procesverslag verdieping 05
Stefan Siderius 15-01-2013
Opleiding Kunst&Techniek, Saxion Enschede

INHOUDSOPGAVE

INLEIDING	P3
RISICOS	P4
HET OBJECT	P5
AANPASSINGEN	P6
TECHNISCHE REALISATIE	P7
BLOKKENSCHEMA	P8
VISUEEL	P9
AFTER EFFECTS	P11
EINDPRODUCT	P14
BYOB#2	P15
EINDPRODUCT	P16
FOTO	P17
REFLECTIE	P18
BRONNEN	P19
BIJLAGEN	P20

INLEIDING/DOELSTELLING

In mijn interne minor horen naast de media battle en media jam ook twee verdiepingen. Ik wilde één van deze verdiepingen gebruiken om een inzending voor Bring Your Own Beamer (BYOB) te maken. In 2011 heb ik namelijk deelgenomen aan de eerste editie in Utrecht en het leek me een geschikt evenement om met een nieuw concept nieuwe technieken uit te proberen.
Ik zag vorig jaar een projectiemapping op een schoen en dacht dat ik dat beter kon doen. Vervolgens ben ik gaan brainstormen over de mogelijkheden. In 2011 heb ik al eerder eens een projectiemapping gemaakt op de Waag in Deventer maar net als veel andere projectiemappings was dat op een statisch object.
Het leek mij juist een uitdaging om op een bewegend object een projectiemapping te maken. Een robot leek mij een mooi object om op te beamen, omdat het een krachtige herkenbare vorm is. Met simpele bewegingen heb je genoeg mogelijkheden om het een boeiende projectiemapping te laten zijn.

De punten waarop ik mij wil verbeteren zijn:
- Interactie met Arduino
- Compositing in 3d
- Motion Design in After Effects
- Aansturing in Flash

RISICO'S

Ik ben begonnen met het stellen van prioriteiten en een simpele risicoanalyse met de onderwerpen waarop mijn project stuk zou kunnen lopen. In willekeurige volgorde:

- De deadline
- Communicatie hard en software
- Beschikbaarheid geschikte beamer
- Synchronisatie beweging en beeld
- Betrouwbaarheid electronica

Allereerst de deadline, deze was namelijk erg strak. Tussen de initiatie van het project en de BYOB zaten slechts vier weken. Omdat ik ook nog andere verdiepingen aan het afronden was en er de week voor de BYOB weer een fulltime project op school begon heb ik zorgvuldig moeten plannen hoe en wanneer ik met het project aan de slag kon gaan.

De meest logische keus om de bewegingen aan te sturen van het object waren servo's met een Arduino. Ik heb even een motorcontroller overwogen maar de voordelen van servo's (nauwkeurigheid en compacte afmetingen) ten opzichte van electromotoren en bijbehorende mechanica waren al snel duidelijk.

Het belangrijkste bij een BYOB is misschien wel de beamer. Naast het feit dat ik een apparaat nodig had om te beamen had ik nog een vereiste en dat was de vorm. Omdat ik op een rechtop staand stuk speelgoed ging projecteren betekende dat dat de beamer op zijn kant moest staan. Ik had dus een beamer nodig die dat, indien mogelijk, zonder aanpassingen kon doen.
Na enig onderzoek hield ik uiteindelijk twee opties over, een oude versleten beamer die ik via werk kon lenen of een pico beamer via school. De oude beamer had een versleten lamp maar was wel rechthoekig. De picobeamer was erg compact en niet heel erg lichtsterk.

Na testen van de twee opties heb ik gekozen voor de oude hoekige beamer omdat deze erg stabiel staat, vast te zetten is met een slot en een 4 bij 3 beeldverhouding heeft. De picobeamer is erg klein en licht waardoor deze snel verschuift. Ook zit er geen kensingtonlock op. Dat vond ik wel een groot risico bij een openbaar evenement met een beamer die niet van mijzelf is.

