also improve the flow of the

game as well as increasing social immersion, as the distant player would not need to continuously ask the referee for this information.

8.5.8. Additional Features

The current version of the application only contains features that were deemed necessary for the user tests. Several tools that should logically be present are not, and they should be if the application is to be released. It should be possible to change the line width when drawing. The actual line drawing should also be improved to use Bezier curves, as they currently look quite ugly when zoomed in. Free-hand drawing and texture-based wall drawing should be separated into different layers, as it would be preferable if the eraser could differentiate between them.

Our user tests indicated that we did not really need more advanced drawing tools though, but more advanced texture drawing would be a nice addition. This especially has to do with texture drawing in squares where a diagonal wall has been drawn. It would be very neat if it was possible to only draw the texture on one side of the wall. A release version should probably have this feature. Another suggestion was the possibility to easily draw the same texture in a lot of squares simultaneously; and in that way quickly draw rooms with different shapes.

Lastly, it would make the application seem a lot more polished if an issue concerning large tokens were to be addressed. Currently, when large tokens move across the grid, it creates tearing when tiles are redrawn. This could be easily solved by adding a separate non-tiled view that displays a token while it is moving.

9. Conclusions

In order to try to improve the feeling of participation in the tabletop role-playing activity for a spatially separated player by the use of Ipads, we have developed an application that tries to aid in upholding or improving different types of immersion in the activity. The activity itself has been studied and analyzed, so that the integral parts of it could be identified.

It was realized that the most integral part of the tabletop role-playing activity is face-to-face interaction between players and the referee, which is something that can only be substituted by a low-delay audio and video conference setup. Since there are already other ways of doing this that are superior to using the Ipad, we suggest that Ipads are not to be used to solve this part of the problem. We do however suggest that Ipads might be used as a complement to other means of audio and video conference, in order to show specific things to other players, such as die rolls. This requires the use of Ipad 2 devices though, as the first Ipad version was not equipped with any cameras.

The application, named Tabula Imaginarium, contains features for the synchronization of diagrammatical signs between Ipads, with the main focus on maps. It can be used as a whiteboard, drawing free-hand or lines that automatically snap to a grid net. It can be used to draw textures unto square grid tiles that are separated by this grid net. One can also place objects from an object library unto the map, as well as placing movable tokens that depict characters or monsters. Generally, the application is geared towards drawing fast and on the fly rather than preparing ahead of time. Tests indicated that this application felt natural to use in the context of tabletop role-playing even when it was not used to aid a spatially separated player, and in the case where a player was spatially separated, it helped in immersing that player in the activity.

The different types of immersion that the application can clearly aid in upholding and improving are spatial and temporal to their nature, but aid can only go so far as the need for diagrammatical interpretation present in the scenario being played. Tactical combat scenarios generally need a lot of this type of interpretation, which means that we know that the application helps during these scenarios, but it could still help in other scenarios as well.

When it comes to social immersion aside from emulating face-to-face interaction, we do suggest that the Ipad can be used for this, although not by using its keyboard function, as actually writing text by the use of finger-drawing is deemed as a preferable means of communication. Any other possible ways of controlling the attention of other participants or of communication that does not interrupt the flow of the game might be useful. Our suggestions are subtle visual feedback from the application that draws attention to it.

10. References

Adobe.com, 2012, Adobe Flash, <http://www.adobe.com/flashplatform/> [Accessed 26 February 2012].

Ambitioussoftware.com. 2011. Dungeon Mapp – Map Creation Tool for RPG's. <http://www.ambitioussoftware.com> [Accessed 2 January 2012].

Apple.com, 2012a, iOS, < http://www.apple.com/ios/ > [Accessed 26 February 2012].

Apple.com, 2012b, iPad, < http://www.apple.com/ios/ > [Accessed 26 February 2012].

Apple.com, 2012c, Objective-C, <https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html> [Accessed 26 February 2012].

Apple.com, 2012d, Xcode, < https://developer.apple.com/technologies/tools/> [Accessed 26 February 2012].

Apple.com, 2012e, Mac OS X, < http://www.apple.com/macosx/> [Accessed 26 February 2012].

Apple.com, 2012f, Bonjour, <https://developer.apple.com/opensource/> [Accessed 26 February 2012].

Apple.com, 2012g, Bonjour, <http://www.apple.com/iphone/built-in-apps/app-store.html> [Accessed 26 February 2012].

AsyncSocket, 2012, AsyncSocket, <https://github.com/robbiehanson/CocoaAsyncSocket [Accessed 26 February 2012].

Bergström, K., Jonsson, S., Björk, S. 2010. Undercurrents – A Computer-Based Gameplay Tool to Support Tabletop Roleplaying. In: Nordic DiGRA, 2010. *Experiencing Games: Games, Play, and Players*. Stockholm, Sweden August 16-17 2010. Gothenburg: Sweden.

Bowman, S. L. 2010. *The Functions of Role-Playing Games*. Jefferson: McFarland & Company.

Call of Cthulhu. 2005. *Call of Cthulhu – Horror Roleplaying, Sixth Edition*. Hayward: Chaosium Publication.

Cocos2d.com, 2010, Cocos2d for iPhone, < http://www.cocos2d-iphone.org/> [Accessed 26 February 2012].

Community.wizards.com Forum. 2009. Forum thread: The Best VTT. http://community.wizards.com/go/thread/view/75882/20168585/The_Best_VTT> [Accessed 2 January 2012].

Cover, J. G. 2010. *The Creation of Narrative in Tabletop Role-Playing Games*. Jefferson: McFarland & Company.

