NTNU

Norwegian University of
Science and Technology

World of Wisdom - World Editor

User-interface for creating game worlds for World of Wisdom

Thor Grunde Krogsæter

Master of Science in Computer Science

Submission date:	June 2009
Supervisor:	Alf Inge Wang, IDI
Co-supervisor:	Bian Wu, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Design and develop a user-interface for teachers in WoW to add new kingdom for teachers. This interface should be a game interface that should be regarded as a part of the game. The functionality needed is to add new parts of the game worlds, objects and monsters etc.

Assignment given: 15. January 2009
Supervisor: Alf Inge Wang, IDI

Abstract

During the fall of 2008 a prototype of an educational multiplayer role-playing game called World of Wisdom (WoW) was developed as part of the specialization project TDT4570. WoW focuses on using knowledge for progressing through the game. The goal of this thesis was to design and develop a user-interface for teachers, that could be used to generate new content for WoW.

In this thesis we described the design and implementation of such a user-interface called the WoW World Editor. The World Editor supports generating new maps, creatures, objects and questions for World of Wisdom. By making it easier to create the worlds, the course staff can focus on creating the knowledge for the game.

For the students to be able to interact with the course staff while playing the game, we suggest a seperate client for the course staff. This client will then have additional functions that can be used to aid the students with problem, and to get valuable feedback from the players.

Preface

This thesis were written during the spring semester 2009 at Norwegian University of Science and Technology (NTNU) in Trondheim, for the subject TDT4900 Master Thesis, Computer and Information Science. It is a continuation on the work done the fall 2008 in the specialization project TDT4570.

I would like to thank the project supervisors, Alf Inge Wang and Bian Wu for their support and guidance. I would also like to thank the students at the studyhall "Fiol" for their feedback during the user testing.

Thor Grunde Krogsæter

Trondheim, June 11, 2009

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction

In the Introduction Part we will look at the World of Wisdom (WoW) prototype. We will go through the research questions for the WoW World Editor project and the research methods used.

In this chapter we will describe the project goal and background in the World of Wisdom project from 2008, and the motivation for this project.

1.1 Project goal

During fall of 2008, the WoW prototype was developed as the specialization project by Esben Føllesdal, Henrik Halvorsen and me. WoWs goal was to be an educational Massively multiplayer online role-playing game (MMORPG), a game that could be used by students to learn their course in an alternative way. Knowledge of a subject would be required to progress in the world. The focus in the prototype that was developed was on the battles between the player and the computer controlled enemies (Enemy Non-Player Character (NPC)s). More detailed description of the WoW project is given in the World of Wisdom chapter.

The goal of this project is to design and develop a world editor for the World of Wisdom prototype. The editor should make it easy to create a new world, and alter existing objects and their attributes.

1.2 Motivation

While we were working on the WoW prototype, we had many ideas for improvements and further development but we did not have the time to implement them. I had some ideas for a editor and administrative tools and that was one of the reason why I wanted to work on this project, so I could continue the work.

I have also over the last 10 years been using many commmercial game editors, and have experience with using many different types. Most of the time it was simply as a hobby and I was more interested in the technical features of the editor and their capabilities then actually creating levels and objects.

1.3 Report Structure

This section describes the structure of the report and the project.

1.3.1 State of the Art

The State of the Art Part II contains research on editors for different types of games. Some of the editors are among the first to go public with simple textbased design, and others are the state of the art, showing the capabilities of the latest game editors. We also take a look at Game Masters as a tool for interacting with the players while the game is running.

We will also look at the state of the art educational game concept, and a similar educational project called Age of Computers (AoC).

1.3.2 Design

The Design Part III contains the architectural design of the WoW World Editor, and how it will be integrated with the WoW prototype. It also includes details of what features that are missing in the World of Wisdom prototype, and the design for the improvements to the prototype and the database. We also look at how Game Masters can be benefitial for World of Wisdom.

1.3.3 Implementation

The Implementation Part IV contains the details of the implementation of the WoW World Editor. It also describes what has been changed in the WoW applications and in the database.

1.3.4 Results & Evaluation

Results and Evaluation Part V will show the results of the design and implementation, and look at the research questions and how they have been answered. We will also look at some user tests that were performed and how they went.

1.3.5 Conclusion

In the Conclusion Part VI we will give the overall conclusion of the project. This part will also include a chapter on Future work. The Future work Chapter 26 will contain all suggestions for the future additions to the WoW World Editor. Details of the requirements that were not implemented, and what needs to be done before implementing them.

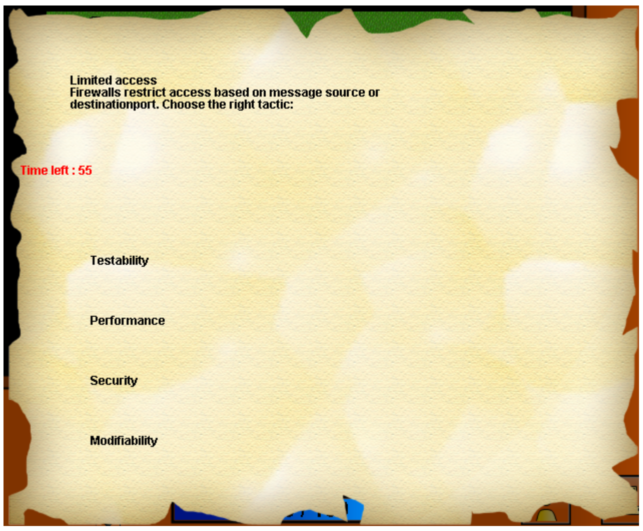
1.3.6 Appendix

The Appendix Part VII contains the models and diagrams from the previous World of Wisdom project, and the User manual, images and Use cases that is not part of the main report, but mainly for the teachers and students to familiarize themself with the application. It will also contain a more detailed guide for setting up the World of Wisdom project and the World Editor in a new environment.

Chapter 2

World of Wisdom

The World of Wisdom prototype was developed the fall 2008 by Esben Andrè Føllesdal, Henrik Halvorsen and me. The goal of the project was to design and implement a prototype of a MMORPG where the primary tool for progressing through the game is knowledge. This was done by giving the user multiple choice questions that needed to be answered correctly to defeat an enemy. Figure 2.1 shows an example of a question from the prototype. Here the player has one minute to answer a question. The question is answered by selecting one of the alternatives below the time-left counter.



Limited access
Firewalls restrict access based on message source or destinationport. Choose the right tactic:

Time left : 55

Testability

Performance

Security

Modifiability

Figure 2.1: Screenshot of Question Panel from the World of Wisdom Prototype

2.1 World of Wisdom terms

There are some terms that are used for World of Wisdom when describing certain elements, and therefore we give a list of some of the more common terms that will be used in this report and a description of each.

Static objects are stationary objects that can not be interacted with other then collision detection. Like houses, trees, rocks, etc.

Friendly Non-Player Character (NPC) are non playable character that can give quests or offer dialog to the players. Dialog or quests were not implemented in the prototype.

Enemy Non-Player Character (NPC) are non playable character that the player can fight with and use knowledge to defeat, as shown in Figure 2.1. Combat was one of the focuses for the prototype, to show how knowledge could be used.

Players are other users. In the game the users can chat with other users, assist when fighting enemy NPC and there are collision detection between player avatars. The players avatar is called Character.

Zone is what we call the areas or maps in World of Wisdom. Each zone can contain several Enemy NPC, Friendly NPC and Static Objects.

Background is an array of small images that are drawn before everything else in the zone, in the prototype there are used two types of tiles, dirt and grass.

Kingdom is what we call the world in World of Wisdom. The kingdom is divided into several zones, which contains the content in the world.

Primary attributes determins the power of a character. (Strength, Dexterity, Intelligence and Constitution)

Skill is divided into two types, offensive skills and defensive skills. Offensive skills give extra damage when attacking, and defensive skills give extra protection against attacks.

Attacks are the abilities the player can use to attack creatures. The more powerful an attack is, the more difficult question the player will get.

2.2 Prototype implementation

The prototype is divided into four application, Client, Lobby Server, World Server and Database Server. The applications also uses a common package called Shared Library, which contains models used by several applications and the shared network implementation used for sending messages between the applications. All the applications were developed in Java. Figure 2.2 shows the architectural overview of the World of Wisdom prototype. The solid lines represent the network connection created between the applications using the Shared library.

2.2.1 Client

The client is the main program for the user, in this case students, that wants to log on to a kingdom. When starting the client it will connect to the Lobby Server, and the user will then have to type in username and password. The Lobby Server will check if the information is correct via the Database Server. If the user is authorized to log on, the Lobby Server will return a list of World Server that the user may connect to.

When the user has connected to a World Server the user will be in the game world. While in the kingdom, the user may fight with Enemy NPCs and walk around and chat with other players. The user can also look at the stats of the character, attributes, attacks and inventory. In the inventory the user can drag and drop items from their bag onto their body to equip items.

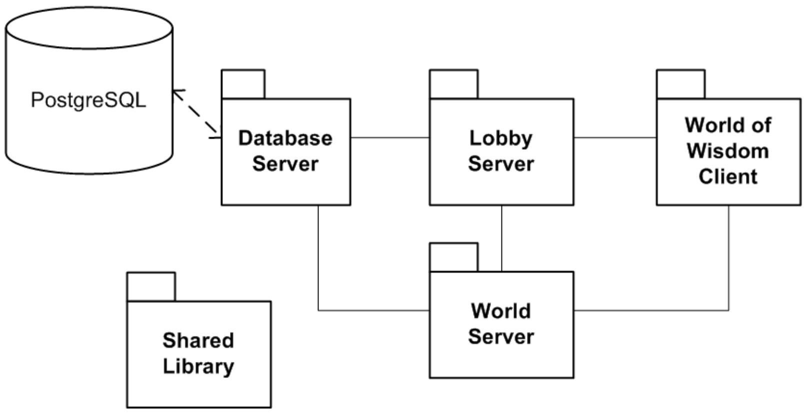


Figure 2.2: Architectural overview of the World of Wisdom Prototype

2.2.2 Lobby Server

The Lobby Server handles the verification of players, and contains a list of the World Server that are available(online). The Lobby Server is a small but central part of the World of Wisdom prototype.

2.2.3 World Server

The World Server contains the game world that the users can move around in. When the world server starts it will register that it is online with the Lobby Server, so that users may connect.

Everytime a users performs an action, like movement or attack, a message is sent to the World Server with information about the action. The action will then be handled, and in the case of movement, broadcasted to the other users so that their world are updated.

2.2.4 Database Server

The Database Server receives and handles requests from the World Server and the Lobby Server for access to the PostgreSQL database. It uses serializable objects to package the information and send it over the network using the TCP/IP protocol in Java.

2.3 My contributions

While working on the implementation of the prototype, we each had our own areas to focus on. I mostly worked on the core network implementation, the World Server application and the graphics for Client application.

Chapter 3

Research

This chapter describes the research questions that will be answered during this project and their motivation. We will look at the different research methods that can be used, and then look at which will be used for each research question.

3.1 Research Questions

RQ1 - What types of tools are needed for creating a kingdom for World of Wisdom?

- We will look into the data structure of the World of Wisdom prototype, how a kingdom is stored and what tools are needed to simplify creation of a kingdom. As currently there is no simple way to create a kingdom.

RQ2 - What improvements are needed in World of Wisdom prototype to make a complete connected world?

- As the World of Wisdom prototype did not focus on the creation of the kingdom, we will look at what improvements is needed to create a kingdom where you can move around between the zones in the kingdom. Most of the focus in the project were on the combat part of the world, and how knowledge could be used there. So there are many parts missing, as it was a prototype.

RQ3 - What similar game editor tools exists, and what can be learned from them?

- Many commercial games include tools that can be used to create new contents for the game. It can be useful to see what basic functions are included in most of the existing editors, and how the editors are used. We will look into how world creation is done in similar educational games, and also look into how the modern commercial game editor works.

RQ4 - How to make it easy to create and maintain a kingdom and what can be done to reduce the time it takes to make a world?

- As a kingdom may be a large game world with a lot of elements, and course staff may not have time to keep working on it everyday of the semester, features for reducing the time needed to make a complete kingdom will be looked into. What can be done to generalize/simplify the way the kingdom is created and altered.

RQ5 - How can a teacher/assistant improve the experience of the students while the game is running?

- For the course staff it could be benefitial to be able to observe and interact with the students while they are playing. To get feedback on what is good/bad, and how the game works in general. We will look at how other similar games interact with their clients (students) , and how this can be used to improve the experience of the user in the World of Wisdom.

3.2 Research Method

In this section we will describe the methods used to answer the research questions for the WoW World Editor project.

3.2.1 Experimentation approaches

Zelkowitz and Wallace describes four main categories for experimentation approaches in software engineering. Following is a short description of these four approaches.[ZW98]

Scientific method The researcher develops a theory to explain a situation. They propose a hypothesis and test several variations of it. During the testing they collect data and either verify or refute the claims of the hypothesis.

Engineering method The researcher observes existing solutions to a hypothesis and proposes a better soulution. They then build or develop their suggestion and evaluate their effort. This is repeated until no more improvements can be found.

Empirical method Using a statistical method to validate a proposed hypothesis. Measure and analyze data to verify the hypothesis.

Analytical method The researchers forms a formal theory or set of axioms, and the results from that theory can be compared with empirical observations.

3.2.2 Validation methods

There are three main categories for data collection, Observational, Historical and Controlled. Each of these categories have 4 sub categories, Table 3.1 shows a list over the 12 methods. The main categories will be briefly described in this section. [ZW98]

Observational The observational methods collects data as the project develops.

Historical The historical methods collects data from already completed projects. The data only needs to be analyzed.

Controlled The controlled method uses multiple instances of observation to provide statistical validity of the results.

Table 3.1: Data collection methods

Observational	Project monitoring Case study Assertion Field study
Historical	Literature search Legacy Lessons learned Static analysis
Controlled	Replicated Synthetic Dynamic analysis Simulation

3.3 Choice of research methods

Here we describe how the research methods will be used for the WoW World Editor project.

Research question 1 - *What types of tools are needed for creating a kingdom for World of Wisdom?*
We will use the Historical method to collect data on the subject of creating worlds in similar games, and by looking at data structure of the World of Wisdom prototype. By using the Engineering method we will then develop and test a solution for the problem, the WoW World Editor.

Research question 2 - *What improvements are needed in World of Wisdom prototype to make a complete connected world?*
Using the Historical method (Legacy and Literature search) we will look at the World of Wisdom prototype and implementation documentation, to find what are the most important improvements needed for the WoW World Editor. Using the Engineering method, designing and implement some of these improvements.

Research question 3 - *What similar game editor tools exists, and what can be learned from them?*
Using the Historical method to collect data on the state-of-the-art game editors and educational games and what can be learned from them.

Research question 4 - *How to make it easy to create and maintain a kingdom and what can be done to reduce the time it takes to make a world?*
By using the Engineering method we will design solutions to automate and simplify creation of kingdoms, and then develop and test the solutions.

Research question 5 - *How can a teacher/assistant improve the experience of the students while the game is running?*
By using the Historical method we will look at existing MMORPGs, and similar educational games to see what can be learned from them. We will then use that information to propose solution that could be benefitial for the WoW prototype.

Chapter 4

Development Tools

This chapter describes the tools used throughout the project and how they have been used.

4.1 TeXnicCenter

TeXnicCenter is a free tool for developing LaTeX documents in Windows. [TeX]

TeXnicCenter is used in this project to write this report.

4.2 Eclipse and Java

A project aiming to provide a universal toolset for development. Eclipse is an Open Source IDE and mostly supports Java, but the development language is independent. [Fou]

Eclipse was used as the main development tool for implementation of the World Editor.

4.3 NetBeans

NetBeans [Micb] is a fully-featured Integrated Development Environment (IDE) for developing applications in Java. It is open source and free, and can be used in Windows, Mac Os X, Linux and Solaris.

In this project it is used to create a draft of the user interface for the design part of this document. As NetBeans is a simple to use IDE that offers drag and drop tools for creating user interfaces for Java applications. But the code created is complicated and difficult to adjust, so it will be only used for a draft and ideas.

4.4 Paint .Net

Paint.NET is a free image and photo editing software that features layers, unlimited undo and special effects.[Bre]

Paint.Net was used to create some of the graphic elements in the implementation of the World Editor.

4.5 Microsoft Office Visio 2007

Visio 2007 is a program designed to make it easier to visualize and communicate complex information. With it you can create advanced diagrams, and it also has several templates for creating, UML models, database diagram and similar.[Mica]

Visio 2007 was used to create all the class diagrams, entity relation models and sequence diagrams in the report.

Part II

State of the Art

Chapter 5

Introduction

In this part we will look at different types of game editors, both old text based and new state of the art game editors. In Chapter 7 we look at the most common method of interacting with the players in Massively multiplayer online role-playing game (MMORPG), Game Masters. We will look at some of the latest educational game concepts, and at Age of Computers (AoC), a similar educational game.

Chapter 6

Game Editors

In this chapter we will look at different types of game editors and show some examples. We will also look at the Aurora Toolset which is an editor for an Role-playing game (RPG) that are similar to World of Wisdom in some aspects.

6.1 General

The earlier type of editors were often simple, using text editors, or by using simple primitives as representative of game elements. Figure 6.1 shows an example of an editor that was made for Wolfenstein 3D[Sof]. It was not an official editor, but shows how colors and letters can be used as representatives when creating a map. The letters represented object, like creatures or power ups, and colors represented floors and walls.

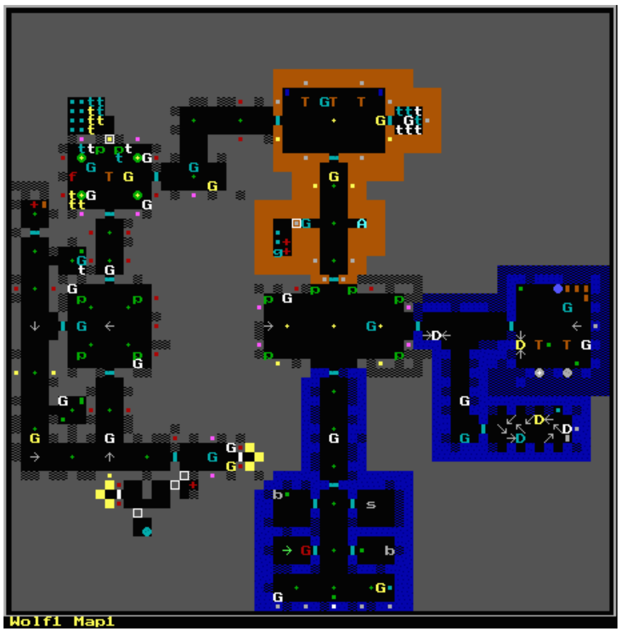


Figure 6.1: Screenshot of the MapEditor for Wolfenstein3D

The more recent editors are alot more advanced and often features ingame graphics that shows how it will look when you play the map later. Some editors even allows for playing the level in the editor, as if you have started the game. Examples are Trackmania which is a arcade racing game, and Farcry [Cry] which is a firstperson shooter. This allows quick testing of changes and also makes developing new maps alot faster. In the following sections we will look at three examples of editors. The Adventure Construction Set from 1985, which could be used to generate entire Role-playing game (RPG)s. Trackmania from 2003 to 2008, which offers a easy to use ingame level editor, and LittleBigPlanet from 2008, a state-of-the-art game with an integrated editor which is a part of the game.

6.1.1 Adventure Construction Set

Adventure Construction Set (ACS) is a program that allows the users to create Role-playing game (RPG)s. It was published in 1985 by Electronic Arts, and was their best selling game that year. It was originally created for Commodore 64, but later ported to DOS and Amiga.

The ACS automate some of the mechanical parts of the game construction, but still requires some good game design and is timeconsuming to create a game from scratch. The construction set included an option to randomly create mazes and rooms. Many would use this function to save time by automatically generate a large empty world and then create their adventure by modifying it. The games created by the construction set could easily be shared with others as standalone programs, and did not need the construction set to run. [Joa]

Figure 6.2 shows a screenshot of the level editor in ACS.

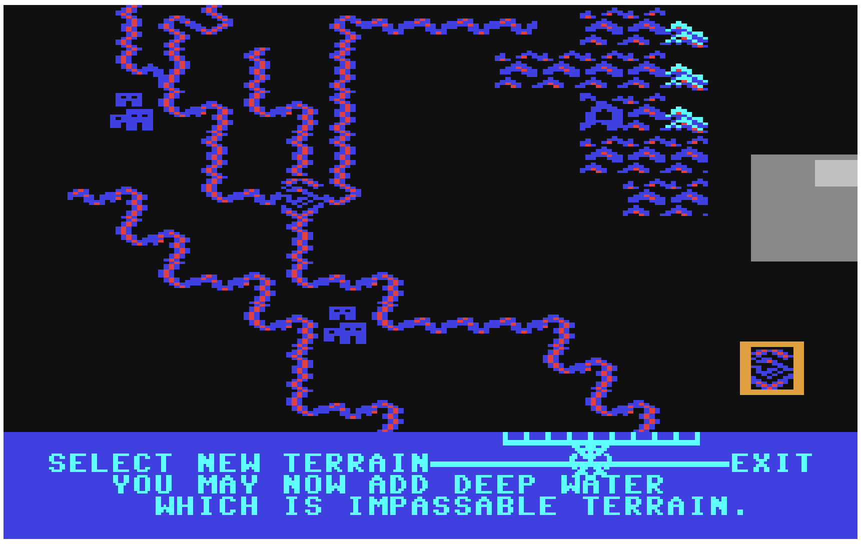

SELECT NEW TERRAIN——EXIT
YOU MAY NOW ADD DEEP WATER
WHICH IS IMPASSABLE TERRAIN.

Figure 6.2: Screenshot of the Adventure Construction Set

6.1.2 Trackmania

Trackmania is an arcade racing game developed by Nadeo. There are four games in the series for PC, Trackmania Original, Sunrise, Nations and United. Trackmania United is the latest and was released in 2006. It offers the game environment used in the three other games, like desert, snow, rally, and so on. Trackmania has several game modes that the player can use, race, platform, crazy and puzzle. The Puzzle game mode utilizes the ingame level editor, where the player must use the editor to complete the track using a few given components. This allows the players to generate several possible solutions to the puzzle with a goal to make the most effective and fastest one. This way of integrating the editor with the gameplay helps the players learn the basics of the editor, so when they have finished the game they can create their own tracks and share them with friends and people online. When joining a game with a track the player does not have, it will be automatically downloaded in seconds. Figure 6.3 shows a screenshot of the trackmania editor. [Nad]

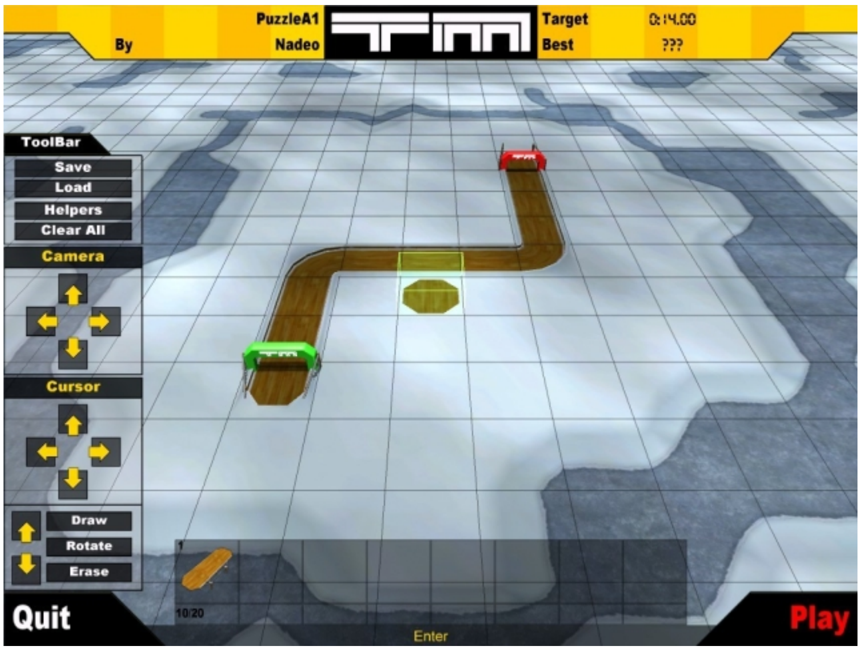


Figure 6.3: Screenshot of the editor for Trackmania United

6.1.3 LittleBigPlanet

LittleBigPlanet is a platform puzzle game developed by Media Molecule for Playstation 3. In LittleBigPlanet the player controls a little character called Sackboy/Sackgirl which can jump, push, grab and run through levels. The game features a robust physics engine that effects the different types of objects in the game like cloth, rubber and wood. When playing through the pre-built levels in the game, the player can collect prizes that unlocks materials and objects they can use in their own levels.

In LittleBigPlanet's Create mode, the player can create their own levels using materials they have unlocked in the Play mode. They can also use a number of basic shapes, like squares and circles, to create their own objects. Then they paint the object into their level while choosing a material type for the object. The game offers many ways to connect the objects together, like strings, bolts, glue or various types of triggers.

All the content that is created, objects and levels, can be shared with other players online through the LittleBigPlanet community. The game also offers multiplayer, so that the players can play a level together either over internet or on the same game console. The players can also create levels together as shown in Figure 6.4. Here two players are playing and creating a level together, one is placing decals to decorate the level while the other is interacting with one of the objects. [Mol]

Figure 6.4: Screenshot of LittleBigPlanet

6.2 Aurora Toolset

Aurora is the game engine developed by Bioware for the game Neverwinter Nights and Aurora Toolset is the world editor that comes with the game. Neverwinter Nights is an online RPG game that has some similar elements to World of Wisdom. With this editor the players may create their own worlds and alter most of the variables in the game, like spells, monsters, NPC dialog, etc. It also has a powerful scripting tool, that can be used to create custom quests and scripted events.[Bioc][Biod]

As described in the Introduction to the Toolset on the Neverwinter Nights (NWN) homepage, the toolset is divided into four main parts.[Biob]

Following is a description of the main parts in the toolset that are shown in Figure 6.5.

