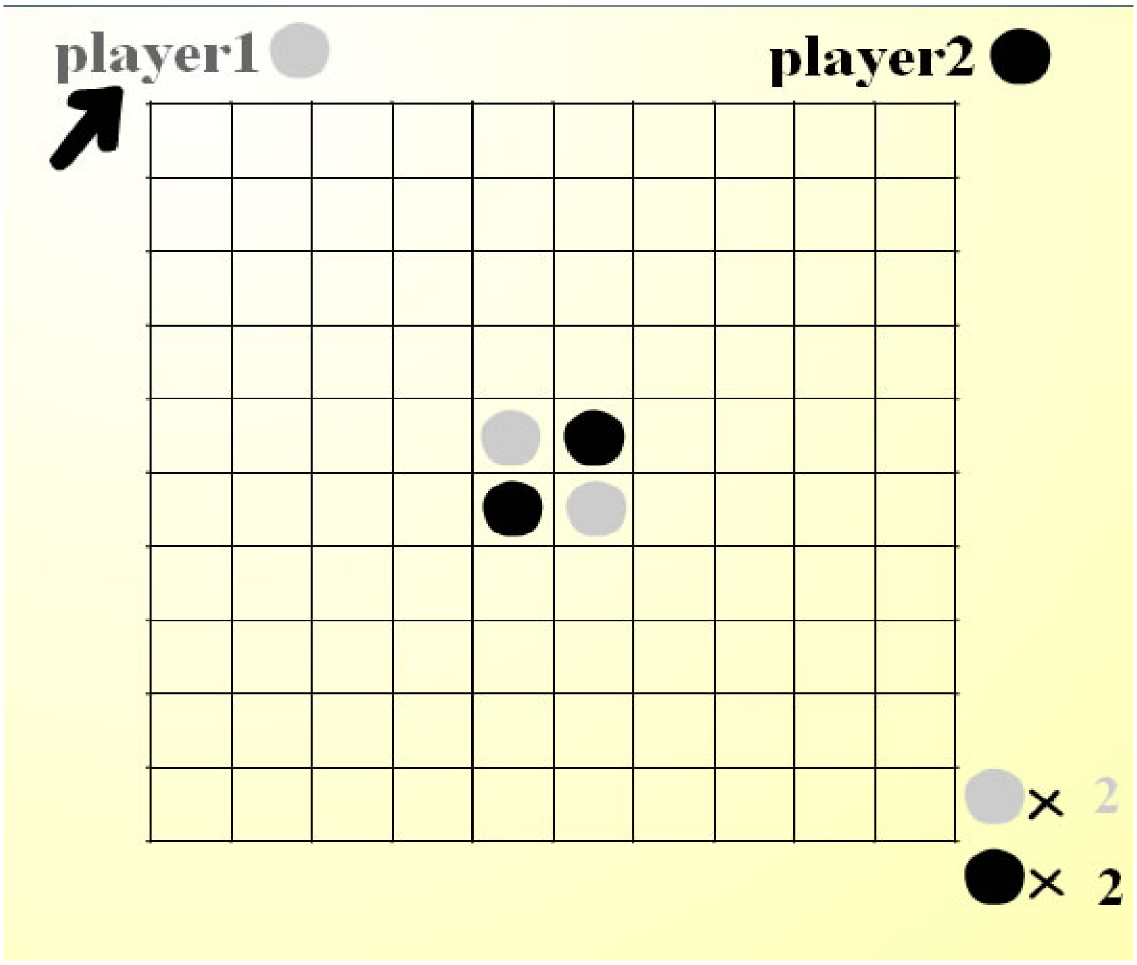
APPLE CHESS GAME



Objective

Write a program of Apple Chess game for two players. The chessboard size is fixed to be 10 by 10. During the game, the players need to follow the rules of the game. At the end of the game, the program will determine who the winner is and display the result.

The game is only to be played on the computer with my family, therefore I don't need to provide artificial intelligence (A.I.) to the program for human player vs. computer. A 6 by 6 chessboard is too simple, while an 8 by 8 is too common, so I will make the chessboard to be challenging as 10 by 10.

Analysis of the Problem

To play the game, the players need to use the mouse to place the chess. However, Turbo Pascal does not allow mouse movement in DOS mode. Therefore, I have to choose a programming language which supports Graphical User Interface. Since this is only a game for leisure, I don't need Microsoft's ® professional programming languages like C or Visual Basic. Instead, I choose Macromedia's ® Flash MX because of the following reasons:

1. it is very popular;
2. the Action Script of Flash is easier to learn;
3. it is good for writing interactive games;
4. even if the player's computer is not installed with Flash MX, he can download the Flash player free of charge from Macromedia and the file size of the Flash player is small.

