Comparative Advantage Learning Software: Application (Off-line) Software with Assessment Capabilities

John Lovett
Department of Economics, Texas Christian University

Abstract

This paper describes software, written by the author, for learning comparative advantage and the specialisation gains from trade. Starting with given absolute costs and available resources for two nations, the student constructs production possibilities curves. He or she then picks a term of trade and feasible levels of exports and imports at which both nations can gain from trade.

This non-commercial software is application software rather than a web application. As such, it is easier to develop than comparable web applications. Traditionally, application software has lacked the relatively secure assessment (i.e. grading) capability of web applications. The software described in this paper, however, produces a verifiable assessment by producing a code based on the student's identifiers and his or her score. This code is then passed to the instructor who, using other software, decodes it.

Introduction

One of the more famous stories in economics is the friendly exchange between the mathematician Stanislaw Ulam and Paul Samuelson. Ulam asked Samuelson to 'Name me one proposition in all of the social sciences which is both true and non-trivial.' Roughly 30 years later, Samuelson, finally came up with a reply: David Ricardo's theory of comparative advantage. Paul Samuelson writes, 'that it is not trivial is attested by the thousands of important and intelligent men who have never been able to grasp the doctrine for themselves or to believe it after it was explained to them' (Samuelson, 1969: 9). Similarly, Paul Krugman, in his recent essay 'David Ricardo's Difficult Idea', states: 'The first thing I need to do is to make clear how few people really do understand Ricardo's difficult idea' (Krugman, 1996: 1).

These examples illustrate the dilemma an instructor faces when presenting the concept of comparative advantage. The concept is central to economics. It explains how all parties can gain from trade despite different abilities. Explaining the concept in a way that students can understand and apply is, however, no easy task.

This paper presents a non-commercial computer application, called 'Mon-kee-con', written by the author. It is designed to lead students to an understanding of both the mechanics and intuition of David Ricardo's 'difficult idea'. The software consists of three different comparative advantage learning modules. The first and shortest comparative advantage module quizzes students based on a table of absolute costs for two different nations. The second and longest module, termed 'The Full Deal', begins by quizzing students on the difference between absolute and comparative advantage. Then, starting with a set of absolute costs and given level of resources, the student builds production possibility curves for two nations. He or she picks a term of trade and then picks exports and imports that are feasible (i.e. matching) and allow both nations to gain from trade. This paper focuses on this, 'The Full Deal', module. The final comparative advantage module starts with already drawn production possibility curves, and then asks students a series of questions about gains from trade.

The software is not a 'web application', i.e. it is not web based. Instead, it runs solely on the user's machine. This is commonly referred to as 'application software'. The software was written in Visual Basic and runs on Windows operating systems. One advantage of application software is that it is much faster and easier to create rich application software than it is to create similar web applications. Web applications, nonetheless, have traditionally offered some advantages over application software. One of these is that web applications offer assessment capabilities whereas application software does not. Mon-kee-con, however, does offer assessment capabilities. As such, Mon-kee-con might serve as a model for other educational application software.

At the end of each session, the software encrypts the student's score, module played and the student's identifiers. The resulting code is then passed to the instructor for decryption. Currently none of the passing of data, decryption or recording of grades is automated. Further, Mon-kee-con does not have the ability to limit the number of attempts. In short, one might consider Mon-kee-con an initial illustration of combining application software with assessment, not a fully refined model.

The author has been using this comparative advantage software, and similar Mon-kee-con supply and demand exercises, in his Principles of Microeconomics course since 2000.

The learning software

Upon starting Mon-kee-con, a student enters his or her identifying information (name and student ID), choosing which 'module' to play. Mon-kee-con has a total of six modules: three supply and demand and three comparative advantage. This paper overviews the main comparative advantage module called 'The Full Deal'. Mon-kee-con also contains a set of supply and demand modules in which students must shift the appropriate curve and interpret the results in response to various market changes. These, however, are beyond the scope of this paper.

'The Full Deal' begins with a modified true/false question overviewing the gains from trade (Figures 1 and 2). Mon-kee-con displays five options chosen from a possible set of 20. Three correct options appear out of the five choices. One option (worth the most points) deals with Ricardian gains from trade. The other two correct options deal with other gains from trade. As with most questions, the student is given feedback before he or she can continue.

