Introducing CandleWeb and Å (awe), Bringing Animation Power to the World Wide Web

Kjell Øystein Arisland, Svein Johansen, Gunnar Rønning

Abstract

The World Wide Web has limited interactive capabilities, and does not support animated graphics well. To allow real-time interaction and animated graphics that are both pedagogically and commercially motivating, we must extend the Web. A new tool called CandleWeb is presented. CandleWeb works together with standard HTML browsers, and uses the hypertext transport protocol (HTTP). The tool has been implemented for X11, and interprets a language called Å (awe) which combines a simple C-like syntax with standardized graphics objects to provide a programming environment in which presentations including animation can be produced efficiently. An authoring tool called Å (awe) Composer allows programmers to save considerable time in implementing animated presentations, compared to text-based programming, using graphics libraries. The CandleWeb client for X11 V1.0beta and the Å (awe) language are openly available on the Internet at the site http://www.oslonett.no/~candle/. ***Keywords:*** *Advertising, animation, awe, authoring, browser, C, CandleWeb, client, commercial, composer, education, graphics, HTML, HTTP, interactive, interpretation, language, programming, real-time, tool, World Wide Web, Å*

Introduction

The World Wide Web is arguably the most useful thing that has happened to the Internet since TCP/IP. However, whenever something new and powerful comes along, there is a desire to make it even better and use it for more than it was intended for. This paper, which is our attempt to expand the power of the Web, discusses interactivity and animated graphics in a new tool called CandleWeb.

Educational Use of the Web

For many years now, the computer has been heralded as a tool that would some day pervade schools and homes and would become both helper and teacher. For an overview of literature in the field of computer applications to education, see [8]. Tools and courseware have been developed to take pedagogical advantage of often very limited hardware, and sometimes powerful hardware has been used for teaching using limited pedagogy. Mostly, however, the hardware, its powers and its availability has been the limiting factor. Today, with multimedia workstations and wide-area networking, the hardware is becoming less and less of a problem, and the era of the computer teacher is about to begin. A naturally useful tool for teaching is the Internet, and more specifically the World Wide Web [2]. It is embraced as such at universities and other educational institutions across the globe.

However, during more than three decades of experience experimenting with computers and learning, many techniqes have been developed that speed up the process of using the Web efficiently for teaching. Therefore, those who wish to unleash the power of the Web on their students can do so without relying on trial and error.

As an example, in spite of many attempts, "electronic textbooks" have never been conclusively shown to be generally superior to a normal printed textbook as a tool for teaching. The so-called "Hawthorne effect" [5] may lead experimenters to believe that electronizing textbooks has intrinsic value, when in reality the process only yields a novelty effect.

Even hypermedia organization of documents may not be more than a passing fancy that does not add any real value in a teaching situation. In any case, hypermedia may or may not add value to instructional material, depending on how it is used [5].

However, there are at least two features of the Web and of computers in general that can improve on the learning environment when used well. These include:

- User/computer interactivity
- Moving graphics (animation)

These features will be discussed in greater detail below.

Commercial Advertising on the Web

The World Wide Web is also useful for commercial purposes, and the advertising industry is gradually becoming aware of its powers of influence and explosive growth. It is interesting to note how teaching and advertising have very much in common. Both fields require capturing the attention of the learner or potential customer (the user), increasing the user's interest in the subject at hand, and finally motivating the user to act either to buy something or to invest time in continued learning. For a thorough introduction to most of the aspects of advertising relevant to Web designers, see [3]. Because of the similarities, both educational and advertising use of the Web require basically the same types of mechanisms to be present in the Web. The two features that we concentrate on in this paper, interactivity and animation, are certainly just as important to commercial advertising as they are to educational use of the Web.

User/Computer Interactivity in the Web

Interactivity is a basic teaching tool that the computer naturally possesses. In comparison, the use of printed material offers very limited interactivity. Interactivity may take many forms depending on the time factor. For day-to-day interactivity, using the Web as a message center for general distributed communication is clearly useful. In a teaching environment, more short-term interactivity (real-time interactivity) may be even more useful. It is a basic tenet of pedagogy that the effect of learner action as opposed to just hearing or seeing is quite strong relative to human ability to retain and recall information. Therefore, an increased degree of interactivity in the Web would be beneficial from an educational as well as a commercial advertising point of view.

