UNIVERSITY OF GOTHENBURG

Digital Tools Supporting Boardgames

Dungeons and Drawings on the Microsoft Surface

Master of Science Thesis in Computer Science

JOHAN FRÖHLANDER

ULF HARTELIUS

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, July 2011

Digital Tools Supporting Boardgames
Dungeons and Drawings on the Microsoft Surface

Johan Fröhlander
Ulf Hartelius

Examiner: Staffan Björk

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover:
The cover picture is the default Microsoft Surface application icon, overlain with some boardgame paraphernalia. The picture is courtesy of Magnus and Viktor Nyström.

Department of Computer Science and Engineering
Göteborg, Sweden, July 2011

Abstract

In this report we present *Tisch*, a computer-based system with a tangible interface to support tabletop gaming in general and tabletop role-playing games and small-scale wargaming in particular. Based on the ideas of tangible and tabletop interfaces from computer science *Tisch* was developed to explore how technology can support tabletop gaming activities without becoming a hindrance or removing the players' agency over the rules. *Tisch* is designed to allow players to behave in their accustomed manner, including having unrestricted social interactions and using whatever rules they wish to whatever degree they wish, while harnessing the advantages computers offer through both offloading unwanted tasks and enhancing immersion. The report describes tasks common for tabletop games and how *Tisch* can support these, the underlying design rationale, the development process, the testing performed, and discusses possible expansions and suggestions for similar systems. Finally, it aims to give a clear picture of the current state of the system by the end of the project and relevant information for anyone wishing to continue working with or making use of *Tisch*.

Table of Contents

Introduction

Tabletop gaming has long been a popular pastime among all age groups, genders and classes, be it games based on skill, as *Chess* and *Go*, or luck, as *Yahtzee* and *Pass the Pigs*. As tabletop gaming has evolved the social aspects have gained increased attention, laying the way for such features as trading and negotiation, while also providing additional complexity through charts and advanced systems.

Wargames and role-playing games are among the most complex game types, with respect to rules and amount of game elements (as evidenced by the thickness of the rule books), with the latter often placing heavy emphasis on social interaction in addition to its rules.

Given the abilities of computers to keep track of rules and large amounts of information, it would seem that these games would be easiest to play as computer-based games. However, while comparing tabletop games with digital games Mandryk et al[1] notes that tabletop games provide *House Rules* and, more importantly, are "present to facilitate interactions between people, not interactions with the game"; aspects typically not present in digital games. This has not hindered tabletop games from successfully incorporating digital elements; examples include *Space Alert*[2], where the game progresses with the aid of a recording, *King Arthur*[3], where an Excalibur-holding artifact gives verbal instructions and keeps track of score, and *Dungeons & Dragons*[4] (described below) where different digital tools have been released to avoid rote tasks and supplement the game itself. More research should be done in order to establish what other kinds of semi-digital gaming platforms there are and how well they fit into the activity of playing games.

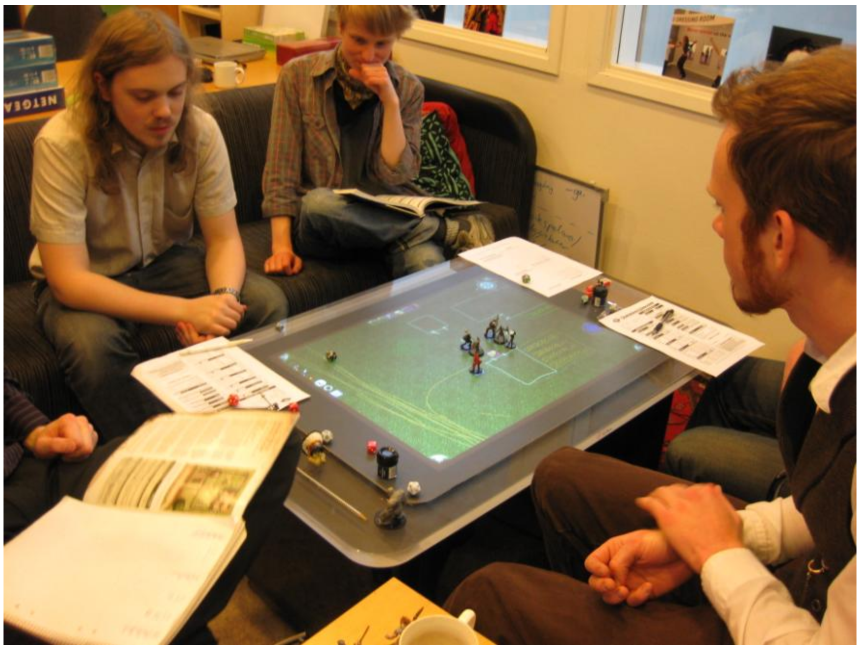

Figure 1: Using Tisch to play Dungeons & Dragons.

[1]Mandryk, R., Maranan, D. & Inkpen, K. (2002). *False Prophets: Exploring Hybrid Board/Video Games*. CHI, Minneapolis, Minnesota, USA.

[2]Chvátil, V. (2008). *Space Alert* [Board game].

[3]Knizia, R. (2003). *King Arthur* [Board game].

[4]Wizards of the Coast (2008) *Dungeons and Dragons Player's Handbook: Roleplaying Game Core Rules* [Tabletop role-playing games, fourth edition]. Written by Heinsoo, R., Collins, A. & Wyatt, J. Dungeons and Dragons originally published 1974.

Purpose

The primary purpose of the project is being a Master's thesis project in Computer Science, directed towards Interaction Design, with the aim to develop a map-making tool for the *Microsoft Surface*[5]. The project is based on this first basic aim and further goals have evolved from there.

With *Tisch* we explore the realms in-between digital and analog gaming. By moving traditional tabletop games to an interactive tabletop we can research in what ways this affects the game and the game experience. More in-depth delving give examples of specific elements that are suitable for digitalization and can heighten the game experience and the immersion.

Based on the ideas of tangible and tabletop interfaces from computer science, the project aims to explore how technology can support tabletop activities without becoming a hindrance to the activity or removing the players' agency over the rules. The technological development will also be in focus and evaluated to see in what ways this kind of tool can be created.

Question

How should one design a generic tool for an interactive tabletop for use with tabletop gaming to enhance the gaming experience without putting restrictions on rule liberty or in other ways hinder the activity?

Method

We planned to research what kind of functionality is necessary for a tool such as *Tisch* and how well it suits the platform: an interactive tabletop. We will to make use of the *Surface's* own Core API and, as those are its supported libraries, XNA Framework and, if deemed necessary, to a lesser extent Windows Presentation Foundation (WPF).

Throughout the project we intended to develop various features to be used in a variety of tabletop games and study how they affect the gameplay of the game and the social interaction amongst the players. The different features will focus on different aspects of the game and take form of various types of aid, be it relieving of tedious tasks, keeping track of game information or merely immersive. The design of the features will be based on own experiences and discussions with players initially, and then evaluated through testing and more discussions in order to determine which features fit the design of the tool. The design will initially also be affected by earlier research which will be read up on early in the project in the forms of research articles and books on related areas.

Goal

The project aimed to produce a dynamic map-making tool for *Surface* with a focus on tabletop role-playing games. It should not be limited to this, but also be usable for a variety of different applications which make use of maps and figures, such as crisis management and wargames. Some basic features that are necessary are *Drawing* and support for *Tokens* interacting with a map. Aside from these features we also thought about implementing some additional functionalities specifically targeting tabletop role-playing games. Features would not target game specific rules, but would rather be used as tools and guides. The set of features in the end of the project, and their respective implementations, will be based on aspects discovered throughout the execution of the project and will be motivated by research, studies, and testing.

The system should be able to run locally on a single *Surface*, but it could be extended to a server/client structure. We also planned investigate the possibility of implementing some kind of

[5]Microsoft (2008). *Microsoft Surface* [Interactive tabletop]. http://www.microsoft.com/surface/ (Visited 2011-05-11).

Undercurrent[6] system. Some effort would also be put into investigating porting to other platforms by looking up what libraries and tools are necessary for these other platforms. Based on this we intended to evaluate the difficulty of porting the system.

In the end, probably outside the scope of this project, the aim was that *Tisch* should be cross-compatible with a variety of platforms, each running as either server or client. These latter goals, however, were secondary to us and would not be taken into much account for the planning, but were rather thought of as possible extensions and future work.

Style

Whenever this report refers to games, features in *Tisch*, or certain prominent and reoccurring elements, these will be written italicized and with the first letter(s) capitalized; e.g. *House Rules*. These will be either described in the report or referenced.

Various references to games or other texts will be given as a foot note the first time the work is referred to in each chapter, with the exception of games we have bluntly assumed are common knowledge. A complete list of referred games and texts is included at the end of the report as well.

Outline

Here follows a short description of the order and contents of the various chapters. The chapters were put in the following order to early present the result and the motivation behind it, and follow up with more detailed descriptions of the working process and the tool itself.

Background and Research covers related work and research which lie as foundation for our work and which we have based much of our work on, whereas the Tisch chapter presents what it resulted in. Motivations for some choices made while developing Tisch based on tests are presented in the following chapter on User Testing and Evaluation. Process provides a more detailed description of the work done and choices and iterations along the way. Technical Specs describes the underlying system for the tool that was developed and information about the technology used. Finally, Discussion covers where we currently stand, what can be improved and more about the results we ended up with.

[6] Björk, S., Bergström, K. & Jonsson, S. (2010) *Undercurrents - A Computer-Based Gameplay Tool to Support Tabletop Roleplaying*, in proceedings of Nordic DiGRA 2010.

Background and Research

In order to successfully carry out a research project, but also lessen one's own workload, delving into the area is necessary to see what has already been discussed and what conclusions have been reached. A lot of research has been done in the area of boardgames and digital counter-parts as well as gaming and social interaction, so there is a lot of work already done that can be used. One can hardly absorb it all, but the project has been based on several works which were found important and relevant to the design and execution.

Related Research

Given the price of a *Surface* it may seem extravagant to develop a support system for wargames and tabletop role-playing games on it. However, the development of *Tisch* is not intended for immediate commercial resale but rather a proof of concept which follows Weiser's *Ubiquitous Computing*[7] path in exploring a future where computational power is present everywhere; a decision taken not because we do not believe that interactive tabletops will never see the light of everyday households but because that setting and the public setting, which was aimed for, are so different that we chose to only explore one of them. The tabs, pads and boards developed by Weiser and his colleagues are direct predecessors to today's smartphones and tablet computers – while this is not necessarily a clear indication that people will have "smart" walls or tables soon it shows that one cannot disregard the idea either. For a historical overview of research on tabletop interaction and developed products, see Müller-Tomfelde[8].

Introducing technology to mediate any activity is likely to change how it is experienced. Weiser et al[9] explored this issue when researching ubiquitous computing, resulting in the introduction of the concept of *Calm Technology* as the design goal of having technology as peripherals that support rather than direct activities. A related design concept is *Social Weight*, introduced by Toney et al[10] as "the measure of the degradation of social interaction that occurs between the user and other people caused by the use of that item of technology." *Tisch* shares both these goals since much of the experience of playing games, especially role-playing games, lie in the social interaction. It should also have *Social Adaptability*, coined by Eriksson et al[11], i.e. be able to handle different amounts of attention without negatively affecting the experience.

It is also important to recognize that the players' actions are not motivated solely by rational thinking. For example, rolling dice can be a way for to, in a sense, trick players into believing they have a tad more influence over events that they actually do or to experience luck (see Fine[12] for further descriptions). Rather than to replace these with buttons it was early decided that *Tisch* would promote rolling of actual dice as has previously been done in *Wizard's Apprentice*[13] and the *Surface* version of *The Settlers of Catan*[14]. Digitalized dice rolling would preferably retain shaking of the

[7]Weiser, M. (1991). *The Computer for the Twenty-First Century*, Scientific American, pp. 94-10, September 1991.

[8]Müller-Tomfelde, C. & Fjeld, M. (2010). *Introduction: A Short History of Tabletop Research, Technologies, and Products*. In Müller-Tomfelde, C. (ed.) Tabletops - Horizontal Interactive Displays. Human Computer Interaction Series, Springer Verlag, pp. 1-24.

[9]Weiser, M. & Brown, J. S. (1996). *The Coming Age of Calm Technology*. Available from http://www.johnseelybrown.com/calmtech.pdf (Visited 2011-05-03).

[10]Toney, A., Mulley, B., Thomas, B. H. & Piekarski, W. (2002). *Minimal Social Weight User Interactions for Wearable Computers in Business Suits*, in conference proceedings of IEEE International Symposium on Wearable Computers 2002.

[11]Eriksson, D., Peitx, J. & Björk, S. (2005). *Socially Adaptable Games*, Lightning round presentation at Changing Views: Worlds in Play, DiGRA conference 2005.

[12]Fine, G. (1983). *Shared Fantasy: Role-Playing Games as Social Worlds*. The University of Chicago Press, Chicago.

[13]Peitz, J., Björk, S. & Jäppinen, A. (2006). Wizard's *Apprentice - gameplay-oriented design of a computer-augmented board game*. Paper at Advancements in Computer Entertainment, Hollywood, USA, 2006.

[14]Vectorform Games (2011). *The Settlers of Catan* [Microsoft Surface version of The Settlers of Catan, original design by Teuber, K.].

device, as has been done in several *iPhone* applications, e.g. *Mach Dice*, *MotionX Dice*, *The Dice*, and the game *M.I.G.*[15]. Careful consideration must also be taken regarding what kind of tasks to relieve the players of, described in detail below as *Excise*. A game element such as dice rolling is very convertible to a digital medium, but it is not necessarily an unwanted task. Many players consider the physical action and the thrill of the randomness as the dice soar through the air an enjoyable part of the game, especially in games with many dice, such as numerous wargames, or if they corporate a major part of the gameplay, as is the case of many role-playing games. Related is also the work of Jamil et al[16,17] on social interaction around interactive tables which has influenced *Tisch* through the findings that direct touch interfaces support reflection and broader discussions, rather than discussing the interaction or other distracting matters, nearly as well as non-digital tables.

Related Systems and Tools

There are several examples of computer support for specific board games without making them virtual. *False Prophets*[18] is an early example noteable for supporting simultaneous player actions. *KnightMage*[19] used a custom built software and hardware system to provide a tabletop-like experience of playing a role-playing game where auxiliary devices were used by game masters to place enemies and by players to handle inventories. The underlying system was also used to provide a version of *Monopoly*, but given the strictness of the implementations it cannot be said to be a general purpose system in the same way as *Tisch*. *Wizard's Apprentice* made use of a custom-based sensor-enriched game board to support a caretaker to only intermittently take part in the game. Mazalek et al[20] developed the *Tviews Table Role-Playing Game* as a proof of concept implementation of how *Dungeons & Dragons* can be supported through tabletop interaction with a computer system. *ReacTable Role Gaming*[21] is similar but provides a dungeon editing tool that allows drawing comparable with that of *Tisch* but more strictly suited for rooms and dungeons; it also supports *Fog of War* (described under *Fog of War*, p.16).

As *Tisch*, *SurfaceScapes*[22] used the *Surface* and tagged miniatures to create a proof of concept aid for tabletop role-playing, but did this beginning with exclusively supporting *Dungeons & Dragons*[23]. The previously mentioned *The Settlers of Catan* game was also an adaption of a tabletop board game for use on the *Surface*, and in doing so showed interesting design solutions where private information was provided through using tagged tinted prisms.

While all of these do support many of the characteristics of tabletop gaming, locking them to one game limits their usefulness. Further, most of them do enforce game rules, partial exceptions include *Wizard's Apprentice* and *KnightMage*, which breaks one of the design assumptions for *Tisch*. While both *Wizard's Apprentice* and Vectorform Games' *The Settlers of Catan* support the rolling of physical dice that the games can recognize, a possibly wanted feature for *Tisch*, the particular solutions used

[15] Compete Now AB (2009), *M.I.G. - Mobile Intelligence Games* [Quiz Game, iPhone version]. Originally published 2001.

[16] Jamil, I., Alexander, J., Subramanian, S., Barnes, S. (2010). *Talking Teenagers and Tables: Communication Styles of Teenagers and Interactive and Non-Interactive Tables*. University of Bristol, UK.

[17] Jamil, I., O'Hara, K., Perry, M., Karnik, A., Subamanian, S. (2011). *The Effects of Interaction Techniques on Talk Patterns in Collaborative Peer Learning around Interactive Tables*. CHI, Vancouver, BC, Canada.

[18] Mandryk, R., Maranan, D. & Inkpen, K. (2002). *False Prophets: Exploring Hybrid Board/Video Games*. CHI, Minneapolis, Minnesota, USA.

[19] Magerkurth, C., Memisoglu, M., Engelke, T., & Streitz, N. (2004). *Towards the next generation of tabletop gaming experiences*. In proceedings of Graphics Interface '04, ACM International Conference Proceeding Series, Vol. 62, Canadian Human-Computer Communications Society, 2004, ISBN 1-56881-227-2, pp. 73-80.

[20] Mazalek, A., Mironer, B., O'Rear, E., & van Devender, D. (2007). *The Tviews Table Role-Playing Game*. Synaesthetic Media Lab, GVU Center, Georgia Institute of Technology, USA.

[21] ReacTable Role Gaming (2008). Student project by Viladamot, R. as part of PFC, Universitat Pompeu Fabra, Barcelona. http://ramonviladomat.com/rrg.html (Visited 2011-05-11).

[22] SurfaceScapes (2010). Student project at Carnegie Mellon University, Pittsburgh. http://www.etc.cmu.edu/projects/surfacescapes/ (Visited 2011-05-11).

[23] Wizards of the Coast (2008) *Dungeons and Dragons Player's Handbook: Roleplaying Game Core Rules* [Tabletop role-playing games, fourth edition]. Written by Heinsoo, R., Collins, A. & Wyatt, J. Dungeons and Dragons originally published 1974.

are infeasible to be applied to all tabletop games (the former using custom-shaped dice and the latter being only practical for 6-sided dice).

