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

Objective:

What are you doing? We are working through in-class tutorials to learn how to code with ActionScript 3.0 within 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 **program with basic Character Movement** with Adobe Animate CC 2017 that is **suitable for publishing**.

Project Overview: We will use Adobe Animate CC 2017 and ActionScript 3.0 to **program basic character movement**. There will be 1 frame on the main timeline, with a character named **hero**. The player will control the hero with arrow keys. A different animation will play based upon which key is pressed. When no keys are pressed, hero will display a static graphic.

Skills Addressed

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

Animate CC 2017 – Character Movement Tutorial

Day 1

Project Overview:

- We will have 3 days to create graphics for a character and program the character to move in 4 directions with the arrow keys. The character will also have a static state.

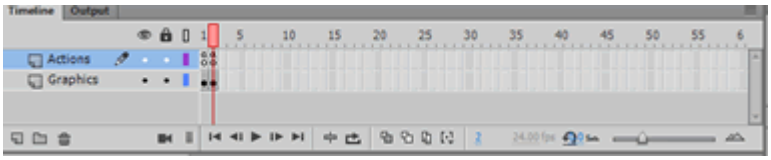
Objectives:

- File creation and basic layout.
- Graphics: import sprites or create new animations for a character.

Assignment:

- Create a new **Animate CC 2017** ActionScript 3.0 file, save to Google Drive as **pd_lastName_characterMovement**
- Set Stage Size = 800px by 600px
- On Main timeline, create 2 layers named **Actions** and **Graphics**
- In Frame 1, code a **stop( )** action.

Timeline:



Code for Frame 1:

```
stop( );
```

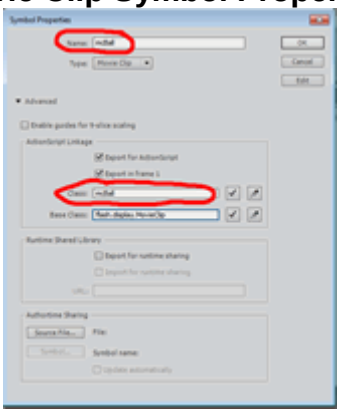
Import movie clips for a **hero character**, you will have the following movie clips in your library at the end of today:

Description	Symbol Name for Tutorial
Static	mcHero
Animation moving right	mcHeroRight
Animation moving left	mcHeroLeft
Animation moving up	mcHeroUp
Animation moving down	mcHeroDown

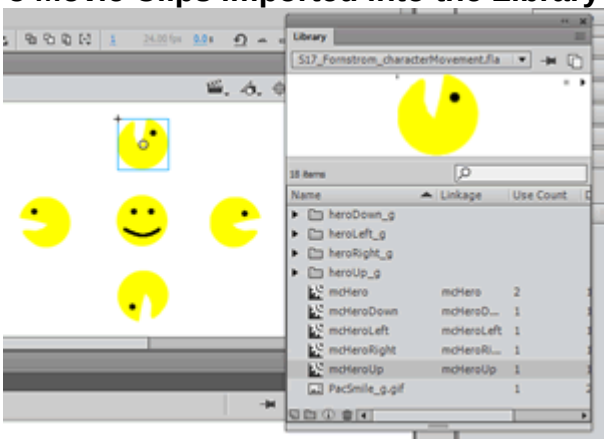
Import Animated GIFs:

- Click File > Import > Import to Library > choose the Animated GIF file that you want to import.
- The Animated GIF is imported as a movie clip. All unique graphics are also imported to a folder that is in the library with a name similar to the movie clip symbol name.
- Change the properties of the movie clip: Right-click on the movie clip in the library > choose **Properties**. Check the Export for ActionScript choice.

Movie Clip Symbol Properties



5 Movie Clips imported into the Library



Animate CC 2017 – Character Movement Tutorial
Day 2
Objectives:

- Create 1 movie clip on the stage that contains the static movie clip and the 4 animations.

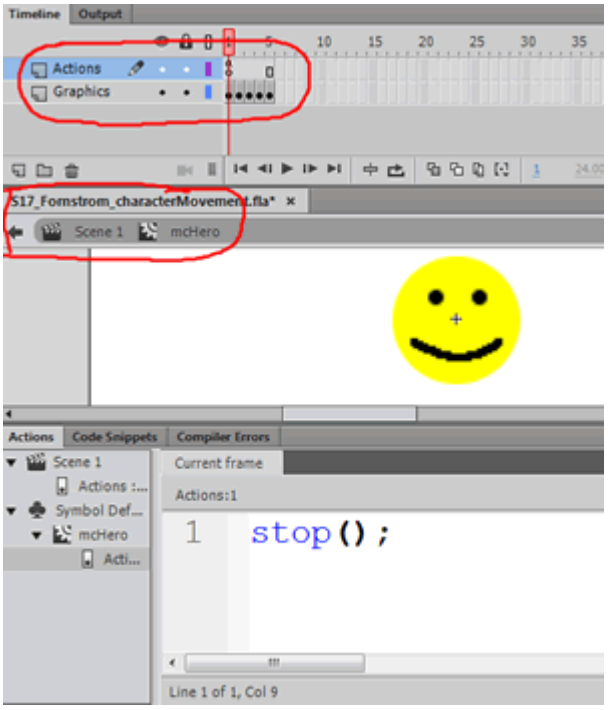
Instance Name: **hero** → hero's timeline will have the following in Frame 1 through Frame 5.

