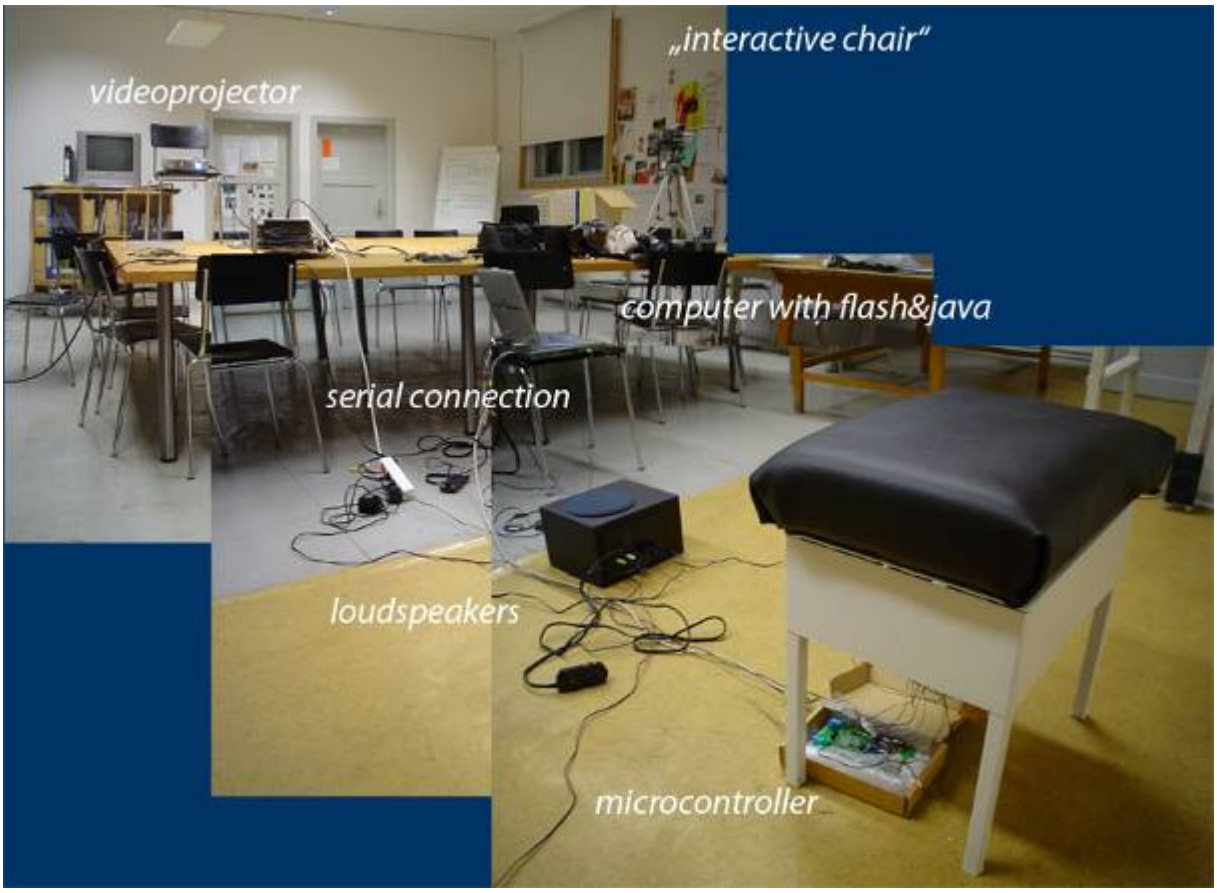
realisiert von Ralph Kistler, im Fachmodul
„Grundlagen der Interaktionselektronik“ von Dipl.-Ing. Andreas Niemand,
Interaktive Medien, Professur Dr. Jens Geelhaar



„Elektrischer Stuhl“

Idee:

Je nachdem, wie man sich auf einem Stuhl setzt, werden Bilder auf einer Projektionsfläche zu sehen und verschiedene Geräusche zu hören sein, die zusammen ein schwerverständliches Gemurmel ergeben können. Der Betrachter sitzt unmittelbar vor dieser projektierten Collage aus diversen Bildern und ist im ersten Moment nicht imstande, aus dieser Bilder- und Tonflut eine vernünftige Information zu erkennen. Entweder er gibt sich diesem Farb- und Klanggebilde hin, oder er fängt durch gezieltes Drücken auf der Sitzfläche an, einzelne Bild- und Klangteile zu erforschen und somit dieses Puzzle zu enträtseln.