Om de Arduino te laten communiceren met software heb ik na onderzoek gekozen voor Flash via tinkerproxy. Ik heb gekozen voor Flash ten nadele van processing omdat ik hier via school al les in heb gehad en zodoende al wat ervaring had. In het tweede jaar hebben we namelijk een physical computing opdracht gehad waar ik destijds twee lego motortjes aanstuurde met een h brug.
Het nadeel van Flash was de noodzaak om tinkerproxy te gebruiken, een extra tussenstap die ik met processing niet gehad zou hebben. Ik vond processing wel erg toegankelijk en het had de opties die ik zocht maar gezien de deadline leek het me beter tijd in het daadwerkelijke object te steken dan binnen drie weken me wegwijs te maken in een nieuwe programmeeromgeving.
Dat neemt niet weg dat ik wel denk dat het had kunnen werken mits ik meer tijd had gehad. Uiteindelijk heb ik dus de robot aangestuurd met Flash door via tinkerproxy een serial byte door te sturen naar de Arduino. Ik heb hiervoor een tutorial gevolgd maar daarover later meer.

Doordat mijn object ging bewegen was belangrijk dat het stabiel, licht en vooral stevig moest zijn. Omdat ik nog nooit met servo's en Arduino samen heb gewerkt vroeg ik me ook af hoe betrouwbaar deze zouden zijn.

HET OBJECT

Nadat ik het concept rond had ben ik op zoek gegaan naar het projectieobject. Ik wilde graag een speelgoed robot omdat ik robots erg tof vind maar ook omdat ik die simpel kan laten bewegen en ik dat vond passen bij de muziek die ik in gedachten had, Daft Punk en Tron. Omdat de deadline zo kort was heb ik niet via ebay besteld omdat de snelheid waarmee bezorgd wordt wisselend is. De webshops op internet van gewone speelgoedwinkels bieden vooral veel games en electronica aan maar geen ouderwetse robots. Ik heb vervolgens dus alle grote speelgoedwinkels in de stad bezocht en vond uiteindelijk bij de Intertoys de enige twee plastic robots die je in Enschede kan kopen.

Ik vond de witte mooier maar de rood zwarte robot was groter en had meer volume dus kon ik daar makkelijker iets inbouwen. Ook had deze al bewegende ledematen dus uiteindelijk heb ik deze gekozen.

De voet waar ik het object opgezet heb is een broodtrommel waar ik een uitsparing heb gemaakt voor de servo. Vervolgens heb ik met karton een aantal dwarsbruggen gemaakt waarmee ik de bewegingen van de servo vastgezet heb. Ik heb ook nog ruimte overgelaten voor de Arduino zodat het één geheel werd en ik zo min mogelijk loshangende onderdelen had.

AANPASSINGEN

Nadat ik de robot gekocht heb ben ik begonnen met het strippen. Ik heb alle ingewanden er uit gehaald en twee micro serco's in de schouders geplaatst waar ik de armen vervolgens aan gelijmd heb. De gewrichten in de benen heb ik vastgelijmd en waar nodig uitsparingen gemaakt zodat de signaal en voedingskabels er in de enkel uit komen. De basis van het geheel is een lunchtrommel waar ik een standaard servo in gezet heb die zorgt voor de draaiing om de verticale as. Op de servo heb ik een stuk hout gelijmd waarop ik twee strips klitteband tape heb geplaatst zodat ik de robot makkelijk kan demonteren voor transport.

Vervolgens heb ik de robot weer uit elkaar gehaald om deze wit te gaan spuiten. Wit omdat dat alle inkomend licht reflecteert dus als ik daar op projecteer komen de kleuren het beste uit. Dit maakt erg veel verschil want voordat ik de robot gespoten heb zag je de projectie bijna niet en op wit was het prima.

