EXERCISE 2 - GAME PLAYING (TIC-TAC-TOE)

1. Introduction

This handout helps you to implement a Animate movie for playing the Tic-tac-toe game. The program will allow you to play against an 'intelligent' computer.
This handout will give you the full ActionScript code to make the basic program work. You can use the code as it is (though you are encouraged to take inspiration from this code and improve the program further.). Alternatively you can write your own Actionscript from scratch. In any case, the code only controls the random playing strategy. After implementing the basic (random) playing strategy, you'll have to write your own function that implements a heuristic function.

2. Preparing the Animate movie

This program uses a 3x3 table with the symbol "X" for the human player, whilst the computer will use an "O".
To program the Tictactoe game in Animate you will need the following components:

- Layer 1 called "actionscript" for the ActionScript code, as in previous exercises.
- Layer 2 called "interaction" for the dialog box, an information text field, a "new game" button and another button to switch between a clever and a random computer playing strategy
- Layer 3 called "playing board" for the 9 tiles (3x3 grid) of the playing board
- Create a movieclip symbol called "tile_mc" that will be used to make the game playing board. This symbol will contain 3 keyframes, each having a graphics for the 3 states of the tictactoe tile: empty (use a square shape), a cross, a circle. Create nine instances of this symbol and organise them in a 3x3 grid. Name the 9 instances exactly as in the following table (*rc* stands for row and column, the first number for row position 1st to 3rd and the second for column position 1st to 3rd):

	rc00_mc	*rc01_mc*	*rc02_mc*	
	rc10_mc	*rc11_mc*	*rc12_mc*	
	rc20_mc	*rc21_mc*	*rc22_mc*	

- Create a button symbol with a label "New Game". Drag an instance on the movie and name it "newgame_btn".
- Create a text field for feedback to the user and call it "info_field".
- Add a movieclip in the Library of symbols and name it "clever_mc", as this will be used to switch on/off the intelligent algorithm. Add two keyframes, the first containing the text "Clever Computer" and the second "Random Computer". This will be used by the `PlayRandom`/`PlayIntelligent` functions (see section 5). When you click on this movieclip, it will change its state from frame 1 (intelligent) to frame 2 (random), and viceversa. Create an instance in the stage of this movieclip (within the interaction layer) and call the instance "clever_mc".

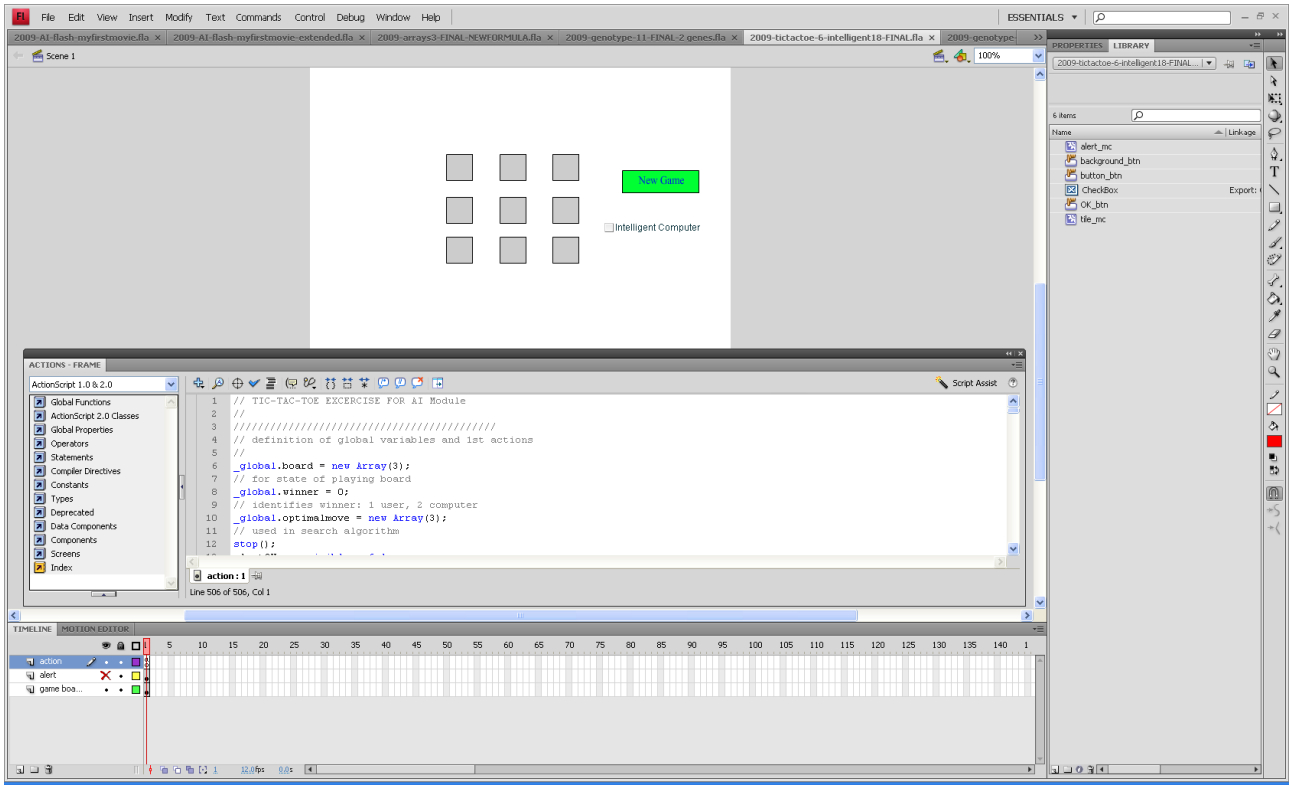
- We will need a new movieclip to use for a dialog box. This will be used to send messages to the user, e.g. “Congratulations, you have won”.
- Create a new movieclip symbol named “alert_mc”. This symbol will consist of (1) a text field named “alert_text” that will be used to show a message to the user; (2) the instance of a new button with a label saying “OK” and its name “alertOK_btn”; (3) a graphics for the dialog box background (it is advisable to use a blank large button symbol, instead of a pure graphics, to make sure that the dialog box background hides the objects behind it);
- Put an instance of the “alert_mc” into the “interaction” layer. Name the instance “alertOK_mc”.
- In the actionscript window add the following initialisation instructions at the top of the scripting window:

```
alertOK_mc.visible=false;// dialogbox not visible
```

