John Ola Tollefsrud

The Educational Game Editor

The Design of a Program for Making Educational Computer Games

Trondheim, March 2006

Norges teknisk-naturvitenskapelige universitet
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

Master's thesis
Programme of study: Master of Science in Computer Science

Supervisor: Arvid Staupe, IDI

NTNU
Innovation and Creativity

Preface

In the autumn semester 2004, Nicolai Friis and I decided to work with e-learning and computer game based learning in our major project. We learned a lot about e-learning today, and also about the effects of using a computer game instead of exercises in a subject in our university.

This gave me inspiration, and I wanted to continue working with e-learning and game based learning on my master thesis. After some discussions with my teaching supervisor, the idea of the Educational Game Editor was born, and the work began.

I want to thank everyone that has given me help and inspiration with the master thesis. I particularly want to thank my teaching supervisor Arvid Staupe. He helped me to come up with ideas for my thesis, and gave my much good advice and inspiration on the way.

John Ola Tollefsrud

Trondheim February 2006

Abstract

This report is about computer game based learning, how to make a program for making educational games, the possibilities to use a hypermedia structure for storage of the data in an educational game, and different learning theories related to computer game based learning.

The first part is about the different learning theories behaviourism, cognitivism, constructivism, socio-constructivism, and situated learning. The different theories are related to learning games, and a classification of game based learning is also given.

Hypermedia is a smart and efficient way of organizing data, and is a relevant solution for use in education and games. The relationship between data, information and wisdom is central, and how the hypermedia base is constructed and different information structures are described. The advantages and limitations of use of hypermedia in education are discussed, and examples of use, as in OPSYS and the Mobile instruction system, are given.

There exist some computer games for use in higher education, and some of them are described. To make a good educational, many certain requirements have to be fulfilled both aspects in game design and learning aspects.

The main part of the report is about the Educational Game Editor. The idea is to design a program for making computer games for use in education. Before the design, the Software Requirements Specification is presented, containing functional and quality requirements, and scenarios to exemplify the requirements.

The conceptual design of the program gives an overall description and describes the phases of creating a game and the elements the game consists of: file management, object management, Library, and Tools.

The main architectural drivers are usability and availability. The program must be easy to use and be stable and not crash.

An example of making a simple game about the history of Trondheim explains how to use the program in steps, and gives a good guide for the users to make their own game.

Table of contents

1. Introduction

1.1 Problem definition

Design of a system for making educational computer games, related to different learning styles and hypermedia as a structure for organizing data

Computer game based learning is a new and exciting area. There exist many different games on the market, but the quality is varying. Many persons that work in the educational business might want to make their own game. The main task is to design a system used by teachers for making computer games for educational purposes. The program is not meant to be implemented.

Learning theories related to computer games in education and the use of hypermedia in education and games is also meant to be covered in the project.

1.2 Motivation for the project

Computer games have been a big business for many years, and the industry is still growing. Games have so far been designed mainly for entertainment and fun. But during the last years, games made for learning have entered the school system. However, most educational games have been made for use in primary school; not many games are fitted for use in higher education. Another problem is the varying quality of this kind of games; it is not easy to make a good game play and achieve a good educational content in the same game.

The information society forces the people employed in the education business to change some of their methods, and the use of computer games in education will probably grow in the future.

To make it easier for people employed in the education business to design their own games for use in their classes, a program for making computer games with educational purposes is a good idea. This program will make it able to make computer games without any knowledge of a programming language or design of computer programs.

1.3 Goals

This report is the first step in developing a program for designing educational computer games, and also relates learning theories to educational games and how hypermedia can be used in education and has the following goals:

1. Give an overview over different learning theories and find out how they relate to computer game based learning.
2. Explore hypermedia as a way to organize data, and find out how it can be used in e-learning and game based learning.
3. Find out what is required to make a good computer game.
4. Make a software requirements specification for a program for making educational computer games.
5. Design a program for making educational computer games.

1.4 Overview of the report

The report is divided in 3 main chapters. Chapter 2 gives an overview over different learning theories and relates them to computer game based learning. Chapter 3 describes hypermedia as a way to organize data in education and games. Chapter 4 contains sections about computer games in education today and what is required to make a good educational computer game.

The main chapter is chapter 5, which the requirements and design of the program Educational Game Editor.

Chapter 6 is discussion about the project, chapter 7 conclusion, and chapter 8 references.

2. Learning theories

There are several learning theories. The different theories are based on studies and theories developed by pedagogues and psychologists, and they are important to understand how we can make benefit of information and communication technology to improve learning. Furthermore, the different theories are important to understand how to design educational computer games.

The different theories are presented first, before a classification of computer game based learning, and the learning theories related to computer game based learning.

2.1 Relevant learning theories

Here is a short summary of relevant learning in information and communication technology.

2.1.1 Behaviourism

Behaviourism is science that includes everything that can be observed and measured. Studying human behaviour under different conditions makes us able to get information about human behaviour and adaptation. This kind of learning is called conditional learning. What happens inside the persons mind is not measurable and is not of interest here. The conditional learning appears as response on given stimuli. If the stimulus is repeated enough times, the person will “learn” which response is correct depending on the given stimulus.

Behavioural learning theory describes learning as a relative change in behaviour that is caused by experience. To accelerate this process, a system of rewards is being used. That means that the person that is supposed to learn is rewarded when he gives correct response on given stimuli [Eidsmo, Kolås 2005].

2.1.2 Cognitivism

The cognitive learning process can be seen as the opposite to behaviourism. Cognitive theories emphasize the inner mental processes that go on during the learning. The scientists are interested in finding out what happens from the stimuli are given until the response is coming. Cognitivism emphasizes the intellectual processes perception and thinking that underlie human changes in behaviour and attitude.

Cognitive theories say that learning can be defined as all changes in human personality that not directly or indirectly lead back to heritable factors. Cognitive theories do not use systems of rewards, but say that learning is a result of the desire to learn and the ability to use earlier gained experience as a basis for new knowledge [Eidsmo, Kolås 2005].

2.1.3 Constructivism

Constructivism is based on cognitivism. It says that all stimuli are expressed through existing knowledge and conceptions of the surroundings. Learning in this context will be a process that concerns the whole human personality. Constructivism emphasizes that man is not a passive object related to learning. The human being takes active part in the learning process in such a way that the given stimuli are being sorted, interpreted and adapted to the human’s system. In this way, the knowledge is constructed by man itself and this construction happens during an interaction between the influences that man is exposed to and what man does with the influences. It is also important that instead of getting rewards, the human being is seeking sense and relations in the existence.

Learning is achieved when the person that is going to learn constructs his own knowledge through the experiences and the knowledge he had earlier on and extends these with new information through taking active part in the learning process. The teacher is not a professional authority that is instructing the student any more. Instead, the teacher is a person on the same level as the students, and he is encouraging and guiding the student in the learning process. The responsibility for learning is moved from the teacher to the person who is learning [Eidsmo, Kolås 2005].

2.1.4 Socio-constructivism

It is common to distinguish cognitive and social orientated constructivism. The cognitive constructivism emphasizes exploration and discovery regarding how to explain the learning process. In social constructivism, the focus is on collaboration as a learning source. Collaboration in groups is valuable because every participant in the group has his own expertise.

Socio-constructivism indicates the importance of using new learning communities. There is often not possible to choose between classroom based learning and authentic experience, and therefore information technology can be used to make learning in classrooms more authentic [Eidsmo, Kolås 2005].

2.1.5 Situated learning

Theories in situated learning emphasizes that learning is a normal consequence of an activity or a context, in a particular culture. This is very different from traditional classroom based learning where the syllabus is presented in an abstract way in no contextual relation.

Situated learning is deeply rooted in literature attached to a context through authentic tasks and situations – in other words, the human being learns best through work experience with real and authentic tasks.

A good example of situated learning is called "master learning". The person who is the master has knowledge about the subject and the learning process is divided into three stages where the knowledge is communicated to the student. In the first stage, the student observes how it shall be performed, in the second stage, the student performs what he shall learn guided by the master, and in the third stage, the master retreats and the student has to manage on his own [Eidsmo, Kolås 2005].

2.2 Learning theories and computer game based learning

In computer game based learning there are a lot of different types of games that can be classified in different way. What learning theory that is relevant for different kinds of learning games, depends on different factors explained in 2.2.1.

2.2.1 Classification of game based learning

Before connecting the different learning theories to educational games, it is important to get an overview of the different kinds of games. There are many types of games used in education, ranging from text based quiz games to 3D based simulations of the real world. The intended use and purpose of the games also vary. A classification makes it easier to compare and analyze different games [Friis and Tollefsrud 2004].

The following list of classification areas will be described:

- Development method
- Age and level
- Game genres
- Type of business and institution
- Area of education
- Purpose
- Learning method

Development method

When developing an educational game, there are three main approaches:

- Custom made
- Modification
- Commercial off the shelf (COTS)

In the custom made approach most of the programming and design are done by the development team. Graphics, sound, game-engine etc. are all built from scratch. The educational content is also made by the developers. This requires skilled programmers, graphics artists, designers and teaching knowledge.

An alternative to do all the programming and graphic work, is to modify an existing game to fit an education purpose. Many games have modifying tools like level and map editors, and scenario builders that can be used to transform an entertainment game to an educational game. This approach requires no or little programming skill and only the educational content must be provided.

The simplest way is to buy an existing product (COTS), made especially for educational use. It can either come with educational content or have tools for adding the content. This approach is often cheaper than the others and can give better results, provided that there exists a product that fits the purpose.

Age and level

A learning game is usually aimed at a specific user group in terms of age and level. The students needs differ between different ages; students at higher levels require different methods than those of lower levels. This can be used to classify games and adding relevant structure. It is not worth comparing a game for teaching the alphabet to a five year old with a diagnostic game for medical students.

There is, however, no direct relationship between age and level of education. Age and level vary between countries and their education systems and at higher levels the age group gets wider for each level. In higher education like universities and colleges the level of a course can be described as:

- Basic
- Intermediate
- Advanced

A game for teaching the alphabet for a 6 year old can be classified basic, and a game for students teaching computer hardware design at university level can be classified as intermediate.

Game genres

There are eight main genres in computer games [Prensky 2001], which can be used to classify computer based learning games:

- Action games
- Adventure games
- Fighting games
- Puzzle games
- Role-playing games
- Simulation games
- Sports games
- Strategy games

A simple game may only belong to one genre, while a more complex game often belongs to or use elements from several genres. An action game can use elements from the adventure and puzzle game genre to enhance the game playing experience, but it will still be classified as an action game. For a game to be classified as belonging to more than one game genre it must have an extensive use of elements from the genres.

Type of business and institution

There are many different areas where learning takes place. The types of business or institutions to use for classification are:

- University
- College
- High school
- Primary school
- Private company
- Government
- Military

For example a game can be classified for use at a university while another is for use in a private company. The different institutions have different requirements to the learning because of difference in age and skills of the persons that are educated. There is a big difference in age and maturity between a pupil in first grade in primary school and a master student, and this has to be taken into consideration. This is an alternative to age or level as classification.

Area of education

Educational games usually focus on a single subject or area within a subject. Classifying games according to what area they teach can be useful for comparing games within the same area or subject. Here are some examples of areas:

- Mathematics
- Language
- Physics
- History

Purpose

Computer based learning games usually have a specific purpose and teach a certain type of content. It can replace an existing part of a course, act as an alternative learning resource or add something new. As very few computer game based learning games are intended to cover all aspects of learning in a course, traditional learning methods as books and lectures are still needed. Here is a list of content types for classification [Prensky 2001]:

- Facts
- Skills
- Reasoning
- Communication
- Creativity
- Procedures
- Systems

A game can for example be a way of practicing skills, learning historical facts or being creative.

Learning method

Every computer based learning game uses some kind of learning method to make the students learn the content. This can be used to classify games according to type of learning method. Here is a list of some of the common methods used in learning games:

- Practice and feedback
- Learning by doing
- Learning from mistakes
- Goal-oriented learning
- Task-based learning
- Role playing
- Question-led learning

A game can use a single method or a combination of several. For example a game can combine learning by doing, with goal-oriented learning.

2.2.2 Learning theories related to learning games

After having classified computer based learning games in different categories, it is time to relate different kinds of games to the different learning theories. Not all the classification categories are relevant to the different kinds of learning theories; I only describe the relations that are relevant. Method of development and age and level are not very relevant here, since none of them are specific to any particular theories.

Computer game based learning and behaviourism

Behaviourism describes learning through experiences, and this is most significant in adventure games, puzzle games, simulation games and strategy games, but also in other genres. In adventure games, the players are exploring the game world, and the players are acting differently depending on what is happening, where they go, and other choices they make. The players’ response to different game elements will change when they get experience and learn what to do to make progress in the game. For example, if a player makes a mistake, he will probably learn from that and not do the same mistake the next time he comes to the same point in the game. When the player completes a task, he gets rewards like points or he advances to another place or a higher level in the game. Another important factor is that the game makes the player able to repeat the same tasks over and over again.

Different types of learning theories are relevant to different areas of education. Some subjects are more suited for games that affect the behaviour than other games. But it is difficult to sort out which areas of education that are more related to behaviour than others.

Games are made for many different purposes, and different learning theories influence different purposes. For learning skills, the students have to train and repeat the same exercises many times. After repeating the same exercise in a game many times, the student will learn what to do, i.e. which response that is correct depending on the stimulus. Behaviourism is also relevant in other content types.

In the different learning methods, behaviourism influences especially learning by doing and learning from mistakes. In learning by doing, the student plays an active role and interacts with the computer while solving problems. The student's way to solve the problems will change and improve after having played the game for some time and learned how to advance and solve tasks in an effective way. The computer gives different stimuli depending on what the student does, and the student will learn how to do it in a correct and effective way.

Learning from mistakes is similar to learning by doing in many ways, but here the student gets feedback that tells him what went wrong. The tasks are repeated until they are done correctly. Trying to solve a task in the wrong way will definitively lead to a change in behaviour.

Computer game based learning and cognitivism

Cognitive learning can be understood as the opposite of behaviourism, and the mental processes are what we focus on. In strategy games, the player has to perceive and interpret the problems before he solves them. It is often necessary to make plans for later stages in the game, and a failure in an early stage can make problems later in the game. The mental processes during the learning are very significant. Cognitive learning theory is also relevant in adventure games, puzzle games, in some way role-playing games and simulation games.

Subjects that require reasoning and perception are related to the cognitive learning process. In mathematics, for instance, the student first perceives the problem, then reasons about it and interprets it, before he gets the solution. This is suited for games for subjects like mathematics, languages, and science subjects.

Different purposes are related to different learning theories. Learning content types like reasoning, procedures and systems are related to cognitive learning theory and the mental processes are very significant. Introducing new and unknown contents into the games supports cognitive learning.

Games that use learning methods including guidance and instructions support cognitive learning. Using practice and feedback, the game tells the player when he solved a task in the right or wrong ways, and the game can give instructions and hints to be able to complete the task. This approach is also possible in other learning methods, in varying extent.

Computer game based learning and constructivism

In constructive learning theory, the player is an active part in the learning process. This is significant in simulation games, where the player uses prior gained knowledge to construct the solution, and in this way making new knowledge. This is also possible in other game genres like puzzle games, where the player also has to use knowledge he has to make new knowledge.

Learning supporting constructivism can be used in different kinds of subjects and content types; therefore I will not mention which areas that are more suitable than others here.

Learning methods involving the students' responsibility for their own learning support constructive learning theory. Learning methods that can use problem based learning are relevant, and methods like practice and feedback and task-based learning can be problem based.

Computer game based learning and socio-constructivism

Computer-Supported Collaborative Learning (CSCL) focuses on collective learning processes and communication and emphasizes the aspects in the technology that support these kinds of processes. CSCL emphasizes collaboration between students that are on different locations, and this can be supported by multiplayer games. Fighting games, role playing games, strategy games and sports games are traditionally made in multiplayer mode, but it is also possible in other game genres.

Computer game based learning and situated learning

The goal in situated learning is to place the student in an environment that reflects surroundings where he is going to use his knowledge in a job situation. Simulation games are very useful for placing the student in a situation that prepares him for his future job. The games can be made as realistic as possible, i.e. medical games where medical students have to make diagnosis on patients in the game, and the patients might die if they make mistakes.

Situated learning supports different purposes, but especially learning skills, as the learning situation in the game should be as similar to the real situation as possible.

The best learning methods in situated learning are learning by doing and practice and feedback.

Conclusion

The different learning theories differ in many ways, and are important in various contexts. It is not possible to say which theory is better than another. Usually more than one theory are taken into consideration in an educational situation, both in traditional learning situations and in computer game based learning. An educational game will usually contain many types of learning elements which will often relate to different learning theories. Behaviourism and situated learning will often be relevant in a game situation, for example in a game simulating a job situation.

3. Hypermedia in education and games

When navigating and finding information on a website or in a program, it is important that the information is organized in an appropriate and efficient way. If the data is organized in a smart and efficient way, a lot of computational power and time can be saved in processing the data. This is very important in management of big amounts of data like in programs used in education which often contain much data of different kinds as text, images and video.

3.1 Hypermedia

An ordinary document or book is read in a linear way from top to bottom. Hypertext is a document containing links in the text to jump to preferred subjects in the document. The prefix "hyper" is used to indicate the differences from traditional linear text. The user in a hypertext document is able to choose his own path through the document.

Hypermedia is often called the hypertext of modern times. The only difference is that pictures, video and sound are added in the same way as text. There are many definitions:

"Hypertext is text that is not linearly limited."

"Hypermedia and Hypertext tend to be used loosely in place of each other. Media other than text typically include graphics, sound and video."

Word Wide Web Consortium, "Hypertext Systems", April 1995.

"Hypertext: ...is a web of possibilities, a web of reading experiences. ...Hypertext is the language of exploration and discovery."

Charles Deemer, "What is Hypertext", 1994

Here is a list of what characterize hypermedia:

- Hypermedia shall be able to use for both writing and reading information.
- The information is organized in a non-sequential structure. This makes it possible to follow the information via several alternative paths.
- The information can follow natural associations from one unit of information to another.
- The information can be structured in a hierarchy.
- The information, or parts of it, shall be shareable between multiple users.
- The information is stored in a database. This means a collection of information that can be gained in some way. It is not an assumption that the information is organized in a specific way or sequence.

[Lowe, Hall 1999].

3.1.1 Data, information, knowledge and wisdom

Hypermedia is about information. The concept "information" is quite vague, but I will explain information in the context of hypermedia:

A hyper system can be presented as a pyramid with data at the bottom, followed by information and knowledge on the top [Lowe, Hall 1999]. But I will add another step on the top; wisdom. The reason for placing wisdom over knowledge is that wisdom is the way to use the knowledge.

- Wisdom: The way the knowledge is used.
- Knowledge: A personal collection of information that is integrated in a way that it can be used for further understanding and analysis of new information.
- Information: Translation of data within a context. This translation is resolved from former achieved knowledge, and from the environment.
- Data: A collection of symbols and text.

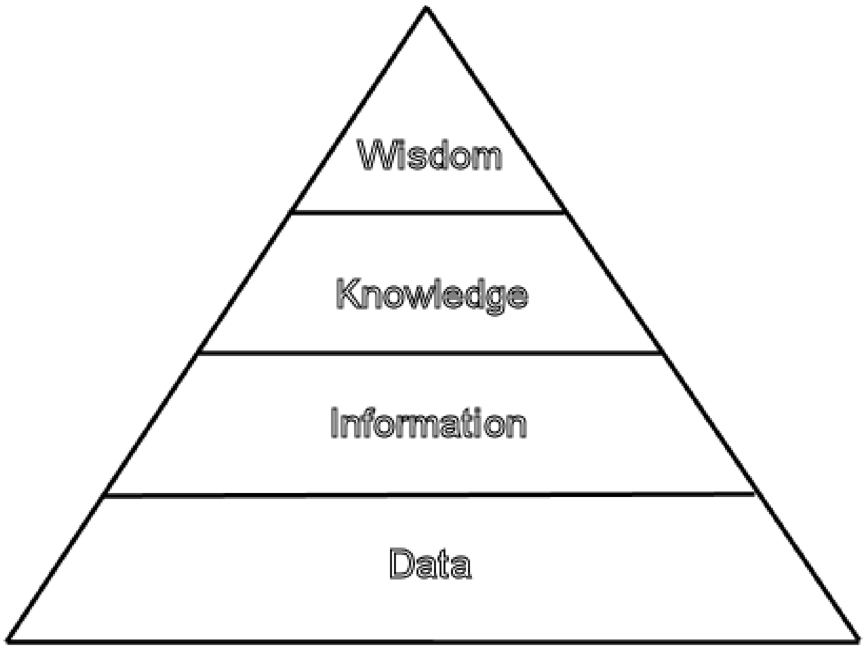


Figure 1 Knowledge hierarchy

From this pyramid, Lowe makes the following conclusions:

- Data standing alone is worth nothing on its own in a hypermedia system. This makes factors preventing an efficient presentation of the data having a negative effect on the users' possibility to interpret the data.
- Data is understood in the context they are presented.
- Information are understood from the environment they are presented in.
- The data should follow a well organized structure to make it easy to extract any useful knowledge from the data [Lowe, Hall 1999].

3.1.2 The hypermedia base

The hypermedia base mainly consists of hyper documents, nodes and links. The hypermedia base is the information base that the hypermedia system is working on top of [Warholm 2000].

Nodes are individual information parts. Nodes can be pictures, sound, text, video etc. It is usual to make these parts so small that they can not be divided into smaller parts without losing their original purpose. They should be made in a way that no more information is needed to understand the contents of the node. The whole node should also fit in one screen shot.

Links bind the nodes together. This makes it possible to make a network of information. There are two types of links: reference links and structural links. Reference links are crossreferences that are typical for hypermedia, and links between the contents of the hypermedia base. An example is a link that gives more information about a subject. Structural links show the structure of the base. Typical structural links are links to the start page and links to pages in the level below the existing page.

