Activity 5

Winning Games: the perfect Tic-tac-toe player

Age group successfully used with:	**modified from a version given to** 10 – 11
Abilities assumed:	general problem solving
Time:	**50** minutes upwards
Size of group:	4-30

Focus

What is a program?
How can computers beat humans at games?

Summary

The group work in teams to create sets of instructions (a "program") to play the game of noughts and crosses / tac-tac-toe. This is followed by a tournament between the different teams' programs to see which plays best.

Aims

This activity introduces programming and explores how a computer is able to win at board games like chess. The emphasis is on programming being about solving the problem rather than about being able to write in a programming language.

Technical Terms

Program, artificial intelligence, game tree.

Materials

- Move card sets: 9 for each team
- Instruction Board for player 1: 1 for each team
- Instruction Board for player 2: 1 for each team
- Set of random move cards: 1 for each team
- paper, pencils
- chess computer (optional – just to wave around!)
- giant noughts and crosses board (optional) - alternatively can be done on a white board

What to do

Preparation

Copy enough sets of materials for the class and cut up the "Move cards" into piles. Each group will need 9 sets of 10 cards.

The Grab
One grab is to start with the "Intelligent piece of paper" activity. It both acts as a grab and gives an idea of what the aim of the main task is. Show the class a chess computer. Explain that humans are no longer the best game players on the planet. Machines are!

Explain the following:

In the 1950s it was suggested that one day a computer would beat the best humans at Chess. Chess players ridiculed the idea. They said it took real intelligence to play chess, something a machine could never have. In 1997 Deep Blue, a chess computer beat Garry Kasparov, the world chess champion, in a tournament rules match. Was Deep Blue the more intelligent? It was just following a program: following instructions written by computer programmers.

How did it do it? In this activity we will explore how computers can win at games by creating a program to play Noughts and Crosses (Tic-tac-toe).

Set up
Split the class into groups of 4-6 students. Each group is challenged to come up with a program (a set of instructions) that plays the game of Noughts and Crosses perfectly. A person who knows nothing about the game should never lose if they follow the group's instructions. If both players play perfectly then the game should end in a draw, so the minimum aim is that the program should never lose. It should be able to win, however if the opponent makes a mistake.

How? Each team creates a sequence of instructions to be followed for each of the 9 moves of a full game. This means programming 5 sequences of instructions for player 1 and 4 sequences of instructions for player 2.

The instructions that can be used are restricted to those on the instruction cards provided. The instructions that are available for each move are as follows:

- **Go for 3 in a line**
- **Block 3 in a line**
- **Go in the opposite corner to my last corner move**
- **Go in the opposite corner to the other player's last corner move**
- **Go in a free space**
- **Go in a free corner**
- **Go in the centre**
- **Go in an edge square (not a corner)**
- **If the other player holds opposite corners then go in an edge (not corner)**
- **Go in a corner on the same side as my last corner move**

The instructions must be put into a pile giving the order in which to try them for each move. Once they have decided the order, the teams write a number on each card to indicate its position in the pile for that move. Then they are put into a pile on the move board. Ultimately each teach should have a pile of one or more cards on each position of their move boards.

Example
The simplest program that plays successfully (if poorly) would be to have a single instruction for every move:

1. Go in a free space

As it does not say which space to go in, the space is chosen randomly from those available (see how below).

Each move's program can be made of more than one instruction that are tried in the order given until one is possible. For example a possible list for a move is:

1. Go in the centre
2. Go in a free corner
3. Go in a free space

That means that the first instruction to be tried for this move is to go in the centre of the board. However, if that is impossible (e.g. because the other player has already gone there) try the second instruction in the list instead: attempt to go in a free corner. As this instruction does not say which corner to go in and several may be free, the specific one would again be chosen randomly (see below). If all the corners are used up too, the third instruction is followed instead and you go in a free space.