The student is then given a table with two nation's absolute costs, in terms of labour hours, for two goods. The program pseudo randomly picks one of approximately 1,200 number combinations. In the vast majority of outcomes, one nation has an absolute advantage in both goods. Occasional instances of each nation having an absolute advantage in both goods are, however, allowed.

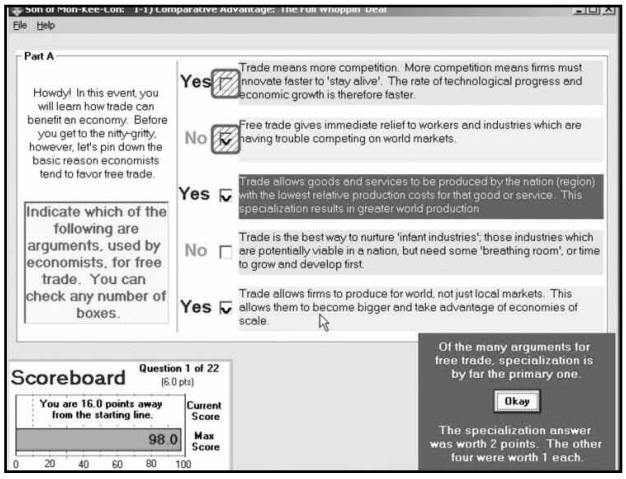


Figure 1. The opening questions

The student next picks which nation has an absolute advantage in each good (Figure 2). He or she then calculates the opportunity costs in terms of other goods which could be produced (comparative or relative costs (Figure 3). After that, the student picks the comparative advantage for each good (Figure 4).

The student then builds a production possibilities curve for each nation (Figure 5) based on a set number of available

