Using Adobe Animate CC 2017 and ActionScript 3.0 to create a Pong Game
Created by: Stacey Fornstrom
Thomas Jefferson High School - Denver, CO
Student Project Examples: http://sfornstrom.tjcctweb.com/

Background:

ActionScript 3.0 is the programming component of Adobe Animate CC 2017. ActionScript can be used to add interactivity to web sites and learn basic programming concepts. ActionScript 3.0 is very similar to JavaScript. The advantage that ActionScript has over JavaScript is that Animate makes it easier for students to incorporate their own graphics and animations in their creations. I have found that one of the best ways to introduce students to ActionScript is to create a "**Pong**" game together as a class tutorial.

I use the Pong Tutorial to teach ActionScript to high school students in the Center for Communication Technology Magnet program at Thomas Jefferson High School. Students have completed 3 weeks of lessons with Animate in which they created multiple projects.

Class sessions are 50 minutes long. This unit covers 5 class sessions. The tutorial was completed with Adobe Animate CC 2017. Each student has their own computer and class sizes range from 20 to 29 students.

These lessons provide many opportunities for differentiation. Students can use any design they wish for the walls, balls, or paddle. The version of Pong that we create in class is very simple. Students add complexity by adding additional paddles, players, lives, balls, bricks, and anything else that they think of. The content interests most students enough to use their own time to enhance what we build in class together.

Daily Lessons:

I have a projector that I use to demonstrate each step required to create Pong. Students follow along with me, at the end of the unit they each have their own game of Pong. I assess daily progress by providing an assignment for each class period and asking students to show me their program when it is working correctly.

Day 1 of the Pong Tutorial is a review of skills we learned previously in Animate.

Objective:

What are you doing? We are working through in-class tutorials to create a Pong game with Adobe Animate CC 2017.

Why are you doing it? To learn programming and ActionScript 3.0 basics; and create a fun game!

What tools are you using? Adobe Animate CC 2017.

How will you know you are successful? We will **create** a **Pong game** with Adobe Animate CC 2017 that is **suitable for publishing**.

Project Overview: We will use Adobe Animate CC 2017 and ActionScript 3.0 to create a basic Pong game. There will be 2 frames on the main timeline. Frame 1 will have instructions and creator information, and a button to start the game. Frame 2 will have the game. The player will control the paddle with arrow keys. A ball will bounce within walls, Score will increase when the paddle hits the ball and decrease when it hits the bottom wall.

Skills Addressed

- Stage
- Timeline
- Toolbar
- Library
- Symbols – movie clips and buttons
- Instance Name
- Properties
- Frames, Key Frames, and Frame Rate
- Events and Event Handlers (functions)
- hitTestObject method

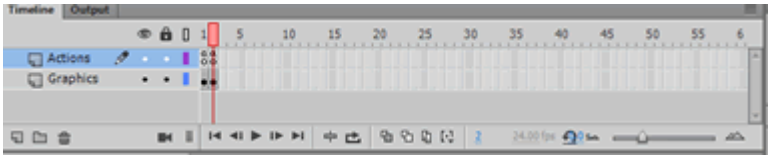
Animate CC 2017 – Pong Tutorial
Day 1
Objectives:

- Create an Introduction screen for a Pong game.
- Create a button to move from the Introduction screen to the Play screen.

Assignment:

- Create a new **Animate CC 2017** ActionScript 3.0 file, save to Google Drive as **pd_lastName_Pong**
- Set Stage Size = 800px by 600px
- On Main timeline, create 2 layers named **Actions** and **Graphics**
- In Frame 1, create instructions of how to play the game, information about creator and date made, and a **Start** button.
- Frame 1 should have an attractive design that uses an acceptable color scheme and effects to make the design unique and memorable. Use appropriate graphics, make an animation in a movie clip, adjust properties and effects.
- Code the **Start** button in frame 1 to go to Frame 2. (This is a review from previous lessons.)

Timeline:



Code for Frame 1:

```
stop( );
// Start Game button
startGame.addEventListener(MouseEvent.CLICK, fl_ClickToGoToAndPlayFromFrame);
function fl_ClickToGoToAndPlayFromFrame(event:MouseEvent):void
{
	gotoAndPlay(2);
}
```

Animate CC 2017 – Pong Tutorial
Day 2
Objectives:

- Describe steps necessary to create a basic Pong game.
- Create movie clips to use in the Pong game.