Such real-time interactivity is the basis of all action-type computer games, and thus clearly has appeal to the masses. Given the fact that children today spend more and more time playing computer games and less time watching television, one could conjecture that the real-time interactivity of the computer has even more appeal to the masses than the traditional story telling that is the basis of more conventional media like television and movies. This conjecture certainly remains to be proven, and only time can tell. However, at the moment, such interactivity is undoubtedly quite attractive to a large percentage of all potential Web users.

Unfortunately, the Web is somewhat limited in its support of short-term or real-time interactivity. This limitation is due to the fact that wide-area networking generally means overly long delays for real-time interactivity.

Still, many have had the desire to use the Web for applications that basically demand real-time interactivity, and have tried to implement different types of games. The Web's lack of support for such interactivity has so far limited most such attempts to just that, attempts. In order not to offend anyone, we offer no references, but there are many examples to be found on the Web.

One method of real-time interactivity found in the Web is that of forms. Forms offer real-time interactivity because the process of filling in the form is supported by the browser itself and the HTML document only provides code for specifying what the form should look like to the user, and what types of fields should be included in the form. The latter can be viewed as a specification of how the browser should allow the user to interact with the form. In this view the browser performs active interpretation of the HTML specification, as opposed to just presenting static information. Forms thus differ functionally from most of the rest of HTML.

Another HTML feature of real-time interactivity is that of maps. In maps, areas within a bit-mapped image can be specified as anchors, and a script in the server can interpret which URL to activate whenever the user presses a button at certain pointer (mouse) coordinates. The only real-time interactivity involved here, however, occurs in determining the mouse coordinates before they are transmitted over the WAN via HTTP. Since this local interactivity demands HTTP communication for each click of the mouse, the only gain in interactivity is in allowing mouse input relative to a graphic area, and not in the speed of the interactivity.

Relative to interactivity, SUN Microsystems' Hot Java [9] needs to be mentioned. The Java language, when used to implement Web browsers, has an advantage over most traditional languages in that it can be compiled to a code that is hardware independent and can be interpreted on different hardware platforms. This makes it possible to extend HTML in various directions to provide, among other things, stronger interactivity.

In conclusion, it is obvious that the need for and desire for real-time interactivity in the Web is considerable, but so far, good solutions have not been plentiful.

Moving Graphics (Animation) in the Web

Animation is one of the most powerful motivation tools available today both for pedagogical and commercial applications. Several generations of people in the industrial world have been raised on cartoons from Disney and Hollywood, and the so-called MTV generation literally demands fast-paced animated material; they simply may not notice commercial presentations that do not communicate in the same exaggerated way as Hollywood cartoons.

Video games and computer games are another reason why many people expect more from the Web in terms of animated graphics than what it is capable of delivering today. There is a world of difference in the liveliness of the graphics in the game Doom, compared to that of Mosaic or Netscape, and one may ask why. One answer is certainly the limited bandwidth on the net, but this is not the whole truth. Lack of standards both in hardware and operating systems is probably just as important.

The two types of graphics supported in the Web are basically bitmap images, either in GIF or JPEG format, and video clips in MPEG format. The process of using methods for storing images is quite wasteful in terms of storage space and bandwidth requirements. Both JPEG and MPEG are certainly state of the art in compression schemes for digitized natural images, however, reproducing natural images is not necessarily the best way to deliver educational or commercial messages.

Much thriftier methods, such as vector graphics and palette animation, are well known in the world of computing and are the basis for much of the mentioned games industry. When small bitmap images are moved around against vector graphics or textured backgrounds, quite powerful animation can be produced over relatively low bandwidth channels.

In conclusion, good reasons exist for enhancing the Web with more visually stimulating animations, and the means for doing so exist as well. There are two obvious questions that arise: Why does the Web not support animation already, and how can it be added?

There may not be a conclusive answer to the first question. However, HTML has its basis in document publishing, and this is most probably the reason. Information structuring as defined by markup is a far cry from animation, and adding so-called media types to a hypermedia document as one adds icing on a cake does not change the cake into steak. To make things difficult, animation is basically pixel and coordinate oriented, while HTML and SGML are strongly text string oriented.