From what I learn in Computer Studies, my analysis is that I have to use the top-down approach and the Apple Chess game program needs the following modules:

1. A start scene to get the players' name so that the players have a personal feeling when they see their names on the chessboard while playing the game
2. An array storing data which represent the chessboard
3. Initialize the array with 4 chesses at the centre of the chessboard in the beginning
4. Get player's input of position of chess
5. Check whether the player's input of position of chess is valid
6. If the player's move is valid, then turn another player's chesses
7. Re-calculate the number of the players' chesses
8. Check whether the chessboard is full and end the game
9. Before the game ends, check who is the winner or the game is draw

The flowchart of the logic is shown in Figure 1

P5

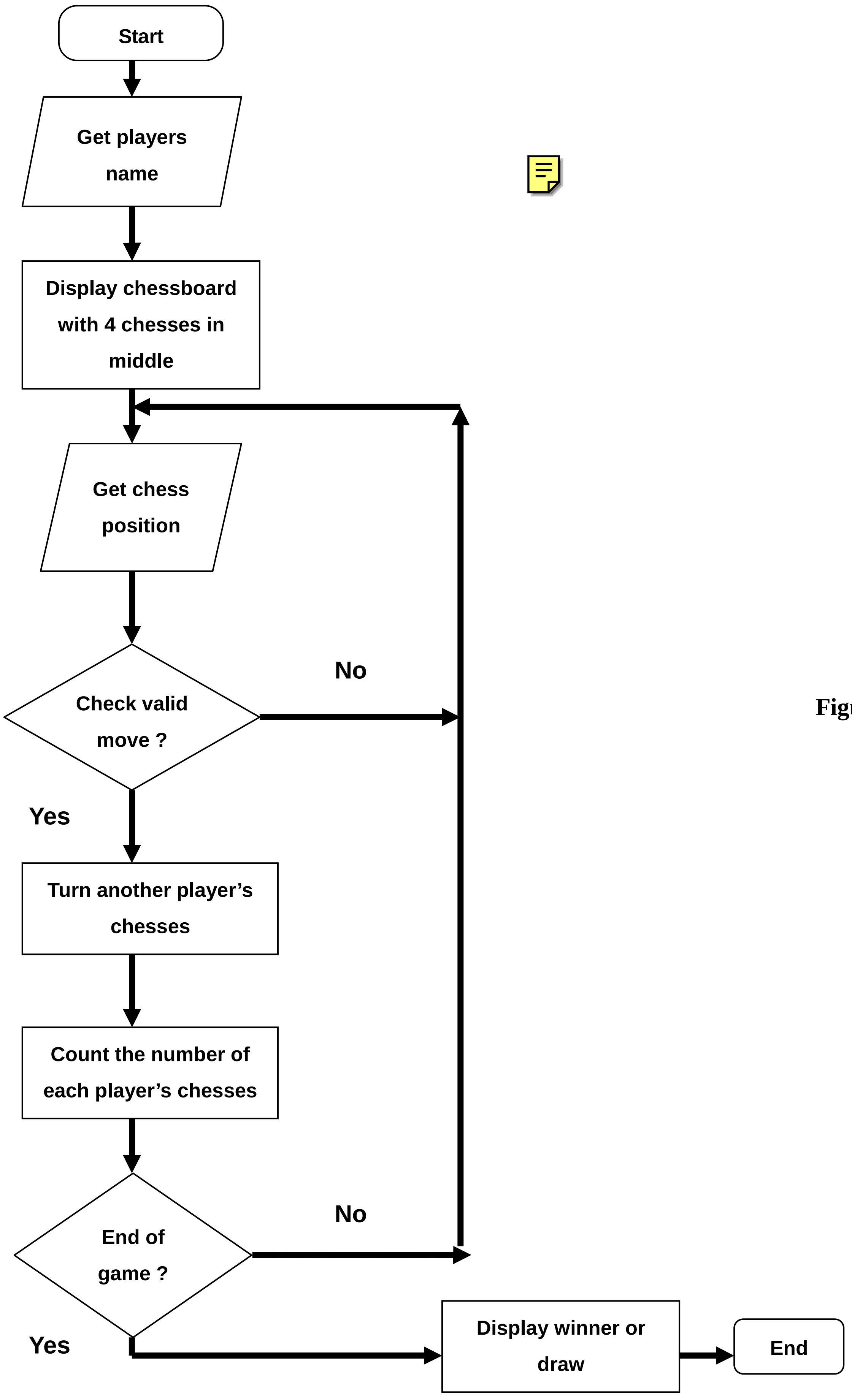
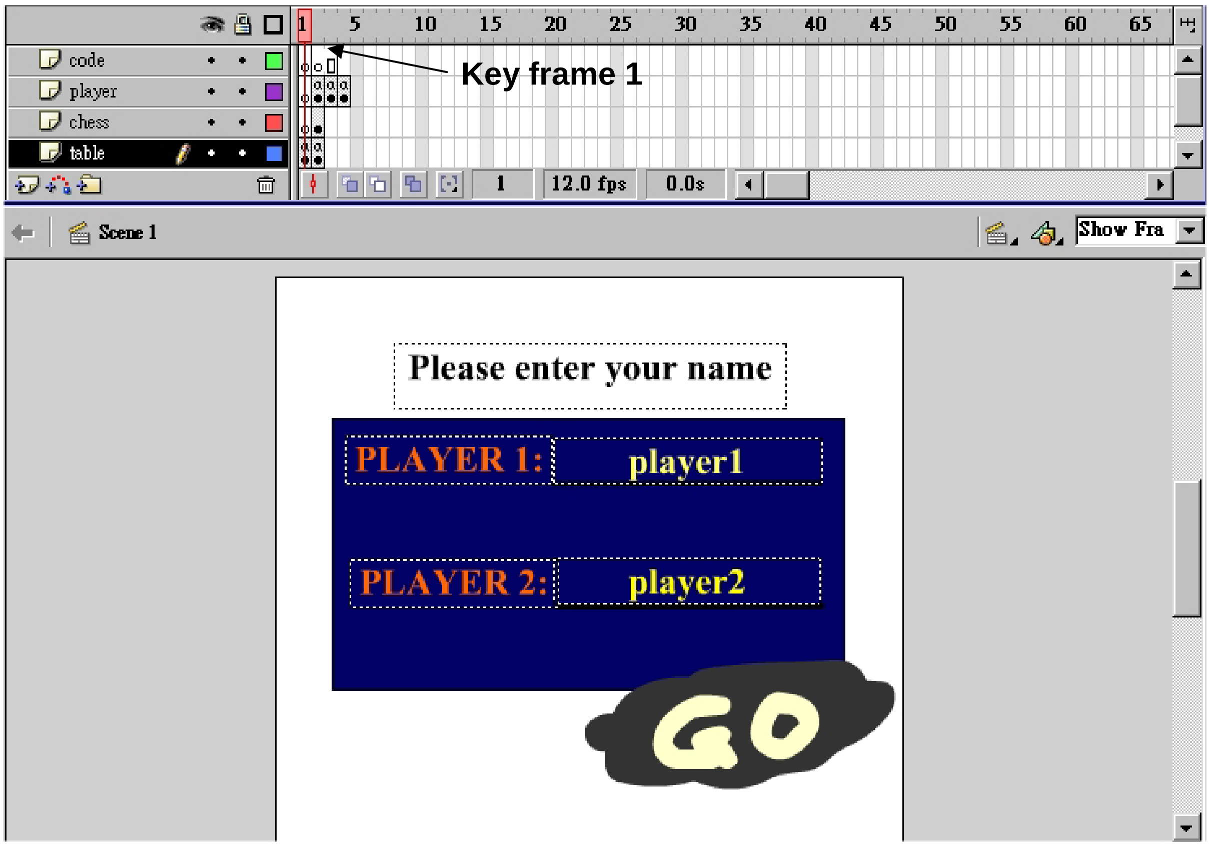