A list of instructions of this form is needed for each move. For example, a simple (and attacking) program for playing first would be:

```
PLAYER 1, MOVE 1:
    1 Go in the centre
    2 Go in a free space
PLAYER 1, MOVE 2:
    1 Go in a free corner
    2 Go in a free space
PLAYER 1, MOVE 3:
    1 Go for 3 in a line
    2 Go in a free space
PLAYER 1, MOVE 4:
    1 Go for 3 in a line
    2 Go in a free space
PLAYER 1, MOVE 5:
    1 Go in a free space
```

The teams should be able to create a better player than this though. It can be beaten! They also need to come up with the moves for player 2.

The instructions "Go for three in a line" and "Block 3 in a line" are obviously very important. They refer to the situations where either you or your opponent already has two pieces in a line with the other square free so a win is possible. They may not help if you have already been forked though!

What if none of the moves apply?
If you get to the end of the list of instructions to try for a move, with none of the instructions being possible then the player resigns. HINT: It is always a good idea to include "Go in a free space" as the last instruction of every list. It ensures you can

always make some legal move.

If you prefer the rules to be less harsh then you can make playing randomly the default move if nothing else is possible.

Random Moves
If an instruction does apply but there are several possible squares it could refer to (such as "Go in a free corner" when all the corners are free), then one of them is chosen randomly. An easy way to do this is to take the number cards provided that correspond to the squares concerned. Shuffle them, place them face down and allow the player to take one at random. The number on the chosen card is the place to move. Use the numbered board provided as a key to where the number chosen requires you to go.

Creating programs
Give the teams the instruction cards and paper and pencils. Their first task is to work out a strategy to play. What is the best first move to play for example? How can you stop the other player winning next move? Get them to play a rapid series of games in pairs to get an idea of how best to play, noting what seems the best thing to do.

Next they should try and turn their ideas into instructions, picking those cards they think are useful for each move and making a pile for that move in its slot on the instruction boards. Suggest at this point they split in two, with some in charge of creating the instructions for player 1 and the others in charge of player 2.

Once they have a program they think works they should use the position box on the cards to indicate the order the instructions should be tried in for that move.

Testing their program
They should next try the program they have created out to see how good it is. One person in the team should play against it. The instructions should be followed exactly. For each move of the program, first take the top card from the pile for that move. If that move is possible, make it. Otherwise discard it and try the next card's instruction.

Emphasise that if the instructions give a free choice, the move **MUST** be made randomly, using the random number cards.

At the end of the move, put the cards back in their original order.

They may wish to then tweak the instructions to improve them.

Tournament
As the last part of the session hold a tournament between the different teams' programs. This is most fun if matches are done one at a time around a large board either on the floor or on a whiteboard. Rather than the team following their own program a neutral person should take them and "execute" them impartially, doing exactly what the piles of instructions say. A referee (you) should ensure fair play.

The format used will depend on the numbers involved. The easiest is to use a round robin format where in each round different pairs of programs play against each other.

Each match should consist of each program playing for both player 1 and player 2. Give 2 points for a win, 1 for a draw and -2 for a loss. Remember the aim is first of all not to lose!

Repeat this over a series of round with different teams playing each other. The champion being the one amassing the most points (or if time you could finish with a knock-out stage).

Round up
Summarise what they have done. They have each created a program (lists of instructions) for playing noughts and crosses. Anyone (or anything) that can follow the instructions can do so and play well (if the instructions are good). That is all a computer does: follow the instructions in its programs. In fact every computerised gadget from a mobile phone to a washing machine is doing just that. They follow the instructions that programmers write. As we have seen, programming is very creative. You have to come up with instructions that solve the problem whatever happens.

Creating good instructions for chess is what the programmers of Deep Blue, the best chess player on the planet, did so successfully!

Is all the intelligence just in the programmers though? Bear in mind that Deep Blue can play chess better than the people who programmed it. They would not have beaten Kasparov. Deep Blue on the other hand could because it can follow instructions incredibly quickly and just as importantly accurately. In fact a chess computers instructions don't tell it what moves to play directly, they tell it how to search for the best moves by playing ahead in the game and checking the results...but that is a future activity.

