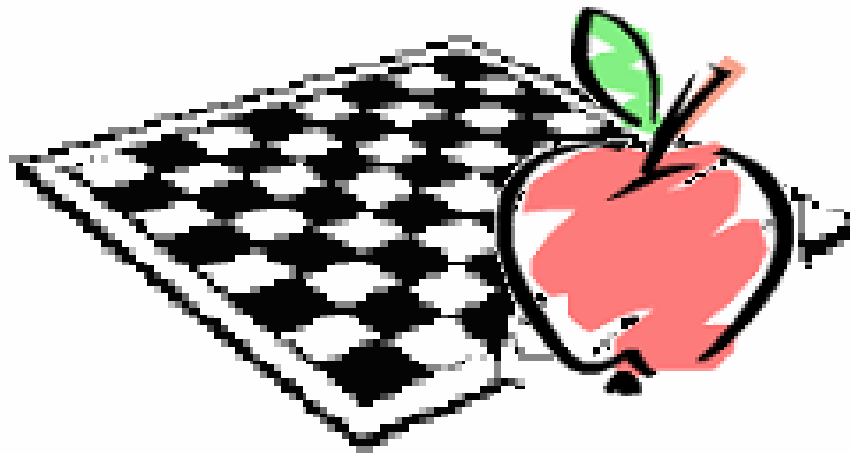


Project Report for Apple Chess



OBJECTIVE

The objective of this project is to create a program to create a chessboard game, namely, Apple Chess. The instructions are as follow.




- Two players will participate in one game of apple chess.
- Players will have the freedom to choose the size of the chessboard.
- Players can either use the default symbol as their disc or select their own disc.
- Follow the rule of apple chess.
- Players can have to chance to surrender to other player at any time.

ANALYSATION OF APPLE CHESS


Apple Chess, also known as, Othello is a registered trademark of Tsukuda Original, licensed by Anjar Co., copyright 1973, 1990 Pressman Toy Corporation.

At the beginning, I don't really know what is apple chess. But after I saw the chessboard, memory starts to come back to me. Now lets take a look at the rules of apple chess.

 Below is a direct transcription (including diagrams) of the rules included in the Pressman Toy Corporation version of the game sold in North America.

Object of the Game

The object of the game is to have the majority of your color discs on the board at the end of the game

 A Minute To Learn

Each player takes 32 discs and chooses one color to use throughout the game.

Black places two black discs and White places two white discs as shown in Figure 1. The game always begins with this setup.

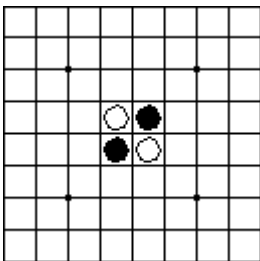


Figure 1

A move consists of "outflanking" your opponent's disc(s), then flipping the outflanked disc(s) to your color.

To outflank means to place a disc on the board so that your opponent's row (or rows) of disc(s) is bordered at each end by a disc of your color. (A "row" may be made up of one or more discs).

Here's one example: White disc A was already in place on the board. The placement of white disc B outflanks the row of three black discs.



White flips the outflanked discs and now the row looks like this:



Othello Rules

1. Black always moves first.
2. If on your turn you cannot outflank and flip at least one opposing disc, your turn is forfeited and your opponent moves again. However, if a move is available to you, you may not forfeit your turn.
3. A disc may outflank any number of discs in one or more rows in any number of directions at the same time - horizontally, vertically or diagonally. (A row is defined as one or more discs in a continuous straight line). See Figures 2 and 3.