Start
Get players name
Display chessboard with 4 chesses in middle
Get chess position
Check valid move ?
No
Yes
Turn another player's chesses
Count the number of each player's chesses
End of game ?
No
Yes
Display winner or draw
End

Figure 1

Design of the Solution

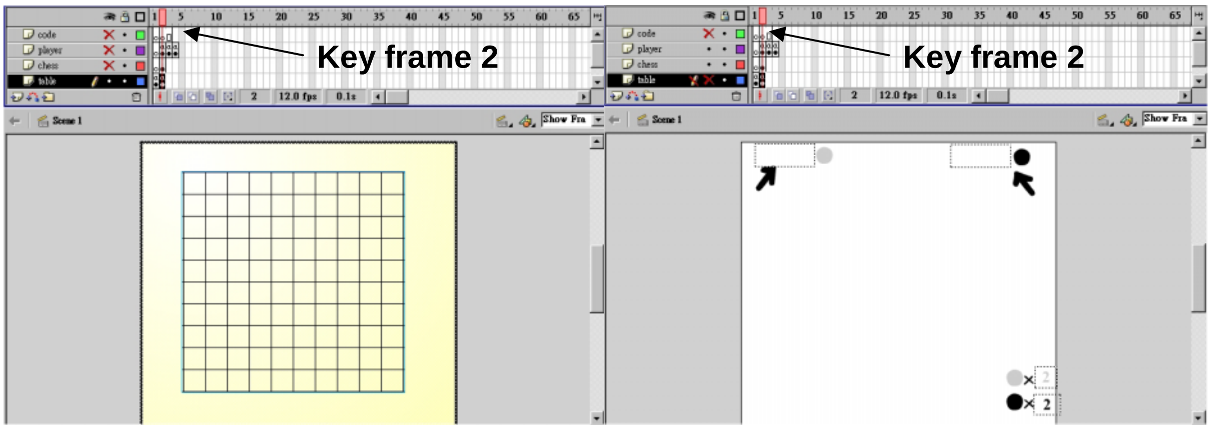
For convenience and simplicity, I used only 4 keyframes to write the game:

Key Frame 1



Key Frame 1 of the "table" layer is the scene where the players input their names, the default names are "player 1" and "player 2"

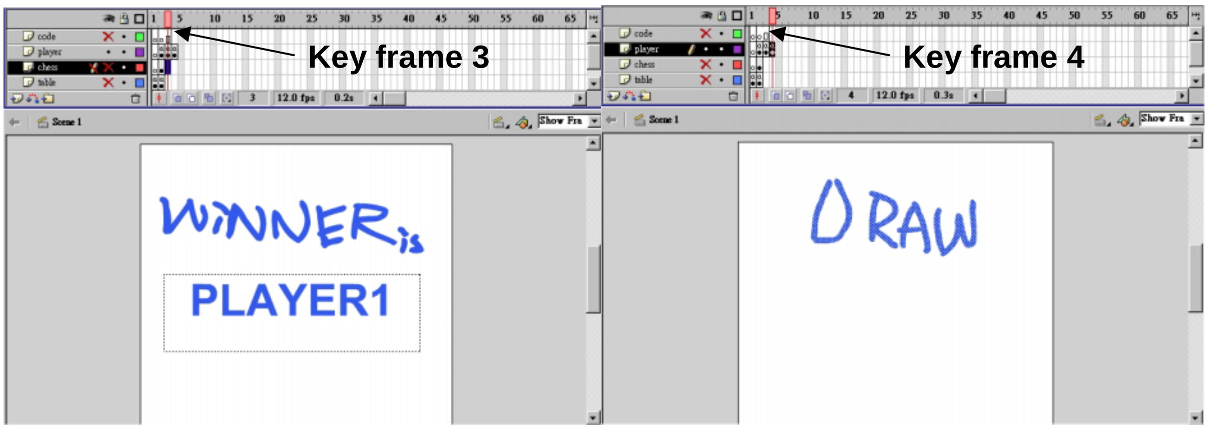
Key Frame 2



Key Frame 2 is the chessboard where the players place their chesses. All Action Scripts are contained in this key frame. The right hand side screen capture depicts the colour of chesses with players' names and the number of chesses of each colour during the game.

P5

Key Frames 3 and 4



Key frames 3 and 4 display the result at the end of the game.

Flash only allows 1-dimensional array, but the chessboard is 2-dimensional. Therefore, I used an array of 100 elements but in my mind I visualized the 1-dimensional 100-elements array as a 10 by 10 2-dimensional array like the following:

01	02	03	04	05	06	07	08	09	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	**45(W)**	**46(B)**	47	48	49	50
51	52	53	54	**55(B)**	**56(W)**	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Table 1

The cells depicted in **Table 1** which corresponded to the 45th, 46th, 55th and 56th elements are used to place the initial 2 white and 2 black chesses.

All the cells are initialized to contain “`nothing`”. Then, before the game is started, the 45th and the 56th elements of the 1-dimensional array are initialized to contain “`white`”, and the 46th and the 55th elements of the 1-dimensional array are initialized to contain “`black`”.

Whether a user placed new chess of a new move is valid, the cells in the North, South, East, West, North-East, South-West, North-West and South-East direction are checked. For example, to check in the North direction:

```
function checkN(a) {
  _global.Nfound = 0;
  _global.Na = 0;
  var N = 20;
  if (square[a-10] == colour || a-10<0 || square[a-10] == "nothing") {
     Nfound == 0;
          }
   else {
     while (a-N>=0 && Nfound == 0 && square[a-N]<>"nothing") {
         if (square[a-N] == colour) {
              Nfound = 1;
              Na = a-N;
          } else {
              N = N+10;
              }
          }
     }
}
```