Variations and Extensions

For a longer session, after the first tournament you could introduce the idea of game trees – exhaustively exploring all the possible moves. The teams should try playing the game in a methodological way and attempt to create a game tree for playing perfectly. For example, first explore the possible games if the first player goes in the centre. Then explore the games if the first move is in the corner. Once done they should create a new set of instructions to play that way.

There is an online version of this activity as a Java applet on the cs4fn website. See http://www.cs4fn.org/programming/noughtscrosses/ This can be used to allow a class to do essentially this activity online just by dragging and dropping instructions. A difference there is that a single list of instructions has to be devised that works for every move.

If you wish to show a good set of rules at the end to compare the class's with, the following play well. Make the tournament the right to play against these "The current World Champion" rules.

Player 1

MOVE 1:

1 Go in a free corner

```
        2 Go in the middle
    MOVE 2:
        1 Go in the opposite corner to my last corner move
        2 Go in a free corner
    MOVE 3:
        1 Go for 3 in a line
        2 Block 3 in a line
        3 Go in a free corner
        4 Go in a free space
    MOVE 4:
        1 Go for 3 in a line
        2 Block 3 in a line
        3 Go in a free corner
        4 Go in a free space
    MOVE 5:
        1 Go in a free space

Player 2
    MOVE 1:
        1 Go in the centre
        2 Go in a free corner
    MOVE 2:
        1 Block 3 in a line
        2 If the other player holds opposite corners then go in an
        edge (not corner)
        3 Go in the opposite corner to the other player's last corner
        move
        4 Go in a free corner
    MOVE 3:
        1 Go for 3 in a line
        2 Block 3 in a line
        3 Go in a free corner
        4 Go in a free space
    MOVE 4:
        1 Go for 3 in a line
        2 Block 3 in a line
        3 Go in a free corner
        4 Go in a free space
```

It is possible to play even better than these rules (though not by much) – think about maximising the opportunities for the other player to make a mistake.

Further Reading

How do computers get to be so smart?
Read about the rules that will make you a perfect Noughts and Crosses player
http://www.cs4fn.org/algorithms/noughtscrosses.html

Write your own perfect player
Write you own program and pit it against our perfect player online.
http://www.cs4fn.org/programming/noughtscrosses/

Created by Paul Curzon and Peter McOwan, Queen Mary University of London
with support from EPSRC and Google

Winning at Nim: computers outwitting humans
One of the earliest games the computers outwitted us at was Nim. Follow the links and you can play our online Hamster Nim too.
http://www.cs4fn.org/binary/nim/nim.php

Go, Go gadget: computers' biggest gaming challenge
Computers are the best game players on teh planet...or are they. Humans can still beat them at the Game of Go
http://www.cs4fn.org/ai/gogogadget.php

Links to other activities

The intelligent piece of paper

Take part in a test of intelligence against an intelligent piece of paper!
This is a good introduction to what a computer program is, and also to start a discussion on what it would mean for a computer to be intelligent.

Sweets learning computers (forthcoming)

Make a computer that teaches itself to play a game perfectly.
A computer can be programmed to work the rules of how to play the game out for itself (so where is the intelligence then?) This activity is one way to illustrate how that can be done.

Spit-not-so (forthcoming)

Play a word game with a surprising twist and learn about GUIs.
Computer Scientists need to be able to do lateral thinking – especially when it comes to solving problems. A good and common computer science way is to find an equivalent problem that has already been solved. This activity demonstrates this whilst also having a message about why Graphical User Interfaces (GUIs) are easier to use than console interfaces.

Number cards for random moves

1	2
3	4
5	6
7	8
9	

Ensure these are printed onto card thick enough that the number is not visible from the back. Use the following numbered board as the key.

