**Build a Mini Golf Game with ActionScript 3.0**

*by*[George Loffelmann](http://hub.tutsplus.com/authors/george-loffelmann)*16 Sep 2009*

In this tutorial I´ll show you how to create nice Mini Golf Game with ActionScript 3.0. We´ll work with Event Handlers, Functions and also the Timer Object which is a really cool feature in AS3.

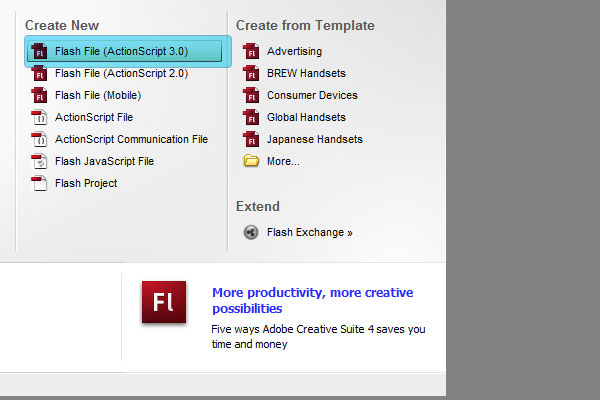
[[https://cdn.tutsplus.com/active/uploads/legacy/flashtuts/source.jpg](http://cdn.tutsplus.com/active/uploads/legacy/flashtuts/088_miniGolfGame/Preview/Preview.zip)](http://cdn.tutsplus.com/active/uploads/legacy/flashtuts/088_miniGolfGame/Preview/Preview.zip)  
 [[https://cdn.tutsplus.com/active/uploads/legacy/flashtuts/demo.jpg](http://cdn.tutsplus.com/active/uploads/legacy/flashtuts/088_miniGolfGame/Preview/preview.html)](http://cdn.tutsplus.com/active/uploads/legacy/flashtuts/088_miniGolfGame/Preview/preview.html)

**Introduction**

You'll find there's a lot of drawing involved with this project. If you'd rather not do all that yourself, just download the file[programmeGolf.fla](http://cdn.tutsplus.com/active/uploads/legacy/flashtuts/088_miniGolfGame/Preview/programmeGolf.zip) and skip to [Step 19](http://code.tutsplus.com/tutorials/flash-game-tutorial-mini-golf--active-1820#step19) where the programming part begins.

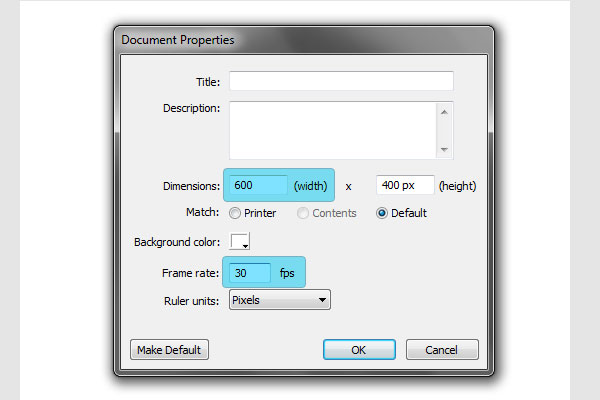
**Step 1: New File**

Let´s start by opening a new ActionScript 3.0 Flash File.



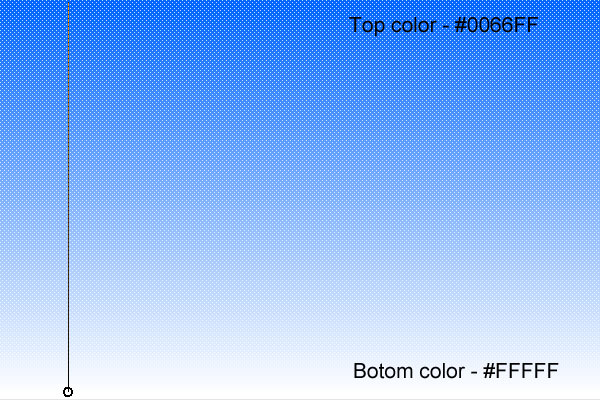
**Step 2: Document Settings**

Change the dimensions in the properties panel (600px by 400px) and set the frame rate to 30 fps.

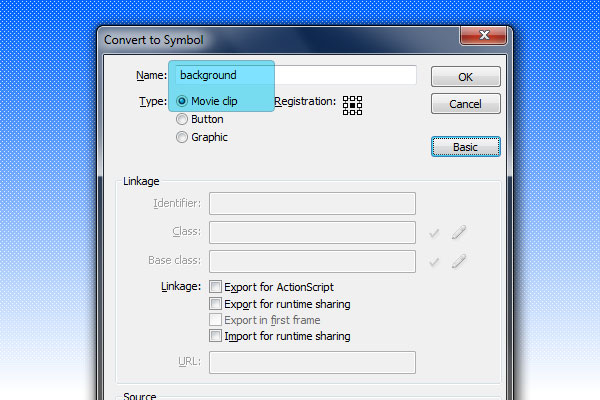


**Step 3: Creating the Background**

Now we´ll prepare some graphics. Create a rectangle (600px by 400px) and place it to coordinates 0, 0. Fill the rectangle with a gradient (#0066FF to #FFFFFF).



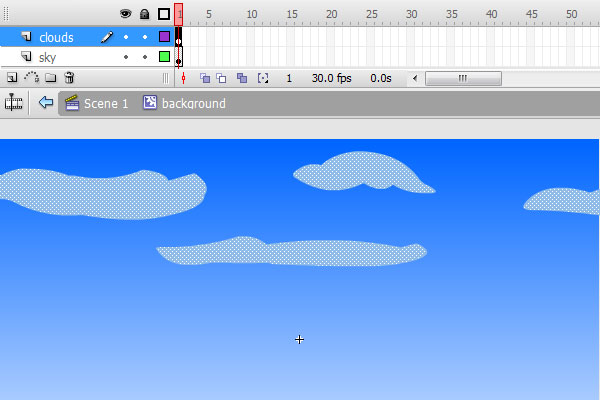
Select the object and press F8. Choose Movie Clip and name it "background".



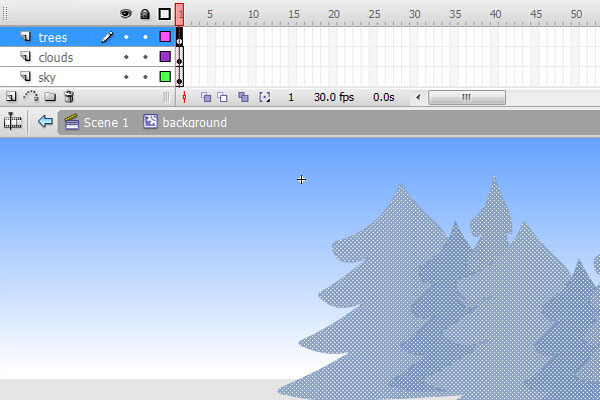
**Step 4: Adding Clouds and Trees**

Go into the background object (double-click on it) and add some graphics. I added clouds and trees by just picking the brush tool and drawing some simple shapes. Do this on a new layer to keep things organized. For the clouds I used a light blue color (#8FBBE8), for trees a light violet color (#8DA5C7). Finally I cropped the clouds and trees along my stage.

