Notes 13: Are we there yet? Are we there yet? Are we there yet?

Any of you ever go on a long family car trip. These were really great in the summer before air-conditioning. Did you enjoy of harassing your parents with frequent queries of “Are we there yet?” Some day you may have kids of your own and experience this pleasure from the other side. In the meantime, we are doing exactly the same thing with the way we are managing the keyboard so far in flash. Every single time we enter a frame we check to see if a key is pressed. This is the equivalent of every mile asking the driver “are we there yet?” A much better approach is to say, “Mom/Dad, please tell me when we get there”, and then just sit back and do something more productive like intimidate a sibling. Likewise, for actionscript, how about if we just tell the keyboard class to let us know when a key is pressed/released rather than checking every freaking frame! We do this with something called a listener. A listener is just a special event handling function that listens for an action to occur and then handle it when it does. It is another example of “event driven programming”, where the event is the keyboard event.

Check out the following code:

```actionscript
_root.onKeyDown = function() {
    trace("you pressed a key")
}

Key.addListener(_root) ;

var count:Number = 0 ;
onEnterFrame = function () {
    trace("Entered frame " + count) ;
    count++ ;
}
```

Run the code with ctrl-enter to get the trace output. While it is running hit keyboard keys. The code adds a function to the _root MovieClip, namely “onKeyDown”. Then, it uses the Key.addListener method to identify _root as an object that has an event handler defined for Keys. Only two event handling methods can be added to the listener: onKeyDown() and onKeyUp(). In this case the _root.onKeyDown() handler just uses trace print out that a key has been pressed.

There are at least two reasons that this is preferable to checking the keys on every frame entry:

1. It reduces code overhead. This is more efficient than running code every frame.
2. When flash can not keep up with the playback of an animation, it drops frames to try to keep up. If a keyboard click happened during a dropped frame it will be lost forever, with this approach flash WILL eventually process the key click.

Often a special object is created to handle keyboard (or mouse) events instead of adding the handlers to _root. Much of the AS code you find on the web will use this approach. The exact same thing as above is done with a keyboard-handling object in:

**a_listenerKeyboard_2.fla**

```actionscript
keyboardHandler = new Object() ;
keyboardHandler.onKeyDown = function() {
    trace("you pressed a key") ;
}

Key.addListener(keyboardHandler) ;

var count:Number = 0 ;
onEnterFrame = function () {
    trace("Entered frame " + count) ;
    count++ ;
}
```

This code does the same thing as the first code but creates a separate object (keyboardHandler) to take care of the key stuff.

Want to see WHICH key was pressed? Compare its code (see Key in help menu). If you run the following (by using ctrl-enter) you will get messages if you hit “a”, “b”, “c”, or “d”, versus the other keys:

**a_listenerKeyboard_3.fla**

```actionscript
keyboardHandler = new Object() ;
keyboardHandler.onKeyDown = function() {
    trace("you pressed a key:")
    if (Key.getCode() == 65) trace("you pressed \"a\" ") ;
    if (Key.getCode() == 66) trace("you pressed \"b\" ") ;
    if (Key.getCode() == 67) trace("you pressed \"c\" ") ;
    if (Key.getCode() == 68) trace("you pressed \"d\" ") ;
}

Key.addListener(keyboardHandler) ;
```
var count:Number = 0;
onEnterFrame = function () {
    trace("Entered frame " + count);
    count++;
}

Of course, you can put calls to manipulate other stuff, like the location of MovieClips, inside the keyboard handling code:

**a_listenerKeyboard_4.fla**

```actionscript
var tempBall:MovieClip = attachMovie("ball", "b1", 1);
var Increment:Number = 5;

tempBall._x = Stage.width / 2;
tempBall._y = Stage.height / 2;

keyboardHandler = new Object();
keyboardHandler.onKeyDown = function() {
    trace("you pressed a key:");
    if (Key.getCode() == Key.LEFT) tempBall._x -= Increment;
    if (Key.getCode() == Key.RIGHT) tempBall._x += Increment;
    if (Key.getCode() == Key.UP) tempBall._y -= Increment;
    if (Key.getCode() == Key.DOWN) tempBall._y += Increment;
}

Key.addListener(keyboardHandler);
```

Publish (F12) this code and run it. Remember, you need to click with the mouse to get focus before the keys will work.

Do you notice anything different about the above code? Yep, NO onEnterFrame() method! That is because the listener is all that is needed to respond to the keys. I could have other stuff in an onEnterFrame() function, but the keyboard handling is independent of such a function.

This use of a listener is also how the mouse works. Check out the following code:

**a_listenerMouse_1.fla**

```actionscript
mouseHandler = new Object();
```
mouseHandler.onMouseMove = function() {
    trace("mouse moved: x,y = " + _xmouse + " + " + _ymouse);
}

Mouse.addListener(mouseHandler);

Because of the trace command, use ctrl-Enter to view the above. Run it and move the mouse around. Like the keyboard handler code, we define an object to handle mouse events. In this case we just trace out the locations of the mouse. Note, since we are viewing by using ctrl-enter the boundaries are all goofed up. A more accurate way to view this program is to use a dynamic text box for the locations and publish (F12) to view the program in execution. Load and publish the following:

**a_listenerMouse_2.fla**

```
mouseHandler = new Object();
mouseHandler.onMouseMove = function() {
    outString = "( " + _xmouse + " , " + _ymouse + " )";
}

Mouse.addListener(mouseHandler);
```

The Mouse() class allows you the following events to be handled by the listener: onMouseMove(), onMouseDown(), onMouseUp(). Given this, we can modify our program above to print out the locations at which the user clicks. Publish and run the following .fla file. To run it, just click at places on the screen.

**a_listenerMouse_3.fla**

```
mouseHandler = new Object();
mouseHandler.onMouseDown = function() {
    outString = _xmouse + "," + _ymouse;
}

Mouse.addListener(mouseHandler);
```

You now have enough tools to make much more interesting flash animations and games. Say you want to make a simple line drawing program. Run this code by publishing (F12) and then moving and clicking the mouse on the screen:

**a_listenerMouse_4.fla**

```
var drawMode:Boolean = true;
```
The above code is just a very simple drawing program, you could add buttons (or buttons embedded inside of what looks like a toolbar) to start and stop the drawing of lines. You could add a rectangle tool, etc. Note, lineTo() and moveTo() are MovieClip methods, which since no MovieClip is specified are applied to the _root clip.