Also relevant are computer-based systems supporting general role-playing or wargaming. Even if these do not consider the social and interface aspects of tabletop gaming, the functionality they exhibit can be appropriate there as well. *Campaign Cartographer*[24] allows the construction of maps on a far greater level of detail than *Tisch*, including the possibility to fractalize maps for more realistic looks, but since it uses a CAD-like interface it does not function well as a sketching tool on a touch-based interface. *Virtual Advanced Squad Leader*[25] supports the handling of maps and tokens for *Advanced Squad Leader* but without enforcing any rules; yet it can provide dice rolls, record moves made, and make some *Line of Sight* calculations. The emphasis of that system is to support the virtual tokens, done through combinations of mouse clicks and key commands, making it non-trivial to translate to a touch-based tabletop interface. A third example is *Vassal*[26], a very generic online engine for all kinds of boardgames and card games, with little to no rule implementations.

RPTools[27] is a computer software suite for creating and managing tabletop role-playing games, where the main application is *MapTool*, the campaign management software. *MapTool* and *Tisch* have similar design aims, but these are carried out very differently. *MapTool* is designed for use on a computer and has an interface designed for mouse and keyboard. Much of the usage is also intended to be carried out beforehand. *Tisch* aims to open up for more spontaneous sessions where players and game masters can improvise maps and scenarios. *YouTube* videos shows *MapTool* running on touch-based tabletop interfaces but no documentation exists of how well the traditional GUI works when used in this fashion, especially regarding simultaneous use by multiple players.

There is also the example of computer-based aids and tools available for the fourth edition of *Dungeons & Dragons*. Most of these consist of tools for generating player characters, monsters, and abilities, and rule lookups. Features of these kinds have been discussed, along with features for managing character information and abilities, and are far from impossible or even difficult to add to the system. The problem is that it is too tightly tied to a specific game. It is difficult to do a general ability creation/management feature for *Tisch*, but fairly easy to do a *Dungeons & Dragons* specific one.

Wargaming and Tabletop Role-Playing

Although board games have existed as cultural objects for millennia, Woods[28] points out wargaming in the 1950s as the earliest form of hobby gaming to appear. The first modern games attributed as wargames are *Kriegspiel* from 1812, see von Hilgers[29], and *Little Wars*[30] from 1913. These were the first games explicitly designed as representing military struggles. Examples from the 1980s include *Advanced Squad Leader*[31], *Rommel in the Desert*[32], and the miniature-based *Warhammer 40,000*[33] and, most recently, *Confrontation: The Age of the Rag'narok*[34]. For the development of *Tisch* and its functionality we focus primarily on miniature-based wargames, which is what is meant in this report when the term wargames is used, but much applies to other types of

[24] ProFantasy Software Ltd. (2006). *Campaign Cartographer*.

[25] Kinney, R. (2002). *Virtual Advanced Squad Leader*.

[26] The Vassal Team (2010). *Vassal* [Generic online boardgame engine].

[27] RPTools (2006). http://www.rptools.net/ (Visited 2011-05-11).

[28] Woods, S. (2010). *Convivial Conflicts: The Form, Culture and Play of Modern European Strategy Games*. PhD thesis, School of Media, Culture and Creative Arts, Curting University of Technology.

[29] von Hilgers, P. (2000). *Eine Anleitung zur Anleitung. Das taktische Kriegsspiel 1812-1824*, in Board Games Studies no.3, pp. 59-77.

[30] Well, H.G. (2004). *Little Wars (A Game for Boys from twelve years of age to one hundred and fifty and for the more intelligent sort of girl who likes boys' games and books)*. Kessinger Publishing. Originally published 1913.

[31] Multi-Man Publishing (2001). *Advanced Squad Leader Rule Book* [Tactical-level board wargame, second edition]. Original design by Greenwood, D.

[32] Columbia Games (2004). *Rommel in the Desert* [Wargame, second edition]. Originally published 1982. Designed by Desinque, C.

[33] Games Workshop (2008). *Warhammer 40,000* [Tabletop miniature wargame, fifth edition]. Originally published 1987.

[34] Rackham (2007). *Confrontation: The Age of the Rag'narok* [Tabletop miniature wargame, fourth edition]. Originally published 1996.

wargames as well. Role-playing games developed from these during the 1970s (see Fine, Mackay[35]) with the focus on playing individual characters that were kept between game instances, as opposed to entire armies and stand-alone gaming sessions. The first commercial success was *Dungeons & Dragons*; other role-playing games include *World of Darkness*[36] and *GURPS*[37].

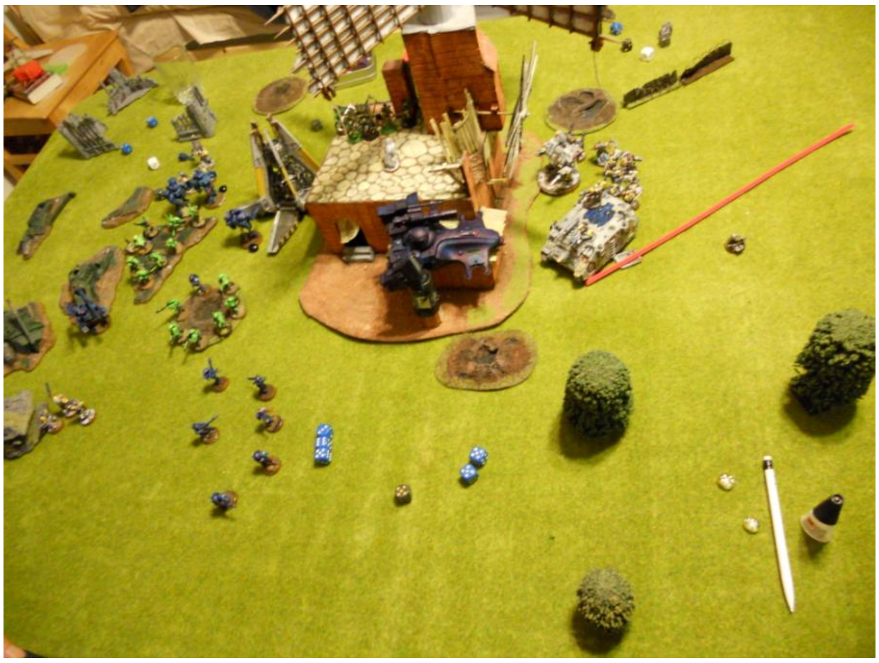

Figure 2: An ongoing Warhammer 40,000 game.

Characteristic for wargames is the use of game worlds in which players control a large number of miniatures representing individual soldiers, squads and vehicles, and moving and engaging these in combat against each other. Frequent design elements include asymmetrical starting conditions and asymmetric goals, terrain making affecting the troops or the rules in some way, use of dice, templates for area effects such as explosive weaponry or unit abilities, and distance measuring rules. A distinctive trait is that wargames often use height differences, with the players constructing battlefields out of various terrain pieces, from trees to buildings, with which troops can interact and break off a clear view to other troops, either completely or partially, which usually affects ranged combat. While terrain does not need to be used and digital counterparts can be implemented instead it is still a very difficult, yet important, element of the games to adapt since it can both be hard to visualize and interactive with using a top-down perspective, especially when it comes to having units on different floors.

Like wargames the first role-playing games focused upon combat, but here pitching players, each controlling a single player character, against enemies controlled by a so called game master. The game master, also known as a dungeon master, a facilitator, or a referee, is responsible for preparing and running the scenario, controlling any characters the players encounter, and maintaining a pace and style comfortable for all participants, the game master included. Players engage in exploration of dungeons and other dangerous locales to find treasure and improve their characters' abilities through experience. At the same time players carry on a great deal of conversation and other non-combat related actions while taking the role of their character, much the same as actors in improvisational theatre. Many later games have supplemented or wholly replaced the dungeon crawling with the exploration of social dilemmas and psychological issues. Regardless of whether done through combating monsters which are diegetically real or merely in one's mind, the games revolve around

[35]Mackay, D. (2001). *The Fantasy Role-Playing Game: A New Performing Art*. McFarland & Company.

[36]White Wolf (2004). *The World of Darkness: Storytelling System Rulebook* [Tabletop role-playing game, second edition]. Written by Bridges, B., Chillot, R., Cliffe, K. & Lee, M.

[37]Steve Jackson Games (2004). *GURPS Basic Set* [Tabletop role-playing games, fourth edition]. Written by Jackson, S., Punch P. & Pulver, D. Edited by Hackard, A. & Jackson, S. GURPS originally published 1986.

character development; be it by improving one's skills or by maturing enough to fight for a greater cause. This is similar to much popular culture, especially that which targets children and young adults.

Challenges to the Activities

Role-playing games and wargames are often complex, both relating to the amount of rules and the number of gameplay elements. This unavoidably creates plenty of challenges to overcome with and around the activity. Below we describe some of these to provide a basis for discussing how computers can help support the activity.

House Rules

Many definitions of games point to the importance of rules, e.g. Abt[38] and Juul[39], and this may seem to argue that computers are candidates for supporting tabletop gaming through being impartial facilitators of the rules. However, Sjöblom[40] and Salen et al[41] have pointed out that the actual rules in use can differ from those provided by the designers, coming alive of their own and evolving as *House Rules*; a notion we agree with. While these can sometimes be explicitly agreed upon beforehand, situations arise where the rules, or how to use them, need to be improvised or interpreted as the game progresses. In role-playing games, the game masters' goals are to create engaging experiences through pre-planned narrations or interesting environments; goals that can be toppled by an unfortunate dice roll which would, e.g., cause all player characters to fall into a bottomless hole in which they will be forced to spend eternity. For this reason it can be preferable for all the participants that game masters act subjectively, bending the actual rules or even cheating (see previously mentioned Fine for descriptions of this type of behavior). Further, in both role-playing games and wargames it is common to let novice players receive handicaps or reprieves. Both types of games can also require improvisation regarding rules since the presence of detailed game worlds encourage players to imagine context-specific actions, while at the same time game masters can ignore rules to speed up role-playing games. All these aspects work against having computers as judges and rule enforcers. Even in purely computer-based games that revolve around a strict rule system, many games implement cheat codes which players can use to open up some degree of freedom in how they play the game.

Creating Game Worlds

Game worlds often need to be explicitly visualized, usually done through the use of maps and the moving of miniatures or other representations of the players' characters. Although pre-made maps can be used, these may need to be modified both before and during gameplay due to, for instance, a role-playing party taking a path not planned out by the game master. Since computers have long been used to support the creation of visualizations, and since this supports rather than restricts the need for improvisation argued for above, it seems that support for sketching and drawing is a viable candidate for supporting players in *Creating Game Worlds*. We reasoned that being able to draw anything, without any restrictions on how to do it, would be the easiest way to allow users to create anything for any kind of game or activity, even if it is not always the optimal interaction. There was some discussion regarding using pre-made maps, either made with a different program or based on plain sketches, but it was disregarded as too circumstantial a solution.

[38]Abt, C.C. (1970). *Serious Games*. New York: Viking Press.

[39]Juul, J. (2005). *Half-Real: Video Games between Real Rules and Fictional Worlds*. The MIT Press, Cambridge, 2005.

[40]Sjöblom, B. (2008). *The Relevance of Rules: Negotiations and Accounts in Co-operative and Co-located Computer Gaming*. Proceedings of the [player] conference, IT University of Copenhagen 2008, August 26-29, 2008, pp. 335-378.

[41]Salen, C. & Zimmerman, E. (2004). *Rules of Play: Game Design Fundamentals*. The MIT Press.

Excise

Both role-playing games and wargames often use a large number of tangible artifacts, be it miniatures, dice, or character sheets. Handling them can take unnecessary amounts of time not specifically beneficial to the activity, e.g. counting the result of thrown dice or moving entire miniature squads. These tasks could be grouped as *Excise*, in that they are undesired work which bring little to no benefit to the actual activity.

However, digitalizing it is not trivial since the line between *Excise* and fun is vague and differs greatly from player to player. Some find the placing and moving of miniatures to be the greatest part of the game while others would much rather play without them entirely, as evident through the absence of miniatures in the design of many modern role-playing games, e.g. *World of Darkness* and *Eon*[42]. Other tasks which may or may not be regarded as *Excise* are finding entries in rule tables, looking up rules, measuring distances between game world objects, and keeping track of health and other attributes of player characters.

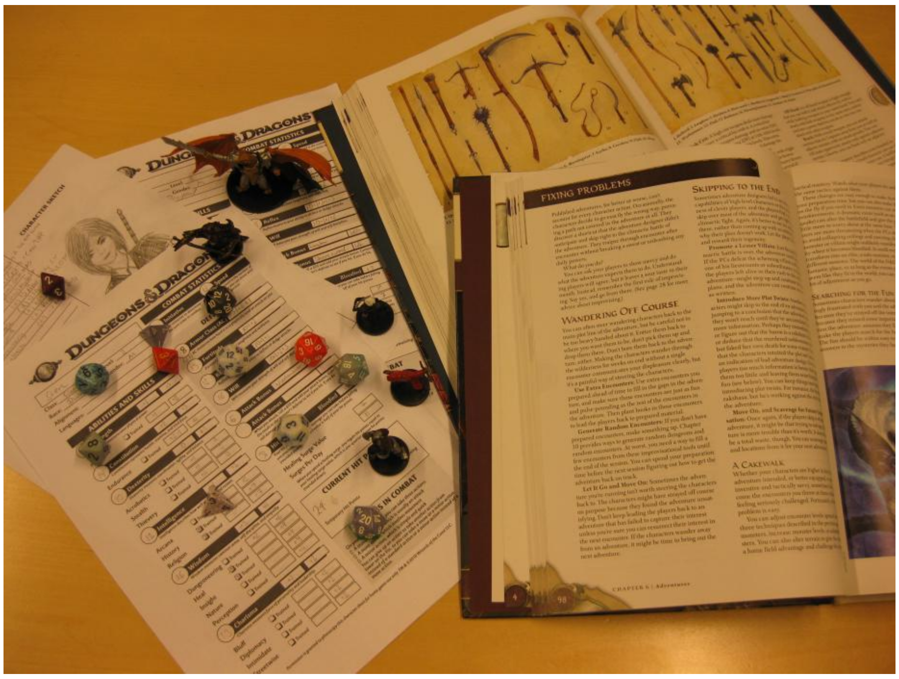

Figure 3: Some role-playing game props. Fun or Excise?

Storing Game States

All things must come to an end, so also game sessions. While gathering and keeping a few loose papers might be easy, to let play resume later, the more complex situations often occurring in combat can be more cumbersome to handle; this due to the specific locations of miniatures, their temporary attributes and similar kinds of data. While repositioning miniatures in their appropriate situation may take time, the real problem is keeping track of their positions between the game sessions; a kind of data handling computers excel at, especially if they already have support for handling the maps on which the games take place.

Design Choices

The tool that *Tisch* is meant to be is not just any tool for any platform. *Tisch* targets interactive tabletops which make for different considerations than a regular computer-based tool. First and

[42]Neogames (2004). *Eon* [Tabletop role-playing game, third edition]. Originally published 1996.

foremost, there is no mouse or keyboard (apart from an optional on-screen keyboard which can be summoned). Menus with lots of options and textboxes will only be slow and inconvenient on a tabletop touch platform, those kinds of interfaces much more suited for a desktop environment with a mouse and keyboard, and will especially become obtrusive and difficult to use with multiple users from various angles all wanting to participate simultaneously. The ability to work with detail and lots of options, at least not presented in the ways of classical interfaces, will not do *Tisch* much good.

Secondly, one should consider where a *Surface* or similar device is usually found, which is not in mom's basement. More common venues are instead expositions, lobbies, or other public places where people can gather around and interact with it. In such settings the users rarely wants advanced interaction or prolonged preparatory work, instead placing the emphasis on spontaneity and that all of them can be active participants at the same time. From this consideration follows that users must be able to easily create game worlds and make use of its creation from scratch, and then alter it in ways that could be demanded by a game session.

In the beginning it was believed that the most optimal tool would be one fundamentally different from what was just described; a tool more akin to the advanced editors found in computer games, with a plethora of menus, options, toolbars, and so on. Yet as it was quickly discovered that such interfaces were cumbersome on the *Surface*, making them more usable for a desktop computer, and would not be desirable or fitting for the *Surface*. In general, we believe that applications should be designed to fit their targeted platform.

Tisch

Tisch was designed for two primary purposes: to lessen the amount of time spent by the players on tasks which deviate from the actual game activity, bundled as *Excise*, and to heighten immersion by adding features which may be difficult or impossible to create in a purely analog setting.

As will be discussed in detail in Development, but which also should be mentioned here, is that *Tisch's* software architecture is centered on a core module with a number of stand-alone extensions built around it. Each extension is called a "feature" and is intended to supply *Tisch* with some sort of unique ability. While features can make use of one another, they shouldn't depend on anything but the core module or other features in the same package.

Design Goals

Based upon the observations made in Background and Research the design goals for *Tisch* were set to the following:

1. Be easy to use and not require technical proficiency.
2. Allow, but not require, preparations.
3. Allow *House Rules* and free interpretation of, and compliance to, rules.
4. Keep *Social Weight* as low as possible to avoid disrupting the social interaction.
5. Reduce or remove *Excise*.
6. Enhance the gaming activity through *Immersive* features.

The two first goals were inspired by other applications for the *Surface*, most of which are intended to be used in public places, conventions or other areas where users will have little to no time for preparations or setup, as described in Design Choices, as well as being able to function with any orientation. For users who wish to make preparations, and to suit more types of games, *Tisch* can be configured in great detail while running. It is also possible for users with programming experience to create their own features as plug-ins. An implicit goal born out of these first two goals is that *Tisch* could familiarize and introduce people to tabletop games in general and role-playing games in particular.

The third goal comes from the observations of Mandryk et al[43], Sjöblom[44], and the developers' own experiences of tabletop games. As discussed under Challenges to the Activities, allowing users to shape the experience on their own terms is important; especially with such a generic project as *Tisch* with the goal of supporting a plethora of games and other interactive experiences.

The fourth goal was not only influenced by the work of Toney et al[25], but also that of Montero et al[45] and Jamil et al[46,47]. While our process did not include any formal measurements regarding communication and social interaction, we made certain to make notes at every test occasion as to the shape and purpose of the present communication between the users; more specifically if they were talking about the actual experience or about the experience's presentation.