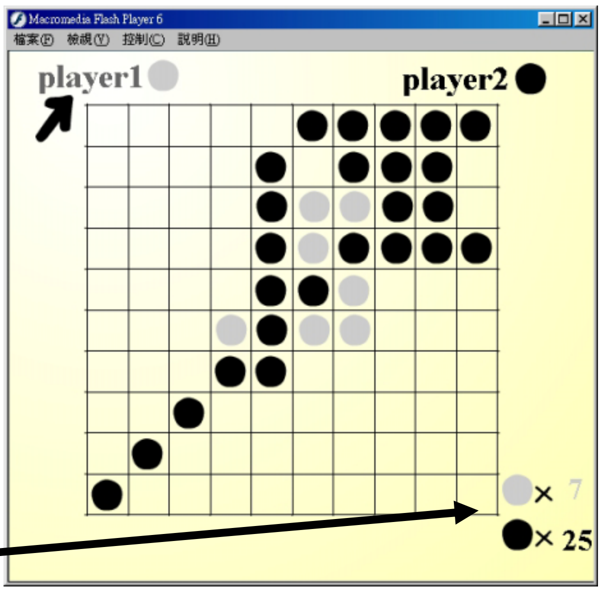
The index of the immediate element of the “upper” row is decreased by 10, hence `(a-10)`. The following table summarizes the index change in different directions:

Directions	Index change
N	`a-10`
S	`a+10`
E	`a+1`
W	`a-1`
NE	`a-9`
SW	`a+9`
NW	`a-11`
SE	`a+11`

If the move is valid, the elements of the opposite player’s chesses will be assigned by the new colour name.

After each valid move and turning the opposite player’s chesses, the number of chesses of each player is re-calculated using the following:

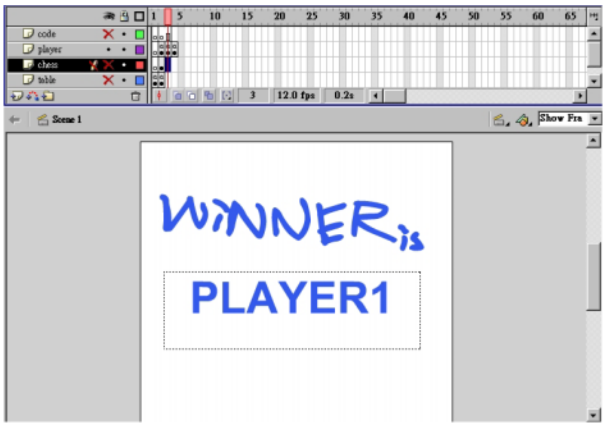
```
function count() {
   sumwhite = 0;
   sumblack = 0;
      for (i=1; i<=100; i++) {
            if (square[i] == "black") {
                sumblack = sumblack+1;
              }
            if (square[i] == "white") {
                sumwhite = sumwhite+1;
              }
         }
   _root.sumb_txt.text = sumblack;
   _root.sumw_txt.text = sumwhite;
}
```



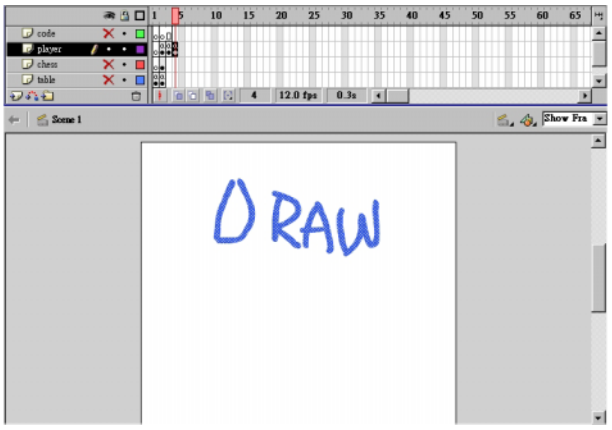
To determine who is the winner or the game is a draw, the following is used:

```
function Win() {
     if (sumwhite>sumblack) {
          winner_txt.text = player1;
          gotoAndStop(3);
          }
     else if (sumblack>sumwhite) {
          winner_txt.text = player2;
          gotoAndStop(3);
          }
     else{
          gotoAndStop(4);
          }
}
```