- Also add the following action for the “alertOK_btn” button of the dialog box.

```
alertOK_mc.alertOK_btn.addEventListener(MouseEvent.MOUSE_UP, alertOK_off);
        function alertOK_off(event:MouseEvent) {
           alertOK_mc.visible=false;
        }
```

The Animate movie layout will be similar to the Figure below.



1. General Functions

Add the following initialisation instructions at the beginning of the “actionscript” Layer 1.

```
// definitions of global variables
clever_mc.gotoAndStop(2);  // by default, starts random
var board:Array=[["-","-","-"],["-","-","-"],["-","-","-"]];
var winner:int=0; // identifies winner: 1 user, 2 computer
```

```
var optimalmove:Array=new Array(3); // used in search algorithm
alertOK_mc.visible=false;
// link between the 9 tiles and choose_this_tile function
rc00_mc.addEventListener(MouseEvent.MOUSE_UP, choose_this_tile);
rc01_mc.addEventListener(MouseEvent.MOUSE_UP, choose_this_tile);
rc02_mc.addEventListener(MouseEvent.MOUSE_UP, choose_this_tile);
rc10_mc.addEventListener(MouseEvent.MOUSE_UP, choose_this_tile);
rc11_mc.addEventListener(MouseEvent.MOUSE_UP, choose_this_tile);
rc12_mc.addEventListener(MouseEvent.MOUSE_UP, choose_this_tile);
rc20_mc.addEventListener(MouseEvent.MOUSE_UP, choose_this_tile);
rc21_mc.addEventListener(MouseEvent.MOUSE_UP, choose_this_tile);
rc22_mc.addEventListener(MouseEvent.MOUSE_UP, choose_this_tile);
//
// actions linked to other buttons/instances
clever_mc.addEventListener(MouseEvent.MOUSE_UP, change_clever);
alertOK_mc.alertOK_btn.addEventListener(MouseEvent.MOUSE_UP,
⇔alertOK_acknowledged);
newgame_btn.addEventListener(MouseEvent.MOUSE_UP, newgame);
```

