CSC 108
Spring 2010
Prof. Yue-Ling Wong

Lab 6: Tic-tac-toe Game
What you have for Lab06 so far

• two 2-D arrays to model the tic-tac-toe game board:
  - `board`: to store the movieclip instance names of the cells
  - `markings`: to store the markings by the two players on the board

• clicking on a cell will:
  - show the marking based on the `currPlayer`
  - update the corresponding value in the `markings` array
  - switch player after clicking on a cell
  - a text giving feedback on whose turn it is
# The Array markings

## Beginning of the game

<table>
<thead>
<tr>
<th>markings[0][0]</th>
<th>markings[0][1]</th>
<th>markings[0][2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>markings[1][0]</th>
<th>markings[1][1]</th>
<th>markings[1][2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>markings[2][0]</th>
<th>markings[2][1]</th>
<th>markings[2][2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The Array markings
After the game has started

<table>
<thead>
<tr>
<th>markings[0][0]</th>
<th>markings[0][1]</th>
<th>markings[0][2]</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>X</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>markings[1][0]</td>
<td>markings[1][1]</td>
<td>markings[1][2]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>markings[2][0]</td>
<td>markings[2][1]</td>
<td>markings[2][2]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tasks to remaining to do

1. reduce code duplication of the current code

2. make the marked cell non-clickable

3. check winner

4. announce who's the winner when there is one

5. take care of things when game over
   - prevent players from clicking on any of the unmarked cells
Let's go over code answer of the pre-lab

See handout
What you should have in the pre-lab

- two 2-D arrays to model the tic-tac-toe game board:
  - board: to store the movieclip instance names of the cells
  - markings: to store the markings by the two players on the board

- clicking on a cell will:
  - show the marking based on the currPlayer
  - update the corresponding value in the markings array
  - switch player after clicking on a cell
  - a text giving feedback on whose turn it is
Step 1.

Reduce code duplication
board[0][0].addEventListener(MouseEvent.MOUSE_UP, makeMove00);
board[0][1].addEventListener(MouseEvent.MOUSE_UP, makeMove01);
board[0][2].addEventListener(MouseEvent.MOUSE_UP, makeMove02);
board[1][0].addEventListener(MouseEvent.MOUSE_UP, makeMove10);
board[1][1].addEventListener(MouseEvent.MOUSE_UP, makeMove11);
board[1][2].addEventListener(MouseEvent.MOUSE_UP, makeMove12);
board[2][0].addEventListener(MouseEvent.MOUSE_UP, makeMove20);
board[2][1].addEventListener(MouseEvent.MOUSE_UP, makeMove21);
board[2][2].addEventListener(MouseEvent.MOUSE_UP, makeMove22);

function makeMove00(evt:MouseEvent):void{
    board[0][0].gotoAndStop("player" + currPlayer);
    markings[0][0] = currPlayer;
    trace(markings);
    if (currPlayer == 1) currPlayer = 2;
    else currPlayer = 1;
    txt_feedback.text = "Player " + currPlayer + ",

Problem:
Lots of code duplication!

We have 9 sets of event listener code.
What we are aiming at reducing the duplicated code

Aim to have **one single** event handler function instead of 9 different ones.

```javascript
function makeMove(evt: MouseEvent): void
{
  .
  .
  .
}
```
Problem:
Lots of code duplication!

Solution:
Organize code using function
Reduce code duplication using functions

Base Steps:
1. identify the duplicated code
2. put them into a new function definition
3. replace the original duplicated code with a function call to this function
board[0][0].addEventListener(MouseEvent.MOUSE_UP, makeMove00);
board[0][1].addEventListener(MouseEvent.MOUSE_UP, makeMove01);
board[0][2].addEventListener(MouseEvent.MOUSE_UP, makeMove02);
board[1][0].addEventListener(MouseEvent.MOUSE_UP, makeMove10);
board[1][1].addEventListener(MouseEvent.MOUSE_UP, makeMove11);
board[1][2].addEventListener(MouseEvent.MOUSE_UP, makeMove12);
board[2][0].addEventListener(MouseEvent.MOUSE_UP, makeMove20);
board[2][1].addEventListener(MouseEvent.MOUSE_UP, makeMove21);
board[2][2].addEventListener(MouseEvent.MOUSE_UP, makeMove22);

function makeMove00(evt:MouseEvent):void
{
    board[0][0].gotoAndStop("player" + currPlayer);
    markings[0][0] = currPlayer;
    trace(markings);
    if (currPlayer == 1) currPlayer = 2;
    else currPlayer = 1;
    txt_feedback.text = "Player " + currPlayer + "," + currPlayer + "," + currPlayer + "," + currPlayer + "," + currPlayer + ",";
}

