Creating a Vertical Shooter
Based on;
accessed Tuesday 27th July, 2010

So, we will create a game using our super hero Knight to kill dragons that are coming from one side of the screen. I have provided an FLA with all of the graphics you will need. This tutorial is more of an explanation of the code with some design work as opposed to a fully-featured tutorial covering every aspect of a vertical shooter.

You have been given the following Movie Clips, sound and picture to use

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arms</td>
<td></td>
</tr>
<tr>
<td>arrow</td>
<td></td>
</tr>
<tr>
<td>arrow01.png</td>
<td></td>
</tr>
<tr>
<td>dragon</td>
<td></td>
</tr>
<tr>
<td>legs</td>
<td></td>
</tr>
<tr>
<td>Pawel</td>
<td></td>
</tr>
<tr>
<td>Whoosh.WAV</td>
<td></td>
</tr>
</tbody>
</table>
On the timeline
1. Insert a layer, name it Actions
2. in the first frame of our timeline type the following ActionScript:

```actionscript
//---- variables ----
var steps:Number = 5;
var spriteX:Number = 265;
var spriteY:Number = 265;

//---- functions ----
function checkKeys() {
if (Key.isDown(Key.RIGHT)) {
spriteX += steps;
} else if (Key.isDown(Key.LEFT)) {
spriteX -= steps;
}

function updatePawel() {
knight._x = spriteX;
knight._y = spriteY;
}

this.onEnterFrame = function() {
checkKeys();
updatePawel();
};
```

Ok, our hero is moving from side to side on the screen but, he can even move outside of the stage! We can animate the legs and the arms so it will look better and it will add the option of throwing arrows (we don't have arrows yet).

If you double click on your knight movie clip, you should be in the timeline for that movie clip. You will see three layers corresponding to the body, arms, and legs:
More importantly, notice that the arms and legs are movie clips with the instance names...arms and legs! If you look inside the arms and legs movie clips, you will find that they have some keyframes that provide some extra animation for movement.

Let's modify our code to take advantage of the new movement in the legs and arms:

```
//---- functions ----
function checkKeys() {
if (Key.isDown(Key.RIGHT) && spriteX<510) {
    spriteX -= steps;
knight.legs.play();
} else if (Key.isDown(Key.LEFT) && spriteX>40) {
    spriteX -= steps;
knight.legs.play();
}
if (Key.isDown(Key.UP)) {
knight.arms.play();
}
```

**Explanation**
Let's go through the code you have used:

```
//---- variables ----
var steps:Number = 5;
var spriteX:Number = 265;
var spriteY:Number = 265;
```

*First we declare the variables, steps are the number of pixels our hero will move, spriteX and spriteY will help us to give value to the X and Y position of our hero.*
The first function checks if we are pressing the keys that are making our hero to do things:

```javascript
if (Key.isDown(Key.RIGHT) && spriteX<510) {
    spriteX += steps;
    knight.legs.play();
}
```

*The above code essentially says that only if we press the Right arrowKey and our hero is not outside of the stage, our hero will move to the right (remember, steps = 5 pixels) and play the legs animation.*

```javascript
else if (Key.isDown(Key.LEFT) && spriteX>40) {
    spriteX -= steps;
    knight.legs.play();
}
```

*This code is the same as above, except I now check for the Left arrow key and ensure that our hero is not outside the left edge of our stage.*

*If we press the Up arrowKey on our keyboard, the animation in our arms movieclip will play.*

---

```javascript
function updatePawel() {
    knight._x = spriteX;
    knight._y = spriteY;
}
```

*This function tells Flash that our hero will take the X and Y values of the variables spriteX and spriteY.*

---

```javascript
this.onEnterFrame = function() {
    checkKeys();
    updatePawel();
};
```

*This onEnterFrame event checks to see if we are pressing keys on the keyboard or if our hero is moving.*
Adding a Background
Now we are able to move our sprite, but our sprite is "in the air." We will fix that by adding a background. To add a background is easy as you already know. Just add another layer in the main timeline and draw your background: You can use any drawing program and paste it into Flash CS-4 or just draw a green rectangle for the grass and a blue for the sky.
Drawing the Arrow

The arrow can be drawn in Flash as well (I used Photoshop to draw mine). If you take a look at my FLA, you will see the arrow movie clip and image in the Library. Whether you create your own arrow, or use my arrow, just ensure that your arrow is inside a movie clip. The arrow should be a movieclip and we should leave it in the library, not on the stage.

Open the Library (Ctrl + L) and right click the arrow movieclip so you can open the Linkage Properties box. From the Linkage Properties dialog box, if you tick in linkage options Export for Actionscript by default the identifier will be arrow, so leave it like that.

Getting back to the Code

Let's modify the code we added in the previous page. Let's add to the script a couple of variables: one to declare the speed of our arrow and another to allow us to shoot one arrow at a time. Now, we also need a function to update our arrow.

Now the script looks like:

```actionscript
//---- variables ----
var steps:Number = 5;
var spriteX:Number = 265;
var spriteY:Number = 265;
var speed:Number = 25;
var arrowActive:Boolean = false;

//---- functions ----
function checkKeys() {
    if (Key.isDown(Key.RIGHT) && spriteX<510>40) {
        spriteX -= steps;
        knight.legs.play();
    }
    if (Key.isDown(Key.UP) && arrowActive == false) {
        knight.arms.play();
        attachMovie("arrow", "arrows", 8);
        arrows._x = spriteX;
        arrows._y = spriteY+50;
        arrowActive = true
    }
}

function updatePawel() {
    knight._x = spriteX;
    knight._y = spriteY;
}

function updateArrow() {
    if (arrowActive == true) {
```
arrows._y -= speed;
}

if (arrows._y<=10) {
    arrowActive = false;
    removeMovieClip(arrows);
}

this.onEnterFrame = function() {
    checkKeys();
    updatePawel();
    updateArrow();
};

Adding the Dragon!
Pawel shoots arrows, moves and there is a beautiful background (hopefully), but Pawel is shooting into an empty sky! We need the dragons.