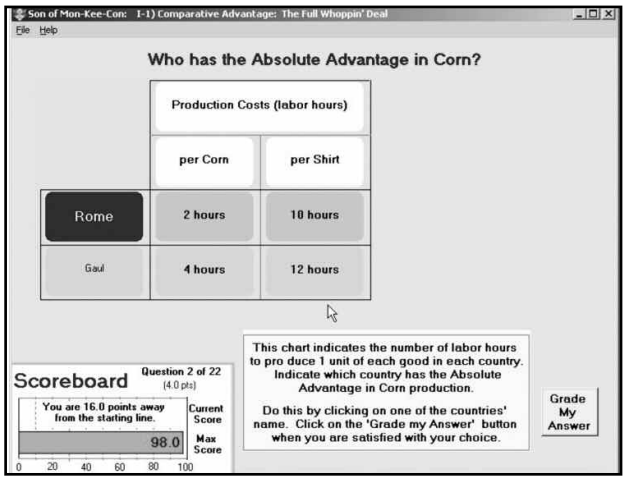


Figure 2. Choosing absolute advantage

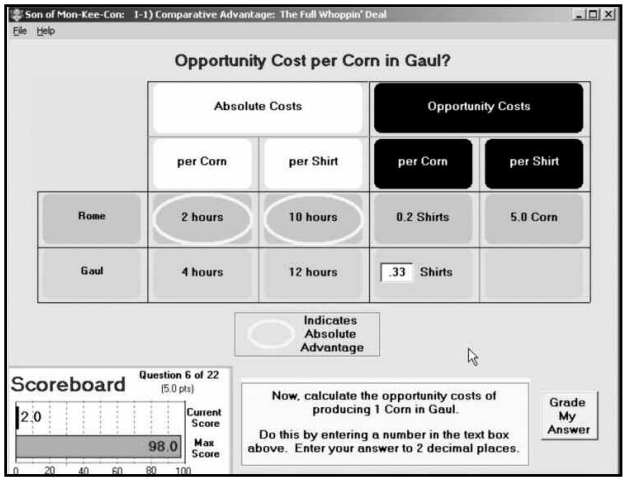


Figure 3. Choosing opportunity cost

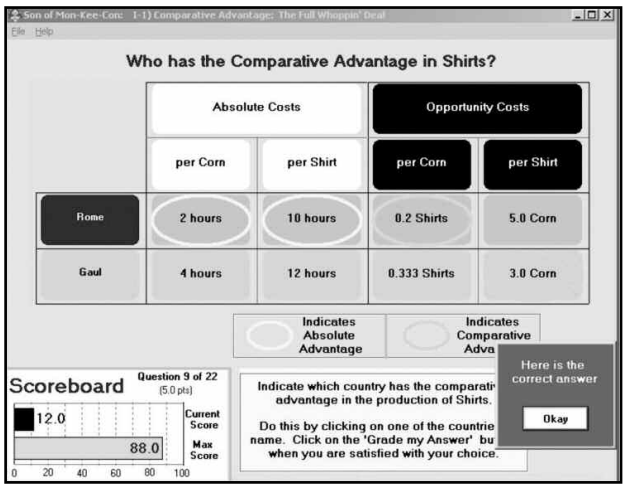


Figure 4. Choosing comparative advantage

labour hours. She is next asked to indicate which points are possible consumption points in autarky (Figure 6).

Next, the student is asked to demonstrate how both nations can gain from trade. He or she first must pick a viable terms of trade (Figure 7). Then, in a three-step process, indicates each nation's production (assuming complete specialisation), chooses feasible exports and imports, and finally indicates the resulting consumption levels (Figure 8).

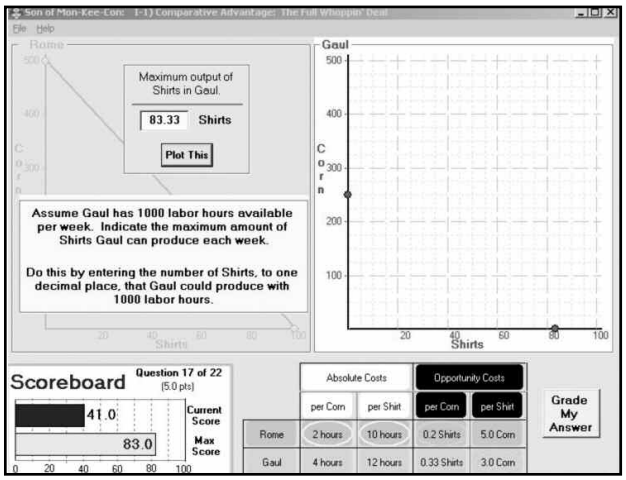


Figure 5. Building the PPC

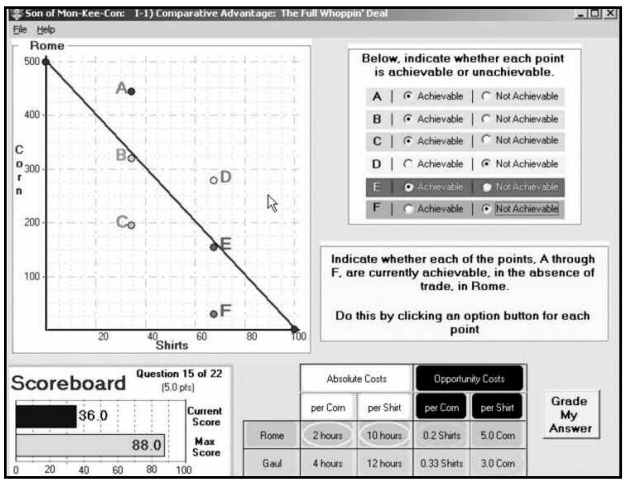


Figure 6. Constrained by our PPC w/out trade

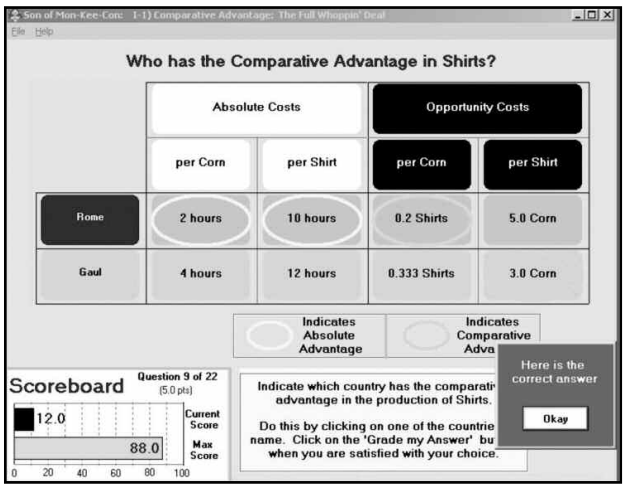


Figure 7. Choosing a terms of trade

As the student attempts the module, he or she has to click through a number of explanations offered by David Ricardo (Figures 9 and 10).[1] After playing a module, the student can print out the resulting score along with their student identifiers and a code used to verify the score. The verification process is discussed in the next section. The student can attempt a module as many times as he or she wishes. As a result, students generally play the module many times and the scores turned in are relatively high.

