Age of Computers II—An Improved System for Game Based Teaching

Asbjørn Djupdal* Lasse Natvig†

Abstract

Age of Computers (AoC) is a web based multi-player game for teaching computer fundamentals at university level. It is a replacement for traditional paper exercises, and supplements auditorium lectures. The main motivation behind AoC is to try using computer game features to motivate students to learn computer fundamentals.

The first version of AoC was thoroughly tested on students autumn 2003. An improved version is to be tested autumn 2004. Improvements include more game features and better on-line course contents that is easier to navigate through for the students.

Keywords: Game based teaching, Learning by doing, Collaborative learning, Educational Computer Game, Edutainment, Distance learning

1 Introduction

Age of Computers (AoC) is a web based exercise environment developed at NTNU. It is developed for, and used in the course *TDT4160 Datamaskiner grunnkurs* at NTNU. This is an introductory course about computer fundamentals for university students. The students learn how a computer and its peripherals work at the hardware level, and the goal is to give the student a broader understanding of computers than typically given in a programming course. The course is given to classes of 200–350 students with varying degree of knowledge to the subject.

We identified two main challenges with such courses: (a) Handling the large class of students in a way that both ensures good learning and is practical to administer. (b) Motivate so that a larger part of the students become willing to "dive deep into the hardware" to get an increased understanding of the inner workings of computers.

Large Number of Students

For a course with several hundred students, the exercises become important. The distance between the lecturer and the student is large in a traditional lecture, so the lecturer will not be able to help the individual students.

*Asbjørn Djupdal, Computer and Information Science (IDI), Norwegian University of Science and Technology (NTNU), djupdal@idi.ntnu.no

†Lasse Natvig, Computer and Information Science (IDI), Norwegian University of Science and Technology (NTNU), lasse@idi.ntnu.no

The number of students is a challenge when it comes to guiding and correcting exercises. Typically, many teaching assistants would be required to give this service to the students, which is both costly and inconvenient.

Facing this challenge, we propose using a web based system for automatically correcting exercises and to give on-line help. This would automate most of the administration of the student exercises, and more important, it makes it easy for students to help other students in a goal-oriented and flexible way.

Motivating Students by Instant Response

Today's students are not like students some decennia ago. As the media industry competes for the attention of the population, the audience get more and more used to being entertained in an increasing number of situations—including education. The possibilities given by improved computer technology has given a large interest in e-learning, and the combination of education and entertainment (edutainment) has become a hot topic [10, 15].

The pace of the society is generally regarded as increasing, and the students are mostly recruited from a group of people often termed at the "restless youth". We are convinced that for many kinds of exercises most students prefer instant feedback in an graphical user interface rather than written notes on a piece of paper several days or even a week after the exercises was delivered. The popularity of such "instant gratification" was confirmed by a student questionnaire.

One could argue that universities should fight this change of society instead of adapting to it, but we think it would be a lost battle. Our responsibility is mainly to teach computer fundamentals, and we must do so with the methods best suited to the intended audience.

Traditionally, students get their theoretical knowledge from attending lectures and reading text books. They apply this knowledge to exercises that typically takes days or even weeks to get corrected. This lack of rapid feedback while studying might be considered discouraging for many of the modern students.

In contrast, computer games are very popular for the same type of people. It is observed that some computer games makes people spend significant amounts of time learning and exploring the game. An exciting multimedia user interface might have something to do with it, but we believe the instant response from solving tasks together with the entertainment value is part of the reason for the popularity. Many games are very addictive, and the player often acquires quite a bit of knowledge when playing. It should be possible to make a computer game with the same motivational factor, but where the students learning efforts result in knowledge of computer fundamentals.

A Modern Solution

AoC is our attempt at designing an exercise system which deals with the before mentioned challenges. The main property of AoC is that it is designed as a computer game. The students logged into AoC are walking around in a virtual world and are playing themselves through the exercises in the course.

