WorkPackage WP9: Socially Adaptable Games

Deliverable D9.5: Socially Adaptable Game Prototypes

Johan Peitz, Interactive Institute
Daniel Eriksson, Interactive Institute
Staffan Björk, Interactive Institute

Release date: 11 October, 2005
Status: public
EXECUTIVE SUMMARY

This document describes the prototypes designed, developed, and implemented during phase one of work package 9, Socially Adaptable Games. Specifically, the prototypes Augmented Diamond Hunt, Wizard’s Apprentice, and Anthills are covered with information about the research goals, the original design, and the final design with details on hardware, software architecture, and rationales for the design changes made. Material is provided to describe the intended gameplay and game experiences as well as supporting reproduction of the prototypes by interested parties.

The deliverable is intended to provide a basis for the evaluation document D9.6 in the Socially Adaptable Games work package as well as provide input to work packages 5, 6, and 7.
Deliverable Identification Sheet

IST Project No.  FP6 – 004457
Acronym  IPerG
Full title  Integrated Project on Pervasive Gaming
Project URL  http://iperg.sics.se/
EU Project Officer  Albert GAUTHIER

Deliverable  D9.5  Terminology Report
Work package  WP9 Socially Adaptable Games

Date of delivery  Contractual M 12  Actual 14 October 2005
Status  version. 1.0  final ?
Nature  Prototype  Report  Dissemination  
Dissemination Level  Public  Consortium  

Authors (Partner)  Johan Peitz (Interactive Institute), Daniel Eriksson (Interactive Institute), Staffan Björk (Interactive Institute)
Responsible Author  Johan Peitz  Email  johan.peitz@tii.se
Partner  II  Phone  +46 31 18 30 15

Abstract (for dissemination)  This deliverable describes the three prototypes implemented in phase 1 of work package 9, Socially Adaptable Games. Only one major prototype was specified for the deliverable but due to the exploratory work done within the work package the results from all three prototypes are described as they have influenced each other and provide a richer description of the overall work. The aim of the prototypes were to explore and gain a richer understanding of social adaptability, or how artifacts can by design actively or passively adapt to suit changing social environments, which has been identified in the project as a theme for game designs within IPerG. By augmenting board games, the work package began with a well-established game type to explore the concept. The
results gained from this exploration will be applied to the novel game type of pervasive games in the second phase of the show case.

**Keywords**
Design, pervasive games, social adaptability, board games

<table>
<thead>
<tr>
<th>Version Log</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Issue Date</strong></td>
</tr>
<tr>
<td>20-09-05</td>
</tr>
<tr>
<td>11-10-05</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

**EXECUTIVE SUMMARY** ............................................................................. 1

**TABLE OF CONTENTS** ........................................................................... IV

**1 INTRODUCTION** ....................................................................................... 1

1.1 Structure of prototype description .......................................................... 1
1.1.1 Research Aim ......................................................................................... 2
1.1.2 Overview of Game ................................................................................... 2
1.1.3 Game Instructions ................................................................................... 2
1.1.4 Technical Requirements and General Architecture ................................. 2
1.1.5 Software Architecture ............................................................................ 2
1.1.6 Hardware Architecture ........................................................................... 2
1.1.7 Content .................................................................................................. 2
1.1.8 Changes from Original Design Document .............................................. 2
1.1.9 Links to external annexes ......................................................................... 2
1.1.10 Literature & Terminology ....................................................................... 3

**2 AUGMENTED DIAMOND HUNT** .............................................................. 4

2.1 Research Aim ............................................................................................ 4
2.2 Overview of Game ....................................................................................... 5
2.3 Game Instructions ........................................................................................ 6
2.3.1 Game Progression .................................................................................... 6
2.3.2 Movement ............................................................................................... 7
2.3.3 Searching ................................................................................................. 8
2.4 Technical Requirements and General Architecture ...................................... 8
2.5 Software Architecture ................................................................................ 9
2.6 Hardware Architecture .............................................................................. 9
2.7 Content ...................................................................................................... 10
2.7.1 Site Location Design .............................................................................. 10
2.7.2 Sound Design ........................................................................................ 12
2.8 Changes from Original Design Document ............................................... 12
2.9 Links to external annexes ........................................................................... 12
2.10 Literature & Terminology .......................................................................... 12

**3 WIZARD’S APPRENTICE** ........................................................................ 14

3.1 Research Aim ............................................................................................ 14
3.2 Overview of Game ....................................................................................... 14
3.3 Game Instructions ....................................................................................... 15
3.3.1 Sessions ................................................................................................. 17
3.3.2 Guidance and Story arcs ........................................................................ 20
3.4 Technical Requirements and General Architecture ........................................ 21
3.5 Software Architecture .................................................................................. 21
  3.5.1 Event Manager ....................................................................................... 22
  3.5.2 Game Engine ........................................................................................ 23
  3.5.3 Play Modes ........................................................................................... 23
  3.5.4 Soundbox .............................................................................................. 23
  3.5.5 Site Parser ............................................................................................ 23
  3.5.6 UI ........................................................................................................ 23
  3.5.7 Utils for authoring and debugging ......................................................... 24
3.6 Hardware Architecture .................................................................................. 25
3.7 Content ........................................................................................................ 29
  3.7.1 Graphical design of game board ............................................................. 30
  3.7.2 Graphical design of Icons ..................................................................... 31
  3.7.3 Sound Design ........................................................................................ 32
3.8 Changes from Original Design Document ..................................................... 32
  3.8.1 Example of gameplay ........................................................................... 38
3.9 Links to external annexes ............................................................................ 39
3.10 References .................................................................................................. 39

4 ANTHILLS ........................................................................................................ 40
  4.1 Research Aim .............................................................................................. 40
  4.2 Overview of Game ...................................................................................... 40
  4.3 Game Instructions ....................................................................................... 40
    4.3.1 Overall structure .................................................................................. 40
    4.3.2 Game Progression ............................................................................... 41
    4.3.3 Player Activities .................................................................................. 41
    4.3.4 Non-player activities ......................................................................... 42
  4.4 Game Objects .............................................................................................. 43
    4.4.1 Units ................................................................................................... 43
    4.4.2 Game board ......................................................................................... 44
    4.4.3 Physical obstacles ............................................................................... 44
    4.4.4 Public screen ....................................................................................... 44
  4.5 Technical Requirements and General Architecture ...................................... 44
  4.6 Software Architecture ................................................................................ 45
  4.7 Hardware Architecture .............................................................................. 45
    4.7.1 Augmented Game Tile ....................................................................... 46
    4.7.2 Controller ........................................................................................... 46
    4.7.3 Token ................................................................................................ 46
    4.7.4 Master Controller ............................................................................... 47
4.8 Content ................................................................................................................. 47
4.9 Changes from Original Design Document ......................................................... 47
4.10 Links to external annexes .................................................................................. 48
4.11 References .......................................................................................................... 48

5 CONCLUSIONS ........................................................................................................ 48

REFERENCES ............................................................................................................ 48

APPENDIX I: COMMUNICATION PROTOCOL FOR WIZARD’S APPRENTICE ........ 49
APPENDIX II: EXAMPLE SITE DEFINITION FOR WIZARD’S APPRENTICE ........... 50

LIST OF FIGURES

Figure 1 - Software Architecture of Augmented Diamond Hunt ............................... 9
Figure 2 – Schematic view of technology use in Diamond Hunt ............................... 10
Figure 3 – Schematic overview of software architecture for Wizard’s Apprentice ...... 22
Figure 4 – Schematic view of technology use in Wizard’s Apprentice ....................... 26
Figure 5 – Architecture for card and die reader ....................................................... 27
Figure 6 – Schematic for position module ............................................................... 27
Figure 7 – Schematics of interconnections between modules (not all module are shown). ................................ ................................ ................................ ............. 28
Figure 8 - Spawn points in Anthills ........................................................................... 41
Figure 9 - Pushing the queen .................................................................................... 43
Figure 10 - The Anthills game board fitted with physical obstacles ............................ 44
Figure 11 - Overview of Anthills augmented board game infrastructure ................. 45
Figure 12 - Schematic for Anthills game tile ............................................................. 46
Figure 13 – Schematic for Anthills game token ....................................................... 47
Figure 14 - Schematic for Anthills master controller .............................................. 47

