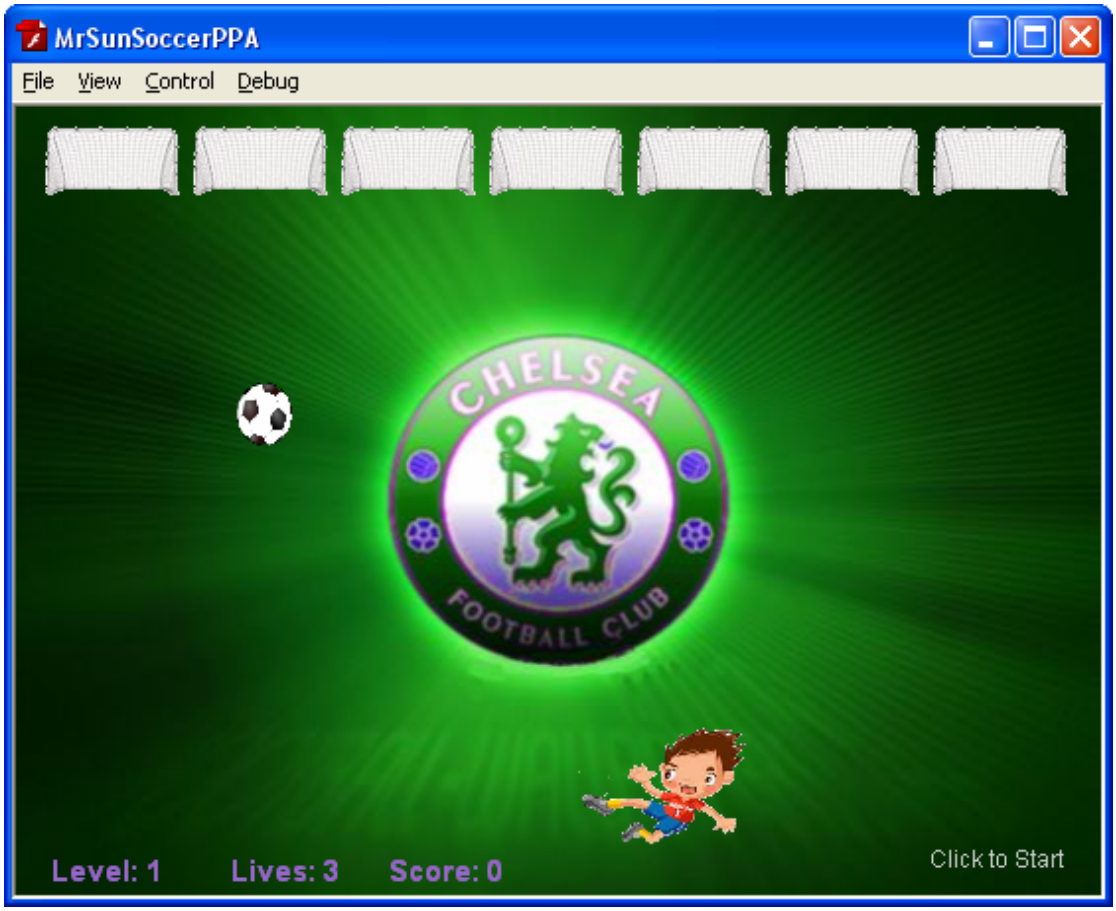
Flash Paddle Ball Soccer Game – AS 3.0 Code Listing

Here is a full listing of the Actionscript 3.0 code for the finished Paddle Ball Soccer game. (There is a separate document about how to build the game).