A is the Toolbar with commands and preferences.

B contains a list of the objects that have been placed in the world. Its organized into three categories, Areas which contains all the maps in the world, Conversations which contains all the dialog for the NPC, and Scripts for the scripted events and quests.

C is the main display that shows the world, and where most of the work is done.

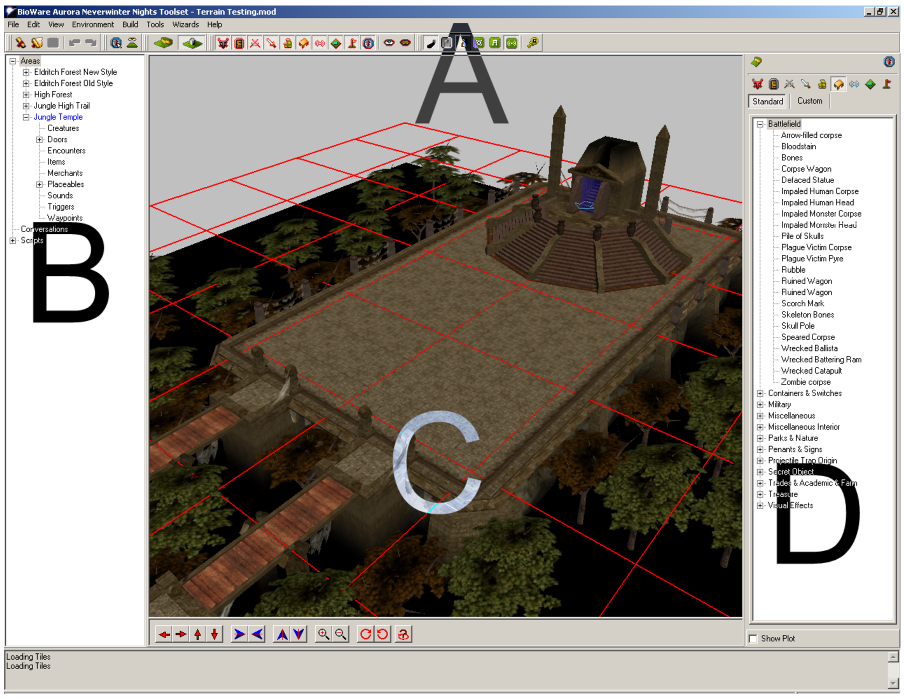
D contains the content that may be placed in the world. The type of list can be changed using the buttons over the list(Merchant, Enemy, Placeable objects, etc.).

Figure 6.5 shows a screenshot of the Aurora Toolset.

Neverwinter Nights has similar world build up as World of Wisdom, in that it is divided into zones, and each zone is divided into an array of squares. Though the game and editor is in 3D and more complex, the basics are similar.

6.2.1 Toolset Features

This section contains the features that the Aurora Toolset supports for NWN.



Loading Tiles
Loading Tiles

Figure 6.5: Aurora screenshot, courtesy of http://nwn.bioware.com/

Map - Design and create a map for a game.

World - To be able to connect several maps and create a gameworld.

NPC - Create or alter the NPC. And add or alter dialog with friendly NPCs.(Select graphics, abilites, attributes, etc.)

Items - Create or alter the items in the game. (Weapons, armors, useable items, etc.)

Quests - Create new quests with dialog. The quests can be created and altered by using a script editor that is built into the Aurora Toolset.

Chapter 7

Game master

Game Master (GM) is the online role playing game version of a Dungeon Master, which is from the Pen-and-Paper Roleplaying game, Dungeon & Dragons. Dungeon Master are responsible for describing the world around the players, and telling them what effects their actions have. They use dices to predict what happens when a player do an action or fight monsters.[Nov08]

In online gaming, the GMs are an intermediary between the developers and players. When players discover bugs or exploit, they can report these to the GMs, and they will either fix the bug there or send a report to the developers. GMs can also enrich the experience of the game for the users, by creating interactive quests, where the players will be talking to creatures which are controlled by the GM. They may also just simply create a challenge by spawning some additional monsters. [Ale05]

GMs can alter certain aspects of the game and control transaction while the game is running. The game masters usually logs on to the game with a special client that has additonal tools for adding more content or altering the content in place. An example of such a client is the Dungeon Master client for NWN which will be described in the next section.

7.1 Neverwinter Nights DM Client

NWN bases its game mechanics on the Dungeons & Dragons rule set, they have also called their game master client "Dungeon Master Client". With the Dungeon Master (DM) Client, the user can log on to existing multiplayer games, or create their own, and enrich the experience of the players of the game. The DM has all the same functions as the normal players, with some additional functions on a extra toolbar only visible for DMs. Figure 7.1 shows the additional control bar available to the DMs. The DM is also intangible, so they can pass through creatures and players, and they move around six times faster then players. [Bioa]

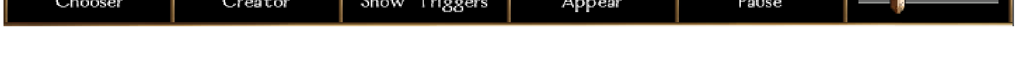


Figure 7.1: Dungeon Master controlbar

7.1.1 Control bar

The Control bar has five buttons and a slider. The two first buttons, Chooser and Creator, are the main functions of the DM client. Following is a short description of each of the buttons and the slider.

Chooser With the Chooser, the DM Client gets an overview of all the areas and their creatures, triggers and waypoints in the game world. They can use the chooser to jump to any of the objects, or to perform other actions (Kill, heal, take control, examine, etc.). The Chooser is shown in Figure 7.2 at the right side.

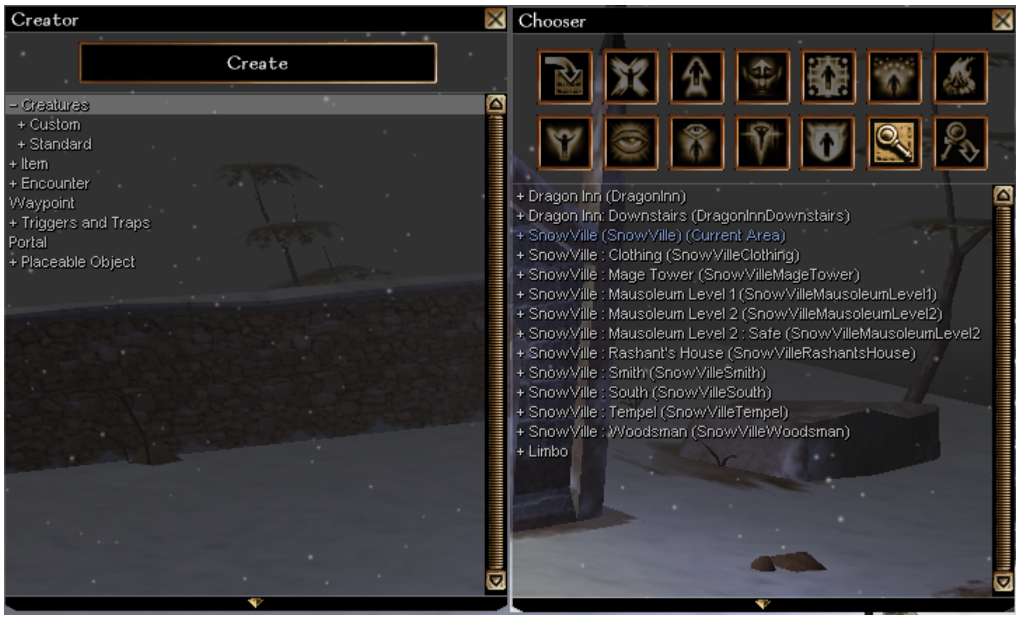
Creator With the Creator, the DM Client can create creatures, items, furniture and more. The Creator shows a tree structure, and the user can select an item and then create it. The user then select the location for the new item. The Creator is shown in Figure 7.2 at the left side.

Show Triggers Toggles to show trigger or hide them. Triggers can be traps, or scripted events. These are hidden for the players, unless they are able to detect them in the game with a detect skill.

Appear Toggles to make the DM visible or invisible. When the DM is invisible, none of the players or creatures can detect it, even with see invisible spells ingame.

Pause Pauses the game world. DM and their functions are not paused.

Difficulty Slider The difficulty level for combat in the game world. Left is easier, and right is more difficult.



- Creatures
  + Custom
  + Standard
+ Item
+ Encounter
Waypoint
+ Triggers and Traps
Portal
+ Placeable Object

+ Dragon Inn (DragonInn)
+ Dragon Inn: Downstairs (DragonInnDownstairs)
+ SnowVille (SnowVille) (Current Area)
+ SnowVille : Clothing (SnowVilleClothing)
+ SnowVille : Mage Tower (SnowVilleMageTower)
+ SnowVille : Mausoleum Level 1 (SnowVilleMausoleumLevel1)
+ SnowVille : Mausoleum Level 2 (SnowVilleMausoleumLevel2)
+ SnowVille : Mausoleum Level 2 : Safe (SnowVilleMausoleumLevel2
+ SnowVille : Rashant's House (SnowVilleRashantsHouse)
+ SnowVille : Smith (SnowVilleSmith)
+ SnowVille : South (SnowVilleSouth)
+ SnowVille : Tempel (SnowVilleTempel)
+ SnowVille : Woodsman (SnowVilleWoodsman)
+ Limbo

Figure 7.2: The Creator and the Chooser

Chapter 8

Educational Games

In this chapter we will look into the Age of Computers educational game, and a state of the art educational game concept for simplifying creation of new educational games and adapting them to the learners needs.

8.1 Age of Computers

Age of Computers is an educational game designed for the course Computer Fundamentals at Norwegian University of Science and Technology (NTNU). The game aims to replace the traditional "paper-exercises" with a game where the user will solve tasks relevant to the course material. The students will have to play through the game to pass the courses exercise program. The types of tasks that the students are given can be categorized into four groups that will be described in Section 8.1.1. [NL04]

The player walks through the different ages of the computer history and earn points by solving tasks. The player starts in the "Age of Mechanical Computers" and when the player has acquired enough points, they can proceed to the next age, "The Age of Vacuum Tubes". This continues until the player reaches the future in "The Pico Age".

8.1.1 Problem types

Multiple Choice

Most of the tasks in Age of Computer are from this category. Some questions involves doing elaborate calculations to find the right alternative, while others requires the user to select the wrong answer out of 4-5 statements. It also encourage the students to read and think about the answers, rather then "trial and error approach". This is done by making the students wait for a certain amount of time after answering wrong on a task before they can try again.

Number problems

These are the types of tasks where the answer is a number. For example, to calculate the transfer time of a file, when you get to know the access time, and speed of the hard disk. It was found to be a significant improvement over the Multiple choice, since it can be used to generate alot of different problems. The formulas needed for solving the task can be used to generate random variables, so that no student get the same numbers.

ALU(Processor) control signals

Arithmetic Logic Unit (ALU) control signals problems was implemented by using the number problem mechanism. The tasks ask the students to give the exact control words for the micro operation.

As with the Number problems, large numbers ALU tasks can be generated relatively simply.

DARK assembler

As the Computer Fundamentals course also included some assembly coding, the DARK assembler was used. DARK was developed by Ola Ågren at Umeå University in Sweden, and offers 4 types of processor architectures (stack machine, load-store machine, memory-memory machine and index machine). DARK uses a simple text edit window for writing the assembler program.

The students are given a problem description, and writes the program i DARK. Sometimes they are given an incomplete assembly program and are asked to fix/complete the program. When the program is finished it is checked by the AoC evaluation mechanism, where it generates five different test cases and then checks if the code given by the student solves them all correctly.

The DARK tasks usually demands more time from the student compared to other tasks, and therefore also gives a lot more points for completing them.

8.1.2 Age of Computers Admin

AoC Admin is a tool for administrating users, levels and game objects. The Admin tool is divided into four parts, Level Editor, Question Editor, Text Editor and Admin View. The manual for the Admin tool is available at the AoC Resource page [Nat].

Level Editor

The Level Editor is used to create the game world for AoC. A screenshot of the Level Editor is shown in Figure 8.1. On the left side is the Level Tree, which contains all the objects in the game world. The Level Editor also has a Map panel for displaying the game world, and an information panel and a placement panel for displaying information on the currently selected object in the Level Tree.

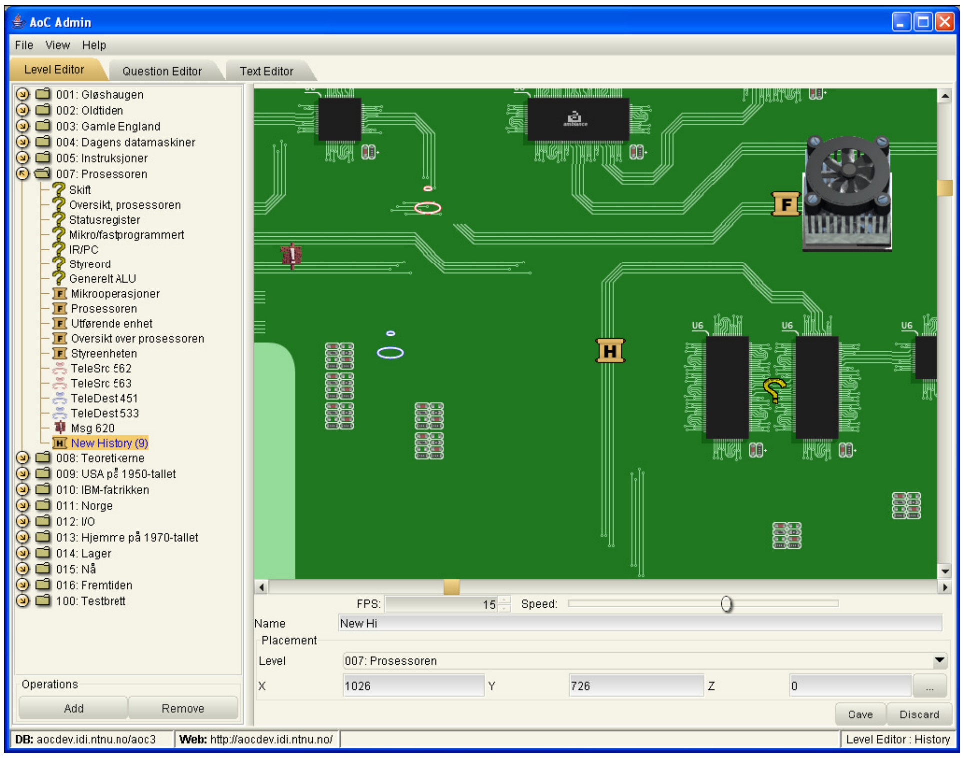
For creating the maps (backgrounds) for AoC and the Admin application, an external tool called Mappy [Til] was used.

Question Editor

The Question Editor is used for creating the tasks described in Section 8.1.1. The question text for the tasks are written in HyperText Markup Language (HTML) or LaTeX. How to create the solutions, answers and code for validating the answers is described shortly in Table 8.1. A screenshot of the Question Editor is shown in Appendix B.1.

Text Editor

The Text Editor is used for adding knowledge to the game as Facts or Histories. Fact object are shown ingame as a scroll with an 'F'. Facts provide the player with a fact that



FPS: 15 Speed:

Name: New Hi

Placement

Level: 007: Prosessoren

X: 1026 Y: 726 Z: 0

DB: aocdev.idi.ntnu.no/aoc3 | Web: http://aocdev.idi.ntnu.no/ | Level Editor : History

Figure 8.1: Screenshot of the Level Editor from the AoC Manual

is useful in its current setting. History objects are shown ingame as a scroll with an 'H'. These provides the players with relevant historical backgrounds. For creating the text for the Fact and History objects, the user writes in HTML or LaTeX. A screenshot of the Text Editor is shown in Appendix B.2.

Admin View

Admin View is divived into two parts, the Settings panel and the User Administration panel. A screenshot of the Text Editor is shown in Appendix B.3.

The Settings panel has three fields:

- **Exam mode** When exam mode is on, all players that has completed the game will have access to all the questions and answers in the game.

- **Style sheet** The style sheet file used for displaying HTML in the game. If no file has been specified, plain style is used.

- **Last Question** The last question in the game, when the player has completed this question and the exam mode is on, he will have access to all the questions and answers in the game.

The User Administration panel is used for administrating the users of the AoC application.

Table 8.1: Solution creation for Questions in AoC

Multiple choice question	The user adds or removes alternatives from a table, and selecting one of the alternative as the correct answer by checking a checkbox next to the alternative in the table.
Input Questions	The user types in the answer for the question that will be checked against the students answers.
Dark Questions	The user creates java code for validating the players assembly code. This code should implement the methods getNumberOfTest(), getArchitecture() and implementTest(). The user also creates a suggested solution to the question.
Auto Generating Questions	The user creates code that generates the question. This can be written in java, and also here it is required a suggested solution to the question. The code will have to implement the three methods, generateParameters(), getQuestionText() and checkSolution().

8.2 Smart Multipurpose Interactive Learning Environment

Smart Multipurpose Interactive Learning Environment (S.M.I.L.E.) is an innovative concept that combines interactive study materials and popular computer games. This is done by giving the teachers the ability to easily transform study materials into educational games without any programming knowledge. The S.M.I.L.E. system supports handicapped learners, deaf or vision impeared, by offering to adapt the user interface by their needs. For hearing impeared, they can get subtitles on all the audio and video files. For the vision impeared, they can control the game by using voice recognition, and all text can be adjusted in size aswell as it can be read to them by computer-synthesized voice. This allows the handicapped to play together with their non-disabled friends and socialize with the community. [DJK$^+$]

The games records the knowledge level of the learners, by recording how well they do on different tasks in the game. It will then be used to offer tasks to the learners at their knowledge level, and will adapt while they progress through the games.

8.2.1 System overview

This section will give a description of the S.M.I.L.E. system and its components. The overview diagram of the S.M.I.L.E. system is shown in Figure 8.2.[DJK$^+$07]

Knowledge Editor

The Knowledge Editor is a fully featured What You See Is What You Get (WYSIWYG) editor, which generates the educational and multimedia materials for the games generated by the S.M.I.L.E. system. The Knowledge Editor has a feature that automatically retrieves multimedia elements (audio and pictures) from the Internet that is relevant to the currently edited knowledge. The multimedia elements can be easily inserted by using drag and drop.

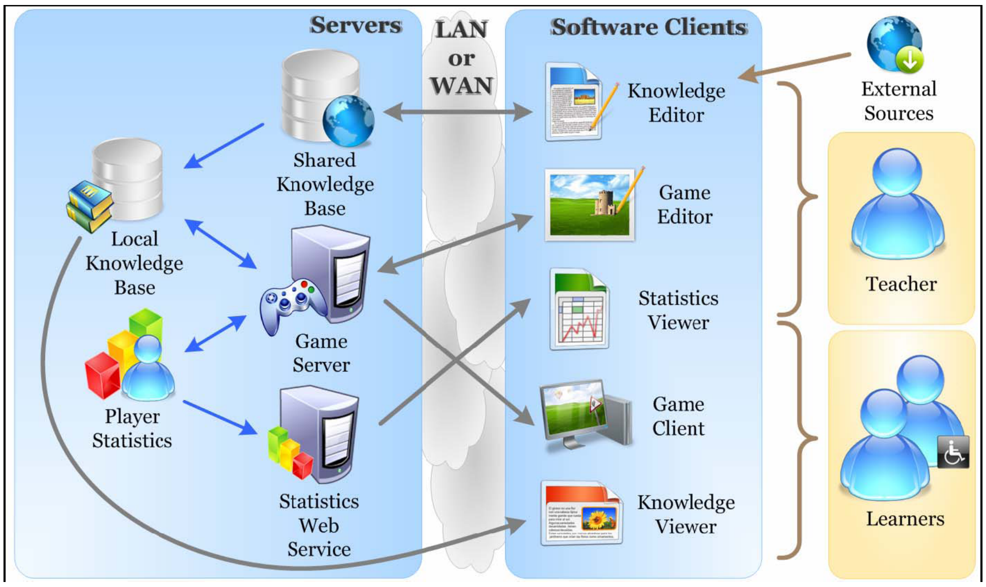


Figure 8.2: S.M.I.L.E. system overview

Game Editor

The teachers can use the Game Editor to create their own educational game, using the stored knowledge materials. They can also store, edit and share the knowledge materials.

Data storage

The games that are generated by the teachers are stored in the Game Server, with all the maps, textures and models used by the games. All the study materials, quests and game objects are stored in the Local Knowledge Base. The study materials is copied to the Local Knowledge Base from the Shared Knowledge Base by the teachers Knowledge Editors. This is done to ensure availability of the study materials and to reduce the load on the Shared Knowledge Base.

Knowledge Viewer

With the Knowledge Viewer the learners can search through the collection of knowledge in the Local Knowledge base. The viewer is available ingame and can be used without interrupting the game.

Part III

Design

Chapter 9

Introduction

In this part we will design an editor for creating new worlds in the World of Wisdom game. This include creating maps (the world), new NPCs and objects, and editing them. In Chapter 10 we describe the most important features for the WoW World Editor, and the improvements for the WoW prototype. We also describe how Game Masters could be a benefitial addition to World of Wisdom. Chapter 11 contains the requirement specification, and Chapter 13 will show how the editor will be integrated with the World of Wisdom game, and more specific details about the design of the editor. This part will also describe the changes needed in the prototype for making a viable editor.

Chapter 10

Features

This chapter contains the most important features to be included in the editor for this project. The chapter also contains an overview of the major features that were implemented in the World of Wisdom prototype and what features should be added in this project. At last we will look at how a Game Master client for WoW can improve the interactions between the teacher and students. In Chapter 11 the more detailed functional and non functional requirements are listed.

10.1 World Editor

In this section we will describe the most important features that will be implemented for the World Editor during this project. Following is a list of the most basic features for the World Editor.

1.Create a world.
This is the most basic part of a world editor, and is needed for creating the areas in the world and making a connection between the areas.

2.Create Non-Player Character (NPC)s.
NPCs are one of the main parts of the World of Wisdom, and a tool for creating and altering the NPC aswell as adding new graphics is needed.

3.Create Static Objects.
Static Objects give much of the feel to the World and is therefore important, as without houses, trees and other objects, the world will feel very empty. A simple tool for adding new Static Objects with graphics are therefore a priority.

4.Painting of background.
Painting of the background is important for the graphics of the world.

Also needed for the World Editor is features that makes it easier and faster for the users to create a world of their own. Following is a list of features for simplifying the creation of a world.

1.Automatic adjusting of attributes for NPCs.
As the NPCs contains a number of attributes, for damage, health, defense, rewards, etc.

It would be benefitial for the users that do not want to use much time adjusting every single attribute, but rather have most of the attribute scale after the level of the NPC.

2.Templates
As creation of a zone may take some time to make it look good, it would be useful to be able to load old zones as templates, and then replace the knowledge used to the desired subject. The final version of the WoW World Editor could then offer some finished zones like cities, forests and similar that the user could use for their kingdom.

3.Automatic gradient background drawing
As drawing every single tile in the background one by one may be tiresome, especially when creating overlays between two different types and make it look nice. Therefore supporting automatic generation of the gradient overlay to the surrounding tiles will make it faster and easier to draw background.

4.Drag and drop
As there may be many objects in a zone, a simple way to move them around and do small adjustment can be done by implementing a drag and drop feature. This would be instead of manually typing in the coordinates for the objects that needs to be placed in the zone.

10.2 World of Wisdom Prototype

In Chapter 2 we explained what the different applications in the prototype are, and what they do. In this section we will describe which specific features were implemented, and what needs to be improved in the prototype during this project.

10.2.1 Implemented features

This section describes the major features that were implemented in the prototype.

Combat

Combat was an important focus in the prototype, this was to show how knowledge can be used in battle. In current implementation it is possible to engage an enemy NPC by using either spells or attack skills. It will then open the Question Panel and the player will have to answer a question in order to damage the enemy. The enemy will retaliate at a certain interval until either the player or the enemy dies.

Movement

Movement is important for exploring the world. In the current implementation it is possible to move around in the map, and for each movement, a message will be sent to the server, which will then send a movement message to all the users connected to that map. When moving, it is also checked for collisions between the users character and the static objects and NPCs.

Network

The network implementation is a large part of the prototype. It sends serializable objects through the network using the TCP/IP protocol in Java.

10.2.2 Improvements

This section describes the major features that should be implemented for making a complete editor.

Zone traversal

Zone traversal means the travel between two zones in the world. This was partially planned in the prototype, but not implemented since there were more important elements to work on.

Zone travelling will be solved by creating squares that can be placed on a map, and linked to another zoning square in another zone. The zoning squares should be adjustable in size, so that it can fit outside an entrance to a house, or in an opening in a forest. When a player moves ontop of the square, they will then be asked if they want to move to the other zone.

Starting of saved world

In the prototype, the world was hardcoded into the startup of the main method of the World Server. This should be change, so when the World Server starts up, it will open a connection to the Database Server and retrieve the names of all the kingdoms that have been saved. You will then have the option to select which kingdom you want to load, and the server will after loading register at the Lobby Server as available.

World parts that need to be stored in database

As the world was hardcoded, there are some tables that need to be added to the database. Also method for storing the objects will need to be added to the Database Server application. These objects are friendly NPC positions, static objects and position table and zone trigger position table. These tables will be described in detail, in the ER model Chapter 14.

Quest

Quests (Missions) are an important part of an RPG game, as they are used to progress and tell the story of the game world. This was not implemented since it is a large project in itself to create a flexible, but easy to use quest system.

The editor should support the possibility of creating quest, this is not implemented in the prototype, but design for how it would work could be made. This would not necessary mean that the quest needs to be implemented in the prototype.

10.3 Game Master

In this section we will look at Game Masters and what features they could use to improve interaction between the course staff and students. The Game Master client will not be designed and implemented during this project.

There are many advantages by having Game Masters in the game. Most commonly in the commercial MMORPG they are used for reporting bugs that are discovered by players, and to temporary fix any problems they come across. They also sometimes creates more dynamic events with the players to make the world feel more alive. Another important role for Game Masters is to look for people exploiting bugs in the game that gives the players an advantage over other players.

When it comes to World of Wisdom, there are several advantages by having Game Masters. In the following sections we will look at some of the more important ones.