LIST OF IMAGES

Image 1- Diamond Hunt game board ................................................................. 5
Image 2 - Example sculpture .................................................................................. 6
Image 3 - Game money (credits) ............................................................................ 6
Image 4 - Different ways to move .......................................................................... 7
Image 5 - Player must roll two or less to continue ............................................... 7
Image 6 - A player searching a site with an experimental magnifying glass ............ 8
Image 7 – Concept art for location sites in Augmented Diamond Hunt ................ 10
Image 8 – Further concept art for location sites in Augmented Diamond Hunt ...... 11
Image 9 – Five clay sculptures and a RFID tag used in Augmented Diamond Hunt... 11
Image 10 – Initial setup of clay sculptures for gameplay. .............................. 12
Image 11 – Example of gameplay situation.................................................. 12
Image 12 – Figurines representing the Apprentices and the Wizard................... 16
Image 13 – The die used in Wizard’s Apprentice.......................................... 16
Image 14 – Close-up of the die pit and information site............................... 17
Image 15 – Typical overview during the Wizard’s phase............................... 18
Image 16 – Typical view of a quest during the Wizard’s phase....................... 18
Image 17 – Information about the apprentice Vladimir at the beginning of a game 19
Image 18 – Apprentice Vladimir succeeding with a charm challenge................ 20
Image 19 – A typical presentation after apprentices have returned to the castle..... 20
Image 20 - The Event Manager Control tool................................................. 24
Image 21 - The Debug Board tool............................................................... 25
Image 22 - The Site Editor tool................................................................. 25
Image 23 – Inner details of the game board for Wizard’s Apprentice.................. 28
Image 24 – Close-up on some position modules, all control modules, die pit, and external USB connector.............................................................. 29
Image 25 – The Wizard’s Apprentice board half-folded................................... 29
Image 26 – The drawing for the pirates’ hideout......................................... 30
Image 27 – The drawing for the location site Port of Call.............................. 30
Image 28 – The complete game map........................................................... 31
Image 29 - Icon used to signify befriend mission........................................... 31
Image 30 - Icon used to signify spy mission................................................ 32
Image 31 - Icon used to signify confront mission......................................... 32
Image 32 – Early map design for Wizard’s Apprentice.................................... 33
Image 33 – Early prototyping with physically manifested game state.................. 34
Image 34 – Exploring the interaction model for the tangible interface............. 35
Image 35 – The original prototype die for Wizard’s Apprentice...................... 35
Image 36 – First prototype of the final die design for Wizard’s Apprentices........ 35
Image 37 – Low Fidelity prototyping of the algorithm controlling the spread of evil 36
Image 38 – Low Fidelity prototyping of commodity distribution..................... 36
Image 39 – Early version of a play mode presentation................................... 37
Image 40 – Example gameplay during quest allocation.................................. 38
Image 41 – Example gameplay just before trying to overcome a challenge........ 38
Image 42 – Example gameplay of the last quest in a session being completed..... 39
1 INTRODUCTION

This deliverable describes the three prototypes implemented in phase 1 of work package 9, Socially Adaptable Games. Only one major prototype was specified for the deliverable but due to the exploratory work done within the work package the results from all three prototypes are described as they have influenced each other and provide a richer description of the overall work. The aim of the prototypes were to explore and gain a richer understanding of social adaptability, or how artifacts can by design actively or passively adapt to suit changing social environments (see [Björk04] for a longer description), which has been identified in the project as a theme for game designs within IPerG [Montola04]. By augmenting board games, the work package began with a well-established game type to explore the concept. The results gained from this exploration will be applied to the novel game type of pervasive games in the second phase of the show case.

Although the prototypes are implementations of different games, they have been created sequentially to allow continuous use of resources and to be able to make use of experience gained in previous prototypes. This has led to the last prototype being completed during the last months of phase 1, and thereby the description of that prototype is partially what has been created and partially what will be created after the completion of this deliverable. The process of choosing prototypes to implement, including brief mentions of the concepts not chosen, can be found in deliverable D9.2 [Peitz05].

The descriptions of the prototypes are given a section each in the deliverable, and make use of the same structure to easy comparisons.

1.1 Structure of prototype description

The description of the prototypes are structured to allow quick access to material related to the research goals and design process, the instruction how to play the game and what resources are required, software and hardware architecture, and examples of content in the prototypes. Parts of the sections describing each prototype will overlap the descriptions in deliverable D9.2 [Peitz05] and D9.4 [Ericsson04] since these deliverable contained the initial design documents and infrastructure specification of the prototypes. However, the descriptions are repeated here to give a complete description of the implemented prototypes as well as reflect differences from the design documents.

As per recommendations in the guidelines of deliverable D5.1 [Björk04], game design patterns [Björk05] are used to describe the game and patterns are indicated by using bold in the text descriptions. The structure below is modeled upon the suggested structured in appendix 2 of deliverable D5.1 but has been modified to better suite the specific type of games developed. In particular, the section of pervasiveness in design and game world section have been removed since explicitly developing these have not been the main objectives of the prototypes (social adaptability taking the place of pervasiveness and permeating the designs). The section on technical setting has also been divided into a software section and a hardware section. Deliverable D9.2 contains descriptions of the original game designs that do follow the structure suggested in D5.1.
1.1.1 Research Aim
To frame each prototype within work package 9, and within IPerG as a whole, the specific research issues that underlie the game design is described first. The research issues are presented in relation to the major and minor themes identified in D5.1 as well as the guidelines developed in D9.1, guidelines for socially adaptable games.

1.1.2 Overview of Game
After the research aims of the game have be presented, an overview of the design game is presented as it would be to a novice player of the game. A focus is placed upon explaining the overarching goals, common tasks and required skills, and the game theme.

1.1.3 Game Instructions
This section describes, from a player’s perspective, how to play the actual game. In contrast to the overview, this description focuses upon game rules and mechanics.

1.1.4 Technical Requirements and General Architecture
Being augmented board games, all the prototypes have technological requirements that have to be fulfilled in order for the game to be playable. This section describes what those requirements are and the general solution chosen.

1.1.5 Software Architecture
New software has been developed for each prototype ranging from simple systems, primarily handling input and output to handle a small subset of the game state, to complex systems, handling a multitude of input sources and maintaining complex game states which are internally updated without players’ knowledge. The description of the software architecture is presented so that it should be able to replicate them for copies of the game or as a basis for software architectures for similar games.

1.1.6 Hardware Architecture
Similar to the section of software architecture, this section describes the hardware solutions developed for a prototype.

1.1.7 Content
All prototypes have had original content developed as part of the creation of the prototype. This section gives an overview of the content created with examples from gameplay.

1.1.8 Changes from Original Design Document
Due to the iterative design process under during the development of all prototypes, the final implementation differs from the original design document. This section highlights the changes and gives motivations why the changes were made.

1.1.9 Links to external annexes
The software at these external resources is directed towards other developers within the IPerG project even though they are available to the general public. The primary
motivations for this is that the software functionality is considered stable since the final evaluation of gameplay has not been completed, and thereby significant changes may still occur. Further, the software requires custom-built hardware to be used as intended.

As per D3.2 [Waern05], the commented software is available from the IPerG Software Repository and for each prototype a contact person is given which is responsible for off-line media storage.

1.1.10 Literature & Terminology

In the cases where literature and terminology in used specifically for one prototype this is mentioned in its own section. All references are also located at the end of the report.
2 AUGMENTED DIAMOND HUNT

Augmented Diamond Hunt was chosen as the first prototype since it allowed a focus upon a few core issues. Rather than having to create a game, with the issues of playability and game balance this requires, using an already existing game allowed us to concentrate fully on the technology augmentation. Since an un-augmented version of the game already existed, this also allowed a comparison between the two versions as well as exploring overall social adaptability of a normal board game.

2.1 Research Aim

Augmenting Diamond Hunt was an exploration in how technology could be added to a board game to improve aspects of gameplay dependent on the use of tangible game tokens to handle randomness. However, a restriction was also introduced that the technical augmentation should interfere as little as possible with the social environment in which the board game was played. This restriction can be described more briefly as minimizing social weight, a concept introduced to signify the effect of artifacts having on social interaction due to their presence and use [Toney02]. Relating to this characteristic during the design process had earlier been identified as one of the guidelines for social adaptability in deliverable D9.1 [Björk04], and the research challenge of the prototype work can be described as how to improve gameplay through computational powers without increasing social weight. We also wanted to add a virtual experience to the physical game as proposed as a major theme [Montola04] where players were to gain an extra dimension in the game by letting the computer handle the distribution of tokens as well as some game Balancing Effects. Also, it was important not to break any of the already inherently covered principles like Support Interruptability [Björk04].

Although being a commercially successful game, Diamond Hunt can still be considered to have a few design flaws due its use of Randomness. Before the game begins, tokens holding items and events are randomly spread face down over the board. This means that the diamond (which is the goal of the game) can be found almost instantly, thus ruining the game. A similar flaw occurs if the diamond is found very late in the game making it impossible to beat the finder by finding a visa since they're all most likely gone (i.e. no Illusion of Influence). Also, if a token is accidentally flipped during the game, it has to be mixed in with the others disrupting the gameplay for all players. Finally, it is possible to cheat by keeping track of a certain token during the initial spreading, secretly looking at the tokens during play, or lying about whether a token was a bandit or something else.

