Texas Hold’em Poker Rules

- Players receive two cards as their hand.
- There is a round of betting where you have the following options:
  - Fold: You don’t put in any more money, but you are out of the hand and cannot win any money.
  - Call: You put in money equal to the amount bet. You stay in the hand. Play continues and the next card is dealt, unless someone else raises.
  - Raise: You put money into the pot equal to the amount bet so far, plus additional money (the raise). Now, all the other players must call your raise or fold.
- Three boardcards are turned simultaneously (known as the ‘flop’).
- Another round of betting occurs.
- A boardcard is turned, with a round of betting.
- The final boardcard is turned, with a round of betting.
- All players still in the game reveal their hands and a winner is decided to take the pot.

The five cards on the table are community cards. Your hand is formed by taking the best hand using any five out of seven cards (5 community + 2 hand cards). All the players have access to the community cards.

Welcome Back to iCS!

The University of Leicester reached 50 years of age in 2008 and “Celebrate Leicester Day” was held on Saturday the 26th April. Visitors joined the Computer Science Department in a computer-based poker tournament held in the Charles Wilson labs and BBC Radio Leicester covered the excitement.

Dale Such created “Suchy’s Poker Tournament” for his final year project when studying Computer Science at the University of Leicester. It was programmed in Java, a language Dale learned in his first year. The software allows the play of poker tournaments against human opponents over a network, in much the same way as you might enjoy multi-player games on your home computer or favourite video console.

Also on “Celebrate Leicester Day”, Dr. Michael Hoffman, Dr Stuart Kerrigan and Dr. Stephan Reiff-Marganiec of the Department of Computer Science were involved in judging the “Leicester Invents” competition.

Continued on page 3
Ask iCS

Dear iCS, I am starting a university course in Computing and I am wondering if you could explain the difference between how courses are taught at school and at university? Are courses taught only by lectures?

Lectures are not the only method of teaching used. For example at Leicester, surgeries, problem classes, laboratories, and student projects are used to teach Computer Science.

Surgeries usually revisit material already covered in lectures. The emphasis is on assisting students with problems, understanding key concepts, and clearing up special difficulties. Quite often advice about coursework is offered. Surgeries may run in the style of a lecture, or alternatively as a session in which staff are present and students may seek help when required.

Computer laboratories (practicals) form a significant part of the teaching in many courses. Students work at computers and may ask staff for help. So laboratories are quite similar to surgeries in terms of the kind of assistance students can expect.

Student projects run in the second and third years. At Leicester we run both group and individual projects. Students learn how to design and implement software systems, and often work with external clients, so that the project work is both realistic and exciting. Everybody has a supervisor who is a member of staff and offers individual help and guidance.

Do you have other questions about Computer Science? What would you like to see appear in future issues of iCS? Please let us know by writing to ics@mcs.le.ac.uk.

If you would like your questions or comments to appear in iCS please give contact details, including your name, age, school year and school address.

The Psychologist’s Perspective

By Effie Law

The marriage between Psychology and Computer Science advances scientific knowledge and the quality of life in general. Let’s look at a scenario you may be familiar with:

Simon, a novice computer user, wants to locate learning resources in a digital library, relevant to a project task. The access is password controlled. Simon has already encountered several problems simply logging in: finding the login button on the homepage that was situated at an unconventional place, having forgotten his weirdly formulated password, and staring anxiously at his email in box in anticipation of the response to his request for a new password. Finally he logs in and enters some search keywords. A plain and rather unfriendly phrase “No record is found!” is returned. Suppressing his surprise, he spots the advanced search mode button. Upon clicking it, an intimidating and complicated electronic form pops up. After several futile trial-and-error attempts, he gives up. Retreating back to the simple search mode, he refines his original keywords from “human-computer interaction” to “human being use computer machine system interaction interface relationship”. Then he is rewarded with a long list of records. However these records are of such low relevance that they are hardly of any practical use to his task at hand. He doubts whether the problems lay in the system design or in his own naivety. Ironically, what Simon experienced can exactly be classified as “issues of human-computer interaction (HCI)”, the topic on which he aimed to explore and that I research.

HCI – a highly interdisciplinary field - amalgamates expertise from Computer Science, Psychology, Sociology, Cultural Anthropology, Linguistics, and other disciplines. Viewed from the perspective of Computer Science, Simon’s frustrating experiences can be attributed to poor metadata structure, search algorithms, interoperability, and semantic web technology. These technical artifacts can be linked to psychological concepts of cognition, knowledge representation and categorization, problem solving, and mental models.

In essence, Psychology studies the human mind, which has an inseparable connection with Artificial Intelligence. Specifically, a sub-discipline Computational Psychology investigates how various psychological phenomena can be explained in computational terms. Traditional usability evaluation involves measuring effectiveness, efficiency and satisfaction with different techniques and tools. Recently, the HCI community has talked more about user experience (UX). How can we measure fun, pleasure, beauty, pride, surprise, anger (everybody at some point is angry with their computer), and other emotions? Strictly, one can measure almost anything in some arbitrary way. The compelling concern, however, is whether the measure is meaningful, useful and valid to reflect the state of nature of the object or event in question. Identifying valid UX measures is an emergent and exciting topic in HCI.
Meet a Computer Scientist

Chris Holloway obtained a BSc in Computer Science from the University of Leicester in 2005, graduating with a first-class degree. He was immediately offered a job at Bloomberg L.P., the world famous financial software services, news and data company. iCS asked him some questions about his work.

iCS: What are you doing now at Bloomberg?