function makeMove01(evt:MouseEvent):void
{
    board[0][1].gotoAndStop("player" + currPlayer);
    markings[0][1] = currPlayer;
    trace(markings);
    if (currPlayer == 1) currPlayer = 2;
    else currPlayer = 1;
    txt_feedback.text = "Player " + currPlayer + "," + currPlayer + "," + currPlayer + "," + currPlayer + "," + currPlayer + ",";
}
board[0][0].addEventListener(MouseEvent.MOUSE_UP, makeMove00);
board[0][1].addEventListener(MouseEvent.MOUSE_UP, makeMove01);
board[0][2].addEventListener(MouseEvent.MOUSE_UP, makeMove02);
board[1][0].addEventListener(MouseEvent.MOUSE_UP, makeMove10);
board[1][1].addEventListener(MouseEvent.MOUSE_UP, makeMove11);
board[1][2].addEventListener(MouseEvent.MOUSE_UP, makeMove12);
board[2][0].addEventListener(MouseEvent.MOUSE_UP, makeMove20);
board[2][1].addEventListener(MouseEvent.MOUSE_UP, makeMove21);
board[2][2].addEventListener(MouseEvent.MOUSE_UP, makeMove22);

function makeMove00(evt:MouseEvent):void
{
    board[0][0].gotoAndStop("player" + currPlayer);
    markings[0][0] = currPlayer;
    trace(markings);
    if (currPlayer == 1) currPlayer = 2;
    else currPlayer = 1;
    txt_feedback.text = "Player " + currPlayer + ",'s turn";
}

function makeMove01(evt:MouseEvent):void
{
    board[0][1].gotoAndStop("player" + currPlayer);
    markings[0][1] = currPlayer;
    trace(markings);
    if (currPlayer == 1) currPlayer = 2;
    else currPlayer = 1;
    txt_feedback.text = "Player " + currPlayer + ",'s turn";
board[0][0].addEventListener(MouseEvent.MOUSE_UP, makeMove00);
board[0][1].addEventListener(MouseEvent.MOUSE_UP, makeMove01);
board[0][2].addEventListener(MouseEvent.MOUSE_UP, makeMove02);
board[1][0].addEventListener(MouseEvent.MOUSE_UP, makeMove10);
board[1][1].addEventListener(MouseEvent.MOUSE_UP, makeMove11);
board[1][2].addEventListener(MouseEvent.MOUSE_UP, makeMove12);
board[2][0].addEventListener(MouseEvent.MOUSE_UP, makeMove20);
board[2][1].addEventListener(MouseEvent.MOUSE_UP, makeMove21);
board[2][2].addEventListener(MouseEvent.MOUSE_UP, makeMove22);

function makeMove00(evt:MouseEvent):void
{
    board[0][0].gotoAndStop("player" + currPlayer);
    markings[0][0] = currPlayer;
    updateGame();
}

function makeMove01(evt:MouseEvent):void
{
    board[0][1].gotoAndStop("player" + currPlayer);
    markings[0][1] = currPlayer;
    updateGame();
}

function updateGame():void
{
    trace(markings);
    if (currPlayer == 1) currPlayer = 2;
    else currPlayer = 1;
    txt_feedback.text = "Player " + currPlayer + ", turn";
function makeMove00(evt:MouseEvent):void
{
    board[0][0].gotoAndStop("player" + currPlayer);
    markings[0][0] = currPlayer;
    updateGame();
}

function makeMove01(evt:MouseEvent):void
{
    board[0][1].gotoAndStop("player" + currPlayer);
    markings[0][1] = currPlayer;
    updateGame();
}

function updateGame():void
{
    trace(markings);
    if (currPlayer == 1) currPlayer = 2;
    else currPlayer = 1;
    txt_feedback.text = "Player " + currPlayer + "'s turn";
}
board[0][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[0][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[0][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);

defunction makeMove(evt:MouseEvent):void {
    board[?][?].gotoAndStop("player" + currPlayer);
    markings[?][?] = currPlayer;
    trace(markings);
    if (currPlayer == 1) currPlayer = 2;
    else currPlayer = 1;
    txt_feedback.text = "Player " + currPlayer + ", turn";
}
board[0][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[0][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[0][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);

function makeMove(evt:MouseEvent):void
{
    board[?][?].gotoAndStop("player" + currPlayer);
    markings[?][?] = currPlayer;
    trace(markings);
    if (currPlayer == 1) currPlayer = 2;
    else currPlayer = 1;
    txt_feedback.text = "Player " + currPlayer + ", current turn";
}
board[0][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[0][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[0][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);