**Clouds**



**Trees**

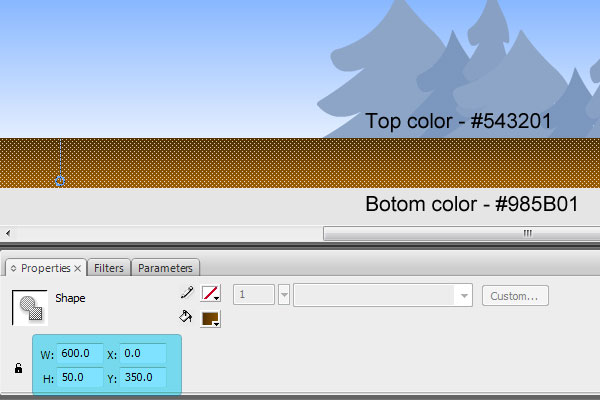


**Final result**

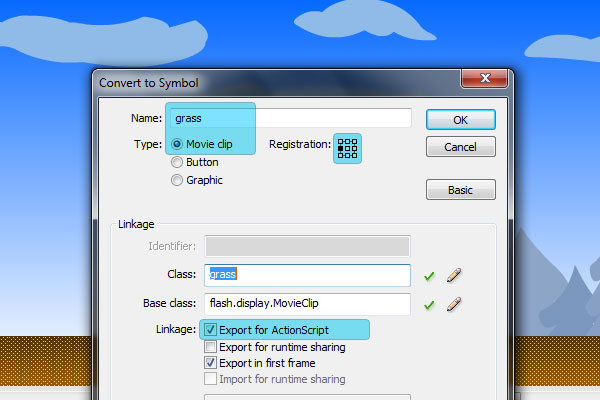


**Step 5: Ground**

On the main scene create a rectangle (600px by 50px) and place it to coordinates 0, 350. Fill the rectangle with a gradient (#543201 to #985B01).

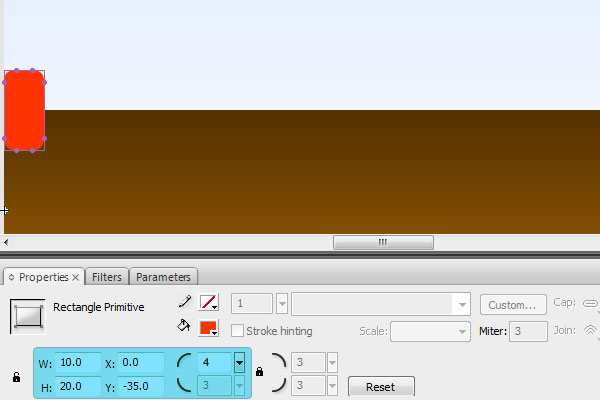


Select the object and press F8. Choose Movie Clip and name it "ground". Change its registration to the left and export object for ActionScript. Click OK.

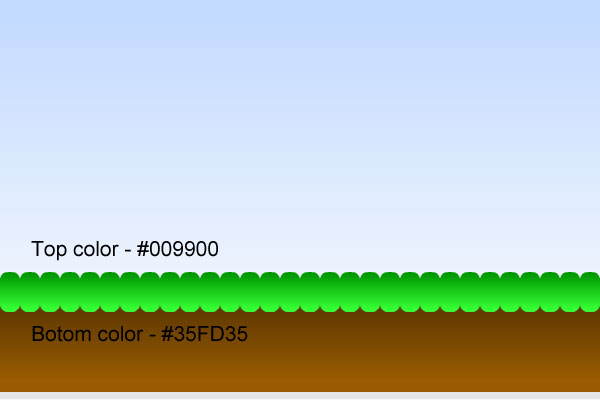


**Step 6: Grass**

Go into the ground object (double-click on it) and create a new layer. Pick the Rectangle Primitive Tool (R), draw a shape (10px by 20px) and place it at coordinates 0, -35. Set the corner radius to 4px.

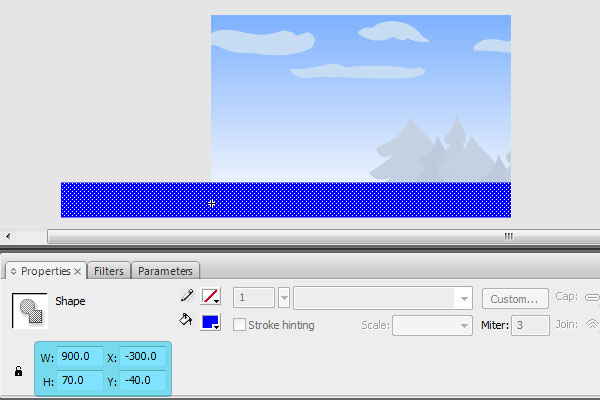


Copy these rectangles across the entire scene length. Now select all of them, make them editable (Ctrl + B) and fill them with a gradient (#009900 to #35FD35).

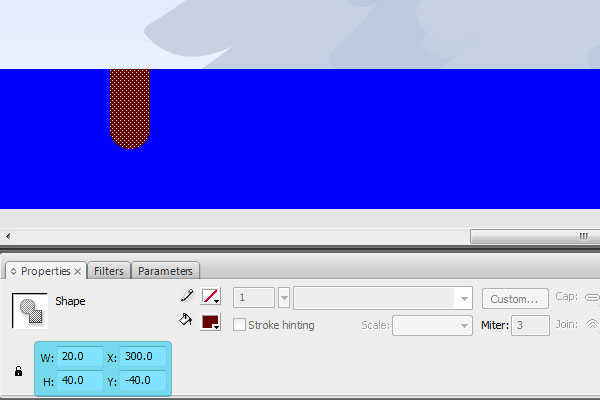


**Step 7: Hole**

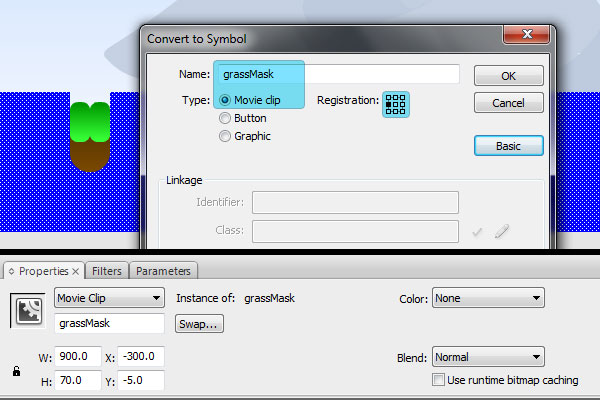
Now we'll create a shape which will mask our ground and grass. Color therefore doesn´t matter. Create a rectangle (900px by 70px) and place it at coordinates -300, -40.



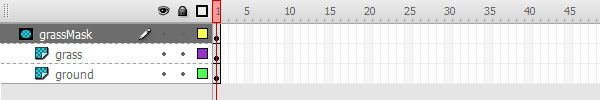
Create a "hole" shape (20px by 40px) using the Oval and the Rectangle Tool then place it at coordinates 300, -40. Now make sure that your hole shape is on the same layer with our blue rectangle created before. If so, select it and delete. You should get the final shape of the mask.



Press F8 and convert it into a Movie Clip. Change the registration point to the left and name it "grassMask". Once your Movie Clip is created, give it an instance name "grassMask".

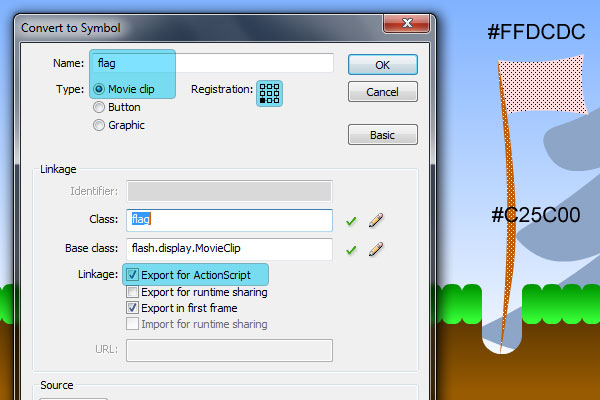


Now we should set the masking. Double-click on the layer where your ground object is and select "masked" option. Do the same with the grass layer. In the layer containing your "grassMask" object choose the "mask" option. Now you can leave the "ground" object and go back to the main stage.