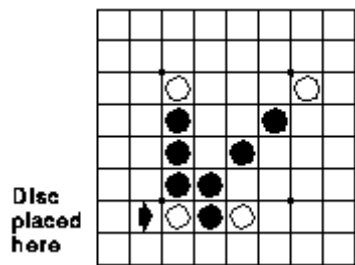


Figure 2

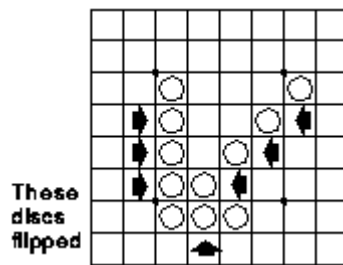


Figure 3

4. You may not skip over your own color disc to outflank an opposing disc. (See Figure 4)

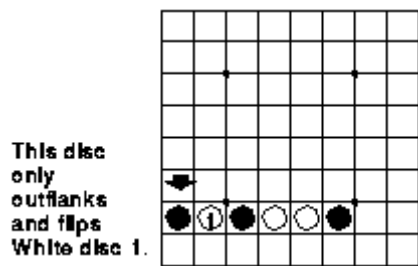


Figure 4

5. Discs may only be outflanked as a direct result of a move and must fall in the direct line of the disc placed down. (See Figure 5 and 6)

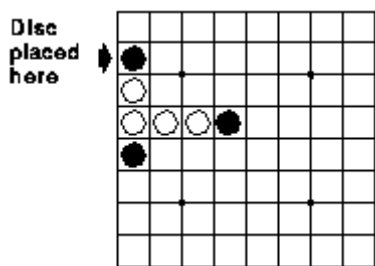


Figure 5

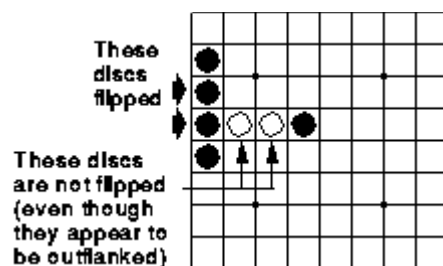


Figure 6

6. All discs outflanked in any one move must be flipped, even if it is to the player's advantage not to flip them at all.
7. A player who flips a disc, which should not have been turned, may correct the mistake as long as the opponent has not made a subsequent move. If the opponent has already moved, it is too late to change and the disc(s) remain as is.
8. Once a disc is placed on a square, it can never be moved to another square later in the game.
9. If a player runs out of discs, but still has an opportunity to outflank an opposing disc on his or her turn, the opponent must give the player a disc to use. (This can happen as many times as the player needs and can use a disc).
10. When it is no longer possible for either player to move, the game is over. Discs are counted and the player with the majority of his or her color discs on the board is the winner.

NOTE: It is possible for a game to end before all 64 squares are filled.

As I mentioned above, the rules are easy to understand, but transforming these rule into any program or logics is a challenging experience.

THE DESIGN OF APPLE CHESS

After knowing the rules of apple chess, here comes the time to design. Seeing all these squares, reminding me of Microsoft Excel. Talking about Excel, the most reminding things are their famous cells, rows and columns. Before we get to the point of choosing the software, programming language and platform of the project, lets take a look how we are going to design our apple chess.



Before the game starts, players will have the freedom to choose the size of chessboard. Classical apple chess chessboard is 8 by 8 containing maximum 64 squares. Why can't we use 9 by 9 like "369", or 3 by 3 like "Tic-Tac-Toe". After taking a good look at the classical apple chess chessboard, the answer is clear. We must have an "even number by even number". Since we start the game having 4 discs in the middle. Having an "odd number by odd number" chessboard will make the game uneven and unfair. So I design to provide the players with the choice of 6 by 6, 8 by 8 and 10 by 10 chessboard.



Then, we need to know the players' name. The program should prompt the users to find out their name. In order to make the game a little bit more different, the program will take the first character of player's name as the disc. With every move made by any user, scores will be kept visible to both players and passing move will be announced automatically.