The two last goals were the initial motivators for the overall process but were intentionally left unspecific to allow user input into the design process. For the most part, features will be categorized

[43]Mandryk, R., Maranan, D. & Inkpen, K. (2002). *False Prophets: Exploring Hybrid Board/Video Games*. CHI, Minneapolis, Minnesota, USA.

[44]Sjöblom, B. (2008). *The Relevance of Rules: Negotiations and Accounts in Co-operative and Co-located Computer Gaming*. Proceedings of the [player] conference, IT University of Copenhagen 2008, August 26-29, 2008, pp. 335-378.

[45]Montero, C., Alexander, J., Marshall, M., Subramanian, S. (2010) *Would you do that? – Understanding Social Acceptance of Gestural Interfaces*. Mobile HCI, Lisbon, Portugal.

[46]Jamil, I., Alexander, J., Subramanian, S., Barnes, S. (2010). *Talking Teenagers and Tables: Communication Styles of Teenagers and Interactive and Non-Interactive Tables*. University of Bristol, UK.

[47]Jamil, I., O'Hara, K., Perry, M., Karnik, A., Subamanian, S. (2011). *The Effects of Interaction Techniques on Talk Patterns in Collaborative Peer Learning around Interactive Tables*. CHI, Vancouver, BC, Canada.

as either *Practical*, intended to reduce or remove *Excise*, or *Immersive*, enhancing the experience with the aid of digital tools.

Features will later be indicated with italics, e.g. *Drawing*.

Drawing

Tisch's most fundamental feature is to support sketching and it is the feature most incorporated as a part of the core module, motivated by our perceived importance of *Creating Game Worlds*. If no other feature takes interest in the contact input, the default behavior of *Tisch* is to act as a sketching tool, creating dots and lines wherever users place their fingers. Unlike many other sketching and painting tools, however, the lines are created come with some additional data which other features can make use of. Being able to draw spontaneously is also reminiscent of the original analog setting, with pens, paper and other tangible objects used completely at the users' discretion. It is thus expected in order to let *Tisch* be an example of *Calm Technology*[48].

Drawing, and the tools enhancing it following shortly, are the tools most targeted towards a multi-user environment with a group of users interacting simultaneously. The lines are made up of a series of points with textured lines drawn between each pair. There have been discussions regarding making use of curves instead, as there are several techniques for more effectively storing and rendering curves, but it was reasoned that it was more important that the drawings made by the users are as identical to their interaction as possible; curves run a risk of manipulating the drawings to an undesirable degree.

Tokens

Most games make use of tokens for representing the players' characters or units and, in the case of tabletop role-playing games, non-player characters and enemies handled by the game master. Normally, metal or plastic figures are used with appearances more or less related to the game's setting. With *Tisch*, though, users are encouraged to place *Surface* tags on the figures' bottom, allowing the application to link figures with virtual representations by entering a simple registration mode and stamping values and icons with the physical token. In addition, the assigned numeric values can be extended to keep track of game-related issues such as turn-order, health, movement allowance and vision range. With *Tokens*, the other features are able to know where the players' characters are and to configure the surroundings and the game accordingly; such as hiding those tokens which are out of the players' sight and displaying an aura around the current player's token. The link between tagged objects and icons allow them to be removed from the table for whatever reasons and having a visual reminder of their current game location.

It was decided that this feature was important enough to be the second feature to be incorporated as a part of the core module, as other features can access it more easily and make more extensive use of it if they can be sure that it is active.

[48]Weiser, M. & Brown, J. S. (1996). *The Coming Age of Calm Technology*. Available from http://www.johnseelybrown.com/calmtech.pdf (Visited 2011-05-03).

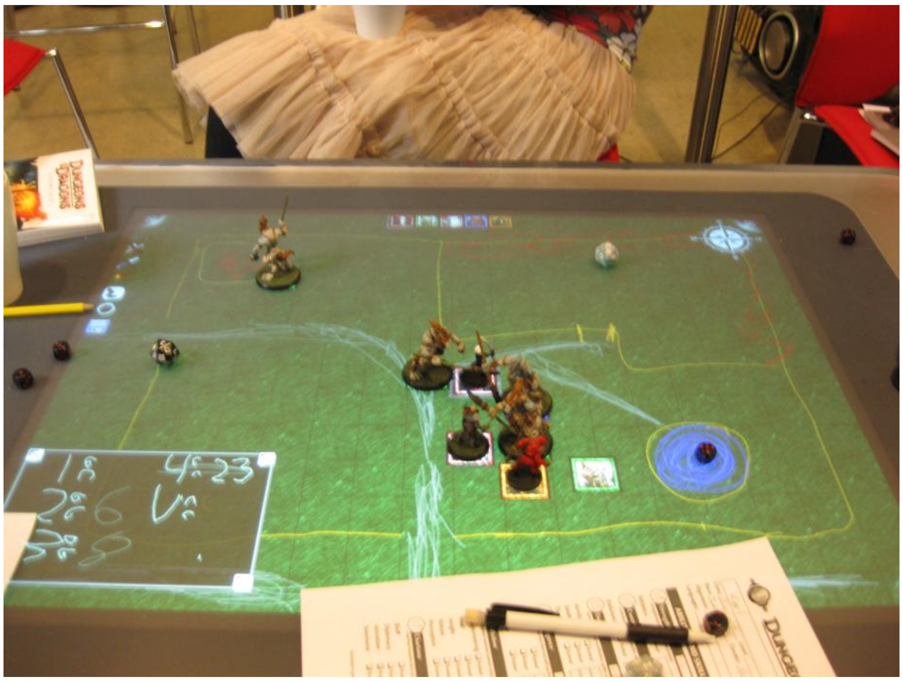
Figure 4: Using Tokens along with tangible props.

TagPrompter

The last core module feature ensures that any other features making use of tags and tagged objects can more easily assign tags to their specific purposes. Should a feature require a tag the *TagPrompter* holds off all contact input and automatically assigns the next tag to the desiring feature.

Practical Features

Drawing Tools

Users are able to *Erase* what they have drawn with a specifically tagged object, mimicking normal erasers. In fact, the object which was used for all but the first few tests was a commonplace eraser. Users can also activate a *Grid* feature of either squares or hexes to support games using tiles, with configurable sizes and, for squares, appearances (thin lines, thick lines and only crosses in the corners are possible). The *Grid* feature employs semantic zoom by automatically hiding if the user zooms out too much. To support further creative control users can select the color with which they draw by spinning a tagged *Palette* object, alter the width of the drawn lines, and make them snap to the present *Grid*. Having this functionality was seen as inherently important since *Tisch* is primarily focused on games which use maps as gameplay areas; *Drawing* and sketching the imagined surroundings in an unrestricted way for all participants was the most flexible feature identified.

Camera

Another of *Tisch's* most fundamental features is giving players control over the *Camera* by letting them pan, zoom and rotate the map. As opposed to *Drawing*, the *Camera* provides an important dissimilarity from both a lot of sketching applications and, more importantly, an analog setup; especially the zooming functionality which in practice makes *Tisch* a limited form of a *Zoomable User Interface*, e.g. Bederson et al 2004[49]. Testing showed that users appreciate having a *Camera* and they created new behaviors with the zooming functionality, such as zooming in and out in order to

[49] Bederson, B. B., Grosjean, J., & Meyer, J. (2004). *Toolkit Design for Interactive Structured Graphics*, IEEE Transactions on Software Engineering, 30 (8), pp. 535-546.

organize the activity as advocated by Bederson's interface paradigm. Users control the *Camera* through between one and three tagged objects, depending on if it is desirable to have the same tagged object control more than one aspect (panning, zooming and rotation).

Saving and Loading

It is a common but irksome issue that game sessions sometimes need to be ended before all the action taking place on a particular map is done. This, and the wish to re-use maps for other scenarios or to prepare them in advance of actual gameplay, requires that users can be *Storing Game States*. In *Tisch* this is done by associating a tagged object with the particular map and all its related data by toggling a save/load mode and placing the token on either the save or the load area. Each feature chooses individually what it wants to save and how it wants to load it; e.g. if tokens are used in a loaded map, their icons indicate where they last stood, allowing their tangible figures to be placed there in order to recreate the previous context. *Saving and Loading's* use of tagged objects can be seen as an example of *Tangible Bits*[50].

Playgrounds

Everything and anything performed in *Tisch* is done in a so-called *Playground*, a section of the screen which handles both input and all features. *Playgrounds* are based on the windows found in most graphical operating systems, with the initial underlying *Playground*, called the *Alfa Playground*, acting as a full-screened window or a desktop. In addition to the *Alfa Playground* users can create their own *Playground* by either clicking on an icon in the toolbar or holding down four fingers in a square, creating one with the bounds of the fingers. This new *Playground* can be moved and scaled at will and, while only containing the most basic features to start with, can superlatively easy be altered in run-time to behave in just as many ways as the *Alfa Playground*. As all *Playgrounds* have the same potential there is practically the same uses and limits as for the rest of *Tisch*. *Playgrounds* can also be tied to a tagged object, being displayed only when and where the tagged object is put down. The reasons for creating *Playgrounds* are primarily for users who want to take notes or if the users split up in different directions or do different things, with each user/group getting a segment of the total screen, possibly interacting with completely different feature sets.

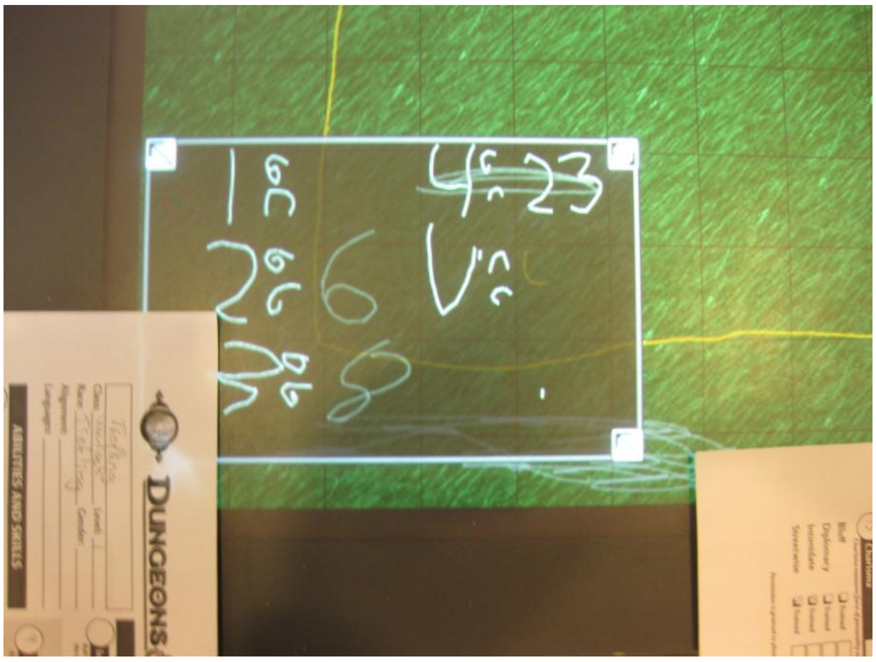

Figure 5: Example of Playground usage

[50]Ishii, H. (2008). *Tangible bits: beyond pixels*. In Proceedings of the 2nd international conference on Tangible and embedded interaction (TEI '08). ACM, New York, NY, USA.

Configurer

While one of the design goals was for users to be able to use *Tisch* without any preparations or configurations, retaining the option to do this is still important for sessions and events which allow it. The *Configurer* was created for this purpose. It takes all of the values from the other features which are saved and loaded to the application's settings file and allows users to interact with them using sliders or buttons; the method of interaction depending on if the value is countable or not. It is also possible to turn on and off features through the *Configurer*. In order to make the most use of the *Configurer* features must be prepared a little in advance by adding some meta-data, but as this meta-data is also used for the purposes of saving and loading the features' values to and from the settings file it was not seen as a major problem.

Line of Sight and Measuring

The detection of *Line of Sight* between one point and another, that is if the first point is able to see the other, is for some games incredibly important. *Line of Sight* also allows for specific checks between any on-screen points by entering an overlaying mode which takes finger input and examines whether the contact points can see one another. A temporary line is then drawn between the contact points, colored green or red depending on if there is line of sight or not, equipped with a *Measuring* tool; measuring the distance in centimeters, inches, or pixels.

Currently, the system distinguishes transparent and opaque drawings through the lines' color; e.g. by default white lines are opaque while all other colors are transparent, but users can at any time toggle which colors count as opaque. The *Line of Sight* feature is used extensively by some other features; *Tokens* can automatically hide any tokens which are not presently in the current player's view, and *Fog of War* (described later) keeps track of what parts of the world the players see using it. The various features uses the same calculations, which are provided by the core module, but it is the *Line of Sight* feature that keeps track of which colors that obscure vision.

RoundKeeper

As most games have players taking turns, it was natural that *Tisch* should include a feature managing this. The *RoundKeeper* can handle either *Tokens* or just numbered players, at one side of the screen displaying the current player and the order with which the others will follow. If *Tokens* are used, an aura pulsates around the current player's *Token* on the screen. Presently, the *RoundKeeper* is the only way of managing which *Token* is the current player and has only a simple round-robin structure. *Tokens* do not require the specific *RoundKeeper* in order to be regarded as the current player, it is merely the only tool currently available for assigning them that attribute. Some other features make use of this attribute; for instance *Tokens* may be visible only when the current player sees them.

Test Features

A few features were made for other purposes than to support tabletop games. The *FrameRater* feature keeps track of the application's frame rate, telling developers clearly when there is more action going on than the application can handle. The *LocationData* feature shows the *Camera's* current values and displays all potentially useful information when the application receives contact input. In order to test the application's breadth, a simple *Pong*[51] clone was made using nothing but the inherent tools.

[51] Atari (1972). *Pong* [Arcade game]. Designed by Alcorn, A.

Immersive Features

Background and Compass

In order to more easily set a mood fitting the current theme or environment, the *Background* feature allows users to select a background texture, with any kind of tint and either scaling the original image to fit the *Playground's* size or tiling it. Another simple, graphical feature is the *Compass* which displays the current *Camera* rotation by rotating the image of a compass in one of the screen corners accordingly. The compass can also be displayed beneath the *Camera's* rotation tag.

Fog of War

Fog of War is the name for when the user sees only the things which her characters or units are seeing, while the things the user has seen previously are displayed in the state they had when she last saw them and the things she has never seen are not displayed at all. It is a common element of digital games, especially strategy games, but it can be difficult to have in an analog game. However, we believe that it is a feature which can add immersion or new takes on many different types of games; whether talking of the dungeons of role-playing games, the fields of strife in wargames, or providing a version of *Monopoly* where you only see the squares around you and the squares you own. The *Fog of War* feature in *Tisch* makes use of *Line of Sight* to find out what parts of the world are seen, displaying them normally, have been seen, displaying them transparent, or not seen at all, not displaying them at all. Using *Fog of War* in a game works to raise immersion, as players automatically see only their present surroundings; allowing them to more easily imagine walking through a narrow corridor or rounding corners without having to pretend that they are not seeing the rooms on the other side of the wall or what awaits them around the corner.

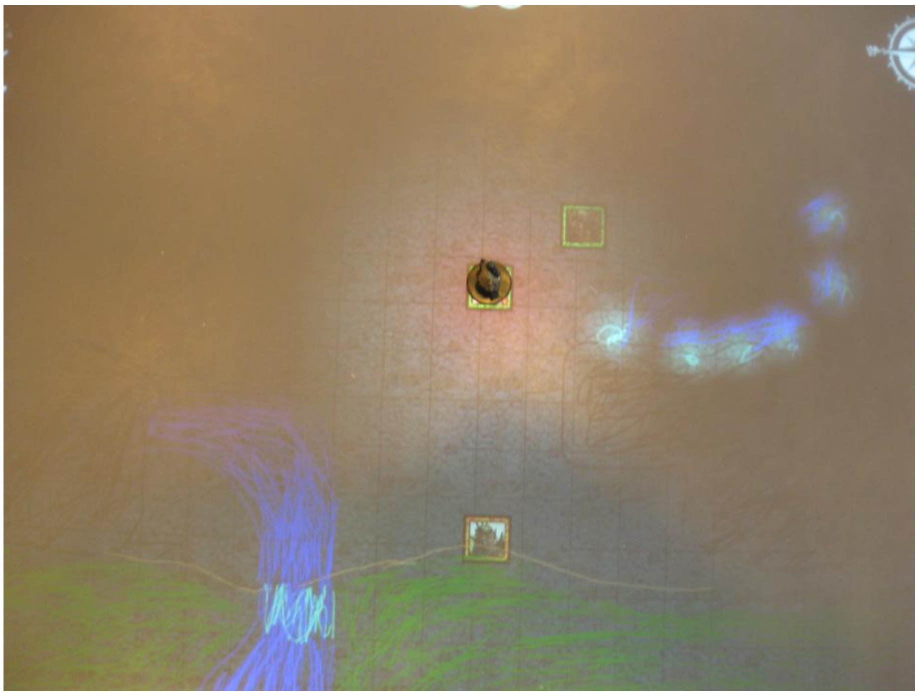

Figure 6: A dark cave with some scattered lightpoints.

Lights and Day & Night

A feature which is difficult to simulate believably in an analog game is the presence of lights; be it torches, flashlights, street lamps, or phosphorescent mushrooms. The *Lights* feature makes the entire world dark, except for where lightpoints, such as the just-mentioned, are placed. *Tokens* are also able to carry a lightpoint, which then follows the *Token* automatically. Lightpoints can be of various sizes and colors, and have the added ability of being able to pulsate to add animation and reality to the scene. *Lights* comes with a small graphical user interface from which users can easily drag-and-drop lightpoints, set their parameters, and preview the lightpoints' appearance.

Day & Night is a complementary feature to *Lights*, exchanging the total darkness applied to the background with a color and darkness depending on the time of the day. This time is not tied to reality, but controlled by sliding a half-circle, with a sun and moon on it, around; one full revolution representing 24 hours. *Day & Night* also comes equipped with displaying the present time in hours and minutes, as well as the day of the week.