The second question, concerning how animation can be added, is addressed in nearly half of the remaining sections, and one approach will be explained in some detail.

Authoring Systems, Multimedia, and the Web

In the field of Computer Assisted Learning, developing educational software has for several years been an industry in its own right. While educational software is produced for the masses by large companies, a large segment of this industry is comprised of corporations with in-house development of educational software for their own purposes. A consulting segment also exists in this field. Common to both segments is that they do, to some extent, use authoring tools for producing their software, and such authoring tools exist in many variations. Examples of tools holding large market shares are TenCore, AuthorWare, and MacroMedia Director.

Common to most authoring systems are implementors that produce educational software more efficiently than when general programming languages are used. Some tools also specialize in supporting simulation in various forms, including application-specific.

For the World Wide Web, authoring has been limited mostly to general text-based formatting such as that in Microsoft's Word for Windows, and conversions of such texts to HTML. This type of authoring is quite primitive compared to the capabilities of multimedia authoring tools like Director; however, the Web does not support the kind of primitives needed to apply such tools today.

CandleWeb and Å (awe)

CandleWeb and Å were designed to increase the functionality of the Web with respect to real-time interactivity and animated graphics. This increase was achieved by:

1. Designing a language (called Å, pronounced "awe," an acronym for "another web extension") with dynamic vector graphics objects and code suited for interpretation
2. Designing and implementing a client application, called CandleWeb, for downloading a program using HTTP and then interpreting this program locally on the user's host computer
3. Designing and implementing a composer tool, called Å Composer, for efficiently implementing animation-based presentations using the new language

The Basic Architecture

Figure 1 illustrates the basic architecture of CandleWeb and Å (awe).

The CandleWeb client application is capable of interpreting *.awe* files; therefore ASCII-files with the extension *.awe* are downloaded using HTTP and your favorite Web browser. When they arrive at your host computer, an application client called CandleWeb is fired up and fed with the . *awe* file. CandleWeb interprets the *.awe* file interactively on your host computer, showing dynamic graphics locally and letting the user interact with the interpreted program through input-objects.

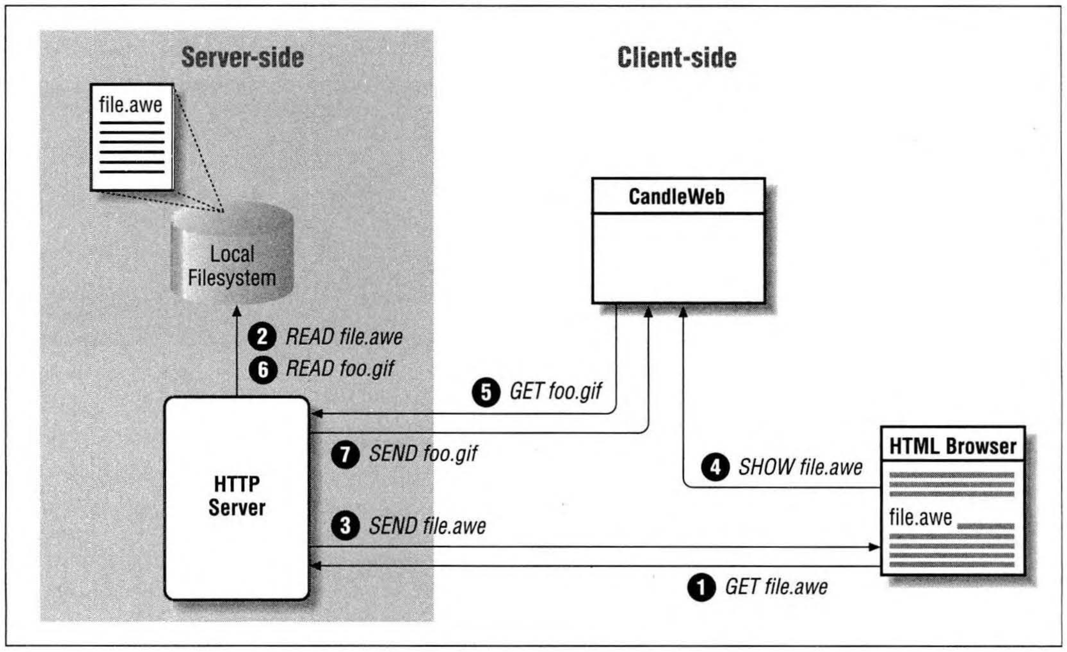


Figure 1: Architecture of CandleWeb and Å (awe)

