DigiPen
INSTITUTE OF TECHNOLOGY

CS 116 – Action Script Animation

Elie Abi Chahine

Introduction

- From the previous chapter you should have a hero and 2 enemies rendered. You can move the hero with the arrow keys. If the hero collides with any enemy it should get repositioned on the screen and the enemy destroyed.

- In this chapter, we are going to see how we can add and manipulate animation in a MovieClip.

Animation

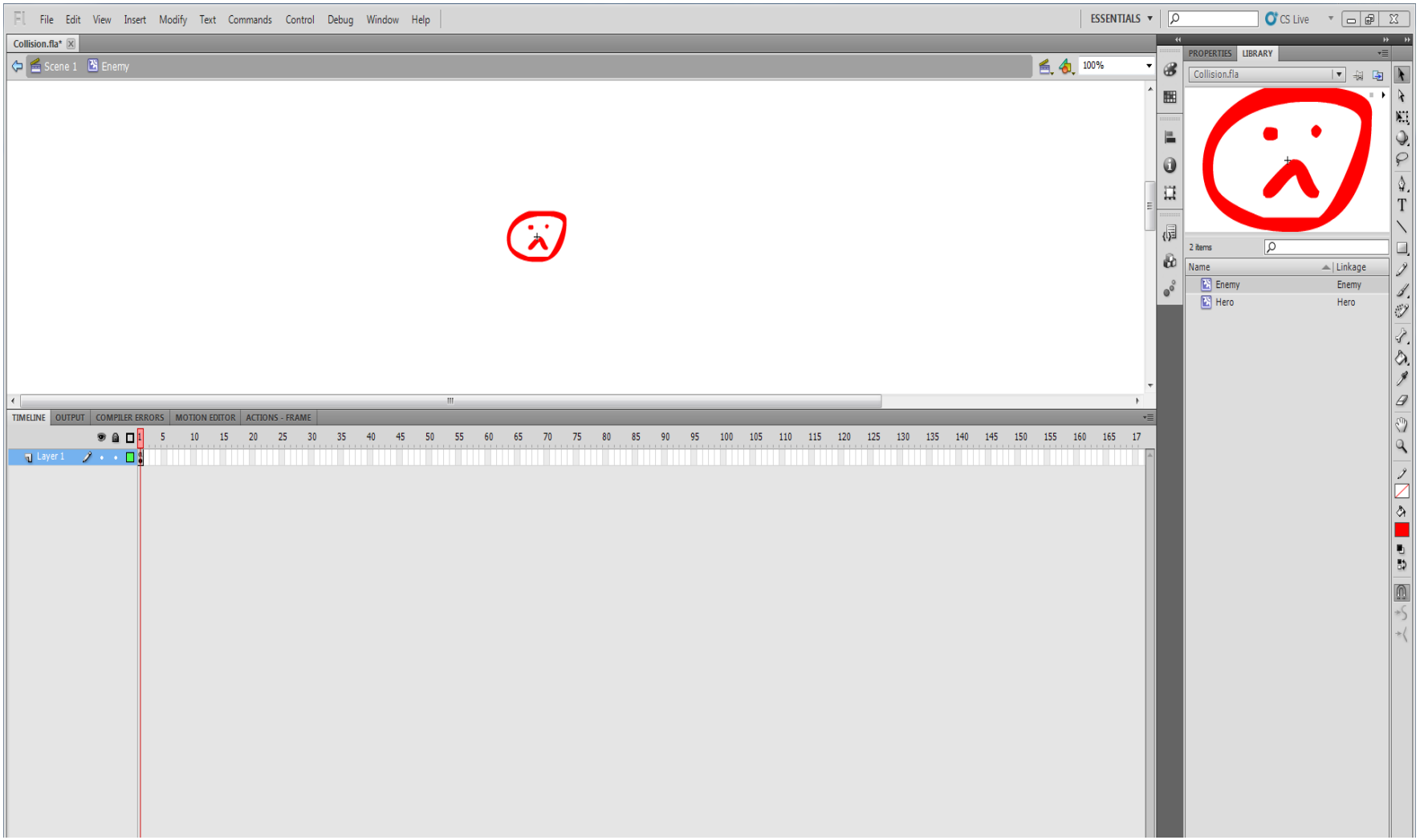
- Animation in action script is of course all based on the timeline and the key frames in that timeline.