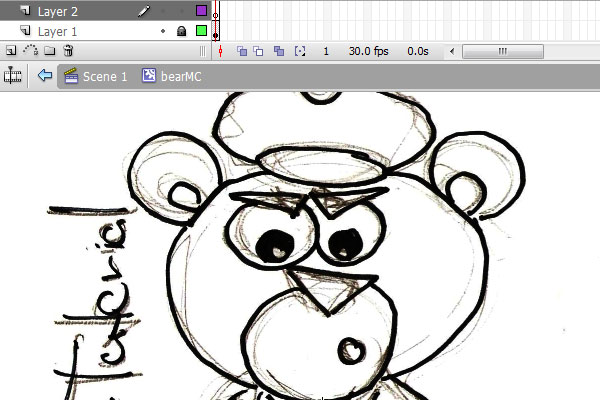
**Step 8: Flag**

Select the Brush Tool (B) and draw a stake object (#C25C00). Now pick the Rectangle Tool (R) and draw a Flag object (#FFDCDC). Select the both objects and press F8. Choose Movie Clip and name it "flag". Change the registration point and export it for ActionScript.



**Step 9: Character**

Now the worst part begins :) drawing a character. I usually draw something by hand and then redraw it in Flash, so let´s use my hand-drawn scanned picture. Download it [here](http://cdn.tutsplus.com/active/uploads/legacy/flashtuts/088_miniGolfGame/Tutorial/bear.jpg) and import it into the library (File > Import > Import to Library ...). Create a new Movie Clip (Ctrl + F8) and name it "bearMC". Drag the bear from the library onto the stage, lock the layer containing the bitmap and create a new layer. We need to separate body parts so each part will be in another layer.

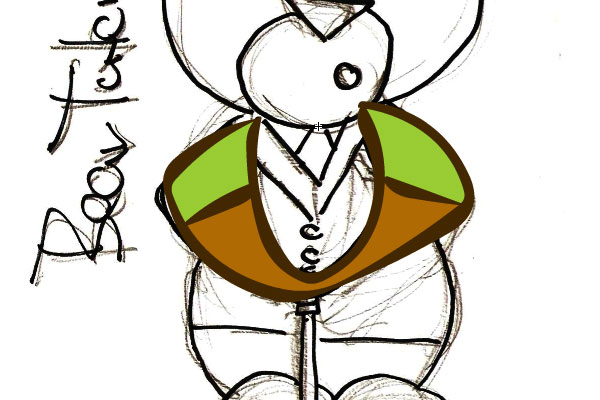


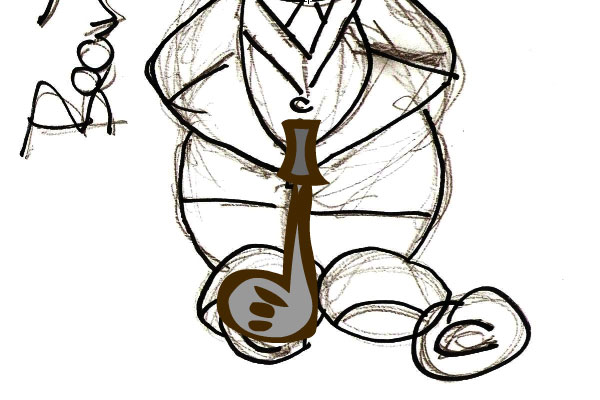
**Step 10: Body parts**

Let´s start with a head. Pick the Brush Tool (B). Select the largest circle shape and brown color (#432801). Start drawing the outlines of the head. Finally, you can pick the Paint Bucket Tool (K) and fill the head with colors. Good isn´t it? :) Do the same thing with the arms, body, legs and golf stick. Don´t forget to place every part into separate layers.



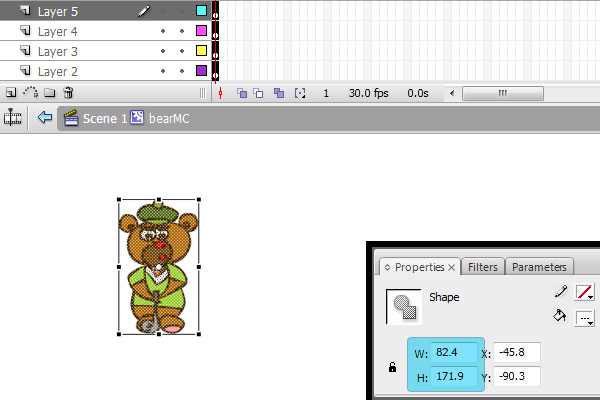






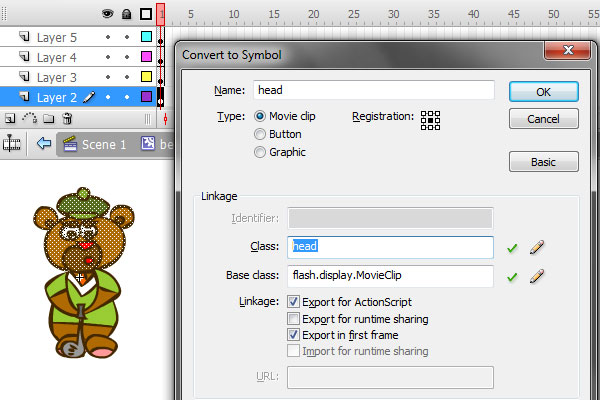
**Step 11: Adjusting Body Parts**

Now we should make some changes. First of all delete the layer with your bear bitmap (we should also delete the bear bitmap from the library). Now select the whole character throughout all layers. Because we'll load the character dynamically using ActionScript, we need to have the correct scale. Pick the Free Transform Tool (Q) and make the bear smaller (about 80px wide).



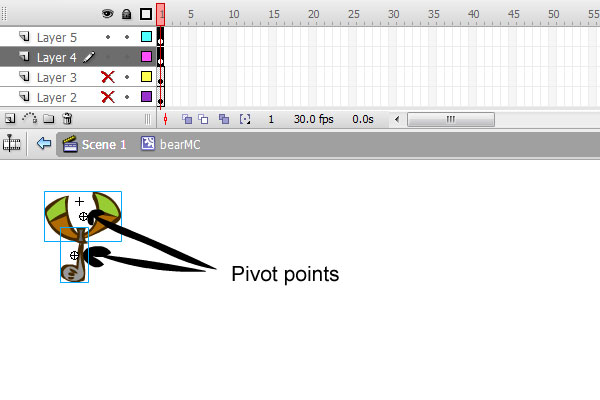
**Step 12: Converting Body Parts into Movie Clips**

Click on the first frame of your head layer. The whole head should be now selected. Press F8 and name it "head". Export it for ActionScript as well. Do exactly the same thing with body ("body"), arms ("arms") and stick ("stick").