TECHNISCHE REALISATIE

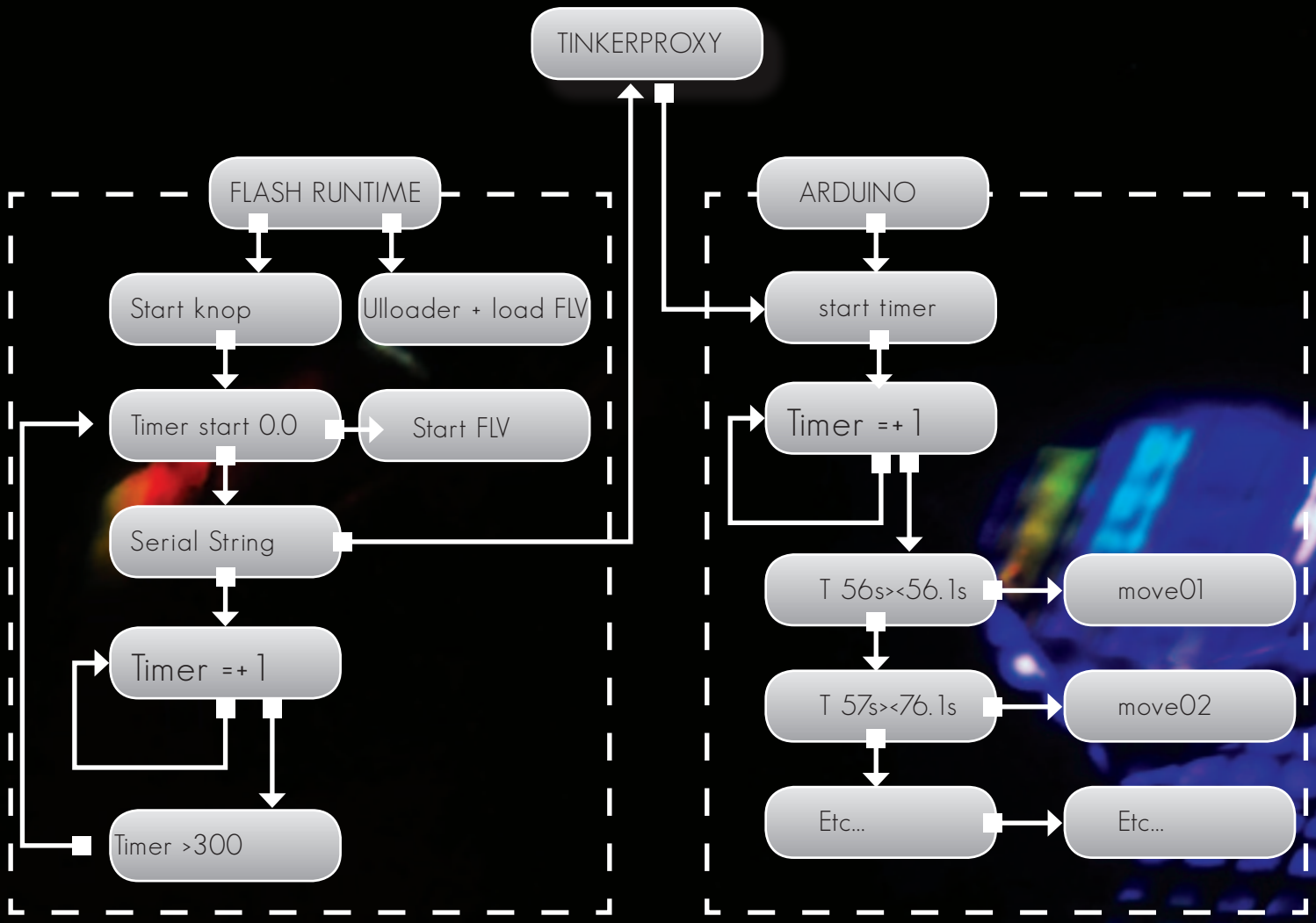
Een van de onderwerpen waar ik weinig kennis van had was de communicatie tussen Arduino en Flash. Na enig uitzoeken vond ik een tutorial over seriele communicatie door middel van het doorsturen van een tekst karakter geschreven door Mike Chambers.
Vervolgens heb deze tutorial gedaan waarmee ik de basis van deze techniek onder de knie heb gekregen. Met deze tutorial kon ik een led controleren door een byte over te sturen met Putty. Het werkt als volgt: Op Arduino draait een stukje code die de seriele poort afluistert naar binnenkomende data. Wanneer er een stukje bruikbare informatie doorkomt dat overeenkomt met een variable in het geheugen van Arduino kan deze hierop reageren. Door bijvoorbeeld een letter a door te sturen via putty of de serial monitor kan je dus een actie, in dit geval het schakelen van een led, oproepen.