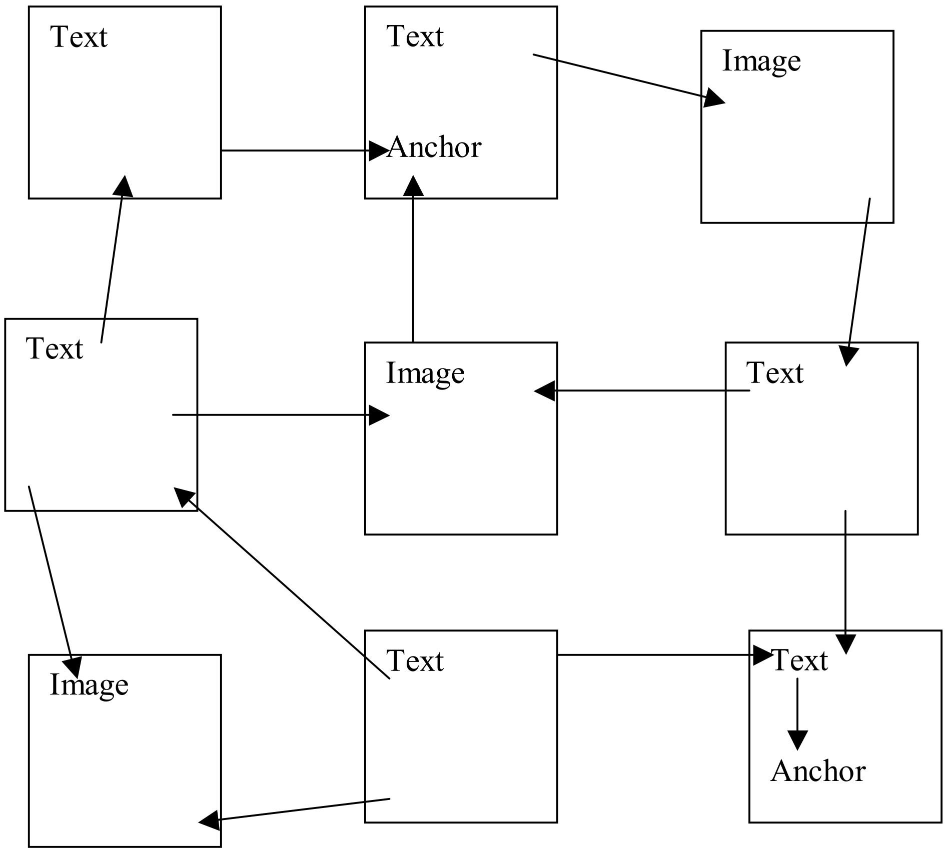


Figure 2 Example on a hyper document

The **anchor** is the point or the text that represents the link. Every link has two anchors; one pointing to and one pointing from. Anchors are usually represented by another colour than the surrounding contents. Links can be either in state as visited or not-visited. The difference can be seen by changing the anchor's colour after it has been visited. Most browsers mark not-visited anchors as blue and visited anchors as purple.

Example on an anchor: "The next paragraph is about paths".

Paths are a collection of links that connect the nodes in a sequential way. Paths are used to help the user to navigate. The user follows the path through the hypermedia system to achieve a specific goal.

The **hyper document** is a collection of nodes and links. The hyper document consists of an independent and bounded subject. The hyper documents are usually separate files, and the most used format is HTML.

The **web** is an abstraction of the links that exist between the hyper documents. The web is a collection of links that can be used to navigate in the hypermedia base. This abstraction makes it possible to have multiple webs on the same hypermedia base, and this is useful in multi-user systems.

Metadata can belong to anchors, links, nodes, and the hyper document, depending on the hypermedia system. The purpose of metadata is to classify and describe the object belonging to it. Usual services to perform on metadata are:

- Support search on information
- Indexing of the hypermedia base
- Give extra information in linking between nodes

[Albinussen 2003]

3.1.3 Information structures

Before developing a hypermedia system, it is important to know how to present the hypermedia base through an information structure. The hypermedia base consists of a network of nodes and links. These networks make it possible to utilize associations between concepts. We are trying to make explicit the knowledge representations which are inherent within the information. Since we only want to communicate certain ideas, we will only make explicit certain subsets of the knowledge representation. Information related to one concept will relate differently to other information depending upon the overall context.

How the information is structured is deciding what services are available, and if the service is functional or not. The structure is often decided from what types of links we will use; a hierarchic is fitted to show structural links, while a network structure is suited to show reference links. There are four main ways to classify information structures: linear, hierarchic, network, and matrix, see Figure 3 [Lowe, Hall 1999].

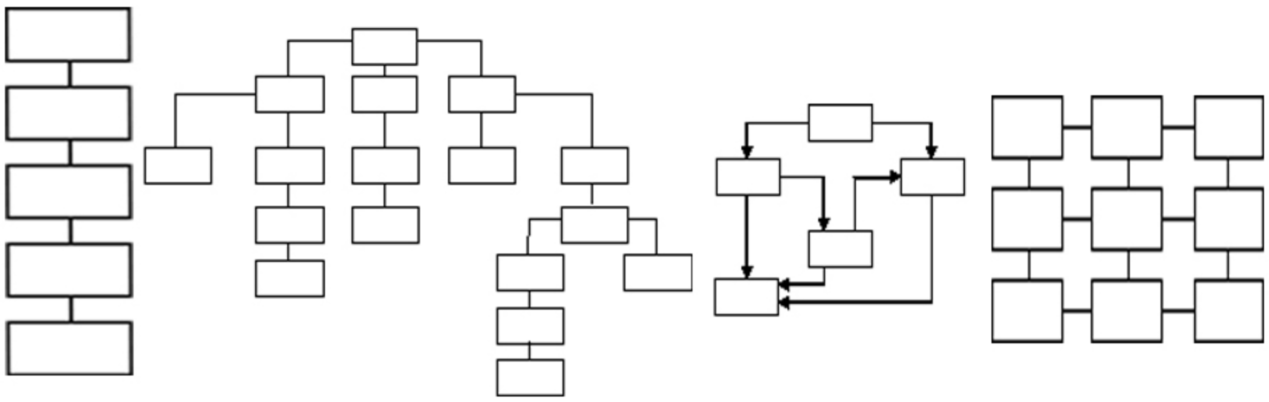


Figure 3 Information structures - from left to right: linear, hierarchical, network and matrix

Linear structures

Linear structures can be used to retain the sequential structure of a paper document, such as a book. It makes good sense to retain the linear structure of the original document. If not, some information can be lost from the original context. The author might present the understanding of a concept in chapter 1 and present the concept once again in chapter 2. If the reader reads chapter 2 before chapter 1, misunderstandings might occur.

A linear structure is predictable and easy to understand. It is well fitted for training material, where it is possible to assure that the contents are gradually getting more difficult [Lowe, Hall 1999].

Hierarchical structure

In a hierarchical structure, all nodes and links are placed in different levels. From the top node it is possible to reach all other nodes. Books are usually divided into chapters and subchapters, and this structure is easy to replicate within a hypermedia database through the use of hierarchical links. The structure is similar to the use of directories in Windows Explorer, where the directories are organized in a hierarchy. A hierarchical structure can be useful in an educational purpose; the information can be more detailed the deeper down in the structure you get [Lowe, Hall 1999].

Network structure

The network structure is most similar to the fundamental ideas behind hypermedia. The structure is mainly based on reference links and is non-sequential, and the common or related concepts are bound in the information space. There are not many restrictions on the use of this structure. The idea is to imitate associative thinking and streaming ideas. The ability to use these links and the resulting network is one of the main advances of hypermedia applications [Lowe, Hall 1999].

Matrix structure

The matrix structure is well fitted for multidimensional categorizing of information. For use in the industry the horizontal axis can represent mechanical problems and the vertical axis can represent information about symptoms, solutions and procedures. This results in one single cell that represents the solution of one specific problem. The matrix structure is often used on the top level in a hyper structure and is well fitted for indexing for lower levels [Lowe, Hall 1999].

3.2 Hypermedia in education

The information age has increased the need for available up to date information. This has also influenced the learning material. Printed material as books last for shorter and shorter time and this has increased the use of computer networks as a teaching area. The need for the last available information and data all the time makes the use of computer based learning very relevant and useful [Warholm 2000].

In industry many tasks have been automated during the last years as a result of new technology. This has created a need for re-education of the employees. New competence is required to perform new tasks, and this is not always possible to be dedicated at the place where people live. “Lifelong learning” is a concept that describes the need of continuous upgrading of skills. As a result of this need, a new group of students is growing; long-distance students [Warholm 2000].

To ensure an educational, reasonable situation of education for this group of students, new methods of education are necessary. New methods of teaching require a restructuring of the way information is communicated and new ways of interaction between both teacher and students, and between the students. There are many big challenges in a new way of thinking of the structuring of teaching material and how to present it in a way that makes it possible to acquire.

Hypertext and hypermedia systems are widely used in educational environments. The active participation of the readers and authors is one of the main advantages of hypertext systems. Cognitive psychology is somewhat supporting this assumption. Deep processing and elaboration usually lead to better understanding than information analysis at higher levels of processing. It is important to make the students do things and take active part in the learning process. [Pohl, Prenner, Purgathofer 2006].

3.2.1 Advantages

Using hypermedia makes the contents more exciting and can hold the student interested for a long time. The use of images and sound along with the text stimulates multiple senses at the same time and expands the channels to the mind.

Use of links can connect ideas from different media sources, for example a sound and an image. The students are also able to navigate through the web structures in their own and individual way. In this way, they build their own unique mental structures based on their exploration. Students and teachers are also able to create their own hypermedia files, and most software supports collaboration [Smaldino et al. 2005].

3.2.2 Limitations

The students can get lost when using hypermedia programs if the clues to where they are in the material are limited or misleading. If the students are used to a more structured guidance they might get frustrated. All students are not able to decide how much information they have to explore.

Some programs can be one way presentations of information without opportunities for interaction with the program, and this might get boring. Advanced programs may be difficult to use, especially for student production where use of a scripting language is required. Hypermedia systems have a nonlinear structure that differs in many ways from traditional learning material as books, and may require much more time to learn to use both for students and teachers.

3.2.3 Long-distance students – off-campus education

There is an increasing need for post-qualifying education at the same location as people work and live. The traditional classroom education shows signs of out of date curriculum because of the use of printed teaching materiel, and this creates the demand for new methods of education.

Assumptions

The possibility for use of off-campus education requires large restructuring in both contents and sometimes presentation of the material. If the education also is asynchronous, the contents have to be self-instructing and ensure the progression according to the measurability of the student's upgrading of skills.

The teachers and authors of the teaching material have to acquire new knowledge and techniques about how to use a dynamic structure in hypermedia according to the traditional linear structure used in books.

Off-campus education is not a new concept, and has existed in Norway since 1914. Correspondence courses have been used to communicate between students and teachers. In the last years, the letters have been replaced by e-mail for the students who have access to the net.

However, this form of education has drawbacks; the students are left on their own and the learning material can contain old information that is not up to date. The follow-up of the progression in the studies is left alone to the students, and this can affect the motivation.

Another important assumption is accessibility to computers and a network. Not all have an internet-connection at home, but during the last years, the development of broadband has increased and many parts of Norway have access to fast internet connection [Warholm 2000].

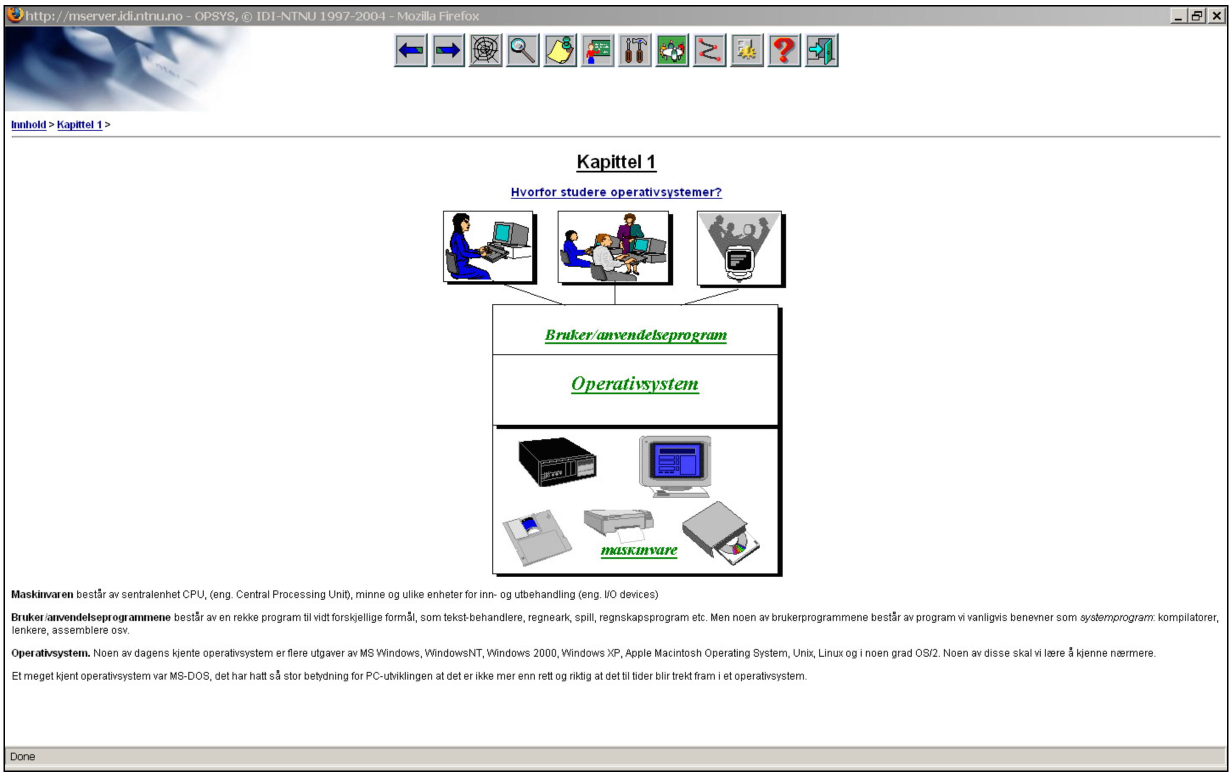
3.2.4 OPSYS

OPSYS is a hypermedia system for education used in the course IT2202 Operating systems at NTNU (Norwegian University of Science and Technology). It is used to teach the students the curriculum in the subject in a different way than only traditional lectures. In OPSYS, the students can navigate through the different chapters of the curriculum and jump to different themes in the sequence and tempo after own choices.

The system consists of four different media types: text, images, animation and video. The text describes the different subjects and objects, and there are pictures of the objects and animation that show how objects interact.

An overview of the system is given in Figure 4. The menu on the top is used for navigating in the system, as going back and forward between pages, a search function, index, a list of given paths in the system, help and other functions.

The students can choose between going through the lectures chapter by chapter, or traversing the curriculum after their own ideas and wishes. There are paths for different themes that the students can choose to follow if they want a structured and guided way to learn a given theme of the curriculum [it2202 2006].


http://mserver.idi.ntnu.no - OPSYS, © IDI-NTNU 1997-2004 - Mozilla Firefox
Innhold > Kapittel 1 >
Kapittel 1
Hvorfor studere operativsystemer?
Bruker/anvendelseprogram
Operativsystem
maskinvare
Done

Maskinvaren består av sentralenhet CPU, (eng. Central Processing Unit), minne og ulike enheter for inn- og utbehandling (eng. I/O devices)

Bruker/anvendelseprogrammene består av en rekke program til vidt forskjellige formål, som tekst-behandlere, regneark, spill, regnskapsprogram etc. Men noen av brukerprogrammene består av program vi vanligvis benevner som *systemprogram*: kompilatorer, lenkere, assemblere osv.

Operativsystem. Noen av dagens kjente operativsystem er flere utgaver av MS Windows, WindowsNT, Windows 2000, Windows XP, Apple Macintosh Operating System, Unix, Linux og i noen grad OS/2. Noen av disse skal vi lære å kjenne nærmere.

Et meget kjent operativsystem var MS-DOS, det har hatt så stor betydning for PC-utviklingen at det er ikke mer enn rett og riktig at det til tider blir trekt fram i et operativsystem.

Figure 4 Screenshot from OPSYS [it2202 2006]

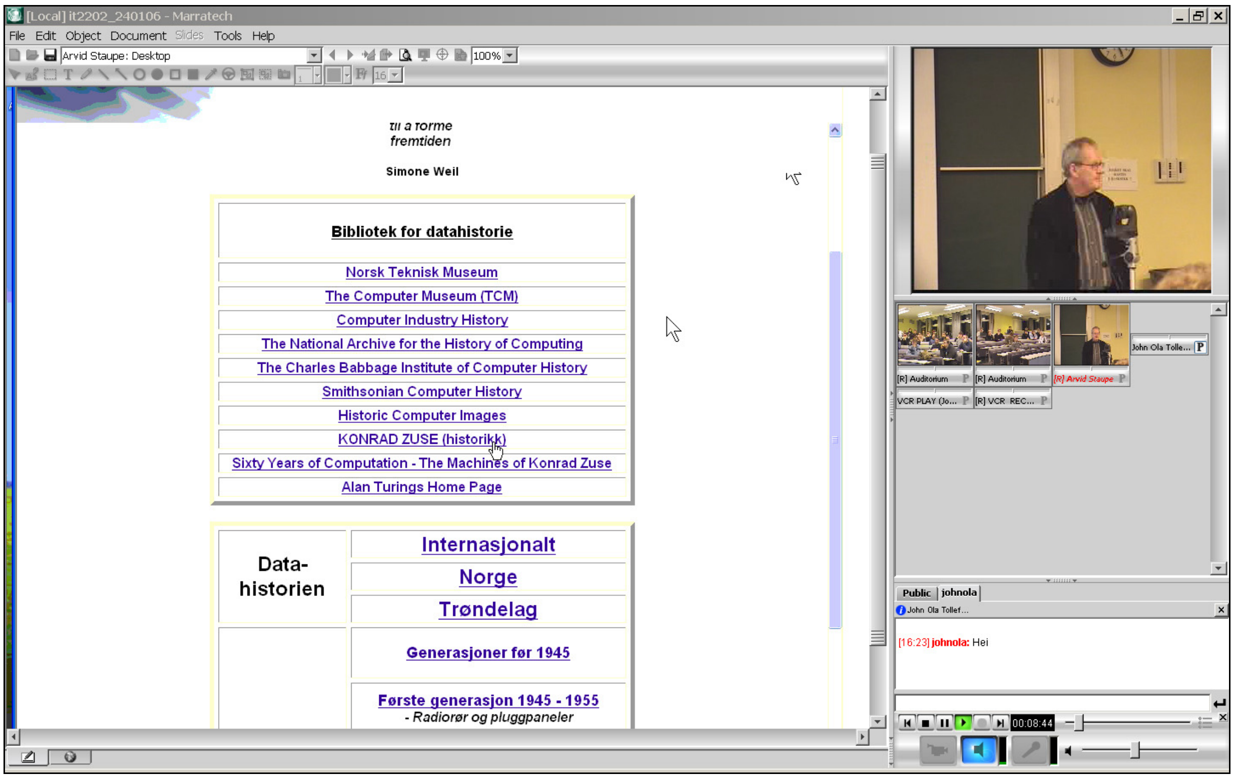
3.2.5 Mobile instruction

In IT2202, the lectures are broadcasted live on video on the web pages of the subject. This makes it possible for students to follow the education independent of geographical location. And the students that follow the lecture from another place than the lecture room are not just passive participants. If they have a webcam and a headset, they can ask questions and take active part in the lectures in the same way as students that actually are in the lecture room.

Figure 5 shows the user interface of the Marratech system used in the subject. The slide used by the teacher is in the left window to the left. The students following the lectures are able to click on the links there, and in that way have a higher degree of interactivity than if they are sitting in the auditorium. The video from the lecture is in the upper right window, and the users are able to switch between different cameras. Figure 6 shows the students on the lecture in the upper right window.

The lectures are also saved and are available from the web pages of the subject. This gives a lot more freedom for the students than traditional education; the students can choose to attend a lecture at any time and place.

The video system in IT2202 opens a whole new world of education. Off-campus education has, as mentioned in 3.2.3, been dominated by using websites with updated information, and e-mail correspondence between students and teachers. The possibility to take part in the lecture and ask questions and make comments to the teacher gives the students the same feeling as if they actually were in the lecture room. This technology makes distance education as good as education on campus. The only difference is the lack of face-to-face conversations, but both persons using webcam and headset is almost as good as a “real” conversation.



til å forme
fremtiden

Simone Weil

Bibliotek for datahistorie

Norsk Teknisk Museum
The Computer Museum (TCM)
Computer Industry History
The National Archive for the History of Computing
The Charles Babbage Institute of Computer History
Smithsonian Computer History
Historic Computer Images
KONRAD ZUSE (historikk)
Sixty Years of Computation - The Machines of Konrad Zuse
Alan Turings Home Page

Data-historien

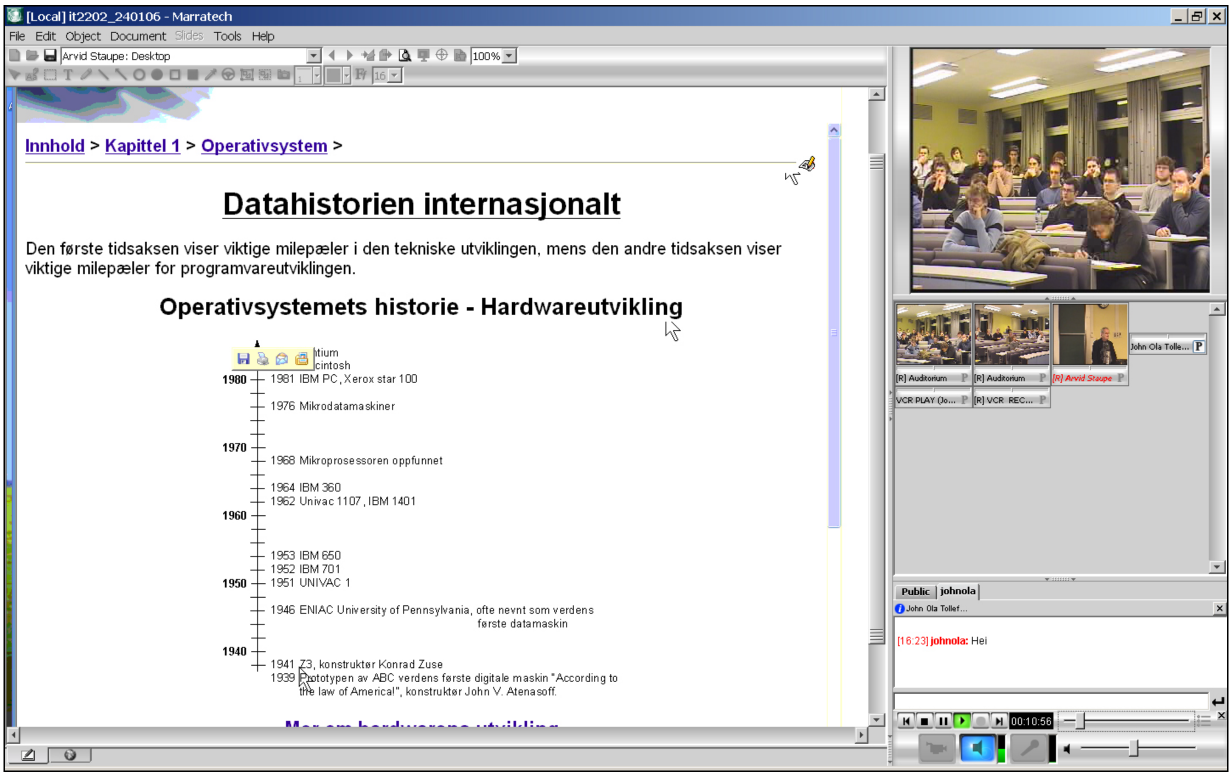
Internasjonalt
Norge
Trøndelag

Generasjoner før 1945

Første generasjon 1945 - 1955
- Radiorør og pluggpaneler

[16:23] johnola: Hei

Figure 5 Screenshot from the video system, showing the teacher Arvid Staupe in the upper right window [it2202 2006]



Innhold > Kapittel 1 > Operativsystem >

Datahistorien internasjonalt

Den første tidsaksen viser viktige milepæler i den tekniske utviklingen, mens den andre tidsaksen viser viktige milepæler for programvareutviklingen.