10.3.1 Logging

One important feature would be to log interactions that happen on the server. The World of Wisdom already have a system for logging actions to the hard drive in text files. This system could be improved to create one text file for each player or character, so when a bug or exploit is reported they can be checked against the log files. These files can then be parsed, and displayed in the Game Master Client, so that the course staff can easily analyse and check the data.

10.3.2 User statistics

As with the Smart Multipurpose Interactive Learning Environment (S.M.I.L.E.) system, as described in Chapter 8, World of Wisdom would benefit from having a statistics system over the students progress through the game. This can then be used to find out which students have trouble with the course materials, and give extra attention and help to them. But it can also be used to find out if there are some questions that are generally too difficult or ambiguous , or that are too easy. Ambiguous questions can be checked by seeing that many students have gone for the same wrong answer, and too difficult questions will be answered wrong by most, while too easy are answered correct by all or most. This can help balance the game and its questions for the next semester or school year. The teacher could also use this information to set up extra lectures on the topics that the students struggle the most with.

10.3.3 Additional functions

When using a Game Master Client, they can log on to the world and run around as any other player. But a Game Master usually have additional functions that are not available to the normal players. As with the Neverwinter Nights Dungeon Master client in Section 7.1. Following is a list describing some important functions for Game Masters.

Travel should be alot faster then for players, as they need to be able to get to the players in need of help quickly. This can be done by both increasing the normal travel speed of the Game Master, and give the option to jump to any zone in the kingdom.

Stats alteration should be possible to allow the Game Master to temporarly fix bugs involving quests or creatures. For example, due to a bug, a quest update was not executed and the player did not receive a needed quest item. The Game Master should then have the ability to create this item and correctly update the quest log. Also if a certain type of creature that is needed for quests stops respawns, or in another way is bugged. Then they should be able to reset them, or create new ones.

Creation , the Game Master should be able to create additional challenges. As World of Wisdom encourages to play in group with others and cooperating, it may increase the fun if they get a some additional challenges.

Apperance can be altered, so that when they want to create an event, they can be displayed as a creature or NPC. They should also be able to turn invisible to the players, so they do not disturb the gameplay when they are observing.

Report , using special ingame system for reporting bugs or exploit, that will be sent to the developers. The report should include some information about the state and what objects were involved, which will make it easier for the developers to fix the problem.

Private chat channel that the players can send to, but not read from. This can be used by the Game Masters to discuss problems amongst themselves, and used by players to report problems to the Game Masters. There should be some indication to a Game Master being online. In Age of Computers (AoC) when some of the course staff was online, their name showed up with colored text in list of people online. Other games like Neverwinter Nights (NWN) announce to all the players when a Game Master logs on.

Chapter 11

Requirements

This chapter contains the functional and non-functional requirements for the project. Each requirement has a priority and difficulty. Requirements with high and medium priority will be implemented in the prototype of the editor. Low priority will not be implemented at first, but if there is time, they will be implemented aswell.

11.1 Functional requirements

11.1.1 Database and Storage interface

FR1 It should be possible to add new or delete NPCs (both friendly and enemy NPC) in the Database.
Priority - High *Difficulty* - Low

FR2 It should be possible to alter information on NPCs. (e.g. Name, Question Topic, Attributes, Respawn time, Dialog etc.).
Priority - Medium *Difficulty* - Low

FR3 The attributes should scale after the level selected for the NPCs (e.g. Primary attributes, offensive and defensive attributes). *Priority* - Medium *Difficulty* - Low

FR4 Be able to add new or delete Static Objects in the Database.
Priority - High *Difficulty* - Low

FR5 Be able to alter existing Static Objects in the Database.
Priority - High *Difficulty* - Medium

FR6 Be able to create quests and connect them to Friendly NPC.
Priority - Low *Difficulty* - High

FR7 Be able to store the world with object references in the Database.
Priority - High *Difficulty* - High

FR8 Be able to load a world from the database.
Priority - High *Difficulty* - High

FR9 Be able to store and load the world locally.
Priority - Medium *Difficulty* - Medium

11.1.2 Map editing

FR10 Be able to create maps(zones) by specifying map size in width and height.
Priority - High *Difficulty* - Medium

FR11 Be able to create Zone Triggers and connection between them.
Priority - High *Difficulty* - Medium

FR12 Drawing background using a tileset. Selecting a individual background tilepicture and placing it in the background.
Priority - High *Difficulty* - Medium

FR13 Drawing background using type (sand, grass, mud etc.). Automatically calculating which tile to use to create overlay between types.
Priority - Low *Difficulty* - High

FR14 Be able to place world objects (static objects, friendly NPCs, enemy NPCs) on the map.
Priority - High *Difficulty* - Medium

FR15 When adding a enemy NPC to the world, the user select a theme topic(Question topic), creature type and level. *Priority* - Medium *Difficulty* - Medium

FR16 Be able to change position of world objects.
Priority - Medium *Difficulty* - Medium

FR17 Use drag and drop to place new creatures or objects.
Priority - Medium *Difficulty* - Medium

FR18 Use drag and drop to change the position of creatures or objects.
Priority - Low *Difficulty* - Medium

FR19 Create quest relations between friendly and enemy NPC.
Priority - Low *Difficulty* - High

FR20 Be able to create templates and load templates of zones from database. Using templates will greatly reduce the time needed to make a world ready for players.
Priority - Medium *Difficulty* - Medium

11.1.3 Prototype improvements

Database Server

FR21 Storing friendly NPC position and zone in the database.
Priority - High *Difficulty* - Medium

FR22 Storing Zone Trigger information in the database (Position, size and target zone).
Priority - High *Difficulty* - Medium

FR23 Storing static object in the database.
Priority - High *Difficulty* - Medium

FR24 Storing static objects NPC position and zone in the database.
Priority - High *Difficulty* - Medium

World Server

FR25 When starting the World Server application, it will connect to the Database Server and retrieve a list over all kingdoms.
Priority - Medium *Difficulty* - Medium

FR26 When a player moves ontop of a zone trigger, the server will send a message to the player, asking if he will move to another zone.
Priority - Medium *Difficulty* - Medium

FR27 Be able to load a kingdom from the Database.
Priority - High *Difficulty* - High

Client

FR28 A popup window will be displayed when a player moves ontop of a zone trigger. A message will be sent to the world server if he answers yes. And he will be moved to the new zone.
Priority - High *Difficulty* - Medium

11.2 Non-Functional requirements

FR1 The development language for the Editor will be Java.
Priority - High *Difficulty* - Low

Chapter 12

Design patterns

Design patterns are a combination of components, class and object, that provides a solution for common design problems. In this chapter we will look at the design patterns that were used in the World of Wisdom Prototype and that will be used in the World of Wisdom Editor.[Bra01]

12.1 Singleton

The idea of the Singleton design pattern is to only have one instance of an object. This is done by hiding the constructor, and the object is retrieved by using method to get the instance. The method will return the object, if it does not exist the method will create a new one. The Singleton is useful in situation where there can only be one object, and the object can be used by several classes.

In the World of Wisdom prototype, singletons were used for network instances and GraphicObjectFactory.

12.2 Object Factory

The idea of the Object Factory design pattern is to collect the creation of several similar types of objects in one class. This is useful when the objects and their constructor are subject to change, and to create different creation methods for the same object.

Object Factory was used in the World of Wisdom prototype for creation graphical objects in the GraphicObjectFactory.

12.3 Observer

The Observer design pattern is divided into to parts, the subject, and the observer. The subject contains a list of observers, and when a change happens in the subject, the observers are notified. This pattern is often associated with the Model-View-Controller (MVC) architectural pattern. In this pattern the model is the subject and the view is the observer. When the model changes, the view is updated with the new information.

In the World of Wisdom prototype the MVC pattern was used in the client, where the Graphical User Interface (GUI) was the view, and the model was the World model.

12.4 Client-Server

The Client-Server architectural pattern divides the application(s) into components where the server components serves the request of the client component. The Client-Server approach has an advantage of a low coupling between the components.

In the World of Wisdom prototype the applications are divided into the Client-Server relationships, where the all the applications except the Database Server acts as clients, and all the applications except the Client acts as Servers.

Chapter 13

System Overview

In this chapter there will be a description of the system overview diagram, and how the implementation of the editor will interact with the World of Wisdom project. In Figure 13.1 an overview of the system is shown with the WoW World Editor as an addition.

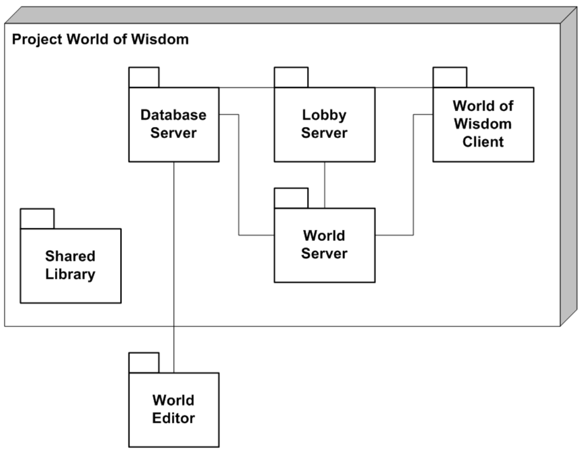


Figure 13.1: Package diagram over the World of Wisdom

13.1 World Editor

This projects goal is to design and implement the World Editor part of the system. It will allow the teaching staff to create and edit a world designed for World of Wisdom. The created world will then be stored on the Database server, and selected when starting the World Server. The World Editor will be described in more detail in Chapter 16.

13.2 Database Server

The database server is the central storage for World of Wisdom. It contains all the information about the world, position of objects, attributes of the objects and authorization data.

The World editor will be able to store the world by sending the information to the Database Server, and retrieve a world by sending request for a specific world.

13.3 World Server

The World Server is the server that keeps control over the status of a world, the objects within and the players that are connected to the world.

When starting the World server, the server will retrieve a list over the worlds that have been stored at the Database server. The teaching staff will then select their world and start the server.

13.4 Shared Library

Shared library contains classes that are used by several of the applications in the World of Wisdom project. Most used are the Models package, that includes objects like Zone, Character, and EnemyNPC. The Network package is used to start a listener for connection, or for connecting to other servers. Network package also contains classes that are serialized for sending messages over the network using TCP/IP connection.

Chapter 14

Entity Relation model

This chapter will describe the alterations and additions needed to the Entity Relation model of the prototype. The original ER-Model of the prototype can be found in Appendix C.

14.1 Alterations

When the prototype were being developed, it was made with the assumption that it would be used for further development in the future. So there are not many alterations that are needed on the database, since most of the database were implemented even though some of them were not even used in the prototype. The original ER-Models are shown in Appendix C.

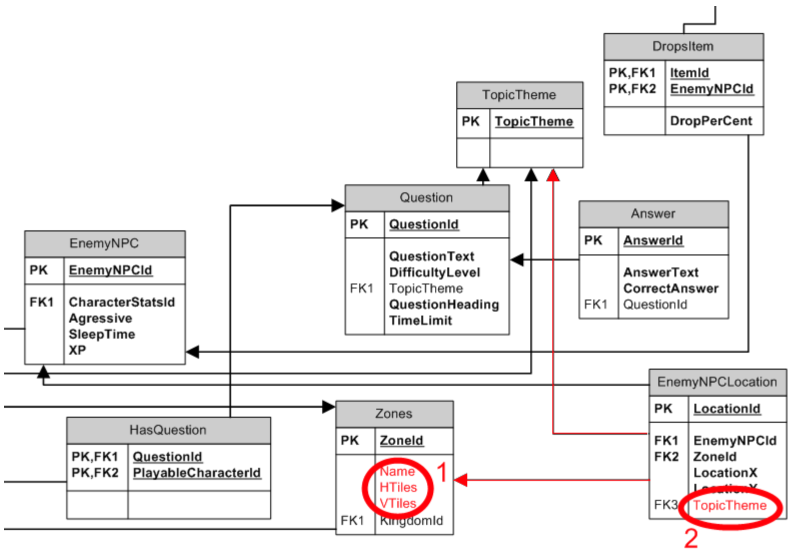


Figure 14.1: Updated change in ER-Model

14.1.1 Zones

In the table Zones, three additional fields will be added, Name, HTiles and VTiles. Name will be the name of the zone. HTiles, the number of horizontal tiles in that zone. VTiles the number of vertical tiles in that zone.

Tile numbers will be used when retrieving the background array from the storage server. The change is shown as number one in Figure 14.1 in red.

14.1.2 Theme Topic

The foreign key in the EnemyNPC table that points to the TopicTheme table will be moved to EnemyNPCLocation. The change is shown as number two in Figure 14.1 in red.

This is to make it possible to add several NPCs of the same type to a world, or even same zone, and have the possibility to have different theme topic connected to them. Then different kingdom can use the same enemy NPCs and thus saving some work when creating a new world.

14.2 New Tables

Some new tables are needed for storing the information generated when creating a world. Shown in Figure 14.2 are the new tables, with the connection to the original tables. Original tables are faded.

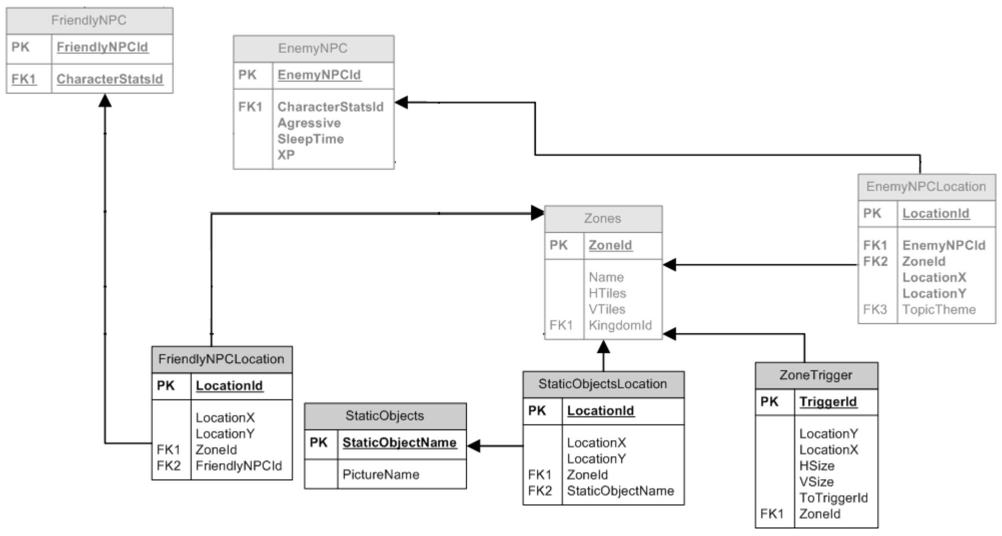
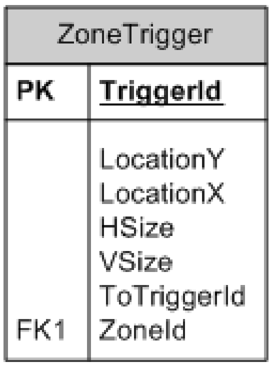


Figure 14.2: New tables in ER-Model

14.2.1 Zone Trigger

For storing location and pointers for Zone Triggers. The table will contain TriggerName, which is the primary key. LocationX and LocationY which tells the coordinates for the trigger in the zone it is placed in. HSize and VSize tells the pixel size of the trigger. ToTriggerId is the id of the trigger it points to, most often to other zones, but can also be used to travel to another place in same zone. And ZoneId, the id of the zone where the trigger is placed. Figure 14.3 shows the table for ZoneTrigger.

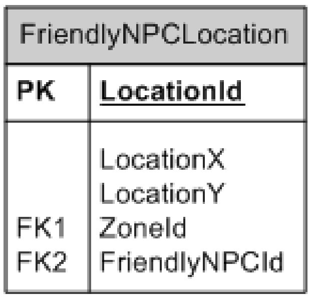


ZoneTrigger	
PK	**TriggerId**
	LocationY
	LocationX
	HSize
	VSize
	ToTriggerId
FK1	ZoneId

Figure 14.3: Table of ZoneTrigger

14.2.2 Friendly NPC Location

For storing location of a Friendly NPC. Primary key is LocationId which is automatically incremented. LocationX and LocationY determines the position of the Friendly NPC in the zone. ZoneId is the id of the zone the Friendly NPC is placed in. And FriendlyNPCId is the id of the NPC that have been placed. This is to make it possible to place the same NPC in several places, in cases of "Commoners" or "Guards" which can be many of in the same zone or world. Figure 14.4 shows the table for FriendlyNPCLocation.

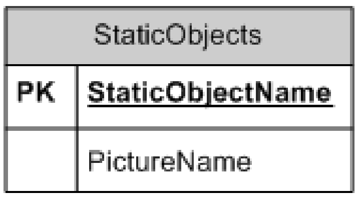


FriendlyNPCLocation	
PK	**LocationId**
	LocationX
	LocationY
FK1	ZoneId
FK2	FriendlyNPCId

Figure 14.4: Table of FriendlyNpcLocation

14.2.3 Static Objects

For storing information about a static object. The primary key is StaticObjectName which is the name of the object. PictureName is the name of the picture assosiated with the static object. Figure 14.5 shows the table for StaticObjects.

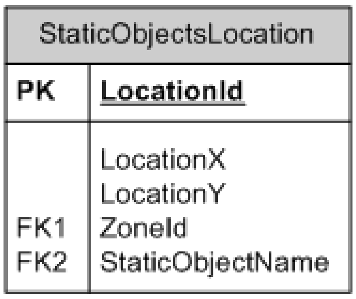


StaticObjects	
PK	**StaticObjectName**
	PictureName

Figure 14.5: Table of StaticObjects

14.2.4 Static Objects Location

For storing the location of a static object. StaticObjectsLocation is similar to FriendlyN-PCLocation 14.2.2. LocationX and LocationY for position in zone. ZoneId for the id of the zone it is placed in, and foreign key StaticObjectName that points to the StaticObjects table. Figure 14.6 shows the table for StaticObjectsLocation.



StaticObjectsLocation	
PK	**LocationId**
	LocationX
	LocationY
FK1	ZoneId
FK2	StaticObjectName

Figure 14.6: Tables of StaticObjectsLocation

Chapter 15

World of Wisdom Prototype

This chapter will describe the additions that are needed to the WoW Prototype, and how they will be implemented.

15.1 Database Server

For the Database Server application, it needs a new network listener that will listens to Editors. Also some methods for retrieving objects from the database and inserting or updating the new objects is also needed which will be described in the following sections.

15.1.1 Editor Listener

As with the Lobby Listeners and World Server listeners from the prototype, the Editor Listeners will be split into two classes. The EditorListener and the EditorConnection-Thread.

EditorListener

The EditorListener listens to new connection on a specified port. When a new connection is made, it creates a new thread that holds the connection in the EditorConnectionThread class.

EditorConnectionThread

The EditorConnectionThread class is the one that listens for messages from the Editor, and retrieves data from the Database and sends back to the Editor.

15.1.2 DatabaseGetters

The DatabaseGetters is a class with static helper methods that are used for retrieving objects from the Database. Since in this project we will add some new objects and tables to the database, new methods are needed for retrieving them. There are also some objects that is implemented in the database, but no methods for made for retrieving them. In Table 15.1 a list of the retrieval methods are described.

Table 15.1: Methods needed in DatabaseGetters Helper class

getStaticObjects	Retrieves all Static Objects from the Database.
getStaticObjectLocation	Retrieves all locations of static objects belonging to a specified Zone.
getZoneTrigger	Retrieves all ZoneTriggers belonging to a specified Zone.
getEnemyNPCLocation	Retrieves all locations of Enemy NPC belonging to a specified Zone.
getFriendlyNPCLocation	Retrieves all locations of Friendly NPC belonging to a specified Zone.
getZone	Retrieves a Zone object by zoneId.
getZones	Retrieves all Zone object by KingdomId.
getKingdom	Retrieves kingdom information by kingdom-Name.
getZoneFromTemplate	Retrieves a template of a Zone.

15.1.3 DatabaseSetters

The DatabaseSetters is a class with static helper methods that are used for updating or insterting objects into the Database. As with the DatabaseGetters, insert methods are needed for the new tables and objects. In the prototype there were some objects that were directly inserted in the Database, by using the PostgreSQL Admin program. These objects will need methods for inserting aswell. In table 15.2 a list of the insert and update methods are described.

15.2 Client

The Client applications need some changes for supporting travels between two zones. This will be a popup window for when the player triggers the ZoneTrigger, and the network messages for sending reply to the World Server. This section will describe the changes that will be done in detail.

15.2.1 TravelPanel

TravelPanel is a new class that will be used to display a message to the user when they use at ZoneTrigger. The message will consist of the name of the place they are about to go, and with options to either travel there or refuse. Figure 15.1 shows the class diagram of the TravelPanel.

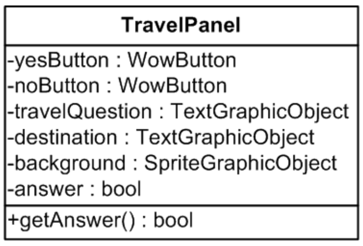

TravelPanel
-yesButton : WowButton
-noButton : WowButton
-travelQuestion : TextGraphicObject
-destination : TextGraphicObject
-background : SpriteGraphicObject
-answer : bool
+getAnswer() : bool

Figure 15.1: TravelPanel class diagram.

Table 15.2: Methods needed in DatabaseSetters Helper class

updateAnswer updateQuestion	Update methods for Answer and Question, update was not implemented in the Admin tool in the prototype.
deleteAnswer deleteQuestion	Delete methods for Answer and Question, delete was not implemented in the Admin tool in the prototype.
insertEnemyNPC insertFriendlyNPC insertStaticObject	Insert methods for World objects, these were manually inserted in the prototype.
insertKingdom	Inserts information about the kingdom.
insertPlayerAccess	Inserts player access to a kingdom. This decides which players will be able to connect to the kingdom, when it has been loaded by a World Server.
updateZone	Updates the Zone information and all the objects within the Zone.
updateEnemyNPCLocation updateFriendlyNPCLocation updateStaticObjectLocation updateTravelTrigger	Helper method called by updateZone.
insertZoneTemplate	Inserts a zone without kingdomId.
insertZone	Inserts a zone connected to a KingdomId.
insertEnemyNPCInZone insertFriendlyNPCInZone insertStaticObjectInZone insertTravelTriggerInZone	Helper method for inserting World Objects in zone. Used by insertZone.
deleteZone	Deletes a Zone by Id and all objects connected to the zone.
deleteEnemyNPCLocation deleteFriendlyNPCLocation deleteStaticObjectLocation deleteTravelTrigger	Used by updateZone when World Objects have been removed.

The GetAnswer method will be used by the MainWorldPanel to check if the player has answered the question or not.

15.2.2 MainWorldPanel

The MainWorldPanel displays the graphic in the game world, and the TravelPanel will be added to this class. When a player triggers a travel to another Zone, MainWorldPanel will create a new instance of the TravelPanel with the information of their destination. It will also wait for the player to answer yes and when he/she does, the MainWorldPanel will send a message to the World Server asking for moving the player to a new Zone.

15.3 World Server

As the kingdom was hardcoded into the GameWorld class, some changes must be done in the World Server application to make it able to retrieve and use kingdoms that are stored in the Database.

15.3.1 ServerControl

Upon starting the ServerControl it needs to retrieve the names of all the kingdoms in the Database. This is done so that the user will be able to select from the list of names, which Kingdom they want to load. This dialog box can be created by using the JOptionPane with the option to display a list. When the kingdom has been selected it will ask the Database for the kingdom and when it is returned, insert it into the GameWorld object.

15.3.2 GameWorld

The GameWorld object holds the Kingdom. In the constructor, the hardcoded test kingdom was placed in the prototype. This will be removed and it will use the Kingdom that is retrieved from the Database instead. Most of the code for handling this was already implemented in the prototype. So not many changes are needed here.

15.4 Shared Package

The Shared Package holds all the models and message objects used by the network. One new model is needed which is the ZoneTrigger that is further described in section 15.4.1. Some new message objects will be implemented for sending the kingdom and all the objects lists over the network to the World Server and Editor.

15.4.1 ZoneTrigger

ZoneTrigger is the new class that stores the information about its current location and size, and its target ZoneTrigger. The class diagram of the ZoneTrigger is shown in figure 15.2.

15.4.2 Zone

Zone will have a new list of objects, which is the ZoneTriggers. The ZoneTriggers will be stored in a HashMap using its Id as key.

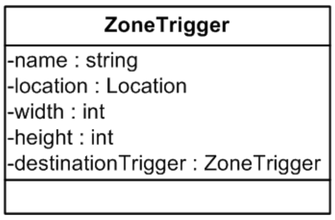


Figure 15.2: ZoneTrigger class diagram.

15.4.3 Network Messages

As new objects are added to the Database, they will need to be added as an object type in the enum ObjectType. ObjectType is used by the ObjectMessage which is shown in Figure 15.3. The message can hold any type of Object, and the ObjectType is used to cast the object into the correct type when it is received. ObjectMessage and ObjectType was implemented and used in the prototype. So only new types needs to be added here.

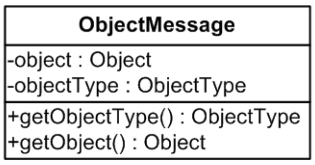


Figure 15.3: ObjectMessage class diagram.

A kingdom message is also needed, as the kingdom does not have a single class, and is comprised of several objects, the ObjectMessage can not be used for this. The KingdomMessage will need to store a list of all the Zones in the Kingdom and the name, id and subject of the kingdom. It also needs a list over all the players that has access to the kingdom. Figure 15.4 shows the class diagram of the KingdomMessage.

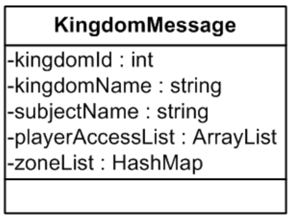


Figure 15.4: KingdomMessage class diagram.