This array `board` is a global matrix of dimensions 3x3, and it corresponds to the 9 tiles of the 3x3 playing board/grid. It contains strings with the following 3 characters "`-`", "`0`", "`X`", i.e. respectively for indicating a tile that is empty, used by human player or used by the computer. The variable `winner` contains the following integer numbers `0`, `1` or `2` respectively to indicate that the winner is none, or the human player, or the computer. The vector array `optimalmove` will be used in the `playIntelligent` function to memorise the best move found. The first two positions of this vector are used to memorise the x and y coordinates of the best move. The third position is used as a tag (-9 when there is not an optimal move, 100 for a winning move, or the numerical result of the heuristic function in the `calculateheuristc2` function)
Now type the function **`newGame`** in the scripting window of the "actionscript" layer. This function will initialise the playing board and reset the global variables.

```
function newgame(event:MouseEvent) {
   var row:int,column:int;
   var tilename:String;
   board=[["-","-","-"],["-","-","-"],["-","-","-"]];
   // "-" = free, "X" = human player, "O" = computer
   winner=0;
   for (row=0; row<3; row++) {
        for (column=0; column<3; column++) {
             tilename="rc"+String(row)+String(column)+"_mc";
             this[tilename].gotoAndStop(1);
        }
   }
   info_field.text="Game started";
}
```

Note the use of the code `this[tilename].gotoAndStop(1)` to change the property of an instance whose name has been saved in the string variable `tilename`.

4. Main playing function (to choose the tile which has been clicked on)

This is the core interaction engine of the game playing, so you are asked to **type** this handler in the “actionscript” Layer 1 window.

```
function choose_this_tile(event:MouseEvent) {
   var clickedtilename:String=event.target.name;
   var myrow:int,mycolumn:int;
   if (this[clickedtilename].currentFrame==1) {// tile is free
        if ( (winner>0) || (boardcheck() == 9) ) {
             // Game over - tie; no free tiles
             alertOK_mc.visible=true;
             alertOK_mc.alert_text.text="Game over! To start
   ⇔a new game, click on the NewGame button";
             return;
        } else {
             myrow=Number(clickedtilename.substr(2,1));
             mycolumn=Number(clickedtilename.substr(3,1));
             board[myrow][mycolumn]="X";
             // the human player has used this cell
             this[clickedtilename].gotoAndStop(2);
             if (checkWinner()==1) {//Human player wins
                  alertOK_mc.visible=true;
                  alertOK_mc.alert_text.text="Congratulations,
⇔you won! Press NewGame button to play again";
                  winner=1;// the human player
                  return;
             }
             if (clever_mc.currentFrame==1) {
                  if (boardcheck()<9) {
                       playIntelligent();// computer uses heuristics
⇔ whose code will be given next week
                  }
             } else {
                  if (boardcheck()<9) {
                       playRandomly();// computer plays randomly
                  }
             }
             if (checkWinner()==2) {//Computer wins
                  alertOK_mc.visible=true;
                  alertOK_mc.alert_text.text="I won!!!";
                  winner=2;// the computer
             } else {
```

```
                    if (boardcheck()==9) {// Game over - tie
                         alertOK_mc.visible=true;
                         alertOK_mc.alert_text.text="Game over! To
⇔start a new game, click on the NewGame button";
                         return;
                    }
               }
          }
     }
}
```

4.1 Service functions

The following functions are used in different parts of the program. Type (or Copy&Paste) them in the “actionscript” layer 1 window.

```
function change_clever(event:MouseEvent) {
     if (clever_mc.currentFrame==1) {
          clever_mc.gotoAndStop(2);
     } else {
          clever_mc.gotoAndStop(1);
     }
}
```

```
function alertOK_acknowledged(event:MouseEvent) {
     alertOK_mc.visible=false;
}
```

```
function randomRangeInteger(min:int, max:int):int {
// returns a random integer between the two extremes min and max
     var randomNum:int = Math.floor(Math.random()*(max-min+1))+min;
     return randomNum;
}
```

```
function boardcheck():int {
// This checks if 9 tiles are used; returns number of used cells
     var row:int,column:int;
     var sum:int=0;
     for (row=0; row<3; row++) {
          for (column=0; column<3; column++) {
```

```
                if (board[row][column]!="-") {
                     sum++;
                }
           }
      }
      return sum;
}
```

```
function checkWinner():int {
//this function checks if computer-or-human has won
//returns winner's number (1 user, 2 computer)
      // first check rows
      var row:int,column:int;
      var you:int,computer:int;
      var player:String;
      var index:int;
      for (row=0; row<3; row++) {
           you=0;
           computer=0;
           for (column=0; column<3; column++) {
                player=board[row][column];
                switch (player) {
                     case "X" :
                          you++;
                          break;
                     case "O" :
                          computer++;
                          break;
                }
           }
           if (you==3) {
                return 1;
           }
           if (computer==3) {
                return 2;
           }
      }
      // check columns
      for (column=0; column<3; column++) {
           you=0;
           computer=0;
           for (row=0; row<3; row++) {
                player=board[row][column];
                switch (player) {
                     case "X" :
                          you++;
                          break;
                     case "O" :
                          computer++;
```

```
                    break;
               }
          }
          if (you==3) {
               return 1;
          }
          if (computer==3) {
               return 2;
          }
     }
     // check 1st diagonals
     you=0;
     computer=0;
     for (index=0; index<3; index++) {
          player=board[index][index];
          switch (player) {
               case "X" :
                    you++;
                    break;
               case "O" :
                    computer++;
                    break;
          }
     }
     if (you==3) {
          return 1;
     }
     if (computer==3) {
          return 2;
     }
     // check 2nd diagonal
     you=0;
     computer=0;
     for (index=0; index<3; index++) {
          player=board[index][2-index];
          switch (player) {
               case "X" :
                    you++;
                    break;
               case "O" :
                    computer++;
                    break;
          }
     }
     if (you==3) {
          return 1;
     }
     if (computer==3) {
          return 2;
     }
     return 0; // detault if no winner found
}
```