Assignment: We will create a basic Pong game with 1 paddle and a ball bouncing within 4 walls. In a Google Doc, describe what a programmer needs to do to create this Pong game. Ideas to include:

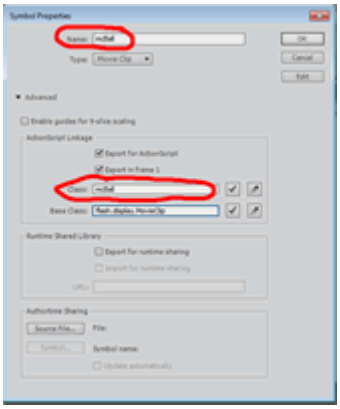
- Movie clips needed
- Variables to define and use
- Events to create
- Decisions to test for

You have 15 minutes to write your description. Save to Google Drive.

Day 2 Demo:

- Create movie clips to use in the Pong game, place in frame 2 in **Graphics** layer:

Shape	Symbol Name	Instance Name(s)
ball	mcBall	ball
paddle	mcPaddle	paddle
horizontal wall	mcHorizontalWall	wallTop wallBottom
vertical wall	mcVerticalWall	wallRight wallLeft



Notes & Things to Triple-Check!

- **Case-sensitive**: symbol names and instance names are both case-sensitive! This means that **Ball** is **NOT** the same as **ball**.
- **Names** must be **unique**; i.e. you cannot have a symbol named **ball** and an instance on the stage named **ball**.

Animate CC 2017 – Pong Tutorial
Day 3
Objectives:

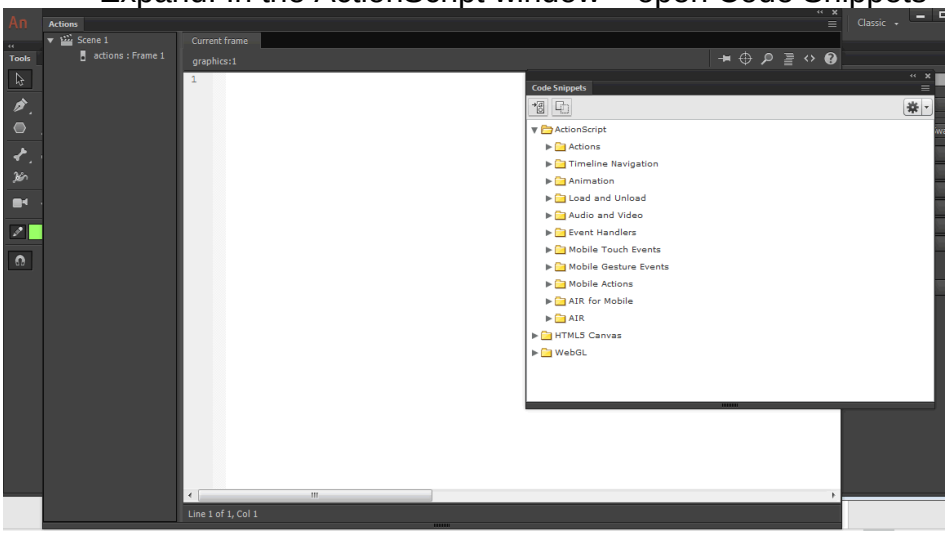
- Declare variables
- Program ball movement
- Program ball to bounce off walls

Day 3 Demo:
Declaring Variables in ActionScript 3.0:
To declare variables in ActionScript 3.0 you must use the **var** statement with the variable name. At the top of the Actions layer, immediately after the **stop( );** statement, add a section named
/* DECLARE VARIABLES */ We will always declare variables in the same location.

/* DECLARE VARIABLES */
var xMove = 5; // variable for ball movement on x-axis
var yMove = 5; // variable for ball movement on y-axis

Code Snippets – use a Code Snippet to create an Enter Frame Event for Frame 2:

- In Frame 2, select **ball**, make sure Instance Name = **ball**
- Expand: In the ActionScript window > open Code Snippets > expand **Event Handlers**



- Double-click **Enter Frame Event**
- Replace the line that says: **trace("Entered frame");** with the lines below

```
ball.x += xMove;
ball.y += yMove;
```