Chapter 16

World Editor

This chapter contains the detailed design of the World Editor. In Figure 16.1 the classes of the World Editor is shown and what is used from the Shared library. The World Editor will use the Client-Server architecture where the World Editor will be the Client, and connect with the Database Server. The World Editor will also use the Model-view-controller pattern, where the WorldModel and objects lists will serve as models and the GUI will be the view and controller.

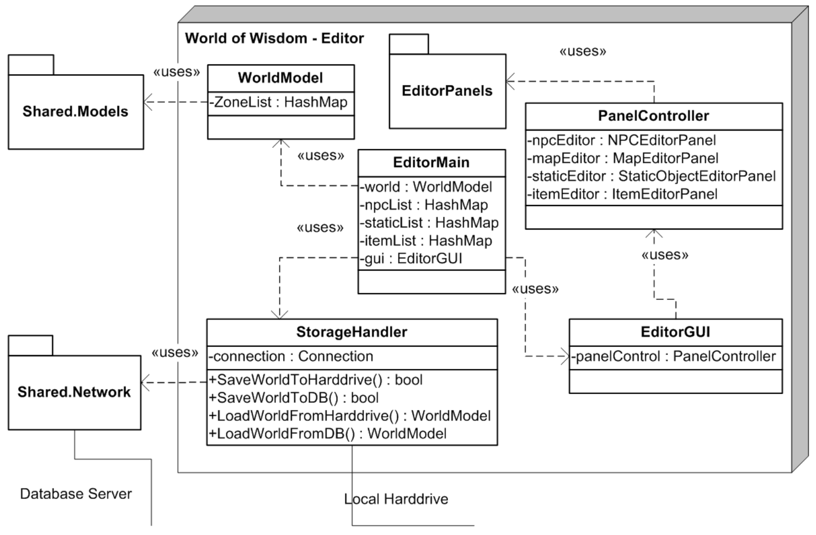


Figure 16.1: Class diagram of the World Editor

16.1 EditorMain

EditorMain is the main class of the editor and contains the main method used by java to start the application. When the editor starts, it will use the StorageHandler to retrieve lists of the objects stored in the database, and the user will be given the option to create a new world, or select a world from the database to edit.

EditorMain will also add the different parts of the GUI as listeners to the Model objects as according to the Observer pattern described in Section 12.3. It will also be the Observer of the StorageHandler, where the StorageHandler will send updates to the EditorMain when receiving objects over the network or from the hard drive.

16.2 WorldModel

WorldModel class contains a list of the zones in the world that is currently being edited. In each of the zones there will be lists of worldobjects, a background array and a list of triggers.

16.3 StorageHandler

StorageHandler will hold the connection to the Database Server, and be responsible for storing and loading objects into the editor. There will be two options for storing and loading the world, one from the database, and the other from the harddrive.

Using the hard drive is for temporary storage, like automatic saving every 5 minutes, in case of system failure. It will also be used for storage when the connection to the database is down, or when the user chooses to store the world on the harddrive. Because it hasn't been finished or isn't working properly.

The StorageHandler will use the Singleton design pattern described in Section 12.1. This is so that the StorageHandler can be used from several parts in the Editor.

16.4 EditorGUI

EditorGUI is the main graphics class, it will contain a menu, the PanelController and will inherit from the JFrame class in the Java Swing library. Figure 16.2 shows a draft of the user interface. The graphic layout is based on similar editors as Aurora Toolset and the level editor for Age of Computers.

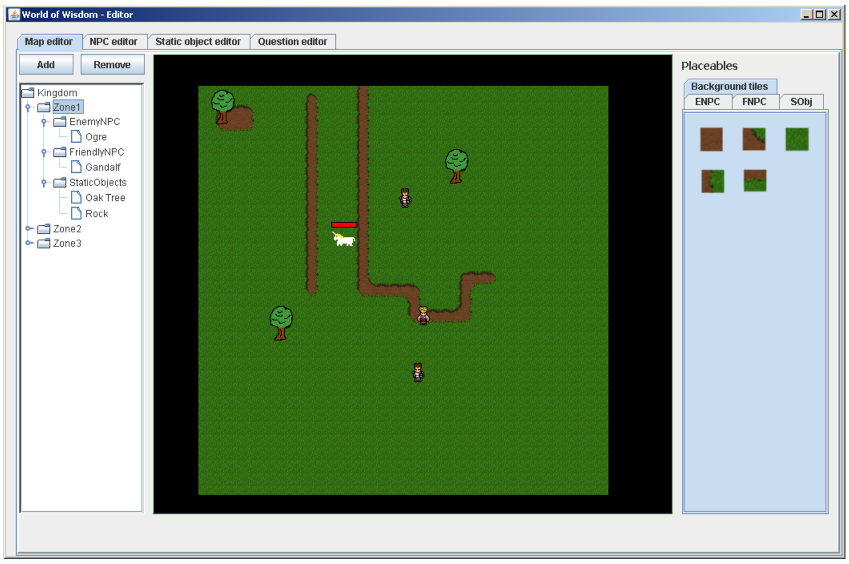


Figure 16.2: Draft of the Editors user interface (Created with NetBeans and Paint.NET)

16.5 PanelController

PanelController contains the panels for the editor, shown in Figure 16.1. PanelController controls which part of the editor is currently displayed, and updates the components when changes occurs to the underlying models.

16.6 EditorPanels

EditorPanels is a package that contains the different panels used in the graphical user interface for the editor. Figure 16.3 shows the contents of the package. In the following section there will be further description of each panel.

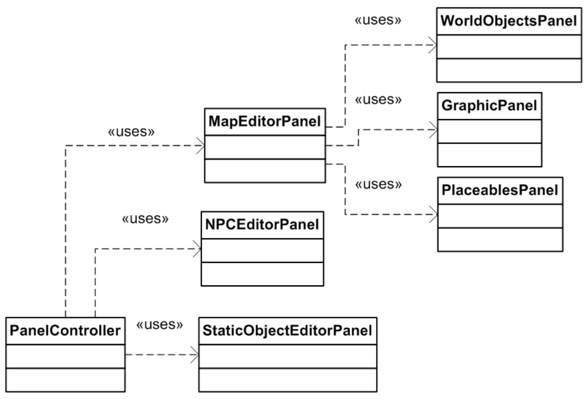


Figure 16.3: Class diagram of the panels in the User Interface

16.6.1 MapEditorPanel

The MapEditorPanel is the largest part of the editor. A draft of the panel is shown in Figure 16.2. In the panel you can add new zones to a world, add NPC and static objects, and draw background using a set of background tiles.

16.6.2 NPCEditorPanel

NPCEditorPanel is an interface to the Database, which allows to edit existing NPCs and add new ones. You can also add new pictures for the NPC.

16.6.3 StaticObjectEditorPanel

StaticObjectEditorPanel can add new static objects templates to the database, or edit existing static objects. The Static Object Editor should also support adding new graphics.

16.6.4 QuestionEditorPanel

QuestionEditorPanel was created during the development of the WoW prototype as an example of an administrating tool. It shows the questions that are already in the Database, and allows for adding new ones.

Part IV

Implementation

Chapter 17

Introduction

This part contains the details of the implementation. In Chapter 18 we describe the changes made to the PostgreSQL database and its tables since the Design part. In Chapter 19 we will look at the changes done to the World of Wisdom Prototype, and in Chapter 20 we will look at the final implementation of the WoW World Editor and its class diagrams.

Chapter 18

PostgreSQL database

From design, only a few minor changes has been done to the PostgreSQL database. This chapter will give an overview of the changes and show the finished ER-Models of the database. The tables with changes done to them are shown in Figure 18.1.

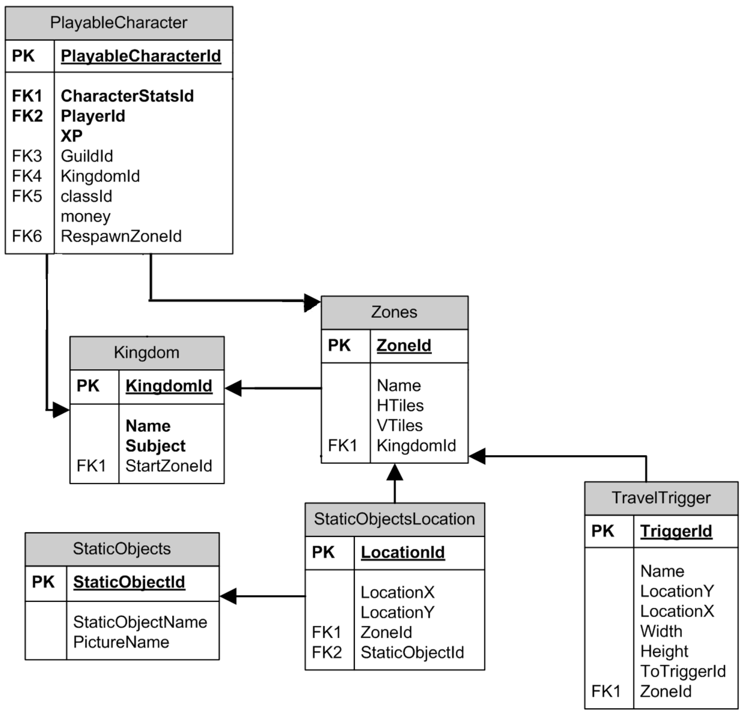


Figure 18.1: ER-Model of the tables changed since design

18.1 Zones

The table that stores the Zones in the world have been changed to include a respawn point. This is the location where the players character will appear when connecting to a world, or when the character is defeated in combat. Two new fields have been added to the table to store this information, RespawnX and RespawnY which stores the coordinates for which the respawnpoint is located.

The field BackgroundPath was also added. This stores the location and name of the background file stored on the computer that runs the Database application.

18.2 RespawnPoint

RespawnPoint was originally thought to be used for the respawn points in the prototype, but was never implemented. With a seperate table it would be possible to store several respawnpoints for one zone. But as the zones are not that large, it would not be needed to have more then one in each zone. The RespawnPoint were then moved to the Zones. And the RespawnPoint table was deleted.

18.3 PlayableCharacter

As the PlayableCharacter stored a RespawnPoint, which would be the place where the players character would be when he logs in, it was changed to fit the new storage of respawnpoints. The PlayableCharacter now stores the id to the zone where the character will be when he logs in.

18.4 StaticObject

In design the StaticObject used the name of the Static object as a primary key, as the name will be unique for each object. This was changed to use a auto incremented integer as Id instead. It is easier to use when making updates to the tables when changing the name, and will not require cascading updates to other tables (StaticObjectLocation) when the name is changed. StaticObjectLocation now uses StaticObjectId as foreign key instead of name.

18.5 Kingdom

To the Kingdom table the field StartZoneId was added during implementation. This is used to tell which zone in the Kingdom is the starter zone for new players.

18.6 TravelTrigger

Known as ZoneTrigger in design. TravelTriggerId was changed to an auto incremented integer, and the field Name was added. The fields HSize and VSize was changed to Width and Height.

Chapter 19

World of Wisdom Prototype

This chapter describes what has been implemented in the World of Wisdom prototype applications. What has been changed since design Chapter 15, and why it has been changed. During the implementation some new features were also added, these will be described in detail and the reason for why they were added.

19.1 Shared Package

This section will describe the changes that was done to the Shared Package and the reason for the changes.

19.1.1 Settings

A new filehandler has been added to the Shared Package called Settings. This class reads a file on the harddrive and sets the variables in the class according to the data in the file. Settings is used to store network settings like ip-adress and port numbers. It is used by all the applications, so that you no longer need to do the changes in the code and recompile, making it easier to set up the system. Figure 19.1 shows the Settings class diagram.

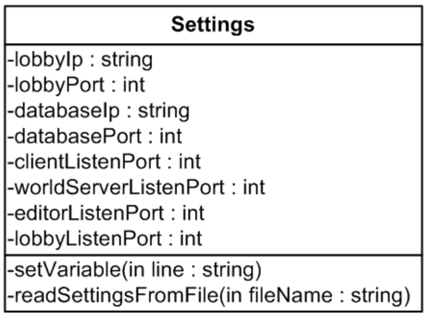


Figure 19.1: Settings class in the Shared Package

19.1.2 KingdomMessage

As the Kingdom table in the Database got an additional field (StartZoneId in Section 18.5), another field has been added to the KingdomMessage object. The StartZoneName, this holds the name of the Zone that a new player will start in. The new class diagram is shown in Figure 19.2.

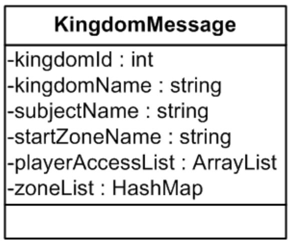


Figure 19.2: KingdomMessage class in the Shared Package

19.1.3 TravelTrigger

Some changes were done to the ZoneTrigger since design. The class was renamed to TravelTrigger, which describes the class better. It now inherits from the WorldObject abstract class in the Shared Package (Figure 19.3). This is so that it can be used easier when drawing the TravelTrigger in the Editor and the Client Applications.

Instead of using another TravelTrigger object as destination, we now use the id of the destination TravelTrigger and the name of the Zone it is in. This is to make it easier when refering to other TravelTrigger in the Editor, and when sending information to the World Server, you only need the id and target zone.

The new class diagram of TravelTrigger is shown in Figure 19.3.

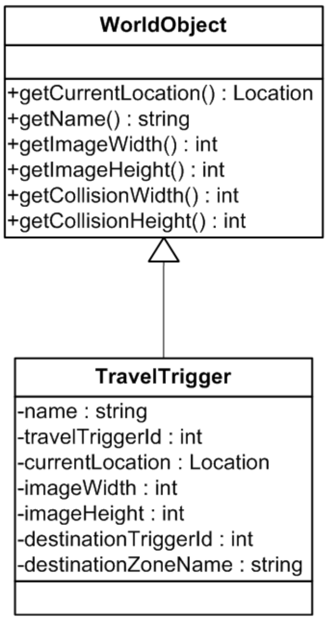

WorldObject
+getCurrentLocation() : Location
+getName() : string
+getImageWidth() : int
+getImageHeight() : int
+getCollisionWidth() : int
+getCollisionHeight() : int
TravelTrigger
-name : string
-travelTriggerId : int
-currentLocation : Location
-imageWidth : int
-imageHeight : int
-destinationTriggerId : int
-destinationZoneName : string

Figure 19.3: TravelTrigger class in the Shared Package

19.2 Database Server

The Database Server is a small application, that does operation on the PostgreSQL database. The Lobby Server, World Server and WoW World Editor connects to the database application through the network, and sends requests for objects and for altering or inserting new objects to the database. The main method in the application is DBServerMain. Figure 19.4 shows an overview of the classes in the Database

Server. The DBServerMain starts an instance of each of the three listeners. In this project EditorListener is the new addition, for listening to requests from the editors that needs objects from the database. When an editor connects, the database starts a EditorConnectionThread that handles requests from the connected editor. The thread then uses DatabaseGetters and DatabaseSetters for retrieving and inserting information to the PostgreSQL database.

The network settings for the database has been moved outside of the code to a file that is loaded on startup of the Database Server. File is located in "Resources/Settings" and is called Database.ini. This contains the port numbers used for listening to connections from the World Editor, Lobby Server and World Server. It also contains information needed to connect to the PostgreSQL database, the name of the database and the ip-adress of the computer which contains the PostgreSQL.

When the user now starts the Database application they will be asked for the username, and then the password for connecting to the PostgreSQL database.

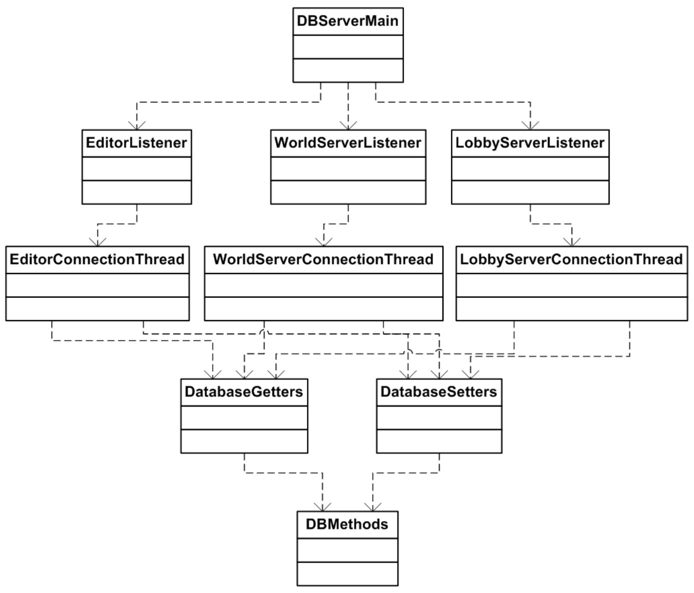


Figure 19.4: Class diagram of the Database application

19.3 World Server

As the networksettings was hardcoded, everytime a server changed ip, the code needed to be changed and recompile. This has now been changed to load the variables from a file on startup instead. For the World Server it loads the settings from Worldserver.ini which is placed in "/Resources/Settings", using the class Settings in the Shared Package.

The kingdom was previously hardcoded with a small test world, used for demonstrating the World of Wisdom prototype. Now when the World Server is started, a list of all the kingdoms in the database will be shown for the user. The user selects the kingdom they want to load into the World Server. When a kingdom has been successfully loaded, the World Server will register at the Lobby Server as available and clients can connect to the world.

In the prototype the players were all put in the same zone at the same coordinates when connecting to the world. Now the client is put in the zone that is set as respawnzone for that character in the database.

Support for the travel trigger system has been added to the World server. This means that when a client selects a travel trigger, and agrees to travel to another zone. A message will be sent to the World server, and the World server will move the client into the desired zone, and return a message with an updated version of the zone the client is moving to. Message flow is shown in the sequence diagram in Figure 19.5. When moving clients to or from a zone, a message is broadcasted to all clients in that current zone that a user has been removed or added.

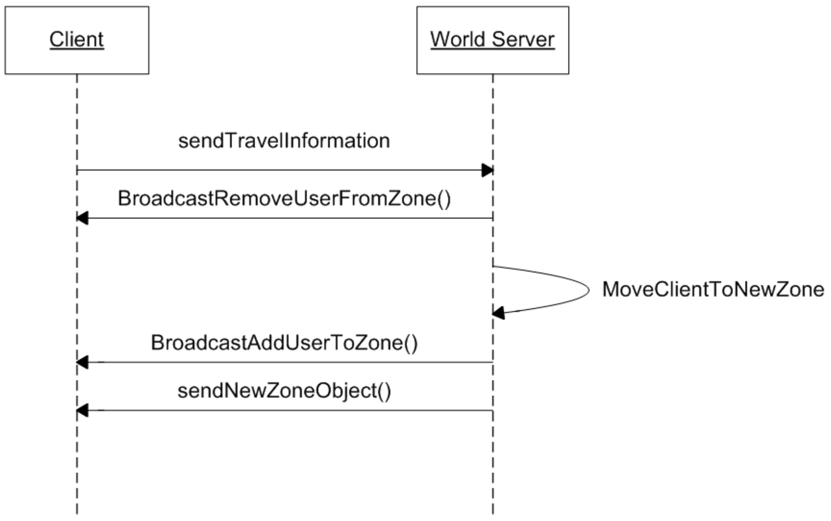


Figure 19.5: Sequence diagram for message flow between World server and client.

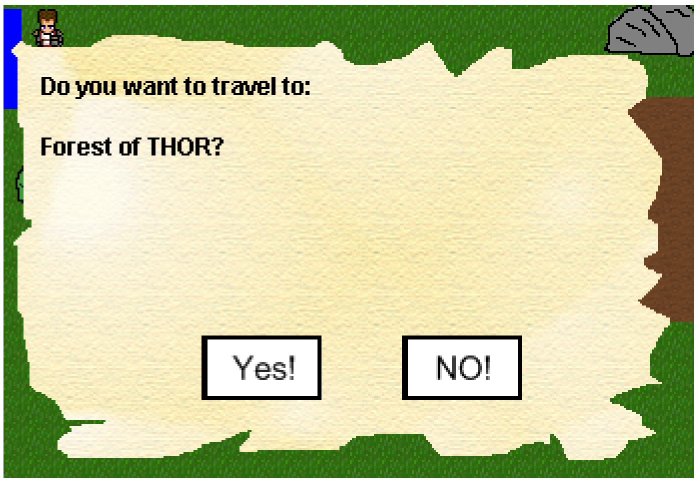
19.4 Lobby Server

When a new kingdom has been created, there are no playable characters connected to that world. As creation of characters happen before connecting to a world, the Lobby server handles request with new playable characters. This has been implemented and the Lobby server receives a message with the name and class of the new character from the client. The Lobby server then creates the new character and sends the object to the database for storage.

As with the World Server, Lobby Server now loads its network configurations from a file, Lobbyserver.ini, on start-up.

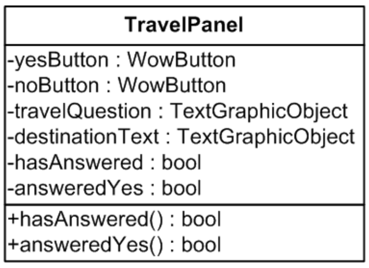
19.5 Client

As the system for travelling between zones in the kingdom has been added, some additions were needed in the Client application. The client now draws the TravelTriggers as light blue squares, of the size and position as shown when creating them in the Editor. When a user right-click on the trigger with their mouse, a popup window will be displayed. In the popup window the name of the zone is displayed, and the user is asked if they want to travel there or not. A screenshot of the Travel Panel is shown in Figure 19.6. If the user answers yes, the Client will send a message with the information to the World server, as shown in the Sequence diagram in Figure 19.5. The class diagram of the new panel is shown in Figure 19.7.


Do you want to travel to:

Forest of THOR?

Figure 19.6: Screenshot of the Travel Panel in the game.



TravelPanel
-yesButton : WowButton -noButton : WowButton -travelQuestion : TextGraphicObject -destinationText : TextGraphicObject -hasAnswered : bool -answeredYes : bool
+hasAnswered() : bool +answeredYes() : bool

Figure 19.7: TravelPanel class diagram.

Like the other applications, Client now loads its network configurations from a file , Client.ini, on startup.

Chapter 20

World Editor

This chapter will give a description of the overall implementation of the Editor and class diagrams with a more detailed description of its functions.

20.1 Overview

In Figure 20.1 a overview of the implementation is shown. The editor is divided into three parts (models, graphics and main with storagehandler). From design there are not many changes to the overview, the graphics parts have been put into its own package, and the same for models. The changes and functions will be described more in detail in the following sections.

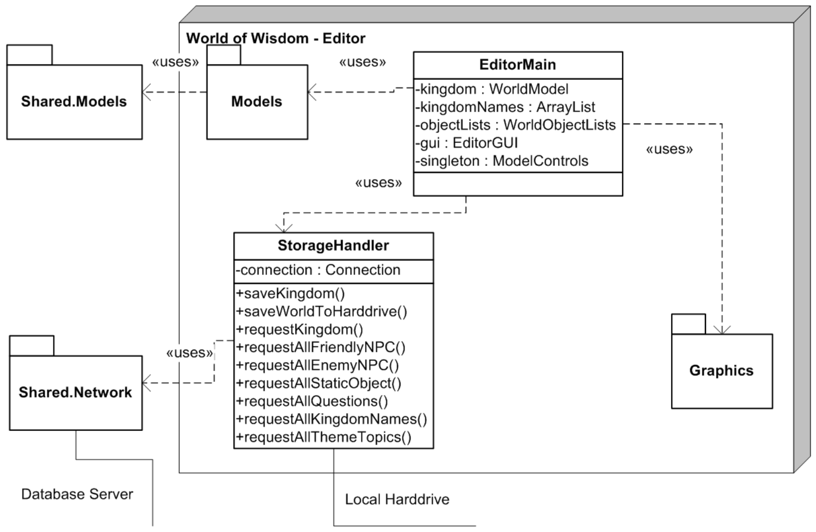


Figure 20.1: Class diagram of the World Editor implementation

20.1.1 EditorMain

EditorMain is the main class in the Editor application. Upon start-up of the editor, EditorMain creates instances of the models (WorldModel and WorldObjectLists), the graphics (EditorGUI) and the storage (StorageHandler). It also creates connections between the models and graphics using interfaces as listeners.

20.1.2 StorageHandler

The StorageHandler handles the network connection between the Database server and the Editor. The StorageHandler is using the singleton pattern and is used for sending objects to the Database. It requests objects and objectlists from the Database, and when received, forwards it to the EditorMain which updates the corresponding models.

When the StorageHandler is saving objects to the Database or waiting for reply, it sends status messages to the EditorGUI which displays the messages to the user in a popup window. While the StorageHandler is handling requests or waiting for reply, the GUI is locked.

20.2 Models

The Models package contains all the objects that are stored within the Editor. In Figure 20.2 the class diagram of the Models package is shown.

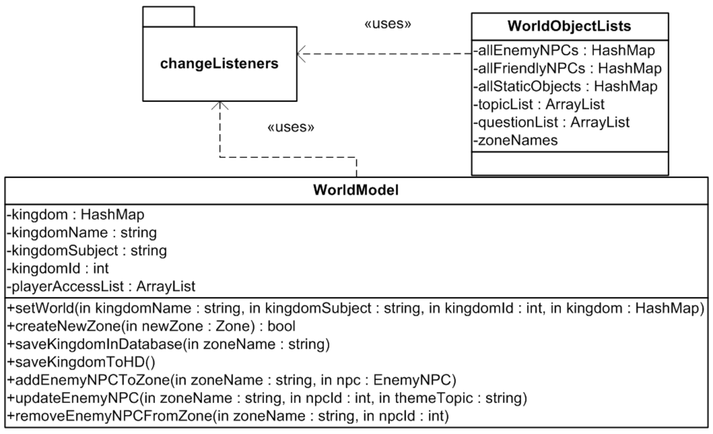

changeListeners
«uses»
WorldObjectLists
-allEnemyNPCs : HashMap
-allFriendlyNPCs : HashMap
-allStaticObjects : HashMap
-topicList : ArrayList
-questionList : ArrayList
-zoneNames
«uses»
WorldModel
-kingdom : HashMap
-kingdomName : string
-kingdomSubject : string
-kingdomId : int
-playerAccessList : ArrayList
+setWorld(in kingdomName : string, in kingdomSubject : string, in kingdomId : int, in kingdom : HashMap)
+createNewZone(in newZone : Zone) : bool
+saveKingdomInDatabase(in zoneName : string)
+saveKingdomToHD()
+addEnemyNPCToZone(in zoneName : string, in npc : EnemyNPC)
+updateEnemyNPC(in zoneName : string, in npcId : int, in themeTopic : string)
+removeEnemyNPCFromZone(in zoneName : string, in npcId : int)

Figure 20.2: Class diagram of the Models package

20.2.1 WorldObjectLists

The WorldObjectsLists contains the lists of all Enemy NPCs, Friendly NPCs, Static Objects, questions and theme topics in the Database. These are used by all the parts of the Editor. In design these lists were in the EditorMain class, but was moved into a separate class during implementation. This was done to keep the models seperate from the main class. The WorldObjectLists allows for adding observers to the lists, so when any changes occures to the lists, the observers will be notified with the updated information.

