AC 2010-959: A CASE STUDY IN THE USE OF ANIMATED VISUAL MODELS IN COMMUNICATIONS ENGINEERING EDUCATION

Dave Pearce, University of York

Dave Pearce is a Lecturer in the Department of Electronics at the University of York. He graduated from the University of Cambridge in 1985, and worked in industry for 11 years designing optical fiber systems and local area networking equipment before commencing a PhD in wireless access methods in 1996. He is now responsible for the teaching of communications engineering at undergraduate and postgraduate level at York.

Phil Barker, Heriot Watt University

Phil Barker is a Learning Technology Adviser at the Institute for Computer-Based Learning in the School of Mathematical and Computer Sciences at Heriot Watt University in Edinburgh. He studied physics at Bristol, before doing a PhD and three years of research on polymer crystallization, followed by one year of research at the CSIC in Madrid. He has worked in education research since 1996, specialising in the use of computers to aid learning in the sciences and engineering.

A Case Study in the Use of Animated Visual Models in Communications Engineering Education

Abstract

The presentation of communication protocols in a traditional lecture format is problematic. Many of the systems and algorithms in common use are dynamic in operation and difficult to understand from static diagrams or verbal descriptions; and engineering students often have a visual learning style and would be expected to gain a more intuitive understanding of systems from seeing them working. This paper reports a study into the use of a series of animated simulation tools in lectures for a sophomore course in communications protocols. The approach has won widespread praise from students, as well as winning a national award for engineering education, and the models have been adopted by several universities worldwide.

Introduction

Despite the ready availability of on-line libraries, text-books and tutorials, the scheduled lecture remains the primary means of presenting information to students. This paper reports a study into a taught module in communications engineering, in which the lectures were used to demonstrate and discuss the operation of algorithms and systems using bespoke animated demonstrations and simulation programs, with the majority of the course content described in printed notes distributed in advance of the lectures.

The operation of protocols is not something that can easily be demonstrated using real physical examples of the protocols: for any attempt to use the visual teaching style preferred by many engineering students[1], there is an immediate challenge in presenting visually what are essentially hidden processes[2].

The technique used previously involved drawing diagrams by hand during the lecture, and then modifying these diagrams to illustrate the dynamic operation of the protocol. There are several problems with this technique: students cannot replay the lectures in their own time for revision; students find it difficult to copy down a changing diagram; there is the possibility of mistakes by the tutor confusing the students; and the resulting diagrams can become complex, hindering their use in revision. A sequence of prepared diagrams showing different stages in the operation of the algorithms is the technique used in many textbooks and similar diagrams could have been used in lectures, however it was felt that this would not achieve the same level of engagement by either the students or tutor, and it does not allow students to experiment.

Rather than using bespoke animations, other existing programs could have been used, for example the use of protocol simulators (such as ns2[3]). However, these protocol simulation programs are complex, require a great deal of time to learn, and lack the simple user interfaces most appropriate for use in lectures. It was a key finding from this study that students prefer simple programs, which are quick to understand and intuitive to use; and when more complex programs are presented, it is important that they are introduced in stages, with features not visible until they are required.

In the remainder of the paper, the benefits and disadvantages of the chosen technique are discussed. A short survey of the available methods for writing suitable programs is then given, followed by a description of selected examples of the programs used during the course. Evaluation of the technique was conducted by student questionnaire and interview, and these results are presented and discussed. Finally, some overall conclusions are drawn.

Benefits and Disadvantages of Animated Simulations

The most immediate benefit expected from the use of animated programs during lectures was the increased levels of student attention resulting from the more active teaching style[4], and the greater focus on a visual teaching mode, likely to be more suited to the students[1].

The most significant disadvantage is the increased preparation time required. While many animations are available on-line (for example the library of Mathematica models at http://demonstrations.wolfram.com), only a few programs were found suitable for illustrating the algorithms covered in this course. Additionally, it has previously been reported that uniformity of the programs is an important feature in minimizing the time required for students to become familiar with each new program[5]; and collecting existing programs from different sources would prevent any uniformity in the user interface.