The Å Language

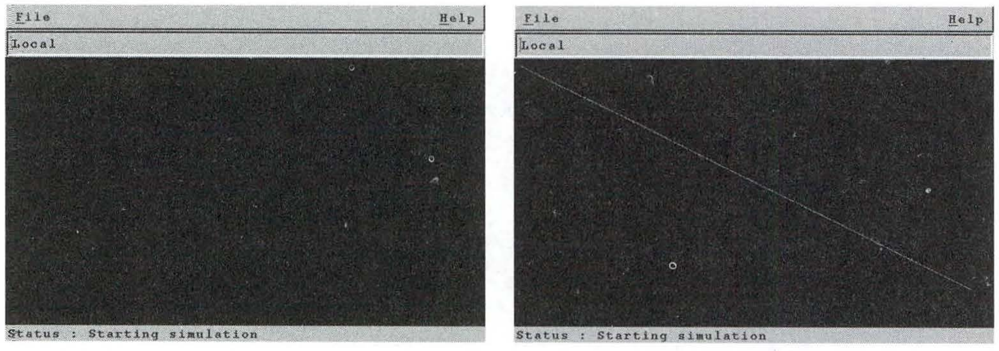
The following is just a short introduction to the Å language. A more detailed description is found in [7], and the language specification is found at [4].

"Å" is the last letter of the Norwegian alphabet. It is an A with a small ring placed right above it, and it is pronounced like the English word "awe. " This single letter was chosen for the language name in keeping with the "C" tradition.

Å was specified as a language in the Algol tradition [11], with a subset of the C programming language syntax [6] as the base, to avoid creating a whole new language. Many of the basic C language constructs are included in Å, with the notable exception of pointers and structures. These were left out mainly to simplify the demands put on the CandleWeb client which has the job of interpreting programs written in Å in a secure manner.

In addition to the traditional features of programming included in Å by means of the C language, Å also includes so-called dynamic graphics objects. These are basically vector graphics objects like lines, boxes, and polygons, but may also be specialized objects like GIF or JPEG bitmaps, text objects, dynamic windows or input objects.

As an example, consider a line object. A line has start and end point coordinates, thickness, color and rendering. For all of these parameters, a fixed value may be specified, or the parameter may be given as the name of a variable, or as an expression involving a variable. In the latter cases, the object is said to be tied or bound to the variable, and the variable is said to influence the object. For all the graphics objects, any parameter may be tied to a variable. When the C code is interpreted, the interpreter keeps track of all objects with parameters bound to variables. Whenever a statement in the code changes a variable, all objects influenced by that variable are also changed. The interpreter executes the changes, and the programmer does not have to



line object, xstart=0 line object, xstart=100

Figure 2: Two line object examples

worry about them. Whenever the programmer wants the current changes to become visible on the screen, the statement "output;" is inserted in the code.

Line object examples are shown in Figure 2. In Figure 2a, the line is shown with variable xstart=0, and in Figure 2b, the same line is shown after variable xstart has changed to xstart=100.

```
Line object, xstart=0 Line object,
    xstart=100
Figure 2a         Figure 2b
```

More specialized objects like bitmaps in GIF or JPEG format may also have the same dynamic parameters. The image itself is fixed, but the coordinates may be bound to variables, thus making it possible to move the bitmaps around in the display window.

Two other types of objects deserve special mention. The first is the *window object*. These are not conventional windows with frames and backgrounds, in fact they are invisible to the user. The windows behave like normal windows in that they can contain other objects such as lines, boxes, bitmaps, and texts, and can perform a normal clipping function. However, as mentioned, the windows themselves are invisible, and serve mainly to group other objects. When the window coordinates are changed, the contained objects are moved around on the screen; when a window's on/off variable is turned off, the entire contents of the window disappears. These features make windows very useful for producing animation in various forms.