- The lines above make the ball move on the x-axis and the y-axis each time a new frame plays.
- Now add code to test if the ball hits a wall using hitTestObject. If the ball hits a wall, multiply **xMove** or **yMove** by -1 to change direction of movement.

```
// test wall hits
    if  (ball.hitTestObject(wallRight)) {
        xMove *= -1;
    } else if (ball.hitTestObject(wallBottom)){
        yMove *= -1;
    } else if (ball.hitTestObject(wallLeft)){
        xMove *= -1;
    }else if (ball.hitTestObject(wallTop)){
        yMove *= -1;
    }  // end if
```

- Test your program, the ball should now bounce within the walls.
- Make the ball bounce off the paddle by adding a new if statement in the Enter Frame event:

```
// bounce off the paddle
    if (ball.hitTestObject(paddle)) {
        yMove*= -1;
    }
```

- Test your program, the ball should now bounce off the paddle.

Notes:

- **Comments:** used by programmers to provide notes and descriptions of what the code is doing. Comments are ignored by the computer. ActionScript 3.0 uses **//** for single line comments or **/*** for multi-line comments ***/**
- **Code Snippet**: pre-made ActionScript code to program common interactions in an Animate program.
- **Enter Frame Event:** Animate movies repeatedly play frames; the Enter Frame event occurs each time a new frame is played. So if the frame rate for a movie is set to 24fps, Enter Frame occurs 24 times per second.
- **hitTestObject method:** method in ActionScript 3.0 to determine if 2 instances touch on the stage. Example: if (ball.hitTestObject(wallRight)) → returns True or False, the programmer provides instructions of what to do if the result = True.

Animate CC 2017 – Pong Tutorial
Day 4
Objectives:

- Review importance of Symbol Names and Instance Names
- Review code for ball movement, hitTestObject
- Review variable declarations
- Program paddle to move with arrow keys

REVIEW

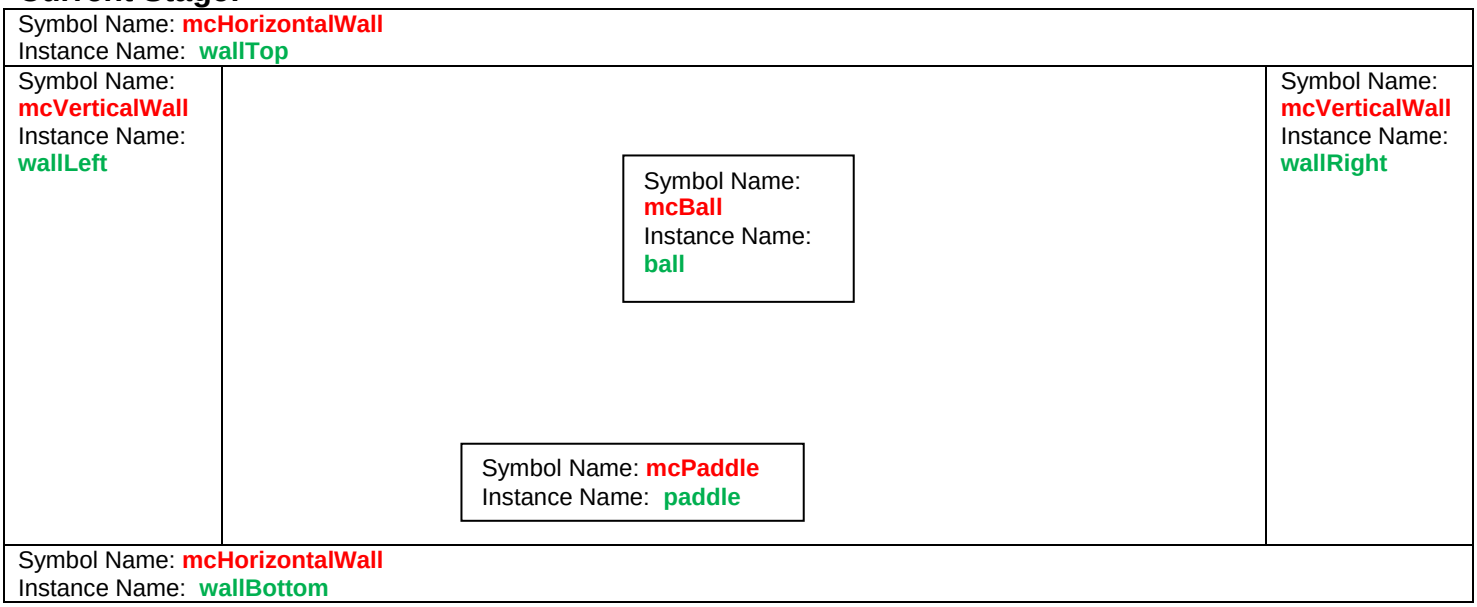
Symbols: A ***symbol*** is a graphic, button, or movie clip that you create once in Animate CC. The symbol is stored in the **library**. Symbols can be reused throughout the document or in other documents.