- I'm going to show you 2 ways to do animations in action script and how to call a certain animation depending on different situations or when an event is triggered.

Animation

- Let's make our enemy animate.
- First thing go to your library, you will see the enemy symbol there.
- Double click on the enemy's image.
- At this point you should only see the enemy's drawing on the screen, instead of the stage (window). We are now in the enemy's world, and the timeline is for the enemy and not for the entire program.

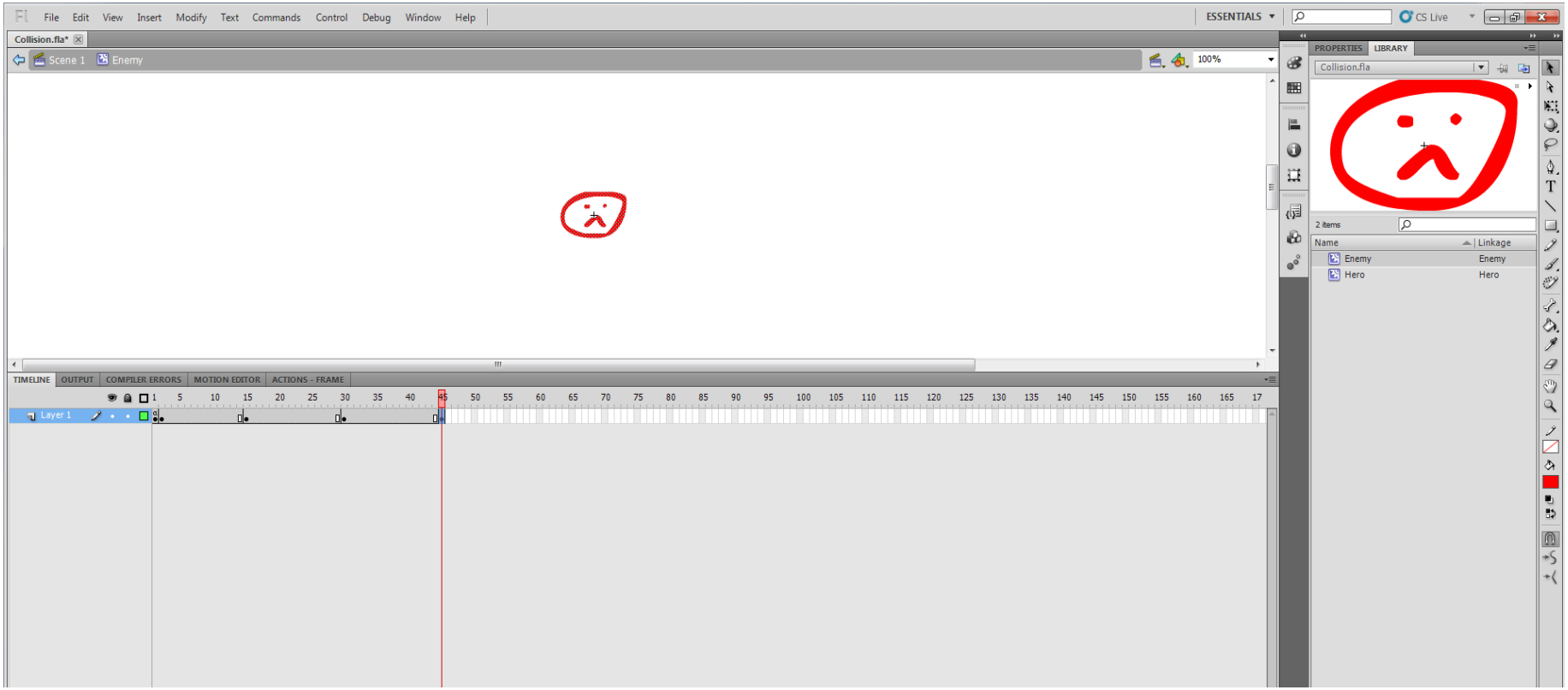
Animation



Animation

- The first frame is always used to add attributes, so starting from the second frame we can start adding animations.
- In the timeline, right click on frame 2 click on “insert Keyframe”. This will be the first frame for our animation.
- Now on frame 15, insert a new key frame, then select the enemy’s drawing and change its color using the “Paint Bucket Tool”
- All the frames between the two key frames will have the same state as the 2nd key frame.
- Do the same with frame 30 and 45… of course color them with different colors so that we see the change.