Weather

The *Weather* feature applies a graphical overlay of either rain, snow, or mist to the screen, with a central parameter controlling the intensity of it. Hopefully *Weather* can aid players to immerse themselves more in scenes set with any of the present effects, be it getting lost in the mist or depicting the British capital.

User Testing and Evaluation

During development we performed a series of tests with different games; two early on (for the prototype version), one approximately half-way through (for the alfa version), and two near completion (for the beta version). We present these here, while the related descriptions of *Tisch's* versions can be found in Development, to provide insight into how user feedback aided the design process and show *Tisch* in actual use.

Formal Surveys

Initially, various surveys and research of the gaming activities we aimed to mainly support were planned to take place. We meant to take part in several game sessions, mostly role-playing games, to study behaviors and elements common to the activity and make judgments on how helpful it would be to have support for various tasks in *Tisch*. Part of these tests would also be discussions with groups of players, and specifically the game masters of role-playing games. Outside of these gaming sessions, we also made the judgment that discussions with game masters in general would give a lot of perspective as well.

Unfortunately this took place to a scarcely small part, with only a single *Warhammer 40,000*[52] session observed, partly because in the beginning it eventually became unclear what the focus of the tool was and had already taken a turn from the initial plan. When the project became more focused, a prototype was developed to test this new aim to try out the new purpose. In these and all subsequent tests a lot of discussion was also held to evaluate the testing itself and explore what the testers felt would be reasonable and good additions.

But even without formal surveys or testing early on, much of the work has still been based on our own experiences and more casual discussions with people interested in the project, most of which have presented several ideas just after being informed what the project is about. Any and all input that has reached us has been taken into careful consideration.

Play Tests

Below is an enumeration of the tests performed, in the chronological order they were performed. Only rarely did the same players participate in multiple game instances of the same game but some players participated in several different tests. While it would have been beneficial with some testers gradually learning the system through several tests, becoming intermediate users, it was not seen as feasible with the relatively few play tests that were performed to spend too many on finding this out. We also present some discussion on the various related game genres.

Tunnels & Trolls

1 game master. 1-2 players. 4 game instances. 30-60 minutes.

Early in the process we performed shorter test sessions using the role-playing game *Tunnels & Trolls*[53]. We chose *Tunnels & Trolls* because of its simplicity and that there were already scenarios available that were very suitable in this stage of the development. The scenario we used was about adventuring in a labyrinth, testing many of the basic features present in the prototype version such as *Drawing* and the *Camera*. The tests were carried out both solo and in pairs. The different sessions also alternated between the map being drawn by the game master or the player(s). The final test using this

[52] Games Workshop (2008). *Warhammer 40,000* [Tabletop miniature wargame, fifth edition]. Originally published 1987.

[53] Flying Buffalo (2007). *Tunnels & Trolls* [Tabletop role-playing games, fifth edition]. Written by St. Andre, K. Tunnels & Trolls originally published 1975.

scenario also utilized a pre-drawn map, making use of the *Grid's* ability to snap the lines to the grid which was very fitting for the pre-drawn map, in which all rooms and corridors were rectangular.

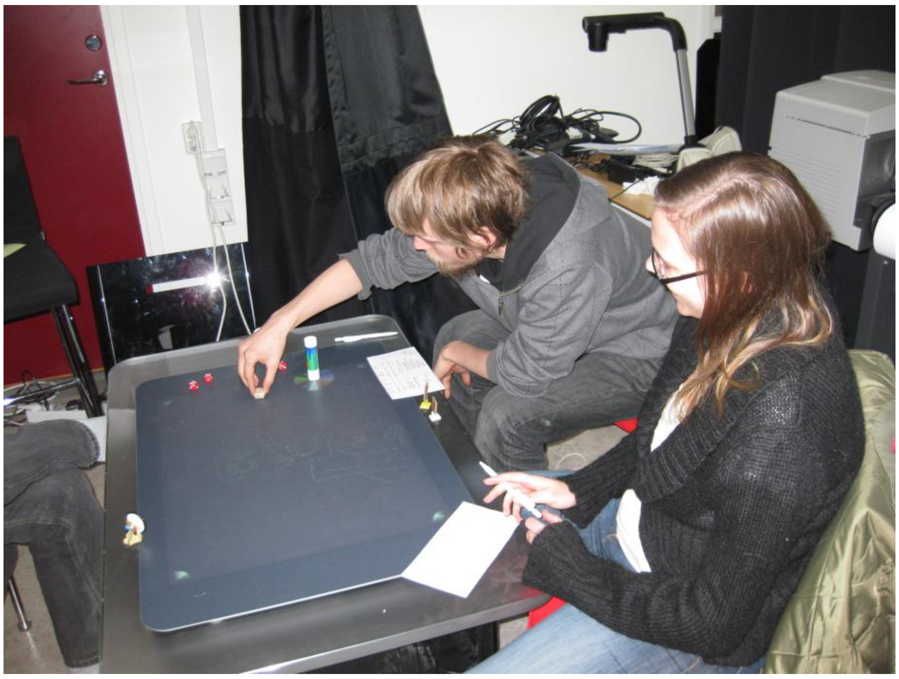
Figure 7: Playing Tunnels & Trolls with the alfa version.

The response was mixed as some testers felt that *Tisch* served little purpose, aside from the *Camera* which pretty much became the focus of the scenario along with *Drawing* the rooms and their contents discovered through the players' spelunking. More use of the digital platform was expected, with some suggestions for possible features. Among the suggestions was *Playgrounds*; when the rest of the game used digital tools, most players felt it would be more natural to also make their notes in the table instead of using a separate sheet of paper.

The Slaughterhouse

1 game master. 1 player. 3 game instances. 30-60 minutes.

The other early test was a small scenario focusing on *Immersive*, instead of *Practical*, features. The scenario was written by us for a single player who woke up without any memory and who had to explore the surroundings looking for clues. Compared to the *Tunnels & Trolls* test, the player made more use of colors and commented on the *Drawing* result, including stating heightened immersion from doing the sketching themselves.

World of Darkness[5] is a commercially successful example of this genre of tabletop role-playing games. It places heavy emphasis on its setting and theme, set in sullen modern times, and many of the skills available to the players' characters are centered on social interaction. Even so, making quick sketches of the geographical features of an outdoor environment or the placement of people in a building can be necessary, and the *Drawing* feature of *Tisch* supports this. The fairly simple rules that these games do have might also be quite easily handled by an extension or with flexible enough configurations.

Frag

6 players. 1 game instance. 60 minutes.

Frag[54] was used to test the potentiality of playing board games. Even though *Tisch's* focus lies on role-playing games and wargames, we wanted to test to what extent it can support board games at large. *Frag* simulates first-person shooter computer games, with the combat being rather similar to that of wargames.

It was the first test to incorporate *Tokens*, *RoundKeeper*, and *Line of Sight*, although the players elected not to use the latter explicitly in order to retain the tangibility and due to *Line of Sight's* functionality being used automatically by the *Tokens*. The test provided good input regarding what functionality was required from the *Tokens* feature and, more generally, questions about rule strictness and allowing *House Rules* (e.g. if *Tisch* should only allow the current player to move). Another important aspect of the test was not the testing in itself but the implementation of the game. The above mentioned functionality that players' markers are hidden during other players' turns and when not within *Line of Sight* is not part of the original game but felt natural in the *Tisch* version. In the standard setup this becomes tedious seeing as the players have to remember where their tokens were when the turn ends and also constantly determine if the tokens pop into *Line of Sight* when the current player moves. With the system keeping track of these details for the players, with the removal of the tokens still voluntary, it instead becomes an additional feature of the game and can be used to add another dimension to it.

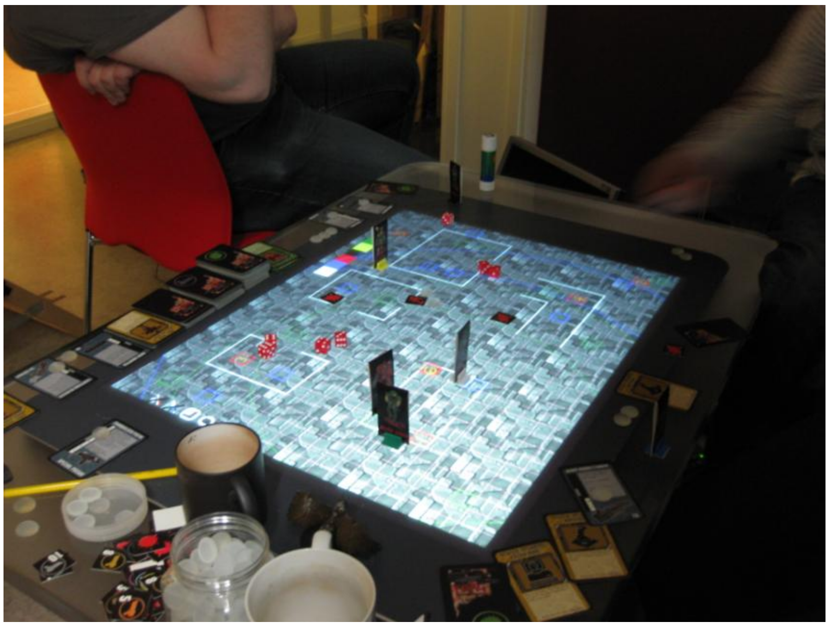
Figure 8: Playing Frag at the end of the alfa version.

Confrontation

2 players. 1 game instance. 45 minutes.

As the representative for wargames we tested *Confrontation*[55]. It can be played in small scale and has fairly simple rules, while retaining central elements such as *Line of Sight*, *Measuring*, and *Tokens*. This made it better suited than *Warhammer 40,000*, one of the more popular wargaming franchises and the one we originally intended to test, which is designed for significantly larger scenarios. *Drawing* should make it simple to create battlefields but a problem with many wargames (less

[54] Steve Jackson Games (2001). *Frag* [Board game first-person shooter]. Designed by Jackson, S. & Reed, P.

[55] Rackham (2007). *Confrontation: The Age of the Rag'narok* [Tabletop miniature wargame, fourth edition]. Originally published 1996.

frequently in role-playing games) is that they often take place in three dimensions, with units climbing up and down buildings and hills, something quite beyond the scope of the simulations made using *Tisch*.

Results from the test indicated that the present features provided little benefit once the *Drawing* preparations were complete, mostly due to the rules being focused on assigned values which *Tisch* knew nothing about and all encounters in the test being close combat which in turn rendered *Line of Sight* useless, the feature which was assumed to be the most prominent for the setting. More specialized wargaming features in general and *Confrontation* features in particular could substantially improve *Tisch's* usability.

Dungeons & Dragons

1 game master. 3-4 players. 2 game instances. 3 hours.

As the representative for role-playing games we tested *Dungeons & Dragons*[56], using the pre-made *Reavers of Harkenworld*[57] scenario. While character development is a core gameplay aspect, the latest edition has placed a heavier emphasis on the complex rules for turn-based combat on small-scale areas (comparable to "levels" in many computer-based games) through, for instance, re-introducing elements from wargaming, such as focusing more on *Tokens*, *Measuring*, and *Line of Sight*, and making the overall experience smoother by discarding many special rules. *Tisch* supports this through both the *Drawing* of the game world and keeping track of *Tokens* on grid-based maps.

Many custom systems and rules are condensed onto specialized character sheets and tables but those are the kind of things a generic system such as *Tisch* is less interested in handling explicitly, as new particularities would always appear with *Dungeons & Dragons'* regular expansions and updates. While not keeping track of the specific rules, the *Measuring* and *Line of Sight* features support their use generally, and players can make notes in a *Playground* about specific abilities and effects. These features support the original experience but the *Fog of War* is arguably a new one; allowing game masters to prepare game maps so that the players' exploration of the world and its dungeons can have more suspense with the appearances of enemies and the area in general occurring instantaneously when within proper range of the player characters. *Lights* can also provide a dynamic presentation making the created worlds more evoking. While having different rules and using a different type of *Grid*, the described use of *Tisch* would work equally well for many other role-playing games, such as the earlier mentioned *GURPS*[58].

The test session progressed in a natural manner with *Tisch's* features used either heightening *Practical* (e.g. *Camera*) or *Immersive* (e.g. *Lights*) aspects, as intended and hoped for. Users were positive and gave plenty of suggestions for improvements, primarily regarding digitalizing more parts of the game to remove *Excise*. The flexibility of *Tisch* received positive feedback even though users did not make many configurations on their own since doing so was perceived as complex and because it obscured the game area.

Pathfinder

1 game master. 4 players. 1 game instance. 2 hours.

The last test we made was with the role-playing game *Pathfinder*[59], which is based on the *Dungeons & Dragons'* 3.5 edition. The important aspect of this test was the manner in which it was conducted: with us acting solely as observers, neither participating nor aiding once we had shown briefly how to use *Tisch*. The test showed that, to be used successfully, *Tisch's* interface and usability

[56]Wizards of the Coast (2008) *Dungeons and Dragons Player's Handbook: Roleplaying Game Core Rules* [Tabletop role-playing games, fourth edition]. Written by Heinsoo, R., Collins, A. & Wyatt, J. Dungeons and Dragons originally published 1974.

[57]Wizards of the Coast (2010). *Reavers of Harkenworld* [*Dungeons & Dragons* scenario]. Designed by Baker, R. and Perkins, C.

[58] Steve Jackson Games (2004). *GURPS Basic Set* [Tabletop role-playing games, fourth edition]. Written by Jackson, S., Punch P. & Pulver, D. Edited by Hackard, A. & Jackson, S. GURPS originally published 1986.

[59]Paizo Publishing (2009). *Pathfinder Roleplaying Game* [Tabletop role-playing game].

needs to be improved. The players made cumbersome workarounds or avoided using features wholly because they were either not aware that the possibility existed or how to do it. The players also preferred playing the game in their own way, with their own *House Rules* and manner of doing things, rather than to settle for compromises which were more suitable for *Tisch*; an example being the *Tokens* used which were of a size that the *Surface* recognized as fingers, causing plenty of accidental *Drawing*.

Analysis

Overall, the tests gave very positive results. In the evaluating discussions during and after sessions, there was often encouragement and plenty of ideas and suggestions. Most importantly, the system was never considered to be a hindrance or in any way obstructing the gaming activity. In some early tests, however, some testers felt that even if it wasn't a hindrance, it wasn't a great help either since it felt more like a fancy sheet of paper. It was discovered that it depended somewhat on how the game was played, if the map was pre-drawn or not, who drew, and so on. More systematic and nearly identical tests could potentially provide interesting data regarding this. But as the *Drawing* and *Camera* functionalities improved, and as more focus was applied to the *Immersive* features, these opinions began to fade away.

No tests were made specifically to investigate the *Social Weight* or the reduction of *Excise*, but every test contained some questions and observations aiming to see where we currently stood and to what degree the current version had improved compared to before. In the same manner, none of the other design goals described earlier have been tested specifically but have rather all been important parts of all tests.

One aspect of role-playing games, which *Tisch* aims to remove, is commonly occurring location based questions. When in a role-playing scenario, the players' characters travel and encounter many things, from misty forests and spaceship cockpits to a gang of hooligans and talking buffalo. It is usually left to the players' inner eye to visualize how these object look from the game master's descriptions, carrying its own pros and cons; much how books have both advantages and disadvantages over films. As long as the environment is built solely in the players' minds there are no boundaries, while a shared space becomes shared by specifically constructing such boundaries to allow multiple people a narrower and more similar view. Although boundless play may be preferred by many it becomes trickier when it comes to the locations of the player characters themselves and how they stand in relation to the objects around them. Especially when encountering a hostile situation with a gang of hoodlums, players may want to know exactly where they and their foes are in order to come up with ways to interact. In such occasions, players often ask questions of the type "How many enemies are there around me?", "Can I reach Bob?", and "Where am I standing in relation to the turret?"; questions which add nothing to the experience but only serve to entertain a shared image of the game world among the players. During testing we monitored how often these types of questions occurred and it was satisfying to discover they came, when using *Tisch*, much more seldom, causing the activities to progress more smoothly and allowing players to act more freely and independently without needing to constantly inquire with the game master.

Since testing was performed with different sets of players, all with mixed experience of the game type that was tested, it was shown that experience with the game did not have much effect on how they viewed *Tisch*. One could assume that a veteran role-player would dismiss technical tools due to being accustomed and satisfied with the analog way of doing things and that players with no experience would be more interested in the tool than in the activity, but no such clear difference was discovered and the gaming activity itself rarely fell out of focus in favor of attention directed towards the tool. Unfortunately, no tests were performed of the games without *Tisch* to evaluate the differences in the activity when played with or without the system. Such tests were initially planned to be carried out during the process, as well as the preparatory and investigative tests described earlier, but were postponed and eventually cancelled due to lack of time. During the question sessions afterwards however, these matters were discussed and the testers were asked for opinions on this matter, pros and cons of using the system compared with playing without it.

The tests showed that *Tisch* could be used for a variety of different games to at least a certain level of success, thanks to being highly configurable. While this was not commented on by testers, as they did not partake in totally different tests or make any major configurations themselves, and while the tested games were not benefited equally, we believe that it goes to show *Tisch's* potential for being generic enough to support a very wide variety of games and users.

Process

The following chapter describes the development process, from the initial planning to the various implementation phases of the tool, along with sections regarding the work manner and our examples of use.

Project Plan

Following is the original, vague structure which the project went with. A more in-depth description, with *Tisch's* resulting versions described, can be found later in the report under Development.

First stage: Preparatory. Perform planning and other pre-production activities. Read documentation and learn the environment for developing on *Surface*. Some of this will be done before the formal start of the project. Try out our examples of use to find out in what ways they can benefit from the project. This stage is described in more detail under Preparatory Work.

Second stage: Structuring. Investigate which areas from the original analog structures in our examples of use, and other games and activities in general, that may translate well to a digital medium. Define more clearly the components of the toolkit that will be constructed, and in what order. Determine the software architecture that will be used. Eventually it was found that a prototype should be developed before setting the architecture in stone.

Third stage: Development. Implement one or a few of the features found in the second stage. This and the next stage are the main components of the development.