D20Pro.com. 2011. d20Pro. <http://www.d20pro.com> [Accessed 2 January 2012].

Dark Heresy. 2008. *Dark Heresy – Core Rulebook*. Nottingham: BL Publishing.

Drakar och Demoner. 1991. *Drakar och Demoner – 1. Rollpersonen*. Äventyrsspel.

Dropbox.com, 2012, Dropbox for iPad, <www.dropbox.com/ipad> [Accessed 26 February 2012].

Dungeon Master's Guide. 2000. *Dungeons and Dragons: Dungeon Master's Guide – Core Rulebook II*. Wizards of the Coast.

Edwards, R. 2001. GNS and Other Matters of Role-playing Threory. <http://www.indie-rpgs.com/articles/1> [Accessed 2 January 2012].

Enworld.org Forum. 2010. Forum thread: Virtual Table Tops – Which for Pathfinder (Pros/Cons)? <http://www.enworld.org/forum/general-rpg-discussion/296491-virtual-table-tops-pathfinder-pros-cons.html> [Accessed 2 January 2012],

Eon Deluxe. 2002. *EON – Andra upplagan av fantasyrollspelet Eon.* Neogames.

Fantasygrounds.com. 2010. Fantasy Grounds – Virtual Tabletop for Pen & Paper Role-playing Games. <http://www.fantasygrounds.com> [Accessed 2 January 2012].

Fine, G. A. 1983. *Shared Fantasy*. Chicago: The University of Chicago Press.

Fröhlander, J., Hartelius, U. 2011. *Digital Tools Supporting Boardgames – Dungeons and Drawings on the Microsoft Surface*. Gothenburg: University of Gothenburg, Chalmers University of Technology.

Gemini. 1998. *Gemini – The Dark Fantasy Roleplaying Game*. Cell Games.

Itunes.apple.com. 2011. RPG Cartographer. <http://itunes.apple.com/us/app/rpg-cartographer/id375814023?mt=8> [Accessed 2 January 2012].

Loponen, M., Montola, M. 2004. A Semiotic View on Diegesis Construction. In: M. Montola and J. Stenros, eds. 2004. *Beyond Role and Play - Tools, Toys and Theory for Harnessing the Imagination*. Helsinki: Ropecon ry. 2004.

MacKay, D. 2001. *The Fantasy Role-Playing Game*. Jefferson: McFarland & Company.

Microsoft.com, 2011, Microsoft Surface, <www.microsoft.com/surface/> [Accessed 26 February 2012].

Mutant Chronicles. 1997. *Mutant Chronicles: Techno-fantasy Rollspelet – Andra Utgåvan*. Target Games.

Pathfinder Roleplaying Game. 2009. *Pathfinder Roleplaying Game – Core Rulebook*. Redmond: Paizo Publishing.

RazeWare.com. 2011. RazeWare – iPhone and iPad Apps for Gamers. <http://www.razeware.com/battle-map/> [Accessed 2 January 2012].

RPG Virtual Tabletop, 2012. Wiki for Information on Virtual Tabletop Applications. <http://rpgvirtualtabletop.wikidot.com/> [Accessed 2 January 2012].

RPTools.net. 2011. RPTools – For the RP Games You Play. <http://www.rptools.net> [Accessed: 2 January 2012].

Skype.com, 2003, Skype, <www.skype.com> [Accessed 26 February 2012].

Somerville, I. 2004. *Software Engineering 7*. Edinburgh: Addison-Wesley Publishers Limited.

Star Wars Roleplaying Game. 2007. *Star Wars Roleplaying Game – Saga Edition Core Rulebook*. Wizards of the Coast.

SurfaceScapes. 2010. Student project at Carnegie Mellon University, Pittsburgh. <http://www.etc.emu.edu/projects/surfacescapes/> [Accessed 2 January 2012].

Tresca, M. J. 2011. *The Evolution of Fantasy Role-Playing Games*. Jefferson: McFarland & Company.

Vampire: The Masquerade 1998. *Vampire: The Masquerade*. White Wolf.

Warhammer Fantasy Role-play. 1995. *Warhammer Fantasy Role-play – A Grim World of Perilous Adventure*. London: Hogshead Publishing Ltd.

Appendix I.

Ipad applications:	Cartographer	Dungeon Mapp	Battle Map
Drawing	No	No	No
Map tools	Yes	Yes	Yes
Token packages	Yes	Yes	Yes
Object packages	Yes	Yes	Yes
Tile packages	Yes	Yes	Yes
Animation on Objects	No	Yes	Yes
Dice roller	No	Yes	Yes
Token information	No	Yes	No
Initiative Tracker	No	Yes	No
Grid	Square/Hex	Square/hex	Square/hex
Fog of War	No	No	Yes
Email	No	Yes	Yes
VGA	No	No	Yes
Network	No	No	No
Price	$10.99	$6.99 - Free version	$29.99

Appendix II.

PC applications:	Maptools	Fantasy Grounds II	d20Pro
Drawing	Yes	Yes	No
Dice Roller	Yes (Text)	Yes (3D)	Yes (Text)
Map tools	Yes	Yes	Yes
Fog of War	Yes	Yes	Yes
Line of Sight	Yes	No	No
Token Creator	Yes	No	Yes
Token information	Yes	Yes	Yes
Character Sheet	No	Yes	Yes
Macro and scripting	Both	Scripting	No
Network	Yes	Yes	Yes
External screen	No	Yes	Yes
Chat	Yes	Yes	Yes
Price	Free	$39.95	$30.00