Die einzelnen Bildfragmente sind Filmschnipsel, die aus P2P stammen oder auch von alten Werbe- und Dokumentarclips wie man sie zum Beispiel im Prellingerarchiv finden kann. Damit soll ein kleiner filmischer Abriss aus der Welt der Nachrichten- und Kommerzmedien entstehen, in der hier vorgestellten Arbeit zur totalen Reizüberflutung führt.

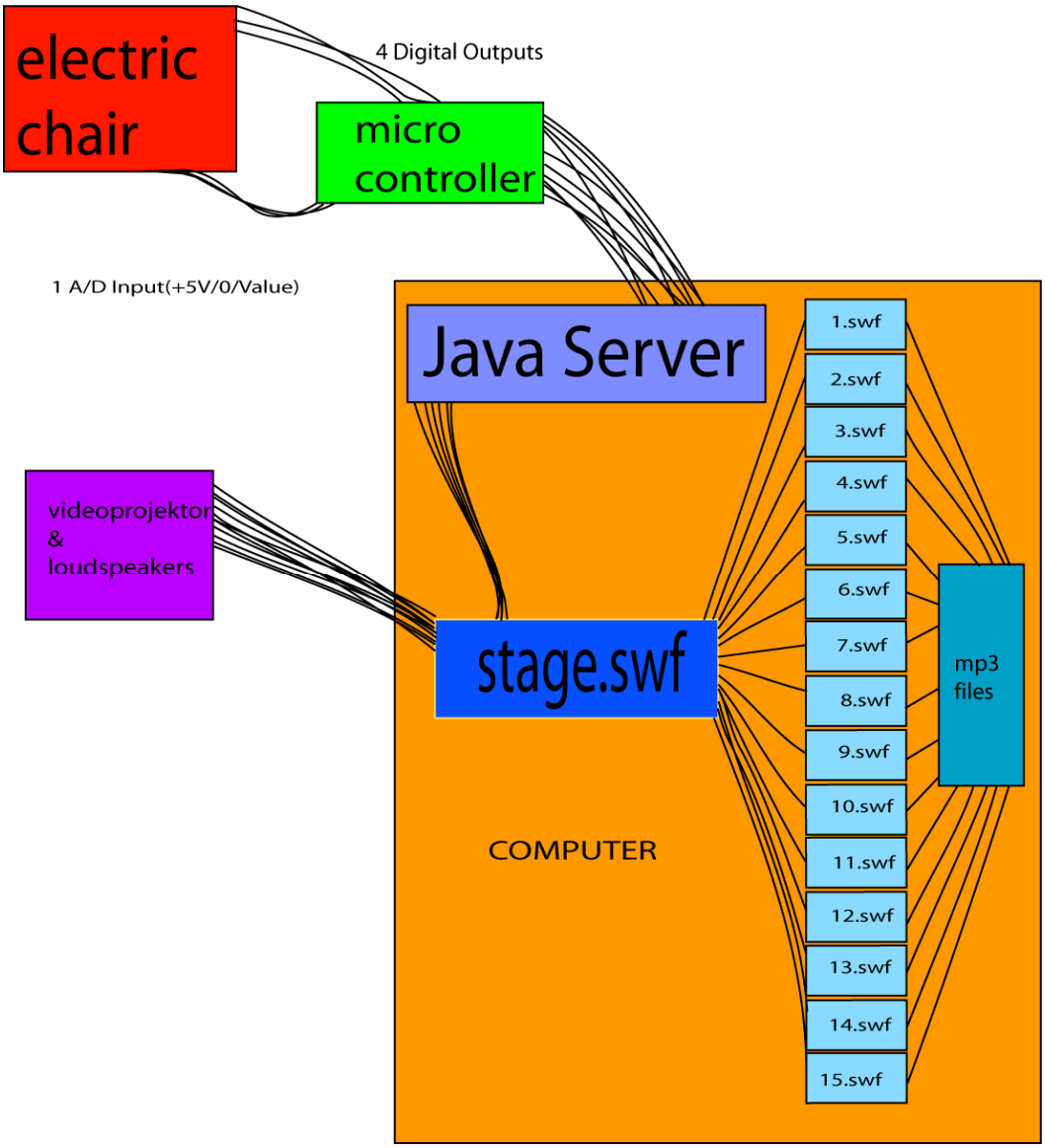
Zusammenfassung des technischen Aufbaus:

Frontansicht, unverkleidet

Detailansicht mit eingeschraubten Schiebepotentiometern

Die Person kommt zunächst in einen dunklen, bzw. abgedunkelten Raum, wo lediglich der Hocker zu sehen ist. Nachdem sie sich –hoffentlich- auf den Hocker setzen wird, geht das Lichtbild- und Toninferno los. Die Projektion und die Geräusche enden jedoch abrupt, sobald man wieder aufsteht und sich umgehend wieder in dem dunklen und stillen Raum befindet.
Unterhalb des Hockers sind 15 Schiebewiderstände angebracht, die mit Stößeln in der Schaumstoffmatte verbunden sind. Je nachdem, wie dieser Schaumstoff nun eingedrückt wird, bewegen sich die Regler der Schiebewiderstände. Ein an den Schiebewiderständen angebrachter Mikrocontroller, wandelt diese analogen Widerstandswerte in Zahlen zwischen 0 und 255 um. Diese Zahlen werden nun gemeinsam mit der jeweiligen Potentiometernummer über die serielle Schnittstelle an den Computer übertragen.
Das Auslesen dieser Daten und die graphische Ausarbeitung sollten mit Macromedia Flash erfolgen. Theoretisch würden sich auch andere Programme wie Macromedia Director oder Max/MSP dazu eignen. Flash hat jedoch den Vorteil, dass es zum einen sehr „robust“ mit dem Flashplayer im Dauerbetrieb ist und zum anderen die hier gewonnenen Erkenntnisse auch später für die Anbindung an ein Netzwerk/ Internet genutzt werden können.

Das Problem bei der Arbeit mit Flash ist die fehlende Kommunikationsmöglichkeit mit der seriellen Schnittstelle. Deswegen musste nun für die Kommunikation zwischen Flash und dem Mikrocontroller an der seriellen Schnittstelle ein Javaserver aktiviert werden, der die Daten von der seriellen Schnittstelle an das Flashprogramm weiter gibt.



Kommunikationsschema der Installation

Wenn man nun ein paar Wochen später all das auf die Reihe bekommen hat, geht es nun an die eigentliche abschließende Arbeit mit Flash. Wie bereits gesagt, stehen uns insgesamt 15 x 255 Bilder zur Auswahl, das bedeutet es handelt sich um insgesamt 3825 Bilder. Insgesamt wären daraus theoretisch 1,98 e+38 Bildkombinationen möglich! (Das aber nur nebenbei angemerkt.) Nun ist es natürlich nicht verwunderlich, dass bei 3825 Bildern der Flashplayer mit Bitmaps zusammenbrechen würde. Deswegen müssen alle Bilder, nachdem sie nun auf die passende Größe gebracht wurden und „auskeyt“ worden sind, auch noch in Vektorgrafiken umgerechnet werden. Das Austanzen der Bilder und ihre Umwandlung in Vektorbilder, reduzieren natürlich die Erkennbarkeit, produzieren aber ästhetisch sehr schöne grafische Collagen, die an abstrakte Bilder erinnern. (siehe screenshots weiter unten)

Theoretisch lässt sich mit diesem Aufbau eine ganze Menge machen. Es müssen ja keine Schiebepotentiometer sein, die hier zum Einsatz kommen. Denkbar sind die verschiedensten Sensoren die mit ihrer Umwelt in Kontakt treten. Auch können die gewonnenen Daten dazu benutzt werden, um Grafiken oder Anwendungen zu generieren und zu steuern, die mittels des Javaservers auch gar nicht in unmittelbarer Nähe der Sensoren sein müssen, sondern über Netzwerke und das Internet transportiert werden können.