Symbol Name: The name used for a symbol in the library. (Example: people, dog)

Instance Name: The name used to refer to an instance of a symbol on the stage. Many instances of a symbol can be used on the stage and they are distinguished by the Instance Name. The Instance Name is case-sensitive when used in ActionScript 3.0. (Example: Joe, Mary, Spot, Ruff)

hitTestObject: method in ActionScript 3.0 to determine if 2 instances touch on the stage. Example: if (ball.hitTestObject(wallRight))

Current Stage:



Current Code for Frame 2 in Pong Game:

```
stop( );
/*  DECLARE VARIABLES  */
var xMove = 5;    // variable for ball movement on x-axis
var yMove = 5;    // variable for ball movement on y-axis

/* Enter Frame Event */
addEventListener(Event.ENTER_FRAME, fl_EnterFrameHandler);
function fl_EnterFrameHandler(event:Event):void
{
        ball.x += xMove;
        ball.y += yMove;
        // test wall hits
        if  (ball.hitTestObject(wallRight)) {
                xMove *= -1;
        } else if (ball.hitTestObject(wallBottom)){
                yMove *= -1;
        } else if (ball.hitTestObject(wallLeft)){
                xMove *= -1;
        }else if (ball.hitTestObject(wallTop)){
                yMove *= -1;
        }  // end if

        // bounce off the paddle
        if (ball.hitTestObject(paddle)) {
                yMove*= -1;
        }
}    // fl_EnterFrameHandler
```

Notes on above Code for ActionScript 3.0:

- **var**: keyword to define a new variable
- **hitTestObject**: used to test if 2 instances are touching
- **{ }** : curly brackets are used to delineate the begin and end of blocks of code; for example: functions, if statements, loops
- Instance Names are case-sensitive

Animate CC 2017 – Pong Tutorial

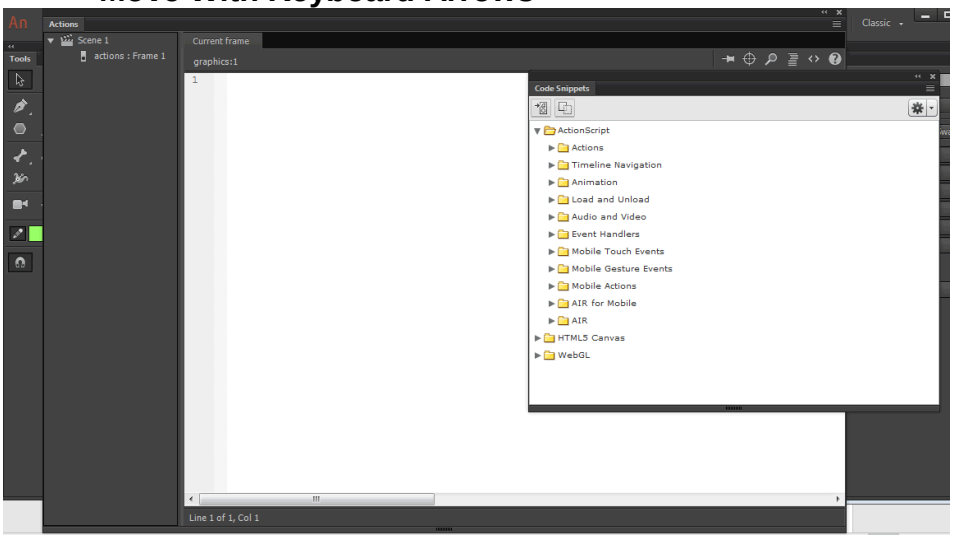
Day 4

Objectives:

- Program Paddle Movement with a Code Snippet
- Program additional key presses.