These are all issues that can be solved and controlled through augmenting the game. By removing the value from the game tokens and letting a computer decide what the value is, cheating is removed entirely and there is no problem of accidentally flipping tokens since no information will be shown. More importantly, if the diamond is found too early the computer can make a quick change with one of the other tokens and the game will not be ruined. In the same fashion the computer can make sure that there is always at least one Visa left when the diamond is found in order to ensure the race part of the game.

As part of the Socially Adaptable Games showcase, the augmentation was done taking the guidelines presented in deliverable 9.1 [Björk04] into consideration. The use of an
existing game was done with the intention of having an already socially adaptable game so that specific parts of the guidelines could be explored individually. This would also allow the augmented version to be compared to the original to make a relative comparison between their social adaptability. Given this, the main focus for the prototype was exploring how to add computational power while minimizing the social weight of the technology used. Not modifying the basic patterns of the game’s design, the augmentation solely focused upon changing how the compute evaluation function (using the terminology found in deliverable 9.3 [Eriksson04]) for determine which event occurs at a site was done. This choice was made to try and resolve the distribution problems that could lead to finding the diamond too early or not being able to find a visa after the diamond had been found. Correcting the other design flaws, accidentally revealing tokens or cheating, were considered secondary objectives.

2.2 Overview of Game

Diamond Hunt (Afrikan Tähti) was originally designed by Kari Mannerla in 1951 [Mannerla51]. It is claimed to be the most important Finnish board game ever, having sold over three million copies with half of them in Finland [BGG05].

Image 1- Diamond Hunt game board

Augmented Diamond Hunt, like the original game, is a board game for two to five players. Players take turns traveling across Africa in search of the great Cullinan diamond. At a first glance it can seem entirely Luck based due to the heavy use of Dice and Randomness but there is room for strategy and tactics through selecting different routes and transportation means. As a family game, Augmented Diamond Hunt can be considered to have a wide audience but not have significant Varied Gameplay.
The original tokens in Augmented Diamond Hunt have been exchanged for small clay sculptures fitting the theme of the game. They are modeled after sand dunes, reefs, old temples, oases, stone formations and other places where the player can expect to find diamonds and other treasures. To cover the board, thirty of these are used.

The actual action required by a player to explore a site is to place the sculpture on a special digging site on the board. This allows the RFID antenna in the magnifying glass to detect the RFID tag in the sculpture which is used as input to the computer.

2.3 Game Instructions

As the original game, Augmented Diamond Hunt has two phases, Exploration and Race. During exploration the players traverse the continent trying to find treasure and eventually the diamond. As soon as the diamond is found, the race starts. The player that found the diamond must then go back to either of the starting positions in order to win. However, the other players can still win by finding a Visa which is a Limited Resource and returning to either of the starting positions before the player with the diamond.

The players get 5000 credits at the start of the game that can be spent on traveling or searching; this requires Resource Management in order to do well. During the game the players can also gain money by selling jewels they find.

2.3.1 Game Progression

The game is Turn Based. To see which player goes first everybody rolls a die and the player who gets the highest number starts. Each turn has two phases, Movement and searching. Movement is optional, but if chosen it is always executed before searching.
2.3.2 Movement

Before moving (unless by air) the player whose turn it is must roll a die. There are three ways of moving around the board.

**By land:** The board displays trails marked with dots that players can follow. For each eye on the die the player moves one dot. However, if the player reaches a site, she may stop.

**By sea:** Some sites are connected with blue lines and dots. These are ferry lines. To start using a ferry the player must have stopped on a site with a ferry line. By paying 1000 credits to the bank she may then move along the blue line just as on land. However, upon reaching land, the player must stop on the site, even if the die shows more dots.

**By air:** A lot of the sites are connected by airlines. If standing on a site with an airline connection, the player can pay 3000 credits to the bank to instantly move to the site at the other end of the connection. The trip cannot be extended. If the player wants to continue by air she must pay an additional 3000 credits the next turn.

If a player ends up on a blue dot with a ring around it, she must stay there until she rolls a one or a two. Also, the player who first arrives in Cape Town is rewarded with 5000 credits.
There is no limit on how many players can occupy the same dot or site. A player may search a site even if another player is standing there. However, the effects of the search only happen to the player who conducted it.

2.3.3 Searching

When standing on a site with a token on it, the player can choose to search it. If she chooses to search the site she has two options. Either she pays 1000 credits to the bank or she relies on Luck and tries to roll a four or more. If the search is successful (either by rolling or paying) the player places the token on the digging site and its content is given through audio feedback. After having evaluated the content of the token it is discarded. There are 7 different contents giving the player a Perceived Chance of Success. Also, there are two special sites in the game which affect the value of the token found there. Jewels found on the Gold Coast are worth twice as much and if a player finds a blank on the Slave Coast she must pass a turn.

Image 6 - A player searching a site with an experimental magnifying glass.

2.4 Technical Requirements and General Architecture

The technical requirements of Augmented Diamond hunt can be divided into three parts. First, the system needs to be able to identify individual tokens. Second, the system needs to be able to control the distribution of events and provide players with information about which event is linked to what location site. Finally, the players need to be able to restart the game and do other functions directly connected to the system. All these requirements should be done with as little social weight as possible, which was seen as prohibiting a traditional computer user interface.
The basic design idea is to postpone what information a marker represents by removing the illustration carrying the information and inserting an RFID-tag in the marker. By lifting the marker, similar to the action in the original game, and placing it on an RFID-antenna the system is informed that the marker has been activated and can randomize an event. The system informs the players of the outcome through sound to minimize the social weight (as argued in deliverable D9.1 [Björk04]) of the system.

2.5 Software Architecture

Handling only part of the game state of a comparably simple game, Augmented Diamond Hunt has a small software architecture focused upon handling input from the RFID reader, processing that input, and presenting the result as sound.

![Software Architecture Diagram]

**Figure 1 - Software Architecture of Augmented Diamond Hunt**

DHMain only starts and handles the initiation of the game. SoundBox is used for dynamic loading and handling of the sound effects used. The DiamondHunt class in the architecture handled input from the RFID reader. It does that by using a communication protocol and software part of the java package se.sics.commx developed in the Accord project [Accord]. GameBoard and Token holds the actual game state.

2.6 Hardware Architecture

Augmented Diamond Hunt has three main hardware components: the original board game without the site tokens, the RFID tags and a corresponding RFID reader, and a computer which can receive signals from the RFID reader and maintain the set of possible outcomes. The current implementation uses a Smart-It [SmartIts] as the RFID reader but any RFID reader which can be connected to a computer using the correct protocol can be used in theory. The protocol is defined in the software architecture.

Since only identity needs to be detected by the RFID reader a simple RFID protocol was used. The RFID reader is based upon the Smart-Its platform [SmartIts] which allows simple connections to laptops or PDAs and high-level programming languages. The RFID tags used are small glass tags using the Hitag-S standard. They were mainly chosen because of their size so that they could be hidden in small clay figurines. A schematic of the setup of the game system can be seen in figure 1.
2.7 Content

The content developed for Augmented Diamond Hunt consists of two distinct parts: design of the tokens representing locations and design of the sounds used. The actual game board was left intact except when the RFID antenna is placed under board; in this case a icon is attached to the board so signify the antenna’s location.

2.7.1 Site Location Design

Functionally, the prototype could have been constructed simply by attaching RFID tags to the original game tokens use for determining site events. However, since the unique identity of the tag was not used directly to map to an event it was possible to customize each token containing a tag. This was done by creating a number of clay sculptures based on the images of the game board. Although each of these clay sculptures represents a specific point on the board, the sculptures can be randomized without breaking the game system. Below follows concept drawings of the site locations and images of the finished clay sculptures with RFID tags embedded:

Image 7 – Concept art for location sites in Augmented Diamond Hunt.
Image 8 – Further concept art for location sites in Augmented Diamond Hunt.

Image 9 – Five clay sculptures and a RFID tag used in Augmented Diamond Hunt.
2.7.2 Sound Design

Sound effects were necessary for Augmented Diamond hunt since the effects of search were given through audio feedback. Simple sound effects and voice acting was created by the lead designer, Johan Peitz, and mixed to fit the theme and mask his identity.