One problem with many of the pre-written animated demonstrations found is that they illustrate systems that work. Engineers have a legendary desire to fix things (even when they are not broken) and showing them something that does not work (or at least is not optimal) and asking them how to fix it has proved a very successful way of maintaining interest and prompting discussions during lectures. The most common suggestions can be pre-programmed into the programs, so that students' ideas can be tested live. This has proved a very popular feature of the programs, with a majority of students reporting downloading and using the programs in their revision.

Another key disadvantage of the technique is that the quantity of material that can be presented during a lecture is reduced due to the time required to explain and watch the animations; however a previous study has suggested that students prefer the more active learning style even when they are required to self-study some of the material[6]. In this study, all the information required for the course was provided in a set of prepared notes given out in advance, with the students advised to read the relevant material before attending the teaching sessions. It was suggested that the provision of a complete set of notes would reduce the attendance levels at lectures; however in the case reported (a sophomore course in communication protocols taught at the University of York) this did not happen, with attendance levels slightly above those of comparable, more traditionally-taught courses.

Simulation Technologies

There are several technologies that can be used to develop animated tools suitable for use in lectures; the choice of technology for this work was made on the basis of the familiarity of the tutor with the programming environment.

Existing mathematical simulation programs such as MATLAB and Mathematica have graphical user interface editors that can animate objects, plot graphs, modify and move text, and allow the user to input parameters. The advantage of these tools is that the programmer has simple access to the extensive library of mathematical and graphical functions provided with the tools; however a disadvantage is that a two-stage process is required if the students wish to run the animations themselves: the download of a player that has to be installed on the student's computer, followed by the download of the demonstration itself. Aside from being a disincentive for the student to experiment with the animations themselves, it can prevent their use in cases where the student does not have the ability to install programs on the computer they are using.

Just about any programming language could be used to develop suitable animations. The programs used in this study were written in Microsoft Visual Basic version 6 (VB6), which is the quickest way to generate user interfaces that the authors have come across. VB6 has a range of graphical plotting routines built-in, and the animations can be compiled to an executable file that can be run directly on any computer running the Windows operating system. Unfortunately, this language is no longer supported.

One notable choice would be the use of a programming language that allows the programs to be run from within a web-browser, notably Java, ActionScript and Silverlight. This technique has the benefit of allowing the animations to be run just about anywhere, and each of these programs is now powerful enough to support complex simulations.

After a survey of currently available technologies and development environments, the authors have chosen to continue development of the simulations using C# and the Windows Presentation Foundation (WPF) graphical subsystem. The advantages of this choice include:

- The use of a general purpose programming language provides greater power to customize animations;
- A wider set of visual objects (sliders, drop-down boxes, scroll bars, progress bars) is available than in more specialized tools, and their use will be familiar to students;
- Microsoft Visual Studio has more advanced debugging features than available in more specialized programming environments;
- WPF makes it easy to re-size animations to fit on different size screens (a real issue with the earlier VB6 animations was the difficulty in sizing the animations to be useable on anything from a netbook to a high-definition monitor);
- Developing with WPF allows the application to run from within a web-browser using the Silverlight framework, extending access to the programs;
- The development tools required are free. Any student wishing to look at the source-code or modify the demos can do so at no cost.

Disadvantages of the choice include:

- The requirement to learn a new programming language;

- Native running of the simulations is restricted to computers using the Windows operating system (although users of other systems will be able to run the programs from within a web-browser);
- The more advanced mathematical routines required for some simulations are not available in libraries already built into the language itself.

We now present a few examples of the programs written for the course.

Example 1 – CRC Generation

A cyclic redundancy check (CRC) is a sequence of bits that can be appended to any frame, and used to detect errors at the receiver. The check-bits are determined from the remainder left after a process of dividing the frame, treated as a very long binary number, by a particular divisor. The technique is widely used at the data-link layer, due to a particularly elegant hardware implementation that calculates the required check-bits. The operation of the circuit is, however, not obvious, and difficult to illustrate with static diagrams.

An animated simulation was written, showing the frame arriving at the receiver and being fed bit-by-bit into a simple circuit that calculates the required remainder. This is much faster (and more reliable) than working through an example by hand, and better able to illustrate exactly how the circuit operates. It also allows an easy way to involve the class in a discussion about the best choice of divisors, and immediately test out their suggestions (a choice of 1000, for example, is a very poor choice, and the reason why can be illustrated very successfully using the simulation).