The first version of AoC [13, 14] was extensively tested during the one-semester course by 250 students autumn 2003. We got positive feedback from the students, but we were not completely satisfied with it. This paper describes our second version of AoC, which is to be tested on students autumn 2004.

Related Work

Much of the reported work on the use of computer games in education has been at the primary and junior high school levels, but there is a growing interest at higher educational levels [9, 12]. Best known is perhaps the MIT Games-to-Teach Project [11] (which now is part of The Education Arcade [3]), where they have designed prototypes of games to support learning in advanced mathematics, science, and engineering. In a broader sense, computer games has definitely become a topic in higher education. Examples from Scandinavia are the new Center for Computer Games Research at the IT University of Copenhagen [1], and the IT University of Gothenburg where they have study programmes called "Entertainment Design & Technology" and "Interaktiva simuleringar och spel" [4]. A useful place to start reading about computer games is `http://www.game-culture.com/`

Paper Outline

The paper starts with explaining the conceptual idea behind the game in section 2, and lists the difference between AoC II and original AoC in section 3. Section 4 describes the technical solution used to implement AoC II, and we end the paper by outlining further work in section 5.

2 Idea Behind the Game

This section explains the idea behind AoC II, i.e what we on a conceptual level want to accomplish. Most of this also applies to the original AoC since the general idea is the same.

Game Story

The main purpose of AoC is to teach computer fundamentals. But if AoC is to be motivational as a computer game, it must have a surrounding story line that makes the student want to make progress in the game.

The game story is based on the history of computer science. This basis was chosen because it is quite interesting in itself, and knowledge to history might give the student a broader perspective to the subject.

The game begins at campus. The player discovers a broken and unusual computer. Once he touches it, he is sent back in time to the ancient Egypt. He is told that in order to come home, he must repair the broken computer. This requires knowledge, which he will get by re-experiencing the history of computers and learn from all the men and women that participated in forming the field of computer science. The game is divided into historical epochs, and each epoch gives the player an opportunity to learn about a specific part of the syllabus.

Conceptual Description of the Game

The player is walking around in a virtual world that consists of a number of levels. Each level represent an epoch or part of an epoch. He may pick up items he finds on his way and put them in his backpack. Items may be book chapters, questions or other items useful later in the game.

The backpack contains three books, each growing as the player finds another chapter for one of his books. There is a text book where computer fundamental theory is explained, a history book with information about historical persons and

events, and a task book where all answered questions are written for the player's reference.

The player is able to look in any of his books at any time, but he can only see the chapters that he has found.

Questions must be answered when the player picks them up, and some pathways may be closed until the player has answered correctly on specific questions. Questions may thus be used to block the player from advancing too far without having correctly answered a number of questions. By blocking the entrance to the next level with questions, it is certain that the player must have learned something in the current level before advancing to the next one.

AoC is a multi-player game, and all other players on the current level are visible. Players on the same map may talk to each other and thus help each other to solve problems.

Some people are motivated by competition. The goal of the game is to progress to the end, but in addition the player collects points for each question correctly answered. A high-score list shows the players with the highest score.

Helping Students

The most important part of AoC is to teach students about computer fundamentals. The game aspects should motivate many students, but there are people that do not enjoy gaming and would rather spend all their time on studying. We should not force gaming on students that does not want to play. To accomplish this we will provide at least two paths through the game. One will be for the gamer to explore, and the other will be straight forward and just contain questions and relevant theory.

To help students with difficult parts of the game, a number of teaching assistants are available at specific times. They offer help when students do not understand some parts of the theory needed to answer a specific question.

3 Why a New Version

Our new version differs quite a lot from the previous version. During the course we discovered several technical problems. It was difficult to modify and add new content due to the system used to link different parts of the game. Access control to parts of the game, i.e. stopping students from advancing in the game without doing the exercises, was also difficult and error prone with the old system. In addition we felt the old version was too restrictive and not game-like enough, so we decided to rewrite most of the system. All the contents (questions and text) were however ported to the new system.