Program Paddle Movement with a Code Snippet:

- Select the **paddle** on Frame 2 of the stage. Make sure the instance name is **paddle**.
- Expand: In the ActionScript window > open Code Snippets > expand **Animation** > double-click **Move With Keyboard Arrows**



This is the code created by the steps above:

```
/* Move the paddle with Keyboard Arrows */
/* Declare variables to track if a key is pressed */
var upPressed:Boolean = false;
var downPressed:Boolean = false;
var leftPressed:Boolean = false;
var rightPressed:Boolean = false;

paddle.addEventListener(Event.ENTER_FRAME, fl_MoveInDirectionOfKey);
stage.addEventListener(KeyboardEvent.KEY_DOWN, fl_SetKeyPressed);
stage.addEventListener(KeyboardEvent.KEY_UP, fl_UnsetKeyPressed);

function fl_MoveInDirectionOfKey(event:Event)
{
	if (upPressed)
	{
		paddle.y -= 5;
	}
	if (downPressed)
	{
		paddle.y += 5;
	}
	if (leftPressed)
	{
		paddle.x -= 5;
	}
	if (rightPressed)
	{
		paddle.x += 5;
	}
}  // fl_MoveInDirectionOfKey
```

```
function fl_SetKeyPressed(event:KeyboardEvent):void
{
	switch (event.keyCode)
	{
		case Keyboard.UP:
		{
			upPressed = true;
			break;
		}
		case Keyboard.DOWN:
		{
			downPressed = true;
			break;
		}
		case Keyboard.LEFT:
		{
			leftPressed = true;
			break;
		}
		case Keyboard.RIGHT:
		{
			rightPressed = true;
			break;
		}
	}
}	// fl_SetKeyPressed

function fl_UnsetKeyPressed(event:KeyboardEvent):void
{
	switch (event.keyCode)
	{
		case Keyboard.UP:
		{
			upPressed = false;
			break;
		}
		case Keyboard.DOWN:
		{
			downPressed = false;
			break;
		}
		case Keyboard.LEFT:
		{
			leftPressed = false;
			break;
		}
		case Keyboard.RIGHT:
		{
			rightPressed = false;
			break;
		}
	}
}	// fl_UnsetKeyPressed
```

Notes on Move With Keyboard Arrows code snippet:

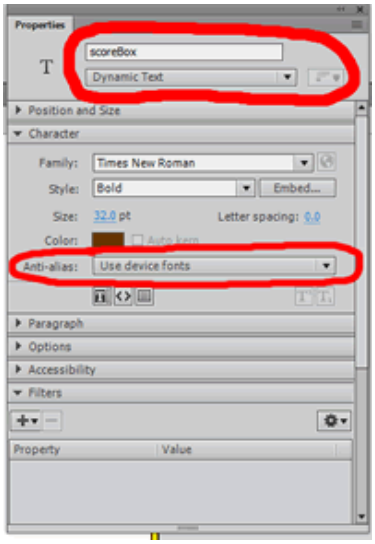
- 4 Boolean variables are created: **upPressed**, **downPressed**, **leftPressed**, **rightPressed**. **Boolean** variables can be **true** or **false**.
- 3 functions are created:
  - **fl_SetKeyPressed** – determines if a key that is listened for is pressed, if yes: set appropriate variable to **true**.
  - **fl_UnsetKeyPressed** - determines if a key that is listened for is released, if yes: set appropriate variable to **false**.
  - **fl_MoveInDirectionOfKey** – is a function that executes on each Enter Frame event for the paddle. If a key variable is **true**, the paddle is moved in the appropriate direction.
- **Conditionals** – used to control program flow; tests a condition and executes the true statement. 3 types in ActionScript 3.0: **if..else**, **if..else if**, **switch**
  http://help.adobe.com/en_US/ActionScript/3.0_ProgrammingAS3/WS5b3ccc516d4fbf351e63e3d118a9b90204-7fce.html
- **switch** statement: useful if you have several execution paths that depend on the same condition expression. It provides functionality similar to a long series of if..else if statements, but is somewhat easier to read. Blocks of code begin with a **case** statement and end with a **break** statement.

Animate CC 2017 – Pong Tutorial
Day 5
Objectives:

- Add score variable
- Display score variable

Add 2 text boxes for the scoreboard.

1. A **Static Text** box, type **Score:** in the box
2. A Dynamic Text box, see instructions and a picture of the Properties Window below.