Created by Paul Curzon and Peter McOwan, Queen Mary University of London with support from EPSRC and Google

Board positions for randomly chosen numbers

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**

Created by Paul Curzon and Peter McOwan, Queen Mary University of London
with support from EPSRC and Google

Move Cards: Player 1, Move 1

Player 1, Move 1 **Go for 3 in a line** Position	**Player 1, Move 1** **Block 3 in a line** Position
Player 1, Move 1 **Go in the opposite corner to my last corner move** Position	**Player 1, Move 1** **Go in the opposite corner to the other player's last corner move** Position
Player 1, Move 1 **Go in a free space** Position	**Player 1, Move 1** **Go in a free corner** Position
Player 1, Move 1 **Go in the centre** Position	**Player 1, Move 1** **Go in an edge square (not a corner)** Position
Player 1, Move 1 **If the other player holds opposite corners then go in an edge (not a corner)** Position	**Player 1, Move 1** **Go in a corner on the same side as my last corner move** Position

Move Cards: Player 1, Move 2

Player 1, Move 2 **Go for 3 in a line** Position	**Player 1, Move 2** **Block 3 in a line** Position
Player 1, Move 2 **Go in the opposite corner to my last corner move** Position	**Player 1, Move 2** **Go in the opposite corner to the other player's last corner move** Position
Player 1, Move 2 **Go in a free space** Position	**Player 1, Move 2** **Go in a free corner** Position
Player 1, Move 2 **Go in the centre** Position	**Player 1, Move 2** **Go in an edge square (not a corner)** Position
Player 1, Move 2 **If the other player holds opposite corners then go in an edge (not a corner)** Position	**Player 1, Move 2** **Go in a corner on the same side as my last corner move** Position

Created by Paul Curzon and Peter McOwan, Queen Mary University of London
with support from EPSRC and Google

Move Cards: Player 1, Move 3

Player 1, Move 3 **Go for 3 in a line** Position	**Player 1, Move 3** **Block 3 in a line** Position
Player 1, Move 3 **Go in the opposite corner to my last corner move** Position	**Player 1, Move 3** **Go in the opposite corner to the other player's last corner move** Position
Player 1, Move 3 **Go in a free space** Position	**Player 1, Move 3** **Go in a free corner** Position
Player 1, Move 3 **Go in the centre** Position	**Player 1, Move 3** **Go in an edge square (not a corner)** Position
Player 1, Move 3 **If the other player holds opposite corners then go in an edge (not a corner)** Position	**Player 1, Move 3** **Go in a corner on the same side as my last corner move** Position

Created by Paul Curzon and Peter McOwan, Queen Mary University of London
with support from EPSRC and Google

Move Cards: Player 1, Move 4

Player 1, Move 4 **Go for 3 in a line** Position	**Player 1, Move 4** **Block 3 in a line** Position
Player 1, Move 4 **Go in the opposite corner to my last corner move** Position	**Player 1, Move 4** **Go in the opposite corner to the other player's last corner move** Position
Player 1, Move 4 **Go in a free space** Position	**Player 1, Move 4** **Go in a free corner** Position
Player 1, Move 4 **Go in the centre** Position	**Player 1, Move 4** **Go in an edge square (not a corner)** Position
Player 1, Move 4 **If the other player holds opposite corners then go in an edge (not a corner)** Position	**Player 1, Move 4** **Go in a corner on the same side as my last corner move** Position

Move Cards: Player 1, Move 5

Player 1, Move 5 **Go for 3 in a line** Position	**Player 1, Move 5** **Block 3 in a line** Position
Player 1, Move 5 **Go in the opposite corner to my last corner move** Position	**Player 1, Move 5** **Go in the opposite corner to the other player's last corner move** Position
Player 1, Move 5 **Go in a free space** Position	**Player 1, Move 5** **Go in a free corner** Position
Player 1, Move 5 **Go in the centre** Position	**Player 1, Move 5** **Go in an edge square (not a corner)** Position
Player 1, Move 5 **If the other player holds opposite corners then go in an edge (not a corner)** Position	**Player 1, Move 5** **Go in a corner on the same side as my last corner move** Position