The screenshot displays the Apple Chess application interface. At the top, there are two Microsoft Excel dialog boxes for entering player names. The first dialog box is titled "Player 1, please enter your name." and has a text input field containing "Me" and buttons for "確定" (OK) and "取消" (Cancel). The second dialog box is titled "Player 2, please enter your name." and has a text input field containing "Teacher" and buttons for "確定" (OK) and "取消" (Cancel). Below these dialog boxes is an 8x8 chessboard with columns labeled A through M and rows labeled 1 through 8. The chessboard shows a game in progress with pieces 'M' and 'T' placed on the board. A small "Apple Chess" dialog box is open in the center, displaying "Me starts first" and a "確定" (OK) button. To the right of the chessboard is a control panel with a "Current Turn =>" label and a text input field containing "M". Below this are two text input fields for scores: "Me" with a value of "2" and "Teacher" with a value of "2". At the bottom of the control panel are three buttons: "Give up", "Reset", and "Quit".




As stated in the objective, players can have to chance to surrender to other player at any time, restart the game at any time and, of course, quit the game at any time.

IMPLEMENTATION

Selecting the suitable programming language

During the planning and designing stage of apple chess, I have made the following considerations.

- Can everyone use the program without installing any software?
- Can everyone use the program without installing any OCX or DLL files?
- Interface design and navigation within the software


Possible programming language: 

- Delphi
- Power Builder
- Visual Basic
- Visual C
- Microsoft Excel

Choosing suitable programming language will make the implementation process more efficient. When considering the programming language for apple chess, we must have some language that can suit the need of the game, easy to distribute to others and/or people who is going to make the project and an interface that is easy to follow.

Delphi, Power Builder, Visual Basic and Visual C are all Object Oriented programming language and all of them contain the event driven property. This will make the programming progress easier and I can have more time on taking care of the logics of flipping discs. I minor sit back of these software is, we don't know what components (OCXs and DLLs) our user might have on their machine, not to mention those OCXs and DLLs with different versions. Although, an installing program can "pack" all OCXs and DLLs together, this might make apple chess impossible to play.

