Age of Computers – Game-Based Teaching of Computer Fundamentals

Lasse Natvig
Computer and Information Science,
Norwegian University of Science and Technology (NTNU),
N-7491 Trondheim, NORWAY
+47 7359 3685
Lasse.Natvig@idi.ntnu.no

Steinar Line
Computer and Information Science,
Norwegian University of Science and Technology (NTNU),
N-7491 Trondheim, NORWAY
+47 7359 3680
Steinar.Line@idi.ntnu.no

ABSTRACT
Age of Computers (AoC) is a new approach to the learning activities that supplements the auditorium lectures in a computer fundamentals course with 250 students. It is a computer game that presents the students a diverse set of problems from the course topics linked to computer history. It is implemented as set of dynamic web pages retrieved from a database. A prototype was used in 2003, and the feedback is positive and a strong motivation for continuing the project. The paper describes AoC, its use and implementation.

Categories and Subject Descriptors
K.3.1 [**Computers and Education**]: Computer Uses in Education – *collaborative learning, computer-assisted instruction (CAI), computer-managed instruction (CMI), distance learning.*

General Terms
Management, Design, Experimentation, Human Factors.

Keywords
Game-based teaching, edutainment, computer games.

1. INTRODUCTION
During development of a new course in computer fundamentals we decided to develop a radically new approach to the student activities that supplements the ordinary auditorium lectures. Traditionally, such activities have been a set of theory problems handed out to the students once a week with at classroom walkthrough by a teaching assistant a week or two later. By replacing the traditional "paper-exercises" by a diverse set of problems integrated in a computer game-like setting – we hoped to increase the students' motivation and thereby improve learning. The course "computer fundamentals" is given to approximately 250 students in their second year of a 5-year Master of Science study in Information Technology at the Norwegian Institute of Technology and Science (NTNU).

ITiCSE '04, June 28-30, 2004, Leeds, United Kingdom.

The students take about 35 courses in total. One or two of the courses have HW lab assignments with supervision of teaching assistants. Many of the courses involve project work in groups, and most of the courses include paper exercises.

We believe that students in general to an increasing extent want new and more diverse learning activities. We also believe that computer history, besides being interesting in itself, might be used to give a deeper perspective on the tremendous improvements we have seen in computer technology. These considerations led to our development of *Age of Computers (AoC)*.

AoC is a web based exercise environment for our new computer fundamental course. It is also a computer game where the player moves through the computer history from "the early mechanical computers", visiting "the vacuum tube age", "the transistor age" etc. ending up solving problems related to embedded systems of "the present age".

The Norwegian society has in general a very large density of personal computers, mobile phones and other modern IT equipment. Computer games are very popular among young people, and most students have their own PC at home. This environment makes it really demanding for the thick textbooks and traditional classroom teaching to compete for the students' attention. AoC is an attempt to meet this challenge.

One of the sources of inspiration has been real-time strategy games such as Age of Empires II: The Age of Kings which has been sold in more than two million copies [1]. In addition to being a fun game, it provides a lot of textual, multimedia and interactive learning experience from many disciplines, not only history. Most impressive is perhaps the advanced graphics and user interface. Our design team had a wish to offer an advanced GUI for the students playing AoC. However, strong limitations of both time and budget forced us to choose simple technical solutions in developing an AoC prototype. It was also a goal that the AoC framework should be possible to adopt by other developers of new course material, not necessarily within computer science.

The AoC project is grounded in the strategy "ICT in learning" supported by our department and NTNU. Both parts have contributed with funding, and the development work has been done by faculty, PhD students and graduate students. AoC was used during the autumn semester 2003 and a questionnaire given to the students taking the course has shown that we have met many of our goals.

The paper is organised as follows. Section 2 gives an overview of AoC with a focus on how it is perceived by the students. Section 3 presents the main types of problems that the players encounter in AoC and is followed by an overview of the main technical solutions in the prototype. Section 5 gives preliminary results from a survey among the students having used AoC for more than two months. The paper ends with a few concluding remarks and a few words about the plans for the project in the future.