Create Text Box > change Properties to **Dynamic Text** > change Instance Name = **scoreBox** > Properties: Character > Anti-alias > choose **Use device fonts**

- Declare a variable named **score** in **/* DECLARE VARIABLES */**

```
var score = 0;
```

- Display score variable; on the next line add:

```
scoreBox.text = score;
```

- Add 1 to the current score when the ball hits the paddle. Update the scoreBox. In the Enter Frame event, adjust the hitTestObject test between ball and paddle so it looks like:

```
// bounce off the paddle
    if (ball.hitTestObject(paddle)) {
        yMove*= -1;
        score +=1;      // new line of code
    }
    scoreBox.text = score;     // new line of code
```

Animate CC 2017 – Pong Tutorial
Day 6 & 7– Extra Features
Objectives:

- Over the next 2 days: complete the **basic Pong** program.
- Design **at least 2 additional features** and include them in your game. Examples: additional pong balls, extra paddles, lives, harder levels, timers, background pictures or colors, additional objects for the ball to bounce off (like a pinball game).

TJ CCT Magnet
Animate CC 2017 – Pong Tutorial

Pong Rubric:

Component	Points
Design – attractive, follows accepted design principles, good color combinations, use creative design components such as effects, animations. Stage size = 800px by 600px. (5 pts)	
Graphics – appropriate, complete, interesting. (5 pts)	
Names – all symbols and instances are named with camelCase naming convention, no spaces in names. (3 pts)	
Intro Screen – Creator information, date created, objective, play instructions. (5 pts)	
Play Screen – Attractive design and layout. (5 pts)	
Paddle – moves with arrows or other keys. (2 pts)	
Ball – moves on Enter Frame; does not jump over walls. (2 pts)	
Score – updated based on events in the game. (3 pts)	
ActionScript code – indented to enhance readability; developer comments included. (5 pts)	
EXTRAS: something of your own design not addressed in the above requirements. (15 pts) **C** = Complete the steps of in-class tutorial to create a "basic" Pong game. Design must be attractive and use the capabilities of Animate such as effects and animations. **B** = Add 1 additional feature to the game. **A** = Add at least 2 additional features to the game. **100%** = Add at least 3 additional, ***complex*** features to the game.	
TOTAL (50 points possible)	

TJ CCT Magnet
Animate CC 2017 – Pong Tutorial

Pong Rubric:

Component	Points
Design – attractive, follows accepted design principles, good color combinations, use creative design components such as effects, animations. Stage size = 800px by 600px. (5 pts)	
Graphics – appropriate, complete, interesting. (5 pts)	
Names – all symbols and instances are named with camelCase naming convention, no spaces in names. (3 pts)	
Intro Screen – Creator information, date created, objective, play instructions. (5 pts)	
Play Screen – Attractive design and layout. (5 pts)	
Paddle – moves with arrows or other keys. (2 pts)	
Ball – moves on Enter Frame; does not jump over walls. (2 pts)	
Score – updated based on events in the game. (3 pts)	
ActionScript code – indented to enhance readability; developer comments included. (5 pts)	
EXTRAS: something of your own design not addressed in the above requirements. (15 pts) **C** = Complete the steps of in-class tutorial to create a "basic" Pong game. Design must be attractive and use the capabilities of Animate such as effects and animations. **B** = Add 1 additional feature to the game. **A** = Add at least 2 additional features to the game. **100%** = Add at least 3 additional, ***complex*** features to the game.	
TOTAL (50 points possible)	