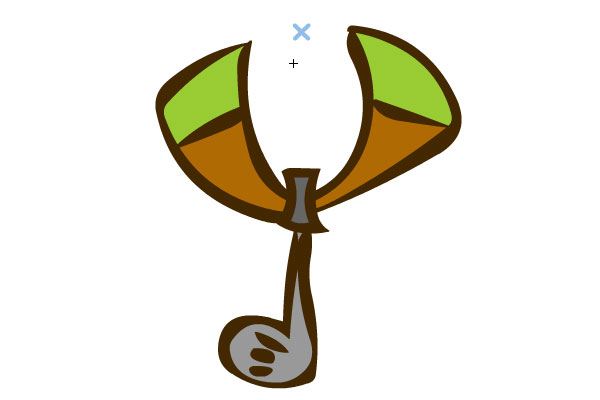


**Step 13: Adjusting Pivot Points**

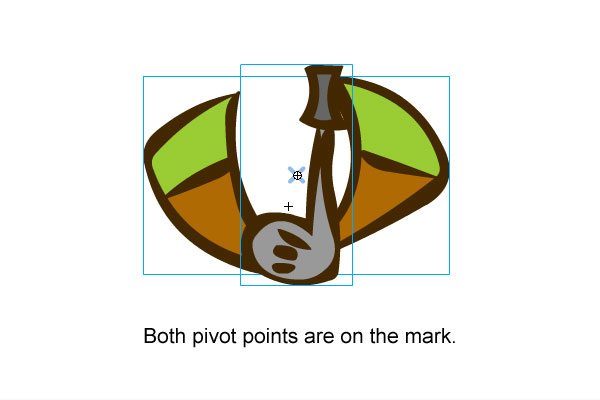
Every object in Flash rotates around a pivot point, so we have to adjust the pivot points of our stick and arms. Hide your head and body layer and select the stick and arms layer. You should see the pivot points of our objects.



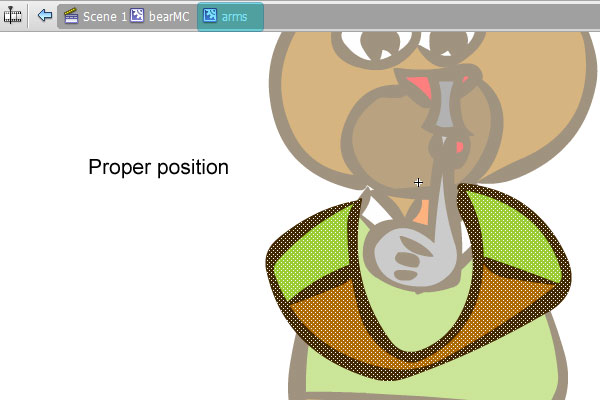
Create some sort of mark on a separate layer where the pivots should really be (between the shoulders) to allow proper rotation.



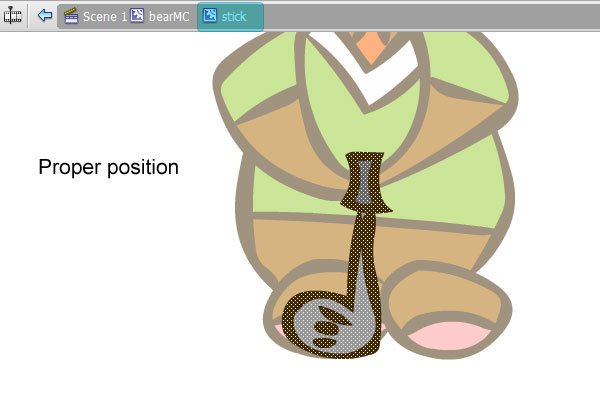
Now take one of the objects and drag it onto the mark (pivot point should be on the mark). Do the same with the second object.



Now unhide all layers and delete the mark we made before. Go into the arms object (double-click on it) so you can place the arms in the proper position.

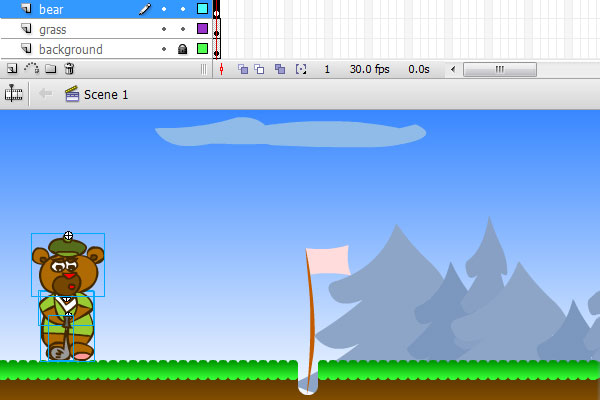


Leave the arms object and do the same thing with the stick.



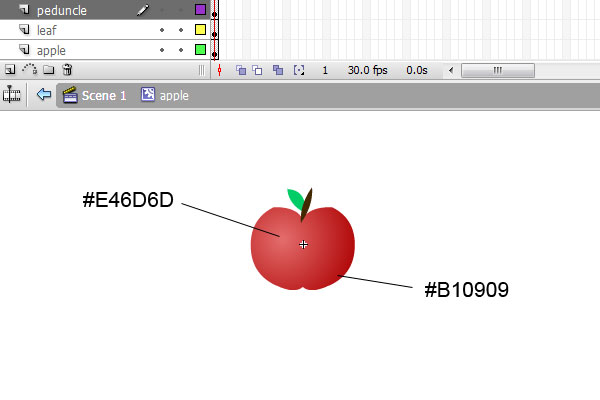
**Step 14: Positioning the Bear**

Go to the main scene and create a new layer, name it "bear". Take "bearMC" from the library and drag it onto the main stage. With our "bearMC" selected press Ctrl+B. Now you can delete "bearMC" from the library.



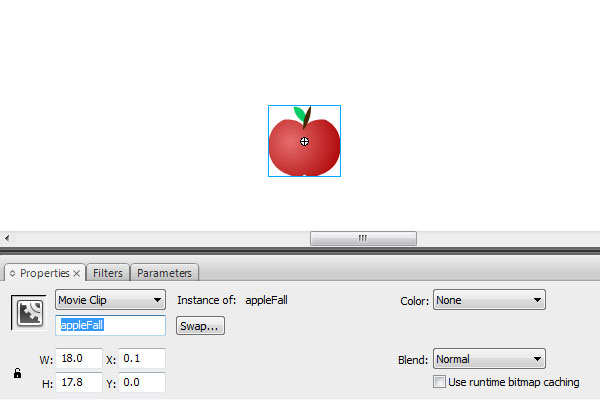
**Step 15: Apple**

Press Ctrl+F8 and create new Movie Clip. Name it "apple" and export it for ActionScript. Use the Brush Tool (B) and draw an apple shape. You can also add leaf and a stalk if you want. Now select the whole object and scale it down using the Free Transform Tool (Q), a maximum of 20px wide.



**Step 16: Apple Animation**

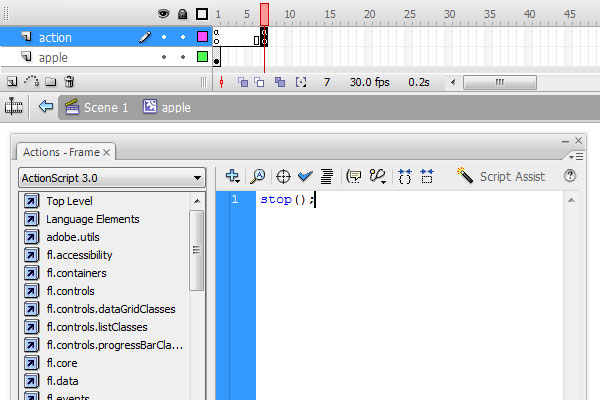
Select the whole object again and press F8. Name it "appleFall". The registration point should be in the middle. Give it an instance name "appleFall" in the properties panel.