2. AGE OF COMPUTERS – OVERVIEW

One of the main thoughts behind AoC is to teach computer fundamentals using history as a governing idea. The goal of the AoC player (a student) is to walk around in different *epochs of the computer history* and earn points by solving problems. The player starts outside the NTNU main building, but after a short introduction to the game she travels backwards in history and meets Charles Babbage working on his Difference Engine. Babbage and other computer pioneers give advice to the player throughout AoC.

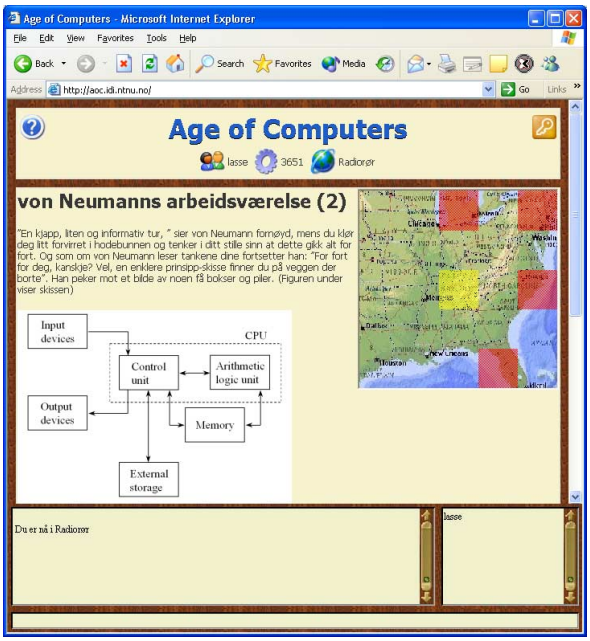

Figure 1. The Graphical User Interface in AoC.

As soon as the player has solved enough problems (and earned enough points) in the "Age of Mechanical Computers" she is allowed to proceed to the next age, "The Age of Vacuum Tubes", and later "Transistor Age", "Integrated Circuits", the "Modern Age" and finally into the future (The "Pico Age"). This controlled progress of a player through the game makes it possible to monitor the learning progress of the students during the semester. The GUI of AoC is shown in Figure 1. (The text is in Norwegian).

AoC is presented to the player as dynamic web pages with a uniform layout. The content is organised in *maps* containing a number of *rooms*. Each room (such as "von Neumanns arbeidsværelse (2)" in Figure 1.) typically contains some text and images giving the right background for presenting a set of problems to the player. The current room is displayed in the main window, which also shows the map in its upper right corner. On the map a yellow square shows the location of the current room, and red squares give the location of other rooms with problems (exercises) to be solved by the player. The player can move around by clicking on the map or by following links in the text. The availability of rooms depends on the player's progress in the game – this is to ensure a pedagogical order of introduced topics that fits with the pace of the lectures.

When entering a new room, the player typically reads through the text and tries to solve the given problems. Often this requires a lookup and a bit of reading in the textbook or information search on the Internet. When the player enters an answer she gets immediate response. If the answer was correct the number of earned *points* is updated. On top of the screen, the name of the player, number of earned points and the current map is shown. In addition there are a Help-button and a log-out button.

The AoC players also have access to a chat window at the bottom of the screen. There is *one* chat-channel for each epoch in the game. Typically it is used by the students to ask for help when encountering difficulties. To the right of the chat window all the players active in this map are listed. Often students help each other, but also teaching assistants "walk around in AoC" watching the chat window and providing help. This is especially true during the time scheduled lab hours 15 hours a week where the TAs supervise the AoC players face-to-face. During the semester we have had three *deadlines*, each requiring that the student has reached a specific room no later than a give date.