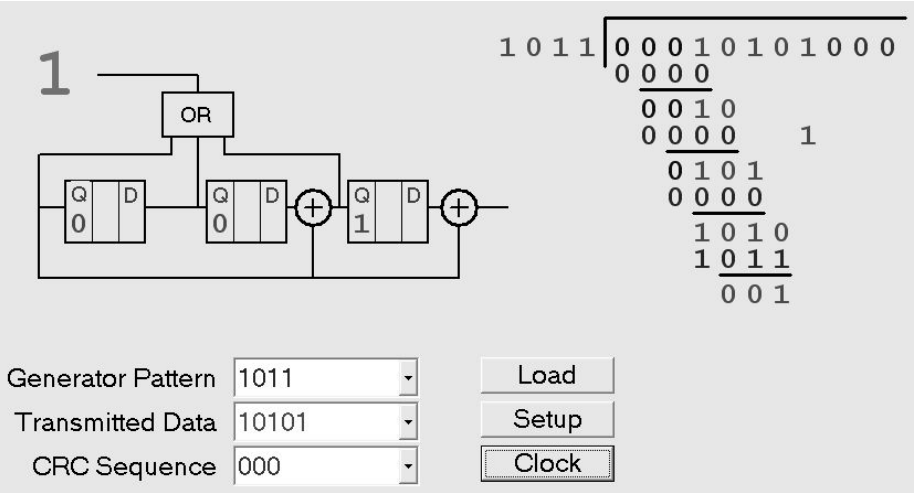

Figure 1 Screenshot from the CRC Checking Simulation

In the screenshot shown above (Fig. 1), a receive circuit is checking a frame arriving with the data field 10101 with a CRC sequence of 000 and a divisor of 1011. If the frame has been received with no errors, then after the entire frame has arrived, there should be no remainder after the division, and the output of the OR gate should be zero.

Example 2 – The Bellman-Ford Algorithm

The Bellman-Ford algorithm is used by routers to build and maintain routing tables. In principle the operation is quite simple: each router sends a copy of its routing table to its neighboring routers which can then calculate how far each of their neighboring routers is from any destination network, and therefore which router provides the best path to that destination.

However, when a link goes down, the Bellman-Ford algorithm can show some unexpected emergent behavior, with routing loops being generated (a routing loop exists when a packet is sent around in a loop between routers, never reaching the destination). Again, it is not immediately obvious how this can happen, or what can be done to prevent it. In the screenshot below (Fig. 2), a six-router network is illustrated, with the routing tables shown at each router a short time after the link from router B to E has broken down. Currently there is a routing loop around routers A, B and C: any packet being sent to destination network f will be passed around between these routers until the packet times out, despite the existence of a perfectly valid path to the destination.

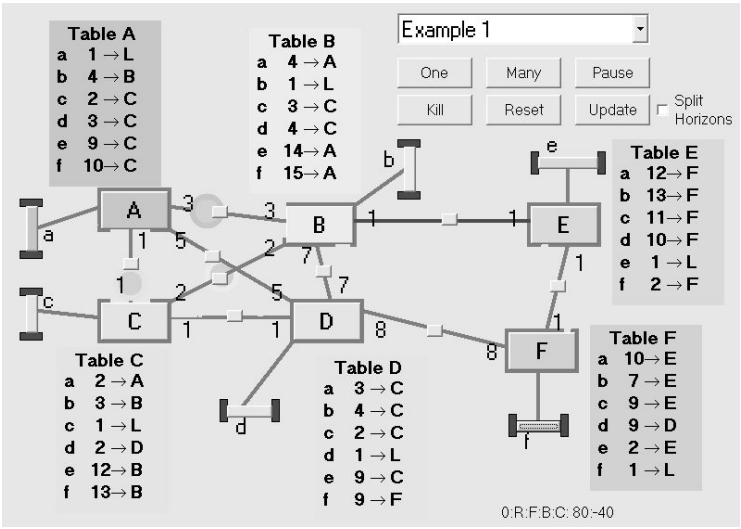

Figure 2 - Screenshot from the Bellman-Ford Simulation

Illustrating the operation of the Bellman-Ford algorithm, and then breaking a link, and showing a routing loop being formed surprises most students, and raises interest in studying exactly how they occur and what can be done to prevent them.