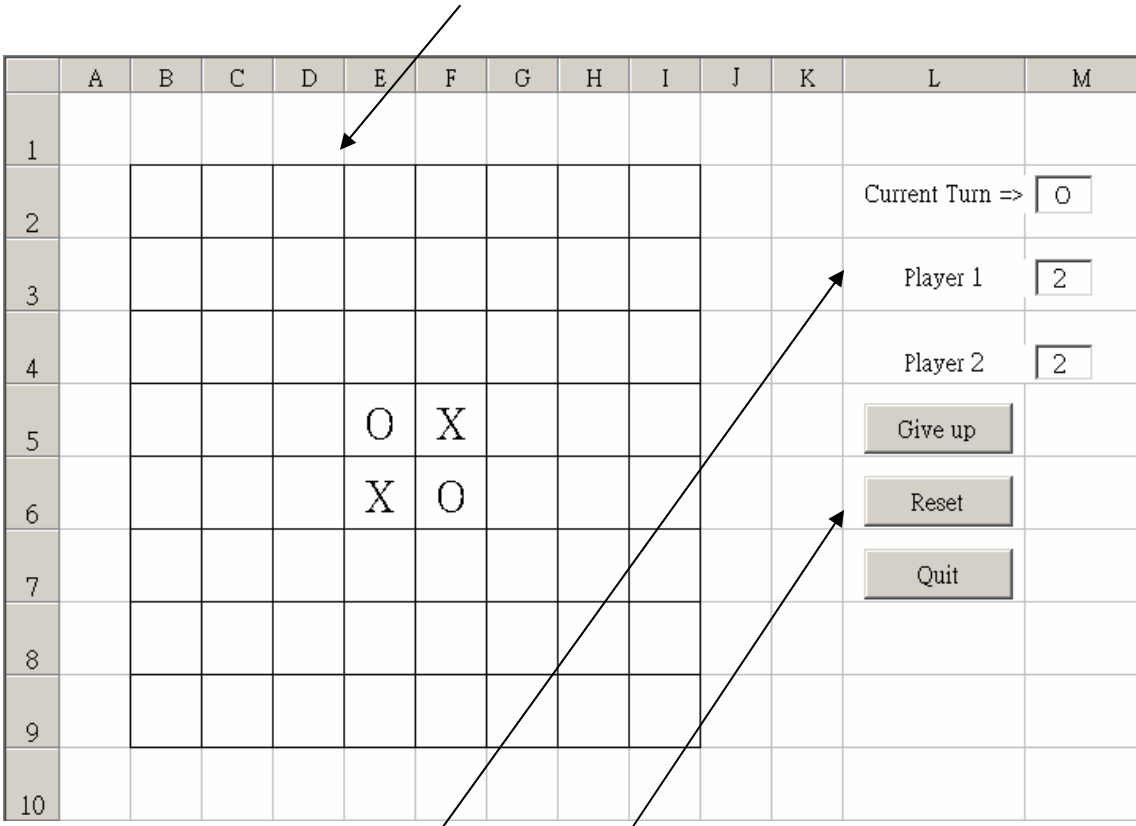


Microsoft Excel can solve this problem. As long as all users are using a same version of Excel, their components are very much the same, if not identical. The cell, row and column property will make the programming process more efficient. While having the benefit of Visual Basic, VBA in Excel can work as powerful as any C++ program. (Sometime too powerful, since we can use VBA to write destructive programs.) The distribution of an Excel file (.xls) is easy; the size is around 100 KB; most computers at school have Microsoft Excel installed. 

Finally, I decide to use Microsoft Excel to do the project. The version I will use is Microsoft Excel 2000 SR-1.

This is the interface of apple chess.

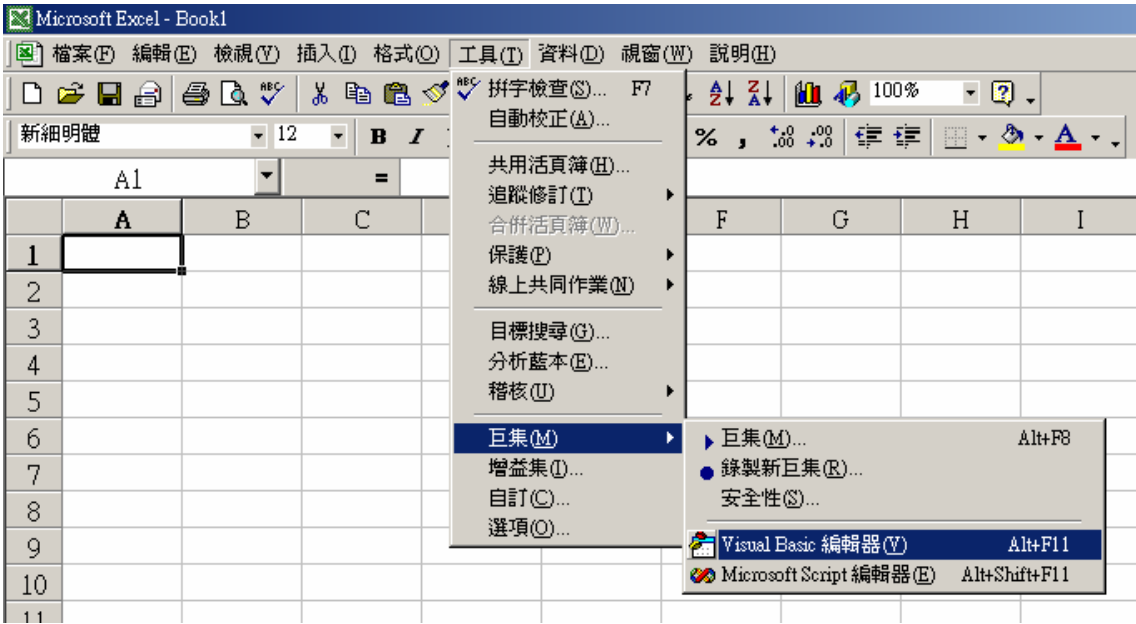
The size of the chessboard will change according to player's choice.