Frame 1 – Start Up Code

```
//IMPORTS
// We are setting the Brick (Goal) up as a "Flash Class"
// The import statement below brings in the "Brick.AS" file
// that defines the Brick as an extension of movie clip.
// Need this so we can add and take away bricks
// (which are actaully soccer goals) from the stage.
import Brick;
// This is in first frame of Flash Project and controls
// the level the player is on by initialising some items.
//Current level player is on
var currentLvl:int = 1;
//The array code for lvl 1 to 3
//All of the later levels add one more row of bricks
var lvl1Code:Array = new Array(1,1,1,1,1,1,1);
var lvl2Code:Array = new Array(1,1,1,1,1,1,1,1,1,1,1,1,1,1);
var lvl3Code:Array = new
Array(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1);
//The array that contains all of the level codes
var lvlArray:Array = new Array(lvl1Code, lvl2Code, lvl3Code);
```

Flash Paddle Ball Soccer Game – AS 3.0 Code Listing

Frame 2 - Main Game Control Code

```
stop();
//These variables are needed for moving the ball
var ballXSpeed:Number = 10;//X Speed of the Ball
var ballYSpeed:Number = 10;//Y Speed of the Ball
// The variable below will tell us how many bricks are on the
// stage, so that we can work out when to go to next level.
var brickAmt:int = 0;
//how many lives you got
var lives:int = 3;
//if the game is over
var gameOver:Boolean = false;
// Because the txtStart field is set up with the instance name of
// txtStart on the Flash properties of the dynamic textbox, we do not
// have to define that variable here, as it is already defined.
// so we do NOT put var txtStart:TextField = new TextField();
//Store the current game score in the following variable
var score:int = 0;
//
// This is where the begin Code Funtion starts
//First we define a function where all of
//the code needed to start the game is placed
//This includes listeners, variable definitions, and other items
function beginCode(event:MouseEvent):void{
      //beginCode had to wait for a user Mouse Click, so now we
      //are here, and the click was done, we now remove the
      //listener for the click
      stage.removeEventListener(MouseEvent.CLICK, beginCode);
      //remove the "Click to Start" Text from the screen
    txtStart.text = '';
      //Adds a listener to the paddle which
      //runs a function every time a frame passes
      mcPaddle.addEventListener(Event.ENTER_FRAME, movePaddle);
      //Adds a listener to the ball which
      //runs a function every time a frame passes
      mcBall.addEventListener(Event.ENTER_FRAME, moveBall);
      //adding a listener to check if the level is done
      // Level is done, when the nbr of bricks is reduced to zero
    addEventListener(Event.ENTER_FRAME, checkLevel);
}
// beginCode Function ends here
//
function movePaddle(event:Event):void {
      //The paddle follows the mouse and the mouse centres onto
paddle
      mcPaddle.x = mouseX - mcPaddle.width / 2;
      //If the mouse goes off too far to the left
      if (mouseX < mcPaddle.width / 2) {
            //Keep the paddle on stage
            mcPaddle.x = 0;
      }
      //If the mouse goes off too far to the right
      // Remebember the x-coord = the left hand edge of the paddle
      // because the registration point of the mc is default top
      // left hand corner of the paddle object.
      if (mouseX > stage.stageWidth - mcPaddle.width / 2) {
            //Keep the paddle on stage
            mcPaddle.x = stage.stageWidth - mcPaddle.width;
      }
```

```
}
//
function moveBall(event:Event):void {
	//Code for moving ball so it bounces off edges of screen
	mcBall.x += ballXSpeed;
	mcBall.y += ballYSpeed;
//Bouncing the ball off of the walls
	if(mcBall.x >= stage.stageWidth-mcBall.width){
		//if the ball hits the right side
		//of the screen, then bounce off
		ballXSpeed *= -1;
	}
	if(mcBall.x <= 0){
		//if the ball hits the left side
		//of the screen, then bounce off
		ballXSpeed *= -1;
	}
	if(mcBall.y >= stage.stageHeight-mcBall.height){
		//if the ball hits the bottom
		//then bounce up and lose a life
		ballYSpeed *= -1;
		lives --;
		//if there aren't any lives left
		if(lives <= 0){
			//the game is over now
			gameOver = true;
			//remove the play game event listeners
			mcPaddle.removeEventListener(Event.ENTER_FRAME,
movePaddle);
			mcBall.removeEventListener(Event.ENTER_FRAME,
moveBall);
			removeEventListener(Event.ENTER_FRAME, checkLevel);
			//go to a lose frame
			gotoAndStop('lose');
		}
	}
	if(mcBall.y <= 0){
		//if the ball hits the top
		//then bounce down
		ballYSpeed *= -1;
	}
	//This code checks to see if the ball hits the Paddle
	// and claculates a rebounding angle for the ball using
	// the calcBallAngle function
	if(mcBall.hitTestObject(mcPaddle)){
	calcBallAngle();
	}
}
//
function calcBallAngle():void{
	//BallPosition is the position of the ball is on the paddle
	// We alter the x coordinate to create a left or right angle.
	// So whcih end of the paddle we hit with, determines direction
	// by adjusting the x-coordinates down (left) or up (right).
	var ballPosition:Number = mcBall.x - mcPaddle.x;
	//hitPercent converts ballPosition into a percent
	//All the way to the left is -.5
	//All the way to the right is .5
	//The center is 0
	var hitPercent:Number = (ballPosition / (mcPaddle.width -
mcBall.width)) - .5;
```

```
      //Gets the hitPercent and makes it a larger number so the
      //ball actually bounces
      ballXSpeed = hitPercent * 10;
      //Making the ball bounce back up
      ballYSpeed *= -1;
}
function makeLvl():void{
//check if indeed there are any more levels left
      if(currentLvl > lvlArray.length){
            //the game is over now and we survived all the levels
        //and have therefore won the Game.
            gameOver = true;
            //remove all the Game Play Event listeners
            mcPaddle.removeEventListener(Event.ENTER_FRAME,
movePaddle);
            mcBall.removeEventListener(Event.ENTER_FRAME, moveBall);
            removeEventListener(Event.ENTER_FRAME, checkLevel);
            removeEventListener(Event.ENTER_FRAME, updateTextFields);
            //go to the special "won the game" frame
            gotoAndStop("win");
      }
      //If game not finished we places bricks onto Level and
      //finding the array length of the lvl code to get nbr of
levels.
    //
      //The index has to be currentLvl-1 because:
      //array indexes start on 0 and our lvl starts at 1
      //our level will always be 1 higher than the actual index of
the array
      var arrayLength:int = lvlArray[currentLvl-1].length;
      //the current row of bricks we are creating
      var brickRow:int = 0;
      //Now, creating a loop which places the bricks onto the stage
      for(var i:int = 0;i<arrayLength;i++){
            //checking if it should place a brick there
            if(lvlArray[currentLvl-1][i] == 1){
                  //creating a variable which holds the brick
instance
                  // The brick (Goal) is set up as a Class with its
own
                  // separate Brick.AS file.
                  var brick:MovieClip = new Brick();
                  //setting the brick's coordinates via the i
variable and brickRow
                  // Our soccer goals are 35 pixels tall, not 15
pixels like bricks.
                  // the brick.y settings below have been modified
accordingly.
                  brick.x = 15+(i-brickRow*7)*75;
                  brick.y = 10+brickRow*40;
                  //checks if the current brick needs a new row
                  for(var c:int = 1;c<=10;c++){
                        if(i == c*7-1){
                              brickRow ++;
                        }
                  }
                  //finally, add the brick to stage
                  addChild(brick);
            }
      }
}
```