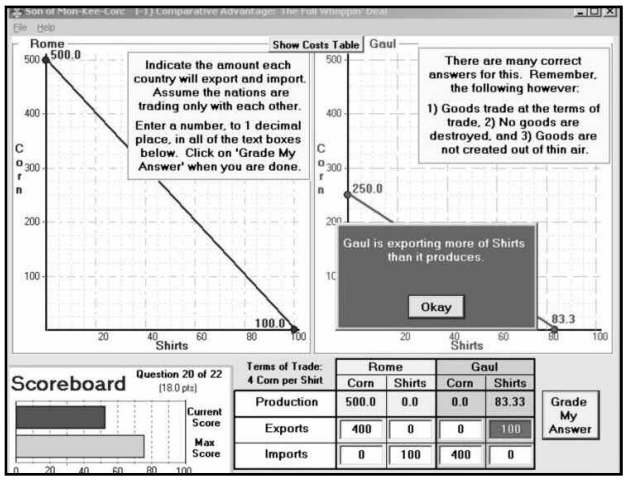


Figure 8. Choosing exports and imports

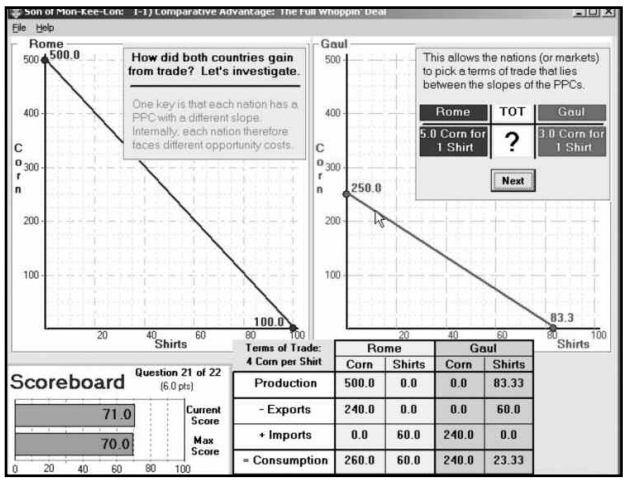


Figure 9. One of David's many explanations

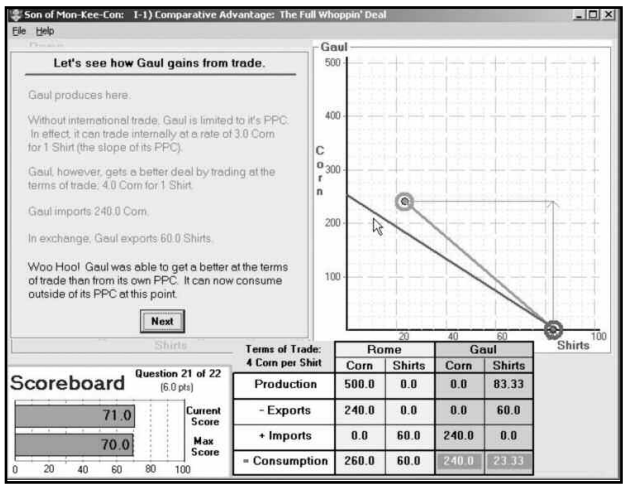


Figure 10. Another of David's explanations

Assessment features: lowering entry barriers

In economics education, there has been a definite movement away from application software to web applications over the last decade or so.[2] Between the mid 1980s and mid 1990s, journal articles surveying economics learning software concentrated on applications software.[3] Many texts were also bundled with application software (e.g. Byrns and Stone, 1996a and 1996b).