Doch nun ins Detail! Ich hoffe, dass die Dokumentation ein wenig Klarheit schafft, wenn man da zum Selberbasteln angreifen will!

Elektronische Schaltung:

Ansicht der Mechanik beim Sitzen

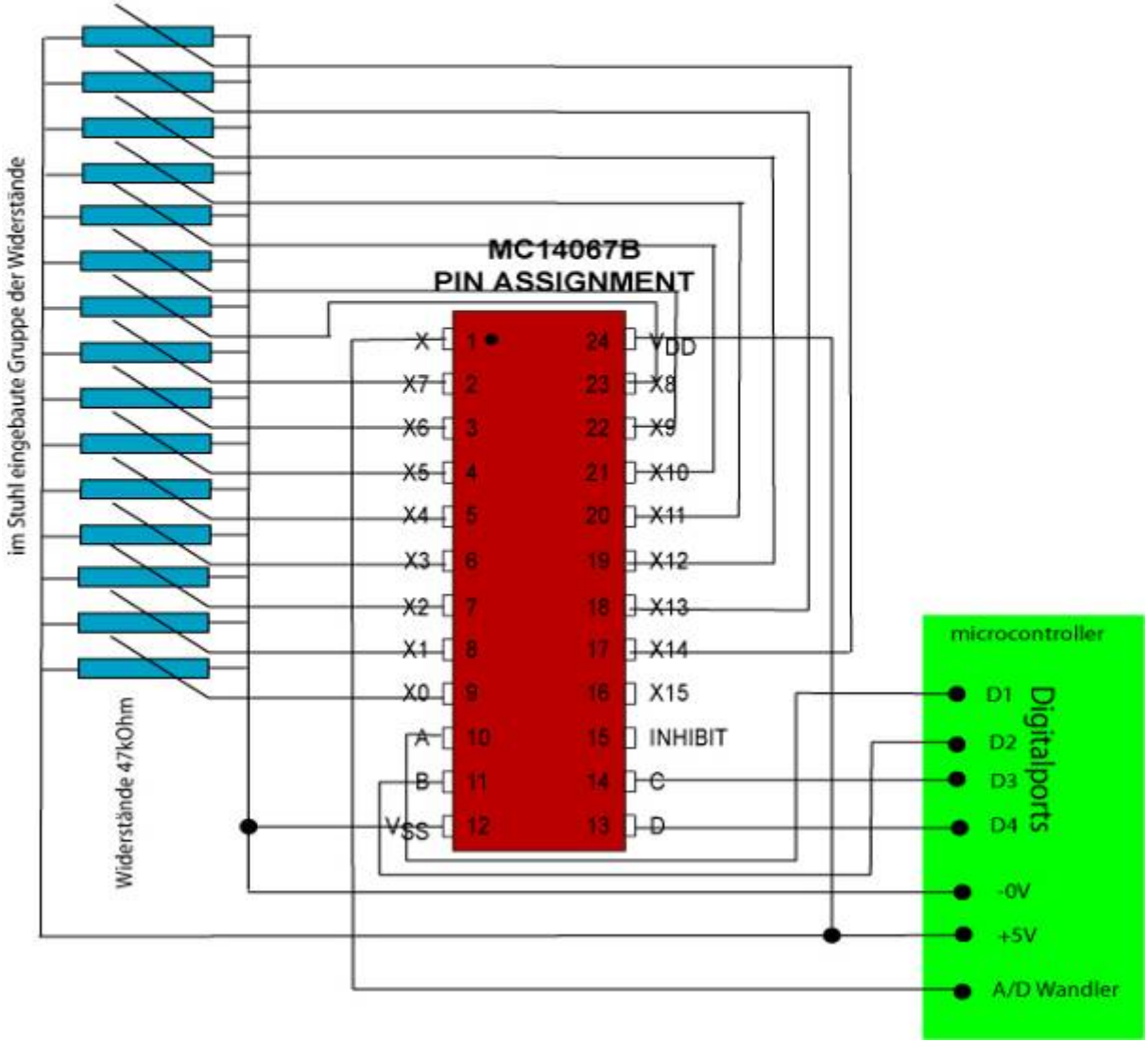
erste Tests mit dem komplett angeschlossenen Hocker