2.8 Changes from Original Design Document

No substantial changes from the original design document [Peitz05] are presented in the prototype. One point worth mentioning is that the original design document had schematic views presenting PDAs while mentioning computers in general in the text. The prototype uses a laptop which is intended to be hidden but placed so that its’ loudspeakers can be heard. A second point is that the physical design of the RFID reader was not specified in the original design document. Two versions of the reader have been tested: one where the RFID reader was hidden under the board and one where the RFID reader was mounted on a looking glass. Advantages, as well as possible improvements, have been identified for both versions but the game has adequate playability in both cases.

2.9 Links to external annexes

The software is available as a compressed file at the IPerG BSCW server:
http://fit-bscw.fit.fraunhofer.de/bscw/bscw.cgi/0/36500562

2.10 Literature & Terminology


3 Wizard’s Apprentice

The choice of Wizard’s Apprentice as second prototype was due to it being judged as technically simpler than Anthills, and that infrastructure developed for the former would be reusable for the latter. With experience from Augmented Diamond Hunt of how to avoid social weight of computer technology the exploration of social adaptability was expanded to look at supporting different player types within one ongoing game and create games that would be impractical to play without augmentation. The latter can be seen as a weak form of adhesion to having core game mechanics based upon the combination of physicality and virtuality, as recommended in the game design guidelines of D5.1 [Björk04].

3.1 Research Aim

The research goals of Wizard’s Apprentice was to explore the two guidelines of deliverable D9.1 of supporting different play modes based upon social roles and analyzing player groups from several perspectives. To a lesser extent, the guidelines of supporting interruptability and designing for external events did also become part of the exploration but mainly by making the game support players to plan for interruptions in gameplay. Minimizing social weight was considered a requirement but the prototype was not designed specifically to explore that concept.

The core concept of Wizard’s Apprentice was based around the idea of supporting two heterogeneous play styles regarding level of participation. Specifically, one player, envisioned as an adult, would only takes active part in the game during short periods of time separated by longer periods of time occupied by the activities of the other players’, which were envisioned as children. The concept of heterogeneous play styles arose from analyzing two conceptual player groups where one wished to play a game while the other was more interested in making sure the first group was satisfied, in principle having the social roles from D9.1 of both negotiator and helper, but could be interested in playing focused for short period of times.

The second research goal can be described as partial game mastering. Somewhat similar to the role of Game Master in a Roleplaying game, the idea is that one player should be able to affect the players’ experience by choice of what goals they have and influence how rewards are distributed. This would let that player, either a more experienced player or a person with external reasons to prioritize the other players’ enjoyment over his or her own, nudge the game to maximize the other players’ experience. This nudging can be done for accommodate several types of demands, e.g. to comply with the players’ maturity regarding fairness, ability to cope with risk or handle possible failures. However, the other players should be able to do things beyond what the partial game master ordered with potentially both positive and negative effects.

3.2 Overview of Game

Wizard’s Apprentice is a board game in which players cooperate to defend a kingdom against the forces of darkness. All players except one play wizard apprentices who visit different parts of the game board to investigate mysterious happenings or directly confront enemies. The remaining player plays the wizard who advises the king and plans how to overcome the enemy forces.
The play style differs greatly between the apprentices and the wizard. The wizard player only actively plays when all the apprentices are gathered at the central castle and solely handles distribution of quests between the apprentices. These decisions are based upon judging the overall game state and the capabilities of the other players, both regarding their characters’ abilities and the players’ social and emotional levels. After distributing quests the wizard player can leave the game until the apprentice players have returned to the castle after trying to complete their quests. The play style for apprentice players requires less decision-making, primarily where to go, and fits the category of “roll and move” found in many games for children.

The game world consists of approximately 50 sites where events can occur and quests can be located. These sites have individual themes and the current state of the sites represent how the evil forces have been active. At the beginning of the game, the areas around the central castle are positively inclined towards the players while hostile powers are located relatively far from the castle. The hostile powers are defined as one arch enemy with three allies, each having a specific site as their base.

The overall gameplay arc starts with the apprentice players doing initial exploration of the sites near the castle, allowing them to gain experience and learn the interaction patterns in the game, e.g. about possibility to collect and trade objects. Due to the short distance required to reach these initial quest sites, the wizard needs to be present relatively often or even continuously. After the initial quests have been solved, the location of new quests become further and further away from each other and the castle, creating longer period of time when the wizard does not need to be present. As the quests become closer to the forces of darkness they also increase in difficulty and hopefully result in the minor allies are found.

Wizard’s Apprentice uses computational augmentation in two main ways that are integral to gameplay rather than being just special effects. First, it keeps part of the game state hidden and plays the part of the dark forces. Second, it supports the player who plays the wizard by presenting overviews of the game state and presenting what tasks are available to be given to the apprentices. These augmentations could only be done with difficulty without a computer, e.g. using another player who manipulates the game state either through the introduction of an extra representation of the game board or numerous game elements.

The design of combined traditional board game and software system required the development of specific semantics for the tangible user interface the game components represented. The interface and semantics is described embedded with the traditional game instructions in the following section.

3.3 Game Instructions

The set-up of Wizard’s Apprentice consists of suitable placing the board, connecting the board to a computer and starting up to program on the computer. Each player picks one of the figurines to play in the game; the characters have different attributes but only the wizard has fundamentally different gameplay.

The apprentices all have three attributes: magic, agility and charm. The values differ between the apprentices but have the same total, with one apprentice (Natasha) being equally good in all attributes and all others have one area of expertise. The attributes are
color-coded, magic as red, agility as green, charm as blue, and these colors are used both in the graphical interface and in the figurines.

![Image 12](image12.png)

**Image 12 – Figurines representing the Apprentices and the Wizard.**

Once the game is started, gameplay continues until either the kingdom is completely invaded or until the apprentices have defeated the leader of the dark forces. In the latter case, the individual apprentices’ achievements can be compared to each other to measure the players’ success.

Players interact with the game through moving their figurines, playing quest cards, and rolling a custom-built die. The die can be used as a normal six-sided die but also functions as an eight-sided die when the systems need to know its’ results.

![Image 13](image13.png)

**Image 13 – The die used in Wizard’s Apprentice.**

The game state is partially stored in a computer and partially stored by positioning the figurines on the board. Besides the approximately 50 active sites on the board two special areas exist. The first special area is the die pit, which is used when the computer system needs to know the result of the die. Due to the technical sensing solution used, the physical design of the die pit as an inverted near-tetrahedron was required to ensure that one corner of the die touched the sensor located there. The second special area is
physically similar to the location sites but provides information about quests or characters when cards and figurines are placed upon it.

![Image 14 – Close-up of the die pit and information site.](image)

Apart from a regular computer running the game engine, Wizard’s Apprentice requires access to a copy of the custom built game board. The game is intended to be played using a projector to display output from the computer but is not functionally necessary. The software uses Java runtime version 1.5 and communication API 1.2.

It is possible to debug the game engine and the game logic without access to the game board since events can generated by the Event Manager Control tool (see 3.5.7.1). Although it is functionally possible to play the game this way also, it violates the intended interaction design and gameplay, and arguably makes the game have significantly less social adaptability.

### 3.3.1 Sessions

Actual gameplay in Wizard’s Apprentice is divided into distinct sessions. Sessions consist of two main parts: the wizard’s phases and the turns taken by the apprentice players. All sessions start with all apprentices located within the castle.

#### 3.3.1.1 Wizard phase 1: Situation Assessment

Each session starts with the wizard evaluating the overall situation based upon gathered information. In the beginning of the game this consists solely of determining which of the initial quests are available, but in later phases this also includes reviewing how the apprentices solved the different quests.

The first wizard phase is initiated as soon as all players’ figurines have the castle site as their last known location and the wizard player places his or her figurine on the castle site. No other actual actions that are registered by the system is performed in this phase, it is purely to provide information to the players and allow planning.

#### 3.3.1.2 Wizard phase 2: Tasks Allocation

In this phase the wizard player gives tasks to players. Individual tasks can be given such that only one player has a task or such that several players share the same task. The wizard player is supported in the task by being given information about what attributes are needed to solve quests.
The main information provided to the wizard for dividing tasks consists of a listing of available quests, which, if any, apprentices have been given the quests, and the location of the quests on the map.

Specific information about a quest, and the ability to assign apprentices to the quest, is enabled by placing the corresponding quest card on the information site. As long as the quest card is kept on that site, players can be assigned the quest by placing them on the quest’s location on the board. Assignments can be canceled by placing figurines in the die pit. Removing the quest card returns the overview map where all player assignments can be seen.

The wizard’s second phase is ended as soon as the wizard is removed from the castle, and this begins the apprentices’ first phase. However, the system can revert to the wizard’s second phase by putting the wizard back in the castle before any apprentices have been placed on location sites.
3.3.1.3 Apprentices' Phase 1: Exploration

An apprentice's turn is based upon a simple roll and move mechanism with the addition of events when a location site is reached. Movement is simple; the player rolls a die and moves his or her apprentice the amount of steps the die indicates. Movement stops if a location site is reached. These rolls are not registered by the system and players can thereby roll the die as in other board games using dice.