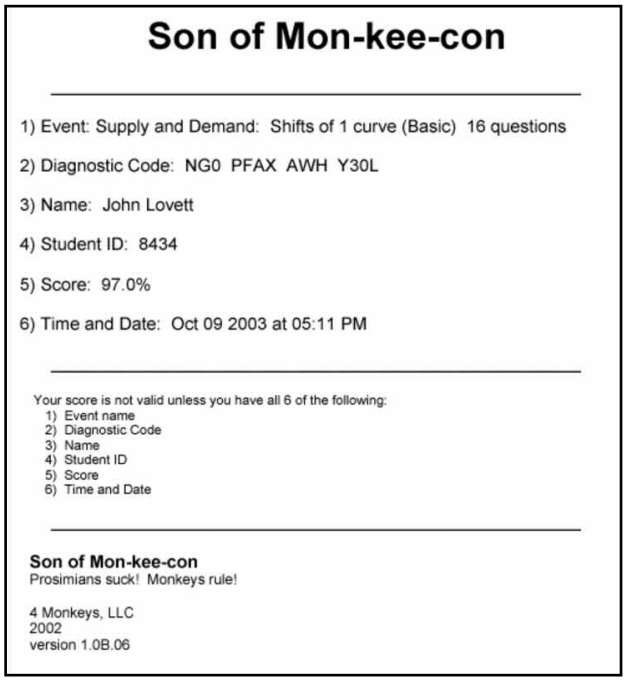
Approximately a decade ago, a shift to web applications occurred.[4] In the late 1990s two major publishers, Irwin/McGraw-Hill and Harcourt, 'pulled the plug' on application software projects that were, respectively, near a beta release and already into a beta release.[5] With the exception of many surviving Excel® based problem sets, web applications are the norm for economics education software today. *Aplia* (2006), *ThomsonNow* (Thomson, 2006), *EconX* (Beginner's Mind Inc, 2007), are examples of current web applications.

More recent evidence of this shift is the conversion of *SimEcon* from application software to a web application. When I began this paper I planned to list *WinEcon* (WinEcon Consortium, 2006) and *SimEcon* (Bresnock and Garston, 2003, 2005) as the two primary examples of remaining application software packages in economics. However, it was recently announced that *SimEcon* is being recoded as a web application (Gartson, 2006).

An obvious attraction of web applications is their assessment capabilities. Web applications can easily produce verifiable grades for a student's work. One can argue, however, that something has been lost in the shift to web applications. In particular, it is easier and faster to develop visually rich, highly interactive application software, than it is for web applications. Less than a year of learning a 'rapid application development' programming language such as Visual Basic®, VB.Net®, Real Basic® (for Macintosh® OS's), or even Java® (presumably using a third party development package) allows a single person to create decent applications. Visual Basic, VB.Net, and Real Basic are particularly fast and easy.

For a given level of time and resources, application programming leads to richer graphics and more interactivity than web application programming (Sequin, 2005, Spolsky, 2004). This is evidenced by the fact that many freeware and low-end commercial application software offers richer visual environments and interactivity than do high-end commercial web applications. Further evidence is simply the fact that we still use application software for most of our computer needs today. A few years ago there were predictions that web applications would replace application software in everything from office software (word processing, spreadsheets, etc.) to graphics packages. This has simply not come true.

Because of its lower development costs, application software may be a means to increasing the supply of educational software in economies. In 1992, Porter and Riley (1992: 376–377) listed development costs as a major impediment to a large supply of learning software in economics. This is still true today. While there is competition in the market for web application packages, it is better described as oligopoly than monopolistic competition. If application software can be given many of the desirable features of web applications, namely assessment capabilities, barriers to entry will be lowered. A greater variety of usable software could be the result.



Son of Mon-kee-con

1) Event: Supply and Demand: Shifts of 1 curve (Basic) 16 questions

2) Diagnostic Code: NG0 PFAX AWH Y30L

3) Name: John Lovett

4) Student ID: 8434

5) Score: 97.0%

6) Time and Date: Oct 09 2003 at 05:11 PM

Your score is not valid unless you have all 6 of the following:
1) Event name
2) Diagnostic Code
3) Name
4) Student ID
5) Score
6) Time and Date

Son of Mon-kee-con
Prosimians suck! Monkeys rule!