Flash Paddle Ball Soccer Game – AS 3.0 Code Listing

```
//
function checkLevel(event:Event):void{
      //checking if the bricks are all gone
      if(brickAmt == 0){
            //reset the level by increasing the level
            currentLvl ++;
            //and re-running makeLvl
            makeLvl();
            //set the dynamic text field words
        txtStart.text = "Click to Start";
            //then reset the ball's and paddle's position
            mcBall.x = 150;
            mcBall.y = 265;
            mcPaddle.x = 230;
            //then removing all of the listeners
            mcPaddle.removeEventListener(Event.ENTER_FRAME,
movePaddle);
            mcBall.removeEventListener(Event.ENTER_FRAME, moveBall);
            removeEventListener(Event.ENTER_FRAME, checkLevel);
            //then listening for a mouse click to start the game
again
            stage.addEventListener(MouseEvent.CLICK, beginCode);
      }
}
//
function updateTextFields(event:Event):void{
      txtStats.text = "Level: "+currentLvl+"       Lives: "+lives+"
        Score: "+score;
}
// END OF ALL THE FUNCTION DEFINITIONS AND NOW WE START EXECUTING THE
GAME
//Wait until the mouse clicks, before beginning the game
stage.addEventListener(MouseEvent.CLICK, beginCode);
//Update the text fields as we pass through each frame of the game
addEventListener(Event.ENTER_FRAME, updateTextFields);
// Set up the bricks for the level the player is on
makeLvl();
//set the dynamic text field words
txtStart.text = "Click to Start";
```