In this phase players can place their figurines on the information site to gain information about attributes, experience, equipment, and current quests. The system also shows the figurines last registered location to help remember where the figurine should be placed when the information no longer is wanted.

Image 17 – Information about the apprentice Vladimir at the beginning of a game.

Reaching a location site triggers one of three possible events:

? Static: The game state and narrative is updated but without need for further player action.

? Choice: The player is presented with a set of choices of which one has to be chosen. This is indicated by placing the die with a certain result in the die bowl.

? Challenge: An event that is determined by a die roll in the die bowl with modification based upon artifacts.

Each quest is tied to one or several location sites and events successfully solved in these complete the quest. When a player has solved all the received quests the player should return to the castle since the events on location sites will all become static and not progress gameplay. An encouragement to solving quests quickly and returning to the castle is that the first apprentice to return receives an additional reward (see wizard’s phases).
The apprentices’ first phase ends when all players have return to the castle. Although not perceivable to players, this phase ends with the system updating the game state to reflect the actions of the dark forces.

3.3.1.4 Apprentice’s phase 2: Quest Result Overview

The second phase for apprentices allows players to see what quests were completed, and by whom. By placing players on the information site detailed information about attributes can be displayed, including visually marking attributes that have increased.

3.3.2 Guidance and Story arcs

Wizard’s Apprentice is intended to have two disparate arcs that describe the level of guidance from the wizard and the story development. The guidance arc starts off with strong guidance from the wizard and is indicated by short sessions with quests that can
quickly be solved. As gameplay progresses the quests become longer and apprentices perform more activities before returning to the castle and the wizard. Game mechanics to avoid over-exploration by players (i.e. visiting more location sites than in the wizard’s assignments) include hiding the basis of evil forces until located in quests, reduction in experience for overcoming challenges at non-quests sites, and is naturally limited to due increasing difficulty as players go farther and farther from the castle.

In contrast, the story arc starts with an unknown threat and quests consisting mainly of scouting or reestablishing contact with allies. As gameplay progresses the nature of the enemy become clearer but the difficulty of the quests increases.

3.4 Technical Requirements and General Architecture

Wizard’s Apprentice technical requirements can be divided into how it handles input, output, and game state.

Regarding input the game design requires that the system can detect actions related to the game elements representing the wizard and the apprentices, a die, and cards. The wizard and apprentice elements need to be detected when they are placed on, or removed from, location sites. The die does not need to be sensed when it is used for determining the movement but does need to be sensed when die rolls are made by players facing challenges. The cards need to be detected differently depending on which type of card they are: the quest cards when they are used to activate quest distribution or inform about quest status and the control cards at any time since they can restart or change the complete game state.

Output from the system to the players needs to be implemented on two levels. First, the system needs to provide feedback that players’ actions are detected. Second, the effect of those actions upon the game system needs to be presented to the players.

The game state needs to be partially hidden from the players and those parts that are public need to be accessed through perspectives of quests or apprentices. The way the game state can be changed is controlled by play modes and the game system needs to not only handle the game state changes but also which play mode is currently active to determine which changes are possible. The system also needs to calculate how the evil forces move between sessions.

The general architecture chosen is to have a game board containing the sensing technology and a traditional computer handling the game state and output. A combination of light sensing and RFID technology is used for input while audio playback and computer projection is used for output.

3.5 Software Architecture

The computer system has two main roles in Wizard’s Apprentice. The first is to generate the starting positions for the dark forces and to create the responses of the dark forces to players’ actions. The second is to keep track of the individual apprentice’s actions to provide feedback and an overview to the wizard when he or she plays. The software architecture is object-based and is comparable to most other game architectures.
Figure 3 – Schematic overview of software architecture for Wizard’s Apprentice.

Although the game board also contains software that is described in the hardware architecture section as it is closely integrated to the hardware and is independent of the other software in all aspects except complying with the communication protocol between the two, described in Appendix I: Communication Protocol for Wizard’s Apprentice.

3.5.1 Event Manager

The event manager handles all input to the game engine. This input is intended to be received from the game board but can also be generated through use of the Event Manager Control (see 3.5.7.1). To enable this functionality, the event manager consists of two functional parts: one part that receives input from a serial communication port (i.e. the game board) and translates it to abstract events and one part that sends the abstract events to game engine.

The part of the event manager that controls abstract event does not only route events but does slight error control. Due to slight delays in the converter between AC and DC power, tags can be sensed as being removed and replaced without so actually occurring. This is handled by the event manager by postponing the sending of events to the game engine for a short period, currently 2000 milliseconds, so that the event can be ignored if a countering event is caught during that period. Although this enforces a waiting period between the uses of two different figurines on one location site, it avoids erroneous game events from occurring and the time period can be easily adjusted to handle different hardware in the game board. Players are notified of when the game events are actually sent to the game engine through two sounds: one for when a figurine or card is placed on a site and one when they are removed.
3.5.2 Game Engine

The game engine ties together all other software components of the game and stores the game state. The game engine does not itself update the game state due to game event; rather the game event sent from the event manager to the game engine is forwarded to the current play mode which interprets the contextual meaning of the event and updates the game state accordingly.

The game engine also handles the necessary initiations needs when starting the game, in particular initiating the Soundbox and instructing the Site Parser to read all location sites.

3.5.3 Play Modes

The gameplay is Wizard’s Apprentice is structurally divided into several different modes of play (as defined in deliverable D5.3 [Björk05b]) which are mirrored in the software architecture as play modes. Roughly matching the phases described in section 3.3.1 above, the play modes control what information is presented by the UI and handles game events; the latter by checking if registered game events are syntactically correct and if so determining their semantic meaning, updating the game state as necessary.

Only one instance of each play mode is created per game instance with information changed within them between uses. The play modes are based upon the standard UI component JPanel but do not expand the output functionality of that component type, as the play modes contain other UI components providing actual information.

3.5.4 Soundbox

The Soundbox component plays all sounds in the game and, being functionally a singleton class, can be accessed by all other parts of the architecture. The component handles loading of sounds and stores them internally, requiring only predetermined clip names to be given by other components to facilitate playback. The Soundbox has no mixing functionality and can thereby only play one sound file at a time.

3.5.5 Site Parser

The game logic and thematic information for each location site is stored in separate XML files. The Site Parser component provides functionality to parse individual files and register them to the game engine, and has utility methods for parsing all files within a directory. The XML parser is built upon the org.xml.sax framework which is part of standard Java software development kits. An example of a site specification can be found in
Appendix II: Example Site Definition for Wizard’s Apprentice

3.5.6 UI

This part of the architecture consists of a set of UI components to present information to the players. Specific classes have been created to support thematic text, gameplay information, images, and the map which can highlight location sites and last known player positions. All the UI components are used through placement within the Play Mode components.

Worth noting is that the UI part of the architecture does not deal with any player input. This since all player input is handled by the Event Manager, which only receives input from the physical game board (and from the Event Manager Control, described below, when debugging).

3.5.7 Utils for authoring and debugging

The software utility tools were developed to support the process of gameplay design and software implementation, e.g. debugging, required for Wizard’s Apprentice. These are briefly described below and are included in the software distribution.

3.5.7.1 Event Manager Control

This tool was created to simulate input from the game board so that implementation of the game system software could be conducted in parallel with the hardware construction of the board (and the low-level software handling the hardware).

![Image 20 - The Event Manager Control tool.](image)

3.5.7.2 Debug Board

This utility application was created to serve several different purposes. A primary purpose was to check that all location sites were stored in correctly defined XML files as well as that they had individual identity codes and the correct neighbours specified. Controlling that each location also had approximately the right coordinates was another primary purpose (the actual fine-tuning of coordinates was done using the actual UI map component when the map design was finalized).

These purposes supported the initial development of the game world but the debug board was also used for initial game balancing. By showing commodities the debug board allowed balanced placing of such one the board. The debug board was also used to fine-tune the algorithm controlling the spread of evil forces, this by showing the current friendliness status of each location site so that the spread per turn could be easily compared.
3.5.7.3 Site Editor
The Site Editor provided a tool for having an overview of all location sites and changing their information without having to edit the XML files directly. The Site Editor makes up of the same Site Parser as in the game and provides functionality to convert site information from the game’s internal state to the used XML format.