We developed a simple XML language to represent the room content. In addition to presenting text and images we have mechanisms for presenting different kinds of problems to the player as described in Section 3. The possibility of closing and opening rooms during the semester made it possible to create many of the rooms *after* the semester had started. Small changes to active rooms could also be done dynamically. It was an overall goal when we filled the rooms to make the content independent of the book [8] currently used in the course. This is because textbooks are typically changed every third or fourth year, while we hope that AoC will get a much longer lifetime.

3. PROBLEM CLASSES & EXAMPLES

It is currently some more than 500 different problems stored in AoC, and the number is expected to increase in future versions. The kinds of problems met by the player can be categorized into four groups.

3.1 Multiple choice

The majority of the problems fall into this category. When authoring such questions we strive to make them as pedagogical as possible. We try to make the questions such that the player has to read *all* alternatives before deciding what the correct one is. Often we give 3 or 4 correct claims about a topic, and one that is wrong. Sometimes more elaborate calculations must be done to find the right alternative.

An important aspect of the mechanism used to present the problems is that the player giving a wrong answer has to wait for a certain amount of time before she can do a new attempt. The students dislike this the first time they experience it. However, when they understand that it is deliberately made so to avoid a dumb "trial and failure approach", and instead motivate for a "read, think and try approach", they accept it.

3.2 "Number problems"

These are problems where the answer is given as a number, such as for instance a calculation of the mean access time to bytes stored on a specified hard disk. Perhaps surprisingly, most of these problems have binary numbers as answers. Binary numbers, the binary representation of integers, negative numbers and floating point numbers are in itself relevant topics that naturally fall into this category.

Another set of problems we have found very useful are to check the understanding of the addressing modes used in computers. By displaying a small set of memory locations with addresses and contents together with a few named registers with contents, it is straightforward to generate many different problems testing the students' knowledge of the topic.

We have found this mechanism to be very versatile and a significant step forward from the simple multiple choice mechanism. We expect to use it for many different new "styles" of questions in the future.

3.3 ALU (processor) control signals

This category was specified as a special kind of problems but is in fact implemented with the number problem mechanism. We have experienced in earlier classes that a detailed understanding of the organisation and behaviour of a simple arithmetic/logic unit (ALU) and data path can be checked by asking the students to give the exact control word for a given micro operation. Also in this case, it is straightforward to generate a large number of questions using relatively simple means.

We are aware of simulators that let the students see the control and dataflow inside simple ALUs or processors and would like to integrate one of these into AoC.

3.4 DARK assembler

There is a tradition at NTNU that the computer fundamentals course contains a significant portion of assembler coding. Although it is being used less frequently in the industry, we still believe it is necessary to give the students a profound understanding of the computer. Also at this level, several simulators are freely available. We decided to use DARK, a set of virtual machines developed by Ola Ågren at Umeå University in Sweden. One of the very nice features of DARK is that it offers four simple processor architectures in the same user interface. The architectures are *stack machine, load-store machine, memory-memory machine* and *index-machine* – all with their own simple and well documented instruction set [9].

The typical working method for a student solving a DARK problem is to study the problem definition and then write an assembler program for the specified architecture. DARK has a simple text edit window for this purpose. In some cases we choose to give the students a flying start by providing an incomplete assembler program in the AoC room presenting the problem. With cut & paste, and a bit of editing, the student is ready to test the program. In the editing window the architecture is chosen, and then by clicking the "Parse code" button a window such as Figure 2 appears.