Players will be able to see the disc of current player from here, together with the current score.

These 3 buttons are activate all the time, and controls the start and end of the game.

When coding the program, we must enter the VBA coding through here.



There are several subroutines I would like to explain in my apple chess program.



Initialization of the program

Private Sub init()

This is the subroutine that will run right after the start of the excel file. Players will be asked for the size of the chessboard, players' name. After knowing the size of the chess board, the program will prepare the desired number of cells and set their font size, clears their value and draws the squares. Finally, placing the starting discs in the proper location.

Check the selected cell for any flipping

Private Sub check_line()

This subroutine will take accept several parameters, the current disc, the location of the cell, and most important, the direction of checking. Probably, no one knows what I am talking about. Let me use a figure to explain.

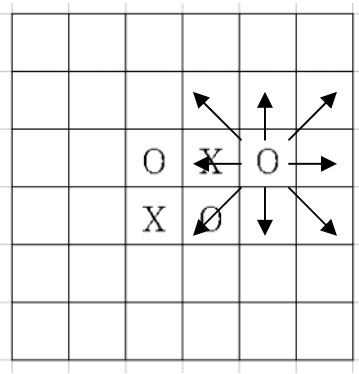
Below is a 6 by 6 chessboard in the beginning stage.

		O	X		
		X	O		

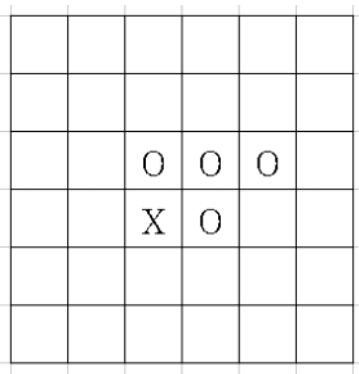
After I click on the cell like this.

		O	X	O	
		X	O		

The program will enter the subroutine and by passing the current disc, the location of the cell, and the direction of the checking. The direction of checking is controlled by 2 for loops and these 2 loops are not located within this subroutine. The 2 for loops can produce 8 directions in total. Again a figure will help.



Knowing the direction of checking is important. With this direction we can find the disc next to the current disc. If the disc is the same, then this is an invalid move. If the disc is the same, then the subroutine will move to the third disc on the same direction and so on. This comparison will stop only when it reaches the end of the line or an invalid move is found. If there are valid moves, we will change the disc according to the number of valid moves. If there are no valid moves, no action is taken. And this is the result of the above figure.



We must cover all 8 directions, before we can really finish the disc flipping process. The subroutine `check_line()`, cannot change direction by itself, it must rely on the 2 for loops. Then we go to the next stage of the program.

Find out the score

Private Sub `check_score()`

This subroutine will go through each and every cell on the chessboard. Using 2 for loops, each disc for each player will bring 1 point and empty cells will have no score.

What if there are no possible moves and the player must pass

Private Sub `check_pass()`

This subroutine will only active when the total number of discs on the chessboard exceeds half the total numbers of cells on the chessboard. The logic of this subroutine is basically the same as subroutine `check_line()` except for 2 features.

The first feature is, instead of just the selected cell, all empty cells on the chessboard will be checked. Since we have to check for all possible moves, we must check everything. For each empty cell, we will follow the same principle subroutine `check_line()`. The second feature is, this subroutine will pass a parameter to subroutine `check_line()` and this parameter will disable the disc flipping action of subroutine `check_line()`. In other words, no discs will be flipped; all we will find out is that if there is any disc can be flipped (possible moves). If there are possible moves, players switch turns and the cycle of this game goes on. However, if there are no more possible moves, the next player's turn is omitted and the turn to play is back to current player.

Testing

There are many things that have to test after the game is implemented.