We have done the following changes to the original AoC:

- **More game like:** The previous version was mostly a collection of web pages connected by a story line. This version contains a 2D map for the player to explore and is more interactive.
- **Technical improvements:** An almost complete rewrite where the problems discovered in the previous version has been avoided.
- **Contents:** More and better information and questions.

- **Question system:** An improved question system with more freedom for the question designer. It includes support for scripted questions for automatically generating variations on questions.
- **Platform independence:** An effort has been done to make sure the client works on most platforms capable of browsing the web. The lack of this was not very popular among students last year.
- **Better navigation through course contents:** Another feature requested by students. Previously, course contents where spread all over the levels, and students wanting repetition had to navigate back through the game. Now, the player keeps all course contents he discovers in his backpack, and may look at it at any time.

4 Technical Solutions

To implement a game with the properties explained in section 2, we chose to use a web based solution. This enables most students to use any computer system they like without having to install extra software.

Client User Interface

All levels in the game are visualised as a 2D map with a graphical image (called an *avatar*) representing the player(s) moving around on this map. The user interface must be flexible and powerful enough to make this possible, and still be able to run in a standard web browser window. We chose to use a Java applet for this.

This applet (shown in figure 1) contains some buttons on the top left. These give the user quick access to the books in his backpack. Status information, i.e the current level and current score is printed on the top right. Under this, occupying most of the window is the map. On the bottom there is a chat window to enable communication between players and teaching assistants. Users in the chat is displayed on the right, with teaching assistants marked with an @-sign.

Questions and book chapters are normal web pages, generated by a C# MS.net framework. They pop up when the player picks them up at the map, or when the player pushes one of the book buttons on top of the applet.

Architecture

The AoC architecture is shown in figure 2. There are four main components: The client with Java applet, the Java server, the web server with MS.net, and a MySQL database.

- **MySQL Database:** Most of the persistent data is stored in the MySQL database. This includes sections for the books, questions and user data with position and status.
- **Web Server:** The web server (Internet Information Services) hosts all HTML pages (generated from an MS.net specification) used by AoC. This includes the login page, book chapters and questions. After login, the web browser downloads the Java applet from the web server. Most of the user interaction happens from this applet.