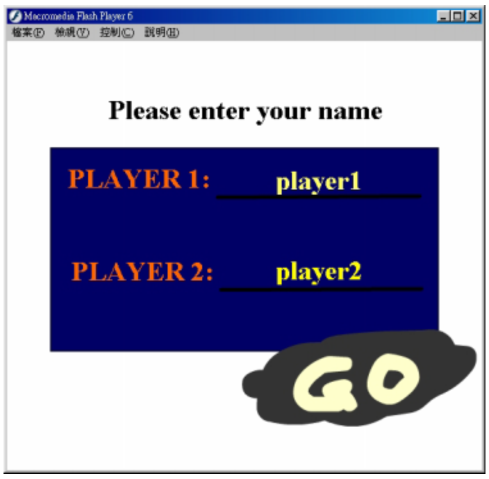
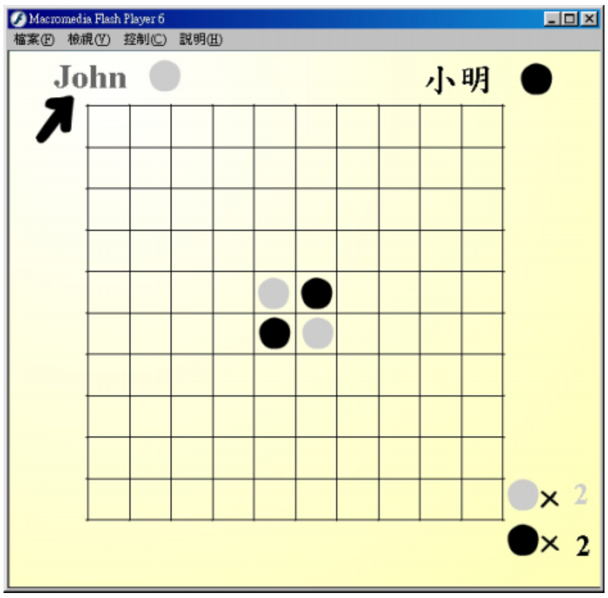
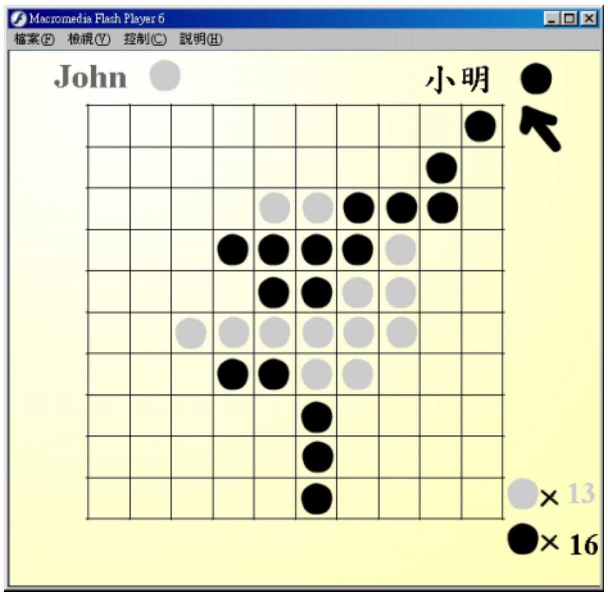
`gotoAndStop(3)` is the Key Frame 3 which will display the player name of the winner.



`gotoAndStop(4)` is the Key Frame 4 which will display the game is a draw.



Implementation

 Macromedia Flash Player 6 檔案(F) 檢視(V) 控制(C) 說明(H) Please enter your name PLAYER 1: player1 PLAYER 2: player2 GO	This is the first scene to get the names of the players, the default names are "player1" and "player2". After the 2 players enter their names, then press "GO" to start the game.
 Macromedia Flash Player 6 檔案(F) 檢視(V) 控制(C) 說明(H) John 小明 × 2 × 2	This is the display of Key Frame 2, a 10 by 10 chessboard is initialized with 2 white and 2 black chesses placed at the centre of the chessboard. The names of the players (John and 小明) are displayed on the top. An arrow is used to indicate who the player of the next turn is.
 Macromedia Flash Player 6 檔案(F) 檢視(V) 控制(C) 說明(H) John 小明 × 13 × 16	It is now the turn of 小明 and the arrow is pointing to the name of the player 2. During the game, the numbers of white and black chesses are displayed at the bottom right corner.

Testing and Evaluation

The game should be run in Windows 98, XP with Flash Player 6.

To test whether the Action Script (program) works correctly, the following has to be tested:

1. Users enter their name, in English or Chinese, and can be displayed on the top of the chessboard
2. Place chess and it is a valid move if there is no chess occupying the cell and it obeys the rule of the game
3. Turn another player's chesses after making a valid move
4. Count the number of black and white chesses manually and check against the numbers displayed at the bottom right corner of the chessboard
5. Change to another player's turn after turning chesses
6. When the chessboard is full, the game ends and display the correct winner or whether the game is a draw

After playing many times, my game satisfied the 6 criteria of the test plan.

Conclusion and Discussion

After writing this game, I think the objectives are met. The game can be put on almost every PC to be played. The game gives the players a feel of user-friendly as the design of the chessboard is on one hand concise and on the other hand displaying all information such as players' names and the number of chesses of each player. The game produced by Flash is small in terms of file size and therefore can be easily sent to people who are interested in this game.

There are cases where no player can make any more valid move before the chessboard is full and the game is "hang".