Fourth stage: Testing. Testing the feature or features implemented in the third phase, either in small scale with our own toy examples or with real tests; the latter being described in detail later, both the intended Examples of Use (shortly) and the finally performed Play Tests (in User Testing and Evaluation). Then return to the third stage, implementing other features and improving the tested features.

Fifth stage: Wrap-up. Finishing touches, aiming to make *Tisch* usable and extendable by others. Serious report writing.

Work Manner

We aimed to implement features and functionality iteratively, with which we basically mean one at a time. After discovering which features we judge as important and rank them accordingly, related to the current phase's goals and the difficulty of implementing it, a feature goes through implementation, testing, and finally improvement. Features will generally be brought to a proof of concept level rather than finished products. The evaluation of the features will be through internal as well as external testing.

The focus will lie on implementing new features rather than debugging and optimizing existing features. This breach against one of Spolsky's[60] central ideas of never ever working on anything new while there are bugs afoot is motivated by our decision to bring *Tisch* to the state of a proof of concept instead of a commercial product.

We intend to make the project very modular, both in the sense that additional parts can be added at any time and that undesired parts can be removed without affecting the remaining system. This is

[60]Spolsky, J (2004). *Joel on Software: And on Diverse and Occasionally Related Matters That Will Prove of Interest to Software Developers, Designers, and Managers, and to Those Who, Whether by Good Fortune or Ill Luck, Work with Them in Some Capacity*. Apress.

because the system can be used for a lot of different applications and in a wide variety of ways; as such we do not wish to impend anyone who would want to use it. We will also try to make our features as configurable as possible, to make the system further usable for as many purposes as possible. Additionally, the underlying system should have a modular design as well to simplify our own development process and allow for others to easily develop extra functionality to the system.

Examples of Use

The project will be targeted primarily to role-playing games which have some moderate degree of rule strictness; as differentiated from those games, such as *World of Darkness*[61], which are more akin to pure acting, which would have little to no use for tools. Apart from role-playing games there should also be some support for wargames and board games in general. Our examples of use have been chosen on this basis to try to fill various needs a game can have and cover as many of our developed features as possible.

Before adding new functionality or making significant changes to existing features, examples of use are to be considered and the need for the changes evaluated. Then, judging the ratio of importance and difficulty, the decision whether or not the changes are to be made will be taken. After changes are made they are to be tested, either by us or, in case of more important features, in a proper user test. Whenever a batch of new functionality has been added, testing should commence as well.

When the project was going through the initial planning several examples of use were set up with the following motivations:

Dungeons & Dragons[62], as a representative of standard dungeon crawling role-playing games. We chose its fourth edition for two reasons: parts of it have been digitalized, which makes it interesting to see in what possible ways one can merge the work done with this project, and the fourth edition is notably simpler than the previous editions.

Eon[63], because of its intricate combat system; to see in what way we can simplify it with the help of a tool. Its complexity matches the easier system found in *Dungeons & Dragons'* fourth edition.

Warhammer 40,000[64], as a representative of miniature wargames, chosen because of its popularity.

We also planned to look at crisis management in some form; details to be decided once we had learned more about crisis management in general during the first week of the project's first phase.

However, quite early in the project the list of examples of use was revised for several reasons. Firstly, the consideration for crisis management was put on hold. We realized after some research of the area that we had an incorrect picture of it and that making the tool cover a too broad area of applications would restrict the research and results we wanted to achieve when it came to games and gaming. Secondly, we replaced *Eon* with less rule-abiding role-playing games. This was not bound to any specific rule system but based on the notion that that area had to be tested as well because that kind of role-playing games make use of other elements and attracts another kind of players than *Dungeons & Dragons*, another of our examples of use. We also exchanged *Warhammer 40,000* with *Confrontation*[65], a wargame of the same kind but better suited to smaller scenarios.

[61]White Wolf (2004). *The World of Darkness: Storytelling System Rulebook* [Tabletop role-playing game, second edition]. Written by Bridges, B., Chillot, R., Cliffe, K. & Lee, M.

[62]Wizards of the Coast (2008) *Dungeons and Dragons Player's Handbook: Roleplaying Game Core Rules* [Tabletop role-playing games, fourth edition]. Written by Heinsoo, R., Collins, A. & Wyatt, J. Dungeons and Dragons originally published 1974.

[63]Neogames (2004). *Eon* [Tabletop role-playing game, third edition]. Originally published 1996.

[64]Games Workshop (2008). *Warhammer 40,000* [Tabletop miniature wargame, fifth edition]. Originally published 1987.

[65]Rackham (2007). *Confrontation: The Age of the Rag'narok* [Tabletop miniature wargame, fourth edition]. Originally published 1996.

Preparatory Work

Before the actual work commenced, some research of the area was done. The first couple of weeks of the projects were devoted to reading up on earlier work and discussing possible extension areas for the project with various researchers at the Interaction Design department. The areas aimed to be covered made up a wide range, including interactive tabletops, social interaction between users of said systems, digitalization of games and gameplay elements, tangible interfaces, and ubiquitous computing. Plans were also made for formal surveys and external testing, whether external in the sense that we would not partake or in that it should take place outside the university; one idea was to have it present at a large role-playing game convention[66] to quickly learn first-hand impressions from a large number of one of the target audiences.

Much of the research performed has already been presented briefly in the Background and Research chapter, accompanied by what was found to be the most useful information. Yet there were also articles or parts of articles which were only tangents to the project and thus have been omitted from this report, with the obvious exception of this paragraph. Other read material include game rule books, to better prepare ourselves for future testing and to better understand what moments might benefit the most from *Tisch*.

Other preparatory work includes discussing and investigating possible examples of use, attempting to discern what benefit each would lend and in what manner these games should be observed, gathering materials suspected to be useful later on, such as role-playing rule books, and setting up the work environment, both in terms of the physical office space and getting the computers' software up-to-date.

Development

From the start of the development of *Tisch* the goal was to create a system that is easy to use, modular, and highly extensible. This dictated that play testing of many different types of games, with users both accustomed to the activity and not, needed to be an integral part of the development. From a technical point of view, the intention was also to make it possible for others to expand the system with their own specific features and modules. The design is very modular; every feature being independent of all others and based around a core module. Additional features can be created from scratch by users with programmatic skills without modifying or even viewing the source code of the core module and are configurable from within the system. The architecture is described in detail in Technical Specs and Appendix A give details on how to continue working on *Tisch*.

The system was designed iteratively, with each feature being added in incremental steps; as a prototype, tested, improved, tested again, and finalized. An initial ranked list of possible features was created through analyzing related work and using our personal experience of playing tabletop games. This list, which was never intended to be fully implemented, was continually updated during the development process based on suggestions and opinions from testers as well as observed needs and newfound insights. Testers were recruited from our pool of friends, colleagues, and acquaintances and, in turn, their connections. This was not seen as a methodological problem since the tests were formative evaluations to provide improvements or new features rather than assess the feasibility of existing ones. The tests are mentioned in this section at each point in time that they were performed, with details about what results we gained from them and how those results affected the development, while leaving the more verbose descriptions to the User Testing and Evaluation chapter.

The importance of *Creating Game Worlds* for tabletop gaming made *Drawing* the identified core functionality of *Tisch*. At first developed with a static view, a *Camera* was added quickly but kept as a separate feature to allow future developers to create alternative versions. *Tokens* and other features were then gradually added as tests showed what features were missed the most (modulated by the estimated time required). For all features several different interface solutions were considered and often tested, since several different options existed (e.g. finger input, tags, and more traditional interfaces such as buttons and menus) and users had preconceptions from other systems. An example

[66]GothCon (2011). *GothCon XXXV* [Annual role-playing game convention]. Gothenburg.

of the latter was that many testers, being familiar with *iPhones* and *iPads*, tried to “pinch to zoom” which was a difficult option for *Tisch* since it allows simultaneous input from multiple users. For the same reason, *Tisch* uses no gesture recognition as even though it might be possible it was not believed that the available time was enough to implement and test it.

A driving force throughout the development of the features, and *Tisch* in general, was the “Why?” question. Since tabletop gaming is possible without digital support it was important to ensure that the features were regarded as useful; not interesting simply due to being novel. Many of the specific details concerning the first features were for this reason focusing on minimizing *Excise* while it was only the later features that started targeting *Immersion*. The project’s four versions are described in more detail in the following sections.

Prototype

The goal of the prototype was to become acquainted with the *Surface* environment and to try out some basic features without paying any heed to a lasting and functional software architecture. The former to explore what was possible and what was not and the latter to put the architecture design on hold until we knew more what we were going to end up with. Knowing from the start that the prototype version was even more a proof of concept than the final product allowed us to discard general solutions and to adopt the functionality to the tests we would conduct.

We knew from the start that we wanted to be able to perform *Drawing* very easily and intuitively. From this we quickly decided that when the application detects a finger contact, if nothing else is done, some kind of line or pixels should be drawn to the screen. The prototype implemented this by telling the graphics card to draw simple lines from contact point to contact point. We also knew that we must be able to erase what we draw. To solve this we detected when two fingers were moved together, close enough, and let them both act as an eraser. However, we discovered quite soon that it was difficult both because the application believed that the two fingers were parts of the same if they were too close and because we found no obvious solution to what would happen when the two fingers parted ways while still touching the surface; should we keep erasing but in two separate parts of the screen? Relating also to *Drawing Tools* is the palette which allowed users to hold down a button, displaying a half-circle of colors above it, and drag the finger in any outward direction to choose the appropriate color. Another basal feature was the *Camera*. In the prototype, this was handled by holding one finger on top of a button by the bottom side and moving another finger across the screen; causing either panning, zooming or rotation depending on which of the three buttons was held down. This worked to a certain extent but it showed the problems of obscuring the screen with buttons and using either rotation or zoom was described by testers as awkward. A *Saving and Loading* system was also implemented wherein users could save their maps with assigned pictures, the pictures being either a screenshot or a doodle drawn inside a picture frame while the *Saving and Loading* mode was active. Users were able to scroll through the saved maps in the bottom half of the screen, while the mode was active, and load a map by clicking on it. Finally, users could toggle a pre-defined music track on and off with a button; the first purely *Immersive* feature.

Two different tests were performed with the prototype, one of each side of the role-playing action versus narration spectrum: the simple, dungeon crawling role-playing system *Tunnels & Trolls*[67] and the self-written, narrative-heavy *The Slaughterhouse* scenario. Results from the *Tunnels & Trolls* test showed that the only aspect which improved the experience, as compared to running the game on a computer or with pen and paper, was the *Camera*, both being able to pan around and to zoom out to view the entirety of the explored dungeon. In *The Slaughterhouse* test we found out that, even with the limited tools at the players’ disposal, *Immersion* was raised significantly as soon as players were able to draw their surroundings by themselves, without interference from the facilitator. This influenced our decision to at a later stage, once enough *Excise* was removed, include features dedicated solely to being *Immersive*; a decision which up until then had been wavering on uncertainty as there was little to prove by what degree such features could improve the experience. We received no significant negative feedback, aside from some technical problems, and the general consensus was

[67]Flying Buffalo (2007). *Tunnels & Trolls* [Tabletop role-playing games, fifth edition]. Written by St. Andre, K. Tunnels & Trolls originally published 1975.

that the application either improved or did not decrease the enjoyment of the game, both of which are central to making *Tisch* useful both as an end-user product and to succeed as a proof of its concept. The technical problems prompted us to implement a logging system which automatically writes whatever happens to a log file, especially when problems arise.

The prototype version lasted from the end of January 24 until February 18.

Alfa

The goal of the alfa version was to create a new system capable of expanding with features more or less ad infinitum and to perform a boardgame test with it, both as a more advanced test than the prototype tests and to explore the possibility of using *Tisch* for boardgames.

The first step of the alfa version was a restart with the architecture redesign which began by the end of the prototype version and continued, in less degree, all through the alfa and beta versions. The new architecture is described in detail in Technical Specs. The most important aspect, however, is the independence between features; while a feature can make use of another, it must be able to stand on its own legs.

In-between the prototype and the alfa version we performed case studies of *Warhammer 40,000* to better understand and analyze what elements of wargames were applicable. These studies showed that an obvious problem with many wargames, *Warhammer 40,000* included, is the physical size of the game board; requiring as much as five or six *Surfaces*. Another problem is elevations, where units are able to ascend on top of buildings which in turn affect their *Line of Sight*, which has no obvious solution in *Tisch*. Other aspects, however, could gain much from being digitalized: templates for area effects, detecting *Line of Sight* between units, along with *Measuring* the distance, looking up values in tables, counting the results of multiple dice, and the management of the tangible figures. These studies lead to the creation of the *Line of Sight* feature with its added measurements, and the *Tokens* feature with auras which can be used to signify area effects.

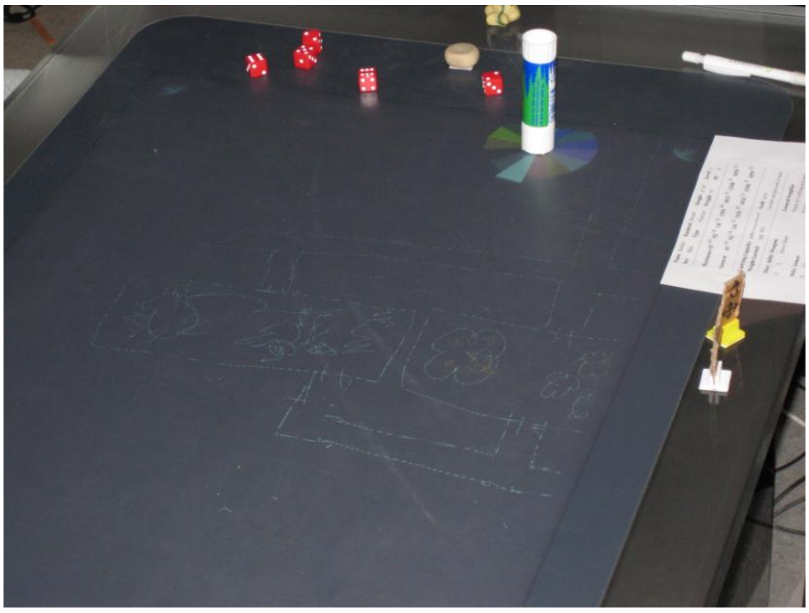
Figure 9: The early alfa version.

Both of these features fitted well with the boardgame eventually chosen: *Frag*[68]. We knew we wanted a boardgame with features which could be digitalized without feeling superfluous, especially the *Drawing*. Choosing *Frag* also prompted us to implement *Grids*, the *RoundKeeper*, and the already mentioned *Tokens* and *Line of Sight* features. With the heavy interaction going on between the *Tokens* and the *RoundKeeper*, keeping track of which *Token* was the current player, and the *Line of Sight*, with the technology to allow only the current player to see other *Tokens*, made us re-shape the

[68]Steve Jackson Games (2001). *Frag* [Board game first-person shooter]. Designed by Jackson, S. & Reed, P.

architecture. *Tokens* were made a standard part of the core module, allowing other features to interact with it more easily, as was the code investigating if there is *Line of Sight*, making the *Line of Sight* feature only responsible for manually checking *Line of Sight*. In later versions, *Line of Sight* would gain control over which colors to regard as opaque and transparent, but in the alfa version there was not yet any way for the independent features to talk with one another. Several features were custom-fitted to suit *Frag*, postponing their generalizations until the next version, such as allowing *Tokens* to assign specific values relevant only to *Frag*.

Another important advance which came during the alfa version was changing the *Drawing* from using primitives, simply telling the graphics card to render a line from one point to another, to using textures, stretched and rotated appropriately. This change allowed us to more easily alter the *Drawing's* appearance and, in turn, made it look better according to inquiries. The two drawbacks with this method are that the rendering calls became more processor expensive, having to handle every line between every set of two points with more care, and that zooming caused undesired behavior: while zooming closer or further from a line should, in reason, cause it to be rendered larger or smaller accordingly, all testers agreed that the effect made zooming useless; either the lines became too obstructive or too thin to see. To resolve this we rendered lines with equal thickness but in their correct relative positions. While it is possible to spot the inconsistency, so far no uninformed viewer has commented on it.

Saving and Loading was re-introduced in the alfa version, this time storing maps to tags. The idea of simply stamping a tag in a square appealed to us, instead of having to scroll through lists or press buttons, and it was the same affinity which caused *Tokens* to be created in the same manner: stamping a tagged object on various fields to assign it properties. It was, however, by no means an obvious decision until we had tried it ourselves. *Saving and Loading* still retains fragments of a support for storing and displaying maps as images, even though it did not reach the end product, while there was much discussion if *Tokens* should instead have a square in which tagged objects could be placed and assigned whatever values were selected from an interface with the use of touch input. It was the ease of stamping, together with the experienced *Excise* of scrolling through lists and other discardable movements, which convinced us to proceed as we did.

The *Configurer* feature, which enables users to modify the application's settings while running it, was added, allowing us to fit *Tisch* to any kind of game within seconds; at least to whatever degree the features allow. The *Background* feature was also added, at first with only colors but quickly expanded to handle tiled and stretched images as well. A number of test features were also added, but they are of little interest here.

The alfa version lasted from February 21 until March 23.

Beta

The goal of the beta version was to have a system capable of aiding a full-scale tabletop role-playing game. For this purpose we chose *Dungeons & Dragons*. The obvious fallacies at the beginning of the beta version was the lacking support the current *Tokens* and *RoundKeeper* features had for games other than *Frag*. The *Tokens* feature was thus expanded to being able to provide with any kind of numeric values, with the options of presenting them as text, an aura, both, or neither, while the *RoundKeeper* adopted the same stamping attitude as both *Saving and Loading* and *Tokens* by allowing users to simply stamp their figures on the *RoundKeeper's* interface.

We had long known that the ability to create a more private interactive area, for taking notes or to be able to split the screen, would be able to extend the application's usability considerably; both the feedback from our testers and the research (e.g. both STARS[69] and Pirates![70] used PDAs) had told us so. To solve this we implemented what would later be called *Playgrounds*, but which at this stage was a part of the core module. The first attempt was to create a new interactive area, or *Playground*, when a user held down a finger. After a few seconds the screen would turn black and the user could draw a

[69]Magerkurth, C., Cheok, A.D., Mandryk, R., Nilsen, T. (2005). *Pervasive Games: Bringing Computer Entertainment Back to the Real World*. ACM 2005.