3.6 Hardware Architecture
The requirements of the hardware architecture can be describe by what types of events the game system needs to sense from the game board. First, the system must sense when an apprentice arrives at the location sites on the game board as well as his or her identity. Second, the system needs to detect when the apprentices or wizard wishes to have more information about quests or the state of the apprentices. Third, the wizard must be able indicate a connection between a certain apprentice and a quest, either that the quest is given to the apprentice or that the reward from the quest should be given to the apprentice. Fourth and finally, the system needs to be able to detect the results of die rolls from the apprentices.
Based upon previous experiences with Augmented Diamond Hunt and MyTheme (see [Lundgren05] for information about the latter) it was early on decided that RFID technology would be used. However, the number of locations that needed to be sensed would require excessive costs for RFID readers. To counter this, an architecture where one RFID reader would handle several RFID antennas was developed. Based upon the frequency with which the antennas would be use it was decided to have to RFID readers: one reader handling the die pit antenna and the information site while the other reader handled all location sites. For the first of these readers it was possible to alternate between the two antennas without any significant delay of sensing physical game components occurring. However, since changing between RFID antennas does require some time due to the need of the magnetic field to stabilize, simply cycling through all location sites would take to long time for the other reader. The solution developed was to position a photo resistor at each location site RFID antenna that could detect if a physical game token was placed there. Although this made the game board susceptible to other types of covering, and that sufficient light was presented when playing, this was judged as not significantly affecting the intended gameplay.

![Schematic view of technology use in Wizard's Apprentice.](image)

Both RFID readers were based upon the BX24 microcontroller which is programmed in a version of Basic. For the die pit and information site reader, the microcontroller controls which antenna is active simply by controlling the power to the antennas. This controls what signal is received by the actual component detecting changes of RFID tag presence on the antennas. The detected changes are sent to the BX24 chip that packages the information and sends it on the serial communication.
Figure 5 – Architecture for card and die reader.

The die pit and information site locations only have an RFID antenna. For each other site, a position module is created by co-locating an antenna with a light dependent resistor to provide the required functionality.

Figure 6 – Schematic for position module.

The RFID reader handling all location sites requires a more complex structure than the other RFID reader. Instead of simply cycling through the antennas, the BX24 microcontroller runs a small event-driven system which activates an antenna of a position module when its’ light dependent resistors has signaled a difference in light. Similar to the other reader, detected differences in RFID tag presences from the RFID decoder is packaged and sent on the serial communication channel.
Figure 7 – Schematics of interconnections between modules (not all modules are shown).

During the development of the system the microcontroller behaved erratically until a bug in the third-party compiler for the microcontroller was pinpointed (the bug related to erroneous responses to relational expressions involving strings). After the code was changed to bypass that bug, the hardware became stable and robust.

Besides the components described above, the game board contains a small hub that can convert serial input from different ports into one USB connection. This allowed simplification of connecting the computer running the main software to the game board.

Image 23 – Inner details of the game board for Wizard’s Apprentice.
Image 24 – Close-up on some position modules, all control modules, die pit, and external USB connector.

Because of its volume, the game board was designed to be foldable so that it could more easily be carried. This required a slightly more complex wiring solution, a protective textile for the joint, and handles to be added to the game board.

Image 25 – The Wizard’s Apprentice board half-folded.

3.7 Content

Besides the actual gameplay design, the content created for Wizard’s apprentice can be divided into the clay figurines representing the apprentices and the wizard, the graphical design of the board, the graphical design of the quest icons, the sound design, and the texts related to locations sites and quests.
3.7.1 Graphical design of game board

The drawings made for each location site was used both on the physical game board and the projected map to minimize cognitive load to map between the two. All drawing was custom-drawn by the lead designer Johan Peitz.

Image 26 – The drawing for the pirates’ hideout.

Image 27 – The drawing for the location site Port of Call.

The drawings for each location site was used both for individual depictions of locations and for the general map. The general map also contains a traditional board game movement graph, with the location sites marked by larger nodes. The map is used both for the physical board and for the overviews in the computer-based presentation.
3.7.2 Graphical design of Icons

Since the quests were designed as generic quests which was made more specific by assigning one of the attributes of magic, agility, or charm as the one used to solve a specific quest. Since this made specific graphical representation of each quest impossible, metaphorical icons were created to represent the quests. The examples below are shown in black, but in the game the quest icons change color depending on what attribute is required to solve the quest. All icons were custom-made by the lead designer Johan Peitz.
3.7.3 Sound Design

Specific sounds were designed to strengthen the visual feedback of important game events. The sounds conveyed emotional meaning, i.e. success or failure, but did not use speech to avoid that the same information was given through language in two different media. Specific sounds were also designed to be played when the system detected the presence or removal of an RFID tag, which supported both debugging hardware and helping player pace their actions to fit the limitation of the hardware.

The sounds are in standard wave format and are part of the software distribution.

3.8 Changes from Original Design Document

In accordance with the design process guidelines of D5.1 [Montola04] and general recommendations on game design [Fullerton04], Wizard’s Apprentice went though several design iterations. These started with low fidelity prototypes using no computer software to intermediate version with some computer support to the version describe here. Actual play testing with the members of the intended user group has not been part
of the cycle; this because the primary requirements of the intended gameplay, i.e. supporting heterogeneous player styles, could not be tested until a prototype with full functionality was available. Input from interaction design and usability experts have however been included through workshops. Actual play testing with external people will occur as part of the evaluation that will be reported in deliverable D9.6. To take the game from its state as a prototype into a game comparable to commercial design would require several additional iterative cycles, which could be significantly shorter since no hardware development and minor software changes would probably be required.

This iterative design meant that the second prototype went through several major redesigns during its implementation. Already at a very early stage the theme of the game was changed from concerning knights, the original name of the game was Knight Errant, to concerning apprentices to a wizard. This change of theme did not require any changes in the planned game design but avoided placing the game world in a male-dominated setting. If also allowed a shift of focus from overcoming challenges through violence to other meanings, which thematically was done by using attributes not intrinsically related to violence.

As a game based much upon exploration, one of the earliest details of the design considered was the map to be used. The initial ideas of having the castle centrally located and having alternative routes to all sites remained throughout the process, but the exact graph used was changed several time to balance the areas and make fullest use of all areas.

![Image 32 – Early map design for Wizard’s Apprentice.](image)

Although little computational powers were used in early play testing the game logic and intended algorithms could be tested by making use of physical tokens. These, and the tokens used to represent the players’ characters were borrowed from existing games, specifically from King Arthur, Carolus Magnus, and counters for playing the card game Bridge.
Image 33 – Early prototyping with physically manifested game state.

This level of prototyping allowed basic gameplay patterns to be explored but did not consider how the physical game pieces would be connected to the computer system, and what the interaction model for this part of the system would be. To do this more low fidelity parts were added, consisting of cups signifying areas where game tokens could be detected and flow diagrams which showed what type of information the computer system would presented at any given instance. This level of prototyping created the foundation both for the tangible user interface model and what modes of play would be present in the software architecture.
One of the ideas for the tangible user interface that had been present from the beginning of the design process was that of an augmented die. Since RFID technology was planned to be used for other aspects of the design it was considered a trivial part to create a die making use of RFID tags that could be sensed by the game board. However, the RFID protocol used could only detect one tag at a time and this required the tags to be a certain distance from each other. Even when placing the tags orthogonal to each other no design of the die could be found that fitted with the normal size range of dice and could reliably read the correct result of rolling the die. The most challenging problem for this design was that a tag from the side of the die rather than the bottom of the die would be read if the die was slightly outside the antenna’s radius.

A novel die design was required as a simple augmentation of the normal die did not work. After some experimenting, it was discovered that placing the RFID tags in the corners of an ordinary die pointing towards its’ centre did work for reading the tags. This in itself was not a useful result since it would require the die would stand on its’ corner after being rolled on the board. Some experiment with original prototype die had included having a bowl so that the location of the die after rolling could be guaranteed. By redesigning the bowl to an inverted and truncated near-tetrahedron the same effect could be achieved for guaranteeing that a corner would be facing directly downward after rolling. This created the design of the die pit (see Image 14) but since the die pit had a RFID antenna at its bottom instead of a point, the die needs to have truncated corners. This lead to the final design of the die used, which is used as a normal die for movement and an 8-sided die when trying to overcome challenges.
Low fidelity prototyping was also used to explore fundamental aspects of how the opposition in the game would be systematized. This allowed for a separation between designing the algorithm and implementing it. This focus in the prototyping also allowed one aspect of the intended gameplay arc, how evil would spread through the game world, to be studied.

Image 37 – Low Fidelity prototyping of the algorithm controlling the spread of evil.

The placement of commodities was redone several times during the design process, both due to redesigns of the game map and due to detected imbalances in the distribution. Although the debug board (see 3.5.7.2) was available in several of these occasions, and could be the source for detecting the need of redesigning the distribution, it was easier to use physical tokens to test arrangements.