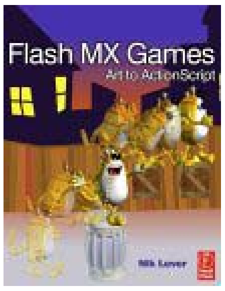
However, there are rooms for improvement:

1. If time allows, I can try to use the randomize function of Flash to implement a version of human vs. computer
2. Players should be able to select chessboard of different sizes
3. Play background music to give a soft atmosphere while players are playing
4. Since Flash swf objects can be handled by web browser, an online game version can be developed so that the two players can play on two different computers through Internet

References

1. Macromedia Flash MX Game Design Demystified: The Official Guide
   by Jobe Makar
   2002

2. Flash MX Games: Art to ActionScript
   by Nik Lever
   2002



Appendix

action script in frame 2

```
pla1_txt.text = player1;
pla2_txt.text = player2;

function Win(sumwhite,sumblack) {
    if (sumwhite>sumblack) {
        _global.winner= player1;
        gotoAndStop(3);
    } else if (sumblack>sumwhite) {
        _global.winner = player2;
        gotoAndStop(3);
    } else {
        gotoAndStop(4);
    }
}
```

action script of an object that in frame 2

```
onClipEvent (load) {
    _global.place = new Array(101);
_global.square = new Array(101);

tar = new Object();
for (var i = 1; i<=100; i++) {
    square[i] = "nothing";
    place[i] = i;
}
i = 45;
tar = eval("_root.a"+i);
tar.gotoAndStop(2);
_root.a46.gotoAndStop(3);
_root.a55.gotoAndStop(3);
_root.a56.gotoAndStop(2);
square[45] = "white";
square[46] = "black";
square[55] = "black";
square[56] = "white";
chess = 1;
colour = "white";
_root.pl2._visible = 0;
for (var i = 1; i<101; i++) {
    eval("_root.a"+place[i]).co = i;
    eval("_root.a"+place[i]).onRelease = function() {
        Nfound = 0;
        NEfound = 0;
        Efound = 0;
        SEfound = 0;
        Sfound = 0;
        SWfound = 0;
        Wfound = 0;
        NWfound = 0;
        checkN(this.co);
        checkNE(this.co);
        checkE(this.co);
        checkSE(this.co);
        checkS(this.co);
        checkSW(this.co);
        checkW(this.co);
```

```
            checkNW(this.co);
            if ((Nfound == 1 || NEfound == 1 || Efound == 1 || SEfound == 1 ||
Sfound == 1 || SWfound == 1 || Wfound == 1 || NWfound == 1) && (square[this.co]
=="nothing")) {
                this.gotoAndStop(chess+1);
                square[this.co] = colour;
                changechess(this.co);
                count();
                changeplayer();
            }
        };
    }
}
onClipEvent (load) {

    function count() {
        sumwhite = 0;
        sumblack = 0;
        for (i=1; i<=100; i++) {
            if (square[i] == "black") {
                sumblack = sumblack+1;
            }
            if (square[i] == "white") {
                sumwhite = sumwhite+1;
            }
        }
        _root.sumb_txt.text = sumblack;
        _root.sumw_txt.text = sumwhite;
    }
    function checkN(a) {
        _global.Nfound = 0;
        _global.Na = 0;
        var N = 20;
        if (square[a-10] == colour || a-10<0 || square[a-10] == "nothing") {
            Nfound == 0;
        } else {
            while (a-N>=0 && Nfound == 0 && square[a-N]<>"nothing") {
                if (square[a-N] == colour) {
```

```
                Nfound = 1;
                Na = a-N;
            } else {
                N = N+10;
            }
        }
    }
}
function checkS(a) {
    _global.Sfound = 0;
    _global.Sa = 0;
    var S = 20;
    if (square[a+10] == colour || a+10>100 || square[a+10] == "nothing")
{
        Sfound == 0;
    } else {
        while (a+S<=100 && Sfound == 0 && square[a+S]<>"nothing") {
            if (square[a+S] == colour) {
                Sfound = 1;
                Sa = a+S;
            } else {
                S = S+10;
            }
        }
    }
}
function checkE(a) {
    _global.Efound = 0;
    _global.Ea = 0;
    var E = 2;
    if (square[a+1] == colour || ((a+1)%10) == 0 || square[a+1] == "nothing")
{
        Efound == 0;
    } else {
        while (((a+E)%10)>0 && Efound == 0 && square[a+E]<>"nothing") {
            if (Square[a+E] == colour) {
                Efound = 1;
                Ea = a+E;
                } else {
                    E = E+1;
                }
            }
        }
    }
    function checkW(a) {
        _global.Wfound = 0;
        _global.Wa = 0;
        var W = 2;
        if (square[a-1] == colour || ((a-1)%10) == 0 || square[a-1] == "nothing")
{
            Wfound == 0;
        } else {
            while (((a-W)%10)>0 && Wfound == 0 && square[a-W]<>"nothing") {
                if (Square[a-W] == colour) {
                    Wfound = 1;
                    Wa = a-W;
                } else {
                    W = W+1;
                }
            }
        }
    }
    function checkNE(a) {
        _global.NEfound = 0;
        _global.NEa = 0;
        var NE = 18;
        if (square[a-9] == colour || a-9<0 || (a-9)%10 == 1 || square[a-9] ==
"nothing") {
            NEfound == 0;