[70]Björk, S., Ljungstrand, P. (2007). *Pirates! Using the Physical World as a Game Board*. Space Time Play. On the Synergy Between Computer Games, Architecture, and Urbanism.

line to select the place and size of the *Playground*. This failed severely, however, as users frequently held down fingers accidentally; or rather the users were not aware of the functionality and, as such, were not aware that holding down a finger was a gesture they should manage. A substantial improvement, especially when it came to avoiding accidental *Playgrounds*, was changing it to four held fingers and creating the *Playground* with those four fingers as the boundaries. With this change we also added the basic functionality of moving and resizing the *Playground*, although it was impossible for users to delete it. The *Dungeons & Dragons* tests, however, informed us that it was still possible to accidentally create *Playgrounds*, especially with multiple users around the *Surface*, and that such surprising and undesired results very easily distracted users and completely breaks the mood.

An important item was added to the core module in the form of what we chose to call an *Eavesdropper*, a message passing system intended to allow future features to make use of earlier features without the earlier features' knowledge, i.e. discarding the manual sending. So far the *Eavesdropper* requires the sending features to perform the sending manually, as we did not find a way to insert automatic sending, but the hope is that we can spend more time on it in a later phase. This is important because it can be difficult to know what information another feature is interested in and because it is undesired to manually write code for sending all types of messages. It should be emphasized that it is not the intention of the *Eavesdropper* to make all information public, but only the properties which are already available to features such as the *Configurer*; that is, the properties which users might change and which may be relevant to other features. The introduction of the *Eavesdropper* allowed other features access to the opaque colors chosen by *Line of Sight* and the *Grid's* size, to name two examples of information which can be highly valuable for other features to know about.

Before we performed the *Dungeons & Dragons* tests we also performed a *Confrontation* test, investigating *Tisch's* usability for wargames. The test showed, as described in greater detail in Play Tests, that there was little current functionality which aided the actual experience, whether relating to removing *Excise* or enhancing *Immersion*. While this was a setback, this made us decide to focus much more strongly on role-playing games as their traits are more easily digitalized. In essence, we believe that a digital aid for wargames would have to be shaped to suit one specific (or a few closely related) wargames and discarding the more generic capabilities.

With the advent of more expansive maps came save files exceeding 50 MB, as there had been no optimization at all in terms of what was saved. The massive amount of data from such large maps made us add some easy optimization methods by only rendering items which were within the screen bounds. Unnecessary *Drawing* data was discarded, essentially by removing points lying wholly or almost in a straight line between its surrounding points. Running the optimization once removed between 4 and 25%, averaging at 8, of the present data without any visible effects. This optimization runs automatically and can, if desired, be run multiple times to enhance the result. The file sizes of the example maps, upon running the optimization 20 times, decreased from 61.0 MB to 11.6 MB, while simultaneously improving the runtime performance by causing only a sixth of the data to be managed.

We began work on a *Fog of War* feature but, seeing as the concept was fairly large, chose to start out with *Lights*. While not available for the *Confrontation* test, by the time of the first *Dungeons & Dragons* test it was. The first test showed that it was immensely powerful in terms of enhancing *Immersion*; upon activating the *Lights* feature the test session became substantially more interesting for all participants. The oppressing darkness and the undisclosed information about what awaited the players a little further in enhanced *Immersion* more than we had hoped. However, a problem was that as the map was being drawn while playing, so were the lightpoints placed out as the game progressed. This forced the *Drawing* to either be done in the dark, before placing the lightpoints, or placing lightpoints where there was, at the time of placing, nothing to see. This experience, combined with the overall feeling that the time spent using the *Drawing* and *Drawing Tools*, caused players to feel that the game was progressing too slowly. With this in mind the following full-scaled *Dungeons & Dragons* test used pre-made maps and pre-placed lightpoints, both to great success. While we still desire for *Tisch* to avoid enforcing preparations, in the case of darkened areas and expansive maps preparations are nevertheless advantageous.

The beta version lasted from March 24 until April 8.

Gamma

Tisch's current version has not had any clear goals, as the earlier versions did, but is simply the one between the end of the beta version and until the end of the project's time period. The largest goals here was the writing of this report and making the system more easily usable and extensible for others, including documenting the code more thoroughly and looking into ways of making the core module more stable; the latter is important because it is highly undesired that a single feature, whether new or old, should be able to crash the core module. The *Configurer*, *Saving and Loading*, and the *Eavesdropper* were also explored in order to find ways of making using them easier for other developers. The *Configurer* was expanded with real support for buttons, whereas such interactions was previously done through what could best be described as a hack, and gained much more automated functionality. *Saving and Loading* was improved in a similar and related manner, as well as becoming more stable in the veins of the same idea as *Tisch's* entirety: if one feature fails its saving or loading, the rest should still function fine.

We improved the faulting *Playgrounds* by both creating a feature to manage it, allowing users more configurable freedom over new *Playgrounds*, and by assigning a button in the toolbar for it, replacing the four held fingers, or at least supplemented if users preferred the previous interaction. Users can also move a *Playground* onto the button to delete it and link *Playgrounds* with tags to only display it while and where the tag is put down.

Encouraged by the success of the *Lights*, we added *Day & Night* and *Weather*; both more in the shape of proof of concepts than thoroughly functioning, but that was the case with the beta version of *Lights* as well. The gamma version of *Lights*, however, was expanded with an optional graphical interface, from which users can more easily create, alter and preview lightpoints.

Meanwhile, *Fog of War* was implemented for real. It proved to be harder than expected to find a reasonable implementation, choosing between one which only altered the drawn lines and one which covered parts of the screen with a transparent or opaque overlay. The latter had the benefit of automatically covering the graphics of other features, such as *Backgrounds* and *Tokens*, but it was found to be difficult to implement the part which handled areas which have been, but are not currently, seen. Instead of spending too much time to solve this problem, we chose the former solution; if nothing else than as a proof of concept. This way of manipulating the drawn lines, altering their alpha values, forced us to remake the *Erase* feature, as it used to only set the alpha value to zero instead of truly removing points. Changing *Erase* to actually delete points and lines had the added benefit of instantly removing the redundant computer work of managing irrelevant data.

The gamma version is the current version, starting April 11.

Technical Specs

For those more technically inclined and interested in the actual workings of the tool which the project resulted in, a specification of the system is included. This chapter lists the basic structure of the program and the platform it is developed for.

Architecture

Tisch's architecture has a core module as its base, containing the most basic functionality. On top of this there are a number of *Draw Areas*, the technical name for what we present to users as *Playgrounds*, of which there must be at least one: the *Alfa Draw Area*. Each *Draw Area* contains a set of features, all of which are loaded from the same assemblies, which may be configured differently and belong as separate instantiations to any number of *Draw Areas*.

The application in itself is based on the *Surface's* Core API, which in turn is based on the XNA Framework and part of the .NET Framework. While we performed some experimentation and investigation of the WPF along with *Surface's* Presentation API, we chose not to use it since we were more familiar with XNA Framework; ensuring us of what features, especially when it came to graphics, we would be able to achieve. *Tisch* makes use of the Core 1.0 API, which is the latest version not counting the one targeting the *Microsoft Surface 2*, the XNA Framework 4.0, and the .NET Framework 4.0; or more precisely C# 4.0. The architecture is as depicted in the below diagram, although somewhat simplified in regards to the relations in the bottom row.

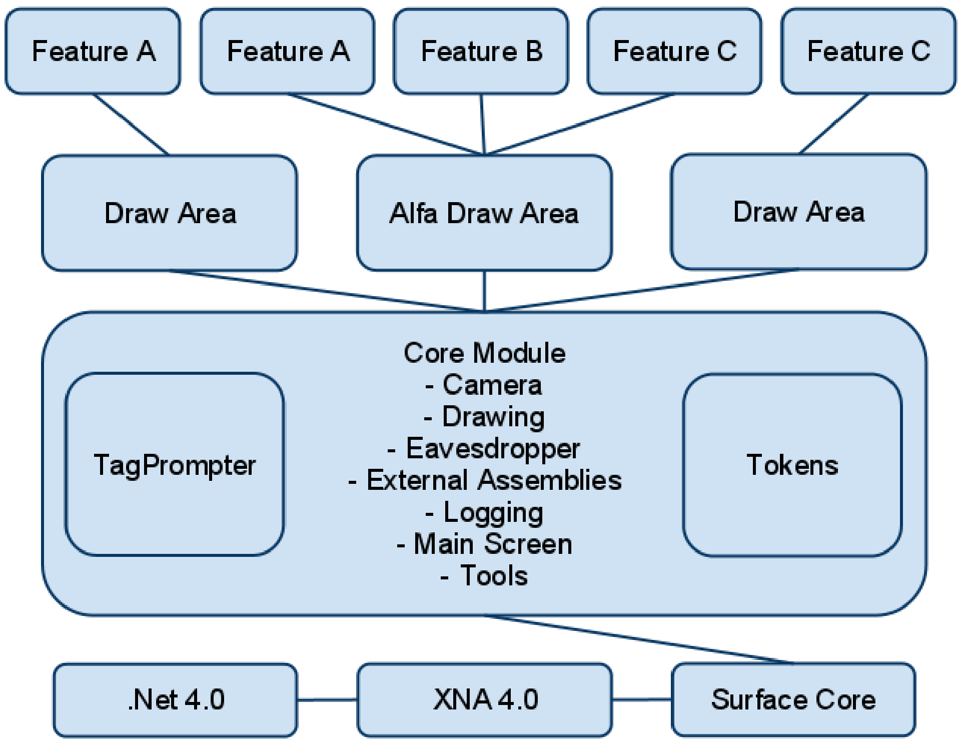

Figure 10: A diagram of Tisch's architecture, with the more specific elements at the top and the broader frameworks at the bottom. Note how some features are shared but uniquely instanced.

Environment

The *Surface* uses cameras to sense touch input, seeing when and where something approaches the screen. As such, the object causing the touch input does not have to physically touch the *Surface*, but may hover slightly above it. The object, depending on its size, is determined to be either garbage (if too small), a finger (if at or around the size of most fingers), or an arbitrary object. Garbage is discarded while the other two types are given a signature stating what they are believed to be. *Tisch* divides all contact input into different sets of methods, allowing features to decide through the method override which input they are interested in. Presently no feature makes use of objects.

Using the same camera technology, the *Surface* can detect two different types of tags: simple Byte tags, containing black dots each representing one of eight digits (256 values), and complex Identity tags, with two 64-bit values (about 3.4*10^38 values). Tags are printed squares, measuring approximately 2x2 centimeters, which can be attached to any fairly flat surface. *Tisch* makes heavy use of tags, with many of the current features requiring tags to function properly (e.g. *Erase*, *Palette*, *and Camera*). So far only Byte tags have been used, but features are in no way hindered from making use of Identity tags; in the case of an expanded *Saving and Loading* feature, for instance, it might be desirable to allow both types.

By default, and in *Tisch*, the *Surface* has two threads: one for contacts and one for the update and render loop. This enables contact info to be taken in immediately, while the update and render thread (by default) targets 60 frames per second. Caution has had to be taken due to the possible interleaving problems resulting from the concurrency, making sure that a change in one thread does not sabotage for the other.

The *Surface* is intended to use a Launcher program to present the available applications, which it automatically loads from a directory. The Launcher contains some useful tools for the applications, such as a touch keyboard, which we found no way of using without the Launcher. The major downside with the Launcher is that it enforces a time-out in case no contact has been made for (by default) two minutes, causing the system to present users with the choice of continuing with the present activity or to return to the Launcher menu. If no further contact is made within (by default) one minute the application closes down. For some games and activities with large amounts of social interaction, where users may be expected to talk and discuss amongst one another, these time-outs are effectively ruining what *Immersion* there is. The time-out durations can be modified, but only in the *Surface's* registry[71].

Core Module

Tisch's core module consists of the following parts:

- A *Camera* keeping track of the current position, rotation and zoom level. Interacting with it, however, is left to features.
- The actual *Drawing* of points and lines. Again, manipulating and using them is up to features; the core module only handles the input which draws them and renders them properly.
- The *Eavesdropper* which is a message passing system intended to automate the sending of messages, to more easily allow future features to subscribe to previously made features without requiring access to the code. At its present state the *Eavesdropper* has some support for this, but is mostly a standard message passing system based on the C# events.
- Two external assemblies: one for writing and reading to a settings file and one for writing and reading to a save file. The former is developed by Gravelyn, N.[72] and the latter by us for another project.

[71]Surface official documentation. Setting the time-out durations. Link: http://technet.microsoft.com/en-us/library/ee692022(Surface.10).aspx (Visited 2011-05-11).

[72]Gravelyn, N. (2010). *EasyConfig* [Library for reading config files]. Link: easyconfig.codeplex.com (Visited 2011-05-11).

- A logging system that automatically writes errors to file, along with some notable events such as the loading of features. The log files are simple text documents which can be found in the application's folder, along with a timestamp.
- The application's main screen which handles the connection between the *Draw Areas* and the *Surface*, both in terms of contact input and rendering output. The main screen creates the *Alfa Draw Area* at startup but receives no further unique treatment.
- A number of tools. This includes sliders, buttons, value selectors and other elements whose functionality is common among many features, as well as a large amount of helper functions, ranging from checking for intersections and *Line of Sight* to converting between types and managing enums. Overall, tools which can be of use to any and all features but which are not instantiated in the core module.
- The *Tokens* and the *TagPrompter* features, described in detail in Tisch.

Some custom types have been made to replace XNA Framework's and *Surface's* inherent types; namely a DrawBatch with additional rendering information, instead of XNA Framework's SpriteBatch, and a DrawAreaContactArgs to better suit *Tisch's* use of *Draw Areas*, instead of *Surface's* ContactEventArgs. In both cases all original functionality and structure is kept, allowing developers to use these types in exactly the same manner as with the original types.

Features

Each feature, with the ones integrated as parts of the core module, are intended to be stand-alone products. They may take advantage of each other, through the *Eavesdropper*, but must not depend solely on another's support. This is to ensure that the features should continue to function even if another feature should fail, be turned off, or simply not present on the current computer. In line with this it has been designed so that in the unfortunate event that a feature should fail, the failing feature will be turned off and restarted immediately with as little possibility of damaging other features as possible. Should the attempt to restart the feature fail as well, no further automatic attempts are made. Users, however, are able to manually reset even failed features, ensuring that problems caused manually can be resolved manually.

It was believed to be important to make *Tisch* scalable both by us, during the project, and by others who may continue where we left off. Planning further ahead, comparing it to an actual product, it could be vital that people can easily add and customize the application without having to modify the core. This was not based on any concrete evidence or previous research or experience, but looking at the success of applications and games supporting plug-ins, such as *Mozilla Firefox*[73] and *World of Warcraft*[74], it seems clear that such functionality at the least adds value to a product.

A feature inherits from an abstract type in the core module which contains a number of useful tools. Features are encouraged to make use of the so-called FeatureInfo attribute for all of their properties, an attribute with some meta-data which allows *Tisch* to automatically load the properties' values from file. The FeatureInfo attribute is also used extensively by the *Configurer* feature, as it contains useful meta-data regarding the properties' contents; such as default values and user-friendly information regarding its purpose.

Features are, generally speaking, their own assemblies and loaded automatically by *Tisch* as long as their assembly files are in the appropriate directory. Exceptions to this are the features incorporated as part of the core module and the features *Fog of War*, *Lights*, and *Day & Night* which are all individual parts of the same assembly.

[73] Mozilla (2011). *Firefox* [Web browser].
[74] Blizzard Entertainment (2011). *World of Warcraft* [Online role-playing game].

Discussion

The current implementation of *Tisch* meets the design goals set out at the start of the development but, as in any exploratory development like this, there is more work possible. Further features removing *Excise* and adding *Immersion* would make it both more useful and more advantageous compared to analog games. Most, if not all, of the current features also have room for improvements, whether through added functionality or making the present functionality more intuitive.

None of the users have voiced experiences, aside from temporary technical issues, that *Tisch* disrupted the social interaction, i.e. it had negligible *Social Weight*[75], or that it hindered the actual gaming experience. These two results are important since they alone could deem the system unusable. If the final system would be considered obstructive in any way users would be discouraged to use it, effectively working against anything the system might add to the experience. It is perhaps not too surprising that users began appreciating the potential of *Tisch* fully, i.e. providing an answer to the "Why?" question mentioned in Development, when the features beyond *Drawing* and *Camera* became available; in particular the *Lights* feature which showed users interaction which is impossible in an analog setting.

We believe a hybrid tool such as *Tisch* is an interesting concept, with focus on immersion and rule freedom, and is the right way to go. Our tests have shown that such a tool has a place amongst play, but where the focus of the tool and the aid it provides is highly dependent on the players and the activity; whether the users prefers generality over very directed yet heavy support, or if the activity is such that is gains greatly from a tool made specifically for it or the flexibility comes in handy. It is true though that with the generality a tool as *Tisch* provides, less focus is put into the tool since not as much of the game is run by the system instead of the players themselves.

Besides these general observations, we present some more specifics below made during the process of developing and testing *Tisch*.

Technical Limitations

One aspect of the *Surface* which makes it difficult to play large scale wargames is its size, measuring 53x107 cm; much less than many wargames require. *Confrontation*[76] works to some extent, in those cases when the game is kept to smaller skirmishes while a standard-sized *Warhammer 40,000*[77] scenario would require an area approximately six times larger, or even more. The size does not impose a problem only because some games require larger surfaces, in order to place and interact with the physical unit *Tokens*. One also has to be careful not to crowd it with interface, as users may target it from any orientation and any statically located interface risks obstructing one or more users. The present solution in *Tisch* is to have a small set of buttons toggling what additional interface is visible which, while functional, takes no consideration to various orientations and will no doubt clutter the screen once too many are added.

Possible Expansions

Not counting the material discussed in Partial Implementations below, expansions to *Tisch* can largely be divided into two groups: features which extend the general usability and features which improve functionality for specific games.