Operativsystemets historie - Hardwareutvikling

```
        tium
        cintosh
1980 — 1981 IBM PC, Xerox star 100
       1976 Mikrodatamaskiner
1970
       1968 Mikroprosessoren oppfunnet
       1964 IBM 360
       1962 Univac 1107, IBM 1401
1960
       1953 IBM 650
       1952 IBM 701
1950 — 1951 UNIVAC 1
       1946 ENIAC University of Pennsylvania, ofte nevnt som verdens
                                         første datamaskin
1940
       1941 Z3, konstruktør Konrad Zuse
       1939 Prototypen av ABC verdens første digitale maskin "According to
            the law of America!", konstruktør John V. Atenasoff.
```

[16:23] johnola: Hei

Figure 6 Screenshot from the video system, showing the students in the auditorium in the upper right window [it2202 2006]

3.3 Hypermedia in computer game based learning

Hypermedia structures are used in many different computer games. In e.g. simulation and strategy games, the underlying structure is usually many different nodes linked together. There are huge amounts of information that has to be organized in a logical and intelligent way. It is also necessary to make the information needed to advance in the game accessible at the right time and spot in the game for the players, if not they might get annoyed and stop playing the game.

Different kinds of educational games contain different amounts and categories of information, and this makes different requirement for how to organize it. A fighting game where two players are boxing will typically contain a very small amount of information compared to a strategy game where the player is conquering the world. I have illustrated a possible use of hypermedia in a game by an example of a game idea I invented.

3.3.1 Use of Hypermedia in a hypothetical education game

As an example, I will choose a strategic game for learning history. The game will require large amounts of information, and this have to be structured in a reasonable way to make the game easy to use and understand.

The plot of the game is to conquer the world through solving tasks and puzzles in history. The tasks will be related to the continent and countries in which the player is playing at the moment. The information required to solve the tasks and answer on the given questions, is given during the game as pop up windows popping up in fixed spots in the game. All the information gathered during the game, is saved and organized in a "history book" where the player can repeat the information and look up needed information. The book is expanding every time the player is gaining new information.

The information has to be organized in a reasonable and efficient way, and a hypermedia base is a good way to do this. The player shall be able to reach the information he likes in a simple and intuitive way, and he shall be able to do this at any time and place in the game. A hypermedia base is a good way to organize the information in this game. The nodes in the base are pieces of information as a task to be solved or an information page, and these are linked together in a reasonable structure, as a hierarchy or as a network, depending on design choices.

A strategy game of this kind will typically consist of different data and media types as text, images, video, sound and animations. The benefit of using a hypermedia structure in this kind of game is that it might save time in processing data when playing the game and a fast game is more motivating and fun than a slow game. Another benefit of such structure is that updating the game, such as adding or deleting data, is easy and can be done without affecting another part of the game. A hypermedia system is compounded of different modules that can be changed independently. This gives such a system a high degree of modifiability, and this can save a lot of time and money for the company that develops games.

The partition of the games into modules makes it easier to design and implement the game. The development team can be divided and work on different parts independently, and the different groups can focus on their expert areas. This is a big architectural benefit, and is cost-effective compared to other architectural solutions.

4. Requirements for educational games and existing games used in education

There exist many different computer games for educational purpose today. I have chosen to focus on higher levels of education, and this chapter contains an overview of computer games used in higher education today.

Educational computer games have to fulfil different requirements to be both a good game and have a learning effect, if only one of them is fulfilled, the game will not be of decent quality.

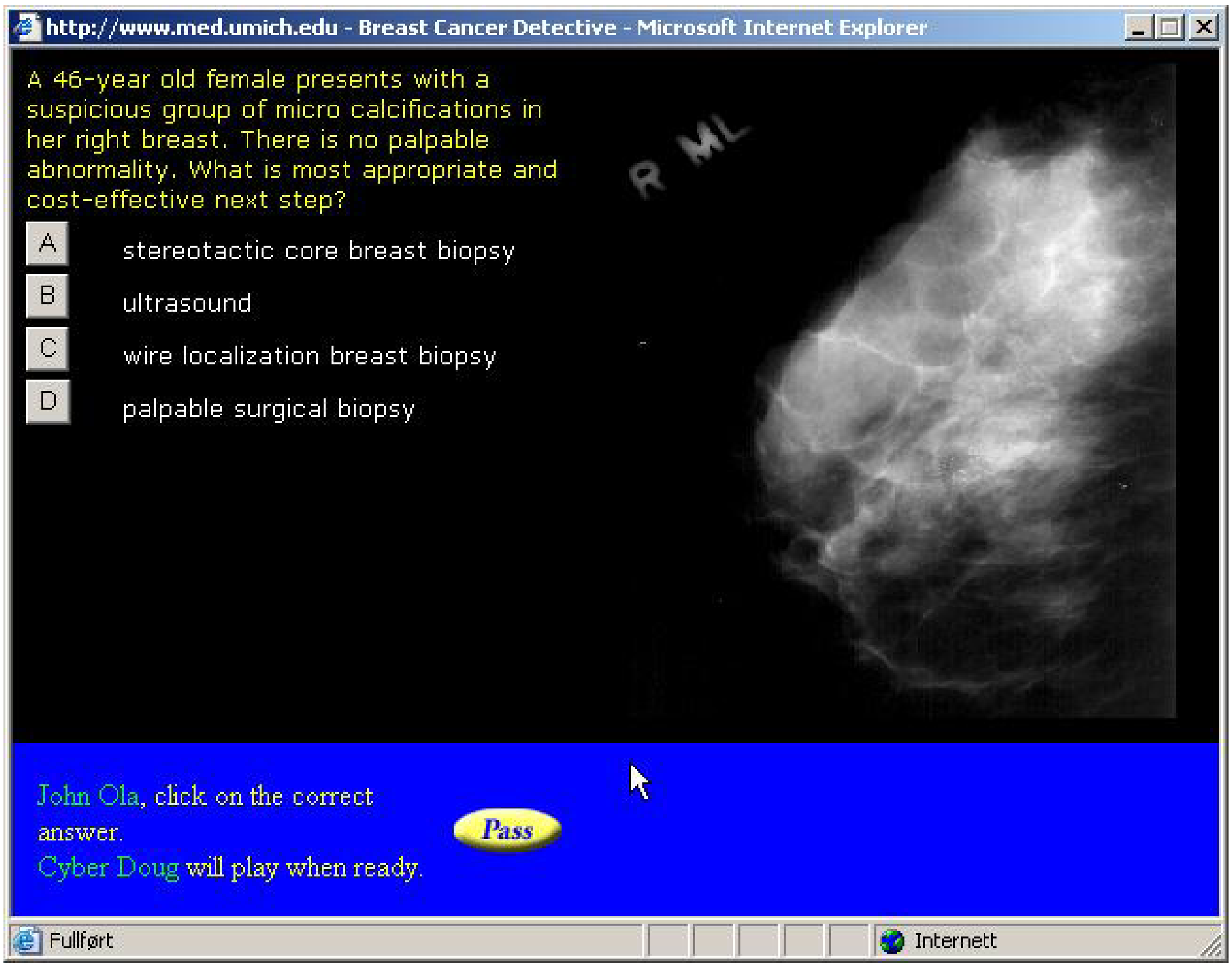
4.1 Educational computer games today

So far, computer games have mostly been used for entertainment, but during the last years, they have entered the school system, mainly on lower age levels. Computer games play a very important role in the life of most youth. They spend vast amounts of time to improve their skills in these games. This has made it worth to investigate how games can be used in education. The following section gives some examples on computer games used in higher education today.

Breast imaging game

In the University of Michigan School of Medicine, the 4th year medical students have used an interactive computer tool for teaching breast imaging. The game is used in radiology elective course, and the traditional teaching was lectures with PowerPoint handouts. With the game, it is able to visualize a breast and that makes it easier to simulate tumours and other diseases [Roubidoux et al. 2002].

To determine content for the game, six faculty members in breast imaging were asked to list three principles of breast imaging that are essential to a 4th year medical student. That resulted in a list of 10 learning objectives. To determine what the students already knew about the subject, a quiz derived from these objectives were given to a group of 16 students during their first week of a radiology elective course.
The game was then designed. 16 case scenarios with multiple choice questions were made based on the 10 objectives. The game was designed to be played by two students or one student and one cyber player. There were programmed two cyber players, one who answered almost all questions right and one who almost only answered wrong.
The students, who played the game, were asked to answer a survey. It contained ten learning objectives covered in the game, and the mean score on the test was 45% right in a range from 13% to 67%. These results indicated that further instruction in these objectives was necessary.


http://www.med.umich.edu - Breast Cancer Detective - Microsoft Internet Explorer

A 46-year old female presents with a suspicious group of micro calcifications in her right breast. There is no palpable abnormality. What is most appropriate and cost-effective next step?

A stereotactic core breast biopsy

B ultrasound

C wire localization breast biopsy

D palpable surgical biopsy

John Ola, click on the correct answer.
Cyber Doug will play when ready.

Figure 7 Graphical User Interface from the Breast imaging game [http://www.med.umich.edu/lrc/breastcancerdetective/]

Game-based tool for learning surgical management algorithms

At the Department of Surgery at the MCP/Hahnemann University in Philadelphia, there has been used a computer-assisted board game as a tool for medical education. This game was entirely transferred to the computer to make it more appealing to the students. The game can simulate the work-up of any clinical problem. The first module of the game is a breast game that simulates evaluation and management of patients with breast problems [Mann et al. 2001].

The game can be played on all computers, which gives it great accessibility. An animated, speaking, virtual facilitator moves around the screen and verbally instructs the players on their options, rules in the game and informs about the game progress.

The game takes about 45 minutes to play. Each player, or team, manages four randomly assigned players, giving a total of eight patients with eight different clinical problems. The

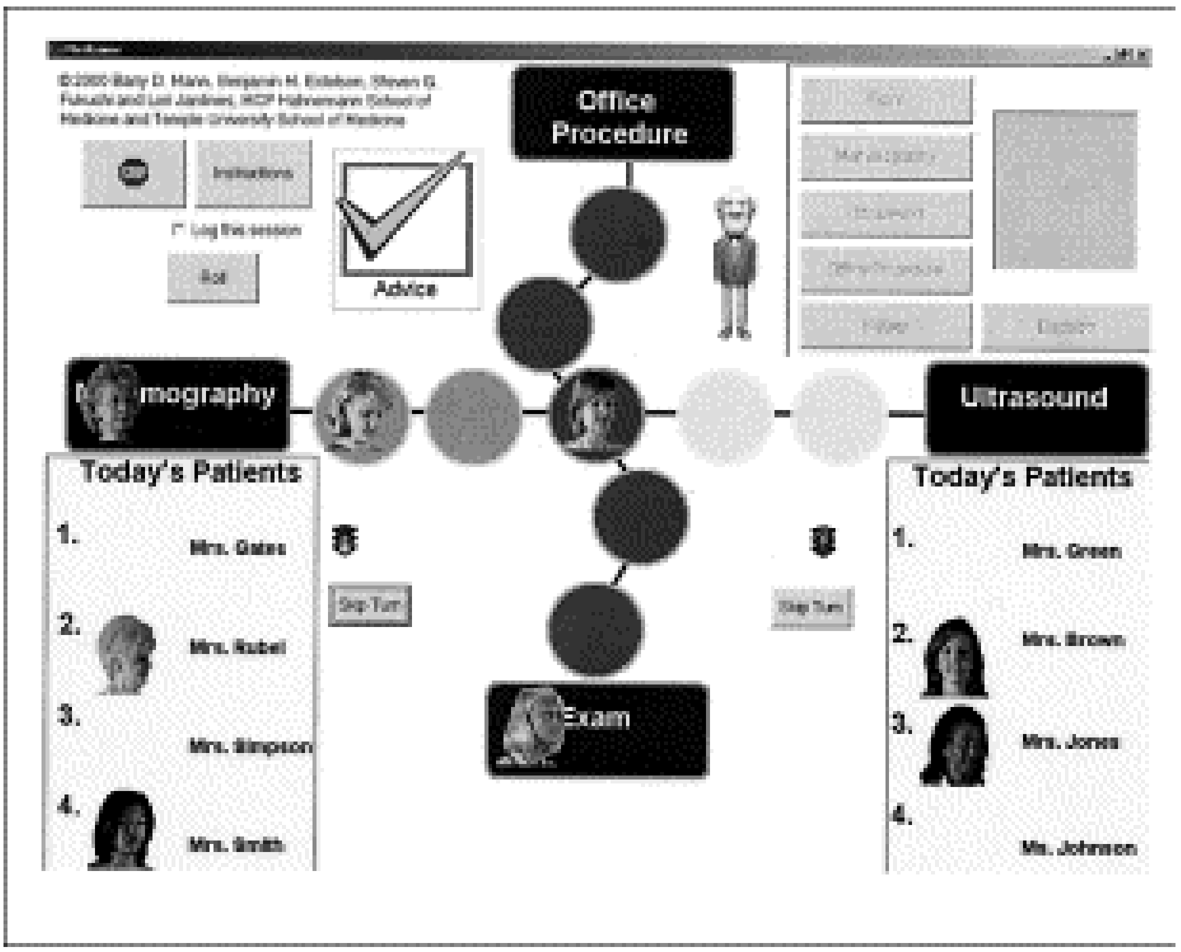


Figure 9 Screenshot from the surgical management game [Mann et al. 2001] Screenshot from the surgical management game [Mann et al. 2001]

figure to the left shows the main screen display. The player rolls an imaginary die and moves the patients around the board in an order of his own choice. In the breast module, the player can send the patient to physical examination, mammography, ultra-sound, or office procedures. Actual radiographs, sonograms, pathology and cytology images, and 3-D physical examination findings are displayed as the players use successive turns and rolls the die to move their patients through various diagnostic options. Each turn, the player can choose to introduce a new patient onto the board, or they can continue to advance a previously introduced patient.

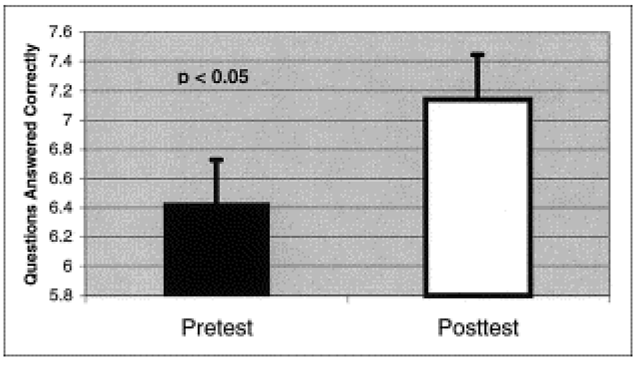


Figure 8 Pregame versus postgame scores on the test of knowledge about surgical management that shows that the game had a positive effect [Mann et al. 2001]

When a player feels he has enough information about a patient, he can choose to make a definitive management definition. He gets five potential decisions to choose from, presented and explained by the facilitator. If the definition is correct, the patient is removed from the board. If it is wrong, the player gets an explanation and looses his turn. The player that gives all four patients successful management wins the game. The “race” makes the players be aware of what they do, wrong choices result in a loss. There are also strategic elements, such as space limitations (the examination/procedure rooms have a maximum capacity of two patients at the same time). One player can effectively block specific diagnostic options from the others through strategic placement of their players on the board.

A group of 33 students tested the game, in various team sizes and settings. With large groups, eight or more, the game was played in an auditorium with the game projected on a screen. Observations suggest that the game has potential as a tool for both self-directed student learning and interactive lectures. All the players got a test with 10 true/false questions on breast problem management both before and after playing the game. Figure 3 shows that the results improved after playing the game. A survey showed that the students feel that they learn more than in traditional teaching and that it is appealing.

FINESSE (Finance Education in a Scalable Software Environment)

At the University of Dundee, Scotland, accountancy staff offer a final year honours module called “Security and Portfolio Management”. They have developed a Web-based game that requires groups of students to manage a portfolio of equities of a notional value of £100 million. Each team can spend their money on the London stock market and the Alternative Investment Market (AIM). The game introduces a level of realism into the analysis by incorporating dividend income and capital gains, by including transaction costs and by using real-time share price data subject to a 20 min time-lag that are updated on a continuous basis. The real-time financial data are provided by two sources: DataStream and UpData. DataStream supplies daily batched data whereas UpData provides price changes every 20 min [Michaelson et al. 2000].

A student can buy and sell shares within their group portfolio, can investigate the effects of buying or selling shares on their portfolio before committing a transaction, can investigate current state of the market and can follow links to appropriate course outlines and relevant company histories available via the Web. A Supervisor, typically a tutor or staff member, can determine the range of shares available to students, can look at particular portfolios and transaction histories, and can also monitor the use of course material and external links. This approach allows flexibility for the students in the ways they form a dynamic focus for the aims and objects the courses involved [Power et al. 1998].

A questionnaire was given to the students before the course was taken. The results indicated that the students enjoyed using computers and did not find it difficult to use the Web. They seemed positive about the potential of group work to improve many of their skills. Another questionnaire was given after the course and statements about enjoying the portfolio game, finding FINESSE easy to use and improving a student’s ability to work as part of a group all had high average scores. But face-to-face meetings had higher scores than e-mail facility in terms of helping students with group work. The students’ familiarity with financial concepts and financial terms had improved after the course and the game [Michaelson et al. 2000].

Revolution

The Games To Teach project which is now a part of The Education Arcade [Education Arcade 2004] started is a cooperative project between Microsoft iCampus and the Comparative Media Studies department at the Massachusetts Institute of Technology. They developed a set of 15 conceptual frameworks for computer based learning games, a subset of these were selected for prototyping. Revolution is one of these prototypes. It is a massively multiplayer online role playing game, set in the 18th century USA right before the revolution.

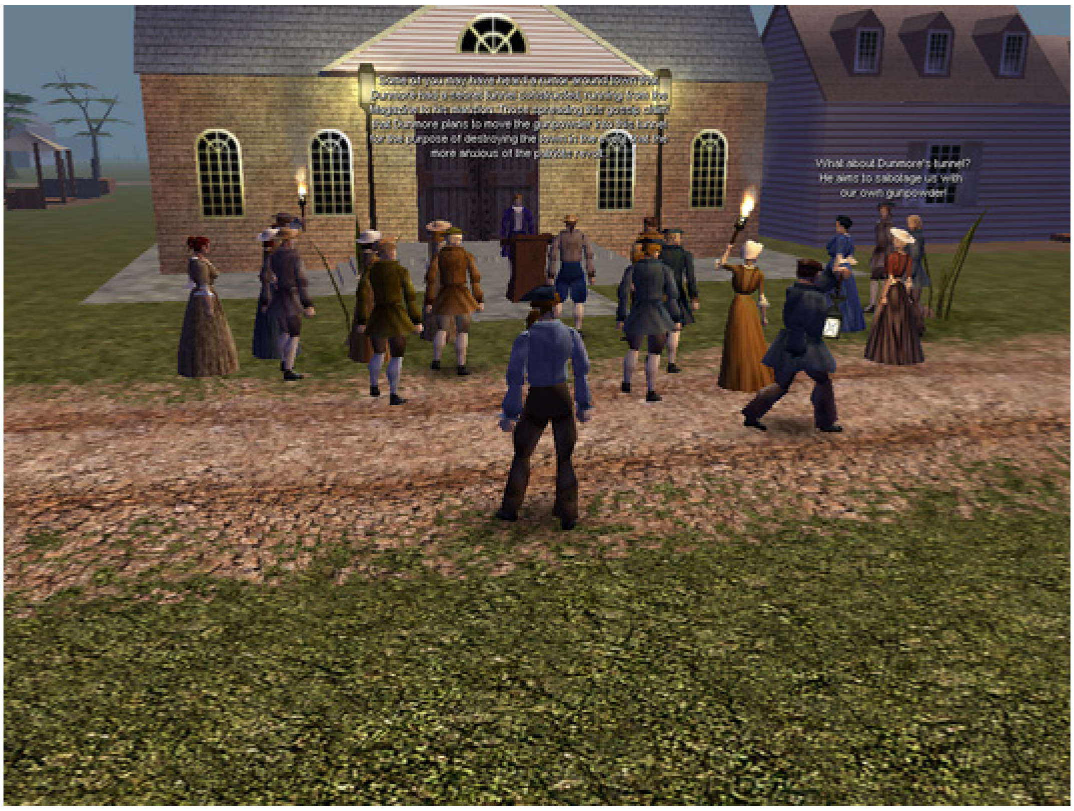

Some of you may have heard a rumor around town that Dunmore has a secret tunnel constructed, running from the Magazine to his mansion. Those spreading this gossip claim that Dunmore plans to move the gunpowder into this tunnel for the purpose of destroying the town in [illegible] the more anxious of the patriots revolt.
What about Dunmore's tunnel? He aims to sabotage us with our own gunpowder!