Animation



- Once you finish adding animations, click on “Scene1” button on the top left which will make you go back to the stage world.

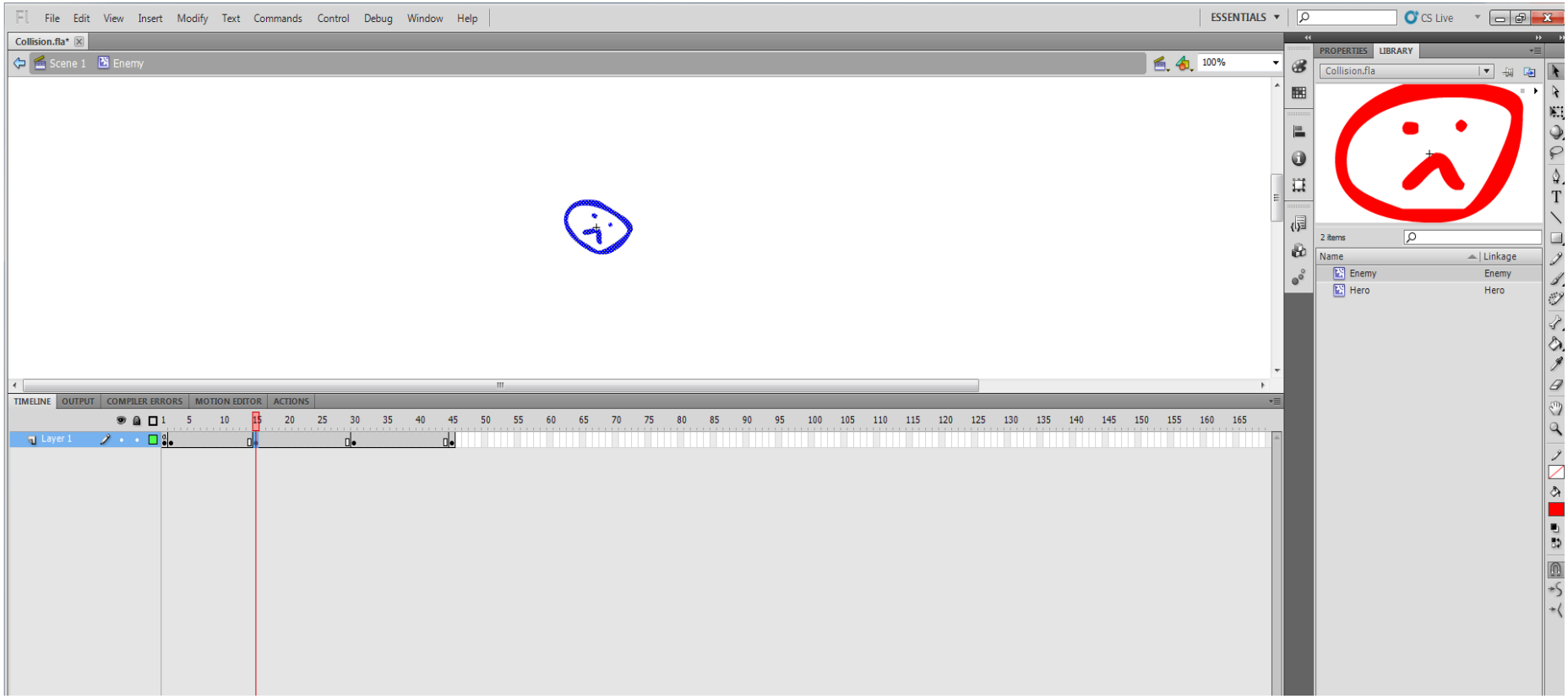
- Let’s run the code. The object should be changing colors.