Image 38 – Low Fidelity prototyping of commodity distribution.

The computer presentations underwent several redesigns similar to the results from the other design disciplines involved in creating Wizard’s Apprentice. Being only information sources and not supporting input functionality, the main changes to the presentations consisted of making the various graphical layouts consistent, strengthening intended graphical elements, improving the quality of the information
visualization, and clarifying differences between thematic and gameplay related information.

Image 39 – Early version of a play mode presentation

Overall, the design process has resulted in removal of concepts and ideas to make the current design more consistent and clear and focused on the central gameplay design goals. As with any game design of similar complexity, when the design is significantly finished is impossible to state without extensive play testing, but the planned evaluation will possibly provide some measure of how near that level of completeness the prototype is.

One example of simplification that has occurred is how artifacts are handled in the game. Originally it was intended that several types of artifacts would be essential for solving quests, i.e. requiring a book of the dead to exorcise an undead force but also requiring a ring of tranquility to do so undisturbed. During the low fidelity prototyping it became apparent that having sufficient content in the game to support this would require too great an investment in authoring and would require a more complex software architecture where each component would be used less. The current design still has artifacts but only as bonuses to attributes that can be acquired by solving certain quests and can be lost by failing to overcome challenges. The simplification also solved the related interface problem of how the wizard should be able to select which apprentice was rewarded with what artifact during the wizard’s first phase.

Another example of simplification concerns the state of each location site. Originally these were intended to have one attribute signifying the relation between the location’s inhabitants and the players and another attribute signifying the evil forces presence in the location. By combining these two attributes into one algorithm to generate possible quests became clearer as became the algorithm for how evil spread. That this simplified that game world into a dualistic one of good versus evil was not considered a problem since this fitted the theme and presented an easier diegesis for the intended players.

A minor change done to the game flow was to move when the evil forces moved from the beginning of the wizard’s first phase to the end of the apprentices’ second phase. This was done not to change gameplay, nor is the difference perceivable while playing, but allowed the system to fall back from the apprentices’ first phase to the wizard’s second phase if no location sites had been visited by the apprentices.
3.8.1 Example of gameplay

The following section shows some images and screenshots from the final prototype to give a flavor of the gameplay possible in the game.

**Image 40 – Example gameplay during quest allocation.**

**Image 41 – Example gameplay just before trying to overcome a challenge.**
3.9 Links to external annexes
The software is available as a compressed file at the IPerG BSCW server:
http://fit-bscw.fit.fraunhofer.de/bscw/bscw.cgi/36500582

3.10 References


4 Anthills

The description of the Anthills prototype differs from the other prototypes in that Anthills have not been completed as of the completion of this deliverable. However, the technical requirements and software and hardware architecture is set. As gameplay evaluation will not be possible with phase one of the work package, the focus has shifted to creating a generic augmented game tile that can provide input to three other work packages: Infrastructure, Massively Multiplayer Reaching Out, and City as Theatre. The final software code will be made available at the location indicated in the links to external annexes.

4.1 Research Aim

The main vision for the game is to allow for both early leaving and late arriving players to play together in a game that runs for a longer period of time. This is one way to support interruptability [Björk04]. We also want to provide a virtual layer [Montola04] to the game by letting the computer handle some otherwise difficult aspects of the game, like unit attributes etc. and displaying these back to the players in a non discrete way [Björk04]. The game should also allow for non-player interaction.

The game uses an interactive board to display information to the players about each placed unit. Each tile on the board has the ability to light up in various intensities and colors, and can also blink. The board also has the ability to sense where tokens are placed on individual tiles. A computer connected to the board handles the game logic and finally public displays are connected that not only display an overview of the game but also can be used by non-players to partially interact with the game.

As part of the Socially Adaptable Games showcase, the game adheres to the guidelines presented in deliverable 9.1 [Björk04] to explore and validate the guidelines. Below the guidelines are numerated and how the game follows them is discussed.

4.2 Overview of Game

It’s dark times for the ants. Food is running short, Mother Nature is grim and the two rivaling ant tribes in the valley are stuck in a never-ending war. In a last desperate attempt to conquer the valley, the ants decide to try to take over their sworn enemies’ anthill by carrying their queen into their nest and have her birth all her baby ants there. If that doesn’t make them understand, nothing will! As if by coincidence, the other tribe comes up with the same idea and the race is on.

4.3 Game Instructions

Two teams battle against each other for domination of Ant Valley. In order to win the game a Team must move their queen into the hill of the other ants.

4.3.1 Overall structure

Every player plays a personal ant Unit (it is however possible to play more than one if needed). Each unit has Budgeted Action Points (AP). These can be spent anytime during the game on movement and on special abilities and are regulated by the game
computer. APs also regenerate slowly (typically 1 AP per minute or 30 seconds) so that new actions can be made. Movement costs 1 AP and executing a unit’s special ability costs anything from 2 to 5 APs depending on the ability. The amount of APs available to a unit will be displayed by the tile it is standing on. However, the amount will not be explicitly told, instead the tile will light up stronger the more points the unit has.

4.3.2 Game Progression

The game is played in real time. As soon a player’s unit has AP, it can be spent. The game continues until one side has won, and a new game can be begun with minimum need of configuration (removing the ants from the board).

4.3.3 Player Activities

Before the player can join the game she has to choose an ant role and then spawn into the game. Choosing a role is done by simply taking an ant unit with the corresponding team color and placing it on the mark with the wanted role. The computer will now know what role this unit is.

4.3.3.1 Spawning

Once a role has been set for an ant, the player can spawn. Spawning is done by simply setting down the ant unit on a tile next to the team hill. Once the unit has accumulated enough APs it can move.

![Figure 8 - Spawn points in Anthills.](image)

4.3.3.2 Moving

Moving costs one AP per tile. In order to move, the player needs only to pick up the ant and set it down on a free neighboring tile. The ant’s AP will be automatically lowered.

4.3.3.3 Special ability

Players can do different things depending on what role their unit has (Orthogonal Unit Differentiation). There are five different ant types for the player to choose from.
<table>
<thead>
<tr>
<th>Ant type</th>
<th>Special ability</th>
<th>Cost</th>
<th>Power</th>
<th>Max AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drone</td>
<td>Can move the queen</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Worker</td>
<td>Can build obstacles and new spawn points</td>
<td>4 and 5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Farmer</td>
<td>Can distribute AP</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Battle ant</td>
<td>Can attack other ants</td>
<td>Power of target ant</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Sacrificer</td>
<td>Same as battle ant but the outcome of any battle is that both ants always die</td>
<td>Power of target ant (unless attacked)</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 1 – Unit types in Anthills

How to activate the special ability is described in the game objects section.

4.3.4 Non-player activities

The public screen that shows the overview of the game can also be used by non-players in order to create external effects on the game. A meter is slowly building up on the screen and when it is full an action may be activated by pressing a button on the touch screen. Other possibilities for non-player interaction could be having a table where putting things on it represents different actions in the game world, or drawing gestures instead of pushing buttons.

4.3.4.1 Sugar rain
Ten random tiles on the board gain a sugar cube.

4.3.4.2 Queen’s Grant
Ants standing on any one of ten random tiles on the board get full AP.

4.3.4.3 Pesticide
Ants standing on any one of ten random tiles get their AP reset to zero.

4.3.4.4 Grasshopper
The ants within the landing area can be killed.

4.3.4.5 Foot of Evil
A huge footprint appears on the board and every ant inside is killed.

4.3.4.6 Wasp Attack
Players will be able to hear the swarm approaching via audio before it arrives. A few seconds before they strike the randomly selected tiles that will be struck will blink and players (if they have the AP) have a tiny chance to move out of the way. Ants standing on the attacked tiles are killed.
4.4 Game Objects

4.4.1 Units

The game features five different ant units that each team has an infinite supply of.

4.4.1.1 Drone

The drones’ only purpose in life is to serve their queen and obey her every wish. Drones are the only ants that can push the queen. In order to push the queen, just move into the tile where the queen is located with 3 AP left and the queen is pushed into the next tile. Any ants standing on the tile where the queen ends up are instantly killed. However, the queen cannot be pushed into the opposite team’s queen. Pushing costs 3 AP. Drones can also destroy defenses and spawn points. Destroying a structure requires the ant to stand on the tile next to it and spend its max AP.

![Figure 9 - Pushing the queen.](image)

4.4.1.2 Worker

The worker ants are clever little creatures that can help build defenses and new spawn points. Defenses are tiles that cannot be occupied by any ants. To build a defense, just spend 3 AP and move your worker ant out of the tile in question. Defenses can be upgraded to spawn points by moving into the same tile, spending 4 AP and moving back. Workers can also destroy defenses and spawn points. Destroying a structure requires the ant to stand on the tile next to it and spend its max AP.