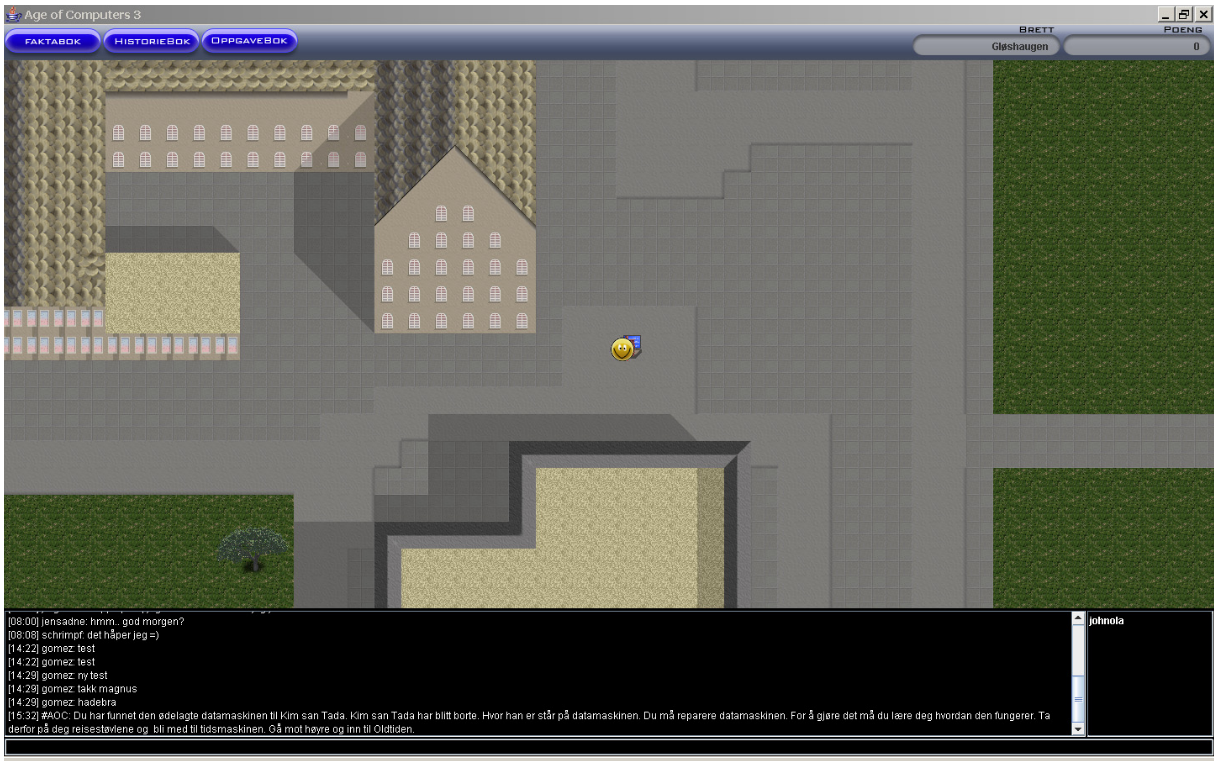
Figure 10 A screenshot from Revolutions [Education Arcade 2004]

The game is based around a multiplayer environment where each player takes on the role of an inhabitant in a small town in Virginia. It can either be played as a classroom activity or over the Internet. There is a strong narrative component in the game, telling the story of the revolution from the view of the different roles. The game is divided into stages which can be used as classroom sessions, that lead up to the outbreak of revolution. In each stage the players act their role and go through a series of events while interacting with other players. After a completed session the players can discuss what they just went through.
The goal of the game is to teach students historical facts and how different aspects of the characters' lives affected the choices they had. Things like class, race, trade, technology, violence and communication. Students will also learn how actions on a local level affect national events and how national events affect people locally.

Age of Computers

Age of Computers (AoC) is a computer game that is used in the subject TDT4160 Computer Fundamentals at NTNU. The game replaces traditional mandatory exercises that are usual in most subjects in the Master of Science education at NTNU. The students are answering questions in AoC instead of delivering a paper or mail with exercises. They get points for their progression in the game, and have to reach a certain goal before they can attend the exam.

The first version of Age of Computers was used in TDT4160 during the autumn semester 2003. AoC II was developed during the winter and summer of 2004 and used in the autumn semester 2004, and version III was launched during the autumn 2005 and was used in the autumn semester 2005 [Natvig, Line 2004].



```
[08:00] jensadne: hmm.. god morgen?
[08:08] schrimpf: det håper jeg =)
[14:22] gomez: test
[14:22] gomez: test
[14:29] gomez: ny test
[14:29] gomez: takk magnus
[14:29] gomez: hadebra
[15:32] #AOC: Du har funnet den ødelagte datamaskinen til Kim san Tada. Kim san Tada har blitt borte. Hvor han er står på datamaskinen. Du må reparere datamaskinen. For å gjøre det må du lære deg hvordan den fungerer. Ta
derfor på deg reisestøvlene og  bli med til tidsmaskinen. Gå mot høyre og inn til Oldtiden.
```

Figure 11 Graphical User Interface in Age of Computers

The game is a board-based puzzle game. **Error! Reference source not found.** shows the user interface of the game. As the name Age of Computers says, the theme of the game is the history of computers. The players start at Gløshaugen, the university campus where the students are based. To advance in the game, the players have to answer different questions and solve tasks.

All knowledge that is gained during the game is saved in three books: the history book, the book containing facts, and the exercise book. These are empty in the beginning of the game and while the player gets more information during the game, the books are filled with this. The player can at any time in the game read in the books to get the required information to answer the questions.

The game contains three different kinds of tasks: multiple choice questions, short-answer questions and programming tasks. When the player has answered a given number of tasks in a level, he can advance to the next level.

Under the game area, there is a chat-window (see Figure 11). It can be used to chat with other players on the same level, and discuss how to solve tasks and other aspects of the game. At some times during the week, student assistants are logged on to the chat to give assistance. This gives the students the possibility to play the game at any given location, they can get help even if they choose to play the game at home, and this gives a lot more freedom than using traditional exercises with guidance in a class room [Friis, Tollefsrud 2004], [aoc.idi.ntnu.no 2006].

4.2 What is required to make a good educational computer game

Some of the problems with traditional class room teaching are that the students think it is boring and not motivating. The young generation of today is used to be entertained all the time and computer games have been introduced in education as an alternative to the traditional way of teaching to meet these requirements.

It is important that educational games are both fun and informative. Both the game play and educational parts have to be prioritized, if not both are done properly; there is no point of using a game in education.

One way to use games in education is to use standard games in an educational context. A game like SimCity can be used to create understanding of economic systems, and Rome Total War can be used to give historic insight in the Roman Empire. But such games are unfortunately not made for educational practice, and, hence, not easy to employ in the classroom. The solution is to create games particularly written for certain educational goals.

A problem with many games made for educational purpose, is the poor quality. There are many reasons for this; one is that the budgets for educational games are way lower than for normal games. This results in disappointed players and the effect may not be as intended. Another problem is that game principles and educational principles often are conflicting. The teacher often wants to take control over what is learned, while in games, the players like to take control over the action. In good games, the players are able to explore some domain and make motivated choices, while in most educational games; the focus is on learning specific facts and skills without providing an adequate motivation within the game world. The best way to focus is on design rather than on existing games. It is easier to construct new games that both have good game play and a good educational part. [Overmars 2005]

4.2.1 Aspects of game design

Creating a computer game involves many different aspects. The game play must be defined, the story must be written, the characters must be made up and designed, levels must be created, and interaction and behaviour of computer controlled entities must be programmed. To test if the players like and understand the game, usability tests have to be performed and marketing and promotion are required to actually sell the game. All these aspects can be used in an educational context.

Language classes could study the important aspects of stories in games. They could write a game story, for example for an adventure game. This would be highly motivating; the story might be turned into an actual game.

In art classes, the students can draw game characters in stead of portraits, or they can paint game backgrounds or other graphics. In music classes, students can study game music and the effects it has on the game player. They could also compose their own game music. [Overmars 2005]

Mathematics

The rules that define the game play are maybe the most essential aspect of the game. Rules describe on one hand the inner mechanisms inside a game (e.g. how much damage a weapon gives on an enemy in a fighting game) and also rules that describes how a player can move in the game, in the same way as in sports games and board games. To design a good game, it is necessary with consistent, balanced, and meaningful rules. This requires not only creativity, but also mathematical skills, in particular in logic and probability calculus.

Systems of mathematical rules are very suitable for an educational context. Students could investigate principles of probability, like the game rock-scissors-paper, where rock beats scissors, scissors beats paper, and paper beats rock. But here, all possibilities have the same probability, and it will not lead to a very funny gameplay. More complex relations are more interesting to study, and in Figure 12 there is an example of the relations in strength between different weapons in a real time strategy game. From this, students can deduce that infantry is not required to win a game. To make this kind of schemes more interesting, costs can be added to entities to see the relative differences in costs between the units.

Probability is also important in many games. Many events in games happen with a certain chance, for example a special bonus or another special feature that can be either good or bad for the player. The game designer must choose such chances correctly, if not, the game play might suffer from this. If the player always wins a game in the first try, or even worse, if he never wins, many might be frustrated.

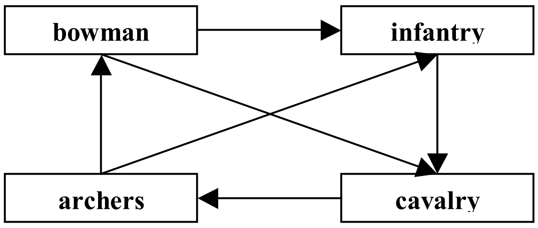


Figure 12 The relation between different units in a strategy game. An arrow indicates the relative strength between units (over). [Overmars 2005]

Figure 13 An example of the grid setup for the Minesweeper game (on the right).

	1	2	3	5
1	●	**1**	**1**	**1**
2	**2**	**2**	**2**	●
3	**2**	●	**3**	**2**
4	●	**2**	**2**	●

Logic is also important, and another important aspect of mathematics in game design is Artificial Intelligence (AI). Rule-based AI is an established approach for programming the behaviour of non-playing characters (NPC). The rules represent the knowledge base of an NPC. The knowledge base holds all the information that an NPC knows about its environments, the objects and their properties, and the relations between them. These rules are constructed from logics.

An example of use of logic is given in the game Minesweeper, illustrated in Figure 13. The game is represented as a grid. The goal of the game is to find all the mines without uncovering and activating them. A cell contains a number (between 0 and 8), a mine, or nothing. When the game starts, all cells are covered, and the player uncovers cells by selecting it. If the player selects a cell with a number, as the one in column 2, row 2 (2,2), he is safe. The number tells how many adjacent cells that contain a mine, and the number here is 2 because there are mines in cell (1,1) and (2,3). If the player has uncovered other adjacent cells that do not have mines, the player can deduce which cells that contain mines. The player can label the cells that might contain a mine with a flag. The game ends when all cells that do not contain a mine are uncovered.

To represent the NPC player's knowledge in Minesweeper, logic is used to decide how it makes decisions when playing the game [Baillie-De Byl 2004].

Computer science

Making a good computer game requires wide knowledge in programming, and graphics and sound, as well as a good game engine are important aspects to create a good game. This can be a problem for people that have not programmed a single line before they start making a game.

A language to implement the game must be chosen, and if an object oriented approach is chosen, JAVA or C++ are alternatives. If the real-time performance of the game is important, C++ might be a better alternative than JAVA. But most important in the choice of language, is that the programmers are familiar and have experience in the language to use.

Many who want to make computer games are not familiar with programming, and to learn a programming language is a demanding subject for people that are not used to work with computers on a daily basis. If the teacher wants to use a game in the classroom, and a game for the intended purpose does not exist, he might want to design a game for the purpose himself.

4.2.2 Learning aspects

While people play a computer game, they learn to master the game better the more they play. In order to create a game that facilitates specific learning, the designer of the game has to make sure that the game properly addresses issues relevant to the learning goals.

Recreational computer game playing may enhance cognitive processes, creative abilities, inductive reasoning, and ability to generate a number of alternative hypotheses for a problem situation. Players playing recreational games are able to simultaneously deal with multiple sets of information without extraneous cognitive load. The challenge is to understand these possibilities in order to retain them in educational games. [Kasvi 2000]

Donald A. Norman [Normann 1993] identifies seven basic requirements of a learning environment:

1. Provides a high intensity of interaction and feedback
2. Has specific goals and established procedures
3. Motivates
4. Provides a continual feeling of challenge, one that is neither too difficult to create frustration nor too easy to create boredom
5. Provides a sense of direct engagement producing the feeling of directly experiencing the environment, directly working on the task
6. Provides appropriate tools that fit the user and task so well that they assist and do not distract
7. Avoids distractions and disruptions that intervene and destroy the subjective experience

Computer games satisfy most of these requirements better than most other learning mediums. The activities a game player performs are identical to those required to learn. Some even think that the motivating aspects of games should be used as a paradigm for general user-interface design.

Game playing can be seen as *the* original educational technology. When animals, as apes, are teaching their children, it is done by playing and not using a blackboard or an overhead projector.

Learning effectiveness

The effectiveness of educational computer games has been discussed widely, with different results. In a project during the autumn 2004, Friis and Tollefsrud [Friis, Tollefsrud 2004] performed a survey where the students in TDT4160 Computer Fundamentals answered questions about the game Age of Computers (AoC), described in chapter 4.1. The main goals of the survey were to find out if computer game based learning improves students' learning, what motivates students, and what the students thought of AoC.

The results showed that the students spent less times on AoC than on traditional exercises (81%), and 58% of them learned more from AoC than from traditional exercises. Only 34% of the students of the course learned most from exercises, which indicates that AoC is not more learning efficient than other ways of learning. Motivation is also important related to learning, and 86% become more motivated by having fun, and 81% would have spent more time on studies if they were fun. 58% found AoC more motivating than traditional exercises, and 68% became more motivated by AoC than by lectures, which shows that a computer game based learning is more motivating than lectures.

Features of a good learning environment

Some computer games are effective learning tools, while some are not. Some key features for what that is important for a good gaming environment are:

- Clear and concise instructions, goals, and objectives should be available for the players to access.
- The game should be challenging.
- The players should have control over gaming options such as speed, difficulty, timing, sound effects, and feedback.
- Aesthetics in the user interface as screen design, graphics, animation, and sound should be of good quality.
- The challenge should be dynamic: the game gets more difficult as the player advances in the game.
- Usability is important. The user interface should be intuitive and only present applicable commands for the current situation.

Coaching is also an important feature. Computer games usually have built in coaching functions. Many games give hints when the players are stuck in a situation and have problems to solve a task or advance in the game. Such a coach should ideally be aware of the player's actions and help the player whenever he needs it. The coaching seems to be most effective when the user can choose to ignore the hints and suggestions when he wants to.

5. Design of a program for making educational games

There exist many different kinds of computer games for educational purpose today, but there is no standard way to develop new games. A framework for making games for education may lead to a standard in the way this kind of games will be made in the future, and it will make it easier to re-use components of other games and it will not be necessary to make new educational games from scratch.

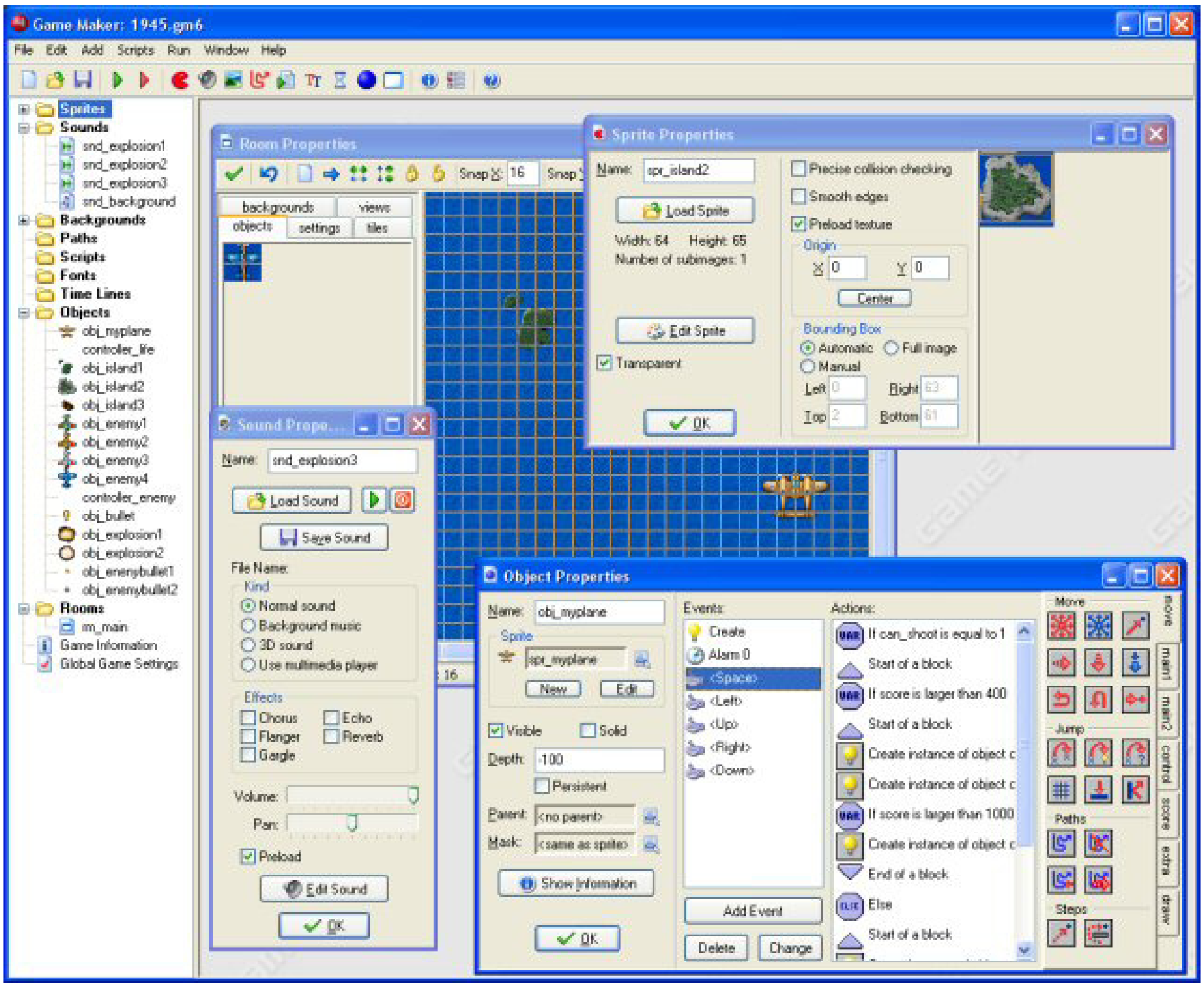
Many people employed in the education business are not familiar with computer programming, and they are not able to learn it. An editor for making games for education might be the solution to people that makes educational material in the future. This will save both time and money for the developers.

There are no doubts that there is a need for a tool for making educational computer games. Many persons employed in the educational business are interested in making games to use in the education, and they want to do it in a not too complex way. A tool for making such games makes it possible for everyone to make their own games without using many hours to learn a programming language before they can begin.

It is important that such an editor is easy and intuitive to use. The graphical user interface has to be easy to understand and use. In addition to focus on usability, the functionality is also very important. The editor must contain functions that the users demand and need to make a fun and challenging game.

5.1 Similar tools

I found one tool for making computer games in an easy way, and that was Game Maker 6.1, developed and designed by Mark Overmars [gamemaker.nl 2006]. The user interface of the program is displayed in Figure 14 . The program is easy to use and makes everyone able to create a computer game. Game Maker is, however, not designed to create games for educational purpose; the tool is made for traditional games.



Sprite Properties: Name: spr_island2; Width: 64 Height: 65; Number of subimages: 1; Precise collision checking [ ]; Smooth edges [ ]; Preload texture [✓]; Origin X 0, Y 0; Bounding Box: [✓ Automatic] Full image, Manual; Left 0, Right 63, Top 2, Bottom 61; Transparent [✓]

Sound Properties: Name: snd_explosion3; Kind: [✓ Normal sound] Background music, 3D sound, Use multimedia player; Effects: Chorus [ ], Echo [ ], Flanger [ ], Reverb [ ], Gargle [ ]; Volume; Pan; Preload [✓]

Object Properties: Name: obj_myplane; Sprite: spr_myplane; Visible [✓]; Solid [ ]; Depth: -100; Persistent [ ]; Parent: <no parent>; Mask: <same as sprite>

Figure 14 The user interface of Game Maker [gamemaker.nl 2006]

5.2 The Educational Game Editor – the idea

I name my tool for the Educational Game Editor (EGE), a tool for making computer games for educational purposes, without any necessary skills and experiences in any programming language. The system shall have an easy to understand user interface with drag- and drop functions, and contain many pre-made elements, but also make the user able to make every elements from scratch. This gives the users freedom to choose between making big and complex games, or small and simple games, depending on time and resources. It should be possible to make games in different genres and with different educational contents and for different user groups and ages.

The pre-made game elements will be saved in a library. The library shall be stored in an object-oriented database. This gives each independent object a unique identity.

The database containing the library with the different pre-made elements shall be organized in a hypermedia structure to make it easy and fast to look up elements, and make it easier to modify and update the EGE if decided.

The following section contains a software requirements specification for the system, and will give a brief description of how the system should be.

5.3 Software Requirements Specification for the Educational Game Editor

This document is based on the IEEE Standard 830-1998.

5.3.1 Introduction

This document contains the requirements specification for the Educational Game Editor (EGE), a tool for designing computer games for use in education. The system is developed during a Master thesis in computer science at NTNU.

The Requirements Specification contains an overall description of the system, functional requirements, quality requirements, and constraints. The functional and quality requirements are exemplified with use cases and scenarios.

5.3.1.1 Overview

Chapter 2 contains an overall description of the system, the perspective and a list of the functions that the system must contain. Further it gives a description of users of the system, constraints, assumptions, and dependencies.

Chapter 3 contains the functional requirements of the system. The most important requirements are listed, and they are explained in textual use cases.

Chapter 4 contains the quality requirements of the system, focusing on the different quality attributes for the system. Scenarios explain the most important requirements.

5.3.2 Overall description

This section gives an overall description of the system, as product perspective, functions, user characteristics and constraints.

5.3.2.1 Product perspective

The EGE is an editor for making computer games for use in education. The system is a new and self-contained program. EGE will be implemented in JAVA and be platform independent. This makes the system independent of software and hardware environment.