Vervolgens heb ik mijn tinkerproxy juist ingesteld en vervolgens het Flash deel van de tutorial gedaan waarmee ik met Flash een serial string naar de tinkerproxy kan sturen.

De simpelste manier om de bewegende robot synchroon te laten bewegen met de projectie was om tegelijk met de video sequence een timer te laten starten op de Arduino.
Aan deze timer kunnen dan bewegingen getimed worden. De code die ik hiervoor gebruikt heb komt uit de opdracht physical computing die ik in het tweede jaar van de opleiding gemaakt had.
De timer werkt met 3 variabelen, currenttime en previoustime.
Currenttime is de tijd dat de Arduino aan staat. Wanneer ik vervolgens de animatie start wordt de previoustime gelijk gesteld aan de currenttime.
Op dat moment begint een runTime functie te lopen die niets anders doet dan de previoustime van de currenttime aftrekken. De waarde die overblijft is de runtime, de tijd dat de timer loopt. Wanneer deze waarde groter is dan een bepaalde waarde, zeg 55 seconden dan kan je daar een beweging op triggeren.

Hiermee komt een minder technisch probleem naar voren maar wel een belangrijke, namelijk de snelheid waarmee de servo's hun bewegingen maken. Ik wil een vaste factor hebben waarin ze een bepaald aantal graden draaien zodat ik daar mijn animatie op kan aanpassen in 3d studio max.
Het makkelijkst is de for loop zoals in het sweep voorbeeld maar deze heeft een groot nadeel en dat is de for loop zelf. Om deze goed te laten werken moet deze 1 keer volledig doorlopen zijn voor je er wat mee kan en ik vond dat niet geschikt. Ik ben vervolgens door gaan zoeken en kwam uit op een aanpassing op de servo.h library genaamd varspeedservo.
Deze library voegt een aantal extra variabelen toe waarmee de snelheid van de servo geregeld kan worden zonder bijzondere programmeer lussen.
Er zijn 3 variabelen nodig, minimum, maximum en speed. Ik heb daar de mid variabele aan toe-gevoegd zodat ik de servo's naar een neutraal stand kan zetten.
Na wat testen kwam ik uit op een snelheid van 30 waarmee ik in één seconde een servo 90 graden kon laten draaien. Deze waarde is sterk afhankelijk van de snelheid van de servo's!

Vervolgens heb ik deze tutorials toegepast om het programma te maken dat ik ging gebruiken voor de projectiemapping. In Arduino heb ik de code geschreven met alle timers en in Flash heb ik een Ulloader op de stage geplaatst die zijn flv uit een bepaalde map haalde. Zodra ik op pijltje naar boven druk verstuurt Flash een serial string naar tinkerproxy, start een timer en begint de ingeladen animatie te lopen. Wanneer de timer driehonderd seconden getelt heeft resetten alle variabelen en begint het filmpje opnieuw. De code kan je vinden in de bijlagen.