20.2.2 WorldModel

WorldModel is the underlying model for the MapEditor. It contains all the zones in the kingdom and the information on the kingdom (name and class subject). The MapEditor is added as an observer of the WorldModel when the editor starts. So any changes to the WorldModel are shown in the MapEditor.

The main methods in the WorldModel is SetWorld, which is used when a kingdom is retrieved from the database. CreateNewZone which is used when the user wants to create a new zone. SaveKingdomInDatabase sends the kingdom information to the StorageHandler which then packages the information in a serializable object and sends it to the database. WorldModel also has methods for updating information on objects in the kingdom, examples of these methods are shown in Figure 20.2 (add, update and remove objects).

20.2.3 ChangeListeners

The changeListeners package contains all the interfaces used by the WorldModel and WorldObjectLists. The different parts of the Editor implements the interfaces needed and are added as listeners to the Models.

20.3 Graphics

The Graphics package contains all the classes related to the user interface. From design the PanelController class has been removed, and the contents has been moved to the main graphics class, EditorGUI, since java has a built in controller for handling tabbed panels (JTabbedPane). The main class diagram for the Graphics package is shown in Figure 20.3.

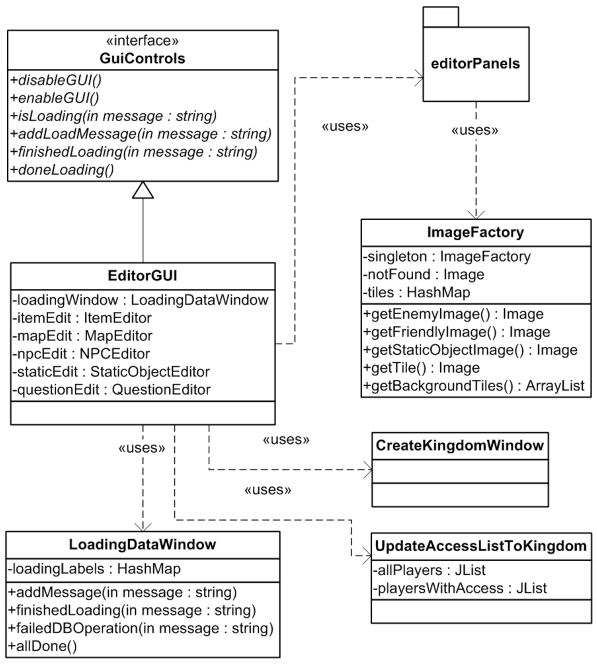

«interface»
GuiControls
+disableGUI()
+enableGUI()
+isLoading(in message : string)
+addLoadMessage(in message : string)
+finishedLoading(in message : string)
+doneLoading()
editorPanels
«uses»
«uses»
ImageFactory
-singleton : ImageFactory
-notFound : Image
-tiles : HashMap
+getEnemyImage() : Image
+getFriendlyImage() : Image
+getStaticObjectImage() : Image
+getTile() : Image
+getBackgroundTiles() : ArrayList
EditorGUI
-loadingWindow : LoadingDataWindow
-itemEdit : ItemEditor
-mapEdit : MapEditor
-npcEdit : NPCEditor
-staticEdit : StaticObjectEditor
-questionEdit : QuestionEditor
«uses»
«uses»
«uses»
CreateKingdomWindow
LoadingDataWindow
-loadingLabels : HashMap
+addMessage(in message : string)
+finishedLoading(in message : string)
+failedDBOperation(in message : string)
+allDone()
UpdateAccessListToKingdom
-allPlayers : JList
-playersWithAccess : JList

Figure 20.3: Class diagram of the Graphics package

20.3.1 EditorGUI

EditorGUI is the main part of the user interface. It contains a JTabbedPane which has a tab for each part of the Editor. It implements the GuiControls interface which is used to enable/disable the main window when a new window is in focus. It is also used to display status messages from the StorageHandler using the LoadingDataWindow.

20.3.2 ImageFactory

ImageFactory is using the singleton pattern, and is used to load images from the harddrive. ImageFactory is mostly used by the MapEditor to display the graphics of the kingdom. When the editor tries to load an image that does not exist, ImageFactory returns a default image. ImageFactory also contains a list of all the background images, since these are access more often then others.

20.3.3 GuiControls

GuiControls is an interface that is inherited by the EditorGUI. It has methods for disabling and enabling the GUI, and methods used for displaying the status when loading.

20.3.4 CreateKingdomWindow

CreateKingdomWindow is a simple window for creating a new kingdom, where the user can input the name of their new kingdom and the name of the class subject.

20.3.5 LoadingDataWindow

LoadingDataWindow is used by EditorGUI when the Editor is loading data. It shows a new window that displays a list of data being loaded/saved, and when its finished loading, it displays a "*Finished*" message behind the item that is done loading. While the Editor is loading, all user input is disabled. When all the data has been loaded, a button will appear that returns the user to the editor.

If an item fails to load, due to errors in the database, the loading will be aborted and displays an error message. The user is then allowed to return to the editor.

20.3.6 UpdateAccessListToKingdom

UpdateAccessListToKingdom is used to give access for players to the currently loaded kingdom. Only the players in the access list will be able to log into a world using the Kingdom and create playable characters there.

UpdateAccessListToKingdom contains two JLists, one with all the players without access and one with all the players that do have access. There are also two buttons for moving the players from one list to the other.

20.4 Editorpanels

The Editorpanels package contains the different panels for the Editor. These panels listens for changes in the underlying models using the ChangeListener interfaces in the models package. Figure 20.4 shows the class diagram of the Editorpanels package.

20.4.1 WorldObjectEditor

This was known as the NPCEditor and StaticObjectEditor. They were merged in the implementation since StaticObject only have two fields, Name and ImagePath, which is already included in the NPCEditor. The WorldObjectEditor is used to create new, alter and delete Enemy, Friendly NPCs and StaticObjects. The WorldObjectEditor uses

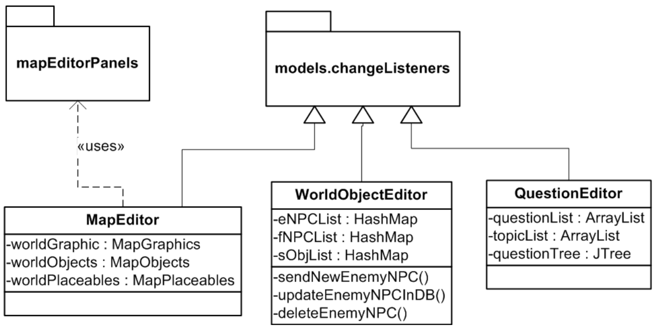
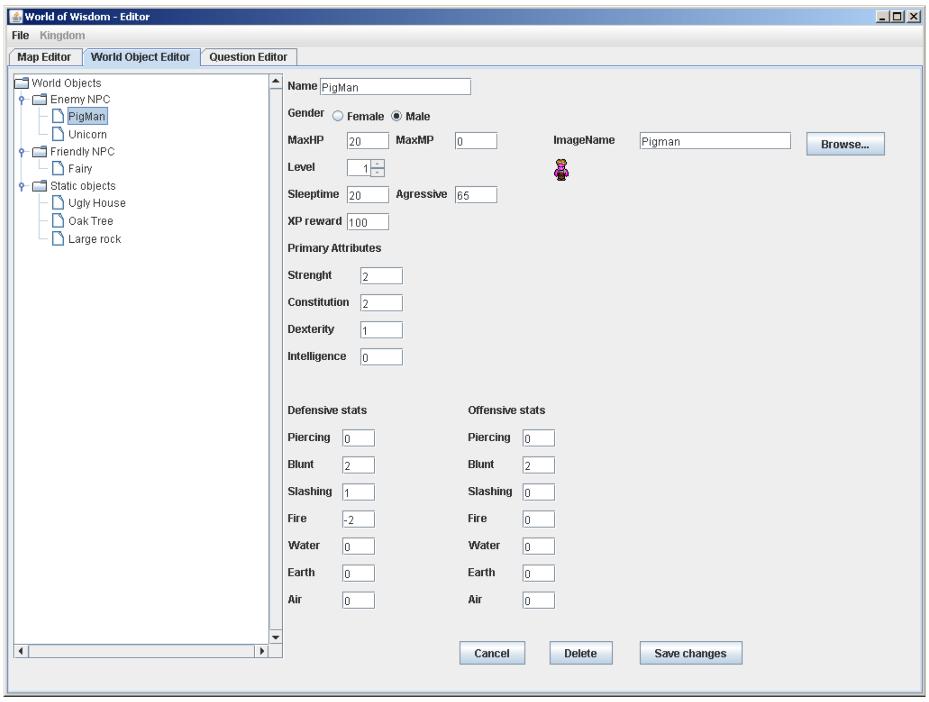


Figure 20.4: Class diagram of the Editorpanels

a Tree structure to organize the WorldObjects, and all the variables of a WorldObject can be altered using Textfields as input. As shown in Figure 20.5.

The WorldObjectEditor uses the EnemyChangedListener, FriendlyChangedListener and StaticObjectChangedListener to listen for changes in the WorldObjectLists.



Name PigMan

Gender Female [✓ Male]

MaxHP 20 MaxMP 0 ImageName Pigman

Level 1

Sleeptime 20 Agressive 65

XP reward 100

Primary Attributes

Strenght	2
Constitution	2
Dexterity	1
Intelligence	0

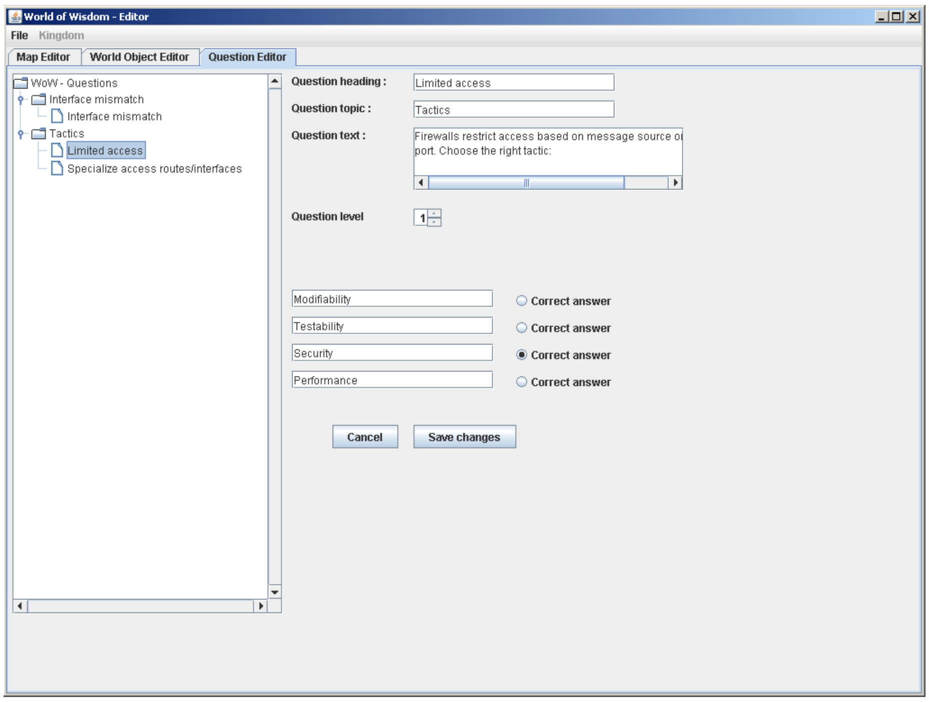
Defensive stats		Offensive stats	
Piercing	0	Piercing	0
Blunt	2	Blunt	2
Slashing	1	Slashing	0
Fire	-2	Fire	0
Water	0	Water	0
Earth	0	Earth	0
Air	0	Air	0

Figure 20.5: Screenshot of the WorldObject Editor panel

20.4.2 QuestionEditor

The QuestionEditor was created for the World of Wisdom prototype. Not many changes has been made to make it fit in the Editor. The original QuestionEditor directly used methods from the database server, so this had to be changed to use the network instead. The graphical layout has also been slightly altered to look like the rest of the Editor.

The ability to save changes to an already existing question and delete questions has also been added in the new version, as these were not implemented in the original. A screenshot of the Question Editor is shown in Figure 20.6.



Question heading : Limited access

Question topic : Tactics

Question text : Firewalls restrict access based on message source oi
port. Choose the right tactic:

Question level 1

Modifiability ○ Correct answer

Testability ○ Correct answer

Security ● Correct answer

Performance ○ Correct answer

Figure 20.6: Screenshot of the Question Editor panel

20.4.3 MapEditor

MapEditor is the main part of the Editor, and is used to create or alter a kingdom. The MapEditor is further divided into three panels (MapGraphics, MapObjects and MapPlaceables) which will be described in detail in the next section.

The MapEditor uses the WorldChangedListener to listens for changes in the WorldModel and updates the graphics accordingly.

20.5 MapEditorpanels

The MapEditorpanels package contains the three panels that is used by the MapEditor as shown in Figure 20.8.

20.5.1 MapObjects

MapObjects is the leftmost section in the MapEditor as shown in Figure 20.7. It contains a tree structure that shows all the zones and the objects within the zones in the currently loaded kingdom (the WorldModel). Buttons to add new zones or travel triggers, and a panel for changing the properties of items selected in the tree structure.

20.5.2 MapGraphics

MapGraphics is the middle section shown in Figure 20.7. MapGraphics displays the currently selected zone in the tree structure of MapObjects. MapGraphics also handles input from the mouse to drag the objects in the zone into the desired positions. The mouse is also used to paint the background graphics to the zone.

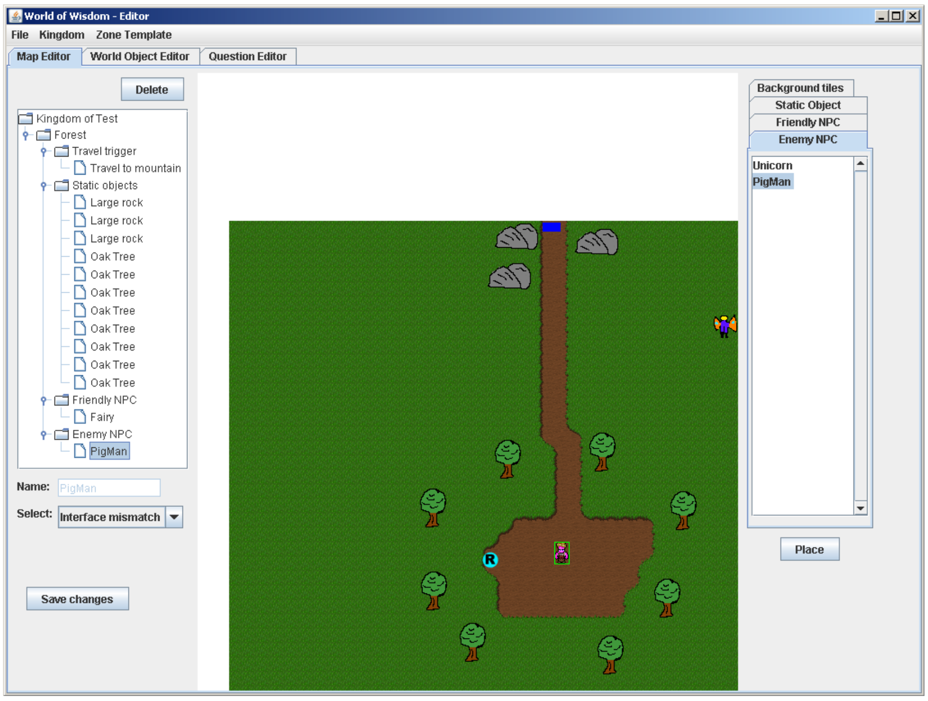


Figure 20.7: Screenshot of the Map Editor panel

20.5.3 MapPlaceables

MapPlaceables is the rightmost section shown in figure 20.7. MapPlaceables contains lists of all the objects that can be placed in the kingdom. There is a separate list for each of the types of objects that can be placed. After selecting an object from a list, the user simply clicks the place button and the object will show up in the MapGraphics and MapObjects panels.

20.5.4 CreateWindows

When creating new zones or travel triggers from the MapObjects panel, a new window will be opened. The CreateZoneWindow and CreateTriggerWindow, which accepts input from the user. In the CreateZoneWindow the user can specify name and size of the zone. In the CreateTriggerWindow the user can specify name and size of the traveltrigger.

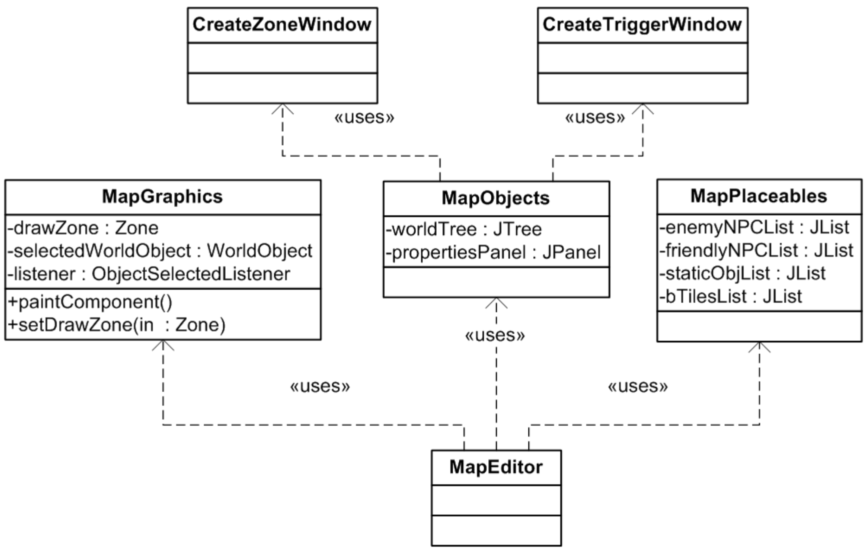


Figure 20.8: Class diagram of the Panels for the Mapeditor

Part V

Results and Evaluation

Chapter 21

Introduction

In this part we will look at the results of the project. First in Chapter 22 we will look at the Requirements that were stated in the Design part, and how they were implemented. In Chapter 23 we will go through the research questions and how they were answered. In Chapter 24 we will see how the user tests were performed and the results of the user testing.

Chapter 22

Requirements

In this chapter we will go through all the requirements stated in the design Chapter 11 and see which requirements were implemented and how.

22.1 Requirement results

In this section we goes through each requirements, and stating if they were implemented. "Yes", for fully implemented. "Partially", for some parts are implemented or done differently. "No", for requirements that were not implemented at all.

22.1.1 Database and Storage interface

FR1 It should be possible to add new or delete NPCs (both friendly and enemy NPC) in the Database.
Priority - High *Difficulty* - Low
Implemented - Yes. It is now possible to create new NPCs or delete them using the World Object Editor. It also supports adding new images.

FR2 It should be possible to alter information on NPCs. (e.g. Name, Question Topic, Attributes, Respawn time, Dialog etc.).
Priority - Medium *Difficulty* - Low
Implemented - Yes. The World Object Editor also allows viewing and editing of existing NPCs.

FR3 The attributes should scale after the level selected for the NPCs (e.g. Primary attributes, offensive and defensive attributes). *Priority* - Medium *Difficulty* - Low
Implemented - Yes. When changing the level of an NPC, the attributes are automatically changed to scale with the level. This is an important feature for making it faster to create NPC as there are 18 different attributes for each NPC. This will need some balancing though, as they are for the moment just adjusted 1 additional point per level.

FR4 Be able to add new or delete Static Objects in the Database.
Priority - High *Difficulty* - Low
Implemented - Yes. It is now possible to create new Static Objects or delete them

using the World Object Editor. It also supports adding new images.

FR5 Be able to alter existing Static Objects in the Database.
Priority - High *Difficulty* - Medium
Implemented - Yes. As with the NPCs, Static Objects can be altered in the World Object Editor and saved to the Database.

FR6 Be able to create quests and connect them to Friendly NPC.
Priority - Low *Difficulty* - High
Implemented - No. Quests are an important part of an RPG, but as it was not implemented in the prototype, it would take to much time to design a good system and implement. So most of the focus was rather on the creation of the objects in the kingdom and the zones.

FR7 Be able to store the world with object references in the Database.
Priority - High *Difficulty* - High
Implemented - Yes. It is now possible to store the entire kingdom in the Database, this is done by pressing the "Save world" button on the menu in the Editor. This was one of the more important features, because if it could not be saved, it could neither be used by the World Server.

FR8 Be able to load a world from the database.
Priority - High *Difficulty* - High
Implemented - Yes. It is now possible to retrieve a kingdom by name from the Database. When starting the Editor, a list of all the kingdoms available in the Database is retrieved, and the user can then select which kingdom to load into the Editor.

FR9 Be able to store and load the world locally.
Priority - Medium *Difficulty* - Medium
Implemented - Yes. It is now possible to store and load the kingdom from the hard drive. This is done manually, by using the menu, similar to when you are saving the world to the Database. This is useful incase the connection to the Database is lost, so that the user do not loose all the work they have done on the kingdom since last saving.

22.1.2 Map editing

FR10 Be able to create maps(zones) by specifying map size in width and height.
Priority - High *Difficulty* - Medium
Implemented - Yes. When creating a zone, a new window will be shown where the user may type in the size and name of the new zone. This is one of the core functions needed for creating a new kingdom.

FR11 Be able to create Zone Triggers and connection between them.
Priority - High *Difficulty* - Medium
Implemented - Yes. When creating a new Travel Trigger, a new window will be shown where the user may type in the size and name of the new travel trigger. Then afterwards the user may select the travel trigger and choose its destination. This is one of the core functions when creating a kingdom, as without it, the players cannot travel between zones.

FR12 Drawing background using a tileset. Selecting a individual background tilepicture and placing it in the background.
Priority - High *Difficulty* - Medium
Implemented - Yes. The user may select a tile from a list in the Editor, and click on the background to replace the tile with the one they have selected. This is important for creating unique zones.

FR13 Drawing background using type (sand, grass, mud etc.). Automatically calculating which tile to use to create overlay between types.
Priority - Low *Difficulty* - High
Implemented - Yes. When the user draws a tile on the background, the surrounding 8 tiles are checked and adjusted accordingly to make it a smooth transition between background types. This greatly improves the speed of drawing the background, and thus the speed of creating a zone.

FR14 Be able to place world objects (static objects, friendly NPCs, enemy NPCs) on the map.
Priority - High *Difficulty* - Medium
Implemented - Yes. This was one of the more important features. The user can add a new world object to the map by selecting the desired object in a list and pressing the "Place" button.

FR15 When adding a enemy NPC to the world, the user select a theme topic(Question topic), creature type and level. *Priority* - Medium *Difficulty* - Medium
Implemented - Partially. When adding a new NPC you first select the creature type, and when placing the NPC a new window is shown where the user can select a theme topic. We chose not to select the creature level when placing, but rather when creating the NPC with the World Object Editor.

FR16 Be able to change position of world objects.
Priority - Medium *Difficulty* - Medium
Implemented - Yes. Position can be changed by first selecting the object, and then dragging the object to the new position.

FR17 Use drag and drop to place new creatures or objects.
Priority - Medium *Difficulty* - Medium
Implemented - No. When adding a new creature to a zone, they are placed at the

point of focus. The user may then alter its position afterwards by using drag and drop.

FR18 Use drag and drop to change the position of creatures or objects.
Priority - Low *Difficulty* - Medium
Implemented - Yes. When a creature or object has been selected, they can be dragged to their new position.

FR19 Create quest relations between friendly and enemy NPC.
Priority - Low *Difficulty* - High
Implemented - No. Quest has not been implemented.

FR20 Be able to create templates and load templates of zones from database. Using templates will greatly reduce the time needed to make a world ready for players.
Priority - Medium *Difficulty* - Medium
Implemented - Yes. It is possible to store the zones that are created using the Editor as templates in the Database. Then other users can load the zone into their own kingdoms. A template can be used in many kingdoms, and using them reduces the amount of work for creating the world. The user still have to change the theme topic of the NPCs to their desired course subjects.

22.1.3 Prototype improvements

Database Server

FR21 Storing friendly NPC position and zone in the database.
Priority - High *Difficulty* - Medium
Implemented - Yes. Storing position information is an important part of saving the kingdom in the database.

FR22 Storing Zone Trigger information in the database (Position, size and target zone).
Priority - High *Difficulty* - Medium
Implemented - Yes.

FR23 Storing static object in the database.
Priority - High *Difficulty* - Medium
Implemented - Yes. It is now possible to store Static Objects in the Database with the imagepath.

FR24 Storing static objects NPC position and zone in the database.
Priority - High *Difficulty* - Medium
Implemented - Yes.

World Server

FR25 When starting the World Server application, it will connect to the Database Server and retrieve a list over all kingdoms.
Priority - Medium *Difficulty* - Medium
Implemented - Yes. When the World Server is started, the user gets a list of all the Kingdoms in the Database.

FR26 When a player moves ontop of a zone trigger, the server will send a message to the player, asking if he will move to another zone.
Priority - Medium *Difficulty* - Medium
Implemented - No. Instead of the World Server checking constantly if players are moving ontop of travel triggers. The Client may click on a travel trigger when they are close enough. Then the Client sends information to the World Server about the travel trigger they have selected. This method of using the travel triggers reduces the load on the World Server. As it does not need to do checks every second. Instead they only need to move the Client to the destination zone when it gets the message.