Example 3 – Multiple-Access Protocols

The task of a multiple access protocol is to determine when each potential transmitter should be allowed to transmit onto a shared medium. Multiple-access protocols should try and minimize delay and collisions, while maximizing the throughput of the network as a whole. In many cases, expressions for the delay and throughput of these protocols can be analytically derived, and in these cases it is particularly important for students to have a clear understanding of the operation of the protocols.

The demonstration written to accompany this section of the course allowed the throughput to be simulated, and this proved both interesting and potentially troublesome (when the simulated result didn't match the derived result). In these cases, the students preferred to believe the result of the simulator, and were more likely to ask what was wrong with the derivation, rather than what was wrong with the simulator. In the screenshot below (Fig.3), a packet is shown moving along the bus, the various colors in the packet representing the structure of a typical frame (destination address, source address, data, and checksum). In this case there are only seven potential transmitters in the simulation; however this number can be changed from three to twenty: many of the simplified derivations of the performance of multiple access schemes assume a very large number of transmitters, and it is interesting to demonstrate how performance changes for more practical numbers.

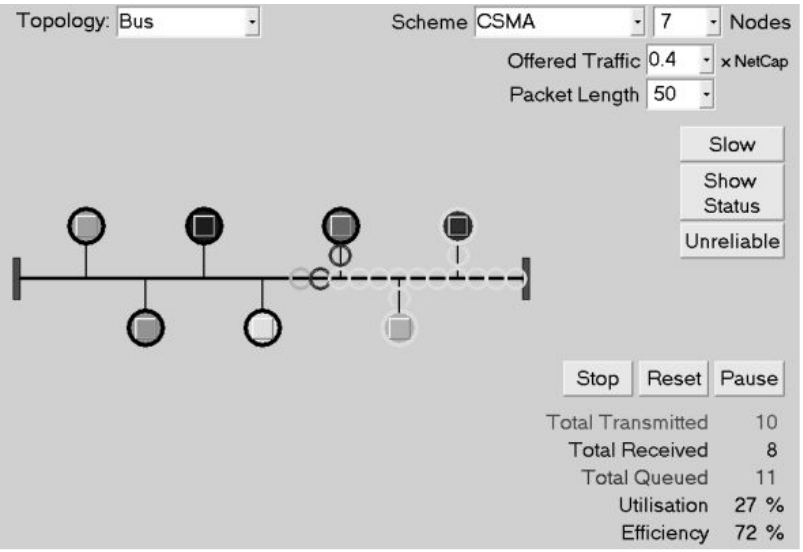

Figure 3 - Screenshot from the Multiple-Access Scheme Simulation

Another unexpected benefit of this technique became evident when developing this program, in that a seemingly spurious result revealed something that the tutor had not fully understood (a case in which the 802.11 Distributed Coordination Function (DCF) multiple-access scheme does not prevent a collision between two data packets).

Evaluating the Technique

The technique was evaluated both by a questionnaire filled out by all students who attended a typical lecture, and by an interview with student volunteers conducted in confidence by an external reviewer.

The external interviewer commented that "the response was overwhelmingly positive, enthusiastic, and supportive of the tutor's rationale and belief that he is succeeding in his aims". Of 31 students who returned questionnaires, 25 strongly agreed, and 5 agreed with the statement that "having seen an animation helped me follow the content of a lecture"; 26 strongly agreed and 5 agreed with the statement that "the animations help me visualize the processes in a communications systems", and 24 strongly agreed and 7 agreed that "being able to visualize the processes in a communications system helps me to understand it". The latter result provides

evidence to support the theory that the animations have a positive effect on learning, as no directly comparable measurement of the effects on learning with and without the programs could be made.

All students returning the questionnaires agreed (17 strongly) that "the computer animated demonstrations helped increase my depth of understanding"; with 16 students agreeing and 10 strongly agreeing that "the use of these computer demonstrations will help me solve problems relating to communication systems". No student agreed with the statements "sitting through animated demonstrations in lectures was boring", or with the statement "the computer animated demonstrations are unnecessary".