BLOKKENSCHEMA

VISUEEL

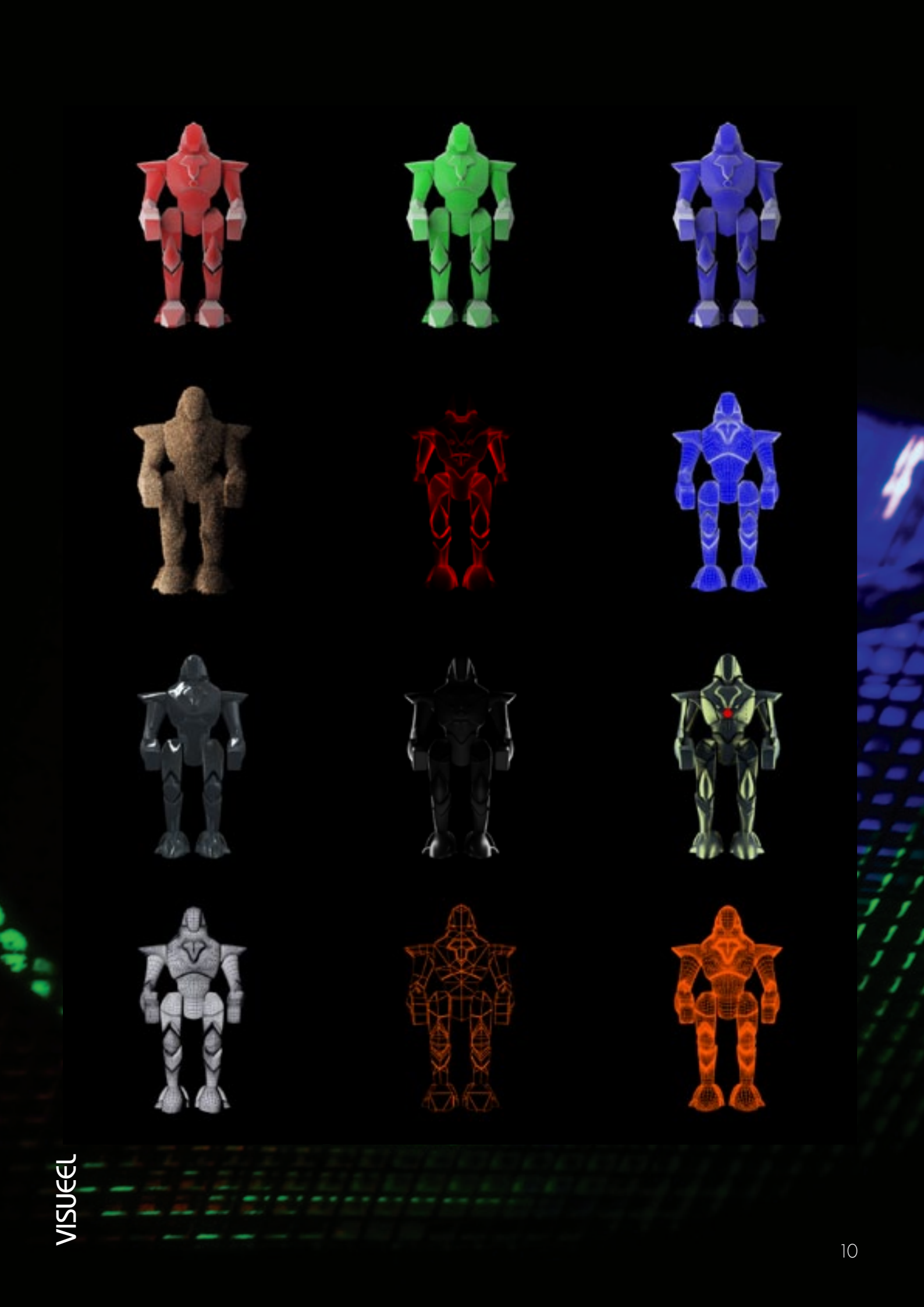
Naast de technische uitdaging moet er ook nog gewoon een animatie gemaakt worden. Zoals al eerder gezegd vond ik Daft Punk en Tron grote inspiratiebronnen. Het uitgangspunt waar ik uit ben begonnen te werken is de muziek. Ik heb wat filmpjes gekeken van projectiemappings en ik vond muziek een belangrijk versterkend element voor de animatie.
Daarom ben ik op soundcloud gaan zoeken naar obscure remixen van Daft Punk omdat ik graag een nummer wilde gebruiken dat niet direct herkenbaar was. Ik vond het ook een voorwaarde dat er voldoende beat in zat om op te kunnen monteren en liefst niet langer dan drie minuten omdat ik denk dat dat de maximale duur is van een projectie zonder dat mensen afhaken. Uiteindelijk kwam ik uit bij het volgende nummer, Daft punk - Derezzed - Robot edit door PIRACY. Het duurt twee en een halve minuut en leek me uiterst geschikt.