As an example, consider a set of images, each of which is contained in its own window. Every window has its coordinate parameters bound to the same variables; thus moving them all requires a change to the two variables for x- and y-coordinates. By turning the windows on and off, one may easily switch between the various images to produce cell-based animation.

The second object that must be mentioned is the *input object*. Practically all user input is handled uniformly through the input objects. The input objects may also be bound to variables, but in this case, the tables are turned. Instead of the variables changing the graphics, the input objects change the variables. The objects themselves are invisible, and are made of virtual rectangles of the screen that react to mouse clicks, keyboard input, or a combination of the two. Whenever some input object is activated, the object's action may directly influence one or more variables. As with graphics output, it is practical to control the exact point in time at which input is allowed to change variables via "input;" statement.

When input objects are used to change variables that influence graphics objects, the user may do so without any code other than the "input;" and "output;" statements.

The CandleWeb Client

As illustrated in Figure 1, the CandleWeb client receives a file with extension *.awe* from the Web browser and interprets this file in its own window on the screen. The *.awe* file may contain references to GIFs, JPEGs or other *.awe* files, and the CandleWeb client will fetch these using HTTP whenever necessary.

Since the Web browser doesn't tell the CandleWeb client the source of the *.awe* file, that file must contain a reference to its place of origin so the CandleWeb client can find any other files that have local references. This referencing is done using a function in the Å subroutine library called setAnchor().

The fact that the Web browser did not include a way for the CandleWeb application to find the source of the *.awe* file is regarded as a deficiency in the current Web application protocols; hopefully, this will be remedied in the future.

As the CandleWeb client is a standalone application started from the Web browser, it is not much influenced by which Web browser is used. Similarly, it isn't influenced by developments in HTML, since it does not use HTML.

When unknown application programs are downloaded, using HTTP, from anywhere in the world, and are to be interpreted on one's own host computer, strict security measures should be taken by the interpreter. One must trust that the interpreter will not allow any malicious program to make changes to the file system. The CandleWeb application does currently not allow any file writes at all. Additionally, there are no functions for exporting information from the host computer; however, functions for importing information using HTTP exist.

The Å (awe) Composer Tool

The direct link between variables and graphics objects in the Å language is a very simple, yet very powerful construct which greatly reduces the amount of program code necessary to produce graphics applications. Typically, applications stay below a few hundred lines of code, and on the average only about a third of the code is program code, while the remaining two thirds consist of declarations of graphic objects.

When the predecessor to CandleWeb and Å, the Candle 1.0 system, was developed for MS-DOS in 1988-90, many recognized quite early that implementation efficiency could be increased considerably by implementing drawing tools for direct manipulation of graphics and automatic generation of the graphic objects code. Therefore, an authoring tool called Chandler was implemented. The authoring tool went through two generations of relatively different implementations and both implementations were used in several projects of developing educational software for apprentices in heavy industry in Norway [1].

One of the many observations noted was that implementing graphics-oriented educational software was three to five times more efficient with the authoring tool as opposed to specifying the graphics by text input. This is hardly unexpected, and quite well established in industries that rely heavily on educational software. Authoring tools like TenCore, AuthorWare, Director, and many others allow programmers, and to some extent nonprogrammers, to produce educational software far more efficiently than with general programming languages.

As a result of the success of the Chandler authoring tool, an Å Composer for X11 is currently being implemented as well. This tool allows direct graphic drawing and manipulation of the graphic objects, the window objects and the input objects in the Å language. The tool also provides specific support for accessing and manipulating the special relationship between variables in the program code and the graphic objects. The Å Composer tool is further described in the paper [10].

A Simple Example

For a simple example of what Å code looks like, the short file *logoflash.awe* is listed below. Note how relatively simple this program is. If possible, compare its simplicity to the magnetic effect it can have on a user at an X11 workstation using CandleWeb. The file can be found at *http://www.oslonett.no/~candle/demos/logoflash.awe* a-long with several other demonstration programs.