1. mcHero	2. mcHeroRight	3. mcHeroLeft	4. mcHeroUp	5. mcHeroDown

Instructions:

- Drag the static hero movie clip onto the stage. Set instance name = **hero**
- Double-click on **hero**. You are now in the timeline for the **hero** instance. Name the 1st layer **Actions**, add a 2nd layer and name this layer **Graphics**
- Select **Actions** layer, put in code: **stop( );**
- Select graphics layer frame 1, right-click and choose **Convert to Keyframe**. Delete static **mcHero** graphic in frame 2.
- Drag **mcHeroRight** into frame 2.
- In frame 2, right-click and choose **Convert to Keyframe**. Delete **mcHeroRight** in frame 3.
- Drag **mcHeroLeft** into frame 3.
- In frame 3, right-click and choose **Convert to Keyframe**. Delete **mcHeroLeft** in frame 4.
- Drag **mcHeroUp** into frame 4.
- In frame 4, right-click and choose **Convert to Keyframe**. Delete **mcHeroUp** in frame 5.
- Drag **mcHeroDown** into frame 5.

The Timeline for the **hero** Instance of **mcHero** will now look like:



NOTES:

- The main timeline has 1 frame. The **hero instance timeline** has 5 frames.
  - Frame 1: hero static graphic.
  - Frame 2: mcHeroRight movie clip
  - Frame 3: mcHeroLeft movie clip
  - Frame 4: mcHeroUp movie clip
  - Frame 5: mcHeroDown movie clip

Animate CC 2017 – Character Movement Tutorial

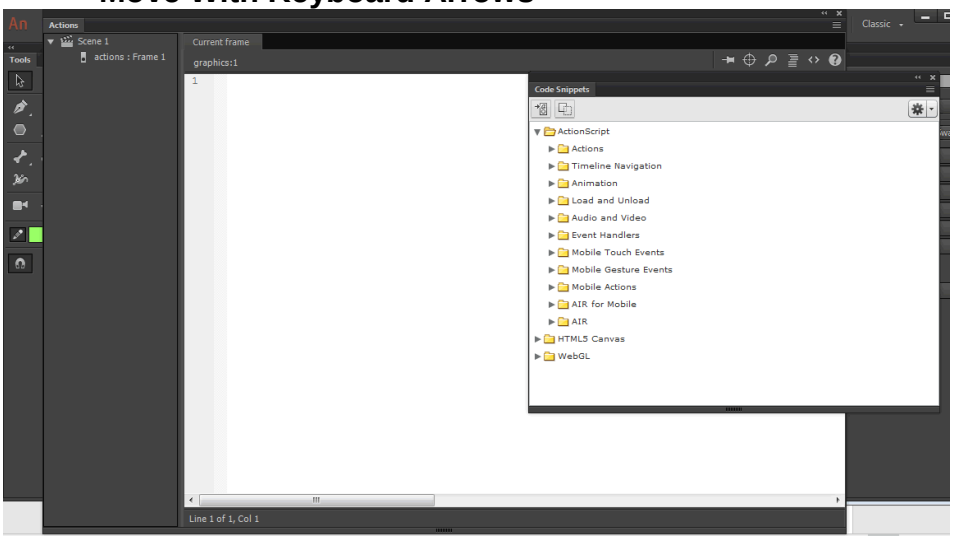
Day 3

Objectives:

- Program Character Movement with a Code Snippet.
- Adjust the code so that the correct animation is playing based on the key press.
- Program additional key presses.

Program Hero Movement with a Code Snippet:

- Select the **hero** on Frame 1 of the stage. Make sure the instance name is **hero**.
- 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:

You need to add the code that is **highlighted in red**.

```
/* Move the paddle with Keyboard Arrows */
/* Declare variables to track if a key is pressed */
/* Move with Keyboard Arrows */

var upPressed:Boolean = false;
var downPressed:Boolean = false;
var leftPressed:Boolean = false;
var rightPressed:Boolean = false;

hero.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)
	{
		hero.y -= 5;
		hero.gotoAndStop(4);
	}
	if (downPressed)
	{
		hero.y += 5;
		hero.gotoAndStop(5);
	}
	if (leftPressed)
	{
		hero.x -= 5;
```

```
                hero.gotoAndStop(3);
        }
        if (rightPressed)
        {
                hero.x += 5;
                hero.gotoAndStop(2);
        }
}

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;
                }
        }
}

function fl_UnsetKeyPressed(event:KeyboardEvent):void
{
        switch (event.keyCode)
        {
                case Keyboard.UP:
                {
                        upPressed = false;
                        hero.gotoAndStop(1);
                        break;
                }
                case Keyboard.DOWN:
                {
                        downPressed = false;
                        hero.gotoAndStop(1);
                        break;
                }
                case Keyboard.LEFT:
                {
                        leftPressed = false;
                        hero.gotoAndStop(1);
                        break;
                }
                case Keyboard.RIGHT:
                {
                        rightPressed = false;
                        hero.gotoAndStop(1);
```

```
                    break;
                }
            }
        }
```

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.