Pong – Fornstrom's Final Code for Frame 2

```
stop( );
// define variables
var xMove = 5;    // variable for ball movement on x-axis
var yMove = 5;    // variable for ball movement on y-axis
var score = 0;    // declared variable
scoreBox.text = score;  // display variable

/* Enter Frame Event */
addEventListener(Event.ENTER_FRAME, fl_EnterFrameHandler);
function fl_EnterFrameHandler(event:Event):void
{
        ball.x += xMove;
        ball.y += yMove;
        // test wall hits
        if      (ball.hitTestObject(wallRight)) {
                xMove *= -1;
        } else if (ball.hitTestObject(wallBottom)){
                yMove *= -1;
                score += 1;
        } else if (ball.hitTestObject(wallLeft)){
                xMove *= -1;
        }else if (ball.hitTestObject(wallTop)){
                yMove *= -1;
        }  // end if
        // bounce off the paddle
        if (ball.hitTestObject(paddle)) {
                yMove*= -1;
                score +=1;
        }
        scoreBox.text = score;
        // test the score
        if (score >= 10) {
                gotoAndPlay(3); // plays next level
        }

}  // fl_EnterFrameHandler

/* Move the paddle with Keyboard Arrows */
/* Declare variables to track if a key is pressed */
var upPressed:Boolean = false;
var downPressed:Boolean = false;
var leftPressed:Boolean = false;
var rightPressed:Boolean = false;

paddle.addEventListener(Event.ENTER_FRAME, fl_MoveInDirectionOfKey);
stage.addEventListener(KeyboardEvent.KEY_DOWN, fl_SetKeyPressed);
stage.addEventListener(KeyboardEvent.KEY_UP, fl_UnsetKeyPressed);

function fl_MoveInDirectionOfKey(event:Event)
{
```

```
		// if (upPressed)
		if ( (upPressed) && !paddle.hitTestObject(wallTop) )
		{
			paddle.y -= 5;
		}
		if (downPressed)
		{
			paddle.y += 5;
		}
		if (leftPressed)
		{
			paddle.x -= 5;
		}
		if (rightPressed)
		{
			paddle.x += 5;
		}
}  // fl_MoveInDirectionOfKey

function fl_SetKeyPressed(event:KeyboardEvent):void
{
		switch (event.keyCode)
		{
			case Keyboard.UP:
			{
				upPressed = true;
				break;
			}
			case Keyboard.DOWN:
			{
				downPressed = true;
				break;
			}
			case Keyboard.LEFT:
			{
				leftPressed = true;
				break;
			}
			case Keyboard.RIGHT:
			{
				rightPressed = true;
				break;
			}
		}
}   // fl_SetKeyPressed

function fl_UnsetKeyPressed(event:KeyboardEvent):void
{
		switch (event.keyCode)
		{
			case Keyboard.UP:
			{
```

```
                upPressed = false;
                break;
            }
            case Keyboard.DOWN:
            {
                downPressed = false;
                break;
            }
            case Keyboard.LEFT:
            {
                leftPressed = false;
                break;
            }
            case Keyboard.RIGHT:
            {
                rightPressed = false;
                break;
            }
        }
}   // fl_UnsetKeyPressed
```

EXTRA NOTES – FAQ's:

Q: How do I make it so the paddle does not move through walls?
A: Add another test in the if statement. Find the function named:
fl_MoveInDirectionOfKey(event:Event)
change this line:
if (upPressed)
to:
if ((upPressed) && !paddle.hitTestObject(wallTop))

What does this do?
&&: means **AND**, this is just like Algebra because both conditions must be true in order for the statement to be true.
!: means **NOT**, so we are testing to see that the paddle is NOT hitting the wall.

Q: How do I make a Countdown Timer and display Time Left?
A: Use Code Snippets to make a Timer. Expand Code Snippets and click:

Q: How do I go to another frame if the Timer reaches 0?
A:

Q: How do I program movie clips to respond to keys other than the arrow keys?
A: If you have programmed the Arrow keys to move the paddle, there are 3 steps
fl_SetKeyPressed – determines if a key that is listened for is pressed, if yes: set appropriate variable to **true**.
- **fl_UnsetKeyPressed** - determines if a key that is listened for is released, if yes: set appropriate variable to **false**.

fl_MoveInDirectionOfKey – is a function that executes on each Enter Frame event for the paddle. If a key variable is **true**, the paddle is moved in the appropriate direction.

Q: I am receiving the following error and don't know how to fix it.
TypeError: Error #1009: Cannot access a property or method of a null object reference.
at S17_Fornstrom_Pong_fla::MainTimeline/fl_EnterFrameHandler( )
A: Let's assume the error is being thrown by the following code that goes to a new level when the score is greater than 10.

```
if (score >= 10) {
        gotoAndPlay(3); // plays next level
}
```

Add the line in **blue** to remove the Enter Frame event listener when leaving Frame 2:

```
if (score >= 10) {
        gotoAndPlay(3); // plays next level
        removeEventListener(Event.ENTER_FRAME, fl_EnterFrameHandler);
}
```

WHY IT WORKS: We left a listener with nothing to do and it's upset. The **removeEventListener** command tells it that it's done with its work and unloads it from the program.