4 Monkeys, LLC
2002
version 1.0B.06

Figure 11. Student results printout

The author has added assessment capabilities to Mon-kee-con using the following method. After a module is completed, Mon-kee-con encrypts the student's score, in combination with identifiers describing the student and the assessment he or she has just taken, into a code. Currently a Feistel cipher with a key length of 440 bits, and many pseudo-random inputs, is used. A more modern encryption technique could easily be substituted. Mon-kee-con then generates a printout of this information. The student prints the information and then hand-carries the page to the instructor (Figure 11). The code is termed 'diagnostic code' in Figure 11.

The instructor or her assistant, then hand enters the score code into decryption software to confirm the student's score (figure 12).

This method is open source. Similar systems (but with electronic passing of the information rather than hand-carrying) have been used as third party supplements to gaming software since at least 1992.[6]

This system does not limit the number of attempts as can be done with web applications. It is also more time consuming for the instructor. Automated passing, decryption and storing of information, discussed in a later section, would alleviate this. Nonetheless, it does produce assessments that are roughly as secure as that offered by web applications.[7] Like web applications, unless one has physical control of the testing site, an instructor cannot verify that the student turning in the assignment did it without assistance from a textbook, notes or other students. One can verify, however, that someone using his or her identifiers used the software to score the printed results.

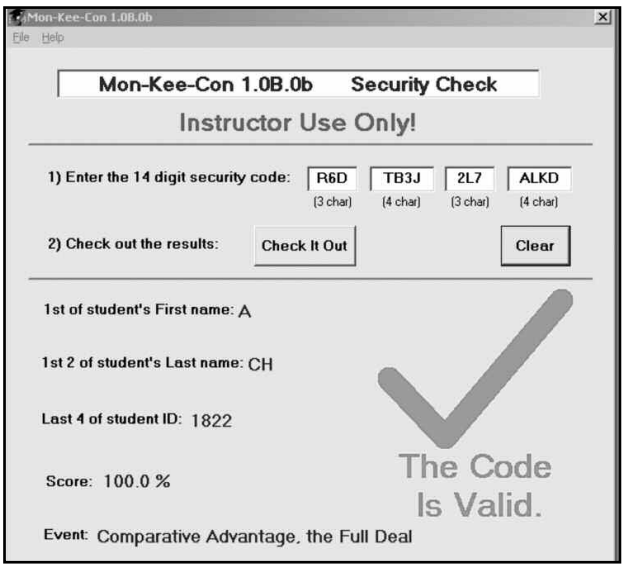


Figure 12. Verifying the results

The software as a learning tool

Daniel suggests three criteria for effective computer based learning applications. These are: ' (1) various stages of difficulty, (2) flexibility in application, and (3) opportunity for experimentation' (1999: 164).[8] When designing Mon-kee-con, a deliberate attempt was made to meet the first two of these criteria. The three comparative advantage modules offer three degrees of difficulty. Further, the modules allow students to restart as many times as they wish. Students often play until they reach a point at which they are making many mistakes, then restart the module and begin again. By doing this, students go through the earlier stages, by now 'old hat', many times. The result is that each time a student plays a module he or she is ascending through increasing degrees of difficulty.

In terms of flexibility, the software is designed for use as a non-graded tutoring/teaching application as well as an assessment tool (i.e. graded). Opportunity for experimentation is more limited. The software does, nonetheless, generate different numeric combinations each time. When a student uses the software repeatedly, he or she is practising on different numeric situations.

I have not conducted a formal assessment of the software's effectiveness in teaching comparative advantage. Nor have I formally accessed how much students enjoy it. Nonetheless, I have found it a useful learning tool for the past six years. Further, Mon-kee-con is the most commonly cited 'favorable aspect of the course' students listed in their course evaluations. How much of this is due to the repeatable nature of the software (and therefore high scores) and how much is due to the software's basis as a learning experience, I do not know. It does encourage me, however, to continue to use the software in future semesters. Further, one could easily argue that the repeated attempts the software encourages lead to significant learning.