Example 1: logoflash.awe

```
int main ()
{ // Simple program demonstrating the power of CandleWeb and   (awe)

// Variable declaration
int x, y;

// Background color
box points = ((0, 0), (800, 600)), fill = 1, color = 0x4444FF;

// Window containing image
window points = ((x,y), (x+130,y+150)), sb=0;
image points=((0, 0), (0, 0)), sb = 0, image = "candle.gif";
endwindow;

// Header text
textobj points=((20, 30)),
outtext="Computers and Learning AS' logo shown at randomly chosen points.",
color=0xFFFFFF, level = 1;

// Setting window size to 800 pixels horizontal and 600 vertical
resizeWindow (800, 600);

// Setting anchor location so the CandleWeb client will find image file
setAnchor("http://www.ifi.uio.no/~candlweb/demos/logoflash.awe");

while (1) {

// Draw random point
x = random( 4 , 660 );
y = random( 50 , 460 );

// Draw screen
output;

// Loop delay
wait( 50 );
}
}
```

Why CandleWeb and Å?

A basic assumption of this paper is that support for real-time interactivity and animation is needed in the Web, and that it will be used a lot when made available. The big question concerns how these capabilities will be provided. This paper presents a full-fledged proposal, including implementations.

The next question, then, is the following: Is this proposal good enough to become a standard in any way? Or phrased differently: Will it be used?

We cannot answer this question now, but we can present some of the reasons for the choices made in designing CandleWeb and Å.

Standards

Even though CandleWeb and Å seem to represent some fairly new and unusual thinking relative to the Web, designers stuck to defacto standards, changing as little as possible with something known to work well.

The Å language itself is such an example. It is based on the C programming language, borrowing most of its constructs from C. For a more detailed introduction to Å see [7] and [4]. The reason for choosing C is that it is one of the best known programming languages today, and a very efficient and uncluttered one. Because the object of designing Å was not to produce a new programming language, but rather to specify a language for a specific purpose, designers used a subset of a well known language and enhanced this subset with the necessary extensions. An added bonus is that a great number of programmers are already familiar with most of the new language, and need only learn which parts of C are not supported and what the extensions are.

The CandleWeb communications architecture is quite simple, and basically uses HTTP. In addition, Å code may include links to other Å files through function calls to links.

Since HTML is basically a markup language, and is not a pixel-oriented graphics language, designers decided that if HTML was to remain reasonably small and uncluttered, they should design a separate and fully graphics-oriented environment rather than extend HTML in a direction that would contradict some of its original intensions. The authors' opinion is that part of HTML's power will be lost if one tries to make it all things to all people. Therefore the CandleWeb/Å graphics environment is a pixel-oriented drawing canvas in a CandleWeb window, separate from the user's HTML browser. An added bonus from this choice is that CandleWeb does not have to compete with HTML browsers, and vice versa.

Interpretation and Security

Since Å is an interpreted language, it is a hardware- and OS-independent programming environment. This feature is very advantageous for producers of educational software. The demand for such software is much greater than what can be produced in a few years: the fact that much of what is produced quickly becomes technically obsolete because of changing hardware poses a problem. An interpreted language can survive several generations of hardware and many versions of the language itself. Currently a few system dependencies exist, noticeably regarding fonts, but the goal is maximum system independence.

Interpretation does present security problems. When code is downloaded and run on a local host, the hosts security relies to a great extent on the interpreter. The Å language has been kept very simple in order to make it easier to keep the interpreter safe. The omission of pointers and structures is one example; not allowing local file access is another.

Speed

Vector graphics can be extremely compact in terms of code needed to produce quite complex pictures. The graphics objects in the Å language are based on a combination of vector graphics and bitmaps. This combination increases the speed of following *.awe* links in two ways:

1. The code to be downloaded is very compact
2. The downloaded code utilizes hardware that is often optimized for vector graphics operations, such as line drawing, polygon fills, and the like.

Authoring

CandleWeb and Å are designed to take advantage of the Å Composer tool, which in turn is the result of several years of work in the field of graphics-oriented software authoring. It is time for Web designers to start using more of the results from related disciplines such as Human Computer Interaction and Computer Assisted Learning. Some of these results are quite general and need not be reinvented or rediscovered.

Current State and Future Plans

The CandleWeb client for X11, V1.0Beta has been released. A complete V1.0 will be released when the feedback on the beta release justifies it.