        } else {
            while (a-NE>0 && NEfound == 0 && square[a-NE]<>"nothing") {
                if (Square[a-NE] == colour) {
                    NEfound = 1;
                    NEa = a-NE;
                } else {
                    NE = NE+9;
                    }
                }
            }
        }
        function checkSW(a) {
            _global.SWfound = 0;
            _global.SWa = 0;
            var SW = 18;
            if (square[a+9] == colour || a+9>100 || (a+9)%10 == 0 || square[a+9]
    == "nothing") {
                SWfound == 0;
            } else {
                while (a+SW>0 && SWfound == 0 && square[a+SW]<>"nothing") {
                    if (Square[a+SW] == colour) {
                        SWfound = 1;
                        SWa = a+SW;
                    } else {
                        SW = SW+9;
                    }
                }
            }
        }
        function checkNW(a) {
            _global.NWfound = 0;
            _global.NWa = 0;
            var NW = 22;
            if (square[a-11] == colour || a-11<0 || square[a-11] == "nothing") {
                NWfound == 0;
            } else {
                while (a-NW>0 && NWfound == 0 && square[a-NW]<>"nothing") {
                    if (Square[a-NW] == colour) {
                        NWfound = 1;
                        NWa = a-NW;
                    } else {
                        NW = NW+11;
                    }
                }
            }
            }
            function checkSE(a) {
                _global.SEfound = 0;
                _global.SEa = 0;
                var SE = 22;
                if (square[a+11] == colour || a+11>100 || square[a+11] == "nothing")
        {
                    SEfound == 0;
                } else {
                    while (a+SE<100 && SEfound == 0 && square[a+SE]<>"nothing") {
                        if (Square[a+SE] == colour) {
                            SEfound = 1;
                            SEa = a+SE;
                        } else {
                            SE = SE+11;
                        }
                    }
                }
            }
            function changechess(a) {
                if (Nfound == 1) {
                    for (var i = Na; i<a; i += 10) {
                        eval("_root.a"+i).gotoAndStop(chess+1);
                        square[i] = colour;
                    }
                }
                if (Sfound == 1) {
                    for (var i = a; i<Sa; i += 10) {
                        eval("_root.a"+i).gotoAndStop(chess+1);
                        square[i] = colour;
                    }
                }
                if (Efound == 1) {
                    for (var i = a; i<Ea; i++) {
                        eval("_root.a"+i).gotoAndStop(chess+1);
                        square[i] = colour;
                    }
                }
            if (Wfound == 1) {
                for (var i = Wa; i<a; i++) {
                    eval("_root.a"+i).gotoAndStop(chess+1);
                    square[i] = colour;
                }
            }
            if (NEfound == 1) {
                for (var i = NEa; i<a; i += 9) {
                    eval("_root.a"+i).gotoAndStop(chess+1);
                    square[i] = colour;
                }
            }
            if (SWfound == 1) {
                for (var i = a; i<SWa; i += 9) {
                    eval("_root.a"+i).gotoAndStop(chess+1);
                    square[i] = colour;
                }
            }
            if (SEfound == 1) {
                for (var i = a; i<SEa; i += 11) {
                    eval("_root.a"+i).gotoAndStop(chess+1);
                    square[i] = colour;
                }
            }
            if (NWfound == 1) {
                for (var i = NWa; i<a; i += 11) {
                    eval("_root.a"+i).gotoAndStop(chess+1);
                    square[i] = colour;
                }
            }
        }
        function changeplayer() {
            if (chess == 1 && colour == "white") {
                chess = 2;
                colour = "black";
                _root.pl1._visible = 0;
                _root.pl2._visible = 1;
            } else {
                    chess = 1;
                    colour = "white";
                    _root.pl1._visible = 1;
                    _root.pl2._visible = 0;
                }
            }

            function Win() {
                if (sumwhite>sumblack) {
                    winner_txt.text = player1;
                    gotoAndStop(3);
                } else if (sumblack>sumwhite) {
                    winner_txt.text = player2;
                    gotoAndStop(3);
                }else{
                    gotoAndStop(4);
                    }
            }

            for (var i = 1; i<=100; i++) {
                if (square[i]<>"nothing") {
                    win();
                }
            }
        }
```