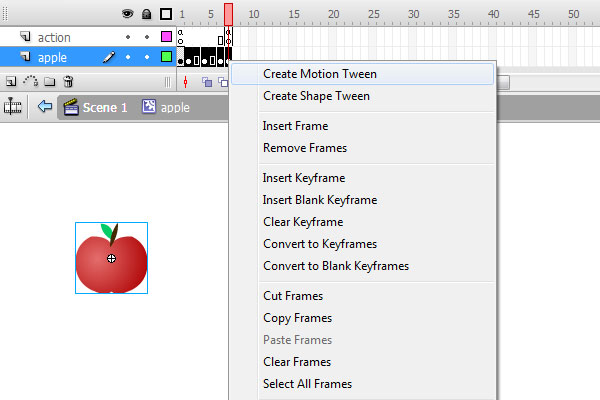
Go into the "appleFall" object (double-click on it). Create a new layer and name it "action". Go to frame 7 and create a new keyframe (F6). Then go to the actions panel (F9) and type:

|  |  |
| --- | --- |
| 1 | stop(); |

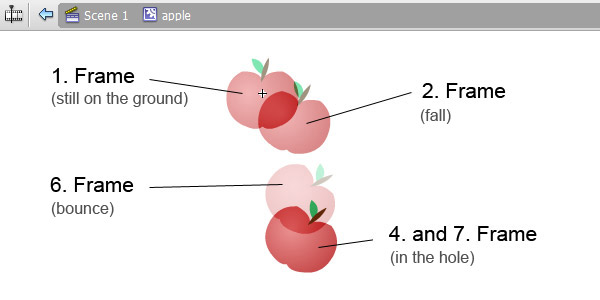
Select the first frame and do the same thing.



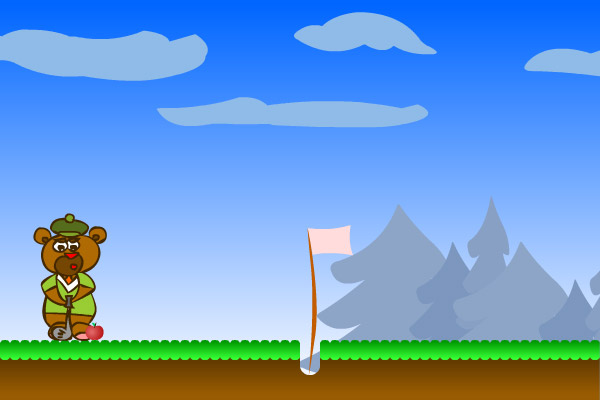
Now add keyframes (F6) into the second, fourth, sixth and seventh frames of the "apple" layer. Select the second to sixth frame and create a motion tween (Rightclick > Create Motion Tween).



Now position apples in the keyframes roughly at the positions shown on the picture below.

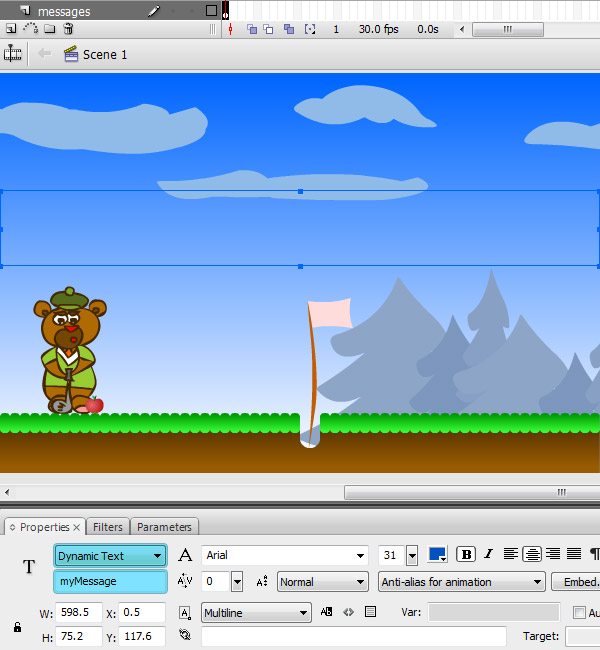


If everything is OK you can go back to the main scene and drag an "apple" object from the library to the stage.



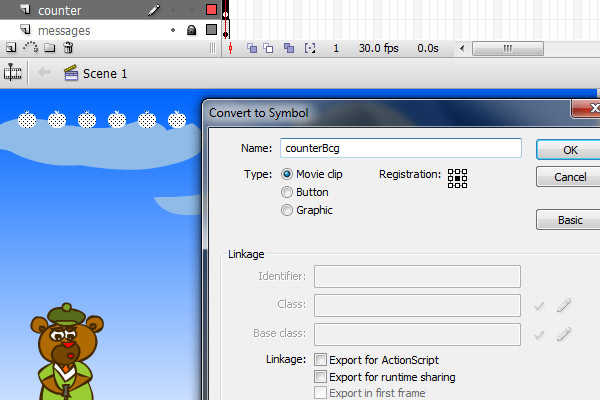
**Step 17: Text field**

Create a new layer and name it "messages". Pick a Text Tool (T) and create a new text field in the middle of the stage (I chose a blue Arial font). Set the field type to Dynamic Text and assign an instance name "myMessage". You can lock the layer now.

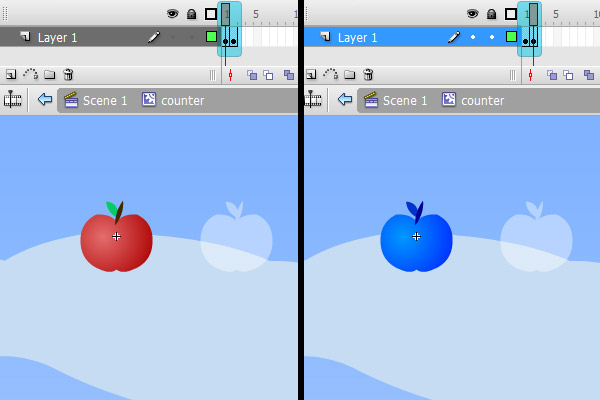


**Step 18: Score Counter**

Create a new layer and name it "counter". Again drag an apple object from the library onto the stage and position it at 30, 30. Press Ctrl + B twice (because we have two movie clips to break - apple and appleFall) and fill the object with a white color. Make five instances and place them in a line. Select all of them and press F8. Name it "counterBcg". Go to the properties panel and change the color from none to alpha at 40%.



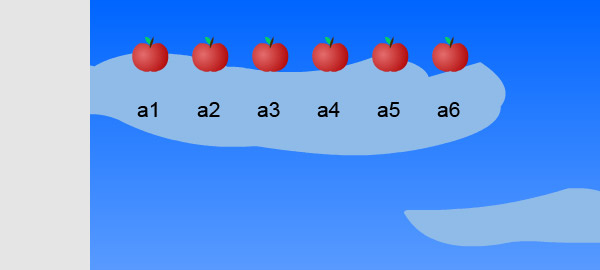
Drag an apple from the library onto the stage, position it at 30, 30 and press Ctrl + B twice again. Press F8 and name the Movie Clip "counter". Go to this object (double-click on it) and add one more keyframe. In this keyframe fill the apple with a blue color (this apple will be displayed in the counter if you miss the hole).