Vervolgens ben ik na gaan denken over hoe je het maximale uit je 3d projectiescherm kan halen en de diepte kan gebruiken om je projectie kracht te geven. Tijdens onze Grolsch projectie werkte extreme belichting en schaduw werking erg goed dus daar ben ik mee begonnen. Ik heb een lamp geanimeerd die vlak langs mijn object beweegt voor de intro en later nogmaals met een slaglicht dat beweegt met een grid projector map. Dit effect gebruik ik later nog eens door een lamp in een helix om het model te draaien terwijl het beweegt.

Een tweede eigenschap is de diepte van je object, uitstekende delen of gaten in je model. Als je een statisch object hebt met bijvoorbeeld ramen dan is het leuk om daar wat mee te doen maar ik vond het erg moeilijk om iets tofs te verzinnen omdat mijn robot best simpel was en helemaal solide. Ik bedacht uiteindelijk een oplossing en dat werd de ruimte scene waarbij de benen van de robot transformeren naar raketten en de kop en vuisten gloeiend rood worden.

Ik kreeg de feedback dat de robot nog vrij kaal was zoals te zien is in de afbeelding helemaal bovenaan. Vervolgens ben ik de details uit het fysieke model na gaan maken en dat levert een veel beter beeld op, zie de afbeelding hiernaast bijvoorbeeld.
Al testend kwam ik er achter dat simpele dingen met veel contrast het allerbest werken en dat je voor maximaal effect best kan overdrijven. Zo heb ik de eerder genoemde details geanimeerd en heb ik het in After Effects nog extra benadrukt en herhaald.

Ik bleef ook erg lang hangen in strakke metaalkleur en een donkere stijl dus ben ik gaan experimenteren met kleur en textuur in 3dsmax. Op de volgende pagina zie je een verzameling van alle testjes.



AFTER EFFECTS

Het animeren in 3d was één onderdeel, het echte werk gebeurde in After Effects. Omdat ik audio een belangrijk uitgangspunt vond heb ik een aantal dingen geautomatiseerd met expressions zodat ze gingen reageren op de muziek. Een voorbeeld hiervan is de begin sequence waarbij ik de opacity van de clip af laat hangen van de audio. Hiervoor heb ik de volgende expressie gebruikt:

thisComp.layer("Audio Amplitude").effect("Both Channels")("Slider")

Deze code staat in het expressie channel van de opacitylaag en zegt dat voor de waarde van de opacity gelijk is aan de waarde van de slider in de compositielayer audio amplitude.

Ik heb eerder al gebruik gemaakt van expressions en destijds een tutorial gebruikt om de basis onder de knie te krijgen. Voor dit project heb ik de tutorial weer opgezocht en nog een keer gedaan om mijn geheugen weer op te frissen. Ik heb nu iets meer ervaring met programmeren en daardoor kon ik ook makkelijker expressions aanpassen. Sommige expressions voor bijvoorbeeld positie zorgden ervoor dat elementen buiten beeld kwamen maar door de rekenwaarden aan te passen heb ik dat opgelost.

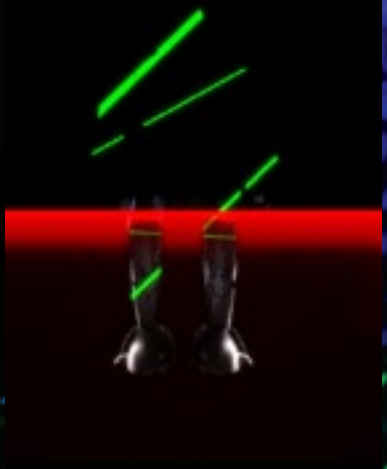
Dit is een erg simpele expressie en toepasbaar op heel veel parameters die je hebt in het pakket. Zo kan je bijvoorbeeld keyframes laten loopen met de volgende code: *loop_ out("pingpong",0)*