Possibilities for the future

Currently, the author is not planning to add additional learning modules to Mon-kee-con.[9] There are plans to further develop the assessment process. In particular, he plans to add increased automation in the reporting, decryption and storage of each student's results. In the first phase, after a student plays a round of Mon-kee-con on his or her computer, he or she could then log onto a page on the author's website. The student would then enter his or her 'score code', name and ID. The web page would then, via encrypted asp coding, decipher the code and record the results in the Microsoft Access® database on the author's website. This would greatly ease the burden of entering score codes on the instructor. The ease of assessment in Mon-kee-con would truly be akin to that offered by online learning software.

Fully automated passing of information, i.e. the software rather than the student passing the information, is possible. However, the author has no active plans to modify Mon-kee-con in this fashion.

Notes

1 The screen shots are from a version of Mon-kee-con in which the author failed to add correctly. The total number of points possible added to 101, not 100.

2 There is a continuum in the degree of sharing the processing duties in web applications. In one extreme the only thing the client machine does is run a browser that interprets html sent by the server. This is often referred to as 'thin-client' because the client machine is doing relatively little. There are also 'thick client' applications in which the server sends information to be used in Java applets or Macromedia®'s Flash® running on the client machine. AJAX and other 'rich-client' applications promise to provide for even 'thicker' client application.

3 See, for example, Beckman (1987), Blecha (1991), Boyd (1993), Dalgaard, Lewis, and Boyer (1984), Khandker and Wehrs (1990), Walbert (1989), and Yoho (1986).

4 See, for example, Daniel (1999). All see Walstad *et al.* (1998).

5 McGraw-Hill-Irwin cancelled a project headed by Ralph Byrns titled 'Economics Interactive' in 1997 after spending nearly $1 million on it. The author was a scripter for this project. Samples from 'Economics Interactive' can be found at Ralph Byrns' website; http://www.unc.edu/depts/econ/byrns_web/Software/Software.htm. Harcourt developed and offered 'Archipelago', a roughly similar CD-Rom based project (see Talley, 2001). Archipelago, however, was dropped shortly after its introduction soon after Thomson acquired Harcourt.

6 Players of Prodigy's Golf Network Tour developed a roughly similar, and open source, process. While I have not identified the originator of this method, numerous e-mails have verified that it was in use by at least 1992.

7 Unlike web applications, Mon-kee-con's system does not introduce the possibility of an attack on the server with the student scores. 'Plaintext attacks' are, however, possible. In a plaintext attack, the code-breaker can view the code resulting from known inputs (i.e. the students, score, module played, ID, etc) (Stallings, 1999: 21 – 26).

This, however, is less probable than it may seem. First, it takes quite a bit of effort, playing the game consistently many times, to generate a good set of score codes. Second, because of the random numbers included, there are 784 possible 'correct' score codes for every combination of last name, first name, student ID, game descriptor, and score. Third, Feistel ciphers switch bases. Each letter of the score code is 5 single bits each of which map to different parts of the encrypted information.

The letter 'A' in the third position may, for example, contain partial information about the student's name, ID, and score, plus two random elements. Likewise, information about the student's name (or any other relevant) information can affect multiple letters of the score code. See Stallings (1999: 21–26) for an overview of types of attacks on encrypted messages.

[8] Daniel (1999) derived his list of desirable features from Walbert's (1989) much longer list.

[9] At one time the author planned to greatly expand Mon-kee-con to include other learning activities. With this goal in mind, a company (4 Monkeys, LLC) was formed in 2001 with the author, Ryan Duryea, Edwin Wong, Mike Rangel. Most of the company's resources were spent making sure the process is truly open source. In 2003, 4 Monkeys, LLC was dissolved.

References

Aplia Inc. (2006), 'Products', retrieved 6 October 2006 from Aplia website: http://www.aplia.com/ .

Beckman, Steven (1987) 'A Microcomputer Program that Simulates the Baumol-Tobin Transactions Demand for Money', *Journal of Economic Education* 18 (3): 309–317

Beginners Mind, Inc. (2007) 'ECONX: Experiments in Economics', retrieved 20 November 2007 from: http://www.econx.com/.

Blecha, Betty (1991) 'Economic Pedagogy and Microcomputer Software', *Social Science Computer Review*, 9 (4): 541–557.

Boyd, David, (1993) 'The New Microcomputer Development Technology: Implications for the Economics Instructor and Software Author', *Journal of Economic Education*, 24 (2): 113–125.