An example of the first category is themes: being able to make the look-and-feel of the system comply with a genre, such as horror, fantasy or science fiction, to better facilitate the intended mood

[75]Toney, A., Mulley, B., Thomas, B. H. & Piekarski, W. (2002). *Minimal Social Weight User Interactions for Wearable Computers in Business Suits*, in conference proceedings of IEEE International Symposium on Wearable Computers 2002.

[76]Rackham (2007). *Confrontation: The Age of the Rag'narok* [Tabletop miniature wargame, fourth edition]. Originally published 1996.

[77]Games Workshop (2008). *Warhammer 40,000* [Tabletop miniature wargame, fifth edition]. Originally published 1987.

of the game. Themes could exchange the interface with graphics suiting the setting better, cause the *Drawings* to be rendered differently, or apply some other graphical filter to the screen. Themes have been more of a vague concept than a clear idea, which is also the reason why we have yet to implement this kind of feature.

Another *Immersive* example is a sound feature, whether for handling both music and sound effects or merely one of them. This has been a requested feature, and we have plans for what it could look like, but we have not deemed it important enough to implement; largely because its presence would affect nothing else. The core idea behind a music feature lies in playlists, or so both we and the testers who have requested it have thought, which in turn could be dependent on the previously mentioned themes.

An example of a feature with a different kind of functionality is layers, similar in nature to what more advanced drawing applications provide, where users may choose which layer they want to affect; useful for instance to prevent backgrounds from being erased and add the possibility of setting up opaque colors before-hand, invisible to the other users, for *Line of Sight* purposes. This has also been a much requested feature from many testers but we have regarded it as focusing too heavily on preparatory work, with little to no usability for spontaneous sessions.

Various features for finding database entries or making table lookups would doubtlessly be of use to many games, yet requires too much preparatory work when it comes to inserting the actual data and tables to be interesting for *Tisch*; at least not in its current version. Such features could, for instance, portray characters' health and maximum movement range or perform calculations using the characters' attributes.

Several testers have requested a color filling tool of the same kind as *Adobe Photoshop's*[78] and *MS Paint's*[79] Paint Bucket. If the *Drawing* had been implemented differently, using a grid or pixels, this would have been fairly trivial to do. With the current version however, which uses dots and lines, this becomes a much greater challenge. With respect to this and to the current sketching style that *Tisch* employs we have so far postponed such a feature.

Giving *Tisch* network support for *Storing Game States* between different *Surface* platforms has also been identified as a possible feature. This would enhance the *Tangible Bits*[80] aspect of *Tisch*, and also make it similar to the *Pick and Drop* technique proposed by Rekimoto[81]. However, given the platform's relative rarity, few users will have access to more than one. Somewhat related to this is the extension of using PDAs or tablet computers as private extensions of the game areas, which could mitigate the problem of the *Surface's* limited screen size; while it is large compared to other digital surfaces it is not quite the size of what many games would require, especially wargames. Such an extension could also give a facilitator or game master means of manipulating maps in secret. To port *Tisch* to other platforms would likely offer little, since much of the present functionality is based on the *Surface's* qualities and using it in a public setting with little to no preparatory work; unlike many of the potential platforms to which *Tisch* could be ported.

Procedural asset generation, whether for producing maps automatically or for suggesting/creating scenario events, is another interesting concept which, as with themes, we have yet to develop further.

A whole slew of possible expansions, which we have not investigated more than briefly, lie in optimizing the application. Some easy optimizations have been made, such as only drawing lines which lie within the screen bounds, but even this optimization is far from done as it is presently recalculated every frame, regardless of whether the *Camera* has changed or not. Other similar improvements are to reduce the amount of objects created and disposed of, instead reusing them, and to improve some of the present algorithms, such as for doing line-to-line intersect and the rendering of *Lights*.

See Appendix A for how others can extend and work with *Tisch*.

See Appendix B for a more complete, if bare-boned, list of our ideas for possible expansions.

[78]Adobe Systems (2011). *Adobe Photoshop* [Graphics editing program]. First developed in 1988.

[79]Microsoft (2007). *MS Paint* [Graphics editing program].

[80]Ishii, H. (2008). *Tangible bits: beyond pixels*. In Proceedings of the 2nd international conference on Tangible and embedded interaction (TEI '08). ACM, New York, NY, USA.

[81]Rekimoto, J. (1997). *Pick-and-Drop: A direct manipulation technique for multiple computer environments*, in proceedings of UIST'97, pp. 31-39.

Partial Implementations

Most features present on a generic system such as *Tisch* can, in theory, never be considered complete as there are always possible expansions, improvements, and wider generalizations to be made. *Tisch*, which has been developed under a limited amount of time and a small workforce, is no exception; on the contrary, the underlying architecture of *Tisch* has been designed to be open for others to implement their own features and further expand it with whatever is deemed useful; whether to remove *Excise* or increase *Immersion* for a wide array of uses or for a specific game or activity.

An aspect of this is that most features, while still functional, are currently lacking either in usability, being difficult to use or understand, in their present functionality, with some parts of a feature functioning badly, or in respect to completely new additions. Taking features to the state of proof of concepts, rather than finished products, was a conscious decision to allow us to test more features, finding out more results regarding what might and what might not work well. It was however realized that more work should have been put into the interaction and the interface, making it more intuitive for an inexperienced user to use, especially since ease of use was one of our design goals. Features were easy to use but difficult to learn, which became clear in later tests when we took a smaller part in the activity and the control of the system that it was lacking in this department, since users rarely made use of more advanced functionality.

Examples of such partial implementations are the *Grid's* ability to snap lines to it that only works for square-shaped grids (not hexagon-based ones), *Saving and Loading* without using tags, *Fog of War* runs the risk of causing lag and can be extended to overlay segments of the screen with an actual fog or darkness, and *Day & Night's* current gradients are haphazard and confuse rather than increase *Immersion*.

Suggestions for Similar Systems

Overall, we are satisfied with our primary design choices and user comments show that we are not the only ones with this conviction. We believe that allowing users to select which features to use, and how, as well as not forcing users to follow rules is fundamental for tools aimed at role-playing audiences in particular and analog games in general. It also allowed us very much freedom when it came to the order of feature implementations. The importance of *House Rules* and the ability to make use of selected parts of the system only cannot be overestimated. An example of where this is not the case is the already mentioned *ReacTable Role Gaming*[82], which prides itself on preventing cheating. While not wrong per se, we believe that leaving this decision to players not only lets them have the experience they wish, but help with many auxiliary activities such as setting up examples or teaching new players; but this is much more important in a generic system such as *Tisch* than in *ReacTable Role Gaming's* incorporated game.

As for generality, we believe there is great room for systems which are both more and less generic than *Tisch*. A tool for playing a particular game will probably be better suited for that particular game, e.g. *SurfaceScapes*[83] and Vectorform's *The Settlers of Catan*[84], while *Tisch* is more suited for the general case; providing helpful tools and knickknacks for any game which lacks a specialized tool. Having a generic tool can also have the added benefit of accustoming users to a single interface and a single method of interaction which can be employed for a variety of games and activities; comparable to how the programs in Adobe's *Creative Suite*[85] all share a common style.

On another topic, there are many game types which *Tisch* is not able to handle; or at least not provide any substantial functionality for. Obvious analog games include traditional board games, card games, and, despite some effort from our part, most wargames. Finding a way of providing support

[82]ReacTable Role Gaming (2008). Student project by Viladamot, R. as part of PFC, Universitat Pompeu Fabra, Barcelona. http://ramonviladomat.com/rrg.html (Visited 2011-05-11).

[83]SurfaceScapes (2010). Student project at Carnegie Mellon University, Pittsburgh. http://www.etc.cmu.edu/projects/surfacescapes/ (Visited 2011-05-11)

[84]Vectorform Games (2011). *The Settlers of Catan* [Microsoft Surface version of The Settlers of Catan, original design by Teuber, K.].

[85]Adobe Systems (2011). Application suite. First developed in 2003.

for all types of games, with a more generic system than *Tisch*, exists with applications such as *Vassal*[38], but that is hardly the only way to do it; especially not if constructed for a different platform than PC, in the case of *Vassal*, and *Surface*, in the case of *Tisch*. While on the subject of platforms, another possible group of similar systems would be those targeting a cross-platform solution, whether using the *Surface* as one of the platforms or not. We believe that this can work wonders when it comes to removing *Excise*, but, intuitively speaking, each user having a digital platform of their own will add a lot of *Social Weight*[86].

When it comes to designing the structure of a generic game system, one problem we encountered with *Tisch* was our decision not to include *Grids* in the core module. This meant that some tasks, such as *Fog of War*, became much more cumbersome and expensive. Having a shared *Grid* would also have given all features a common ground, literally, which would be useful for many purposes. That said, note that a *Grid* is no reason to limit entities to be located only at specific points; it should only act as a common ground and a technical aid for the features.

Finally, we believe that the design of applications should take the intended platform into consideration. In the case of *Tisch*, this took the shape of everything from how we regarded *Social Weight* and what was deemed as *Excise* to the interface and the heavy usage of tags. This is a consideration we found very valuable and one which is, most likely, necessary to take in order to bring about the best possible system.

Development Process

The manner by which *Tisch* was developed with several iterative phases, each with its own goal, as described in the Development chapter, has worked fairly well. It allowed us at every iteration to focus on a subset of features, depending on which features would be useful for this phase's goal, rather than drowning in the vast sea of possibilities. Having each phase climax with *Tisch* being capable, to a greater or lesser extent, of servicing one kind of game: the alfa phase had boardgames and the beta phase had role-playing games.

Our way of implementing prototype features first, with only the most bare-boned functionalities and more than once shock full with bugs, allowed us to test a plethora of features at the expense of having a safe program; most testers were unfortunately encountered with obvious bugs or crashes, in turn breaking the mood entirely. Eventually we added more and more protection, from inside *Tisch's* core module, to make errors cause as little damage as possible, but if the *Grid* crashes, refuses to restart, and the *Grid* played a vital role in the on-going test, then of course the test will suffer. Our reasoning is that since *Tisch* is a proof of concept, rather than a commercial product, it was more important to have tests with many and faulty features rather than few and secure.

Similarly, we early on decided to ignore performance; reasoning that since we were developing for a product with a static specification we knew that as long as it run fine on our *Surface* it would run fine on all. As with bugs, we also reasoned that it was more important to have a memory wasteful feature than to not have the feature at all. It was not until the later tests of the beta phase that we began to notice the application's lacking performance, but by then we knew to a much greater detail in what manner *Tisch* should and should not behave; allowing us to easily track down areas we suspected were at fault.

Emergent Behavior

During the tests we encountered several occasions where users discovered or invented new ways of interacting and behaving with the system. A few noteworthy cases follow.

One of the *Dungeons & Dragons*[87] maps, in the test with pre-made maps, depicted the large valley in which the scenario took place. This valley had a couple of small settlements, a few small forests, a

[86]Toney, A., Mulley, B., Thomas, B. H. & Piekarski, W. (2002). *Minimal Social Weight User Interactions for Wearable Computers in Business Suits*, in conference proceedings of IEEE International Symposium on Wearable Computers 2002.

[87]Wizards of the Coast (2008) *Dungeons and Dragons Player's Handbook: Roleplaying Game Core Rules* [Tabletop role-playing games, fourth edition]. Written by Heinsoo, R., Collins, A. & Wyatt, J. Dungeons and Dragons originally published 1974.

river, and some roads. Most commonly, in tabletop role-playing games, players simply tell the game master where they wish to go and, without further ado, the game continues from there; alternatively with an event or two along the way. In this test, however, players chose by themselves to trek along the roads with a zoom level that basically only showed as far as their characters would see. This led to situations where the players arrived at a crossroad and debated as to the best way to take; trying to remember the lay of the land and unsure of remembering the directions they had received earlier. This is a very interesting behavior which neither of us has ever seen before in an analog game, only made possible with the combination of the *Camera* and the pre-made maps. If the game master would have drawn the map as the players advanced across it any crossroad would feel as though it served a much more specific purpose, whereas these roads could simply lead to a small and uninteresting hamlet. Comparing it to digital games is closer but still different as any road in a digital game must lead somewhere, at the very least to a dead end, while an analog game allows the game master to improvise along the way. This situation was a spot-on example of where *Tisch's* hybrid setup worked wonders.

An example where maps drawn as the game progresses is better came during the early *Tunnels & Trolls* and *The Slaughterhouse* tests, where we tried both letting a game master and the players draw. Generally speaking, the tests where players would draw their surroundings themselves were experienced as more enjoyable and *Immersive*; with the possible reason being that *Tisch's* sketching style requires users to employ their imagination, which it is easier to do if the user has control over the presentation of the content she should fantasize about.

In the *Frag* test we noted the possibility of making changes to the map while playing, opening up for completely new game behaviors. Eventually we decided not to, in the test that was run, but it shows, along with the usage of *Line of Sight*, in what way boardgames can evolve with *Tisch*.

As mentioned previously, players in several different tests organized the activity by zooming in and out; examples being finding out where they were going, where they had been, and what their surroundings looked like. This allowed players to make higher precision movements close up and get overviews from afar, an example of Bederson's interface paradigm[88] of utilizing zoomable interfaces to structure the interaction.

One tester layered *Playgrounds* on top of one another to display different information on different layers, while simultaneously preventing the background from being edited when *Drawing*.

In tests with multiple users, primarily the *Dungeons & Dragons* tests, players would divide the tasks among themselves naturally; one would handle the *Camera*, another the *RoundKeeper*, a third the *Drawing* and the *Drawing Tools*, and so on. This made the experience more fluent and gave each user a unique responsibility.

During the first *Dungeons & Dragons* test, as well as to a lesser degree in other tests, bored users would sometimes use *Tisch's Drawing* to entertain themselves by sketching. Such possibilities, along with individual *Playgrounds* and features such as *Pong*, can help alleviate drawn-out activities by offering users without any current influence, e.g. if it is not their turn, something to do. The fact that users became bored, we reason, has less to do with *Tisch* and more to do with the supported activity.

[88]Bederson, B. B., Grosjean, J., & Meyer, J. (2004). *Toolkit Design for Interactive Structured Graphics*, IEEE Transactions on Software Engineering, 30 (8), pp. 535-546.

Conclusions

With this Master's thesis project it has been attempted to design a generic tool for the *Microsoft Surface* to aid and enhance tabletop gaming, as well as research various aspects around such a tool. The tool aims to assist in the gaming activity without any restrictions to the player's freedom or in any other way hinder the original activity.

The system supports multi-user *Drawing* of maps with *Camera* control, multiple *Playgrounds*, *Tokens*-based interaction, *Saving and Loading* functionality, tools for *Measuring*, a *RoundKeeper*, checking *Line of Sight*, creating *Fog of War*, and adding *Lights*, *Day & Night*, *Weather*, and *Backgrounds*. The various features are independent of each other and users can freely choose at any time which to use and how to use them.

The design goals described earlier state that *Tisch* should be easy to use, not require preparations, allow *House Rules*, keep *Social Weight* low, reduce *Excise* and enhance the activity with *Immersive* features. At the system's current state we believe that we have kept all of these goals, at the expense of most features being limited in regards to present functionality and usability. Tests with role-playing games showed that there is interest for systems such as *Tisch*, users being attracted by the ease of *Creating Game Worlds*, reducing *Excise*, and adding *Immersion*, although more features would need to be added; primarily features supporting the specifically targeted game(s). Tests with wargames showed that, while the interest and need exists, *Tisch* lacks some fundamentals in order to be of use; primarily smooth means for creating and managing large number of units, having a larger physical area, and understanding elevation with a three-dimensional *Line of Sight* feature. Tests with boardgames showed that *Tisch* can be used to more easily create variation and aid players in keeping track of some data, but its use is limited since most boardgames come equipped with their own game plan.

Overall, to support any kind of game to a higher degree than the current requires a system to read in, understand, and present the game's data and, albeit to a much lesser extent, rules; considering that both role-playing games and wargames have complex rules and large amounts of abilities, values, and other table- and list oriented data. The fact that *Tisch* prioritized *House Rules* and allowed users to freely select what to do and how to do it is paramount to the success of any supporting system and we believe that *Tisch* shows one way for computer applications to support gaming without controlling it and thereby form a fruitful combination of technology and the social experience of tabletop gaming.

Bibliography

Abt, C.C. (1970). *Serious Games*. New York: Viking Press.

Bederson, B. B., Grosjean, J., & Meyer, J. (2004). *Toolkit Design for Interactive Structured Graphics*, IEEE Transactions on Software Engineering, 30 (8), pp. 535-546.

Björk, S., Bergström, K. & Jonsson, S. (2010) *Undercurrents - A Computer-Based Gameplay Tool to Support Tabletop Roleplaying*, in proceedings of Nordic DiGRA 2010.

Björk, S., Ljungstrand, P. (2007). *Pirates! Using the Physical World as a Game Board*. Space Time Play. On the Synergy Between Computer Games, Architecture, and Urbanism.

Eriksson, D., Peitx, J. & Björk, S. (2005). *Socially Adaptable Games*, Lightning round presentation at Changing Views: Worlds in Play, DiGRA conference 2005.

Fine, G. (1983). *Shared Fantasy: Role-Playing Games as Social Worlds*. The University of Chicago Press, Chicago.

Gravelyn, N. (2010). *EasyConfig [*Library for reading config files]. Link: easyconfig.codeplex.com (Visited 2011-05-11).

von Hilgers, P. (2000). *Eine Anleitung zur Anleitung. Das taktische Kriegsspiel 1812-1824*, in Board Games Studies no.3, pp. 59-77.

Ishii, H. (2008). *Tangible bits: beyond pixels*. In Proceedings of the 2nd international conference on Tangible and embedded interaction (TEI '08). ACM, New York, NY, USA.

Jamil, I., Alexander, J., Subramanian, S., Barnes, S. (2010). *Talking Teenagers and Tables: Communication Styles of Teenagers and Interactive and Non-Interactive Tables*. University of Bristol, UK.

Jamil, I., O'Hara, K., Perry, M., Karnik, A., Subamanian, S. (2011). *The Effects of Interaction Techniques on Talk Patterns in Collaborative Peer Learning around Interactive Tables*. CHI, Vancouver, BC, Canada.