Der Schaltplan ist eigentlich relativ einfach. Das einzige Problem, dass sich jedoch ergibt, ist, dass der verwendete Mikrocontroller, das C-Control der Firma Conrad, nur 8 Analog/Digital Wandler hat. Um aber die 15 analogen Schiebewiderstände in den Griff zu bekommen, kommt nun ein so genannter Multiplexer zum Einsatz.

Belegungsschema des Multiplexers

MC14067 TRUTH TABLE

Control Inputs					Selected Channel
A	B	C	D	Inh	
X	X	X	X	1	None
0	0	0	0	0	X0
1	0	0	0	0	X1
0	1	0	0	0	X2
1	1	0	0	0	X3
0	0	1	0	0	X4
1	0	1	0	0	X5
0	1	1	0	0	X6
1	1	1	0	0	X7
0	0	0	1	0	X8
1	0	0	1	0	X9
0	1	0	1	0	X10
1	1	0	1	0	X11
0	0	1	1	0	X12
1	0	1	1	0	X13
0	1	1	1	0	X14
1	1	1	1	0	X15



SCHALTPLAN

Dieser famose Chip ist eine Art Umschalter, der immer nur einen seiner Analogports ausliest und an den Mikrocontroller weitergibt. Das Modell welches ich hier verwende, kann bis zu 16 verschiedene analoge Daten durchschalten. Mit den Digitalausgängen des Mikrocontrollers „sagen“ wir dem Multiplexer, welchen der 16 (bei mir nur 15) Ports er auswählen soll. Dafür brauchen wir vier Digitalports, die 2x2x2x2=16 verschiedene Kombinationen ergeben. Das dazugehörige Basicprogramm, welches auf dem Mikrocontroller läuft, sieht folgendermaßen aus:

```
define leds byteport[1]
define poti ad[1]
define a byte
DEFINE i byte
```

```
#abfragen

leds = &B00000000
print "a"; poti
print "a"; poti
leds = &B00000001
print "b"; poti
print "b"; poti
leds = &B00000010
print "c"; poti
print "c"; poti
leds = &B00000011
print "d"; poti
print "d"; poti
leds = &B00000100
print "e"; poti
print "e"; poti
leds = &B00000101
print "f"; poti
print "f"; poti
leds = &B00000110
print "g"; poti
print "g"; poti
leds = &B00000111
print "h"; poti
print "h"; poti
leds = &B00001000
print "i"; poti
print "i"; poti
leds = &B00001001
print "j"; poti
print "j"; poti
leds = &B00001010
print "k"; poti
print "k"; poti
leds = &B00001011
print "l"; poti
print "l"; poti
leds = &B00001111
print "m"; poti
print "m"; poti
leds = &B00001101
print "n"; poti
print "n"; poti
leds = &B00001110
print "o"; poti
print "o"; poti
goto abfragen
end
```

Mittels dem leds Befehl werden die vier Digitalports angesteuert. Danach wird der Wert des jeweils angesprochenen Potentiometers ermittelt. Das heißt, mit dieser Schaltung benötigen wir lediglich einen A/D Port als Eingang und 4 Digitalports um den Multiplexer anzusteuern.
Jedem Poti wird nun ein Buchstabe (a-o) zugeteilt und danach der umgewandelte Widerstandswert in einer Zahl zwischen 0 und 255 ausgegeben.
In der Praxis hat sich gezeigt, dass der printbefehl zweimal hintereinander ausgegeben werden muss, damit es auch reibungslos funktioniert. Das hängt wohl mit den unterschiedlichen Abfragefrequenzen der einzelnen Geräte zusammen und muss als „elektronisches Mysterium“ einfach mal so hingenommen werden….