Animation

- Now let's add more animations to the enemy. Let's say we want to rotate him in the animation.
- Go back to the enemy's world double clicking on his image in the library.
- On frame 15, rotate the enemy using the "Free Transform Tool"
- Do a different rotation on frame 30 …

NB: You can create multiple layers to do different animations on the object, just do what you prefer and of course what the animation requires .

Animation



- Click on “Scene1” button to go back to the stage world.
- Let’s run the code. The object should be changing colors and rotating.

Animation

- At this point, nothing was done using code and the animation is constantly looping.
- The MovieClip class contains functions and variables in order to manipulate animations (Go to a certain frame, stop the animation, play the animation, stop when reaching a certain point…)
- In the next slide I'll go more into details and will talk about those variables an functions.

NB: ***Of course, you can get those function by going to the MovieClip section in the help.***

Animation (Functions)

- play() /* will start playing the animation depending on the frame you last stopped at */
- stop() /* will stop the animation */
- gotoAndPlay(10) /* will go to frame 10 and will start the animation from there */
- gotoAndStop(10) /* will go to frame 10 and will stop there */
- nextFrame(); /* will jump to the next frame in the animation */
- prevFrame(); /* will go back to the previous frame in the animation */

Animation (Variables)

- currentFrame /* will give us the frame that we are at in the animation */

- totalFrames /* will give us the total number of frames in the MovieClip */

Although I didn't talk about Labels yet but I will include the following variables that we will use later on:

- currentFrameLabel /* tells us the label at the current frame in the timeline. If the current frame has no label, currentLabel is null. */

- currentLabel /* The current label in which we are located in the timeline. If the current frame has no label, currentLabel is set to the name of the previous frame that includes a label. If the current frame and previous frames do not include a label, currentLabel returns null. */

- currentLabels /* Gives us the array containing all labels we have in our MovieClip */

Animation

- If we run the code we see that both our enemies are animating all the time.
- We want them to animate when an event happens (like colliding with them)
- First thing is to make them stop animating. In the enemy's class, add the following code in the "Initialize" function:

stop();

- This should make the enemy get stuck at frame 1 until we say otherwise at runtime in the code (play, gotoAndPlay, gotoAndStop …)

NB: If you don't use the Initialize function after creating the enemy the animation won't stop.

Animation

- Next, we want the animation to play when we collide with the enemy.

- As a refresher:
  - We are checking for collision in the Layer's Update function that runs every frame.
  - When we collide with an enemy we are placing the hero back to it's original place and setting the enemy's bDead variable to true so that it will be removed at the end of the frame

```
function Update(e_:Event)
{
    for(var i:int=0; i<aObjects.length; ++i)
    {
        if(mcHero.hitTestObject(aObjects[i]))
        {
                mcHero.x = 150;
                mcHero.y = 200;

                aObjects[i].bDead = true
        }
    }
}
```

Animation

- Let's update the code so that we play the enemy's animation instead

```
function Update(e_:Event)
{
    for(var i:int=0; i<aObjects.length; ++i)
    {
        if(mcHero.hitTestObject(aObjects[i]))
        {
                mcHero.x = 150;
                mcHero.y = 200;

                aObjects[i].play();
        }
    }
}
```

- Now if we run the code we will have:
  - Enemies not animating
  - Once the ship hits an enemy he will start animating.

Animation

- One of the things I don't like about our Update function is that it contains a lot of code that is specific for every object we are creating!!! The object oriented approach doesn't like that !!!!

```
function Update(e_:Event)
{
    for(var i:int=0; i<aObjects.length; ++i)
    {
        if(mcHero.hitTestObject(aObjects[i]))
        {
                mcHero.x = 150;
                mcHero.y = 200;

                aObjects[i].play();
        }
    }
}
```

- We want every object to be responsible for his collision reaction…
- How about we add a collision reaction function for every object !!! ☺

Animation

- Inside the Hero Class:

```
function CollisionReaction()
{
        x = 150;
        y = 200;
}
```

- Inside the Enemy Class:

```
function CollisionReaction()
{
        play();
}
```