5. playRandomly function

The `playRandomly` function is called when the state of the `clever_mc` movieclip is on frame 2 (i.e. random play). This used the `checkWinner` function (as previous section) which is executed after each move. If checks all 8 rows/columns/diagonals combinations to see if there are three Xs or three Os lined up.
Copy this function in the Animate movie.

```
function playRandomly():void {
// Computer chooses a random free tile
   var computerrow:int,computercolumn:int;
   var tilename:String;
   var done:int=0;
   while (done == 0) {
        computerrow=randomRangeInteger(0,2);
        computercolumn=randomRangeInteger(0,2);
        if (board[computerrow][computercolumn]=="-") {
             board[computerrow][computercolumn]="O";
   tilename="rc"+String(computerrow)+String(computercolumn)+"_mc";
             this[tilename].gotoAndStop(3);// "O"
             done=1;
        }
   }
}
```

6. PlayIntelligent function

The computer will use the intelligent heuristic function (`playIntelligent`) when the state of the `clever_mc` movieclip is on frame 1 (i.e. intelligent computer).
Initially leave this as an empty function that does noting.

```
function playIntelligent():void {
// nothing to do - see assignment specification
}
```

Now that you have written all functions you should be able to play the Tic-tac-toe game against the computer. Remember that the `clever_mc` movieclip currently only works for random playing.

7. Interface improvement

Redesign the user interface of the present game, as the one suggested here is very simple. For example, you can add a function for showing the player's and computer's scores, improve the graphics, and so on.

ASSESSMENT EXERCISE (Exercise 2)

You will be assessed both for the implementation of the Random Playing strategy, as per code given above, as well as for the implementation of the MINIMAX algorithm for an Intelligent Playing strategy.

Adding intelligent (heuristic) function to your game

Your next task is to write the ActionScript code that implements a simplified version of the MINIMAX heuristic function.

For simplicity, only calculate the heuristic function of one level of depth (i.e. all the possible moves in the step), so that the computer can choose the one with the highest heuristic value.

For the heuristic function, you can either implement one of the two heuristics discussed during the lecture, or choose a different one.

- Heuristic function 1

$$3X_2 + X_1 - (3O_2 + O_1)$$

  where X_n is the number of rows with n "X" and no "O"

- Heuristic function 2

$$Open_{MAX} - Open_{MIN}$$

  where $Open_{MAX}$ is the number of complete rows/columns/diagonals that are still open for MAX

Modify the PlayIntelligent function

The `playIntelligent` function will be used when the state of the `clever_mc` movie clip is in frame 1 (intelligent computer).
In the previous part, this was an empty function that did nothing. In this new part of the assignment, add the code that calculates the best next move (i.e. the one with the highest heuristic function result).
If you want, use part of the `playRandomly`() function to place the "O" in the board.

```
function playIntelligent():void {
// here you'll add the code that implements your MINIMAX strategy

// First calculate the heuristic value for each of the possible
new moves

// identify row/column of the next play with highest heuristic
```

```
  // set 'computerrow' to the chosen row index
  // set 'computercolumn' to the chosen row column
  // use these two variables to set its content to 'O'

  }
```

Now that you have written all functions, including the `PlayIntelligent()` one, you should be able to play the Tic-tac-toe game against the computer.
Remember to set the "intelligent_CheckBox" on, so the `PlayIntelligent()` function is chosen.

Marking criteria

This second exercise is worth 15% of the module marks.

To mark this exercise, the following criteria will be used:
a) Does the basic program work correctly? (2 marks)
b) Has the graphic design and user interface been improved? (5 marks)
c) Is the heuristic function working? (8 points)