4.4.1.3 Farmer Ant

Every ant can use some extra help sometime and this is where the farmer ant comes into play. The farmer ant can drop AP onto tiles that can then be picked up by ants moving through the tile. Dropping an AP on the current tile costs 3 AP.

4.4.1.4 Battle Ant

The battle ant is the main unit in the ant army. It can attack any other ant by simply walking into their tile. If the battle ant has more or as many APs left as the target ant has power, the target ant dies. Otherwise the battle ant must step back to the tile it attacked from.

4.4.1.5 Sacrificer

Any anthill with self-respect has fanatic followers. The sacrificer is a special kind of battle ant. When it attacks it works exactly as a battle ant but if the attack succeeds, the
sacrificer also dies. The sacrificer’s special power comes into play if it is attacked by another ant; in this case, both units die.

4.4.2 Game board

The game board is made up of 91 hexagonal tiles set up in a larger hexagon with six tiles on each side. Two opposite corners mark the home and hills of each ant tribe. This is where new ants spawn and where the queen starts.

![Figure 10 - The Anthills game board fitted with physical obstacles.](image)

4.4.3 Physical obstacles

The board can be fitted with physical objects that snap onto the grid. These are placed during the game’s setup phase and are customized by the players.

4.4.4 Public screen

On the public screen non-players get an overview of the game and also have the possibility to interact with the game every now and then. The actions available differ from locust attack to sugar rain to pesticide. Once applied, they are somewhat delayed on the game board to allow players to react.

4.5 Technical Requirements and General Architecture

The main requirements of Anthills are that the system has to detect and identify the tokens placed on discrete positions, tiles, and that while they are on the tile there orientation can be measured. The game tokens also need to be able to send information to the system that their special action is activated. In addition to this, the game also needs a public display with input capabilities to allow non-players to interact with the game state. The system needs to be modular so that the number of augmented tiles can be varies for uses in other augmented board games.

The general architecture is to embed simple microcontrollers within the game tokens and let these sends a unique code through an IR-LED to the game tiles. A button on each token will change the code to show when an action is being activated. Each game tile contains a module that handles communication with a master controller, shows status of the tile, and detects if a game token is placed on the position. The maintenance of the game state is considered to be done on an external device, typically a computer, which uses the master controller to set the states on the tiles. In addition to keeping
track and altering the game state, the external device is responsible for the device that presents and receives information from non-players, for instance a large touch screen.

Figure 11 - Overview of Anthills augmented board game infrastructure.

4.6 Software Architecture

No software architecture has been developed to support the gameplay design of Anthills. However, the architecture of Wizard’s Apprentice provides the basic input and output control needed for Anthills, and could after some redesign be used as a basis for the game system. The software developed for controlling the hardware is described in the software distribution (see section 4.10).

4.7 Hardware Architecture

To detect the identity and location of the tokens every tile has a module that actively tries to read the IR transmitted code from the token. This can be done using an IR phototransistor coupled to a microcontroller. Since there are numerous tiles and therefore modules, the modules communicate with a main controller on a bus. The main controller asks the modules in turn about their state and sends updates to the game state. The module will then reply with the identity of the token present. To show information related to the tile or the token placed on it the module will have a RGB LED. Instead of showing absolute information about the action points the LED showing this instead light up with different levels of intensity. This are managed with pulse width modulation (PWM).
4.7.1 Augmented Game Tile

Every augmented game tile is a module with the following functionality:

- Communicate with a master controller
- Detect IR codes
- Control an RGB LED (ON/OFF, intensity and color)

![Schematic for Anthills game tile.](image)

4.7.1.1 Communication

The protocol has to support communication with several devices on few wires. This implies that the protocol will be addressed and support enough addressed devices (>100). The protocol has to be easy to implement in a cheap controller. I2C and SMBus are two protocols that fulfill the requirements, 2 wires for communication to 128 devices. In fact the two protocols are very similar and usually compatible. I2C was chosen as a protocol because it is more widely spread than SMBus.

4.7.1.2 IR Code Detection

Codes are received from a phototransistor that reacts to IR. The signal are be read asynchronous and identified.

4.7.1.3 LED Control

To be able to select different colors as well as intensity of an RGB LED, three PWM outputs are be used.

4.7.2 Controller

A CPLD (Xilinx CoolRunner II XC2C32) is used as a controller because of its low cost, ability to do parallel tasks and the support for the I2C-protocol.

4.7.3 Token

The tokens have to be active to work together with the module. They repeatedly send an IR code (or different codes for different events) for the module to receive. This is accomplished with a CPLD and an IR diode. The same CPLD as for the module is used, again because of the price and the low power consumption.
4.7.4 Master Controller

A PC is used as a master controller and to send commands to the modules using the I2C protocol. A microcontroller is used to translate between the RS232 and the I2C protocols. A buffer is used to translate the levels from the microcontrollers’ TTL to the modules’ 3.3V CMOS.

4.8 Content

Since a functional game has not been created, no content has been developed for Anthills.

4.9 Changes from Original Design Document

No substantial changes have been made regarding the gameplay of Anthills, mainly due to the fact that the resource available for the prototype has been focused upon the hardware development. The original plan was to expand upon the technical platform using in Wizard’s Apprentice but during the implementation of that prototype it became apparent that Anthills would require a different hardware solution to be viable. This
meant that the use of RFID technology was exchanged to IR transmitted code, and this signified a change of approach where the game tokens could be active components in the sense that they can initiate information exchange. Although the current token design allows activation of one action and orientation to be detected, the design is generic enough to allow more complex hardware tokens without redesigning the hardware of the game tiles.

4.10 Links to external annexes

The hardware specification and software code is available as a compressed file at the IPerG BSCW server:

http://fit-bscw.fit.fraunhofer.de/bscw/bscw.cgi/36500587

4.11 References


5 CONCLUSIONS

This document has described the three prototypes which have been explored during phase one of work package 9. Out of the three, one original game and an augmented version of an already existing game have been created and delivered to be evaluated. The remaining prototype has been used as a vehicle to explore the design of a generic augmented board game tile, with potential uses in the showcases Massively Multiplayer Reaching Out and City and Theatre.

Being not only proof-of-concepts prototypes for a game genre, the prototypes have been designed to support the refinement on design and evaluation guidelines of IPerG, and the guidelines on social adaptable games in specific. However, this aspect of the prototypes are not described in this deliverable as it will be the focus of deliverable D9.6, the evaluation of initial play testing.

REFERENCES

For accessibility to IPerG deliverables, please go to the official IPerG website: http://iperg.sics.se.


**APPENDIX I: COMMUNICATION PROTOCOL FOR WIZARD’S APPRENTICE**

The protocol is a string of characters containing data. Each piece of information in the string is separated by a period and the string ends with a question mark. The first two characters are always "HW" in order to ensure that the software can start to parse the string at the right place. After that follows either "B", "I" or "D" to tell if it was the board, information area or the dice pit that generated the event. If the event was a board event, the id of the site in question follows immediately. Then a single character describes if something was placed or lifter event by "1" or "0". Finally the tag of the object in question ends the string. Examples:

HW.B.27.1.112004108050

The object with tag 112004108050 was placed on site 27 on the board.

HW.D.1.112047186100

The die corner with tag 112047186100 is placed in the die pit.

HW.I.0.112004108050

The object with tag 112004108050 was lifted from the information area.
APPENDIX II: EXAMPLE SITE DEFINITION FOR WIZARD’S APPRENTICE

<?xml version="1.0" ?>
<site>
    <!-- off-game information -->
    <admin lastmodifier="JP" version="0.1" creator="Johan Peitz" modifydate="20050525"/>
    <!-- in-game stuff -->
    <data name="Lonely Peak" type="knowledge" distance="3" id="60"/>
    <!-- base skill and trading -->
    <attributes skill="magic" weakness="" makes=""/>
    <neighbours>
        51 52
    </neighbours>
    <!-- quests, possible actions: nop, trade, choice, roll -->
    <quest type="default" action="nop">
        <description>
            The cave at the mountain top is empty.
        </description>
    </quest>
    <quest type="knowledge" action="roll">
        <description>
            The eremit knows a lot and can give you some good advice, but he wants you to show some magic tricks first.
        </description>
        <success>
            Some fancy lighting effects later the eremit want to offer you something in return.
        </success>
        <failure>
            You turn your own foot into cheese. The eremit laughs at you.
        </failure>
        <alternative value="2">
            A charm course.
        </alternative>
        <alternative value="1">
            Get information about %target.
        </alternative>
    </quest>
</site>