5.3.2.2 Product functions

The EGE should have functionality to make games in an easy and efficient way. The program must contain the following functions:

- An easy to understand user interface with drag- and drop functions
- A library of pre-made elements
- An editor to draw elements
- A tool for making sounds and music
- A tool for making animations
- An editor to make game rules
- A tool for adding and editing tasks and questions
- A text editor
- A tool for compiling the game
- A tool for debugging the game
- A function to test the game

5.3.2.3 User classes and characteristics

The system shall be designed for an educational purpose, and the typical users will be persons employed in the educational business. The technical expertise of the users will vary from very low to high. Most of the users will have an education of a high degree, and will be familiar with using computers in their job. The skills in using and computer will vary in some way, and many have no experience beyond using Word and other standard Windows applications, while others are used to programming. This results in different requirements for users with different skills and experience in using a computer, and the system must be fitted both novice and expert users.

5.3.2.4 Constraints

The program must be platform independent. To fulfil this, JAVA will be used as programming language if it is going to be implemented.

5.3.3 Functional requirements

This section contains the functional requirements of the EGE. These requirements describe what functionality the system has to fulfil. Only the most important requirements with high priority are mentioned here.

The system must have the following functions:

1. Be able to perform basic file management
   The system shall have functions for creating a new project, save it, and open an existing project.
2. Be able to create an object
   The system shall have functions for creating a new object and give it desired properties and save it in the library.
3. Be able to open and edit any existing object in the library and save the changes.
4. Be able to draw different game elements in the drawing editor and save them in the library.
5. Be able to make sounds and music in the composing and sound-making tool and save them in the library.
6. Be able to make animations in the animating tool.
7. Be able to add and edit game rules in the rules tool.
8. Be able to add and edit tasks and questions in the task and question tool.
9. Be able to write and edit text in the text editor
10. Be able to make scripts
11. Be able to compile the game whenever the user wants to by using the compiler tool.
12. Be able to debug the game in the debug-tool if it is not working properly after have been compiled.
13. Be able to test the finished game by the testing function.

5.3.4 Textual use cases

The textual use cases describe situations regarding the functional requirements, and describe the basic path of events when fulfilling a requirement. Alternative paths are also described where they exist. The use cases are illustrated with figures.

5.3.4.1 Textual use case to save a project

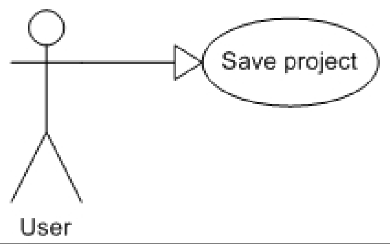
Use case name:	**Save project**	
ID:	F-UC-1	
Priority:	High	
Iteration:	Conceptual	
Summary:	The user must be able to save a project whenever he wants to	
Supports requirement:	1	
Basic path of events:	1. The user chooses save from the file-menu 2. The user chooses the location to save the file 3. The user chooses a filename 4. The user selects save and the project is saved	
Alternative path:	1. The user chooses the save-button on the toolbar 2. If the project is saved before, the project is saved 3. If the project is not saved, the user is asked to type a filename and the project is saved.	
Pre-conditions:	A project is started	
Post-conditions:	The project is saved	
Date:	17.02.06	
Author:	John Ola Tollefsrud	

Table 1 Functional use case 1

5.3.4.2 Textual use case to open a project

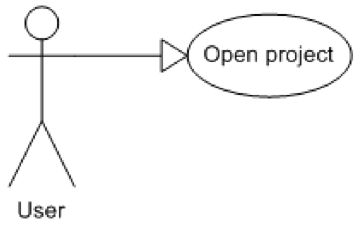
Use case name:	**Open project**	
ID:	F-UC-2	
Priority:	High	
Iteration:	Conceptual	
Summary:	The user must be able to open an existing project	
Supports requirement:	1	
Basic path of events:	1. The user chooses open from the file-menu 2. The user finds the location of the file to open 3. The user finds or types the filename of the file to open 4. The user selects open and the project is opened	
Alternative path:	1. The user chooses the open-button on the toolbar 2. The user finds the location of the file to open 3. The user finds or types the filename of the file to open 4. The user selects open and the project is opened	
Pre-conditions:	There must exist saved projects	
Post-conditions:	A project is opened	
Date:	17.02.06	
Author:	John Ola Tollefsrud	

Table 2 Functional use case 2

5.3.4.3 Textual use case to create an object

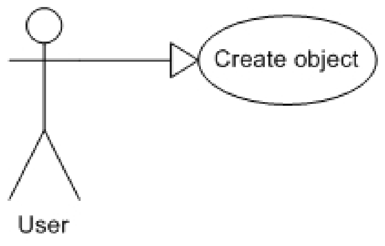
Use case name:	**Create an object**	
ID:	F-UC-3	
Priority:	High	
Iteration:	Conceptual	
Summary:	The user must be able to create a new object and save it	
Supports requirement:	2	
Basic path of events:	1. The user chooses create object from the object-menu 2. The user names the object 3. The user sets the object's properties 4. The user chooses save object and the object is saved in the library	
Alternative path:	-	
Pre-conditions:	A project is started	
Post-conditions:	An object is created	
Date:	17.02.06	
Author:	John Ola Tollefsrud	

Table 3 Functional use case 3

5.3.4.4 Textual use case to edit an existing object in the library

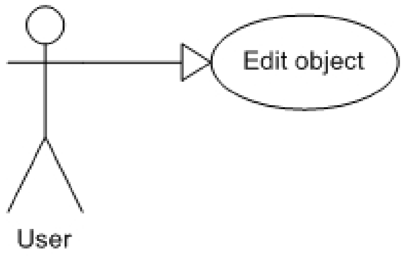
Use case name:	**Edit an existing object**	
ID:	F-UC-4	
Priority:	High	
Iteration:	Conceptual	
Summary:	The user must be able to open an existing object, edit it and save it	
Supports requirement:	3	
Basic path of events:	1. The user chooses open from the library-menu 2. The user finds the desired object to edit or types the object's name 3. The user changes the desired properties 4. The user chooses save object and the object is saved in the library	
Alternative path:	-	
Pre-conditions:	A project is started and contain existing objects	
Post-conditions:	The object is edited and saved	
Date:	17.02.06	
Author:	John Ola Tollefsrud	

Table 4 Functional use case 4

5.3.4.5 Textual use case to draw a game element

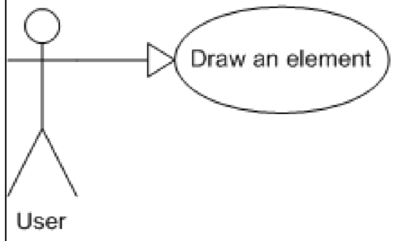
Use case name:	**Draw an element**	
ID:	F-UC-5	
Priority:	High	
Iteration:	Conceptual	
Summary:	The user must be able to draw a game element in the drawing tool	
Supports requirement:	4	
Basic path of events:	1. The user chooses draw from the tools-menu 2. The user chooses create new drawing 3. The user draws an object 4. The user chooses save, names the object, and saves it in the library	
Alternative path:	-	
Pre-conditions:	A project is running	
Post-conditions:	A new object is drawn and saved in the library	
Date:	17.02.06	
Author:	John Ola Tollefsrud	

Table 5 Functional use case 5

5.3.4.6 Textual use case to compose a tune

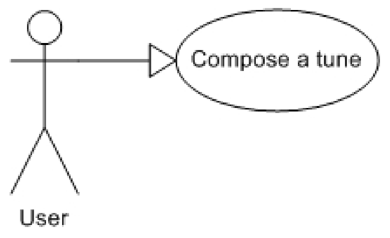
Use case name:	**Compose a tune**	
ID:	F-UC-6	
Priority:	High	
Iteration:	Conceptual	
Summary:	The user must be able to make a sound or tune in the sounds and music-tool	
Supports requirement:	5	
Basic path of events:	1. The user chooses sounds and music from the tool-menu 2. The user chooses create new tune 3. The user composes a tune 4. The user chooses save, names the tune, and saves it in the library	
Alternative path:	-	
Pre-conditions:	A project is running	
Post-conditions:	A new tune is composed and saved in the library	
Date:	17.02.06	
Author:	John Ola Tollefsrud	

Table 6 Functional use case 6

5.3.4.7 Textual use case to make an animation

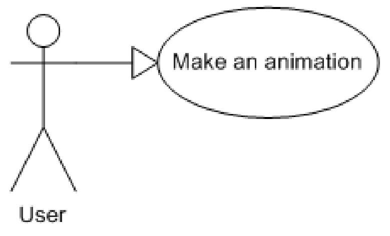
Use case name:	**Make an animation**	
ID:	F-UC-7	
Priority:	High	
Iteration:	Conceptual	
Summary:	The user must be able to make an animation in the animation-tool	
Supports requirement:	6	
Basic path of events:	1. The user chooses animation from the tools-menu 2. The user chooses create new animation 3. The user makes an animation 4. The user chooses save, names the animation, and saves it in the library	
Alternative path:	-	
Pre-conditions:	A project is running	
Post-conditions:	An animation is created and saved in the library	
Date:	17.02.06	
Author:	John Ola Tollefsrud	

Table 7 Functional use case 7

5.3.4.8 Textual use case to make a game rule

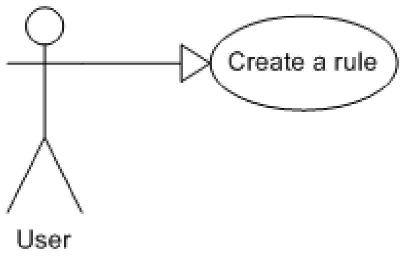
Use case name:	**Create a rule**
ID:	F-UC-8
Priority:	High
Iteration:	Conceptual
Summary:	The user must be able to make rules for the game
Supports requirement:	7
Basic path of events:	1. The user chooses the rule editor from the tools-menu 2. The user chooses make new rule 3. The user writes the rule 4. The user sets properties for the rule and which objects that are involved 5. The user chooses save, names the rule, and saves it in the library 
Alternative path:	-
Pre-conditions:	A project is running and contain objects
Post-conditions:	A game-rule is made and saved in the library
Date:	17.02.06
Author:	John Ola Tollefsrud

Table 8 Functional use case 8

5.3.4.9 Textual use case to write a question

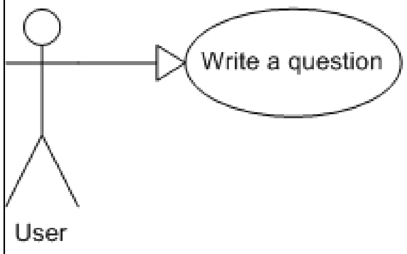
Use case name:	**Write a question**
ID:	F-UC-9
Priority:	High
Iteration:	Conceptual
Summary:	The user must be able to write questions
Supports requirement:	8
Basic path of events:	1. The user chooses the question tool from the tools-menu 2. The user chooses make new question 3. The user chooses the desired type of question 4. The user writes the answer 5. The user chooses save, names the question, and saves it in the library 
Alternative path:	-
Pre-conditions:	A project is running
Post-conditions:	A question is made and saved in the library
Date:	17.02.06
Author:	John Ola Tollefsrud

Table 9 Functional use case 9

5.3.4.10 Textual use case to write text

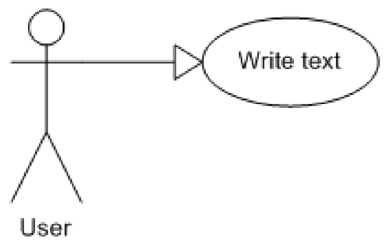
Use case name:	**Write text**	
ID:	F-UC-10	
Priority:	High	
Iteration:	Conceptual	
Summary:	The user must be able to write text	
Supports requirement:	9	
Basic path of events:	1. The user chooses the text editor from the tools-menu 2. The user chooses make new text object 3. The user writes the text 4. The user chooses save, names the text object, and saves it in the library	
Alternative path:	-	
Pre-conditions:	A project is running	
Post-conditions:	A text object is made and saved in the library	
Date:	17.02.06	
Author:	John Ola Tollefsrud	

Table 10 Functional use case 10

5.3.4.11 Textual use case to make a script

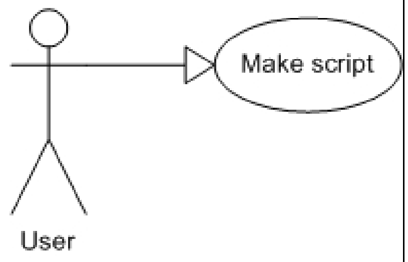
Use case name:	**Make a script**	
ID:	F-UC-11	
Priority:	High	
Iteration:	Conceptual	
Summary:	The user must be able to make scripts	
Supports requirement:	10	
Basic path of events:	1. The user chooses the script editor from the tools-menu 2. The user chooses make new script 3. The user writes the script 4. The user chooses save, names the script, and saves the script in the library	
Alternative path:	-	
Pre-conditions:	A project is running	
Post-conditions:	A script is made and saved in the library	
Date:	18.02.06	
Author:	John Ola Tollefsrud	

Table 11 Functional use case 11

5.3.4.12 Textual use case to compile a game

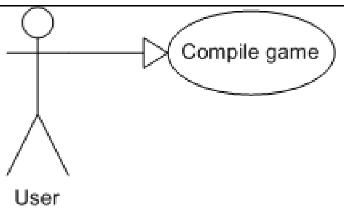
Use case name:	**Compile a game**
ID:	F-UC-12
Priority:	High
Iteration:	Conceptual
Summary:	The user must be able to compile a game
Supports requirement:	11
Basic path of events:	1. The user chooses compile from the tools-menu 2. The game is compiled if it does not contain bugs 3. The game is saved as a runnable file 
Alternative path:	2a. The game contain bugs and is not compiled 2b. The user removes the bugs, go to step 2
Pre-conditions:	A project is running
Post-conditions:	A runnable game is made and saved
Date:	18.02.06
Author:	John Ola Tollefsrud

Table 12 Functional use case 12

5.3.4.13 Textual use case to debug a game

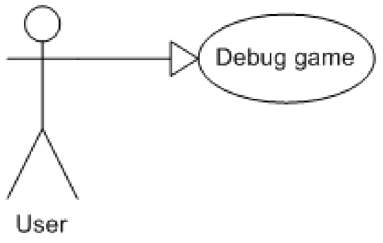
Use case name:	**Debug game**
ID:	F-UC-13
Priority:	High
Iteration:	Conceptual
Summary:	The user must be able to debug a game that not is running properly
Supports requirement:	12
Basic path of events:	1. The user has failed to compile a game 2. The user chooses the debug tool from the tools-menu 3. The user runs the debugger to discover and remove the bugs 4. The game is debugged, saved and can be run 
Alternative path:	-
Pre-conditions:	A game is made and contains bugs
Post-conditions:	A runnable game without bugs is made and saved
Date:	18.02.06
Author:	John Ola Tollefsrud

Table 13 Functional use case 13

5.3.4.14 Textual use case to test a game

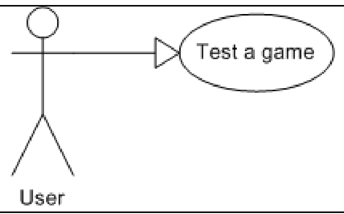
Use case name:	**Test a game**
ID:	F-UC-14
Priority:	High
Iteration:	Conceptual
Summary:	The user must be able to test a finished game to se if it runs properly
Supports requirement:	13
Basic path of events:	1. The user chooses test game from the tools-menu 2. The game is running and the user tests the functions and if it behaves as desired 
Alternative path:	-
Pre-conditions:	A finished game is saved in the library
Post-conditions:	A game is tested
Date:	18.02.06
Author:	John Ola Tollefsrud

Table 14 Functional use case 14

5.3.4 Quality requirements

This section describes the quality requirements, focusing on the different quality attributes. The most important quality tactics in this system is usability, but availability is also important. As the program is running on single computers, security is not taken into consideration.

5.3.4.1 Availability

1. Fault detection. It is not realistic with a system that never fails, and this makes it important to discover faults when they are occurring.
2. Fault recovery. When a fault occurs, it is important that the system can repair it and continue to run in normal mode. Scenarios are presented in Table 15 and Table 16.

Availability scenario 1

Source	Internal
Stimulus	The program crashes
Environment	Normal
Artefact	Process
Response	The system gives an error report and asks the user to save his work
Response measure	Give error report in 99 of 100 occurrences

Table 15 Availability scenario 1

Availability scenario 2

Source	Internal
Stimulus	A fault occurs
Environment	Normal
Artefact	Process
Response	The system gives the user a report about the failure, and the system tries to fix the problem
Response measure	Fix the problem in 99 of 100 occurrences

Table 16 Availability scenario 2

5.3.4.2 Testability

1. To ensure a high degree of availability, it is important that the system is tested properly to see that it not is failing. Scenario for testability is presented in Table 17.
2. There must be possible to control the program's sequence of states and actions.

Testability scenario 1

Source	Developer
Stimulus	Performs a test
Environment	At development time
Artefact	System module
Response	The module gives the desired output
Response measure	95 % of tested statements executed

Table 17 Testability scenario 1

5.3.4.3 Modifiability

1. The system must be divided into different and independent modules. This makes it able to change or replace one module without affecting other parts of the program.
2. Good documentation is important to make it easy to understand how the system is working. Both well-documented source code, a brief description of the system, and a user manual are important. See scenarios in Table 18 and Table 19.

Modifiability scenario 1

Source	Developer
Stimulus	Wants to change a module
Environment	At update time
Artefact	System module
Response	The modification is performed without problems
Response measure	Performed in 5 minutes

Table 18 Modifiability scenario 1

Modifiability scenario 2

Source	Developer
Stimulus	Wants to add a new module
Environment	At update time
Artefact	System module
Response	The module is added and the system is running as before the update
Response measure	Performed in 5 minutes

Table 19 Modifiability scenario 2

5.3.4.4 Performance

1. Performance is not of the most important attributes here, but operations in the program should be finished in reasonable time. See scenarios in Table 20 and Table 21.

Performance scenario 1

Source	User
Stimulus	Open a file
Environment	At runtime
Artefact	System
Response	The user opens a file
Response measure	Performed in 5 seconds

Table 20 Performance scenario 1

Performance scenario 2

Source	User
Stimulus	Compile a game
Environment	At runtime
Artefact	System
Response	The game is compiled
Response measure	Performed in 30 seconds

Table 21 Performance scenario 2

5.3.4.5 Usability

1. The user must learn and understand how the system is working and how to use the tools. The functions must be self-explanatory.
2. The system must have an interface that makes it easy for the user to make the program in an efficient way.
3. Minimize the impact of errors done by the users.
4. The system must give the user confidence and satisfaction. The user must feel that he manages the system and is able to make working games. For usability scenarios, see

Usability scenario 1

Source	User
Stimulus	The user wants to open a game file
Environment	At runtime
Artefact	System
Response	The file is opened
Response measure	Performed in less than 5 seconds

Table 22 Usability scenario 1

Usability scenario 2

Source	User
Stimulus	The user wants to undo an operation
Environment	At runtime
Artefact	System
Response	The program return to the state before the operation that the user wanted undone
Response measure	Performed in less than 1 second

Table 23 Usability scenario 2

5.4 Conceptual design

This section describes in detail the conceptual design of the program, covering what parts it consists of and how they relate to each other, and how to use the program to make a game. Since the program has not been implemented, the design might be incomplete in some areas because some weaknesses can be discovered during the implementation of the program.

5.4.1 Overall description

The user of the Educational Game Editor (EGE) is set to make computer games to use in educational situations. The user makes different game elements as rooms where the game takes place, human and non-human characters that move around in the room, different problems that the player must solve and tasks and question that have to be answered and solved. In contrast to making traditional computer games, the goal of the games created in the EGE is not only to entertain the user, but also to educate the user. Therefore, in addition to make a good game play, good learning elements must be added to the games.

An overview of the basic phases in creating a game and how they are connected is shown in Figure 15. The phases are:

- Create new game
- Make objects
- Make tasks
- Make rules
- Compile game
- Debug game if the compilation contains failures
- Test the game

The user first has to create a new game. Then he has to create the game objects of different kinds and further make different tasks the players have to solve. The game have to contain rules that gives limitations on what the players are allowed to do and also rules that defines how the different objects in the game shall interact. When the user has finished these phases, he has to compile the game to check if it is made in a correct and runnable way. If the game compiles without failures, the game can be tested to see if the design and functionality has been as the user wants to. If the game does not compile and contains failures, the user can debug the game to detect the errors in the game and repair them. After the errors have been removed, the game can be tested.

The design process is evolutionary; after one step in the design process is completed it is often necessary to go back to the previous step in the process. This process continues until the program created has the desired functionality and quality.