Chris: I work as a systems programmer. Our core product is “The Bloomberg Terminal”, a thin-client application (an application where processing activities are performed primarily by a central server, in this case operated and maintained by Bloomberg, rather than on your own personal computer that acts as a “client”) through which users can monitor and analyze real-time financial data, place trades, view news and so on. My team writes software to support the terminal, such as software for managing market data streams, our in-house database, inter-process communication infrastructure, and core infrastructure and application libraries.

iCS: What do you find especially satisfying about your role?

Chris: I find it quite satisfying to track down an obscure bug! For example, you might have a problem that occurs only rarely, or limited information about what went wrong. It then becomes a puzzle that you have to solve.

iCS: How has studying Computer Science at Leicester helped you in your career at Bloomberg?

Chris: Just about every part of my course was beneficial! All of the subjects I studied in Computer Science have helped me to develop strong programming skills. In particular I found ‘Algorithms and Complexity’ useful - utilizing the best algorithm is nearly always better than trying to micro-optimize program code.

iCS: How did studying Computer Science influence your career?

Chris: Computer Science is a fairly specialised degree but there are a number of distinct roles you can choose from. There are a large number of positions available in the job market for those roles, so from that perspective there is a lot of choice.

Welcome Back to iCS!

Continued from page 1

The “Leicester Invents” competition is an annual event run by the University to help find budding school inventors. The invention designs this year were on show all day to visitors and there was a prize giving ceremony. Prizes included Nintendo DS and Nintendo Wii consoles as well as £200 of science equipment for the overall winner's school.

Winners included Clickncut.com, a USB peripheral providing computer-controlled haircuts (and thoughtfully a pair of earmuffs for safety) by Tom Horeckyj of Old Mill Primary, Broughton Astley; flashing cats eyes that alert drivers to black ice, by Joseph Wildsmith of Hemington Primary School; a crop growing bubble for enhanced food production, by Lauren Bate of Bushloe High School; and a wristband to indicate when its wearer is in danger of suffering sunburn, by Sarah-Louis Mutton of King Edward VII College. The overall winner was Emily Davis of Dixie Grammar School for her invention of “Improved Hospital Doors”. These doors disinfect a person's hands when they open the door.

Do you have an idea for an invention using computers you might like to share?

What connection does the year 1918 have with the University of Leicester?

If you think you can answer either of these questions then email ics@mcs.le.ac.uk and we may print your answers in a future edition of iCS. Please include your name, age, year in school and school address.
Parallel Computation on Massive Datasets: By Google Inc.

Computer Science often concerns itself with the development of new ways of programming. In undergraduate courses it is often only possible to give simple examples when introducing new ideas, so students often ask about real world applications. This short article explains ideas that originated in Functional Programming some years ago, but which have been applied by the Google Corporation in more recent years to web programming problems which involve massive datasets, some amounting to many terabytes (10^{12} bytes) in size.

When writing financial software we might encounter

Write ("Please enter, using capital letters only, your account number"); and

Write ("Please enter, using capital letters only, your name").

There may be other data entry commands. It is wasteful to repeat text in the code, so we might introduce an abstraction by coding a function

Function enter (data) ("Please enter, using capital letters only, your" + data)

allowing us to write enter ("account number"). The code becomes more compact and easier to read. Now suppose that you frequently validate accounts, credit accounts, close accounts, and set interest rates on accounts. If the account numbers are stored in an array, then you might write separate functions for each of these tasks, such as

Function validate (acc_nums) {<code to validate each account>}
and the <code to traverse array> will be repeated in each task function definition. However if we could pass a function as an argument to another function (say setup) we could write the <code to traverse array> once and for all

Function setup (acc_nums, validate)
{<code to traverse array> validate (acc_nums)}
and then the shorter task functions such as

Function validate (acc_nums) {<code to validate each account>}

In this example we have an array of data and various tasks; we perform a particular task on each element of the array. This is known as mapping and setup is an example of a map function.

Let us consider another example. We can consider the rainfall per month as an array (indexed by months).

Given a collection of such arrays, that is, rainfall data over a period of years, one can calculate the total rainfall for each individual month by adding up all of the rainfall values for that (index) month. Computations on values for the same index from different arrays are known as reducing computations.

If you run a business you may well wish to know how often certain web pages are accessed. Your computers can store (arrays of) log files of web page requests. We can count up URL access frequency as follows. First we apply a function to each of the logs; this function outputs each URL in the log, tagged with a value of 1. This activity is very similar to the mapping task above, except that we are dealing with arrays with repeated index elements. Having done this, for each individual URL we then add up all of the values of 1 to produce the access count for that URL. This is a reducing computation. Now, in the case of Google, arrays of data can often be very large indeed. Moreover the tasks performed on each element of the array – mapping – are independent of each other. This means that we can split the array into chunks and process each chunk of the array on a different machine – the array is processed in parallel. In many of Google’s applications the output of a mapping is an array of arrays, and the next computation is reducing; again, such computations can be done in parallel. Even more important is the run-time organization of these parallel computations, for example dealing with machine failure (or fault tolerance). Google has set up a general map-reduce function in a special library, along with dedicated capabilities for parallel processing. Users can supply their own functions to feed into map and reduce, and the run-time organization is handled by the computers on behalf of the user.

In summary, early programming languages could only pass very simple data as arguments to functions. Computer Scientists developed Functional Programming in which functions could be passed as arguments in very clean and simple ways, and they developed standard ways of calling functions such as mapping and reducing. These techniques have then enabled an international corporation to process and analyse large amounts of data in a variety of ways, but with each way following the map-reduce programming model which lends itself to very large scale parallel processing.