Go to the actions panel of every keyframe (F9) and type:

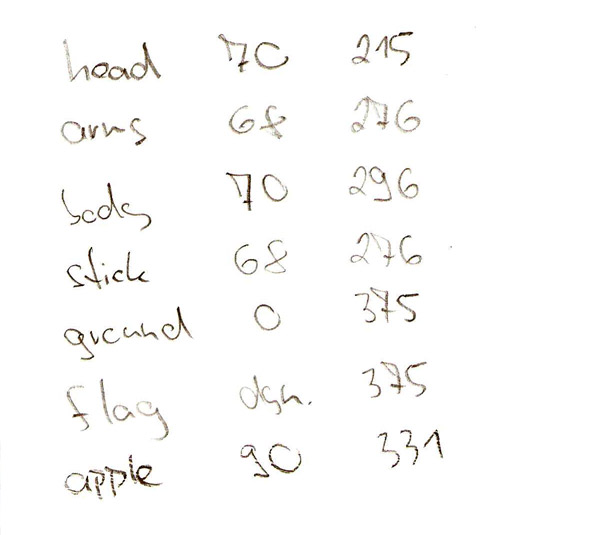
|  |  |
| --- | --- |
| 1 | stop(); |

Go to the main scene, create five instances of the apple, again forming a line. Each apple will have an instance name ("a1", "a2", "a3" etc.).

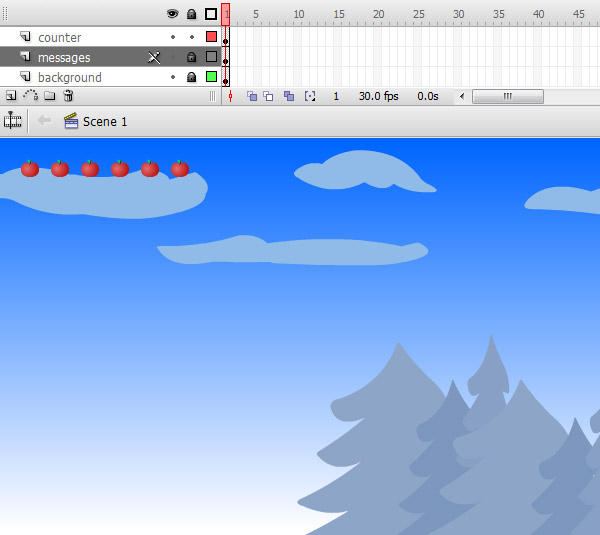


**Step 19: Preparing for Programming**

So here we are. The drawing part is over. Take a piece of paper and note the coordinates of the head, arms, body, stick, apple, ground and flag so we can attach them dynamically.



Now you can delete them from the scene.



Here's a quick break-down of how we'll approach the programming:

* [Step 20](http://code.tutsplus.com/tutorials/flash-game-tutorial-mini-golf--active-1820#step20) **Attaching and positioning objects**  
  All objects are properly attached and positioned on the scene.
* [Step 21](http://code.tutsplus.com/tutorials/flash-game-tutorial-mini-golf--active-1820#step21) **Counter**  
  Counter.
* [Step 22](http://code.tutsplus.com/tutorials/flash-game-tutorial-mini-golf--active-1820#step22) **Programming the Stick**  
  Making the stick rotate.
* [Step 23](http://code.tutsplus.com/tutorials/flash-game-tutorial-mini-golf--active-1820#step23) **Programming the Apple**  
  The apple moves when the stick hits it.
* [Step 24](http://code.tutsplus.com/tutorials/flash-game-tutorial-mini-golf--active-1820#step24) **Hit**  
  If the apple's close enough, it falls into the hole.
* [Step 25](http://code.tutsplus.com/tutorials/flash-game-tutorial-mini-golf--active-1820#step25) **Programming a Counter**  
  If it was a successful hit, the counter adds one.
* [Step 26](http://code.tutsplus.com/tutorials/flash-game-tutorial-mini-golf--active-1820#step26) **Creating Next Try Using Timer Object**  
  Once the attempt is over, we can try again after two seconds.
* [Step 27](http://code.tutsplus.com/tutorials/flash-game-tutorial-mini-golf--active-1820#step27) **Adding Text Messages**  
  We add some messages for the player.
* [Step 28](http://code.tutsplus.com/tutorials/flash-game-tutorial-mini-golf--active-1820#step28) **Game Over**  
  If the game is finished, we can play again.

**Step 20: Attaching and Positioning Objects**

All of the programming is placed in the first frame of the actions layer. So let´s create a new layer and name it "actions". Firstly, we'll add three blocks of comments to our code:

|  |  |
| --- | --- |
| 1  2  3  4  5 | //Variables    //EventListeners    //Functions |

When the comments are ready we can add following code which will attach our objects from the library to the stage.

(Additional lines 2-8, 13-23)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | //Variables  var myHead = new head();  var myArms = new arms();  var myBody = new body();  var myStick = new stick();  var myGround = new ground();  var myFlag = new flag();  var myApple = new apple();    //EventListeners    //Functions  function resetScene():void {      addChild(myHead);      addChild(myArms);      addChild(myBody);      addChild(myStick);      addChild(myGround);      addChild(myFlag);      addChild(myApple);      }    resetScene(); |

Now we have all our objects on the scene at coordinates 0, 0. Objects are attached on the scene everytime the resetScene function is called. Let´s refer back to our piece of paper where we noted the coordinates. We'll also use a random function for generating the x position of our hole and flag.

(Additional lines 9, 22-39)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | //Variables  var myHead = new head();  var myArms = new arms();  var myBody = new body();  var myStick = new stick();  var myGround = new ground();  var myFlag = new flag();  var myApple = new apple();  var grassMask:Number;    //EventListeners    //Functions  function resetScene():void {      addChild(myBody);      addChild(myArms);      addChild(myHead);      addChild(myStick);      addChild(myGround);      addChild(myFlag);      addChild(myApple);      grassMask = (320 + Math.random()\*260);      myBody.x = 70;      myBody.y = 296;      myArms.x = 68;      myArms.y = 276;      myArms.rotation = 0;      myHead.x = 70;      myHead.y = 215;      myStick.x = 68;      myStick.y = 276;      myStick.rotation = 0;      myApple.x = 90;      myApple.y = 331;      myGround.x = 0;      myGround.y = 375;      myGround.grassMask.x = grassMask - 610;      myFlag.x = grassMask;      myFlag.y = 375;      }    resetScene(); |

**\*grassMask: variable description**

1. Our stage is 600px long
2. Math.random function generates numbers from 0 to 1
3. If we multiply this value by 260 we'll get a number between 0 and 260.
4. If we then add 320 to this result, we'll get the range we're after: 320 to 580. This is the region along the x axis where we can place our hole and flag.

**Step 21: Counter**

All objects should be positioned properly and our mask and flag should be at a random x position. Add the following lines under the resetScene function. It will hide the apples for counting our score.

(Additional lines 6-15)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14 | ...    resetScene();    function invisibleApples():void {      a1.visible = false;      a2.visible = false;      a3.visible = false;      a4.visible = false;      a5.visible = false;      a6.visible = false;      }    invisibleApples(); |

**Step 22: Programming the Stick**

Now we'll add listeners for our keyboard. If we press any key, the onDown function will be called. If we then release any key, the onUp function will be called. We also need to add an enter frame listener.