Somit wird eine Zahlenkolonne an die serielle Schnittstelle übermittelt, die folgendermaßen aussehen kann:

```
a0
b35
c240
d22
e57
f187 usw.
```

Javaserver

Wie bereits angesprochen, braucht man nun einen Javaserver dazwischen, da Flash die serielle Schnittstelle nicht lesen kann.
Erfreulichweise hat Dan O'Sullivan ein Dozent am ITP in New York, bereits einen solchen Javaserver für die Arbeit mit Flash entwickelt. Den kann man downloaden und alle wichtigen Infos, wie man ihn installiert und zum Laufen bringt, gibt's hier. (Aber nur in English)

Wer sich nun erfolgreich mit dem Javaserver auseinandergesetzt hat, kann sich nun dem Flashteil zuwenden.

Programmierung in Flash

„wortwörtlich…. screenshots"

In Flash wird zu Beginn nur ein Clip gestartet, den ich „stage.swf" genannt habe. Dieses swf besteht aus nur aus einem Frame, in dem das Actionscript steht. Mit diesem Actionscript werden 15 Clips (swf´s) geladen, die ihrerseits jeweils 255 frames haben. Jeder Clip bekommt seine eigene Bezeichnung und kann später dann „auf den Frame genau" angesteuert werden. Sind alle Clips geladen, wird nun die Verbindung mit dem Javaserver gestartet. Nachdem die Verbindung aufgebaut ist, müssen die Daten vom Mikrocontroller ausgelesen werden. Das ist ein wenig „tricky", weil in Wirklichkeit keine Zahlenwerte ankommen, sondern ASCII Codes, die erst mühsam in verwertbare Daten umgewandelt werden müssen. Die hier vorgestellte Lösung ist sicherlich nicht die eleganteste, aber sie funktioniert zumindest. („Mit Kanonen auf Tauben schießen….."). Das untenstehende Actionscript kann man mit Copy/Paste gleich verwenden. Ich hab es noch ein bisschen kommentiert, vielleicht wird es ein wenig klarer dadurch.

Vielen Dank auch an dieser Stelle an Martin der den Teil für das Reinladen der Clips gelöst hat. Der Serverteil für Flash ist auch von der Seite von Dan O´Sullivan kopiert.

```
// erstelle zwei array, die clips und zugehörige container beinhalten
// creates two arrays containing its clips and his respective containers
var clips = new Array();
var containers = new Array();

// lade clips-load clips
for (var i=1; i<=15; i++) {

        // erstelle container und lade clip-create container and load clip
        var container       = containers[i] = _root.createEmptyMovieClip("container" + i, i);
        var clip            = clips[i]          = new MovieClipLoader();
```

```
		clip.loadClip("nummer" + i + ".swf",container);
		//trace(container);
}
serialServer = new XMLSocket();

//127.0.0.1 is the same as "localhost" ie an alias to your local machine
//it is concievable to that you would want to connect from another machine and you would change this
serialServer.connect("127.0.0.1", 9000);

serialServer.onConnect = function(success) {

		trace("connected " + success);
		serialServer.send("HOWDY FROM FLASH"+new Date().toString());
};

serialServer.onClose = function() {		trace("closed"); };

serialServer.onData = function(data) {
		/*An dieser Stelle wird die Laenge des Datastromes gecheckt Maximallaenge ist [Buchstabe]+
		  [dreistelliger Wert] + [Zeilenumbruch (vermute ich)]; Danach werden je nach Laenge, die einzelnen
		  Zahlen, die bis dahin ja noch ASCII Zeichen sind, in Zahlenwerte umgewandelt; die 48 kommt von ASCII Zeichensatz her
		  There will be checked the length of the data created by the microcontroller, which is composed by character,
		  a maximum of three numbers(still being ASCII code) and an additional code character. Once knowing the length of the
		  data stream, the ASCII code will transformed in numerical data. The 48 has to do with the ASCII code */

		var l = data.length;
		switch (l) {
		case 3 :
			var d = (data.charCodeAt(1))-48;
			var sum = d;
			break;
		case 4 :
			var a = (data.charCodeAt(1))-48;
			var b = (data.charCodeAt(2))-48;
			var sum = a*10+b;
			break;
		case 5 :
			var a = (data.charCodeAt(1))-48;
			var b = (data.charCodeAt(2))-48;
			var c = (data.charCodeAt(3))-48;
			var sum = a*100+b*10+c;
			break;
		}

		/* Hier wird der Buchstabe, der an der ersten Stelle liegt (a,b,.....,o) in Zahlen von 1 bis 15 gewandelt
		   The first character of the datastream (a,b,.....,o) will be transformed in values of 1 to 15 */
		var i = (data.charCodeAt(0))-96;

		/*Hier wird nun dem jeweiligen Film seine aktuelle Bildnummer (1-255) zugewiesen. Bei dieser Installation gibt der
		  Mikrocontroller fuer Bild Nummer 1 den Wert 255 aus, deswegen musste ich (256-sum) eingeben.
		  Haengt von der Einbaurichtung der Widerstaende ab.
		 Now the specific movie gets his actual frame number (1-255). In my specific installation the frame number 1
		 has the  microcontroller value of 255, so the correct value is (256-sum)*/

		containers[i].gotoAndStop(256-sum);

};
```