function makeMove(evt:MouseEvent):void
{
    evt.currentTarget.gotoAndStop("player" + currPlayer);
    markings[?][?] = currPlayer;
    trace(marking);
    if (currPlayer == 1) currPlayer = 2;
    else currPlayer = 1;
    txt_feedback.text = "Player " + currPlayer + ", turn";
}
board[0][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[0][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[0][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);

function makeMove(evt:MouseEvent):void
{
    evt.currentTarget.gotoAndStop("player" + currPlayer);
    markings[?] = currPlayer;
    trace(markings);
    if (currPlayer == 1) currPlayer = 2;
    else currPlayer = 1;
    txt_feedback.text = "Player " + currPlayer + "," + currPlayer;
}
Basic idea

• find the position (the two indexes) of currentTarget **in the board** array

• use the two indexes in markings[?][?]
Recall the gradebook example...
var gradebook:Array = new Array(
    new Array(82, 91, 73),
    new Array(64, 76, 55),
    new Array(78, 95, 68),
    new Array(87, 72, 92)
);

for (var row:int = 0; row < gradebook.length; row++)
{
    for (var col:int = 0; col < gradebook[row].length; col++)
    {
        gradebook[row][col] += 5;
    }
}
OK, now return to our tic-tac-toe code
for (var row:int = 0; row < board.length; row++)
{
    for (var col:int = 0; col < board[row].length; col++)
    {

    }
}


for (var row:int = 0; row < board.length; row++)
{
    for (var col:int = 0; col < board[row].length; col++)
    {
        if (evt.currentTarget == board[row][col])
        {
            markings[row][col] = currPlayer;
            break;
        }
    }
}
board[0][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[0][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[0][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][0].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][1].addEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][2].addEventListener(MouseEvent.MOUSE_UP, makeMove);

function makeMove(evt:MouseEvent):void
{
    evt.currentTarget.gotoAndStop("player" + currPlayer);
    for (var row:int = 0; row < board.length; row++)
    {
        for (var col:int = 0; col < board[row].length; col++)
        {
            if (evt.currentTarget == board[row][col])
            {
                markings[row][col] = currPlayer;
                break;
            }
        }
    }
    trace(markings);
    if (currPlayer == 1) currPlayer = 2;
    else currPlayer = 1;
    txt_feedback.text = "Player " + currPlayer + ", it's turn";
}
Step 2.

Make the marked cell non-clickable
removeEventListener()

Syntax:

target.removeEventListener(eventType, handlerName);

Example:

board[0][0].removeEventListener(MouseEvent.MOUSE_UP, makeMove);

When should you remove the mouse event listener?
function `makeMove`(`evt:MouseEvent`):`void`
{
    .
    .
    .
    .
    `???`.removeEventListener(`???`, `???`);
}

The `???` in the code should be `identical` to what's in the addEventListener statement of whatever event you want to remove (or disable).
Let's look at the second handout
Step 3.

Check Winner
For Player 1 to Win
For Player 2 to Win
Separate them into multiple
if/else-if statements
function hasAWinner():void
{
    if (all the cells in the row0 is 1)
        statement(s);
    else if (all the cells in the row1 is 1)
        statement(s);
    else if (all the cells in the row2 is 1)
        statement(s);
    else if (all the cells in the column0 is 1)
        statement(s);
    else if (all the cells in the column1 is 1)
        statement(s);
    else if (all the cells in the column2 is 1)
        statement(s);
    else if (all the cells in the diagonal0 is 1)
        statement(s);
    else if (all the cells in the diagonal1 is 1)
        statement(s);
}

returning a value in a function...
function hasAWinner(): Boolean {
    if (all the cells in the row0 is 1)
        return true;
    else if (all the cells in the row1 is 1)
        return true;
    else if (all the cells in the row2 is 1)
        return true;
    else if (all the cells in the column0 is 1)
        return true;
    else if (all the cells in the column1 is 1)
        return true;
    else if (all the cells in the column2 is 1)
        return true;
    else if (all the cells in the diagonal0 is 1)
        return true;
    else if (all the cells in the diagonal1 is 1)
        return true;
    return false;
}
function hasAWinner(): Boolean {
    if (all the cells in the row0 is 1)
        return true;
    if (all the cells in the row1 is 1)
        return true;
    if (all the cells in the row2 is 1)
        return true;
    if (all the cells in the column0 is 1)
        return true;
    if (all the cells in the column1 is 1)
        return true;
    if (all the cells in the column2 is 1)
        return true;
    if (all the cells in the diagonal0 is 1)
        return true;
    if (all the cells in the diagonal1 is 1)
        return true;

    return false;
}

Translated to Actionscript:

markings[0][0] == 1 &&
markings[0][1] == 1 &&
markings[0][2] == 1
function hasAWinner(): Boolean
{
    if (markings[0][0] == 1 &&
        markings[0][1] == 1 &&
        markings[0][2] == 1)
        return true;
    if (all the cells in the row 1 is 1)
        return true;
    if (all the cells in the row 2 is 1)
        return true;
    if (all the cells in the column 0 is 1)
        return true;
    if (all the cells in the column 1 is 1)
        return true;
    if (all the cells in the column 2 is 1)
        return true;
    if (all the cells in the diagonal 0 is 1)
        return true;
    if (all the cells in the diagonal 1 is 1)
        return true;
    return false;
}