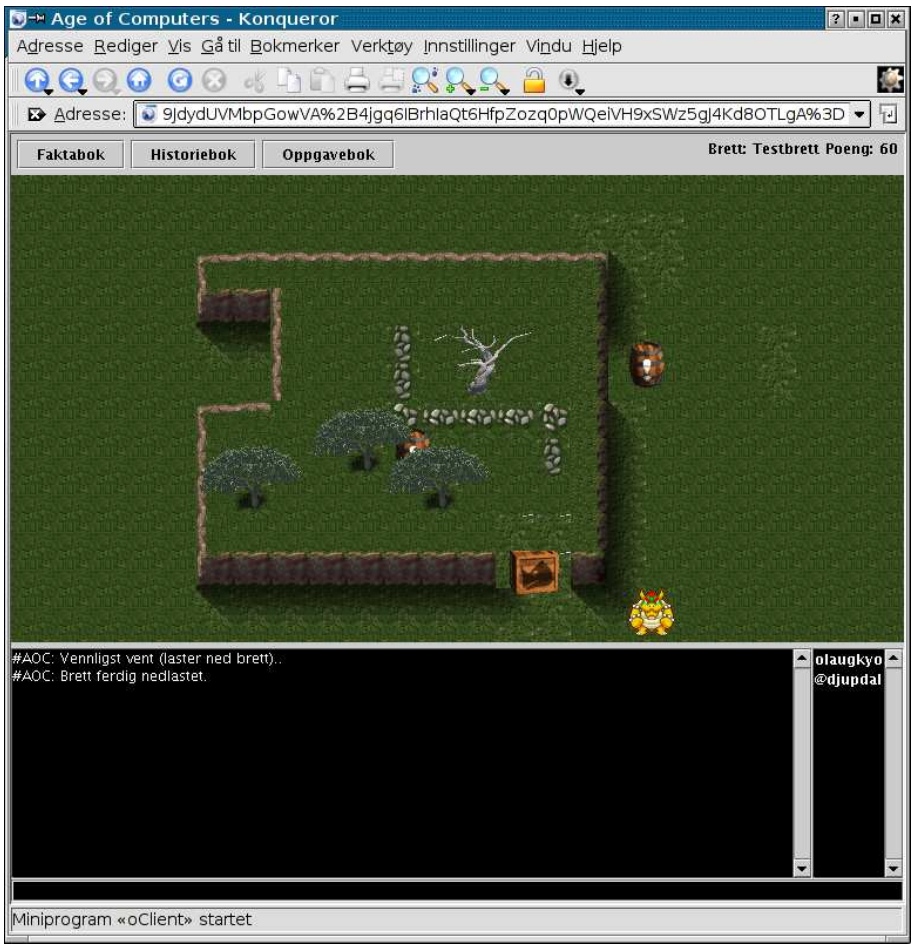


Figure 1: Main user interface for client.

- **Java Server:** The Java server keeps track of where all players are, relays chat messages between users and controls access to various parts of the system. It also feeds the clients with map data.
- **Client:** The client is a web browser, mainly running a Java applet that displays the map for the user. This applet communicates with the Java server.

Course Contents

All course contents (book chapters and questions) are stored in the database as LaTeX.

We considered XML, which was used in our previous version of AoC. XML is a general markup language, and must be defined for the purpose intended. Our previous version of AoC used a very simple XML format that could be translated directly to HTML. This was far too limiting when used to specify structured text documents, as all the problems with HTML was reflected onto our XML format.

A much better alternative would be DocBook [2] which aims at being used for writing technical documents. It is an XML DTD (Document Type Definition) with a large set of tags useful for structuring typical technical documents.

We had however several reasons for choosing LaTeX as our main format. Its extensible nature was important, as it allows us to extend the language with non standard tags (like specifying multiple choice questions) without braking

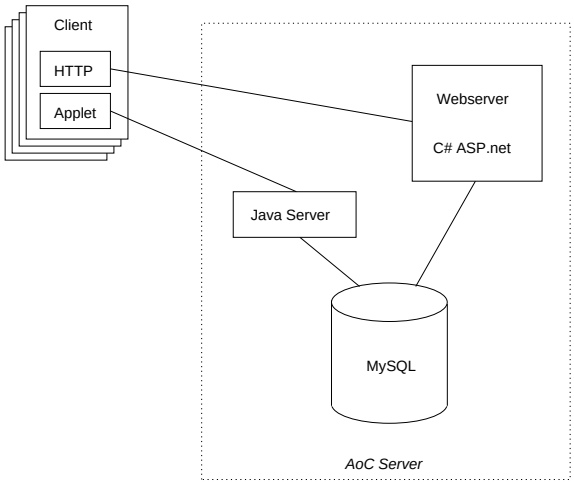


Figure 2: Architecture overview.

compatibility with existing tools. LaTeX (and its tool chain) is also very mature and gives very high quality output. These things, and due to our previous knowledge of LaTeX and its tools, led to our decision of using it in AoC.

LaTeX seems to be working well for our use, and is much better than our original XML format. It is not known if it is more suitable than DocBook because of our limited experience with DocBook.

In AoC we generate both HTML and PDF from the LaTeX source at run time, no HTML or PDF is stored in the database. The LaTeX to HTML converter used by AoC, called TtH [6], is fast and generates an HTML document in fractions of a second.

Books

One important feature of AoC II is the dynamically generated books. The player builds his own books from chapters found on his way.

The student has access to a web page with links to all the chapters in a given book. When he "picks up" a new chapter for a book, the link to this chapter gets activated so he may read it.