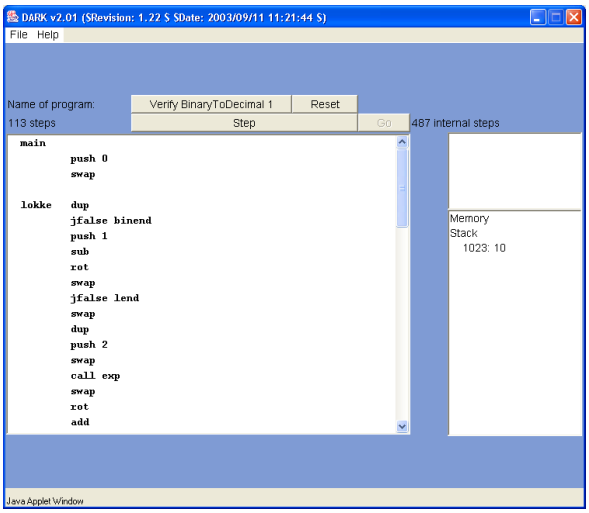

Figure 2. User Interface in the DARK simulator.

In AoC, the state of the DARK computer is typically preset with a few values in memory so that the student can test her program simply by clicking the step button. Very often, a few single-steps reveal logical errors in the program and the student re-enters the editing window for correcting these.

When the program seems to work correctly it is checked by a special AoC evaluation mechanism: For every student AoC generates five random test cases. Then the user must press the "Verify problem" button to see whether it solves each of the five test cases correctly. Many students find this five stage testing a bit exciting. If it passes all five tests it is assumed to be correct, and the problem is registered as solved. Since problems of this kind normally require much more effort than multiple choice questions it will typically give much higher payment in points.

4. TECHNICAL SOLUTIONS

In the initial phase of the AoC project a prestudy was carried out where different technical solutions were inspected. One of these was a framework for a MUD (Multi User Dungeon). Thereafter, mostly due to time and budget limitations, we decided to implement a web based interface where the only interaction among users was via the chat system. In future versions of AoC it is desirable to introduce other ways for the students to interact and influence on each others playing. Many examples can be found in online computer games.

4.1 The AoC Software Architecture

The interface of AoC was written in C# ASP.Net [2] (see Figure 3). To implement the interactive chat window a Java [4] chat applet was also created. A chat server coordinates the communication between the chat applets of the users. It keeps each chat channel's content, so when a student logs in, her chat applet will show the last contributions to that channel.

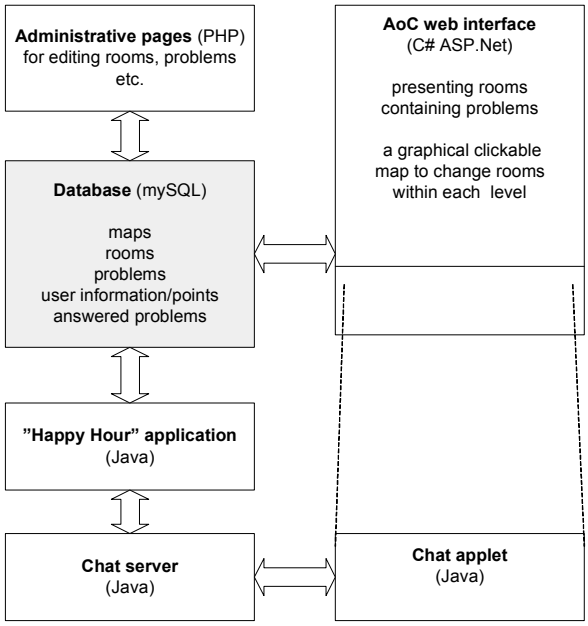

Figure 3. The AoC Software Framework.

AoC has a special mechanism called "Happy Hour" for presenting simple problems in the chat window. The concept was developed to motivate the students to use AoC during the time scheduled lab-hours. The Happy Hour presents a multiple choice question in the chat window every second minute (adjustable) giving the students a chance to earn extra points beyond those in the game. The system has been running five hours a week all semester and has shown to give a positive influence on students working in lab-hours (although many has instead logged in from their home location which we think is quite all right).

4.2 The AoC Database

MySQL [5] is used to implement the AoC database. It contains:

- Information about the content of each room including problems (XML presented on the web page by C#).
- Information about each epoch and the placement of rooms on the map used in that epoch.
- Information about users and user activity. This includes among other things which questions each student has answered right or wrong. It makes the system able to behave differently for each student depending on her progress in AoC.