Juul, J. (2005). *Half-Real: Video Games between Real Rules and Fictional Worlds*. The MIT Press, Cambridge, 2005.

Mackay, D. (2001). *The Fantasy Role-Playing Game: A New Performing Art*. McFarland & Company.

Magerkurth, C., Cheok, A.D., Mandryk, R., Nilsen, T. (2005). *Pervasive Games: Bringing Computer Entertainment Back to the Real World*. ACM 2005.

Magerkurth, C., Memisoglu, M., Engelke, T., & Streitz, N. (2004). *Towards the next generation of tabletop gaming experiences*. In proceedings of Graphics Interface '04, ACM International Conference Proceeding Series, Vol. 62, Canadian Human-Computer Communications Society, 2004, ISBN 1-56881-227-2, pp. 73-80.

Mandryk, R., Maranan, D. & Inkpen, K. (2002). *False Prophets: Exploring Hybrid Board/Video Games*. CHI, Minneapolis, Minnesota, USA.

Mazalek, A., Mironer, B., O'Rear, E., & van Devender, D. (2007). *The Tviews Table Role-Playing Game*. Synaesthetic Media Lab, GVU Center, Georgia Institute of Technology, USA.

Montero, C., Alexander, J., Marshall, M., Subramanian, S. (2010) *Would you do that? – Understanding Social Acceptance of Gestural Interfaces*. Mobile HCI, Lisbon, Portugal.

Müller-Tomfelde, C. & Fjeld, M. (2010). *Introduction: A Short History of Tabletop Research, Technologies, and Products*. In Müller-Tomfelde, C. (ed.) Tabletops - Horizontal Interactive Displays. Human Computer Interaction Series, Springer Verlag, pp. 1-24.

Peitz, J., Björk, S. & Jäppinen, A. (2006). *Wizard's Apprentice - gameplay-oriented design of a computer-augmented board game*. Paper at Advancements in Computer Entertainment, Hollywood, USA, 2006.

Rekimoto, J. (1997). *Pick-and-Drop: A direct manipulation technique for multiple computer environments*, in proceedings of UIST'97, pp. 31-39.

Salen, C. & Zimmerman, E. (2004). *Rules of Play: Game Design Fundamentals*. The MIT Press.

Surface official documentation. Setting the time-out durations. Link: http://technet.microsoft.com/en-us/library/ee692022(Surface.10).aspx (Visited 2011-05-11).

Sjöblom, B. (2008). *The Relevance of Rules: Negotiations and Accounts in Co-operative and Co-located Computer Gaming*. Proceedings of the [player] conference, IT University of Copenhagen 2008, August 26-29, 2008, pp. 335-378.

Spolsky, J (2004). *Joel on Software: And on Diverse and Occasionally Related Matters That Will Prove of Interest to Software Developers, Designers, and Managers, and to Those Who, Whether by Good Fortune or Ill Luck, Work with Them in Some Capacity*. Apress.

Toney, A., Mulley, B., Thomas, B. H. & Piekarski, W. (2002). *Minimal Social Weight User Interactions for Wearable Computers in Business Suits*, in conference proceedings of IEEE International Symposium on Wearable Computers 2002.

Weiser, M. & Brown, J. S. (1996). *The Coming Age of Calm Technology*. Available from http://www.johnseelybrown.com/calmtech.pdf (Visited 2011-05-03).

Weiser, M. (1991). *The Computer for the Twenty-First Century*, Scientific American, pp. 94-10, September 1991.

Well, H.G. (2004). *Little Wars (A Game for Boys from twelve years of age to one hundred and fifty and for the more intelligent sort of girl who likes boys' games and books)*. Kessinger Publishing. Originally published 1913.

Woods, S. (2010). *Convivial Conflicts: The Form, Culture and Play of Modern European Strategy Games*. PhD thesis, School of Media, Culture and Creative Arts, Curting University of Technology.

Ludography

Adobe Systems (2011). *Adobe Photoshop* [Graphics editing program]. First developed in 1988.

Adobe Systems (2011). Application suite. First developed in 2003.

Atari (1972). *Pong* [Arcade game]. Designed by Alcorn, A.

Blizzard Entertainment (2011). World of Warcraft [Online role-playing game].

Chvátil, V. (2008). *Space Alert* [Board game].

Columbia Games (2004). *Rommel in the Desert* [Wargame, second edition]. Originally published 1982. Designed by Desinque, C.

Compete Now AB (2009), *M.I.G. - Mobile Intelligence Games* [Quiz Game, iPhone version]. Originally published 2001.

Flying Buffalo (2007). *Tunnels & Trolls* [Tabletop role-playing games, fifth edition]. Written by St. Andre, K. Tunnels & Trolls originally published 1975.

Games Workshop (2008). *Warhammer 40,000* [Tabletop miniature wargame, fifth edition]. Originally published 1987.

GothCon (2011). *GothCon XXXV* [Annual role-playing game convention]. Gothenburg.

Kinney, R. (2002). *Virtual Advanced Squad Leader*.

Knizia, R. (2003). *King Arthur* [Board game].

Microsoft (2007). *MS Paint* [Graphics editing program].

Microsoft (2008). *Microsoft Surface* [Interactive tabletop]. http://www.microsoft.com/surface/ (Visited 2011-05-11).

Multi-Man Publishing (2001). *Advanced Squad Leader Rule Book* [Tactical-level board wargame, second edition]. Original design by Greenwood, D.

Mozilla (2011). Firefox [Web browser].

Neogames (2004). *Eon* [Tabletop role-playing game, third edition]. Originally published 1996.

Paizo Publishing (2009). *Pathfinder Roleplaying Game* [Tabletop role-playing game].

ProFantasy Software Ltd. (2006). *Campaign Cartographer*.

Rackham (2007). *Confrontation: The Age of the Rag'narok* [Tabletop miniature wargame, fourth edition]. Originally published 1996.

ReacTable Role Gaming (2008). Student project by Viladamot, R. as part of PFC, Universitat Pompeu Fabra, Barcelona. http://ramonviladomat.com/rrg.html (Visited 2011-05-11).

RPTools (2006). http://www.rptools.net/ (Visited 2011-05-11).

Steve Jackson Games (2004). *GURPS Basic Set* [Tabletop role-playing games, fourth edition]. Written by Jackson, S., Punch P. & Pulver, D. Edited by Hackard, A. & Jackson, S. GURPS originally published 1986.

Steve Jackson Games (2001). *Frag* [Board game first-person shooter]. Designed by Jackson, S. & Reed, P.

SurfaceScapes (2010). Student project at Carnegie Mellon University, Pittsburgh. http://www.etc.cmu.edu/projects/surfacescapes/ (Visited 2011-05-11).

The Vassal Team (2010). *Vassal* [Generic online boardgame engine].

Vectorform Games (2011). *The Settlers of Catan* [Microsoft Surface version of The Settlers of Catan, original design by Teuber, K.].

Wizards of the Coast (2008) *Dungeons and Dragons Player's Handbook: Roleplaying Game Core Rules* [Tabletop role-playing games, fourth edition]. Written by Heinsoo, R., Collins, A. & Wyatt, J. Dungeons and Dragons originally published 1974.

Wizards of the Coast (2010). *Reavers of Harkenworld* [*Dungeons & Dragons* scenario]. Designed by Baker, R. and Perkins, C.

White Wolf (2004). *The World of Darkness: Storytelling System Rulebook* [Tabletop role-playing game, second edition]. Written by Bridges, B., Chillot, R., Cliffe, K. & Lee, M.

Appendices

Appendix A - Passing the Torch

This section is intended for anyone interested in continuing development on *Tisch* or merely to use it (in which case only the next paragraph will be of interest). From the beginning, we hoped that it would be possible to pass *Tisch* on to the creative and productive hands of others once our time was up. If you think that you have what it takes to improve, expand, or just set things straight, we give you a resounding "Yes!" for approval.

Tisch presently runs on the *Surface*. In order to get it running you will need:

- 1 Surface.
- Tisch.dll and the feature assemblies in a folder on the *Surface*, along with the Tisch.xml (edited to have the same folder for its ExecutableFile, IconImageFile, and PreviewImageFile tags) moved to the *Surface's* %PROGRAMDATA%\Microsoft\Surface\Programs folder.
    - See http://msdn.microsoft.com/en-us/library/ee804900(v=surface.10).aspx (Visited 2011-05-09) for more thorough descriptions of deploying applications.
- Something to do with it.

The first requirement to get started is the correct versions of the frameworks: Surface Core API 1.0, XNA Framework 4.0, and C# 4.0. We have used *Microsoft Visual Studio 2010* for development, which is the program compatible with the structural source files of types .csproj and .sln, but you are free to use whatever you are the most comfortable with.

Features are generally their own projects, with references to Tisch.dll (aside from the already mentioned frameworks). Tisch.dll contains three namespaces:

- Drawing, containing anything and everything related to rendering graphics.
- Features, containing the abstract Feature.cs class which your own features should inherit from, the *TagPrompter* and *Tokens* features which are parts of the core module, and some various minor classes aimed at supporting features. Inside this namespace lies a separate namespace specifically for the *Tokens* feature, of interest to features which intend to interact with it.
- Tools, containing a variety of types to instantiate and use, as well as a static Helper.cs class with a set of useful methods.

Most likely, any new feature is interested in using all three namespaces. First and foremost, a new feature should inherit from Tisch.Features.Feature, giving it all the necessary tools. New features should have a constructor where they simply pass along two arguments; a *Draw Area* and a name, both of which are set and handled automatically. Any values that a feature is interested in loading and saving automatically should be properties with the custom FeatureInfo attribute. Note that FeatureInfo has several different constructors, each with its own purpose.

Inheriting from the abstract Feature class provides new features with a variety of interesting methods to override. These are as follows:

- Rendering methods. These are divided into RenderScreen, RenderWorld, RenderPrimitive, RenderEffects, and RenderEffectsPreparation.
    - RenderScreen works with screen coordinates and is rendered top-most, which is perfect for interfaces and the like.
    - RenderWorld works with world coordinates and is rendered bottom-most, which is perfect for anything drawn as part of the "world" or game.
    - RenderPrimitive also works with screen coordinates and is rendered top-most, but is intended for drawing primitives; i.e. making direct line and triangle rendering calls to the graphics card. This has been separated from RenderScreen due to problems with the order in which they were rendered.

    - RenderEffects are for shaders and other overlaying graphical effects, being called after RenderWorld (allowing modification of what has been rendered) but before RenderScreen and RenderPrimitive (not obscuring the interface).
    - RenderEffectsPreparation is a complementary render call for RenderEffects, being called before anything has been drawn to the screen in case preparations or comparisons are needed.
- Update method, taking the elapsed time since the last frame as argument. Changes to collections and other modifications should preferably be done here, instead of in the contact methods, due to concurrency issues.
- Contact methods, divided into separate sets of methods for fingers, objects, and tags. Each set in turn contains a method for when a contact is added, changed, and removed, as well as if a tap or a hold is detected. A tap is when a small amount of time has elapsed between a contact's add and remove, without much change in position in-between. A hold is when a contact has been added, followed by no remove and small or no changes during the next two seconds.
- The matching Save and Load methods allow features to save and load whatever data they wish. As the methods are generic enough to handle any kind of data, features must also use the AddSaveType method to specify what types they wish to save. This is due to the nature of the C# XmlSerializer. A desired improvement for *Tisch* is to remove this requirement of using AddSaveType but we have not yet had the time to spend on finding a better solution.
- The *TagPrompter* feature automates some parts of assigning tags to features but since features may use multiple tags or wish to perform additional tasks when a tag is set, the SetTag method must be overriden and implemented by features wishing to use tags for specific purposes. SetTag's base method handles saving the assigned value to the settings file so that users only have to set tags the first time a feature runs.
- The Dispose method is called when a feature is turned off, for whatever reason, and allows features to release resources, undo changes they have made to the core module, or whatever else might be desired. Note that most destructor work is handled automatically by the garbage collector and, as such, does not need to be performed here.

The features' assemblies should be built, or at least placed, into *Tisch's* bin\x86\debug\ directory, from which they will be automatically loaded. Any assets should simply be copied into the same directory and loaded in the feature's constructor or some other reasonable place. The present features only load graphical assets and to load other types the core module may have to be extended or the feature in question may need to handle it by itself.

Appendix B - Possible Expansions

Improvements - Core module

- General
    - Be able to move things, e.g. togglers and the RoundKeeper's tracker.
    - Is it possible to have gesture recognition?
    - Restrict features from using more than a set amount of CPU and memory, to ensure that the rest of program runs fine?
- Eavesdropper
    - Get rid of OnPropertyChanged() in the features' setters.
    - You still want the possibility (e.g. Grid.GridType might want to send a message as GridSize).
    - Is it possible to combine multiple senderProps to the same message? (e.g. Grid.SquareSize and Grid.HexSize so that listeners only have to listen to Grid.GridSize).

    - Fairly solved at present by creating a new property which is updated whenever either of the relevant properties are updated, but how is this changed in case explicit messages are no longer sent?
- TagPrompter
    - Be able to ignore a prompted tag, in case users don't want to set a tag for it.
- Tokens
    - When do they disappear?
    - Use tokens with fingers, i.e. without being forced to use tagged objects.
    - Give tokens names.
    - Snap-to-grid with the token textures?
    - Rename/change a numeric value, update all Tokens.
    - Change/display attributes from a different DA?
    - Tokens might want to be static across all DA's, to be able to be accessed from all of them.

Improvements - Features

- Backgrounds
    - Automatically load images from a directory.
    - Make a GUI to select and change backgrounds more easily.
- Configurer
    - Make it usable from multiple directions or movable/rotatable?
    - HasToggler should add/delete the toggler immediately, but is at present only used in the feature's constructor.
    - Make your own shortcuts by dragging settings out of the Configurer, to be altered at will later even after closing the Configurer.
    - ResetToDefault probably shouldn't reset Active.
    - It is possible, somehow, to get a bug where the Configurer GUI jumps in and out.
    - Allow configuring values which shouldn't be saved?
    - Allow saving values without showing them in the Configurer?
- Erase
    - Set the eraser size by spinning? Or some other way which is easier than going through the Configurer.
- Fog of War
    - Walls, doors, windows? How to do it?
    - Optimize. It currently lags when there are too many lines about.
    - Add a misty layer on top of everything that has been seen but is not currently seen? And a dark layer on top of everything that has never been seen?
    - How to handle objects other than lines?
- Grid
    - Snap-to-grid doesn't take zoom into account.
    - Snap-to-grid for hexagonal grids.
    - Isn't friends with Rotation.
    - Fix so that zoom and pan works as well for backgrounds and other things.
    - Square grids (not Corners or SquareTextures) crashes. Fix that!
- Lights
    - Block with lines; lightpoints can presently be seen through walls and stuff.
- Line of Sight
    - Optimize.
    - Choose what is opaque and transparent through different layers instead of colors? At least add the option of doing it that way.
    - Add measuring with number of grid squares.
- Palette
    - Show the current color.

    - Draw with multiple colors simultaneously? How?
    - Choose colors by clicking with a finger in the palette? Add the option, don’t remove the existing option.
- Playgrounds
    - Create, modify, and move things from one DA to another?
    - Presently DA’s cannot be removed for real, but are just hidden.
    - Which features are present in new DA’s is hard-coded into DrawArea.cs. The Playgrounds feature should manage this.
    - Effects in new DA’s get positioned in the top-left because, maybe, it resets its Viewport’s position when a RenderTarget is set.
    - Be able to split the alfa DA in two so that both get “equal” status?
- Rotation
    - Rotating around a point, instead of the center.
    - At present Rotation has been largely disregarded during the second half of the project, because it was messy and irrelevant.
- Save
    - Use auto-save to a special save file that can be loaded with a button/field from Save/Load, in case Tisch crashes or something else unforeseen happens.
- Weather
    - Shaders/effects? Get them to look better overall.
    - Add sound?
- Zoom
    - Zoom around a point, instead of the center.
    - Interface showing the current zoom level?

New features

- Chat
    - Integrate some kind of chat system.
    - Combine with the Undercurrents system mentioned earlier in the report?
- Copy
    - Some way of encircling/choosing something from the screen and copying it, both to use it straight away and to add it to some collection for future use.
- Counters & Effects
    - Add a graphical notation for Tokens for keeping track of some counter or whether this or that effect is active.
- Database
    - Access an external database with abilities or other relevant data and present it in some way.
- Dice
    - Handle dice rolling.
    - Read physical dice? Read tagged dice? Just simulate dice rolling? Something else?
- Floor levels
    - Add some way of displaying/using different floor levels.
    - Combine with Layers?
- Link maps
    - Load a map automatically when performing some kind of interaction.
    - Create a temporary DA with the linked map if a token steps on the linkpoint?
- Lock areas/Layers
    - Be able to lock certain parts of the map from change.
- Paint bucket
    - Fill an area quickly with a single color.
- Player data
    - Display some kind of data related to the player or the player characters.

- Position shortcuts
    - Method of quickly switching between various locations.
- Quickrooms/Geometry
    - Quick way of creating rooms and/or standard geometry, e.g. rectangles, circles.
- Room abilities
    - Create events which will automatically trigger when players enter a room or an area.
- Scenario randomizer
    - Create scenarios, events, characters, encounters, and whatever may be interesting on the fly, to encourage spontaneous play sessions and reducing the amount of preparatory work done.
- Sound
    - Make it possible to have playlists with various songs in them; possibly creating playlists out of directory names in the feature and adding whatever music files lie in each directory as songs?
    - Access and make use of Spotify? Would be cool.
    - Sound effects?
    - Combine with Themes?
- Stamps
    - Using stamps with images and placing them on the screen.
    - Combine with Copy?
- Textures
    - Select among textures what the drawn lines or filled areas will look like.
    - Combine with Themes?
    - Combine with Paint bucket?
- Themes
    - Cause graphical changes, GUI changes, filters, or whatnot to better set the mood.
    - Theme tools, to create themes of your own?
- Tables
    - Look up values from tables.
    - Combine with Dice and Tokens?
- Upload maps
    - Upload image files as background maps, such as a city layout.
- Write
    - Type things in with the Surface on-screen keyboard or some other better suited method of writing than with your fingers.