Also notable was the effect on student motivation: a majority (18 out of 31) students agreed or strongly agreed with the statement that "the computer animated demonstrations helped increase my motivation to learn". Despite it not being required, a similar majority had downloaded and used the animations in their own time to study outside of lectures: this figure was substantially greater than the number who claimed to have read any supporting material from textbooks or the Internet. This was attributed to the greater engagement of the students with the simulation programs, and the preference of students for a more active learning activity.

The most common negative comment from the students was the fact that there were no instructions provided with the simulations, making them difficult to use outside lectures; other criticisms were that some simulations were too complex, with too much happening at once, making the operation of the whole system difficult to follow; that not all of the features of the simulation were used in the lectures; and that the look of them was unappealing. (The first and last of these points are simple to address, with a set of video tutorials in preparation, and a conversion of the demos to a programming language more easily able to adapt the user interface to the different requirements of lectures and netbook computers.)

The fact that these students were used to a style of teaching in which the lecture was the primary source of information was confirmed by the questionnaire, with 25 out of 27 students ranking lectures as the most important source of information for their learning, with "the animated demonstrations" and "working through set problems" coming second and third. Textbooks, other texts and the Internet came a poor fourth, with only four out of 27 students ranking each of these possibilities amongst their top three sources of information.

Despite this lecture module requiring the students to read notes outside of the lectures to compensate for the time used with the programs, no student objected to this; the most common comment made in response to a request for suggestions about how the demos could be used differently was to "use more of them". The lecture style permitted by the simulations was praised, being described as "interactive", the tutor "not talking at us, just showing us how it all works: this makes it interesting", and that changes could be illustrated "at the click of a button, rather than having to draw another ten waveforms on the board" which would be "direly boring".

From the tutor's perspective, the only significant disadvantage was the time required to write the programs; however it was found that once proficient in a suitable programming language, useful and engaging animations and simulations could be written in an evening; and many of the

simplest examples used proved the most effective. This latter effect was attributed to the more focused nature of the programs, and the reduction in distractions caused by features of the programs not being used at the time[2].

An incidental but nonetheless important factor was the enjoyment of the tutor, as writing and showing animations proved much more fun than drawing and presenting a series of diagrams. Writing the programs became an enjoyable hobby and lectures gave an immediate reward of attentive and grateful students. As many of the animations are also simulations with a random element, the tutor found that his mind was also kept more active in the lectures, as he had no idea exactly what would happen either.

Conclusions

In most respects, the experiment to base a taught course around a series of animated demonstrations has been entirely successful, with increased attendance at lectures and increased student satisfaction with the course. One predicted outcome (that provision of a full set of notes before the course started would reduce attendance at lectures) did not occur, as students reported finding the lecture more engaging and useful than more traditional lecture styles. Students also reported that the programs helped them to gain a deeper understanding of the subject. The only disadvantage found with the technique was the amount of time required to prepare the programs.

The key conclusions from the study are that it is better to spend time writing a larger number of simple programs than a smaller number of more advanced simulations; that care must be taken to write the programs so that they can be run on a variety of screen sizes; that students do not object to the lecturer not introducing all the required material in the lectures provided they see the benefits of the lectures; that programs that allow some simulation are more popular and more likely to be used by students outside the timetabled classes; and that there are real advantages in terms of the rewards and motivation of the lecturer in writing simple bespoke programs, even when existing programs are available.

The programs used in the study are available at http://www-users.york.ac.uk/~dajp1/Demos

Bibliography

1. R.M. Felder and L.K.Silverman, “Learning and Teaching Styles in Engineering Education”, Engineering Education, vol.78(7), pp. 674-681, 1988.
2. R. Fleischer and Ludek Kucera, “Algorithm Animation for Teaching”, Lecture Notes in Computer Science, Springer, Volume 2269/2002, pp.640-642.
3. The Network Simulator, online. http://nsnam.isi.edu/nsnam/index.php/Main_Page.
4. L.P. Rieber, “Animation in Computer-Based Instruction”, Educational Technology Research and Development, Vol.38, No. 1, pp. 77-86, March 1990.
5. L. Davison and N. Porritt, “Using Computers to Teach”, Proceedings of the Institute of Civil Engineers, vol.132, pp. 24 -30, February 1999.
6. J.C. Reis, “Experiments with Various Teaching Methodologies”, Frontiers in Education Conference, 1998.