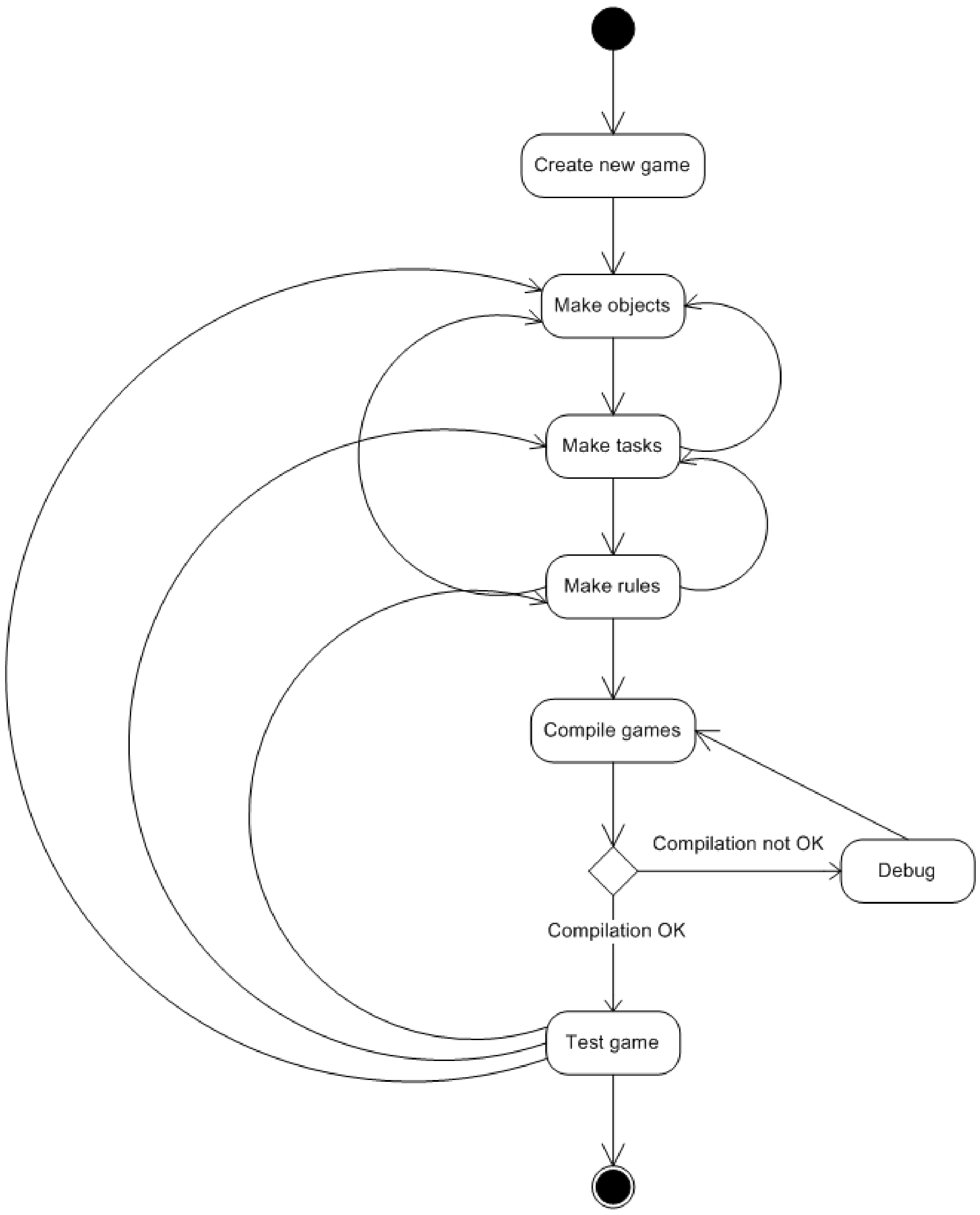


Figure 15 Basic phases in creating a game in EGE

5.4.2 Program elements

The program consists of different elements that are necessary to make the users able to make games that satisfy the users. The program consists of the following main elements:

- File management
- Object management
- Library
- Tools

5.4.2.1 File management

File management is necessary in a program if the user shall be able to save his work. The program contains the following functions for basic file management:

- **New** – creates a new game project.
- **Save** – saves the current project with the current name. If the game is saved for the first time, the user has to give the game-file a name and chose the location to save it.
- **Save as** – saves the current game with another name if the user wants to save a different copy.
- **Open** – opens a game project.
- **Exit** – exits the game.

5.4.2.2 Object management

The program shall make object-oriented games, and this makes the object as the basic unit of the games. An object can be of different classes, as a room, a player character, a task, or a rule. The objects have different properties depending on what class it belongs to. Object management has two basic functions:

- **New** – creates a new object. After the object is created, the attributes of the object can be set, see Table 24 and saved when finished.
- **Open** – opens an object, and gives the user the possibility to change some of the attributes of an existing object and save the changes.

Name	The name of the object
Class	The object's class, that can be of type animation, character, room, image, background, text, sound, task
Movement	If the object is movable, the way it moves is described here
Visibility	Set true or false, depending on if the object is going to be visible on the screen
Dependencies	If the objects are dependent on other objects, it is listed here
Parent	If the object has a parent object which it inherit from, it is set here
Rule(s)	Rules that decides how the object can behave and interact with other objects, including limitations on what an object is allowed to do, how it can score points if it is a playing character, and many other properties
Script(s)	If there are made scripts containing functions or properties for the objects, they are added here

Table 24 Object attributes

5.4.2.3 Library

The different objects are all organized in a library. The library consists of many pre-made elements, and as the user makes new game elements, the library will grow as the user-made objects are added to the library. The library is divided into parts that correspond to the different object classes. The user can choose to add any object he creates to the library in the corresponding object class.

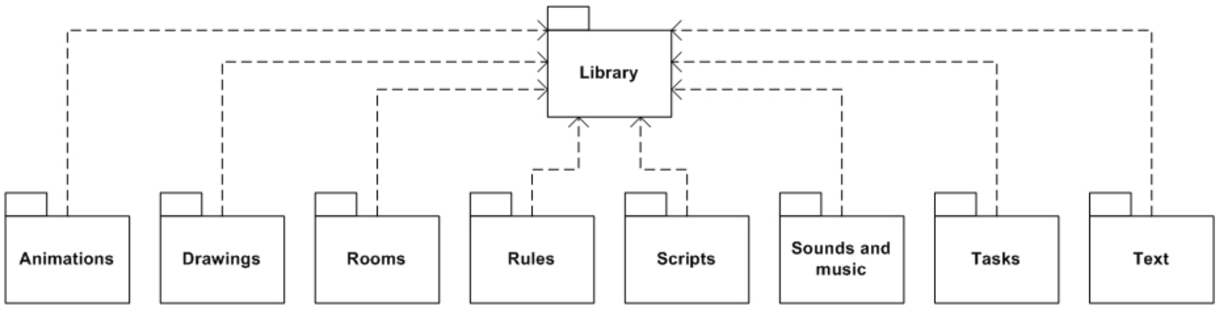


Figure 16 The organization of the library

How the library is divided and organized is shown in Figure 16. Each different object class is saved in a distinct part of the library.

5.4.2.4 Tools

The program consists of different kinds of tools to create different game elements. These are necessary to make games of decent quality without too much work for the users. The program consists of the following tools:

- Animation tool
- Compiler
- Debugger
- Drawing tool
- Rule editor
- Script editor
- Task tool
- Test tool
- Text editor

Animation tool

To make animations without proper tools, is very demanding, and takes a lot of time. The tool for making animations makes this easy for the users. The user first has to create an object that he wants to animate, using the drawing tool, or use a pre-made object in the library. Then he chooses how the object is going to move, how the different parts of the object change position on the screen. This is done by first dividing the object into different parts and drag them into different positions or write the coordinates for how the position shall change.

Compiler

When the user creates a game and saves the file, the game file is saved as source code that is not runnable. To create a program file that can be run independent of the program, the source code must be compiled. The compiler translates the source code to an executable program. If the game file contains errors, a message will appear on the screen, describing what went wrong, and where the error is, i.e. in what line number in the code the error occurs.

Debugger

If the game not compiles, the user has to find and correct the error(s). In many cases, this could be a difficult and demanding job. The debugging tool makes this easier and faster to do. The debugger walks through the source code in steps, and at each step, the user can see how the programs behave; what statements are executed with what variables as arguments. When a statement can not be executed because of an error, the debugger stops and explains the user why it stops, and the user can correct the error.

Drawing tool

The drawing tool makes it easier for the user to draw objects. The tool contains all the most important functions to make different images and pictures that are needed to make a computer game with graphics of high quality. The library consists of many pre-made elements that can be used and modified by the user; this saves a lot of time and work.

Rule editor

To organize the game and set conditions on what the user is allowed to do and how the non-human characters and elements behave, rules are necessary. A game without rules is not working very well. In the rule editor the user can make rules for single game objects and rules for entire games. For example in a quiz game, the user must make rules to decide what answer is correct on each question, and rules that decide how to score points and how to win the game, and also what happens when the players answer the questions incorrectly.

Script editor

The game files made in the program, will be made as source code. This is an easy way to do it for users not familiar with computer programming. But for users that are used to program, it may be easier and faster to write the programs directly, without using the tools. The script editor makes the users able to program parts of the game direct. The scripts can either be whole programs or parts that describe special properties of an object.

Task editor

To make a game challenging, the user must meet problems and hinders of some way. Many games contain tasks that must be solved to advance in the game. Since the program is made for making educational games, the educational content is best added in different kind of tasks the players must solve. The task editor is a tool for making different kind of tasks in the game, i.e. questions that have to be answered in a quiz game, lands that have to be conquered in a map-based strategy game, or stocks to be sold in an economic game.

Test tool

After having finished parts of, or the whole game, it is important to test it to see if it works as intended. A compiled game-file does not make sure that the game has been made as desired. The test tool makes the user able to test the game and control the behavior of the game and change different variables if the user wants to change anything during the test.

Text editor

Many games contain text in some way, either as information windows, which is a way to present some educational contents, text in dialogues, describing game elements and many other situations. The text editor makes the user able to write text that will appear on the screen, and format it in the way he likes to.

5.5 Design and architecture

This chapter describes the underlying design of the program. First, a technical description of the program is given, which shows the architecture, illustrated with figures in UML-notation. The user interface and how to use the program are described after the architecture.

5.5.1 Architectural drivers

The most important driver for the program is usability; however availability is also important; it does not help much that the program is easy to use if it crashes all the time. The drivers are:

- **Usability** is the most important driver. The program has to be easy to use and have a user interface that is not too complicated. The program functions and tools must be easy to find and use.

- **Availability** is also important. The system must be stable, and have a long meantime between failures. The impact of failures must be as low as possible without damaging the work done by the user.

- To secure a high degree of availability and avoid breakdowns, **testability** is an important driver. If the program has a high degree of testability, the developers can be saved for a lot of time. A testable system must have availability to control the state of each component and input and observe the output.

- **Modifiability** deals with the cost of changes made on the system. This implies what can be changed, when the changes can be done, and who performs them. When a change has been specified, it can be necessary to design, implement and test a new implementation. To maintain this, the system should be made up by independent modules that can be changed without affecting other parts of the system.

- Another important driver is to make the users of the problem as satisfied and confident as possible. The program must make the users happy and make them make the games that they have fantasized about, but thought were too hard to accomplish.

5.5.2 Central scenarios

This section describes central scenarios when using the program. It describes some of the basic functions, and they are illustrated by diagrams.

5.5.2.1 Open file

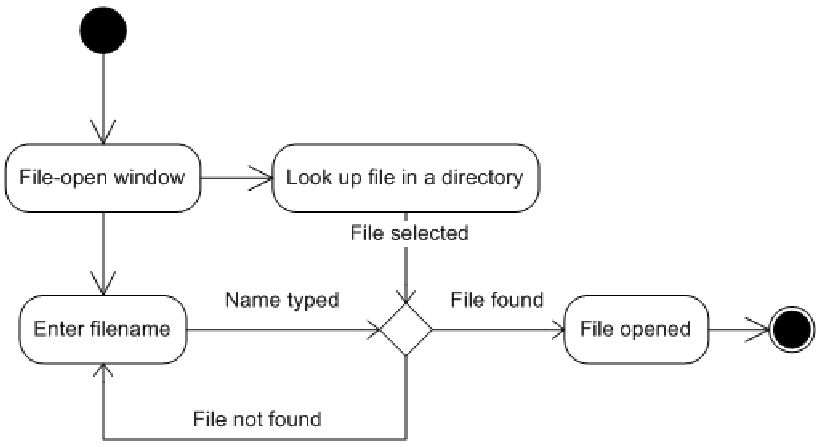


Figure 17 Activity diagram for open file

This scenario happens every time the user wants to open a file. The user is asked to type a filename, or look it up in a directory. If the file is found, it is opened. If the file is not found, the program returns to the enter filename window.

5.5.2.2 Save file

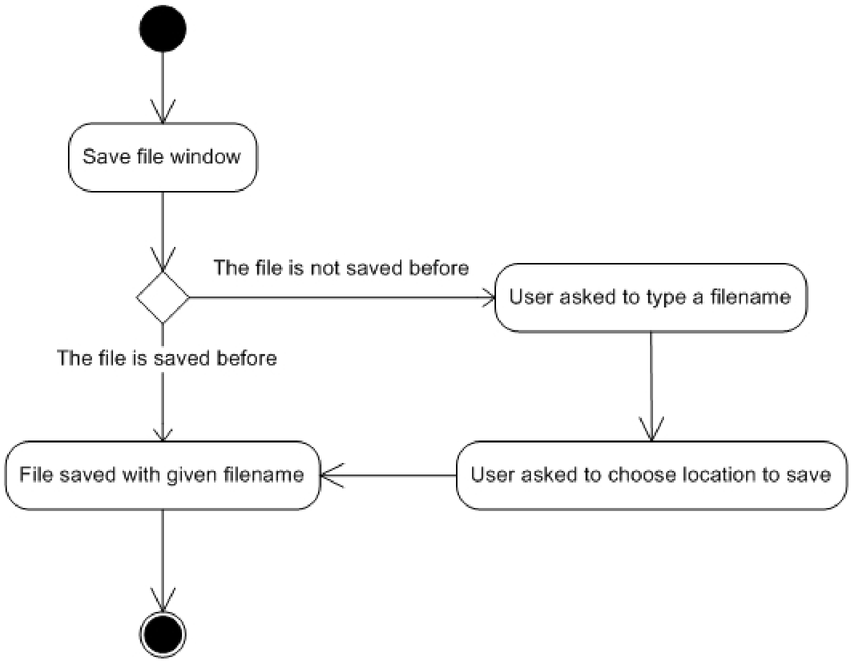

Save file window
The file is not saved before
User asked to type a filename
The file is saved before
File saved with given filename
User asked to choose location to save

Figure 18 Activity diagram for save file

This scenario describes what that happens when the user wants to save a file. If the file is saved before, the file is saved with the same filename, replacing the last version of the file. If the file is saved for the first time, the user has to type a filename and choose a location for where the game shall be saved.

5.5.2.3 Exit program

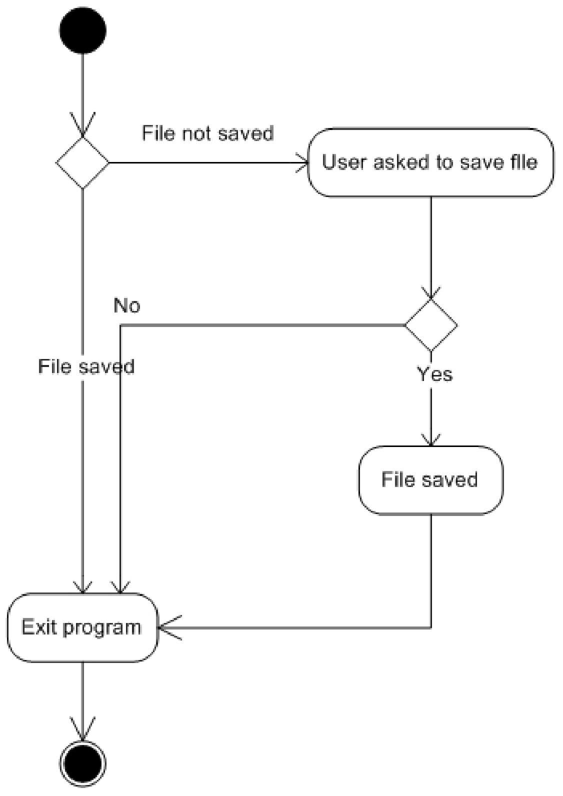


Figure 19 Activity diagram for exit program

When the user wants to exit the program, the program asks the user to save his work if it is not saved. The user can choose to save his changes if he likes to. If the file is already saved, the game exits without any questions from the program.

5.5.2.4 Create an object

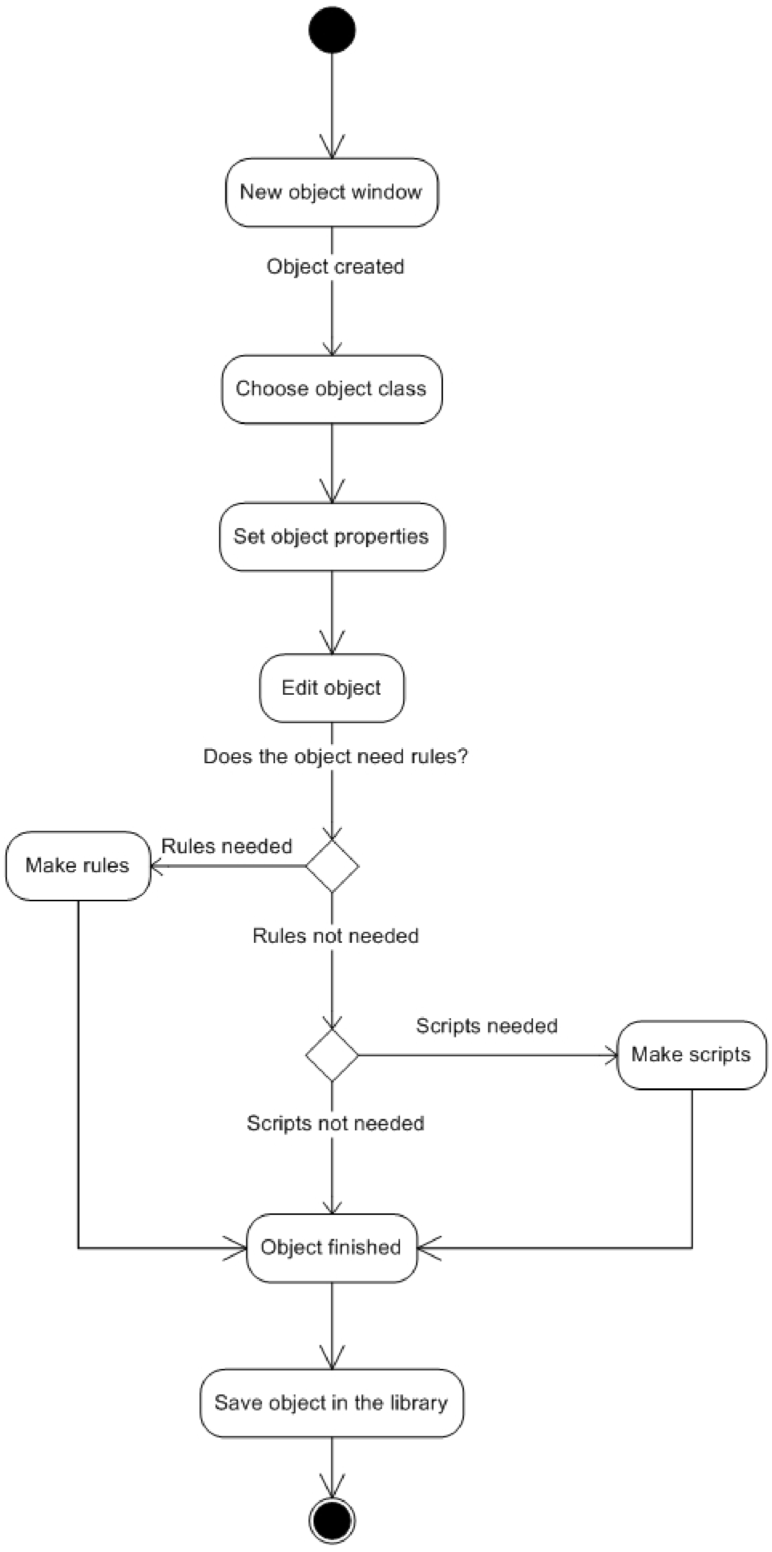


Figure 20 Activity diagram for create object

The user first selects what class of object he wants to create, before setting the different properties and attributes. Then he edits the objects, draws it if it is an image or composes a sound in the tools for the given class. It is optional to add rules and scripts to objects; some types of objects can not have any rules associated. When the object is finished, it is saved in the library.

5.5.2.5 Open an object in the library

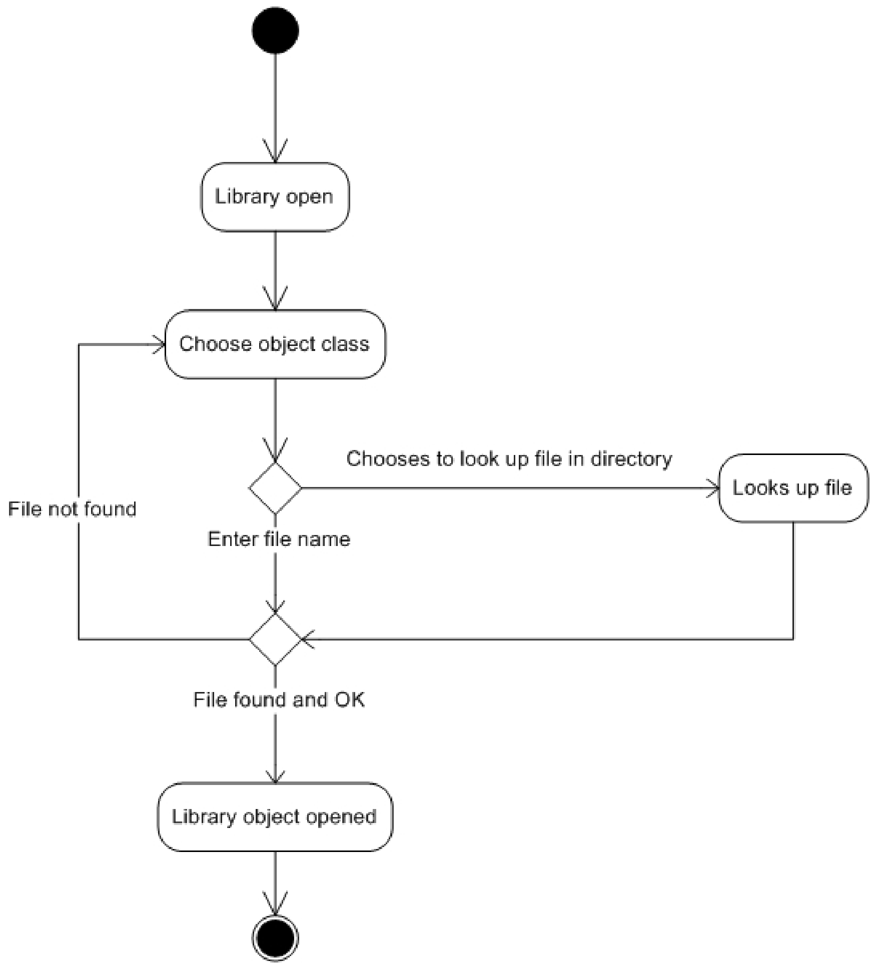


Figure 21 Activity diagram for opening an object in the library

When the user opens the library, he is asked to choose what class the object belongs to. Then he can either type the file name, or look up the file in a directory. If the chosen file exists, it is opened, if not, the program returns to the state of choosing object class.

5.5.3 Program structure

This section describes the structure of the program. It is not too detailed, since the program is not implemented, and a too strict design might add too many limitations if the program is going to be implemented in the future.