FR27 Be able to load a kingdom from the Database.
Priority - High *Difficulty* - High
Implemented - Yes. This was one of the more important features for the World Server, so that the kingdoms created in the Editor could be tested.

Client

FR28 A popup window will be displayed when a player moves ontop of a zone trigger. A message will be sent to the world server if he answers yes. And he will be moved to the new zone.
Priority - High *Difficulty* - Medium
Implemented - Partially. The popup window does not appear when the player moves ontop of the travel trigger, but when the player right-click the trigger. This was done to avoid the popup window from appearing when it was not needed.

22.2 Summary

During the implementation of the World Editor and the improvements to the World of Wisdom prototype, most of the requirements were implemented. A table with the requirements and their implementation status is shown in Table 22.1. The requirements that were not implemented, were either implemented in a different way, as with the Travel Trigger handling (FR26), or they were concerning the implementation of Quests in the prototype and Editor. Quests were simply not implemented because of time constraints. Designing and implementing a good and flexible Quest system would have taken too much focus away from the implementation of the other features of the World Editor.

During the implementation we also found that there were some additional classes and alterations needed. The most important one of those were the Settings class. All the network settings in the prototype and the World Editor was hardcoded, and this made it difficult to try out the applications on other computers. The Settings class, which is described in the implementation Chapter 19, now reads the network configurations (Ip addresses and port numbers) from the hard drive. This make it very easy to move the applications and servers to other computers.

In addition to the network configurations, the username and password needed for the Database Server to connect to the PostgreSQL database was hardcoded. This has been changed to ask the user for the username and password with dialog windows when the Database Server is started.

Some minor problems was discovered aswell during implementation that was fixed. Like for example, the lists containing all the creatures in a zone, used the name of the creature as identifier. This made it impossible to have two or more of one creature type in one zone, which was not intended in the first implementation. This was simply fixed by using the id that is generated by the database as identifier instead of the name.

Table 22.1: Implementation status on the requirements

Requirement	Implemented
FR1 It should be possible to add new or delete NPCs in the Database.	Yes
FR2 It should be possible to alter information on NPCs.	Yes
FR3 The attributes should scale after the level selected for the NPCs.	Yes
FR4 Be able to add new or delete Static Objects in the Database.	Yes
FR5 Be able to alter existing Static Objects in the Database.	Yes
FR6 Be able to create quests and connect them to Friendly NPC.	No
FR7 Be able to store the world with object references in the Database.	Yes
FR8 Be able to load a world from the database.	Yes
FR9 Be able to store and load the world locally.	Yes
FR10 Be able to create maps(zones).	Yes
FR11 Be able to create Zone Triggers and connection between them.	Yes
FR12 Drawing background using a tileset.	Yes
FR13 Drawing background using type (sand, grass, mud etc.).	Yes
FR14 Be able to place world objects on the map.	Yes
FR15 When adding a enemy NPC to the world, the user select a theme topic, creature type and level.	Partially
FR16 Be able to change position of world objects.	Yes
FR17 Use drag and drop to place new creatures or objects.	No
FR18 Use drag and drop to change the position of creatures or objects.	Yes
FR19 Create quest relations between friendly and enemy NPC.	No
FR20 Be able to create templates and load templates of zones from database.	Yes
FR21 Storing friendly NPC position and zone in the database.	Yes
FR22 Storing Zone Trigger information in the database.	Yes
FR23 Storing static object in the database.	Yes
FR24 Storing static objects NPC position and zone in the database.	Yes
FR25 When starting the World Server application, it will retrieve a list over all kingdoms.	Yes
FR26 When a player moves ontop of a zone trigger, the server will send a message to the player.	No
FR27 Be able to load a kingdom from the Database.	Yes
FR28 A popup window will be displayed when a player moves ontop of a zone trigger.	Partially

Chapter 23

Research Questions

In this chapter we will look at the Research Question stated in the Research Chapter 3 and summarize how they have been answered throughout this project.

What types of tools are needed for creating a kingdom for World of Wisdom?

For creating a kingdom for World of Wisdom, we created a adminstrative tool called the World Editor. This tool is divided into three parts, Map Editor, World Object Editor and Question Editor. The Question Editor was created with the World of Wisdom Prototype. The Map Editor is the tool for creating the kingdom, and its zones, and for populating the kingdom with Non-Player Character (NPC) and other objects. Throughout this project the World Editor was designed (Part III) and implemented (Part IV).

What improvements are needed in World of Wisdom prototype to make a complete connected world?

The improvements needed in the World of Wisdom prototype where discussed in Chapter 10. The most important improvement needed was the Zone travel. As without the ability to travel between maps, it would not be much of a world to travel through and explore.

In the prototype the world was also hardcoded, meaning that tables for storing the kingdom was missing from the database, aswell as the World Server did not have any method of retrieving the kingdom from the database.

In design Chapter 14 we created the entity relation models for the tables that were missing. In Chapter 15 the improvements were designed for the World of Wisdom prototype and the implementation is described in Chapter 19.

What similar game editor tools exists, and what can be learned from them?

In the Prestudy Part II we looked at both old and new game editors and some of the state of the art educational game concepts, and how they edited their game worlds.

In several of the game editors there were one part with an overview of what had been placed in the game world and one part that shows how the world with ingame graphics.

For the World of Wisdom World Editor we used the component setup as in the Aurora toolset which is described in Section 6.2. The Map Editor is divided into three parts. One that shows a tree structure with all the objects that has been placed in the world, organized by maps and categories. One that shows how the world will look ingame, and one part that display list of all the objects that can be placed within the maps.

We used this structure as the Aurora toolset is based on a Role-playing game (RPG) with a similar game structure as World of Wisdom. Though there are many game editors also use a similar structure, as with the Age of Computers (AoC) Admin tool described in Chapter 8.

How to make it easy to create and maintain a kingdom and what can be done to reduce the time it takes to make a world?

Here we focused on features that makes the World Editor easy to use as drag and drop, and automatically adjusted variables to reduce the work. When painting the background in the kingdom, the background is automatically painted around the tile that was placed, to make a smooth transition to the other background tiles. This greatly reduces the time it takes to paint the background, as placing every single tile type is quite timeconsuming.

We also created the possibility to save and load single zones from the database as templates. This means that anyone can create a zone and then save it to the database as a template, for example a finished city, with all buildings and Non-Player Character (NPC)s placed. Then anyone can load this city into their own kingdom, and save alot of time designing and creating their own city. They can ofcourse do their own adjustment to the zone, and later save it in their kingdom. This creates the possibilites for someone to design entire kingdoms that can be used by others that do not have the time to create their own kingdoms, and only want to replace the knowledge materials used with their own.

How can a teacher/assistant improve the experience of the students while the game is running?

For this we have looked at how MMORPG usually interact with their players when the game is running in Chapter 7. Game masters , as they are called, are people that log on to the game world using special clients that often can do temporary alteration to the game world. In Section 10.3 we look at how the Game Masters can improve the interaction between students and teachers. In World of Wisdom this can be used to interact with the students, to get reports if there are any bugs, or problems. An important usage is to find out where the students is struggling, and if they are having problems because the gameplay is too difficult or the questions and tasks are. It can also be used to find out what parts of the course are the most difficult, and if it is because the questions are too ambiguous this can be fixed using the WoW World Editor to alter the question. If it is because the course material is particular hard to understand, the teachers can set up additional lectures on the topic.

Chapter 24

User tests

When testing the WoW World Editor textual use cases were used. The format used for the use cases are from the book UML Distilled[Fow03]. An example of one of the textual use cases are shown in Figure 24.1. This use case shows how to create a new kingdom step by step. The rest of the use cases are in Appendix D. In addition to testing with use cases, some of the tests were done without much guidance to see how they would use the application.

There were only a few user tests which were short and mostly focusing on the functions of the WoW World Editor and not on creating an entire kingdom as this is timeconsuming work. But during the tests we did uncover some bugs, aswell as some design issues which will be further discussed in this chapter.

Table 24.1: Use case: Create new kingdom

Use case : 2.1 Create new kingdom.
Level : Map Editor
Main Success Scenario: 1: User selects "Map Editor" tab. 2: User selects "Kingdom" on menu. 3: User selects "Create new Kingdom". 4: System displays new window. 5: User enters name for new kingdom and class subject name. 6: User selects "Create". 7: System shows the name of the Kingdom in the tree structure.

24.1 Bugs

In this section we will look at some of the bugs that were discovered, and how they can be fixed.

24.1.1 Deleting World objects types

When deleting a World Object (NPC or Static Objects) in the World Object Editor, these are automatically removed from the kingdoms where they were used in the database. This is done automatically at the Database Server, but if the user has a kingdom open which also has the object, the Map Editor will give an error as it has not been removed from the currently loaded kingdom, only from the database.

This can be fixed by checking if objects of the type exists in the currently loaded kingdom and deleting them there aswell. But there are also other issues to consider, as deleting an object in the World Object Editor, you might delete an object that someone else were using. Alternatively the user could be told when he is deleting, that the object exists in a kingdom and can not be deleted. Requiring the user to delete the object manually first from the kingdoms where it has been used.

24.1.2 Object collision in map

Currently in the Map Editor, new object can be placed anywhere in the zone, including ontop of each other. This is not much of an issue with the Static Objects, as it may only look a bit strange but not actually cause any problems. But with the NPC this can be a problem as they can not move if they are inside another object. This could be fixed by checking the placement for other objects when they are being placed.

24.1.3 Category folders closes

When a World Object or Question has been altered in the database, a new list of the objects or questions are retrieved. They will then refresh the list in the different parts of the WoW World Editor that uses them. This causes the folder of the category that has been updated to close. This can be fixed by forcing the folders to be open after the lists has been updated.

24.2 Design issues

In this section we will look at some issues with graphical design and user-friendliness.

24.2.1 Deleting objects from kingdom

When you want to delete an World Object from the Kingdom, you select the object in the tree structure, and either press the Delete key, or the delete button above the tree structure. But if you select the object in the graphical representation of the zone and press the Delete key, nothing happens. This is because the tree structure needs to be in focus to be able to delete using the key. This could be fixed by adding an additional keylistener to the MapGraphics panel.

24.2.2 Load window

Whenever the WoW World Editor needs to either save something to the database or retrieve some data from it, a window pops up showing the progress. When the loading is done, an "Ok" button appears, that needs to be pressed to continue.

As most of the times when loading to the database only takes a few seconds, it is interrupting that you have to manually close the window before you continue. A solution to this is to automatically close the window when the loading is finished, except when an error has occured as the user would need some time to read it.

24.2.3 Background tiles

A list in the Map Editor shows all the background tiles that can be painted in the zones. As the tiles around the painted tile is automatically changed to make a smooth transition to other background types, it is no longer necessary to have all the different tiles in the list. But to only have the main types (mud and grass) in the list. It was also pointed out that the tiles were a bit too small in the list.

This has been fixed in the latest version of the WoW World Editor. The list now only shows the two types, and they have been resized. In Figure 24.1 the change is shown with a before and after screenshot.

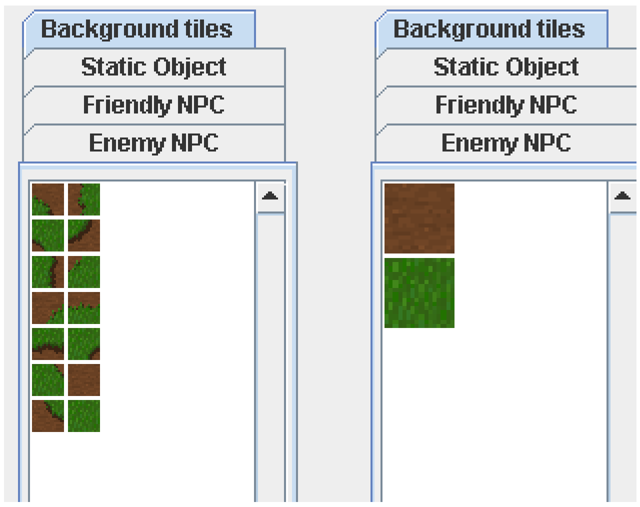


Figure 24.1: Screenshot of the change in the tile list in Map Editor. Before on the left and after on the right.

24.2.4 Create new World Object

One thing that was discovered when the user wanted to create a new World Object was that they instinctively clicked the "Create new Object" button before typing in the information about the object. This is because they expected a new window to open where they would type in the information. The World Object Editor then stored an empty object in the database.

This has been fixed by changing the name of the button to "Save to Database" to make it more clear. Also if the button is pressed before typing in a name for the object, the user will get a message asking them to enter a name before saving. This should prevent similar cases in the future.

24.2.5 Placing creatures and objects

To place creatures or objects in a zone in the Map Editor, the user first selects the object in one of the object lists, and press the "Place" button below the list. This will place the object in the zone at the middle of the screen.

When the testers wanted to place objects, and without using the use case which states how, they first selected the object and then mouse clicked at the desired position. This is similar to how the painting of the background tiles work, but was not done for the object lists because the objects can be moved when selected in the graphics panel.

This can be fixed by making a toggle button that changes between selection and placing of objects. This is common in several applications to have a toolbar where you can change between the actions the mouse can do (selection, movement, painting, etc.).

Another option is to make it so when the user selects the object in the list, the Map Editor changes to enable placing the object, and when the object is placed, it goes back to selection mode. Only problem with this approach is that then a action for aborting the placement of the object must be added aswell.

Part VI

Conclusion

Chapter 25

Conclusion

The work during this project has resulted in the implementation of the WoW World Editor. With this tool you can add more content to the WoW prototype, in form of new maps, creatures, objects and questions. The World Editor simplifies this by offering features to automate parts of the creation of new content, and by making it possible to use templates of maps to reduce the time needed to make a kingdom. The WoW World Editor greatly improves the efficiency of generate kingdoms for the prototype, as well as making it easy for anyone to create new content. This allows the teachers to focus on creating the knowledge for the game.

The WoW prototype has been improved to allow travel between maps in the kingdom. This was considered the most important missing part when it comes to creating a kingdom using the WoW World Editor. During the implementation we also found that it was too difficult to move the World of Wisdom project to other computers, due to the network settings being set in the code. This was changed to read all the network settings and database settings from files on start-up instead. So now it is easy to move the server applications to other computers. Further details about how to install the World of Wisdom on other computers is in the Installation Guide Appendix F. There are more that needs to implemented, and this is further discussed in the Future Work (Chapter 26). As the current implementation of WoW is a prototype, and not a finished game, the WoW World Editor will need further work as WoW is still a project in progress.

A common addition to Massively multiplayer online role-playing game (MMORPG)s are Game Masters. These are special clients used to log on to the game by the representatives for the game world. In the case of WoW this would be a great addition for the teachers and teacher assistants to interact with the students while they are playing the game. It can be used to log the activity in the kingdom to find out which students are struggling, or if there are parts of the kingdom that are more difficult than intended. This information can then help the balance of the game and tasks to provide a better experience for the students.

Chapter 26

Future Work

In this chapter we will look at what further improvement and work that can be done on the World Editor and work on the World of Wisdom prototype related to the Editor. Most of the implemented features of the World Editor concerns the creation of zones and populating them with NPCs and Static objects. But there are other parts of the World of Wisdom that could use administrative tools. There are also many improvements that can be done on the current implementation of the World Editor. In the following sections the improvements and additions we concider most important will be described.

26.1 World of Wisdom objects

In this section we go through the improvements and additions related to World of Wisdom objects.

26.1.1 Items

The items (armor, potions, weapons, etc.) was implemented in World of Wisdom, but a way of obtaining the items was not. Like merchants that buy or sells items, creatures that drops items when they are defeated or items that could be rewarded through finishing quests. When a system for this is implemented, the Editor will need to support these features. In creation of Friendly NPCs it should be possible to classify the NPC as a merchant, and add items from a list of all items, that the merchant will sell. Similar the Enemy NPC will have a list of items that they can drop when they are defeated. When creating quests, it should be possible to add one or more items as reward for completing it.

These are features that needs to be added to already existing parts in the Editor, but a Item Editor is also needed for creating and managing the items in the database.

26.1.2 Attacks

The attacks used by the players and the NPCs need an Editor for creating and managing them. Though everyone that creates a kingdom may not need to create any attacks, as they can use attacks that others have created. In addition to a separate Editor for attacks, the World Object Editor should be able to add attacks to the NPCs. Which will be the ones they use when they fight against the players.

26.1.3 Quests

Quests are a very important of any Role-playing game (RPG) as it is usually the main method of progressing through the game. As the World of Wisdom aims to be used for several types of courses and to be general, it would need a very flexible and easy to use quest system. With this implemented, the Editor would need to be able to generate these quests, and then connect the quest to a specific Friendly NPC. This could either be done in the World Object Editor, binding all Friendly NPC of one type to a quest, or by doing as with the theme topic and Enemy NPCs and add the connection when they are placed in the kingdom.

26.1.4 Question

Currently only multiple choice question type is implemented, and thus only multiple choice question can be created in the Question Editor. So when creating support for more types of questions, the Editor must also be updated to support creation of the new types of questions.

26.1.5 Background

In the current implementation the set of background tiles are hardcoded, they can be replaced by other images, but for adding more a separate Background Editor is needed. With an Editor for the background tiles it would need to register which tiles exist in the Database.

26.2 World Editor improvements

In this section we will go through some of the improvements that can be done to the current implementation of the World Editor.

26.2.1 Security

In the implementation of World Editor, there were not much focus on the security. Meaning that anyone that has access to the World Editor can do changes to all of the kingdoms in the database.

Authentication

Every user of the World Editor should have their own username and password that is required at the start-up of the Editor. This should ensure that only people with proper authority can do changes to the kingdoms.

Edit Rights

At the moment, as long as the user has access to the Editor, they can do changes to everything the Editor supports. So if someone has created a few objects that they want to use for their kingdom, others can change them later. This could cause problem for the original creator of the objects. So it should be possible to give ownership rights to the objects and worlds the user have created, so no one else can do changes to them. The user should have the option to make the objects they make public or private, and if they can be edited or if it will only be read rights for others.

26.2.2 Availability

The Database needs to be online to do any work in the World Object Editor and the Question Editor. If the connection is lost while the user is editing a question or world object, any changes done is lost when the connection is reestablished. This is because when the connection is up again, it will automatically get the latest version of the list of objects and questions.

This is not a problem when working on a kingdom, but to be able to place objects in the kingdom, the object list must have been retrieved once. So it is not possible to work on a kingdom offline without first retrieving all the objects and question topics that can be used.

26.2.3 Performance

Though performance may not be as important as availability, it should not cause the user to wait for drawing updates while working. Compared to the Client, the Editor does not update the graphics of the kingdom 30-60 times per second. It only updates the graphic if a change has been done or the focus position is moved. Something that can increase the performance of these updates would be to buffer the images that are loaded from the hard drive. Currently they are read from the hard drive each time the zone is drawn. Instead they could be put in a list after the first time they are loaded, and later be retrieved from this list instead. This is done for all the background tile when the ImageFactory is instanciated.

References

[Ale05] Alexander, Thor: *Massively Multiplayer Game Development 2*. Charles River Media, 2005.

[Bioa] Bioware: *Intro to the dm client*. `http://nwn.bioware.com/dms/dmclientintro1.html`, [Online; accessed 12-May-2009].

[Biob] Bioware: *Intro to the toolset - part 2*. `http://nwn.bioware.com/builders/toolsetintro2.html`, [Online; accessed 22-January-2009].

[Bioc] Bioware: *Neverwinter nights community site*. `http://nwn.bioware.com/`, [Online; accessed 22-January-2009].

[Biod] Bioware: *Neverwinter nights: For builders*. `http://nwn.bioware.com/builders/index.html`, [Online; accessed 22-January-2009].

[Bra01] Braude, Eric J.: *Software Engineering - An Object-Oriented Perspective*. John Wiley & Sons, Inc., first edition, 2001, ISBN 0-471-32208-3.

[Bre] Brewster, Rick: *Paint.net*. `http://www.getpaint.net/`, [Online; accessed 05-May-2009].

[Cry] CryTek: *Farcry*. `http://www.crytek.com/games/far-cry/overview/`, [Online; accessed 22-January-2009].

[DJK$^+$] Diveky, Marko, Peter Jurnecka, Rudolf Kajan, Ludolf Omelina, and Maria Bielikova: *Smart multipurpose interactive learning environment*. `http://imaginecup.fiit.stuba.sk/2007/index_en.html`.

[DJK$^+$07] Diveky, Marko, Peter Jurnecka, Rudolf Kajan, Ludolf Omelina, and Maria Bielikova: *Adaptive educational gameplay within smart multipurpose interactive learning environment*. Semantic Media Adaptation and Personalization, International Workshop on, 0:165–170, 2007.

[Fou] Foundation, The Eclipse: *Eclipse*. `http://www.eclipse.org/`, [Online; accessed 05-May-2009].

[Fow03] Fowler, Martin: *UML Distilled, 3rd Edition*. Addison Wesley, 2003.

[Joa] Joachim, Froholt: *Den kreative revolusjonen*. `http://www.spillverket.no/artikler/den_kreative_revolusjonen/66786/1`, [Online; accessed 12-May-2009].

[Mica] Microsoft: *Microsoft office visio 2007*. `http://office.microsoft.com/en-gb/visio/default.aspx`, [Online; accessed 05-May-2009].

[Micb] Microsystems, Sun: *Netbeans*. http://www.netbeans.org/, [Online; accessed 05-May-2009].

[Mol] Molecule, Media: *Littlebigplanet*. http://www.littlebigplanet.com/, [Online; accessed 12-May-2009].

[Nad] Nadeo: *Trackmania united*. http://www.trackmania.com/tm/index.php?rub=united, [Online; accessed 22-January-2009].

[Nat] Natvig, Lasse: *Age of computers resource page*. http://www.idi.ntnu.no/~lasse/AoCshare.php, [Online; accessed 18-May-2009].

[NL04] Natvig, Lasse and Steinar Line: *Age of computers - game-based teaching of computer fundamentals*. pages 101–111, 2004.

[Nov08] Novak, Jeannie: *Game Development Essentials*. Delmar Cengage Learning, second edition, 2008, ISBN 978-1418042080.

[Pos] PostgreSQL: *Postgresql download*. http://www.postgresql.org/download/, [Online; accessed 22-January-2009].

[Sof] Software, ID: *Wolfenstein*. http://www.idsoftware.com/games/wolfenstein/wolf3d/, [Online; accessed 22-January-2009].

[TeX] TeXnicCenter: *Texniccenter*. http://www.texniccenter.org/, [Online; accessed 05-May-2009].

[Til] TileMap.co.uk: *Mappy editor*. http://tilemap.co.uk/mappy.php, [Online; accessed 18-May-2009].

[ZW98] Zelkowitz, Marvin V. and Dolores R. Wallace: *Experimental models for validating technology*. IEEE Computer, 31(5):23–31, 1998.