Here is how my dragon looks like:

![Dragon Image]

For the sake of this tutorial, let's leave it as an image. You can make an animation of the wings/tail/etc. if you really really want to.

We don't need instances of our dragon on the stage but we need it in the Library as a movieclip. Like we did with the arrow, let's change it's Linkage Properties.

Open the Library and right click on the dragons movieclip to open the Linkage Properties window, tick Export for Actionscript, and leave the identifier as default (dragon).

Back to the ActionScript
We now need to add some more variables:

```actionscript
var dragons:Number = 3;
var i:Number = 0;
var score:Number = 0;
```
Dragons is the number of dragons we will have on the stage.

The variable $i$ is a bit difficult to explain right now, for it does several things. Hopefully with the rest of the code, you will see what it does.

Score is to store our score, we can add a dynamic text box on the stage with variable name `score` so we know how good we are killing dragons.

We need to add two new functions, one to initialize the dragons and another one to update them. Add the following code to the end of your existing code:

```javascript
function initDragons() {
    for (i; i<dragons; i++) {
        attachMovie("dragon", "dragon"+i, i);
        dragon = _root["dragon"+i];
        updateDragons(dragon);
        dragon.onEnterFrame = function() {
            if (this.hitTest(arrows)) {
                score += 5;
                trace(score);
                arrowActive = false;
                removeMovieClip(arrows);
                updateDragons(this);
            }
            if (this._x>0) {
                this._x -= this.velo;
            } else {
                updateDragons(this);
            }
        };
    }
}
initDragons();

function updateDragons(which) {
    which.gotoAndStop(random(4));
    which._x = random(100)+530;
    which._y = random(80)+20;
    which.velo = random(10)+2;
```
In the function `initDragons()`, we start with the loop `for (i; i<dragons; i++)`, usually this sort of loops look the same, with a variable to initialize the loop, the condition to loop and then increment the variable value after each loop iteration.

When the value of `i` is less than the number of dragons, we will attach an instance of our movieclip dragon with this script:
```
attachMovie("dragon", "dragon"+i, i)
```

The new name of this instance is "dragon"+i, the value of `i` is incrementing, so the first one will be called dragon0, the second dragon1 and because we set dragons number as 3, the last instance will be dragon2 finishing the loop.

To make our life easier, we will set a variable to store the name of each dragon, `dragon = _root["dragon"+i];`

To update the dragons that have been just created, we need to call the function `updateDragons(dragon);`

```
if (this.hitTest(arrows)):
    this.line checks if the arrows hit them, and if it happens, we increment the score by 5: score += 5;

arrowActive = false: sets the arrow's status to be inactive

removeMovieClip(arrows): because the arrow has already hit something, we can safely remove it. This line does that.

updateDragons(this): we will update the dragons using the soon to be explained updateDragons function.
```

```
if (this.x>0) {
    this.x -= this.velo;
} else {
    updateDragons(this);
}
```

if the dragon is on the stage, then its x position is bigger than zero. We will set it to move horizontally (x axis), but if the x position is not greater then zero, then we send that dragon to our updateDragons function.

The update Dragons function is pretty straightforward. It is run whenever a dragon has been hit with an arrow or if it moves outside of the stage. We simply try to simulate a new dragon by specifying a new vertical position, horizontal position, and speed.
Your final code should look like the following:

```javascript
//---- variables ----
var steps: Number = 5;
var spriteX: Number = 265;
var spriteY: Number = 265;
var speed: Number = 25;
var arrowActive: Boolean = false;
var dragons: Number = 3;
var i: Number = 0;
var score: Number = 0;
//---- properties ----
knight.swapDepths(10);
//---- functions ----

function checkKeys() {
    if (Key.isDown(Key.RIGHT) && spriteX < 510) {
        spriteX += steps;
        knight.legs.play();
    } else if (Key.isDown(Key.LEFT) && spriteX > 40) {
        spriteX -= steps;
        knight.legs.play();
    }

    if (Key.isDown(Key.UP) && arrowActive == false) {
        knight.arms.play();
        attachMovie("arrow", "arrows", 8);
        arrows._x = spriteX;
        arrows._y = spriteY + 50;
        //arrowActive = true
    }
}

function updatePawel() {
    knight._x = spriteX;
    knight._y = spriteY;
}

function updateArrow() {
    if (arrowActive == true) {
        arrows._y -= speed;
    }
}
```
if (arrows._y <= -10) {
    arrowActive = false;
    removeMovieClip(arrows);
}

function initDragons() {
    for (i; i < dragons; i++) {
        attachMovie("dragon", "dragon"+i, i);
        dragon = _root["dragon"+i];
        updateDragons(dragon);
        dragon.onEnterFrame = function() {
            if (this.hitTest(arrows)) {
                score += 5;
                trace(score);
                arrowActive = false;
                removeMovieClip(arrows);
                updateDragons(this);
            }
            if (this._x > 0) {
                this._x -= this.velo;
            } else {
                updateDragons(this);
            }
        }
    }
}
initDragons();

function updateDragons(which) {
    which._x = random(100)+530;
    which._y = random(80)+20;
    which.velo = random(10)+2;
}

this.onEnterFrame = function() {
    checkKeys();
updatePawel();
updateArrow();
};