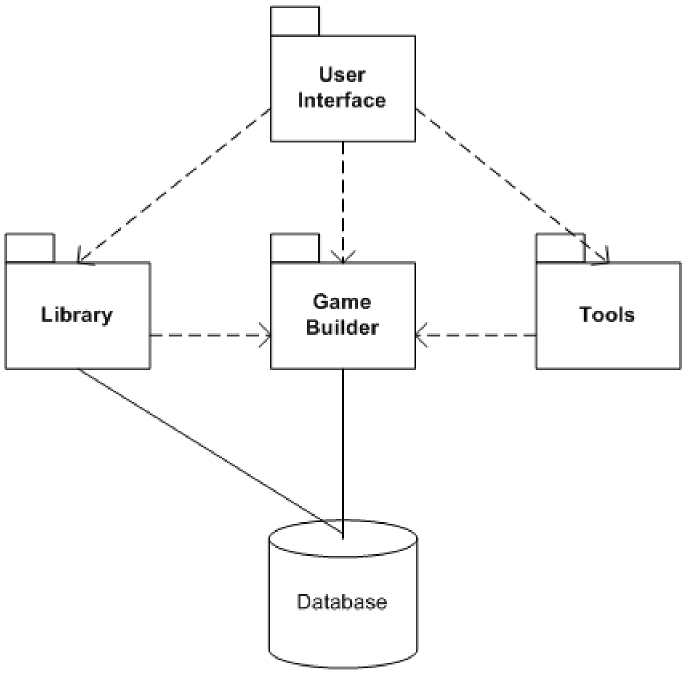


Figure 22 Overview of the program structure

The program is divided into four main packages: User Interface, Library, Game Builder, and Tools, and they are connected to the Database. The Game Builder is the core of the program, which the other packages are dependent on.

The Library and Game Builder packages have interface to connect to the Database and are able to load and store data in the database.

5.5.3.1 User Interface

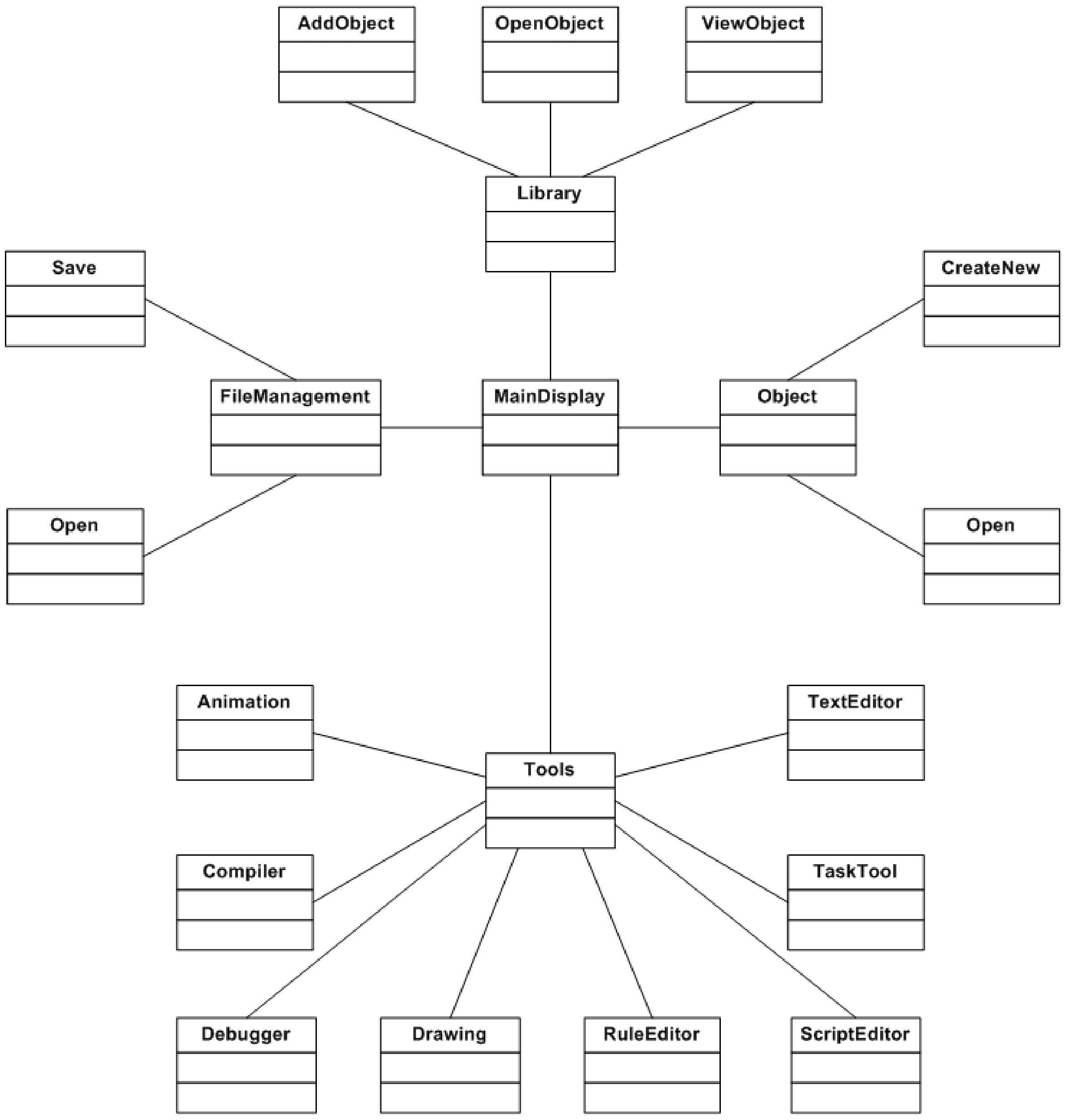


Figure 23 Class diagram for User Interface

The User Interface package contains one class for each different interface in the program. The MainDisplay is the core; all the other classes are directly or indirectly connected to this class. More different user interface classes might be needed when the game is implemented.

5.5.3.2 Library

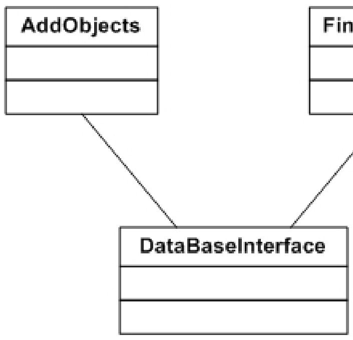


Figure 24 Class diagram for Library

The Library package contains 3 classes, one to add objects, one to find, and a class that handles the connection to the database.

5.5.3.3 Tools

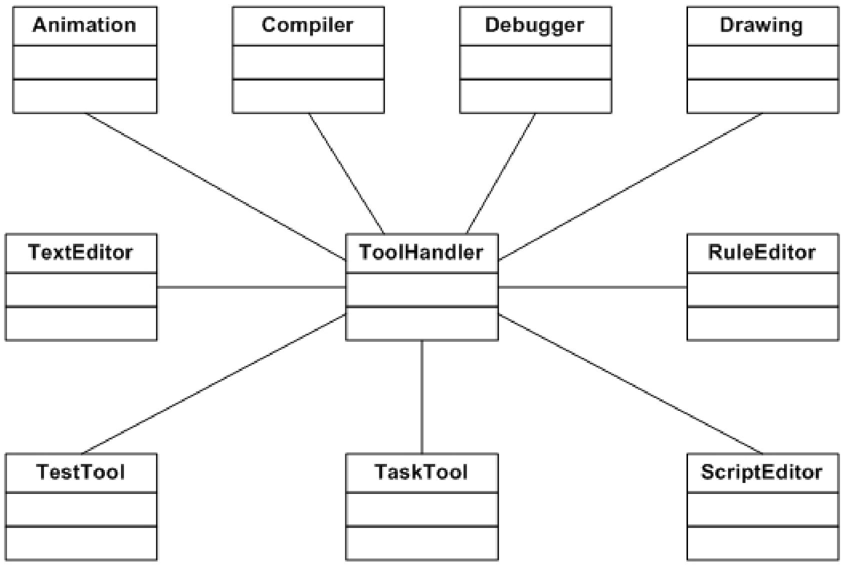


Figure 25 Class diagram for Tools

The Tools package contains one class for each different tool and one ToolHandler class that connects them and communicates with the GameBuilder package.

5.5.3.4 Game Builder

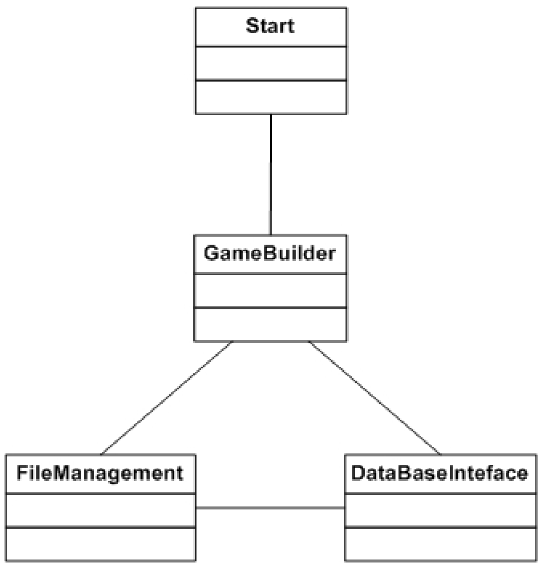


Figure 26 Class diagram for Game Builder

The GameBuilder connects all the game elements and builds the actual game. It starts the program, handles file management, handles database transactions and communicates with the other packages.

5.5.3.5 Database

The system will use an object oriented relational database with one record for each independent object, and are sorted after the object classes. The database interface in the program will have a hypermedia structure, with different objects of different classes connected in a network. The database is not described here because the design considerations are not yet decided.

5.5.4 User interface

The user interface of the program is shown in Figure 27. It shows the program in the way it looks like when no projects are running. The menu-line on the top contains the menus File, Object, Library, Tools, and Windows. The objects of the menus are showed in Figure 27. The toolbar is placed under the menu line. The icons and their corresponding function in the menus are explained in Table 25. More windows are presented and described in chapter 5.6.

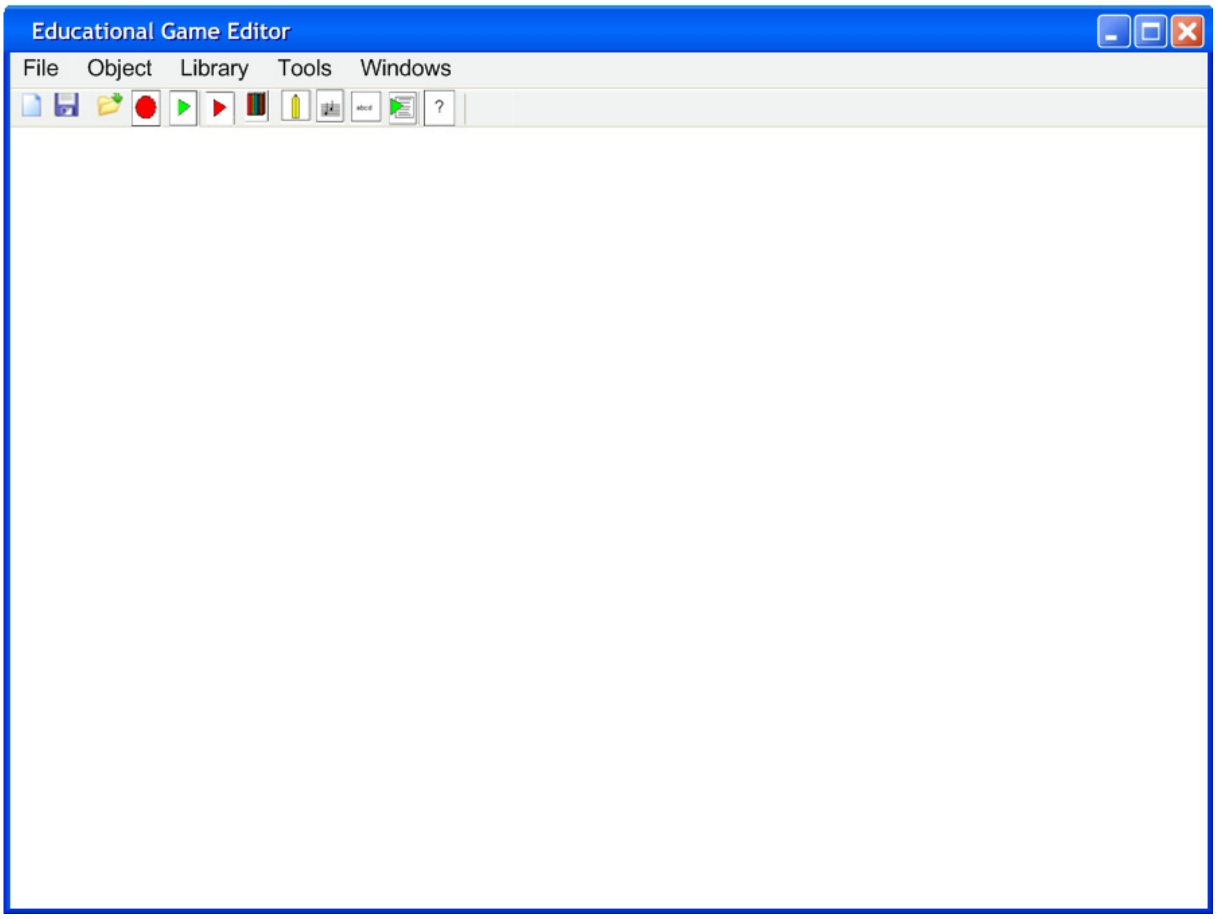


Figure 27 User interface of Educational Game Editor

The working area of the program is the main window under the toolbar. The game itself is made here, and the objects are placed here and interact in the way the user wants.

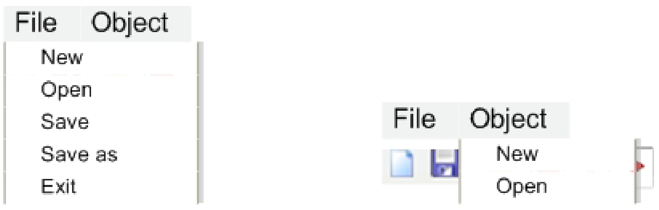


Figure 28 File and Object menus

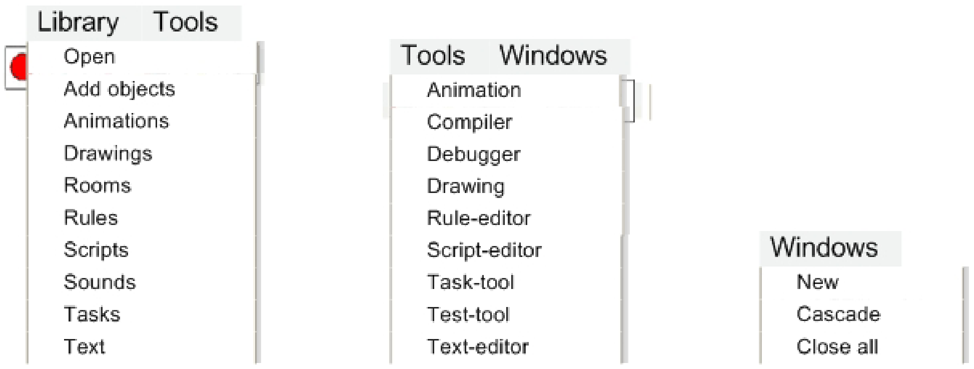


Figure 29 Library, Tools and Windows menus

📄	File: new. Creates a new project.
💾	File: save. Saves the current project.
📂	File: open. Opens a project.
●	Object: new. Adds a new object.
▶	Tools: test-tool. Tests the game.
▶	Tools: debugger. Debugs the game.
📚	Library: open. Opens the library.
✏	Tools: drawing. Opens the drawing-tool.
♩	Tools: sounds. Opens the tool to make sounds and compose music.
abcd	Tools: text-editor. Opens the text-editor.
▶	Tools: script-edit. Opens the script-editor.
?	Tools: task-tool. Opens the task and question tool.

Table 25 Explanation of the icons in the toolbar

5.6 Example on use of the program – how to make a game

This section gives a demonstration of the use of the program, and shows the program in the way I want it to function if it some day will be implemented. Not all details are presented, most because of they are not yet clear. Each step of the process of making a game in the EGE is presented.

In this example I explain how to make a simple adventure game where the player controls a character who moves around in Trondheim through history. The goal of the game is to learn the history of the city. On his way through the city, the player meets different historical persons that give valuable information needed to solve the tasks and problems the player meets.

5.6.1 Create and save project

The user starts a new game project by either pressing the new-button, or by choosing new from the file menu. The Save Project window (Figure 30) pops up; the user has to name and save his project before he can add anything.

The name of the game project is Trondheim, saved in the folder History of Trondheim.

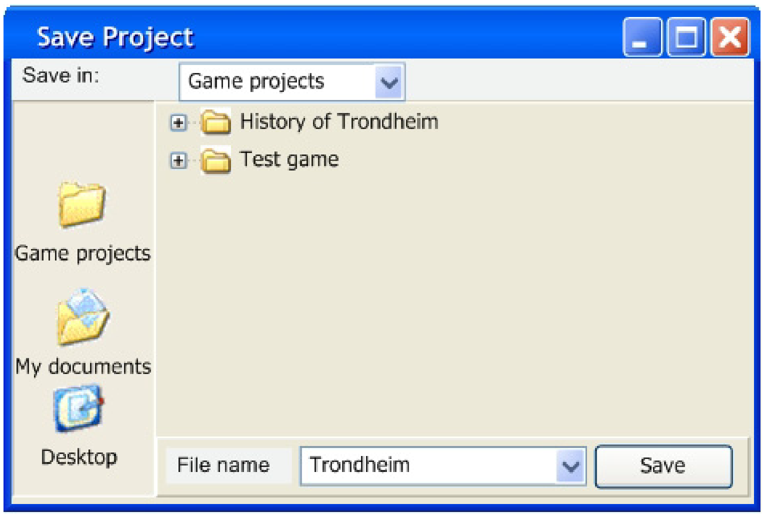


Figure 30 Save Project window

5.6.2 Create a new object

After a project is created, objects have to be created and added to the game. The user adds a new object by choosing new from the object menu. The Add Object window (Figure 31) will pop up. The first thing the user must do is to give the object a **Name**, and it is recommended that the name reflects the object in a recognizable way.

The only attributes that must be set to create the object, are the **Name** and **Class**. The class is picked from a menu, and can be player character, non-player character, room, background, and several other, the users can also define their own classes. The other attributes are optional, and can be set at a later time if necessary. When the name is

The name of the object in my example is Player, and the class is Player character.

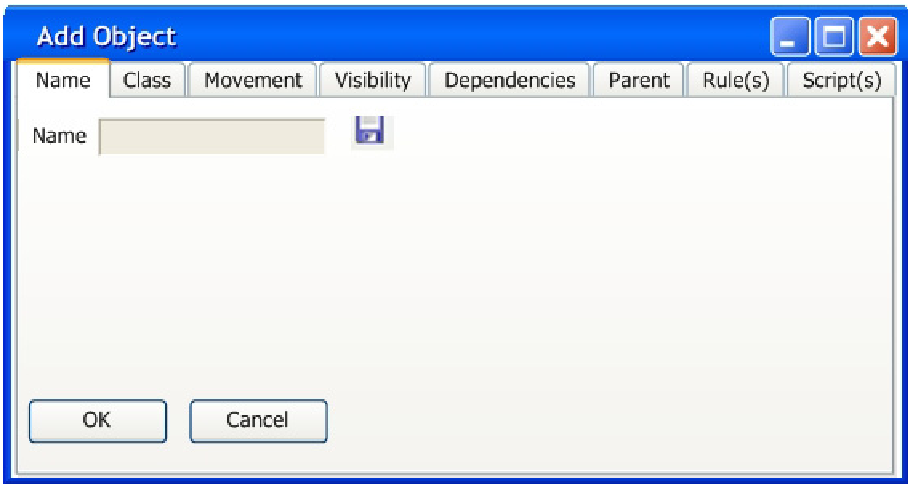


Figure 31 Add Object window

The **Movement** attribute decides if the object can move or not, and how it moves on the screen. Restrictions about the movement can also be set.

The movement attribute for the Player object is set to movable, and there are no restrictions on where the object can move.

Visibility defines if the object is visible for the players or not. Many games contain both visible and invisible objects.

The visibility attribute for the Player object is true.

In the attribute **Dependencies** the user can list other objects that the actual object is dependent on.

There are no dependencies for the Player object.

The objects can inherit properties from other objects, and these are defined in the **Parent** attribute.

The Player object has no parents.

A game needs rules to avoid chaos and mess, and the rules for each object can be defined in the **Rule(s)** attribute. The user can choose to make a new rule or use a pre-defined rule from the library.

The Player object has no rules yet; they are to be defined later.

Object can have related scripts, and they are defined in the **Script(s)** attribute.

The Player object has no scripts.

The Player is now created, but a game must contain more than one object to be fun to play. To be able to move around the player needs a "playing board". Objects of class **Room** are the areas the game takes place in the Educational Game Editor. The main room can contain many small rooms.

In my game, the main room is called Trondheim-map.

The game now contains two objects, Trondheim-map and Player. The objects appear as icons, showed in Figure 32.