Part VII

Appendix

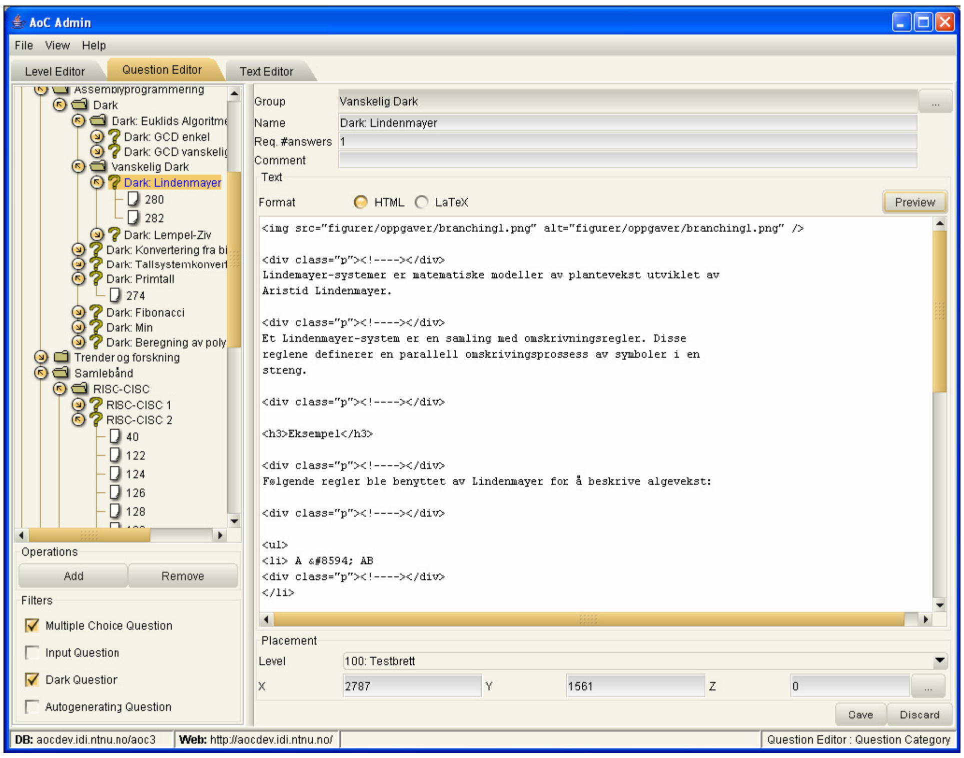
Appendix A

Acronym

ACS	Adventure Construction Set
ALU	Arithmetic Logic Unit
AoC	Age of Computers
DM	Dungeon Master
GM	Game Master
GUI	Graphical User Interface
HTML	HyperText Markup Language
IDE	Integrated Development Environment
MMORPG	Massively multiplayer online role-playing game
MVC	Model-View-Controller
NPC	Non-Player Character
NTNU	Norwegian University of Science and Technology
NWN	Neverwinter Nights
RPG	Role-playing game
S.M.I.L.E.	Smart Multipurpose Interactive Learning Environment
WoW	World of Wisdom
WYSIWYG	What You See Is What You Get

Appendix B

Age of Computer Screenshots



Group: Vanskelig Dark
Name: Dark: Lindenmayer
Req. #answers: 1
Comment:

Filters: [✓ Multiple Choice Question] Input Question [✓ Dark Question] Autogenerating Question

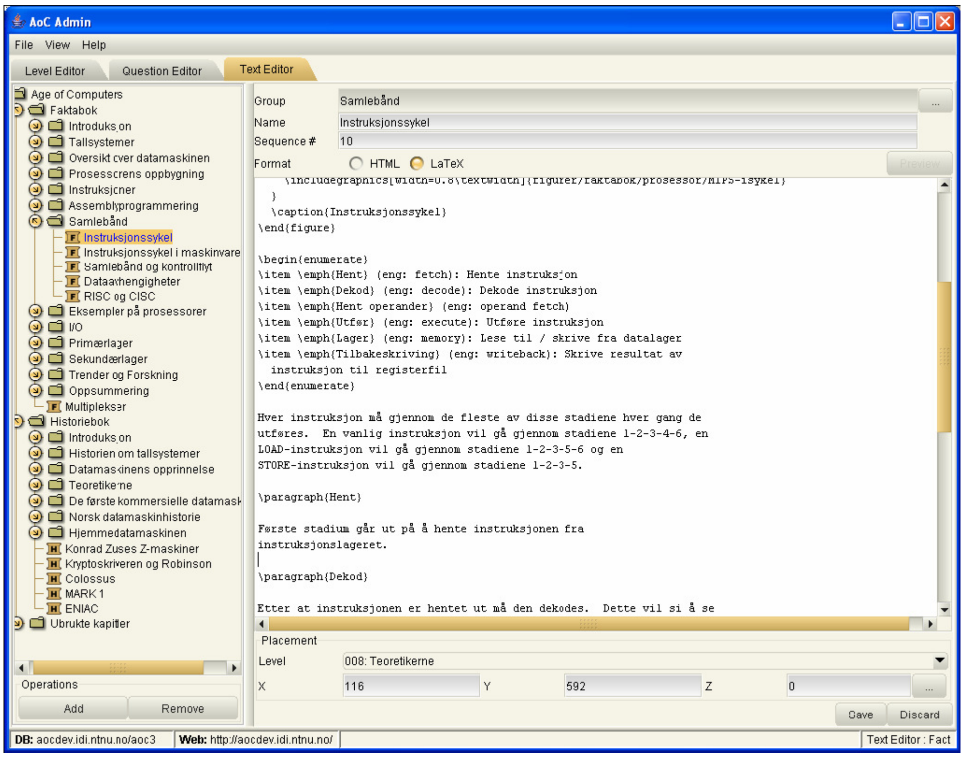
Text — Format: [✓ HTML] LaTeX

```
<img src="figurer/oppgaver/branching1.png" alt="figurer/oppgaver/branching1.png" />

<div class="p"><!----></div>
Lindemayer-systemer er matematiske modeller av plantevekst utviklet av
Aristid Lindenmayer.

<div class="p"><!----></div>
Et Lindenmayer-system er en samling med omskrivningsregler. Disse
reglene definerer en parallell omskrivingsprossess av symboler i en
streng.

<div class="p"><!----></div>

<h3>Eksempel</h3>

<div class="p"><!----></div>
Følgende regler ble benyttet av Lindenmayer for å beskrive algevekst:

<div class="p"><!----></div>

<ul>
<li> A &#8594; AB
<div class="p"><!----></div>
</li>
```

Placement — Level: 100: Testbrett; X: 2787; Y: 1561; Z: 0

DB: aocdev.idi.ntnu.no/aoc3 | Web: http://aocdev.idi.ntnu.no/ | Question Editor : Question Category

Figure B.1: Screenshot of the Question Editor from AoC Manual



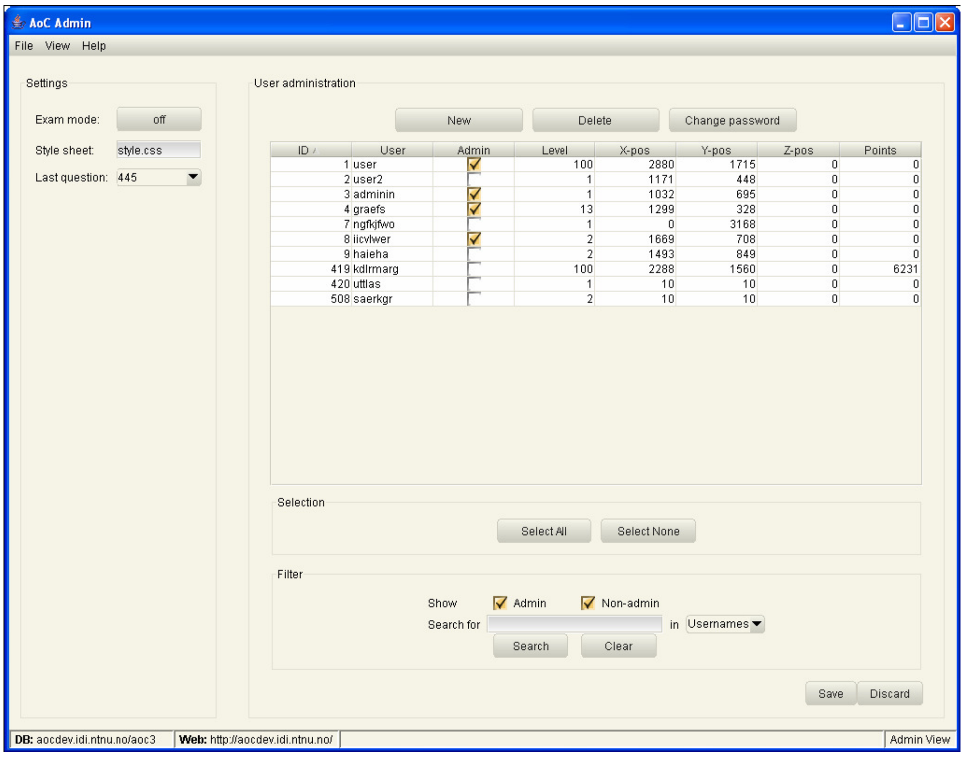
Group: Samlebånd
Name: Instruksjonssykel
Sequence #: 10
Format: HTML [✓ LaTeX]

```
    \includegraphics[width=0.8\textwidth]{figurer/faktabok/prosessor/MIPS-1sykel}
  }
  \caption{Instruksjonssykel}
\end{figure}

\begin{enumerate}
\item \emph{Hent} (eng: fetch): Hente instruksjon
\item \emph{Dekod} (eng: decode): Dekode instruksjon
\item \emph{Hent operander} (eng: operand fetch)
\item \emph{Utfør} (eng: execute): Utføre instruksjon
\item \emph{Lager} (eng: memory): Lese til / skrive fra datalager
\item \emph{Tilbakeskriving} (eng: writeback): Skrive resultat av
  instruksjon til registerfil
\end{enumerate}

Hver instruksjon må gjennom de fleste av disse stadiene hver gang de
utføres.  En vanlig instruksjon vil gå gjennom stadiene 1-2-3-4-6, en
LOAD-instruksjon vil gå gjennom stadiene 1-2-3-5-6 og en
STORE-instruksjon vil gå gjennom stadiene 1-2-3-5.

\paragraph{Hent}

Første stadium går ut på å hente instruksjonen fra
instruksjonslageret.

\paragraph{Dekod}

Etter at instruksjonen er hentet ut må den dekodes.  Dette vil si å se
```

Placement
Level: 008: Teoretikerne
X: 116 Y: 592 Z: 0

DB: aocdev.idi.ntnu.no/aoc3 | Web: http://aocdev.idi.ntnu.no/ | Text Editor : Fact

Figure B.2: Screenshot of the Text Editor from AoC Manual



Settings
Exam mode: off
Style sheet: style.css
Last question: 445

User administration

ID	User	Admin	Level	X-pos	Y-pos	Z-pos	Points
1	user	✓	100	2880	1715	0	0
2	user2		1	1171	448	0	0
3	adminin	✓	1	1032	695	0	0
4	graefs	✓	13	1299	328	0	0
7	ngfkjfwo		1	0	3168	0	0
8	iicvlwer	✓	2	1669	708	0	0
9	haieha		2	1493	849	0	0
419	kdlrmarg		100	2288	1560	0	6231
420	uttlas		1	10	10	0	0
508	saerkgr		2	10	10	0	0

Selection

Filter
Show ✓ Admin ✓ Non-admin
Search for in Usernames

DB: aocdev.idi.ntnu.no/aoc3 | Web: http://aocdev.idi.ntnu.no/ | Admin View

Figure B.3: Screenshot of the Admin View from AoC Manual

Appendix C

Er-Models

These are the ER-Model from the original World of Wisdom project.

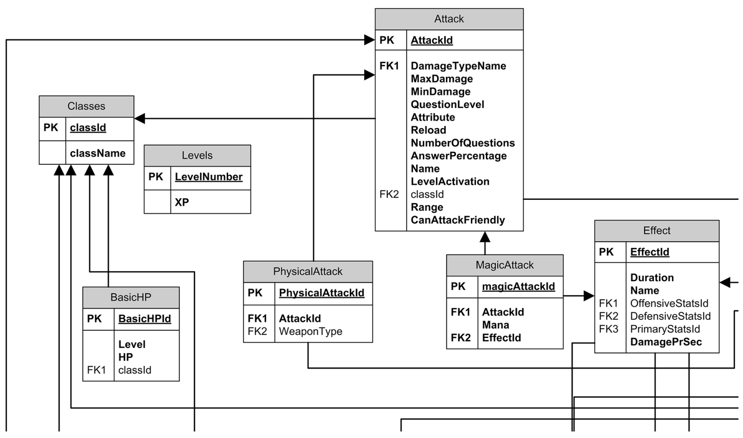


Figure C.1: ER Model 1-1

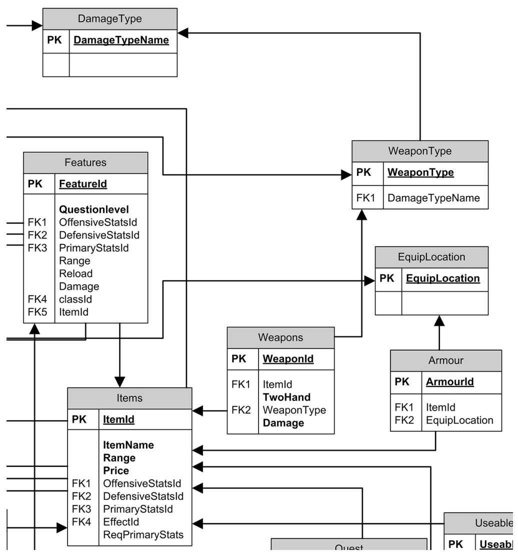


Figure C.2: ER Model 1-2

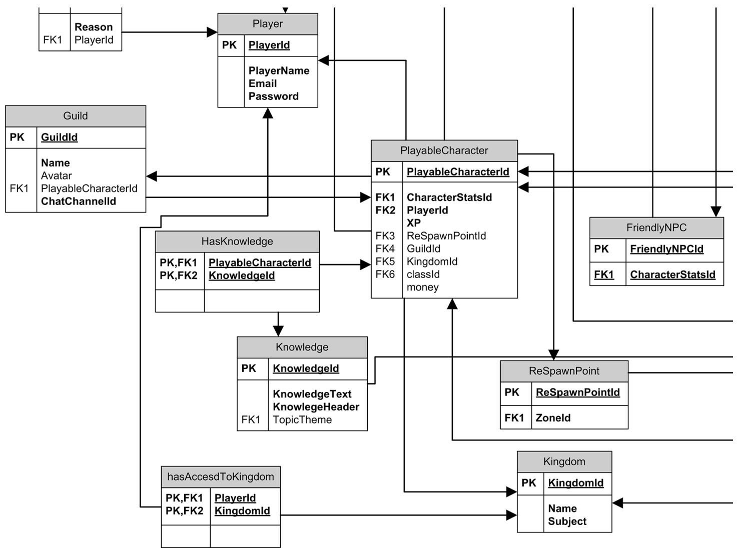


Figure C.3: ER Model 1-3

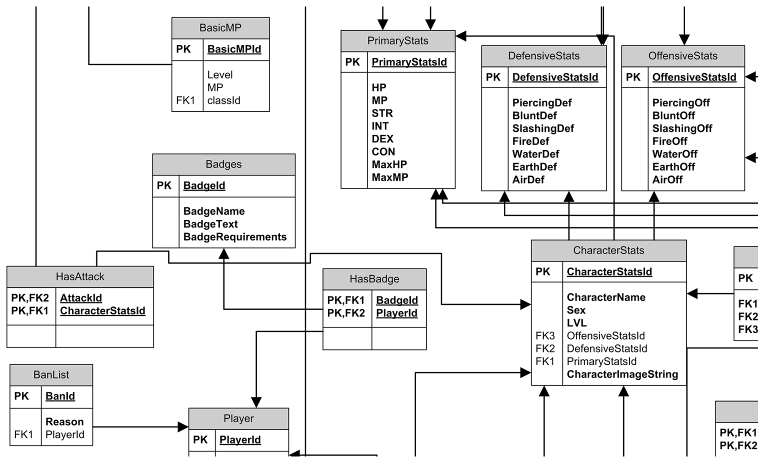


Figure C.4: ER Model 2-1

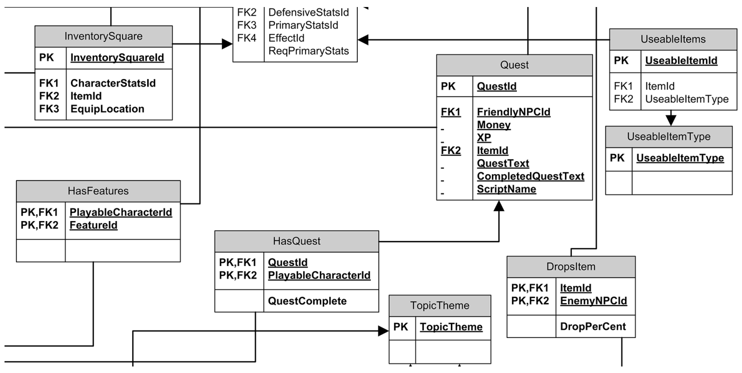


Figure C.5: ER Model 2-2

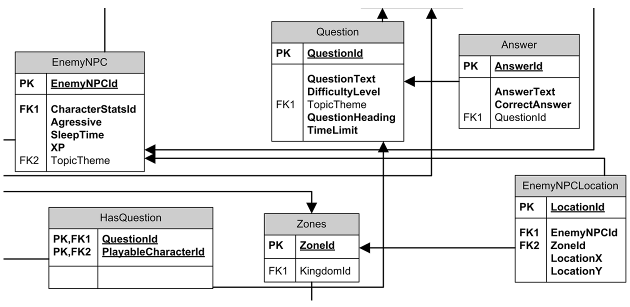


Figure C.6: ER Model 2-3

Appendix D

Use case

This chapter shows the textual use-cases that were used during user testing. Format used is from the book UML Distilled[Fow03].

D.1 Basic Functions

Table D.1: Use case: Start editor

Use case : 1.1 Start editor
Level : Basic functions
Main Success Scenario: 1: User starts the editor. (Editor.jar) 2: System display two windows, "World of Wisdom - Editor" and "Database connection". 3: System shows progress of "Database connection". When finished, displays an "OK" button. 4: User press OK button. 5: Database connection window is closed and editor is ready to use.
Extensions: 3a: Error occurs during loading. .1: System displays an error message "Connection timed out". .2: System shows an "OK" button. .3: Returns to step 4 in MSS.

Table D.2: Use case: Reconnect to Database Server

Use case : 1.2 Reconnect to Database Server
Level : Basic functions
Main Success Scenario: 1: System lost connection to Database Server 2: User selects "File" menu and then selects "Reconnect to DB" choice. 3: System shows progress of "Database connection". When finished, displays an "OK" button. 4: User press OK button. 5: Database connection window is closed and editor is ready to use.
Extensions: 3a: Error occurs during loading. .1: System displays an error message "Connection timed out". .2: System shows an "OK" button. .3: Returns to step 4 in MSS.

D.2 Map Editor

Table D.3: Use case: Create new kingdom

Use case : 2.1 Create new kingdom.
Level : Map Editor
Main Success Scenario: 1: User selects "Map Editor" tab. 2: User selects "Kingdom" on menu. 3: User selects "Create new Kingdom". 4: System displays new window. 5: User enters name for new kingdom and class subject name. 6: User selects "Create". 7: System shows the name of the Kingdom in the tree structure.

Table D.4: Use case: Create new Zone

Use case : 2.2 Create new Zone.
Level : Map Editor
Main Success Scenario: 1: User selects the name of the Kingdom. 2: User press "Create zone". 3: System displays new window. 4: User enter name and size of zone. 6: User selects "Create". 7: Systems shows the new zone in the tree structure.

Table D.5: Use case: Save kingdom to Database Server

Use case : 2.3 Save kingdom to Database Server.
Level : Map Editor
Main Success Scenario: 1: User selects "Kingdom" on menu. 2: User selects "Save kingdom". 3: System shows a combobox with names of zones in Kingdom. 4: User selects the zone they want new players to start in. 5: System saves the kingdom to the Database. 6: User press "OK" when the kingdom has been successfully saved and reloaded.
Extensions: 6a: Connection to database is lost. .1: System gives an error message "Lost connection". .2: User may attempt to reconnect to Database. .3: Return to step 1 in MSS.

Table D.6: Use case: Load kingdom from Database Server

Use case : 2.4 Load kingdom from Database Server.
Level : Map Editor
Main Success Scenario: 1: User selects "Kingdom" on menu. 2: User selects "Load kingdom". 3: System shows a combobox with names of Kingdoms in database. 4: User selects a kingdom and press "OK" . 6: System loads the kingdom and shows progress in a new window. 7: User press "OK" when the kingdom has been successfully loaded.
Extensions: 7a: Connection to database is lost. .1: System gives an error message "Lost connection". .2: User may attempt to reconnect to Database. .3: Return to step 1 in MSS.

Table D.7: Use case: Save kingdom to Hard drive

Use case : 2.5 Save kingdom to Hard drive.
Level : Map Editor
Main Success Scenario: 1: User selects "Kingdom" on menu. 2: User selects "Save kingdom to HD". 3: System displays a confirmation box that the kingdom has been saved.

Table D.8: Use case: Load kingdom from Hard drive

Use case : 2.6 Load kingdom from Hard drive.
Level : Map Editor
Main Success Scenario: 1: User selects "Kingdom" on menu. 2: User selects "Load kingdom from Hard drive". 3: System shows a file dialog where the user can browse through their local files. 4: User selects a kingdom they want to load. 5: System shows a loading window while loading. 6: User press "OK" when the kingdom has been successfully loaded.

Table D.9: Use case: Save zone to template

Use case : 2.7 Save zone to template.
Level : Map Editor
Main Success Scenario: 1: User selects the Zone they want to save. 2: User selects "Zone Template" on menubar. 3: User press "Save zone" button. 4: System prompts user for a name for the template. 5: User inputs name for template. 6: System saves the zone in the Database. 7: User press "OK" when the zone has been successfully saved.
Extensions: 5a: A template with that name already exists in the database. .1: System gives a message to user that the name is not unique. .2: Returns to step 4 in MSS. 6a: Connection is lost .1: System gives error message "Lost connection" .2: User may try to reconnect and return to step 2 in MSS.

Table D.10: Use case: Load zone from template

Use case : 2.8 Load zone from template.
Level : Map Editor
Main Success Scenario: 1: User selects the name of the Kingdom. 2: User selects "Zone Template" on menubar. 3: User press "Load zone". 4: System displays a list of zones. 5: User selects a zone and press "OK". 6: System prompts user for a name for the zone. 7: User enter name for zone and press "OK". 8: System adds zone to the kingdom.
Extensions: 7a: The name already exists in the kingdom. .1: System gives a message to user that the name is not unique. .2: Returns to step 6 in MSS.

Table D.11: Use case: View contents of a zone

Use case : 2.9 View contents of a zone.
Level : Map Editor
Main Success Scenario: 1: User selects the name of the zone. 2: System draws the contents of the zone in middle of editor. 3: System shows content of zone in tree structure. 4: System displays lists of objects that can be placed in zone.

Table D.12: Use case: Add object to zone

Use case : 2.10 Add object to zone.
Level : Map Editor
Main Success Scenario: 1: User selects the Zone. 2: User select list of object on right side(Enemy NPC, Friendly NPC or Static Object). 3: User selects an object in the list. 4: User press place. 6: System adds object to zone and selects it. 7: System draws a square around the selected object.
Extensions: 4a: Type selected is Enemy NPC. .1: System prompt the user to select a theme topic for the NPC. .2: User selects a topic and press "Ok". .3: Return to step 6 in MSS.

Table D.13: Use case: Change position of object in zone

Use case : 2.11 Change position of object in zone.
Level : Map Editor
Main Success Scenario: 1: User selects a Zone. 2: User selects an object by selecting it from the tree structure or by click on the object where it is drawn. 3: System draws a square around the selected object. 4: User clicks the left mouse button inside the square and drags the square to the desired position. 6: System shows the new position by moving the square while the user drags the object. 7: User releases the button at the desired position. 8: System moves the world object to the new position.

Table D.14: Use case: Remove objects from kingdom

Use case : 2.12 Remove objects from kingdom.
Level : Map Editor
Main Success Scenario: 1: User selects object in tree structure (Zone, World objects or travel trigger). 2: User press "Delete" key. 3: System presents a confirmation window. 4: User press "Yes". 6: System deletes the object from the kingdom.

Table D.15: Use case: User draws background in a Zone

Use case : 2.13 User draws background in a Zone.
Level : Map Editor
Main Success Scenario: 1: User selects the Zone. 2: User select list background tiles on right side. 3: User selects a tile in the list. 4: User then left click where they want to draw that tile. 6: System changes the tiles in the background to the new tile. 7: System changes surrounding tiles to make a gradient transition.

D.3 WorldObject Editor

Table D.16: Use case: View a world object

Use case : 3.1 View a world object.
Level : World Object Editor
Main Success Scenario: 1: User selects "World Object Editor" tab. 2: System shows a tree structure with World Objects (Enemy NPC, Friendly NPC and Static Object). 3: User opens a World Object category. 4: System shows all World Object of selected category. 5: User selects a World Object. 6: System presents the information of the selected World Object.
Extensions: 3a: Empty list. .1: System is not connected to Database or no world object in that category exists.

Table D.17: Use case: Add a new world object

Use case : 3.2 Add a new world object.
Level : World Object Editor
Main Success Scenario: 1: User selects a World Object category. 2: User fills in information on NPC. 3: User press "Create new Object". 4: System saves the new Object to database and refreshes the World object list.
Extensions: 4a: Error occurs during saving. .1: System gives an error message, "Lost connection". .2: User press "OK" to return to the Editor.

Table D.18: Use case: Save changes to world object

Use case : 3.3 Save changes to world object.
Level : World Object Editor
Main Success Scenario: 1: User selects "World Object Editor" tab. 2: System shows a tree structure with World Objects (Enemy NPC, Friendly NPC and Static Object). 3: User opens a World Object category. 4: System shows all World Object of selected category. 5: User selects a World Object. 6: System presents the information of the selected World Object. 7: User alters the information. 8: User press "Save Changes". 9: System save changes to database and updates the list.
Extensions: 9a: Error occurs during saving. .1: System gives an error message, "Lost connection". .2: User press "OK" to return to the Editor.

Table D.19: Use case: Delete a world object

Use case : 3.4 Delete a world object.
Level : World Object Editor
Main Success Scenario: 1: User selects "World Object Editor" tab. 2: System shows a tree structure with World Objects (Enemy NPC, Friendly NPC and Static Object). 3: User opens a World Object category. 4: System shows all World Object of selected category. 5: User selects a World Object. 6: User press "Delete" button shown. 7: System prompts the user for confirmation on deletion. 8: User press "Yes". 9: System deletes the object and updates the list.
Extensions: 9a: Error occurs during saving. .1: System gives an error message, "Lost connection". .2: User press "OK" to return to the Editor.

D.4 Question Editor

Shows the use case examples for the Question Editor.

Table D.20: Use case: View a question

Use case : 4.1 View a question.
Level : Question Editor
Main Success Scenario: 1: User selects "Question Editor" tab. 2: System shows a tree structure with Questions and some empty fields. 3: User double-click "WoW - Questions" in the Tree structure. 4: System shows the topics. 5: User selects a topic. 6: User selects a question within the topic. 7: System presents the information for that question.
Extensions: 3a: Empty list. .1: System is not connected to Database or no questions exists.

Table D.21: Use case: Add a new Question

Use case : 4.2 Add a new Question.
Level : Question Editor
Main Success Scenario: 1: User selects "Question Editor" tab. 2: System shows a tree structure with Questions and some empty fields. 3: User double-click "WoW - Questions" in the Tree structure. 4: System shows the topics. 5: User selects a topic. 6: User types in Question Heading, text, question level and answers. 7: User selects the correct answer alternative. 8: User press "Add new question". 9: System saves question to Database. 10: System reloads the list of questions.
Extensions: 9a: Error occurs during saving. .1: System displays an error message "lost connection". .2: System shows an "OK" button.

Table D.22: Use case: Save changes to a Question

Use case : 4.3 Save changes to a Question.
Level : Question Editor
Main Success Scenario: 1: User selects "Question Editor" tab. 2: System shows a tree structure with Questions and some empty fields. 3: User double-click "WoW - Questions" in the Tree structure. 4: System shows the topics. 5: User selects a topic. 6: System displays the questions for that topic. 7: User selects a question. 8: System presents information for that question. 9: User alters the information given by the system. 10: User press "Save changes". 11: System saves the question and updates list.
Extensions: 11a: Error occurs during saving. .1: System displays an error message "lost connection". .2: System shows an "OK" button.

Table D.23: Use case: Delete a Question

Use case : 4.4 Delete a Question.
Level : Question Editor
Main Success Scenario: 1: User selects "Question Editor" tab. 2: System shows a tree structure with Questions and some empty fields. 3: User double-click "WoW - Questions" in the Tree structure. 4: System shows the topics. 5: User selects a topic. 6: User selects a question. 7: User press "Delete" key. 8: Systems prompt user if they are sure they want to delete. 9: User press "Yes". 10: System deletes question and updates list.
Extensions: 10a: Error occurs during deletion. .1: System displays an error message "lost connection". .2: System shows an "OK" button.