Å Composer for X11 is currently being implemented.

A CandleWeb client for Windows is planned for implementation and the work will be starting early in August 1995.

Å Composer for Windows is planned for implementation starting in 1995.

Both the CandleWeb client for X11 and the Å language specification have been released for public, academic, private, and commercial use (the latter only in unmodified form) at no charge. For details, see the license at *http://www.oslonett.no/~candle/license.html.*

Source code for CandleWeb client for X11 is available at no charge for academic use (research and education).

Conclusion

We have presented a new tool called CandleWeb that extends the capabilities of the World Wide Web on the Internet to include real-time interactivity and full-screen graphics animation. The tool works with standard HTML browsers, and uses HTTP. The tool has been implemented for X11, and is the successor of a similar tool, Candle, that has been used very successfully for several years implementing animations for MS-DOS-based hardware.

The tool also includes an interpreted language called Å, which combines a simple C-like syntax with standardized graphics objects to provide a programming environment in which presentations including animation can be produced efficiently.

Finally, the tool includes an authoring tool called Å Composer that allows programmers to save considerable time in implementing animated presentations, compared to text-based programming, using graphics libraries.

The CandleWeb client for X11 V1.0beta and the Å language are openly available on the Internet at *http://www.oslonett.no/~candle/*.

Acknowledgments

Kjell Øystein Arisland, Yngvar Berg and Arne Kinnebergbråten specified the original Candle architecture and language and implemented MS-DOS-based clients. Knut Tvedten implemented most of the Candle composer-tool called Chandler for MS-DOS.

The Federation of Norwegian Process and Manufacturing Industries funded several of the early development projects involving pedagogical software from which the Candle architecture and language got its start. The following people deserve special mention for their support: Bjørn Lassen, Svein Hyggen, Morten Allum, and Per Nørbech.

Svein Johansen and Gunnar Rønning have specified the Å language as a successor to the Candle

1.0 Language, and have implemented the current CandleWeb client V1.0beta for X11.

Tore Engvig, Bjørn Thirud, and Kent Vilhelmsen are currently implementing the Å Composer for X11. ■

References

1. Arisland, K.Ø., "The Good, the Bad, and the Unusual in Computer Assisted Learning," Proceedings MULTICOMM'94, Vancouver, Nov 2-3, 1994.
2. Berners-Lee, T., *http://www.w3.org/hypertext/WWW/Protocols/HTTP/HTTP2.html*
3. Faison, E.W.J., Advertising: A Behavioral Approach for Managers. John Wiley & Sons, 1980.
4. Johansen, S., and G. Rønning, "Å (awe) specification.", Web document at *http://www.ifi.uio.no/~candleweb/spec/spec.html*, 1995.
5. Hutchings, G.A., W. Hall, J. Briggs, N.V. Hammond, M.R. Kibby, C. McKnight, D. Riley, Authoring and Evaluation of Hypermedia for Education. Computers Educ. Vol 18, No. 1-3, 1992, 171-177.
6. Kernighan, B.W., and D.M. Ritchie, "The C Programming Language," Prentice-Hall, Inc, 1978.
7. Rønning, G., S. Johansen, and K.Ø. Arisland, "Å (awe), an interpreted animation language for the Web," Paper in preparation, 1995.
8. Rubincam, I., "A Taxonomy of Topics in Computer Applications to Education Based Upon Frequently Cited Books, Articles and Reports." Journal of Research on Computing in Education, 1987 - Winter, 165-187.
9. *http://java.sun.com*
10. Vilhelmsen, K., B. Thirud, T. Engvig, and K.Ø. Arisland, "Å (awe) Composer, graphics authoring for the animated Web," Paper in preparation, 1995.
11. Naur P., "Revised Report on the Algorithmic Language ALGOL 60.," Comm. ACM 6, 1963.

About the Authors

Kjell Øystein Arisland
Department of Informatics, University of Oslo
Norway
kjell@ifi.uio.no

Svein Johansen
Department of Informatics, University of Oslo
Norway
sveinj@ifi.uio.no

Gunnar Rønning
Department of informatics, University of Oslo
Norway
gunnarr@ifi.uio.no