The principle is very simple; if you press any key, the variable checker will contain the value 1. If you release the key, the variable checker will contain a value of 0. The Hit function, which shows up in every frame of our program (30 times per second in our case) will act along those values (affirmative or negative rotation). Variables speedUp and speedDown are just numbers which tell us how big the rotation will be. Variables minPower and maxPower tell us the minimum and maximum degree of stick rotation. Type following lines:

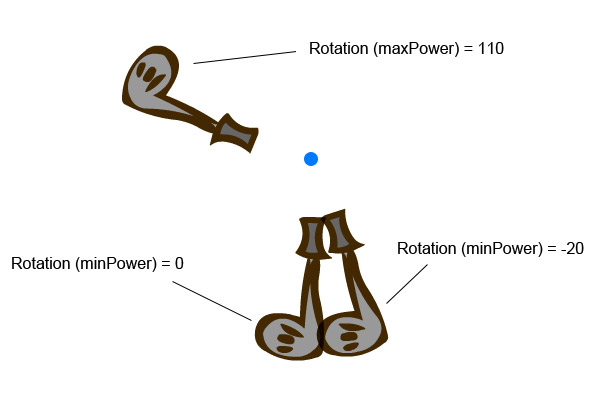
(Additional lines 12,13,18-37)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | //EventListeners  stage.addEventListener(KeyboardEvent.KEY\_DOWN, onDown);  stage.addEventListener(KeyboardEvent.KEY\_UP, onUp);  stage.addEventListener(Event.ENTER\_FRAME, hit);    //Functions            ...            myFlag.x = grassMask;      myFlag.y = 375;          checker = 0;          minPower = 0;      }    resetScene();    function onDown(e:KeyboardEvent):void {      checker = 1;      minPower = -20;      }    function onUp(e:KeyboardEvent):void {      checker = 0;      }    function hit(e:Event):void {      if((checker == 1) && (myStick.rotation < maxPower)) {          myStick.rotation += speedUp;          myArms.rotation += speedUp;          } else {              if(myStick.rotation > minPower) {                  myStick.rotation -= speedDown;                  myArms.rotation -= speedDown;                  }              }      } |

We're using a few new variables in the code, so we have to initialize them. Add the following lines to the top. Variables checker and minPower are declared in resetScene function because they'll be changed during the program and we'll need to reset them later.

(Additional lines 7-17)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | //Variables    ...    var myApple = new apple();  var grassMask:Number;  var speedUp:int = 2;  var speedDown:int = speedUp \* 8;  var checker:int;  var minPower:int;  var maxPower:int = 110; |



**Step 23: Programming an Apple**

We'll use method hitTestObject to retrieve information about the collision between our stick and apple. If a collision happens, the shot function will be called.

(Additional lines 7,8,16,30-32,36-40)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | //Variables    ...    var minPower:int;  var maxPower:int = 110;  var strength:Number;  var applePower:Number;    //Functions    ...    function onUp(e:KeyboardEvent):void {      checker = 0;      applePower = (myApple.x + myStick.rotation) \* 3.2;      }    ...    function hit(e:Event):void {      if((checker == 1) && (myStick.rotation < maxPower)) {          myStick.rotation += speedUp;          myArms.rotation += speedUp;          } else {              if(myStick.rotation > minPower) {                  myStick.rotation -= speedDown;                  myArms.rotation -= speedDown;                  }              if(myStick.hitTestObject(myApple)) {                  stage.addEventListener(Event.ENTER\_FRAME, shot);                  }              }      }    function shot(e:Event):void {      strength = (applePower - myApple.x);      myApple.x += strength / 30;      myApple.rotation += strength / 4;      } |

**Step 24: Hit**

Shooting should work. Let´s now code an apple falling into the hole. We'll be testing in our shot function whether the distance from the hole is small enough and the strength is low enough. Should this be the case, we can play our clip where an apple falls into the hole. My rules (number values) are very strict. You can define your own lighter rules if you prefer.

(Additional lines 17,27-34,37,38)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | //Variables    ...    var strength:Number;  var applePower:Number;  var distance:Number;  var target:int;    //Functions  function resetScene():void {            ...        checker = 0;      minPower = 0;      target = 0;      }    resetScene();    ...    function shot(e:Event):void {      strength = (applePower - myApple.x);      distance = grassMask - myApple.x;      if((Math.round(distance) <= 10) && (Math.round(distance) >= 3) && (strength <= 65)) {          if(target == 0) {              myApple.appleFall.play();              myApple.rotation = 0;              target = 1;              }          } else {              if(strength > 5) {                  myApple.x += strength / 30;                  myApple.rotation += strength / 4;                  }              }      } |

Here is a breakdown of how the shot function really works. It´s very simple. We'll add some lines into it later, but this is the core.

**Shot function scheme**

* Set strength variable
* Set distance variable

**If distance and strength is low *and the target hasn't yet been hit***

* Play clip with apple falling into hole
* Set apple rotation to zero
* Tell our variable that the target was hit

**If distance and strength isn't alongside our values *and if strength is still large enough - lower than 5 and the apple stops moving***

* Apple keeps moving along the x axis
* Apple keeps rotating

**Step 25: Programming the Counter**

Now we'll make a score counter. Firstly we have to initialize a new variable called tryShot; it counts the attempts we have remaining by incrementing one every time the resetScene funcion is called. If we hit the hole an apple will appear. If we miss the hole an apple will also appear, but it will jump to the second frame of the clip (displaying a blue apple).

(Additional lines 6,7,15,16,31,37-40)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | //Variables    ...    var distance:Number;  var target:int;  var tryShot:int = 0;    //Functions  function resetScene():void {            ...        minPower = 3;      target = 0;      tryShot ++;      }    resetScene();    ...    function shot(e:Event):void {      strength = (applePower - myApple.x);      distance = grassMask - myApple.x;      if((Math.round(distance) <= 10) && (Math.round(distance) >= 3) && (strength <= 65)) {          if(target == 0) {              myApple.appleFall.play();              myApple.rotation = 0;              target = 1;              this["a" + tryShot].visible = true;              }          } else {              if(strength > 5) {                  myApple.x += strength / 30;                  myApple.rotation += strength / 4;                  } else {                      this["a" + tryShot].visible = true;                      this["a" + tryShot].gotoAndStop(2);                      }              }      } |

**Step 26: Creating Next Try Using Timer Object**

Add the following lines into the code. They will ensure that the countdown function (which resets the scene and removes the listener for the shot function) will be called two seconds after any given shot is over (it doesn´t matter if the hole was or wasn´t hit). We also have to add a condition to our hit function which restricts the number of attempts to six.

(Additional lines 7,13,22,30,44,55,64,69-72)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 | //Variables    ...    var target:int;  var tryShot:int = 0;  var myTimer = new Timer(2000, 1);    //EventListeners  stage.addEventListener(KeyboardEvent.KEY\_DOWN, onDown);  stage.addEventListener(KeyboardEvent.KEY\_UP, onUp);  stage.addEventListener(Event.ENTER\_FRAME, hit);  myTimer.addEventListener(TimerEvent.TIMER, countDown);    //Functions  function resetScene():void {  ;          ...        target = 0;      tryShot ++;      myApple.appleFall.gotoAndStop(1);      }    resetScene();    ...    function hit(e:Event):void {      if(tryShot < 7) {          if((checker == 1) && (myStick.rotation < maxPower)) {              myStick.rotation += speedUp;              myArms.rotation += speedUp;              } else {                  if(myStick.rotation > minPower) {                      myStick.rotation -= speedDown;                      myArms.rotation -= speedDown;                      }                  if(myStick.hitTestObject(myApple)) {                      stage.addEventListener(Event.ENTER\_FRAME, shot);                      }                  }              }      }    function shot(e:Event):void {      strength = (applePower - myApple.x);      distance = grassMask - myApple.x;      if((Math.round(distance) <= 10) && (Math.round(distance) >= 3) && (strength <= 65)) {          if(target == 0) {              myApple.appleFall.play();              myApple.rotation = 0;              target = 1;              this["a" + tryShot].visible = true;              myTimer.start();              }          } else {              if(strength > 5) {                  myApple.x += strength / 30;                  myApple.rotation += strength / 4;                  } else {                      this["a" + tryShot].visible = true;                      this["a" + tryShot].gotoAndStop(2);                      myTimer.start();                      }              }      }    function countDown(e:TimerEvent):void {      resetScene();      stage.removeEventListener(Event.ENTER\_FRAME, shot);      } |

**Step 27: Adding Text Messages**

Now we should create some text messages, to let the player know what's going on.