De positie van de RGB wireframes wordt bepaald door maskers die op hun beurt weer reageren op de audio expression:
temp = thisComp.layer("Audio Amplitude").effect("Both Channels")("Slider");
*[(temp*1.2)-500, 300];*
Omdat positie 2 variabelen heeft zie je dat er met twee temp waarden gerekend wordt. Temp is de waarde die uit de slider komt en wordt gebruikt voor de x en y as. De hoogte is variabel door de audio en de y waarde staat vast op 300 zodat het masker niet in de horizontale as uit beeld schuift.

Ik heb ook time-remapping gebruikt en gekoppeld aan de audio en wel door de volgende code:

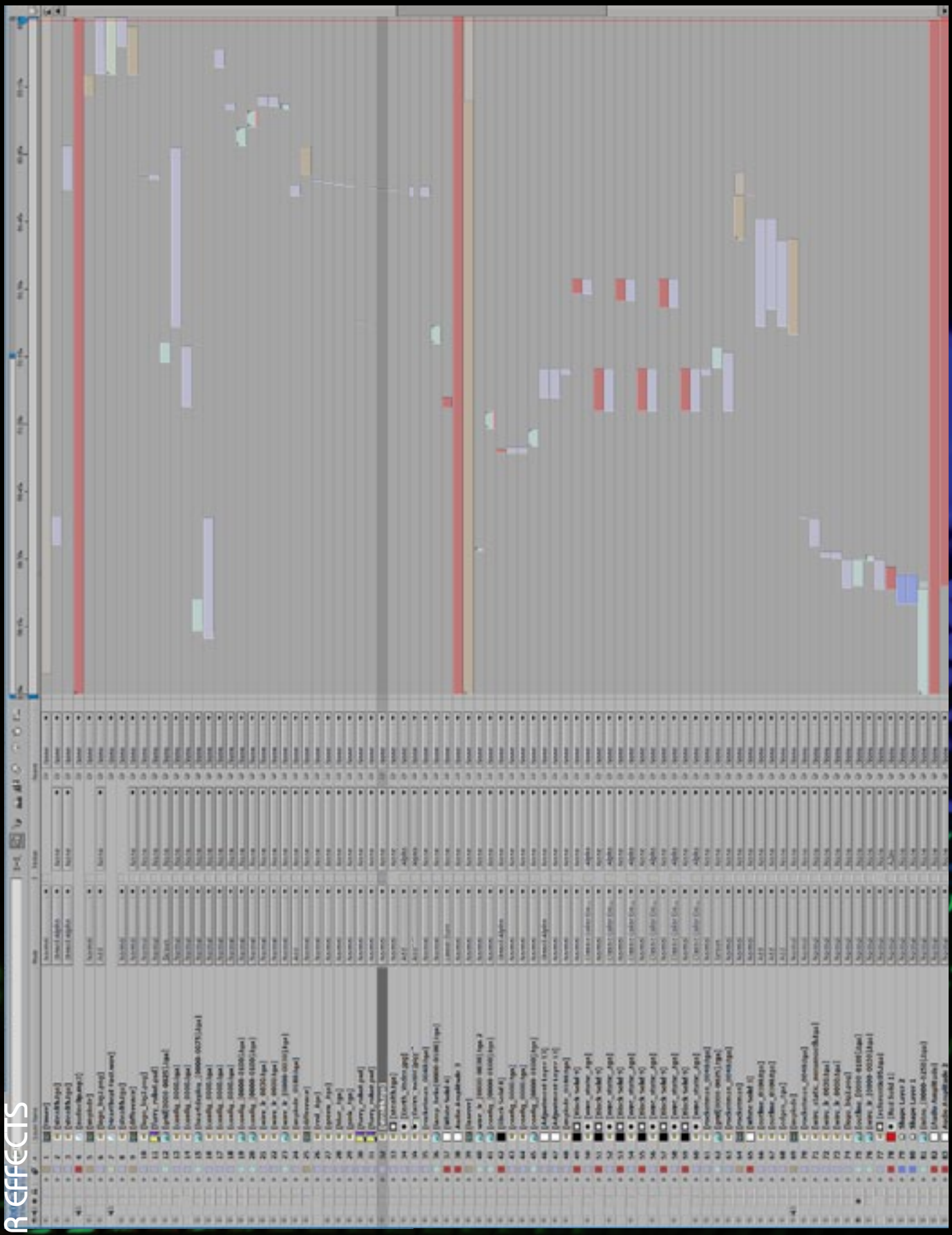
*0.6*thisComp.layer("Audio Amplitude 2").effect("Both Channels")("Slider")*