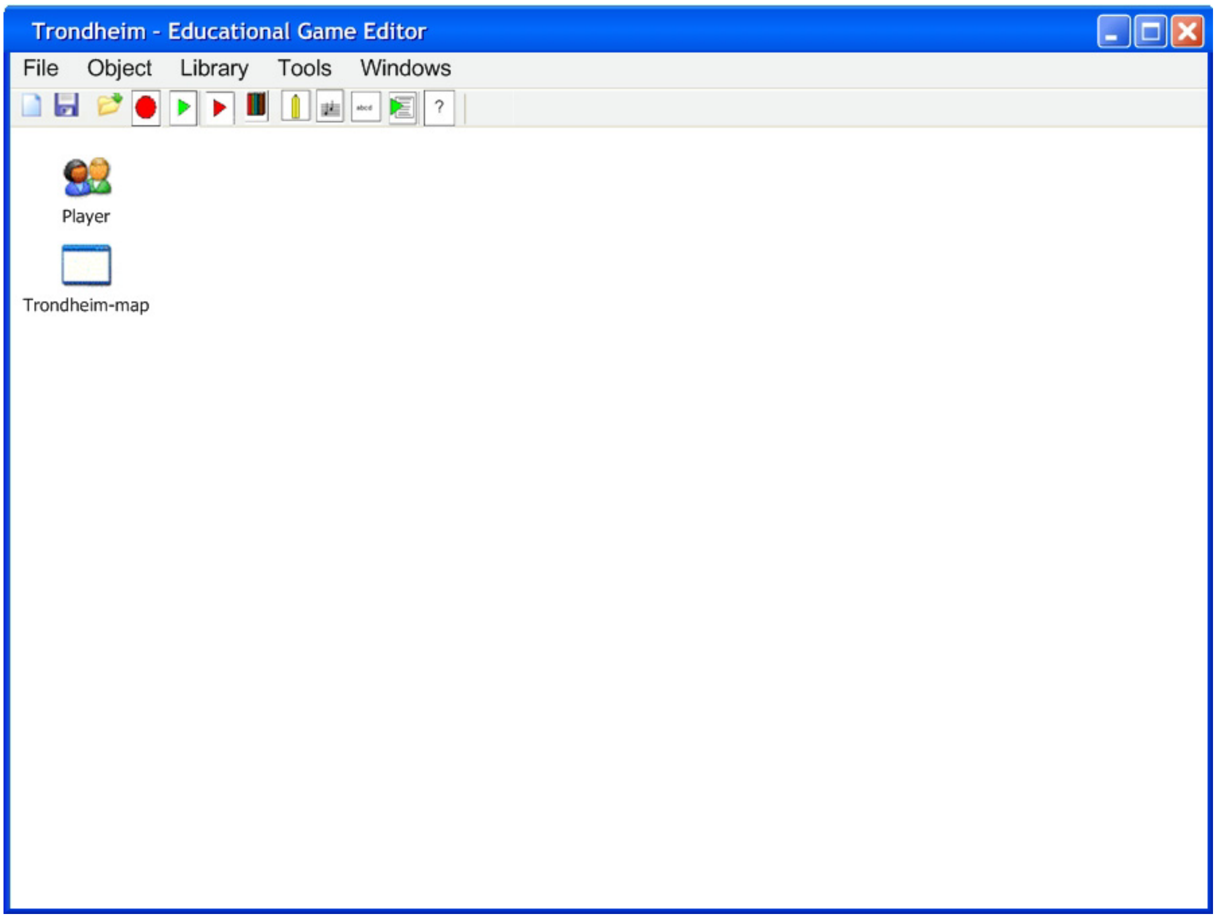
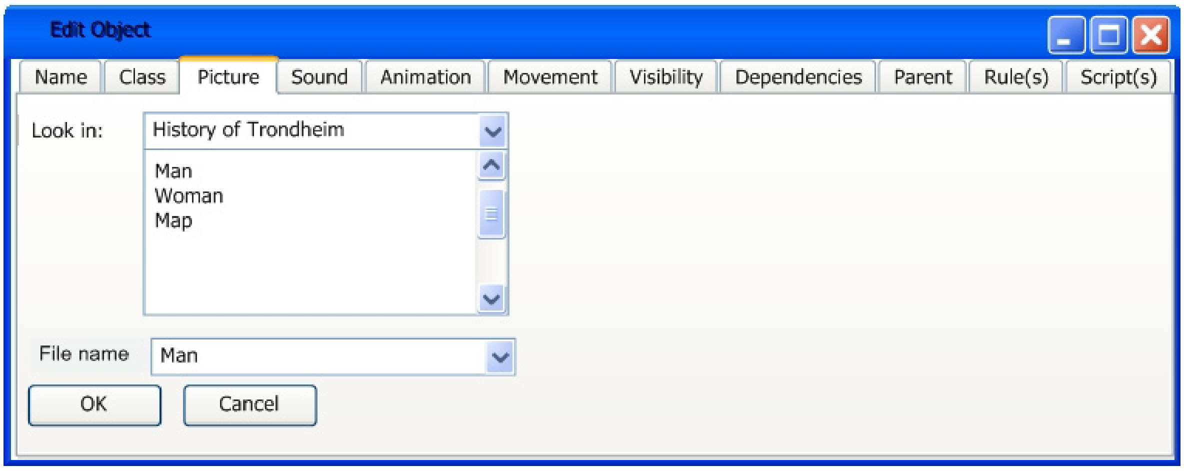


Figure 32 Main window containing two objects

5.6.3 Edit objects

The objects have not yet got any appearance. Visible objects have to be drawn or animated, and sound or music objects have to be composed. To edit an object, select the object icon and double click on it and the Edit Object window appears. After Player object has been clicked, the window shown in Figure 33 pops up. Since the Player is a visible object, the user has to define the appearance, and this is done by adding an image or graphics file to the object. A pre-made graphics object can be used, or a new created in the drawing tool.

In my example, a pre-made picture is used to define the appearance of the Player object. A pre-made picture is also used as background for the game in the Trondheim-map object. The picture added to the Player is called Man, and the picture added to Trondheim-map is called Map.



Look in: History of Trondheim

Man
Woman
Map

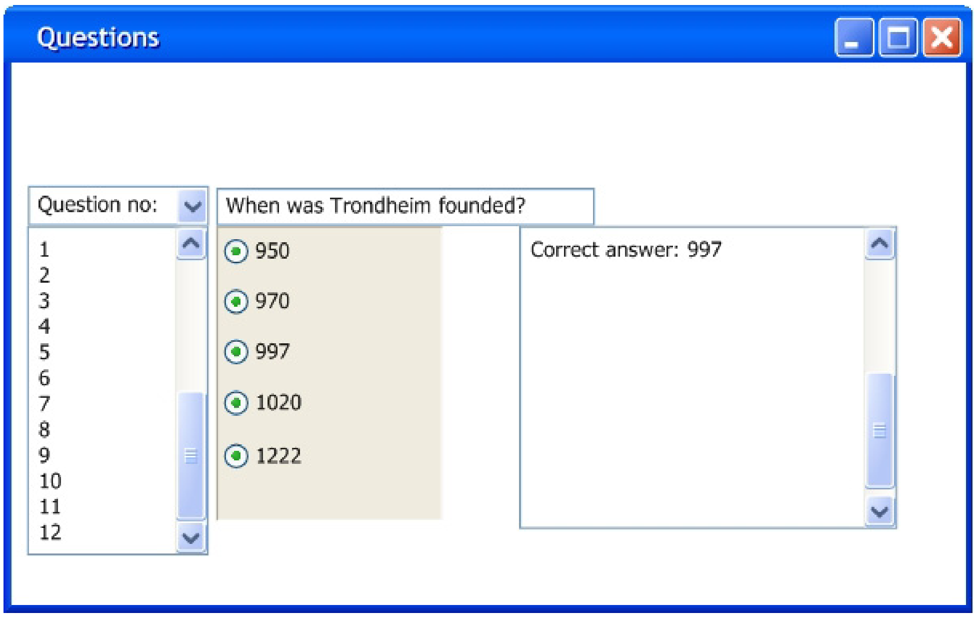
File name: Man

Figure 33 Edit Object window

The objects are now possible to view by the players, but the game is yet not very fun. Some tasks and challenges have to be added to the game to make the players entertained. Rules are also necessary to organize the game.

5.6.4 Make tasks

A game needs tasks of some way, either scoring goals in a football game, or answering questions in a quiz game. The EGE contains a tool made for making tasks; the Task-tool. The tool is opened in the Tools-menu. The tool has functions for making different kind of tasks, as questions to be answered, obstacles that must be forced, opponents that must be killed etc.



When was Trondheim founded?

- 950
- 970
- 997
- 1020
- 1222

Correct answer: 997

Figure 34 Window for making questions in the Task-tool

Figure 34 shows a proposal on how the task tool will look like when the user wants to make questions. The questions are numbered, and alternatives are made and the correct alternative has to be defined. There will be able to decide what that happens when the correct answer is given, and also when an incorrect answer is given. The tool will contain many functions not yet decided, and the figure only shows an incomplete proposal.

In my example, there are questions about the history of Trondheim. A given number of questions have to be answered correctly to advance in the game.

5.6.5 Make rules

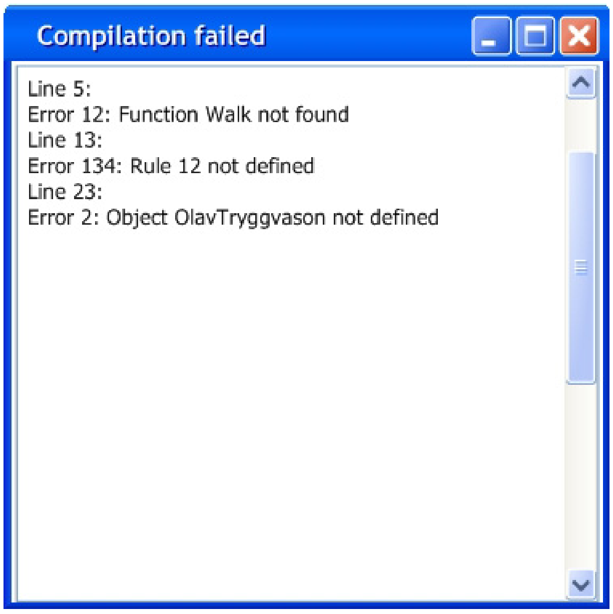
Rules are necessary in a game. The program contain a tool for making rules, and the rules can be defined to decide the behavior of one single object, how the object is related to other objects, and general rules for a whole game. The tool gives the user an overview of all objects in a game, and the user can click on an object to define the specific rule for that object, for example where in the game area a playing character is allowed to move, how he is killed, consequences of giving an incorrect answer on a question, etc. Rules defining relationships between player controlled and computer controlled objects are also necessary, for example the consequences of entering a dangerous area in the game, how to kill an enemy, etc.

The game also needs general rules for the game. This will be rules about how many points scored for killing an enemy or answering correctly on a question, what the players are allowed to do and not, for example how far a player can move in one turn.

The game in my example contains rules for where in the game area the Player is allowed to move. The Map is divided in to different areas, one that free area and one restricted. The tasks in the game is questions about Trondheim, and rules defining this says that the Player advance to the next level in the game after he has answered correctly on 10 questions on each level. Nothing happens if a wrong answer is given. The game is won when the player has completed level 10, i.e. after 100 correct answers.

5.6.6 Compile and debug game

When the game is finished, or, the user wants to test if the game is made in a correct way, he can compile the game. The compiler translates the generated source code to a runnable program. If the source code does not contain any faults, the program will compile without problems. However, if faults occur, a window as shown in Figure 35 will pop up. The line the fault occurs and the error number and explanation is displayed.



```
Line 5:
Error 12: Function Walk not found
Line 13:
Error 134: Rule 12 not defined
Line 23:
Error 2: Object OlavTryggvason not defined
```

Figure 35 Window that pops up when compilation fails

Persons familiar with programming and the source code generated in the program usually fix the faults appearing without problems. The program is, however, designed for people unfamiliar with programming, and many have problems understanding error reports given by the compiler. This problem is solved by the debugger. The debugger goes through the source code step by step, and the user can see what that happens in the program, what statements executed etc.

5.6.7 Test game

Even if the game compiled, it is not sure it will run and function as intended. A successful compilation is only indicating a source code without errors, not a game that runs as intended. The Test-tool makes the user able to test the finished game. The player can run the game and test if runs correctly. The user can stop the game, change different parameters if something goes wrong, and start the game at any state.

5.6.8 Summary

This is only a brief example of my idea about how the program will work. The need for new functions will probably occur during implementation and testing of the program, and aspects in the game design and functionality might be changed. However, this is meant to give an idea of how the Educational Game Editor actually will function as a tool used to design computer games.

6. Discussion

This chapter contains some of my views and experiences related to the project, further work and possible improvements of the system, and an evaluation of the project.

6.1 Is there a need for the Educational Game Editor?

Computers entered the educational institutions many years ago, and the use will still grow in the coming years. Computer games, however, have not been used very much. Some reasons for this are the varying quality and the price. A tool for making computer games for educational use might be the solution to this?

6.1.1 Are computer games in education a good idea?

The information society has obtained new requirements for the way the education takes place. We are surrounded by computers almost everywhere, and we are able to get information about what we desire at any time.

The pupils and students of today are raised with computer games and are called the games generation. They are used to be entertained and expect everything to be fun. Many have problems sitting on a chair and listening to a teacher for an hour. Traditional teaching with books and a teacher writing on a blackboard or using an overhead is in many cases out of date. New methods are necessary to keep the students' attention.

Another problem in education is the lack of fun and entertainment. If you ask a random pupil about the worst thing about school, a probable answer is that school is boring. The young generation of today wants to be entertained.

Computers have been a part of the school system for many years, but mainly as a tool used in subjects to solve tasks that were solved in other ways earlier, as writing an exercise in a word processor.

To meet some of the demands of fun and entertainment, computer games used in an educational context might be the solution. Most games are fun and entertaining and can keep the attention of the players for many hours. Computer games with educational context can combine the fun of the game play with the learning elements of the educational part.

A problem with traditional exercises and test, is the time from they are done and handed in to the pupils get the results. In a game, the players can get their score at any time; they do not have to wait for days or weeks to get the result. If the player is asked a question, he will get the response if he answered correctly immediately.

Surveys performed on students tell that they get more motivated when having fun, and if the studies were more fun, they would spend more time on studying. Most young people find playing a game more fun than solving traditional exercises. From this, the conclusion must be that computer games can make students spend more time on their studies, and from this, probably achieve better results and learn more.

The technological development is not possible to stop, and this will affect the way education takes place in the future. The use of information technology today is probably only a small beginning of what that will take place in the future. I do not think e-learning and computer games will totally replace traditional lessons. The human aspects are important, and the use of computers is impersonal and not social in the same way as discussion between the students and the teachers.

6.1.2 Potential users of the Educational Game Editor

A way to achieve both low price and decent quality for the games is to let the teachers make their own games for use in class. This gives the teacher the full control over what the pupils are learning, and he can also choose how difficult and demanding the games are. A program for making computer games for educational use would be a good idea in this case. Instead of making a traditional exercise to the class, the teacher can make a game. This will be much more fun for the pupils, and the motivation and concentration will probably increase.

Teachers are the main target group of the program, as teachers usually make tests and exercises to their students. But also other people in the education business might use the Educational Game Editor (EGE) to design games. Authors of textbooks might want to attach games that reflect the contents to their books, and using a tool as EGE to make such games is much cheaper and faster than designing a game from scratch.

Students and pupils can also make their own games using the EGE, either as an exercise given by the teachers for their own use, or games for other students to play. Many learn better from making their own things than only using pre-made material, and making computer games is a fun and motivating activity for the students.

6.1.3 Subject areas

Educational computer games have so far been made mainly for use in the lower classes of the primary school. Simple games for learning the alphabet or adding numbers are examples. The situation in higher education is different. Some games exist, but the potential for using games in higher education is not utilized.

The possibilities for game based learning in higher education are many. Social sciences might have the largest potential. Adventure and simulation games can reflect the situations the students learn about in their studies. In language subjects, games to learn grammar is easy to make, and can be much more fun than doing exercises the students have to hand in to the teacher or check the solutions in a book. Games will respond immediately if the tasks are answered right or wrong.

In subjects like history and geography, many game ideas exist. Strategy games where the players have to solve historical tasks and questions to conquer neighbour territories, or simulation games that simulate wars and learns the player how the war took place are examples.

The subject area that causes most problems to most pupils and students, are scientific subjects like mathematics and physics. Many have problems understanding how to multiply two numbers, and even more difficult; differentiation, integration, differential equations and other fields in mathematics. When a person does not understand a concept or a rule, he might get frustrated and loose the motivation. The result is often that nothing is learned. Games to learn mathematics might be a good solution for persons with problems understanding the subject.

6.1.4 Different pupils, different needs

The school system in Norway is dominated by the thought of the unit school. Instead of giving the best pupils challenges, and make it easier for the weakest, the idea of an average pupil has been the basic idea.

This idea is, however, not the case in the real world. Pupils and students are different human beings with different needs. The most gifted and the weakest students cannot be happy if they get the same exercises. Individualized training is a solution on this problem, and computer games make it easy to achieve this. A computer game might have different levels, from easy to expert. This makes it able to adapt the games to all groups of users.

6.1.5 The market

After a search on the net, I did not find any programs made particularly for making educational computer games. I am not sure if anyone has tried to produce and market such a program before, but it seems like the market is open.

The program should not be very expensive to develop, and the price will probably not be a hindrance for interested customers. Every school that are interested should have the funds to buy the Educational Game Editor.

The education business is a global industry, and different versions of the program can be made to fit different demands in different parts of the world. I think the program has a big potential and the program will be worth to realize.

6.2 Further work

This project is only the start of the developing process of the Educational Game Editor. It is a foundation for further development, and I hope it will be realized.

I hope the program will be implemented. The project is only in the initial phase, and a lot of work remains before the program is finished. But an implementation of the system will require more than one person, as the size of the program will require a lot of work. The development of the program might be fitted as a group project for students in software engineering or other related subjects.

The idea is not perfect and complete, improvements and additions will probably be necessary. New requirements might occur, and this may result in new functions and tools in the program.

7. Conclusion

The project started with many different ideas around the subject e-learning and use of computer games in education. After some discussions with my teaching supervisor, the idea of the Educational Game Editor emerged. No such program existed, and the need is absolutely present.

I had different goals for the project, and the first was to explore different learning theories and relate them to computer game based learning. The different theories are relevant for different kinds of computer games, and it is not one single solution on which theory that supports one class of computer games.

Hypermedia is an efficient and exciting way of organizing data. Educational contents require different kinds of information, and they often have to be linked together. Hypermedia is maybe the best way of organizing data in e-learning systems and educational computer games.

There exist many different computer games for use in education today, and the requirements for making such games are many. A program for designing computer games for educational purposes will make everyone able to make their own games that fulfil the requirements. This has resulted in the design of the Educational Game Editor; a program I hope will make it possible to make educational games of decent quality in the future.

8. References

[Albinussen 2003] Albinussen, Trond (2003). *"Navigering i internet og hypermedia"* Master thesis, Department of computer technology and information technology, NTNU Trondheim

[aoc.idi.ntnu.no 2006] Age of Computers (2006). Website: http://aoc.idi.ntnu.no/

[Baillie-De Byl 2004] Baillie-De Byl, Penny (2004). *"Programming Believable Characters for Computer Games".* Charles River Media

Bass, Len, Clements, Paul, and Kazman, Rick (2003). *"Software Architecture in Practice".* Addison Wiley

[Education Arcade 2004] The Education Arcade (2004). Website: http://www.educationarcade.org

[Eidsmo, Kolås 2005] Eidsmo, A. and Kolås, Line (2005). *"Kap 1 – IKT og læring - skrevet for bruk i faget Datastøttet læring 2005"* Høyskolen i Nord-Trøndelag

Friis, Nicolai (2005). *"Computer game based learning- SimComp".* Master thesis, Department of computer technology and information technology, NTNU Trondheim

[Friis and Tollefsrud 2004] Friis, Nicolai and Tollefsrud, John Ola (2004). *"Computer Game Based Learning, Studies and Learning"*, Department of computer technology and information technology, NTNU Trondheim

[gamemaker.nl 2006] *"Game Maker Pages"* (2006). Website: http://www.gamemaker.nl/

Hallford, Neal, and Hallford, Jana (2001). *"Swords & Circuitry: A Designer's Guide to Computer Role-Playing Games".* Prima Tech

[it2202 2006] *"IT2202 Operativsystemer"* (2006). Website: http://www.idi.ntnu.no/emner/it2202/

[Kasvi 2000] Kasvi, Jyrki J.J. (2000). *"Not Just Fun and Games – Internet Games as a Training Medium"*, Helsinki University of Technology, Laboratory of Work Psychology and Leadership, Helsinki, Finland. Website: http://www.knowledge.hut.fi/people/jkasvi/NJFAG.PDF

[Lowe, Hall 1999] Lowe, David and Hall, Wendy (1999). *"Hypermedia & the Web – an engineering approach"* Wiley

[Mann et al. 2001] Mann, Barry D., Eidelson, Benjamin M., Fukuchi, Steven G., Nissmann, Steven A., Robertson, Scott, and Jardines, Lori (2001).
"The development of an interactive game-based tool for learning surgical management algorithms via computer".
The American Journal of Surgery 183 (2002) 305-308

[Michaelson et al. 2000] Michaelson Rosa Helliar Christine Power David and Sinclair Donald (2000).
"Evaluating FINESSE: a case-study in group-based CAL"

Computers & Education 37 (2001) 67-80

[Natvig, Line 2004] Natvig Lasse and Line Steinar (2004).
"*Age of Computers – Game-Based Teaching of Computer Fundamentals"*.
ITiCSE '04 June 28-30, 2004, Leeds, UK.

[Norman 1993] Norman, David A. (1993). *"Things that Make Us Smart: Defending Human Attributes in the Age of the Machine.* New York: Addison-Wesley

[Overmars 2005] Overmars, Mark (2005). *"Game Design in Education"* Institute of Information and Computing Sciences Utrecht University, Utrecht, the Netherlands

[Pohl, Prenner, Purgathofer 2006] Pohl, Margit, Prenner, Peter, and Purgathofer, Peter.
"Hypermedia in Education – Monitoring the Development of Hypermedia Documents"
Internet article, website:
http://igw.tuwien.ac.at/igw/menschen/pohl/yorkzwo.html

[Prensky 2001] Prensky, Marc (2001). "*Digital game base learning*" New York McGrawHill

Roblyer, M.D. (2005). "*Integrating Educational Technology Into Teaching"* Pearson Merrill Prentices Hall

[Roubidoux et al. 2002] Roubidoux Marilyn A. Chapman Chris M. and Piontek Mary E. (2002).
"*Development and Evaluation of an Interactive Web-based Breast Imaging Game for Medical Students"*.
Academic Radiology, Vol 9, No 10, October 2002 1169-1178

[Smaldino et al. 2005] Smaldino, Sharon E., Russel, James D., Heinich, Robert, and Molenda, Michael (2005). *"Instructional Technology and Media for Learning"* Pearson Merill Prentice Hall

Sommerville, Ian (2001). *"Software Engineering"* Addison Wesley

[Warholm 2000] Warholm, Hans-Olav (2000). *"Hypermedia i undervisningen"* Master thesis, Department of computer technology and information technology, NTNU Trondheim

Wehus, Håvar and Berg, Arnstein (2005). *"Læringsrom i nett"* Master thesis, Department of computer technology and information technology, NTNU Trondheim