(Additional lines 8,29-31,43,53)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | //Functions  function resetScene():void {            ...        tryShot ++;      myApple.appleFall.gotoAndStop(1);      myMessage.text = "Press any key!";      }    resetScene();    ...    function hit(e:Event):void {      if(tryShot < 7) {          if((checker == 1) && (myStick.rotation < maxPower)) {              myStick.rotation += speedUp;              myArms.rotation += speedUp;              } else {                  if(myStick.rotation > minPower) {                      myStick.rotation -= speedDown;                      myArms.rotation -= speedDown;                      }                  if(myStick.hitTestObject(myApple)) {                      stage.addEventListener(Event.ENTER\_FRAME, shot);                      }                  }              } else {                  myMessage.text = "Game Over";          }      }    function shot(e:Event):void {      strength = (applePower - myApple.x);      distance = grassMask - myApple.x;      if((Math.round(distance) <= 10) && (Math.round(distance) >= 3) && (strength <= 65)) {          if(target == 0) {              myApple.appleFall.play();              myApple.rotation = 0;              target = 1;              this["a" + tryShot].visible = true;              myMessage.text = "Cooool!";              myTimer.start();              }          } else {              if(strength > 5) {                  myApple.x += strength / 30;                  myApple.rotation += strength / 4;                  } else {                      this["a" + tryShot].visible = true;                      this["a" + tryShot].gotoAndStop(2);                      myMessage.text = "Bad Luck!";                      myTimer.start();                      }              }          } |

**Step 28: Game Over**

As we've built such a gorgeous game, there's a good chance that our player will want to play again :-) so let´s make that possible. If the player reaches six attempts (the block of code where we write "Game Over") we should add a final listener to our stage. It's a mouse listener and calls a function if you click anywhere.

(Additional lines 12-17,37,38,42,47)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | //Functions    ...    function invisibleApples():void {      a1.visible = false;      a2.visible = false;      a3.visible = false;      a4.visible = false;      a5.visible = false;      a6.visible = false;      a1.gotoAndStop(1);      a2.gotoAndStop(1);      a3.gotoAndStop(1);      a4.gotoAndStop(1);      a5.gotoAndStop(1);      a6.gotoAndStop(1);      }    ...    function hit(e:Event):void {      if(tryShot < 7) {          if((checker == 1) && (myStick.rotation < maxPower)) {              myStick.rotation += speedUp;              myArms.rotation += speedUp;              } else {                  if(myStick.rotation > minPower) {                      myStick.rotation -= speedDown;                      myArms.rotation -= speedDown;                      }                  if(myStick.hitTestObject(myApple)) {                      stage.addEventListener(Event.ENTER\_FRAME, shot);                      }                  }              } else {                  myMessage.text = "Game Over\nClick to play again";                  stage.addEventListener(MouseEvent.MOUSE\_DOWN, playAgain);          }      }    function playAgain(e:MouseEvent):void {      resetScene();      invisibleApples();      tryShot = 1;      stage.removeEventListener(MouseEvent.MOUSE\_DOWN, playAgain);      } |

Advertisement

**Final Code**

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049  050  051  052  053  054  055  056  057  058  059  060  061  062  063  064  065  066  067  068  069  070  071  072  073  074  075  076  077  078  079  080  081  082  083  084  085  086  087  088  089  090  091  092  093  094  095  096  097  098  099  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147 | //Variables  var myHead = new head();  var myArms = new arms();  var myBody = new body();  var myStick = new stick();  var myGround = new ground();  var myFlag = new flag();  var myApple = new apple();  var grassMask:Number;  var speedUp:int = 2;  var speedDown:int = speedUp \* 8;  var checker:int;  var minPower:int;  var maxPower:int = 110;  var strength:Number;  var applePower:Number;  var distance:Number;  var target:int;  var tryShot:int = 0;  var myTimer = new Timer(2000, 1);    //EventListeners  stage.addEventListener(KeyboardEvent.KEY\_DOWN, onDown);  stage.addEventListener(KeyboardEvent.KEY\_UP, onUp);  stage.addEventListener(Event.ENTER\_FRAME, hit);  myTimer.addEventListener(TimerEvent.TIMER, countDown);    //Functions  function resetScene():void {      addChild(myBody);      addChild(myArms);      addChild(myHead);      addChild(myStick);      addChild(myGround);      addChild(myFlag);      addChild(myApple);      grassMask = (320 + Math.random()\*260);      myBody.x = 70;      myBody.y = 296;      myArms.x = 68;      myArms.y = 276;      myArms.rotation = 0;      myHead.x = 70;      myHead.y = 215;      myStick.x = 68;      myStick.y = 276;      myStick.rotation = 0;      myApple.x = 90;      myApple.y = 331;      myGround.x = 0;      myGround.y = 375;      myGround.grassMask.x = grassMask - 610;      myFlag.x = grassMask;      myFlag.y = 375;      checker = 0;      minPower = 0;      target = 0;      tryShot ++;      myApple.appleFall.gotoAndStop(1);      myMessage.text = "Press any key!";      }    resetScene();    function invisibleApples():void {      a1.visible = false;      a2.visible = false;      a3.visible = false;      a4.visible = false;      a5.visible = false;      a6.visible = false;      a1.gotoAndStop(1);      a2.gotoAndStop(1);      a3.gotoAndStop(1);      a4.gotoAndStop(1);      a5.gotoAndStop(1);      a6.gotoAndStop(1);      }    invisibleApples();    function onDown(e:KeyboardEvent):void {      checker = 1;      minPower = -20;      }    function onUp(e:KeyboardEvent):void {      checker = 0;      applePower = (myApple.x + myStick.rotation) \* 3.2;      }    function hit(e:Event):void {      if(tryShot < 7) {          if((checker == 1) && (myStick.rotation < maxPower)) {              myStick.rotation += speedUp;              myArms.rotation += speedUp;              } else {                  if(myStick.rotation > minPower) {                      myStick.rotation -= speedDown;                      myArms.rotation -= speedDown;                      }                  if(myStick.hitTestObject(myApple)) {                      stage.addEventListener(Event.ENTER\_FRAME, shot);                      }                  }              } else {                  myMessage.text = "Game Over\nClick to play again";                  stage.addEventListener(MouseEvent.MOUSE\_DOWN, playAgain);          }      }    function playAgain(e:MouseEvent):void {      resetScene();      invisibleApples();      tryShot = 1;      stage.removeEventListener(MouseEvent.MOUSE\_DOWN, playAgain);      }    function shot(e:Event):void {      strength = (applePower - myApple.x);      distance = grassMask - myApple.x;      if((Math.round(distance) <= 10) && (Math.round(distance) >= 3) && (strength <= 65)) {          if(target == 0) {              myApple.appleFall.play();              myApple.rotation = 0;              target = 1;              this["a" + tryShot].visible = true;              myMessage.text = "Cooool!";              myTimer.start();              }          } else {              if(strength > 5) {                  myApple.x += strength / 30;                  myApple.rotation += strength / 4;                  } else {                      this["a" + tryShot].visible = true;                      this["a" + tryShot].gotoAndStop(2);                      myMessage.text = "Bad Luck!";                      myTimer.start();                      }              }          }    function countDown(e:TimerEvent):void {      resetScene();      stage.removeEventListener(Event.ENTER\_FRAME, shot);      } |

**Conclusion**

Pretty simple wasn´t it? Now you can fool around with all sorts of things; you can add other levels, terrains, you can change the character, add cheats into the game, add a game menu or intro and so on. There are unlimited possibilities, so I hope you found it useful and thanks for reading!