Bresnock Anne and Neil Garston (2003, October) 'SimEcon: Design, Usage and Assessment', paper presented at the *West Coast Teaching Economics Conference*, Fullerton, California, USA.

Bresnock, Anne and Neil Garston (2005, June) 'Designing and Testing Simulation Software for Economic Education, The Case of SimEcon', paper presented at the *4th Global Conference on Business and Economics*, Oxford, England.

Bryns, Ralph and Gerald Stone (1996a), *Economics*, Addison-Wesley.

Byrns, Ralph (1996b), *Ecostudy* [computer software], Retrieved 2 October 2006 from Ralph Byrn's University of North Carolina at Chapel Hill website: http://www.unc.edu/depts/econ/byrns_web/Software/EcoStudy/Installing_EcoStudy.htm

Dalgaard, Bruce, Darrell Lewis, and Carol Boyer (1984) 'Cost and Effectiveness Considerations in the Use of Computer-Assisted Instruction in Economics', *Journal of Economic Education*, 15 (4): 309–324.

Daniel, Joseph (1999) 'Computer-Aided Instruction on the World Wide Web: The Third Generation', *Journal of Economic Education*, 30 (2): 163–174.

Gartson, Neil (2006), *Sample SimEcon Conversion*, Retrieved 19 October 2006 from http://instructional1.calstatela.edu/ngarsto/SimEcon/.

Khandker, A. Wahhab and William Wehrs (1990) 'Integrated Microcomputer Graphics and Simulation in Open-Economy Macroeconomics', *Journal of Economic Education*, 21 (2): 167–180.

Krugman, Paul (1996) 'David Ricardo's Difficult Idea', Retrieved 6 October 2006 from the Massachusetts Institute of Technology website: http://web.mit.edu/krugman/www/ricardo.htm .

Lovett, John (2004) 'Off-line Comparative Advantage Games Computer Games', *Papers and Proceedings of the 2003 Economics in the Classroom Conference*, 98–108.

Porter, Tod, and Teresa Riley (1992) 'CAI in Economics: What Happened to the Revolution?', *Journal of Economic Education*, 23(4): 374–378.

Samuelson, Paul (1969) 'Presidential Address of the Third Congress of the International Economics Association', *International Economic Relations: Proceedings of the Third Congress of the International Economic Association*, edited by Paul Samuelson, London: MacMillan.

Sequin, Karl (2005), 'ASP.NET Spiced: AJAX', Retrieved 7 October 2006 from Microsoft website: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnaspp/html/ASPNetSpicedAjax.asp.

Spolsky, Joel (2004), 'How Microsoft Lost the API War' Retrieved 7 October 2006 from the website: http://www.joelonsoftware.com/articles/APIWar.html

Stallings, William (1999) *Cryptography and Network Security* (2nd ed.), Upper Saddle River, NJ, Prentice Hall.

Talley, Daniel (2001) 'Microeconomics Online', *Campus Technology*, Retrieved 6 October 2006 from the http://www.campus-technology.com/article.asp?id=3699.

Thomson (2006), 'Take a Tour', Retrieved 6 October 2006 from the ThomsonNow website: http://www.thomsonedu.com/thomsonnow/tour/ .

Walbert, Mark (1989) 'Writing Better Software for Economics Principles Textbooks', *Journal of Economic Educatio*n, 20 (3): 281–289.

Walstad, William, Ann Fender, Jean Fletcher and Wayne Edwards (1998) 'Using Technology for Teaching Economics', in *Teaching Undergraduate Economics: A Handbook for Instructors*, W. Walstad and P. Saunder (eds), 269–283, Boston, Irwin/McGraw-Hill.

WinEcon Consortium (2006), 'WinEcon: award winning software', Retrieved 6 October 2006 from WinEcon's website: http://www.winecon.com/

Yoho, D.L. (1986) 'Captive Microcomputer Software for Economic Instruction: A Review', *Social Science Microcomputer Review*, 4 (4): 425–437.

Contact details

John Lovett
Instructor
Department of Economics, Texas Christian University, USA

Email: j.lovett@tcu.edu
Web page: http://facukty.tcu.edu/jlovett