Book chapters are stored in the MySQL database. Each chapter is displayed individually when viewed on the web. A mechanism for joining available chapters into a full LaTeX document is used when generating a PDF file.

Questions

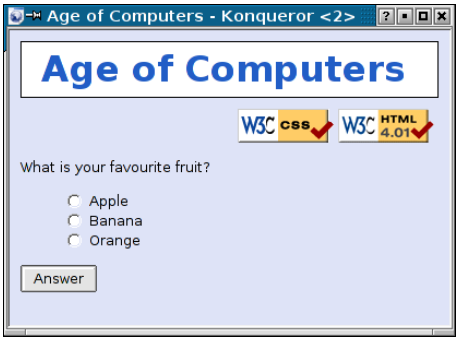
Like book chapters, questions are stored in the database as LaTeX code. The question is to be displayed as an HTML page in the browser, and this page should provide some kind of input mechanism so the student may give his answer.

Generating HTML from the LaTeX code is done as for the book chapters. A complicating factor is that the input mechanism to use should be specified in the LaTeX code. Providing user input like this is not normally done with LaTeX, but was still quite easy to accomplish. We implemented some LaTeX macros that generate HTML input forms when used for HTML pages. These macros are easy to use in the question code, and displays nicely. There is no limitation to the number of input mechanisms used in a question, so questions may be designed quite freely.

```
What is your favourite fru

\begin{radio}
  \radioitem Apple
  \radioitem Banana
  \radioitem Orange
\end{radio}
```

(a) LaTeX code specifying multiple choice question.



(b) Resulting HTML page for the same question.

Figure 3: Example of a simple multiple choice question.

Most of the questions are multiple choice. Other questions use a text input field. When the player answers a question, the answer is automatically checked against the database and the player gets immediate feedback. For a description of the different question classes, see one of our previous papers [13, 14].

An example of the LaTeX code for a multiple choice (or radio button) question is shown in figure 3(a). The resulting HTML output is shown in figure 3(b).

Automatically Generated Questions

We needed more freedom for some of our questions. This led to the implementation of a script system. A scripted question has two associated scripts, stored in the database with the question. The first script runs when a new question of the given type is needed (i.e. a player picks up a question). This generates parameters for the question, for example a number the player is supposed to convert to another number system. The second script is run after the player has answered, and verifies if the answer is correct.

Many questions can be scripted and therefore give the players similar but not equal questions. This reduces the possibility of cheating.

We use C# for scripting. C# code is stored in the database, and is compiled to DLL files before executed.

Administrative Tools

All content is entered into the database either with general database management software, or with specially designed editors. We plan on providing web based tools for entering course contents, but currently only the question editor is available.

Maps are designed with a special map design tool. Items (chapters, questions etc.) are placed on this map using the game applet itself. The map designer may place items on his current position on the map with special administrator mode commands in the chat window.

Platform Independence

A goal for this version of AoC has been to make the client run on most of the available platforms. To accomplish this, all HTML pages are tested against the W3C HTML [8] and CSS validator [7]. The motivation is that a correct and validated HTML page should render correctly in all browsers that complies to the standard. In addition, Java 1.4 [5] is required to run the applet. Java 1.4 is ported to most operating systems of today. The result is a requirement for the client for a standard compliant browser with a Java 1.4 applet plug-in.

The system has been tested against the following browsers: Internet Explorer 6 (Windows), Mozilla 1.6 (Windows and Linux), Opera 7 (Windows and Linux) and Konqueror 3.2 (Linux).

5 Discussion and Further Work

This version of AoC has not yet been tested on students. The first run will be autumn 2004. Even though most of the implementation and content is finished, there is still work to do with designing levels and making it an interesting game. Improving content is also prioritised.

Feedback from students after our first version (see [13] and [14]) indicates that many find the AoC approach more motivational and with better learning effect than traditional exercises. When we asked about preferred improvements many answered better theory explanations and questions. Even though few find it important to make a more fancy presentation of the game, we have done quite an improvement in this area. This was partly a side effect of improving other less visible but important technical problems of the first version.