Created by Paul Curzon and Peter McOwan, Queen Mary University of London
with support from EPSRC and Google

Move Cards: Player 2, Move 1

Player 2, Move 1 **Go for 3 in a line** Position	**Player 2, Move 1** **Block 3 in a line** Position
Player 2, Move 1 **Go in the opposite corner to my last corner move** Position	**Player 2, Move 1** **Go in the opposite corner to the other player's last corner move** Position
Player 2, Move 1 **Go in a free space** Position	**Player 2, Move 1** **Go in a free corner** Position
Player 2, Move 1 **Go in the centre** Position	**Player 2, Move 1** **Go in an edge square (not a corner)** Position
Player 2, Move 1 **If the other player holds opposite corners then go in an edge (not a corner)** Position	**Player 2, Move 1** **Go in a corner on the same side as my last corner move** Position

Move Cards: Player 2, Move 2

Player 2, Move 2 **Go for 3 in a line** Position	**Player 2, Move 2** **Block 3 in a line** Position
Player 2, Move 2 **Go in the opposite corner to my last corner move** Position	**Player 2, Move 2** **Go in the opposite corner to the other player's last corner move** Position
Player 2, Move 2 **Go in a free space** Position	**Player 2, Move 2** **Go in a free corner** Position
Player 2, Move 2 **Go in the centre** Position	**Player 2, Move 2** **Go in an edge square (not a corner)** Position
Player 2, Move 2 **If the other player holds opposite corners then go in an edge (not a corner)** Position	**Player 2, Move 2** **Go in a corner on the same side as my last corner move** Position

Created by Paul Curzon and Peter McOwan, Queen Mary University of London
with support from EPSRC and Google

Move Cards: Player 2, Move 3

Player 2, Move 3 **Go for 3 in a line** Position	**Player 2, Move 3** **Block 3 in a line** Position
Player 2, Move 3 **Go in the opposite corner to my last corner move** Position	**Player 2, Move 3** **Go in the opposite corner to the other player's last corner move** Position
Player 2, Move 3 **Go in a free space** Position	**Player 2, Move 3** **Go in a free corner** Position
Player 2, Move 3 **Go in the centre** Position	**Player 2, Move 3** **Go in an edge square (not a corner)** Position
Player 2, Move 3 **If the other player holds opposite corners then go in an edge (not a corner)** Position	**Player 2, Move 3** **Go in a corner on the same side as my last corner move** Position

Created by Paul Curzon and Peter McOwan, Queen Mary University of London
with support from EPSRC and Google

Move Cards: Player 2, Move 4

Player 2, Move 4 **Go for 3 in a line** Position	**Player 2, Move 4** **Block 3 in a line** Position
Player 2, Move 4 **Go in the opposite corner to my last corner move** Position	**Player 2, Move 4** **Go in the opposite corner to the other player's last corner move** Position
Player 2, Move 4 **Go in a free space** Position	**Player 2, Move 4** **Go in a free corner** Position
Player 2, Move 4 **Go in the centre** Position	**Player 2, Move 4** **Go in an edge square (not a corner)** Position
Player 2, Move 4 **If the other player holds opposite corners then go in an edge (not a corner)** Position	**Player 2, Move 4** **Go in a corner on the same side as my last corner move** Position

Created by Paul Curzon and Peter McOwan, Queen Mary University of London
with support from EPSRC and Google

Instruction Board, Player 1

PLAYER 1, MOVE 1
PLAYER 1, MOVE 2
PLAYER 1, MOVE 3
PLAYER 1, MOVE 4
PLAYER 1, MOVE 5

Created by Paul Curzon and Peter McOwan, Queen Mary University of London
with support from EPSRC and Google

Instruction Board, Player 2

PLAYER 2, MOVE 1
PLAYER 2, MOVE 2
PLAYER 2, MOVE 3
PLAYER 2, MOVE 4