Animation

- The Update function will look like this:

```
function Update(e_:Event)
{
    for(var i:int=0; i<aObjects.length; ++i)
    {
        if(mcHero.hitTestObject(aObjects[i]))
        {
                mcHero.CollisionReaction();
                aObjects[i].CollisionReaction();
        }
    }
}
```

- This is better but still has some problems!!!
- What if I have more than a Hero and an Enemy… What if I have a bullet object and the enemy reacts differently when a bullet collides with it !!!!
- We need to tell the object with which object he is colliding with…. We need to have a specific ID for every object

Animation

- Let's add an integer variable called iID for every object we create

```
var iID:int;
```

- Initialize this variable inside the object's **"Initialize"** function

```
iID = 0;     /* Inside the Hero class */

iID = 1;     /* Inside the Enemy class */
```

- Change the Collision reaction function to accept an integer as parameter (so that we can tell it with which object we collided with)

```
function CollisionReaction(iID_:int)
```

Animation

- The Hero's collision reaction will become:

```
function CollisionReaction(iID_:int)
{
	switch(iID_)
	{
		case 1:	/* Checking if collided with the Enemy */
		{
			x = 150;
			y = 200;
		}
		break;

		default:
		{
			trace("Wrong ID sent!!!");
		}
		break;
	}
}
```

Animation

- The Enemy's collision reaction will become:

```
function CollisionReaction(iID_:int)
{
	switch(iID_)
	{
		case 0:	/* Checking if collided with the Hero */
		{
			play();
		}
		break;

		default:
		{
			trace("Wrong ID sent!!!");
		}
		break;
	}
}
```

Animation

- The Update function will become:

```
function Update(e_:Event)
{
    for(var i:int=0; i<aObjects.length; ++i)
    {
        if(mcHero.hitTestObject(aObjects[i]))
        {
                mcHero.CollisionReaction(aObjects[i].iID);
                aObjects[i].CollisionReaction(mcHero.iID);
        }
    }
}
```

Animation

- One of the problems we still have is that once the enemy start animating he will never stop!!!

- To solve this problem, the enemy must check if he reached the last frame in the animation and make the animation stop and go back to the first frame.

- So once we collide with the enemy, we will add an event listener that will do the above check on every frame.

Animation

```
function CollisionReaction(iID_:int)
{
    switch(iID_)
    {
        case 0:
        {
                play();
                addEventListener(Event.ENTER_FRAME, Die);
        }
        break;
        .
        .
    }
}

function Die(e_:Event)
{
    if(currentFrame == totalFrames)
    {
         removeEventListener(Event.ENTER_FRAME, Die);
        gotoAndStop(1);
    }
}
```

Animation

- So far, the code will run the animation at collision then go back to the first frame and stop there.

- In games, we usually want the animation to play then totally delete the enemy.

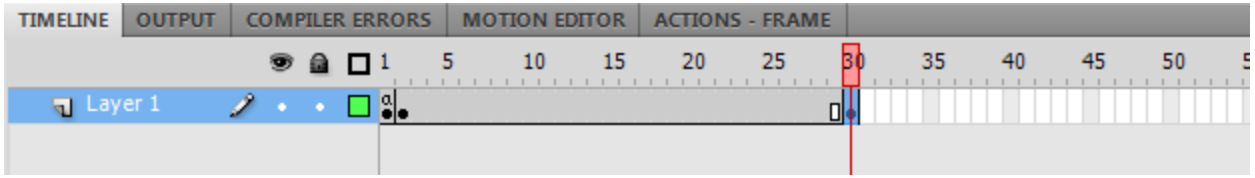
```
function Die(e_:Event)
{
        if(currentFrame == totalFrames)
        {
                removeEventListener(Event.ENTER_FRAME, Die);
                bDead = true;
        }
}
```

Shape Tween Animation

- Now let's do another type of animation, but this time for the MainCharacter.
- The nice thing about a tween animation is that you can specify two key frames and flash will interpolate between them.
- The only problem is that you don't have total control on the animation, it is the algorithm that is running the show.
- With practice though you will start knowing what is going to happen and tweak in it.

