DigiPen
INSTITUTE OF TECHNOLOGY

CS 116 – Action Script Collision

Elie Abi Chahine

Template Project

- In this chapter, we are going to see how we check for collision.
- But first as a recap, let's create two classes, Hero(MainCharacter) and Enemy, linked to MovieClip.
- Let's make our hero move when we press UP, DOWN, LEFT or RIGHT
- If we run the code, we should have the hero and an enemy rendered and the user can control the hero.

Collision

- The collision function given in ActionScript's Library is called "hitTestObject".
- Every DisplayObject has it in its class, and since the MovieClip class extends in a way from DisplayObject that means it can access it.
- The "hitTestObject" is a bounding box collision check function.
- You need to send, as parameter, the object you want to test with.

hitTestObject(ObjectVariable) /* Returns true if it collides, false if it doesn't */

Collision

- We need to check collision every frame.
- So let's add an event listener to the stage that will run an Update function on every frame.

```
stage.addEventListener(Event.ENTER_FRAME, Update);
```

- Inside the Update function the hero will check for collision against the enemy

```
function Update(e_:Event)
{
   if( mcHero.hitTestObject(mcEnemy1) )
   {
      mcHero.x = 150;
      mcHero.y = 200;
   }

}
```

Collision

So at this point, we have a function called Update that is being called every frame (game loop). Inside it, the hero instance is checking collision with an enemy instance.

What if I want to check collision with more than one object???

Let's add another enemy to the stage. Position it at (350 , 300).

How would you change the Update function???

Collision

We can change our Update function as follows:

```
function Update(e_:Event)
{
   if( mcHero.hitTestObject(mcEnemy1) ||
   mcHero.hitTestObject(mcEnemy2) )
   {
      mcHero.x = 150;
      mcHero.y = 200;
   }
}
```

- **Can you see the problem ????!!!!**

  What if I have 100 enemies? What if they are all extended from MovieClip but different types of enemies or objects ?

  Can you think of a solution???

Collision

- Let's create an Array that will hold all the enemy objects then use it to check collision with.

```
var aObjects:Array = new Array();
```

Now we will create the enemies inside the Array. The new code will look like this:

```
aObjects[0] = new Enemy();
aObjects[0].Initialize(350,200);
stage.addChild(aObjects[0]);

aObjects[1] = new Enemy();
aObjects[1].Initialize(350,300);
stage.addChild(aObjects[1]);
```

Collision

The Update function should look like this:

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

Collision

- The last thing we want to do is remove the enemy when we collide with it.
- To remove the enemy from the screen we should simply remove it from the stage's children list.
- In order to do so we will use the ***"removeChild"*** function found in the stage
- Add the following code when the hero collides with an enemy

```
stage.removeChild(aObjects[i]);
```

Note: ***aObjects[i] represents the object that we collided with***

Collision

- Now try running the code again.
- The enemy will be removed but one slight problem occurs. Take the character to the same position where the enemy was positioned and you will see that even though he is not rendered, he still exists at that position.
- So removing him from the stage will only prevent him from being rendered.
- In order to totally remove the enemy, we need to delete it (it's like killing the instance and freeing the memory that was allocated for it) then splice the empty slot inside the array.

```
aObjects[i] = null;
aObjects.splice(i,1);
```

Collision

- Our Update function will become like this:

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

            stage.removeChild(aObjects[i]);
            aObjects[i] = null;
            aObjects.splice(i,1);
      }
   }
}
```

Collision

Now let's clean this up and make it the more professional way…

Every Object should have a Boolean variable that represents if he is Dead or Alive

Add a Boolean inside the Hero and the Enemy

var bDead:Boolean;

Make that Boolean equal to false in the Initialize function for both classes

bDead = false;

Now when you want to kill that object you will make that Boolean equal to true and we will let some code clean it up at the end of every frame.

Collision

- Our Update function will become like this:

```
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;
      }
   }
}
```

Collision

- Let's add the function that will remove all the dead objects
- First thing is to add an event listener that will run a function called RemoveDeadObjects at EXIT_FRAME (inside our InitializeGame function)

```
stage.addEventListener(Event.EXIT_FRAME, RemoveDeadObjects);
```

The RemoveDeadObjects function will be as follows:

```
function RemoveDeadObjects(e_:Event)
{
   for(var i:int=0; i<aObjects.length; ++i)
   {
      if(aObjects[i].bDead == true)
      {
            stage.removeChild(aObjects[i]);
            aObjects[i] = null;
            aObjects.splice(i,1);
      }
   }
}
```

Collision

Now let's make the enemies rotate all the time. We need to add an event listener that runs every frame and rotates the enemy object.

```
function Update(e_:Event)
{
   rotation += 3;
}
```

Of course we need to add this event listener when we initialize the object

```
addEventListener(Event.ENTER_FRAME, Update);
```

Collision

VERY IMPORTANT NOTE:

If the object you are destroying has eventListeners, you need to remove them before removing from the stage or deleting!! **removeEventListener(…)**

But it is very hard to keep track how many event listeners we added to every object… That's why we will create a Destroy function for every class we create

```
function Destroy()
{
   removeEventListener(Event.ENTER_FRAME, Update);
}
```

Collision

The RemoveDeadObjects function will become:

```
function RemoveDeadObjects(e_:Event)
{
   for(var i:int=0; i<aObjects.length; ++i)
   {
      if(aObjects[i].bDead == true)
      {
            aObjects[i].Destroy();
            stage.removeChild(aObjects[i]);
            aObjects[i] = null;
            aObjects.splice(i,1);
      }
   }
}
```

The End ☺