Frame 5 – End of a Losing Game Code

```
//The lose game frame

//resetting the game if the mouse is clicked
stage.addEventListener(MouseEvent.CLICK, resetGame);

function resetGame(event:MouseEvent):void{
      //removing this listener
      stage.addEventListener(MouseEvent.CLICK, resetGame);
      //resetting the game
      gotoAndPlay(1);
}
//Display the final score as part of the You Lose screen
txtScore.text = "Final Score: "+score;
```

Flash Paddle Ball Soccer Game – AS 3.0 Code Listing

Frame 8 – End of a Winning Game Code

```
// There are not may lines of code needed
// for this "Winner Screen" frame
//First display the player's final score
txtScore.text = "Final Score: "+score;
// then wait for a mouse click to start a new game
stage.addEventListener(MouseEvent.CLICK, resetGame);
```

"Brick.AS" External Code for Adding and Removing Goals

```
// This is the "Brick.as" file that creates a class for the Brick
// (the bricks are replaced by the soccer goal in my game).
//  This AS file which is in the same folder as the other game
// components controls everything in relation to the brick (goals).
//
//Classes must always be wrapped in a package
package {
      //importing display elements that we can use in this class
      import flash.display.*;
      import flash.events.*;

      //defining the class name and saying that it
      //extends the MovieClip class, meaning that it has the same
      //properties as a movieclip
      public class Brick extends MovieClip {
            //The main timeline!
            private var _root:MovieClip;
            //all classes must have a function that runs every time
            //an instance of the class is put on stage
            public function Brick(){
                  //Code that will be run when the brick is added to
the stage
                  addEventListener(Event.ADDED, beginClass);
                  //Enter frame code
                  addEventListener(Event.ENTER_FRAME,
enterFrameEvents);
            }
            //private function are just functions that you can't
access
            //from the main timeline, but only within the class
itself
            private function beginClass(event:Event):void{
                  //defining _root as the document root
                  _root = MovieClip(root);
                  //incrementing how many bricks are on the stage
                  _root.brickAmt ++;
            }
            private function enterFrameEvents(event:Event):void{
                  //checking if the player has lost
                  if(_root.gameOver){
                        //destroy this brick
                        this.parent.removeChild(this);
                        //stop running this code
                        removeEventListener(Event.ENTER_FRAME,
enterFrameEvents);
                  }
                  //hit testing with the ball
```

```
                  if(this.hitTestObject(_root.mcBall)){
                        //making the ball bounce off vertically
                        _root.ballYSpeed *= -1;
                        //destroying this brick
                        this.parent.removeChild(this);
                        //stop running this code
                        removeEventListener(Event.ENTER_FRAME,
enterFrameEvents);
                        //decrementing the amount of bricks on stage
                        _root.brickAmt --;
                        //increasing the score
                        _root.score += 10;
                  }
            }
      }
}
```

REFERENCES :

Note that nearly all of the AS 3.0 code contained in this document came from Mr Sun's excellent tutorial on making a "Brick Breaker" game, which can be found at this website:

http://www.blogcatalog.com/blog/mr-sun-studios

I have merely put his code together and added a few changes and extra comments and explanations along the way.

Kenny Sun is the one who deserves the credit for the overall game structure and coding. His website site is great for learning Actionscript Games.
Do yourself a favor and check it out sometime !

Hopefully one day, if we keep doing lots of tutorials, we will all be able to design games and write complete Actionscript code like this. ☺

Enjoy,
Passy