Based on our experience with the previous version, we are optimistic with respect to the success of the new AoC version. We especially look forward to see if the increase in game appearance of AoC is received well among the students.

There are many things that may be improved in AoC II. Our main goal is to make AoC motivational as a computer game, and we may still lack some of the properties that make a computer game fun. It is difficult to find the balance between making a pedagogically sound system which ensures a steady progress and learning for the students, and at the same time make it fun to play AoC. IT technology offers a rich set of possibilities for doing education more motivating and interactive. We have only seen the beginning of the future developments in e-learning and edutainment. Age of Computers is meant to be a small but early contribution in this field.

6 Acknowledgements

We would like to thank Simon Thoresen for doing most of the work on implementing AoC, Olaug Kyoko Namba Østhus for some of the new content, Steinar Line as the former project leader, and all others that have done work on AoC.

References

[1] Center for computer games research.
`http://game.itu.dk/`.

[2] DocBook webpage.
`http://www.oasis-open.org/docbook/`.

[3] The education arcade.
http://www.educationarcade.org/.

[4] IT-universitetet i Göteborg.
http://www.ituniv.se.

[5] Java.
http://java.sun.com.

[6] TtH: the TEX to HTML translator.
http://hutchinson.belmont.ma.us/tth/.

[7] W3C CSS validator service.
http://jigsaw.w3.org/css-validator/.

[8] W3C HTML validator service.
http://validator.w3.org/.

[9] BECTA (British Educational Communications and Technology Agency). Computer games in education project.
http://www.becta.org.uk/.

[10] J. Kirriemuir. Video gaming, education and digital learning technologies: Relevance and opportunities. *D-Lib Magazine*, 8(2), February 2002.

[11] MIT. Games-toTeach project.
http://www.educationarcade.org/gtt/.

[12] TEEM (Teachers Evaluating Educational Multimedia). Report on the educational use of games.
http://www.teem.org.uk/publications/teem_gamesined_full.pdf.

[13] Lasse Natvig and Steinar Line. Age of Computers – game-based teaching of computer fundamentals. In *ITiCSE*, 2004. To appear.

[14] Lasse Natvig, Steinar Line, and Asbjørn Djupdal. "Age of Computers:" an innovative combination of history and computer game elements for teaching computer fundamentals. In *ASEE/IEEE Frontiers in Education Conference*, 2004. To appear.

[15] K Squire. Gaming in higher education part I.
http://www.joystick101.org/story/2001/2/18/111322/124.

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.Â @inproceedings{Natvig2004AgeOC, title={Age of computers: game-based teaching of computer fundamentals}, author={Lasse Natvig and Steinar Line}, booktitle={ITiCSE '04}, year={2004} }. Lasse Natvig, Steinar Line. Published in ITiCSE '04 2004. Computer Science. Computer Science degrees teach you, at some level, to be able to code and program. These are highly portable skills that will serve a computer scientist well for designing games. "Game designers are often required to use scripting languages which are similar to programming languages to create gameâ€™s levels, missions, and other player interactions." - gamingindustrycareerguide.com. Source: EvgeniyShkolenko/iStock. Of course, a Computer Scientist would need some additional training on the job to be able to create games. But they have a good head start. What do video game designers use? However, the first game system designed for commercial home use did not emerge until nearly three decades later, when Ralph Baer and his team released his prototype, the â€œBrown Box,â€ in 1967. The â€œBrown Boxâ€ was a vacuum tube-circuit that could be connected to a television set and allowed two users to control cubes that chased each other on the screen.Â The Modern Age Of Gaming.Â These advances could signify an amazing new chapter for gaming â€” especially if combined with VR, as they could allow games to interact with characters within games, who would be able to respond to questions and commands, with intelligent and seemingly natural responses.