But this is only checking whether Player 1 wins.

What about Player 2?
Solution:
use variable!
function hasAWinner(): Boolean {
    if (markings[0][0] == currPlayer &&
        markings[0][1] == currPlayer &&
        markings[0][2] == currPlayer)
        return true;
    if (all the cells in the row1 is currPlayer)
        return true;
    if (all the cells in the row2 is currPlayer)
        return true;
    if (all the cells in the column0 is currPlayer)
        return true;
    if (all the cells in the column1 is currPlayer)
        return true;
    if (all the cells in the column2 is currPlayer)
        return true;
    if (all the cells in the diagonal0 is currPlayer)
        return true;
    if (all the cells in the diagonal1 is currPlayer)
        return true;
    return false;
}
if (hasAWinner())
{
    display in the dynamic textbox the winner is currPlayer
}

Question:
But where do you place this if/else-if statement, i.e. how often should you call this function to check winner?
A. every time after a player makes a move
B. check it constantly at frame rate
if (hasAWinner())
{
    display in the dynamic textbox the winner is currPlayer
}
else
{
    swap player
    display in the dynamic textbox who's turn
}
if (hasAWinner())
{
    display in the dynamic textbox the winner is currPlayer
    gameOver();
}
else
{
    swap player
    display in the dynamic textbox who's turn
}
Step 4.

code to take care of game over
When game over…

• When there is winner, the game ends.

• And, any unmarked cells should be non-clickable. How do we code to make all cells non-clickable?
function gameOver():void
{
    removeEventListener for all the cells
}

OK, this is just a function definition.
But where in the code do you call this function?
Let's review the second handout
Step 5.
(Referred to as Stage 2 in the third handout)

use of nested loop to compact the code
Use a nested for-loop to compact it.

See the gradebook example and the nested for-loop in makeMove() function we have just gone over.
board[0][0].removeEventListener(MouseEvent.MOUSE_UP, makeMove);
board[0][1].removeEventListener(MouseEvent.MOUSE_UP, makeMove);
board[0][2].removeEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][0].removeEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][1].removeEventListener(MouseEvent.MOUSE_UP, makeMove);
board[1][2].removeEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][0].removeEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][1].removeEventListener(MouseEvent.MOUSE_UP, makeMove);
board[2][2].removeEventListener(MouseEvent.MOUSE_UP, makeMove);

Similarly, use a nested for-loop to compact it.
Extra Credits
4 Extra Credits

More need to be done to make this tic-tac-toe game complete:

1. check tie and tell the player it's tie

2. use the hand cursor

3. allow restarting the game

4. use of for-loop to compact 6 if-statements in the hasAWinner() function
Submitting the Extra Credits

• submit a separate file for each extra credit
Extra Credit #1: up to 10 points

add code to check tie and tell the player it's tie

Think about these questions first:

How do you define tie? What are the conditions?
Where should you place the code to check tie?

Hint: You need to create a variable to keep the count of the number of marked cells.
Extra Credit #2: up to 5 points

- make the hand cursor appear when the mouse is over an unmarked cell
- regular pointer cursor if the cell has been marked

**Hint:** use `buttonMode` property.

For example:
- to turn on the hand cursor for a moveclip instance:
  ```javascript
  movieclipInstance.buttonMode = true;
  ```
- to turn off the hand cursor for a movieclip instance:
  ```javascript
  movieclipInstance.buttonMode = false;
  ```
Extra Credit #3: up to 15 points

add code to allow restarting the game, say upon hitting spacebar, at any time

Hints:
• You will need to organize some of the statements into a function (that serves as a reset).
• Such a reset function is invoked at the beginning and when you press spacebar.
• Don't forget to make the cells to gotoAndStop(1)
Extra Credit #4: up to 5 points

• In the `hasAWinner()` function, use a for-loop to compact 6 (out of the 8) if-statements.

• Hint: You can either:
  – use two for-loops, each contains one if-statement
  or,
  – use one for-loop, which contains two if-statements