The **[Give Up]**, **[Reset]** and **[Quit]** buttons beside the chessboard can easily be checked by any players during the game.

Then



- Initialization;
- which player's turn;
- score at any moment during the game; and
- who's the winner

must be checked and also can easily be checked by any players during the game.



However, there are two things that needed to be checked, but they require lots of trials to check. They are

- a game end when one of the players outflanked all the discs of the opponents before the chessboard is filled; and
- a player has no valid possible move and must pass to the opponent

The testing for ending a game when one of the players outflanked all the discs of the opponents before the chessboard is filled was checked with a 6x6 chessboard size and the following moves:

F4 D3 C4 F5 E2 F3 G4 C3 E6 (Coordinates follow Excel's column and row numbers)






<p>Start</p>		<p>Move 1 F4</p>	
<p>Move 2 D3</p>		<p>Move 3 C4</p>	
<p>Move 4 F5</p>		<p>Move 5 E2</p>	
<p>Move 6 F3</p>		<p>Move 7 G4</p>	
<p>Move 8 C3</p>			
<p>Move 9 E6</p>			

In fact, the above screen captures also showed evidence that

- Initialization (Private Sub init())
- which player's turn (Private Sub check_line());
- score at any moment during the game (Private Sub check_score()); and
- who's the winner

are all functioned correctly.

The testing that a player has no valid possible move and must pass to the opponent was checked with a 6x6 chessboard size and the following moves:

 **F4 F5 F6 F3 G3 F7 C6 B7 G4 D3 C4 E6 D6 B4 G2 D7 G7 G5 E7 G6 C7 F2 G3** (Coordinates follow Excel's column and row numbers)

Since there are 23 moves, only the last 2 moves were screen-captured.

Move 22 F2		A	B	C	D	E	F	G	H	I	J	K	L	M
	1													
	2						X	O					Current Turn =>	O
	3				X	X	X	O					Player 1	12
	4		X	X	X	O	O	O					Player 2	15
	5				X	O	X	X					Give up	
	6			X	O	X	X	X					Reset	
	7		X	O	O	O	O	O					Quit	
8														
Move 23 G3		A	B	C	D	E	F	G	H	I	J	K	L	M
	1													
	2						X	O					Current Turn =>	O
	3				X	X	X	O					Player 1	10
	4		X	X	X	O	X	O					Player 2	16
	5				X	O	X	X					Give up	
	6			X	O	X	X	X					Reset	
	7		X	O	O	O	O	O					Quit	
8														

Apple Chess

No possible move for X << PASS >>

確定

So, the game is implemented without bugs.

CONCLUSION AND DISCUSSION

Some words from the student

Doing this project provided me with the chance to find out more about Visual Basic and VBA. Their basic function and operators are the same, but if we use VBA within Microsoft Excel or other Microsoft Office product like Word, PowerPoint and Access, we must have an extensive knowledge towards the special built-in functions.

It really caused me some time to find the way to select an individual cell using two variables. Hard coding the program is easy but selecting a cell using variables is much harder. To make things worse, we cannot use the numeric result of an operation directly in the cell selection statement.

The cause to the first problem is that we don't have enough VBA reference books. The cause to the second problem is that the result numeric result is a long integer, and the cell selection function won't take long integer as input.

We learn from mistakes and gain experience, for situation like this, it is always good to search the web for answers. Time is another element to the project. Make sure to have more time to the programming stage since we might never know what we will encounter.

Discussion

No program is bug free. There are always rooms for improvement. The interface can be better. More colors should be added. There are several improvements I can make to the apple chess program. Adding **AI** or computer opponents to the program can provide players with more fun. But this will require more coding and will increase the time between steps and workloads of the CPU.

My teacher had mentioned to me that I could add background colors to each cell and different players can have different background color. This is a very good idea and can be done. While talking about color, a picture can take the role of a disc and take part in this project. This will increase the fun of the game. Finally, adding some background music is another good idea.

In conclusion, this had been a very good experience.