Appendix E

User manual for World Editor

E.1 Menubar

The menubar is divided into three parts, File, Kingdom and Zone Template. Under File, there are currently just a button for reconnecting to the Database Server if the connection is lost. A screenshot of the menubar and the choices for Kingdom are shown in Figure E.1. Following is a list of the choices and a short description.

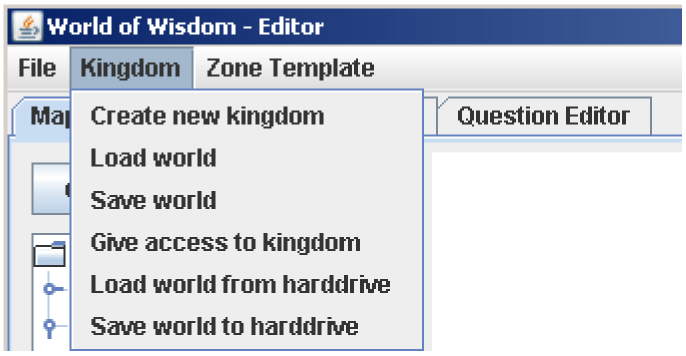


Figure E.1: Screenshot of the Menubar in the World Editor

Create new kingdom Displays the Create Kingdom window in Figure E.3.

Load world Displays a list with all the names of kingdoms in the database. You can then select one and load it into the editor.

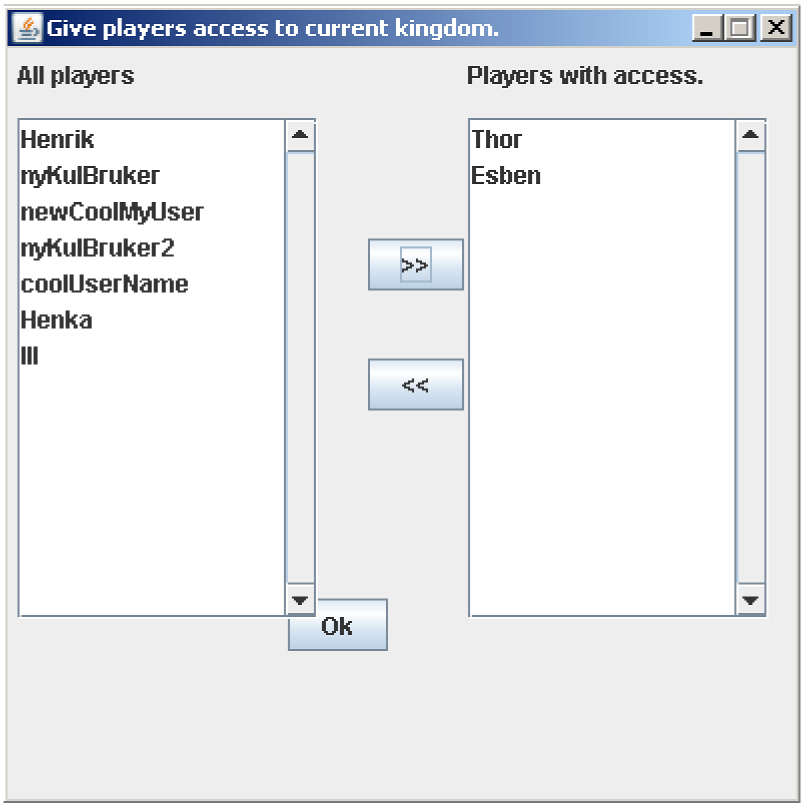
Save world Saves the current kingdom to the database.

Give access to kingdom The user can select which player that will have access to join the current kingdom. Screenshot is shown in Figure E.2.

Load world from harddrive Displays a file dialog that can be used to browse the hard drive for a saved kingdom. The selected kingdom can be loaded into the editor.

Save world from harddrive Saves the current world to the harddrive in the "/Resources/KingdomBackups/" folder.

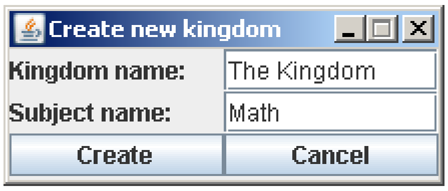
Under Zone Templates the choice to Load a Zone template from the hard drive is given, and to save the currently drawn zone to a template. For both options, the user are given the choice to give a new name to the zones.



Henrik
nyKulBruker
newCoolMyUser
nyKulBruker2
coolUserName
Henka
lll

Thor
Esben

Figure E.2: Screenshot of the Player Access window in the World Editor



Kingdom name: The Kingdom
Subject name: Math

Figure E.3: Screenshot of the Create Kingdom window in the World Editor

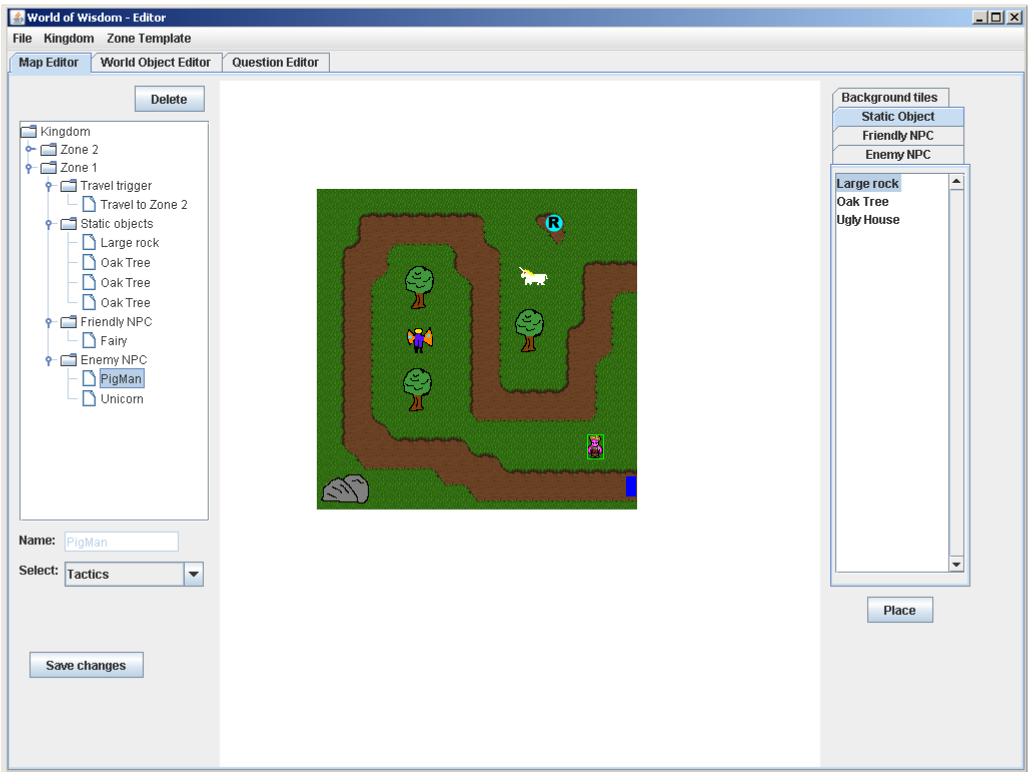
E.2 Map Editor

When starting the World Editor the Map Editor will be empty. To start select Kingdom from the menubar and either "Load World" from database or hard drive, or create a new Kingdom. When creating a new Kingdom, a new window will appear where you type in the name of the Kingdom and the course subject. Screenshot of the Create Kingdom window is shown in Figure E.3.

The Map Editor is divided into three panels, MapObjects, MapGraphics and MapPlaceables. A screenshot of the Map Editor is shown in Figure E.4.

E.2.1 MapObjects

The MapObjects panel is divided into three parts, top buttons, a tree structure, and properties panel.



Name: PigMan

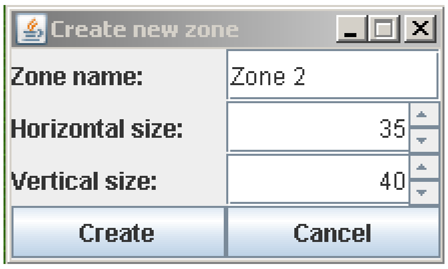
Select: Tactics

Figure E.4: Screenshot of the MapEditor in the World Editor

Top buttons

Above the tree structure are two buttons, "Create New" and "Delete". These buttons are hidden when they can not be used.

The "Create New" button can be used to create new Zones, if the root or a zone is selected. Or to create new Travel trigger in a zone, if the travel trigger category is selected, or a trigger within the zone. A screenshot of the Create Zone window is shown in Figure E.5.



Zone name: Zone 2

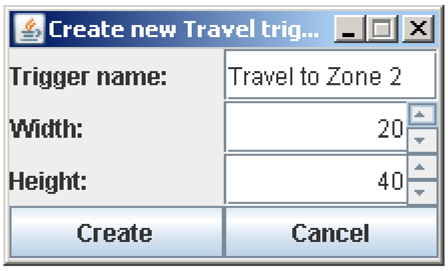
Horizontal size: 35

Vertical size: 40

Figure E.5: Screenshot of the Create Zone window in the World Editor

Create new Travel Trigger

To create a new Travel Trigger, select the Travel Trigger category in the desired zone and click the "Create new" button. This will display the Create Travel Trigger window shown in Figure E.6. Here you select the name and size of the trigger. After it has been created you can select the destination for the trigger by selecting it in the tree structure and changing the Destination value in the properties panel beneath the tree.



Trigger name:	Travel to Zone 2
Width:	20
Height:	40

Figure E.6: Screenshot of the Create Travel trigger window in the World Editor

Tree structure

The tree structure that contains all the elements in the currently loaded kingdom. When a zone is selected, the zone is drawn in the MapGraphics panel. When a object within a zone is selected, a green square is drawn around the selected object in the MapGraphics panel. In addition to use the "Delete" button for deleting selected objects, the delete key on the keyboard can be used.

Properties panel

The properties panel displays information on the currently selected object in the tree structure. The information displayed differ for the types of object selected.

Enemy NPC gives the option to change the theme topic. While Travel Trigger gives the option to change the size and destination of the travel trigger.

E.2.2 MapGraphics

MapGraphics panel draws the zone that is selected in the tree structure.

Screen focus

The screenfocus can be moved by clicking the right mouse button. This will make the place clicked in the zone the center of the drawing panel.

When objects are placed from the MapPlaceables panel, they will be placed at the center of focus and automatically selected.

Selection and Movement

Objects in the zone can be selected by clicking the left mouse button on the object, or selecting the object in the tree structure. A green square will be drawn around the selected object. When an object has been selected it can be moved to another position by pressing the left mouse button, and dragging the object to a new position. While dragging, the green square will show the position, and when the mouse button is released the object is moved.

Drawing background

When a tile has been selected in the Background Tile list in the MapPlaceables panel, you can draw the tile in the background by clicking the left mouse button on the desired location. The tiles around the placed tile will automatically be changed to make a smooth transition.

E.2.3 MapPlaceables

The MapPlaceables has four lists of object types that can be placed in a zone.

World Objects

World Object (Enemy NPC, Friendly NPC and Static Objects) can be placed in the kingdom by selecting one of them in the list, and clicking the Place button beneath the list.

Background Tile

In the Background Tile list, when a tile is selected, you can draw in the MapGraphics panel. To cancel drawing, and be able to select objects again, change to one of the World Object lists.

E.3 World Object Editor

The World Object Editor can create new, change existing and delete Enemy NPC, Friendly NPC and Static Objects. The World Object Editor has a tree structure on the left side, and the variable textfields are shown on the right side. A screenshot of the World Object Editor is shown in Figure E.7.

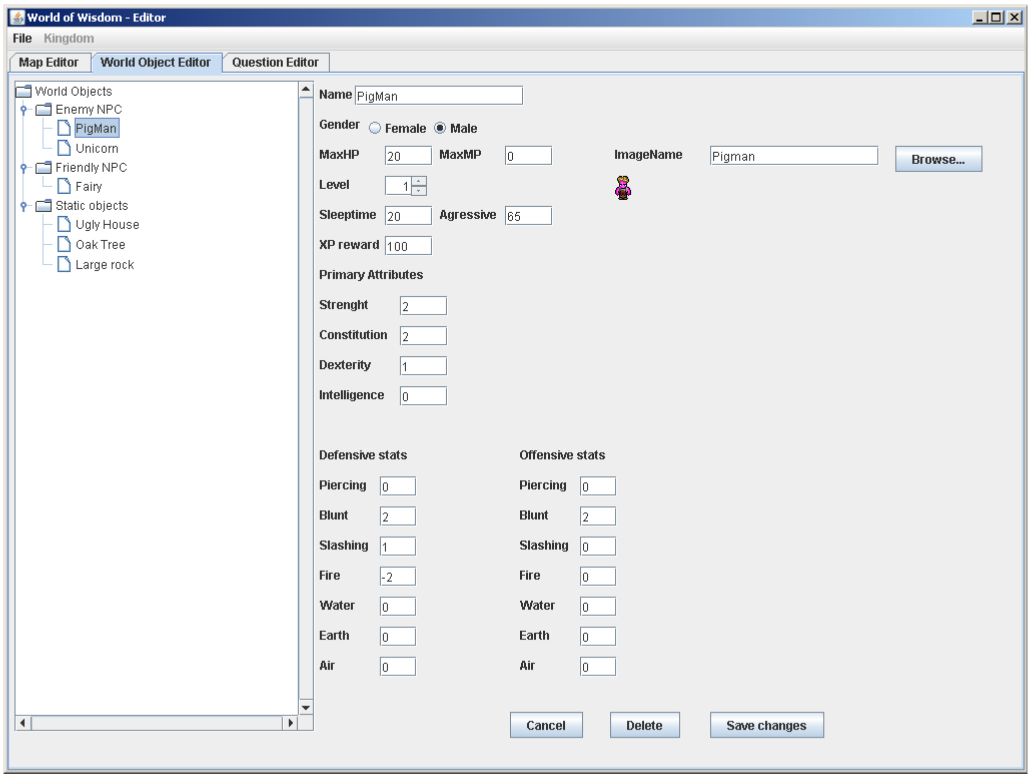
E.3.1 Create new object

To create a new object, select the category name of the object type you want to create. The variables should then be cleared in the right panel. Fill in the information and click the "Create new Object" button in the bottom right corner. When the create button is pressed, it will send the object to the database, and return with the new list of objects.

E.3.2 Selecting image

For selecting an image for the object, either type in the name if you know the name of the image and it is placed in the correct folder ("/Resources/Graphics/WorldObjects/"). The image beneath the imagepath should be automatically updated and display the image if it is found.

Alternatively you can press the Browse button. This will display a file dialog that can be used to browse the hard drive for an image. The selected image will be automatically copied in the correct folder, and the imagename will be automatically filled in. If the image is already in the right folder, it will not be copied but the imagename will still be filled in.



- World Objects
  - Enemy NPC
    - PigMan
    - Unicorn
  - Friendly NPC
    - Fairy
  - Static objects
    - Ugly House
    - Oak Tree
    - Large rock

Name: PigMan

Gender: ○ Female [✓ Male]

MaxHP	20	MaxMP	0	ImageName	Pigman
Level	1				
Sleeptime	20	Agressive	65		
XP reward	100				

Primary Attributes

Strenght	2
Constitution	2
Dexterity	1
Intelligence	0

Defensive stats		Offensive stats	
Piercing	0	Piercing	0
Blunt	2	Blunt	2
Slashing	1	Slashing	0
Fire	-2	Fire	0
Water	0	Water	0
Earth	0	Earth	0
Air	0	Air	0

Figure E.7: Screenshot of the World Object Editor in the World Editor

E.3.3 Alter information & Deletion

To alter information on an object, select the object in the tree structure, change the information and press the save changes button. The changes will be saved to the database, and the updated list will be retrieved.

E.3.4 Level

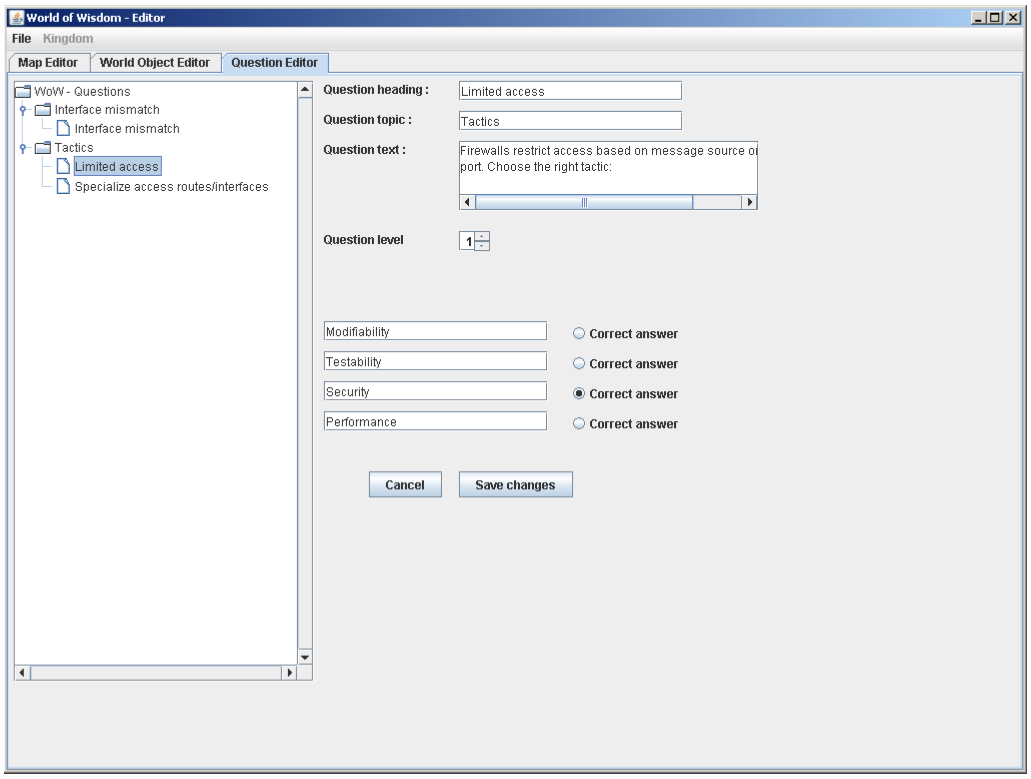
When altering the level of the NPC the attributes are automatically altered.

E.4 Question Editor

The Question Editor is used to manage the questions in the database. A screenshot of the Question Editor is shown in Figure E.8. The tree structure on the left contains all the questions in the database and their theme topics. The topics are the folders and the leafs are the questions.

E.4.1 Create new Question

To create a new question, select the topic in the tree structure and fill out the information and press "Add new question" button. To create new topic, rename the Question Topic to another name.



WoW - Questions
- Interface mismatch
  - Interface mismatch
- Tactics
  - Limited access
  - Specialize access routes/interfaces

Question heading : Limited access

Question topic : Tactics

Question text : Firewalls restrict access based on message source o
port. Choose the right tactic:

Question level 1

Modifiability ○ Correct answer

Testability ○ Correct answer

Security ● Correct answer

Performance ○ Correct answer

Figure E.8: Screenshot of the Question Editor in the World Editor

E.4.2 Delete question

Select the question you want to delete in the tree structure and press the delete key. A confirmation box will be displayed.

E.4.3 Alter question

To change the information in a question. Select the question and change the fields and press the "Save changes" button.

E.4.4 Field explanation

Short description of the fields for a question.

Question Heading is the heading for the question.

Question Topic is the theme topic, this is used when a creature is added to a world. The creature will give questions that belongs to this topic.

Question Text is the main text for the question.

Question Level determins the difficulty of the question. Level 1 is easy, while level 5 is the hardest questions. The type of attack the player uses determines which question they get. The more powerful attack, the more difficult question. If a question of the difficulty does not exist, a lower level question is shown.

Answers The alternative answers to the questions. There are four answers, and the correct one is selected by using the radio button.

Appendix F

Installation Guide

This chapter contains a thorough explanation of how to setup the World of Wisdom project and the new World Editor.

F.1 PostgreSQL Database

First the database that runs in the background is needed. For this you first need to install the PostgreSQL software. This can be found on the homepage of PostgreSQL[Pos]. There are two types of installer, one click installer, and the pgInstaller. The one we used for these projects is the pgInstaller. Both should work fine, aslong as during the install you remember to also include the JDBC drivers, which is needed for interaction with Java applications.

Step by step install of PostgreSQL Database program:

1. Download PostgreSQL install (One click or pgInstaller version 8.3.6 or later).

2. Start the installation file.

3. The standard settings in Installation options are fine, so click next there.

4. Select desired Account name and password for the service. (Passwords are fairly strict in PostgreSQL, and is required to be quite long and include signs/numbers compared to other softwares).

5. Initialize database cluster. Here you type in a superuser name and a new password (though we used same password for both).

6. Enable procedural languages. Keep PL/pgsql selected.

7. Enable contrib modules. Keep as is.

8. When installation is done. Launch Stack Builder at exit.

9. In Stack Builder, select the PostgreSQL Database service and click next.

10. Open the Database Drivers category and select "pgJDBC" (currently v8.3-604-1). And next to install package.
    This one is needed for java to connect to the database.

Now that the PostgreSQL software is installed on the computer. Start "pgAdmin III" , this is the user interface for the PostgreSQL database.
Step by step to restore the database from a *.backup postgreSQL file:

1. On startup, select PostgreSQL Database Server, right-click and select Connect. (You might need to change username, which can be done in properties).
2. Type in password you created during install.
3. When logged in, select Databases, right-click and select "New Database..."
4. Type the desired name for your database, and select your user as owner and click Ok.
5. Now right-click your newly created database and select "Restore...".
6. In fileName select the backup of the World of Wisdom database and click Ok. (If your username is not postgresSuper it will give a message that 100+ errors where ignored during restore. This is because we had set postgreSuper as owner of the tables and views. It will then automatically set your current username as the owner of these tables. So no need to run restore again, click cancel.)

Now the database has restored and is ready for use by the Database Server application.

F.2 Database Server application

In the World of Wisdom applications there are one settings file for each application. These are located in "/Resources/Settings/". The settings file for the Database Server is called "database.ini". This contains the network settings for the server. Contents and description are shown below, used camel-case on the names for easier reading, though they are not case sensitive when being read by the applications:

LobbyListenPort this is the port number that the Database Server listens for connection from Lobby Servers.

EditorListenPort as above, but for World Editors.

WorldServerListenPort as above, but for World Servers.

PostgreSQLip this is the Ip address to the computer that has the PosgreSQL Database Server installed. If the Database is run on the same computer as the PostgreSQL server you can type in "Localhost".

DatabaseName this is the name of the database created in the "pgAdmin III" tool.

Changing the port numbers is not necessary, unless you want to run several instances of one server on one computer. When the settings has been set, the Database can be run by starting the Database.jar. The main method is "databaseServer.DBServerMain.java". When the Database Server starts, the user will be prompted with a username and a password. The username and password are the ones that you use to connect to the PostgreSQL Database through the "pgAdmin III" application.

F.3 Lobby Server application

As with the Database Server, Lobby Server uses settings file for network settings, "lobbyServer.ini". Contents are shown below:

DatabaseIP ip adress to the Database Server you want the Lobby server to connect to.

DatabasePort the port number that the Database Server is listening to Lobby Servers.

ClientListenPort listens for connection from Client applications.

WorldServerListenPort listens for connections from World Server applications.

As with the Database Server, listen ports are not necessary to change unless you want several Lobby Servers on one computer. When settings has been set, the Lobby Server can be run by starting the LobbyServer.jar. Main method is "clientSessionHandler.LobbyCommand.java".

F.4 World Server application

WorldServers network settings are stored in "worldServer.ini". Contents of the "world-Server.ini":

LobbyIP ip adress to the Lobby Server that the World Server will register its availability to.

LobbyPort the port number that the Lobby is listening to World Servers.

DatabaseIP ip adress of the Database Server that the World Server will use.

DatabasePort the port number it will connect to the Database Server.

ClientListenPort the port number it listens for Client applications to connect to.

When the settings has been set, the World Server can be run by starting the WorldServer.jar.

When the WorldServer starts, the user will be asked to select a kingdom for the World Server to load from the Database. When the kingdom has been successfully loaded into the World Server, it will connect to the Lobby Server as an available Kingdom and is ready for Clients to connect. Main method is "serverControls.ServerControl.java".

F.5 Client application

The Client applications network settings are stored in "client.ini". Contents are shown below:

LobbyIP ip adress of the Lobby Server.

LobbyPort port number that the Lobby Server listens to Client on.

Client is started by running the Client.jar. Main method is "clientState.ClientGameLoop.java". The Client jar must include all the graphics that is used in the Kingdom they want to connect to. When the Client is started, the user types in username and password, and they will get a list over kingdom they have access to. They can the connect, select a character and play in the kingdom.

F.6 World Editor

The settings for the World Editor is in "editor.ini". Contents are shown below:

DatabaseIP ip adress of the Database Server.

DatabasePort port number it uses to listen for World Editors.

The World Editor is started by running Editor.jar. Its main method is "Editor-Main.java". Upon startup the Editor will connect to the Database Server and download the data needed for the Editor. An user manual for the Editor main is in appendix E.

F.7 Summary

This summary will have a short list over the action that needs to be done to start the World of Wisdom project.

1. Install PostgreSQL software. Section F.1
2. Restore World of Wisdom database. Section F.1
3. Set the settings for all Server applications and the Client.
4. Start Database Server, this one is needed by all other applications and must be started first. Section F.2
5. Start Lobby Server, this one is needed by Client, World Server. Section F.3
6. Start World Server, needed by Client for playing in a kingdom. Section F.4
7. Start Client, the last application in the World of Wisdom project. Section F.5
8. Start World Editor, this one can be started as soon as the Database Server is up, and does not require, Lobby Server, World Server or Client to be running. Section F.6

   This should summarize the setup for running the project. And the order of the applications.