Shape Tween Animation

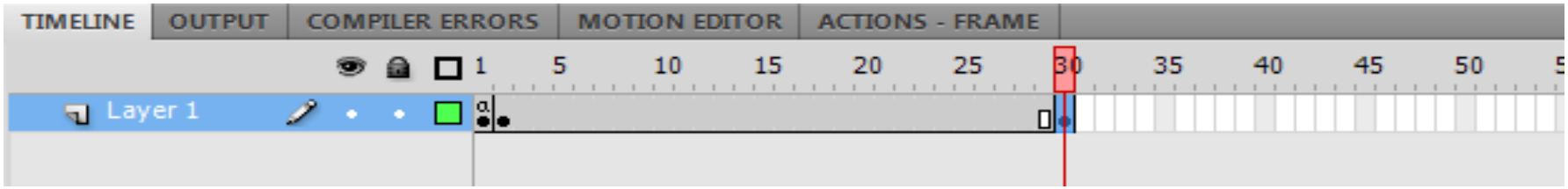
- Go to the Library and double click on the hero's class
- Now we are in the main character's world.
- Go to his Timeline, right click on frame 2, and insert a key frame
- Go to frame 30 and also insert a key frame



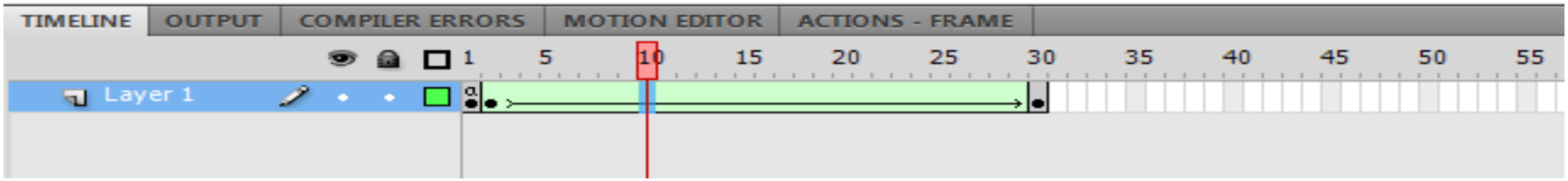
NB: The only difference between a key frame and a frame is that the frame will take the shape of the closest key frame before it, but the key frame's shape can be changed.

Shape Tween Animation

•You should have something like this.



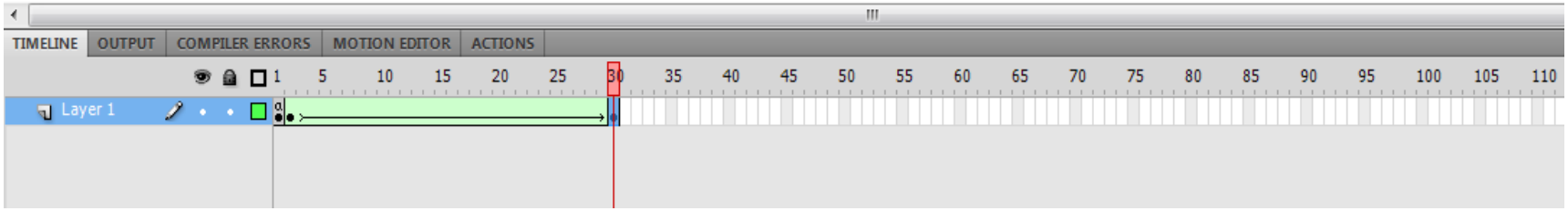
• Now right click on any frame between frame 2 and frame 30 and select create shape tween, which will give you the following:



• You just created a shape tween animation between those 2 key frames

Shape Tween Animation

- Click on frame 30, you will see that the object is selected.
- Change his shape to the final explosion shape you want.



Shape Tween Animation

- If you run the code right now, you will see that the ship is constantly playing it's explosion animation.

- Go to the hero's "Initialize" function and call the stop() function.

- For you to do:

  - When the ship collides with an enemy play the explosion animation
  - When reaching the total frames, make the ship go to (150,200) and the animation go to the frame 1 and stop.

The End ☺