Die 15 Clips ihrerseits laden noch die dazugehörigen Soundfragmente aus einem Ordner namens „mp3files“ rein, und starten diese dann an verschiedenen Stellen des Films mit

```
Sound_nummer.start();
```

Die Sounds müssen alle einzeln ansprechbar sein, damit sie beim Aufstehen alle auch sofort verstummen. Dies würde theoretisch auch mit StopAllSounds(); gehen, jedoch sitzt man ja nicht alle Potis „runter“, dass bedeutet eines steht immer auf Frame 1, wo das StopAllSounds(); die ganzen anderen Sounds der gedrückten Potis abwürgen würde.

Der erste Frame eines der 15 Clips (hier Clip Nummer 6) schaut also dann exemplarisch so aus:

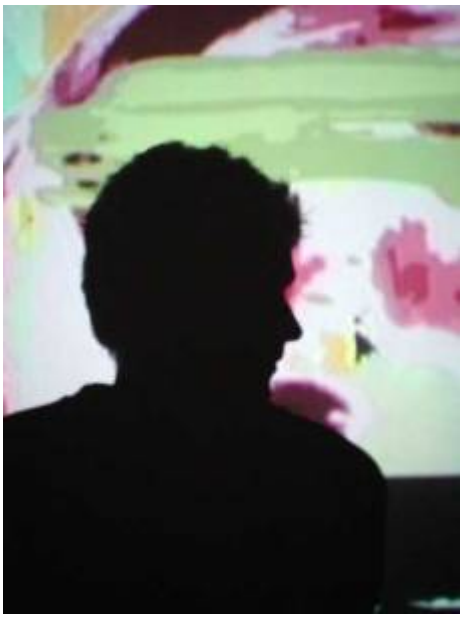
```
sound61.stop()
sound62.stop()
sound63.stop()
```

```
var sound61 = new Sound()
var sound62 = new Sound()
var sound63 = new Sound()
sound61.loadSound("mp3files/firstdeadsoldier.mp3", false);
sound62.loadSound("mp3files/freshegg.mp3", false);
sound63.loadSound("mp3files/goodcomercial.mp3", false);
sound61.setVolume(7);
sound62.setVolume(15);
sound63.setVolume(15);
sound61.setPan(-30);
sound62.setPan(80);
sound63.setPan(-70);
stop();
```

Das false bei loadSound kommt daher, dass der Sound erst beim event eintreten soll (streaming=false). Das letzte stop(); hält den Film, solange er noch nicht vom stage.swf angesprochen wird.

Das war´s! Wer noch Fragen hat, kann mir auch eine E-Mail schicken. Eine ausführliche Linkliste zu Sensoren, Mikrocontrollern und vielen spannenden Sachen mehr gibt’s hier auf der Seite der Bauhausuniversität in Weimar.

BAUHAUS UNIVERSITÄT WEIMAR



MEHR FOTOS UND FILMDOKU UNTER

********** http://www.subtours.com **********