From the content of the database much interesting data may be found about how and when the students work with AoC. We plan to investigate these to improve both the content and the behavior of the system.

4.3 The AoC Tool Set

During the development of AoC we identified needs for various kinds of tools to support both the development and use of AoC during the semester. They can be divided into three groups.

- Tools for *entering course contents* into the game environment. This includes a room editor and a question editor, and also tools to inspect which question (problem) is used in a given room. We have also made a simple map/room editor to create and place rooms onto maps.
- Tools for *helping the course staff* to run the course, and monitoring the progress made by students during the semester. This includes a tool presenting how many has reached the goal of the next deadline.
- Tools for *helping the students navigate* throughout the whole game. This includes a graphical presentation of the progression of a student from the first room in the game giving links to each room she has been to, and showing the remaining path to the next deadline. There is also possible to list out the rooms in alphabetical order to ease searching for a specific room.

Because of varying competence in using C# among the developers in the project, the administrative tools of AoC were written in PHP [7].

5. EVALUATION OF THE PROTOTYPE

To assess the educational value of using AoC in the new Computer Fundamentals course the students were asked to anonymously complete a questionnaire with 35 questions. We summarise a few results in this section. About 55% of the 246 enrolled students have filled in the web based questionnaire – all given anonymously. Many of the questions were followed by a field for "additional comments". 24% of the respondents are females.

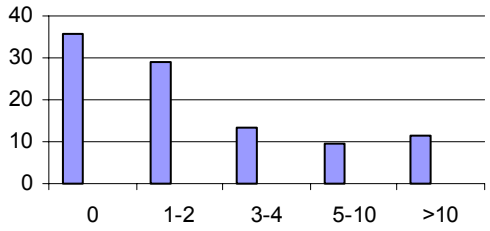

Figure 4. Hours/week spent by students on computer games.

In our first question we wanted to get a rough measure on the amount of time students spend on ordinary computer games. On the question "*Give a rough estimate of the number of hours you are using through the week on different computer games (not AoC)"* the students were given five alternatives as shown in Figure 4. The distribution of the answers is given in percentage.

On the statement "*AoC is more motivating than traditional exercises"* close to 95% strongly or somewhat agreed! In this, and most other questions, we used a Likert scale by posing a statement and asking the students whether they 'Strongly Agree (A)', 'Somewhat Agree (a), 'Undecided (?) ', 'Somewhat Disagree (d)' or 'Strongly Disagree (D)'.

Slightly more than one third of the students do not play computer games regularly. We interpret this fact combined with the quite positive feedback on the motivational effect of AoC as a strong indication that the AoC GUI and playing rules are simple enough for *all* students. One student clarified what probably is the most important reason for the increased motivation in only two words; "*instant gratification"*.

A crucial aspect that is difficult to measure is how much the students learn from using AoC. On the statement *"I perceive the learning effect by using AoC as good compared to traditional exercises"* we got the percentage distribution shown in Figure 5.

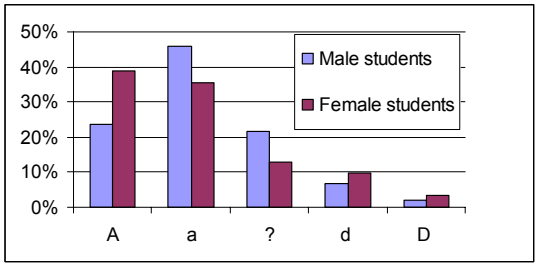

Figure 5. Student's opinion on learning effect (see text.)

The majority of the students clearly feel that they learn more through AoC than by traditional exercises. This is even more promising since another question showed that the students use an equal amount of or slightly less time on AoC compared with the time used on exercises in other courses. Also, the female students are slightly more positive to the learning effect in spite of being less used to computer games! Preliminary results from a more recent questionnaire on the perceived learning effect show that the students consider the time used on AoC as clearly more efficient than attending lectures or reading in the textbook. AoC is made available to the students through the intranet used by all students and employees at the University, and about 100 persons *not* following the course have tried AoC during the semester!

As expected, the DARK problems are the most time-consuming and difficult problems in AoC. We included some really difficult but optional problems to challenge the most eager students. An interesting, but perhaps not surprising, result from the AoC questionnaire is that the DARK problems both are judged as the most popular by some students and the most disliked by other students!

We also asked the students to suggest technical improvements. They gave highest priority to better possibilities for cooperation among players, more figures and images and better playing rules. Fancier layout or inclusion of sound were given low priority by the students. The results from the questionnaire will be further analysed and reported in a later publication.

6. CONCLUSION AND FURTHER WORK

AoC has been very well welcomed by the students as a new approach for course activities supplementing ordinary lectures. We are not aware of other similar work, but recently a workshop titled "Using History to Teach Computing" was held at the FIE-03 conference [3]. AoC is clearly motivating, and the results so far are promising with respect to an increase in the overall learning. We have good experience in conducting HW lab assignments for big classes with more than 400 students in a previous version of the computer fundamentals course [6]. In comparison, AoC is simpler to administrate, but we do *not* think that AoC should replace such hands-on experience. However, as an alternative to traditional paper based exercises we believe that AoC exemplifies a new learning approach that will become more widespread in the future. The possibilities are numerous, and we think AoC is a small but early step in this new direction.

From the beginning of the AoC project it has been necessary to do a strict prioritizing of features to include, postpone or reject. In addition, the questionnaire has given us a lot of ideas for further development of AoC. A straightforward enhancement will be to add more course contents with related problems in AoC. This will increase the learning potential and also make the "AoC world" richer, more diverse and less predictable – making it more fun to play. A more challenging extension is increased interaction and cooperation among players. The positive feedback motivates us to continue the project and an enhanced version will be used in subsequent versions of the course.

The area of e-learning and edutainment is very exciting. The technology offers a rich set of possibilities, and computer games may be an important factor in making e-learning more successful. We believe Age of Computers is a small but early step in this new direction.

7. ACKNOWLEDGMENTS

We would like to thank Ola Ågren for letting us use the DARK source code, Bård Kjos for strong administrative support at our department (IDI), and NTNU for financial support, and finally but not least the many contributors to AoC (in alphabetical order); Tanveer Hussain Awan, Asbjørn Djupdal, Bård Terje Fallan, Ole Andreas Hegle, Ole Kristian Hoel, Lars Ivar Igesund, Per Kristian Lehre, Bjørg Peggy Sveen Lyngstad, Simon Thoresen, Håvar Valeur, Jeppe Andreas Berg Weinreich and Olaug Kyoko Namba Østhus.

8. REFERENCES

[1] Age of Empires II: The Age of Kings. http://www.microsoft.com/games/age2/

[2] ASP NET. http://msdn.microsoft.com/vcsharp/

[3] FIE-03 - Workshop on Using History to Teach Computing. ASEE/IEEE Frontiers in Education Conf., Nov. 2003.

[4] JAVA. http://java.sun.com/

[5] MySQL. http://www.mysql.com

[6] Njølstad, Tormod and Natvig, Lasse, *Experience from a 450 Students/Year Course on Digital Logic and Computer Fundamentals using FPGAs and microcontrollers*, Proc. of 2001 Int'l conf. Microelectronic Syst. Education, pp.18-19.

[7] PHP. http://www.php.net

[8] Stallings, William. *Computer Organization & Architecture*, 6th edition, Prentice Hall, 2003.

[9] Ågren, Ola. *Virtual Machines as an Aid in Teaching Computer Concepts*. In Workshop on Computer Architecture Education 2000. See http://www.cs.umu.se/~ola/Dark