Dus 60% van de waarde van de slider is waar de video op dat moment hoort te zijn. De rest van de composition is ouderwets monteren en best veel herhaling van bestaande assets. Ik heb wel veel aanpassingen gedaan met bijvoorbeeld het tint effect waarmee je volledig andere kleuren toe kan passen, veel curves, fill en channelmixers. De sequence op het eind met de snelle kleurwisselingen was een ongelukje maar ik vond het zo'n mooi ongeluk dat ik het nog weer wat uitgebouwd heb. Toen de compositie bijna klaar was vond ik dat er nog te weinig samenhang in zat dus heb ik het pulserende middelpunt toegevoegd als een soort rode draad die in alle beelden voor komt.



AFTER EFFECTS

Compositie breakdown



AFTER EFFECTS

AFTER EFFECTS

Eerste opzet

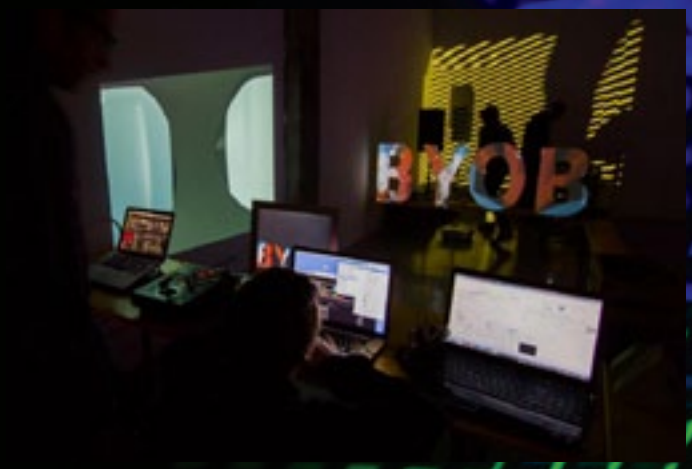
AFTER EFFECTS

Eindproduct 16 November

AFTER EFFECTS

BRING YOUR OWN BEAMER#2

De hele week hard gewerkt om de puntjes op de i te zetten. De timing van de armen blijft een hardnekking struikelblok, ik heb niet het idee dat de servo's bijzonder nauwkeurig zijn.
Dan is het eindelijk 16 November. Om vier uur was ik aanwezig om op te bouwen. Na een uurtje schuiven met tafels, beamers en robots snel getest en alles werkte. Om acht uur komen de eerste mensen en de installatie gaat aan. Ik had thuis al uitgebreid getest maar het is goed om te zien dat het herhalen van de movieclip en de aansturing van de robot prima werkt. Ik had nog meer tijd willen steken in geluidseffecten maar na tien minuten BYOB zie ik in dat het zinloos is. De aanwezige DJ draait hard en zelfs als je naast mijn installatie staat hoor je mijn zorgvuldig getimede muziek niet. Mensen zijn enthousiast en het trekt de absoluut de aandacht. Ik wilde mijn installatie nog graag filmen maar helaas gooit een defecte camera van school roet in het eten. Om middernacht gaat het licht aan en kan tevreden terug kijken op een succesvolle avond. Een werkende installatie, enthousiaste mensen en een vette projectiemapping dus ik ben helemaal tevreden.



fotos: Ton Nieuwenhuis

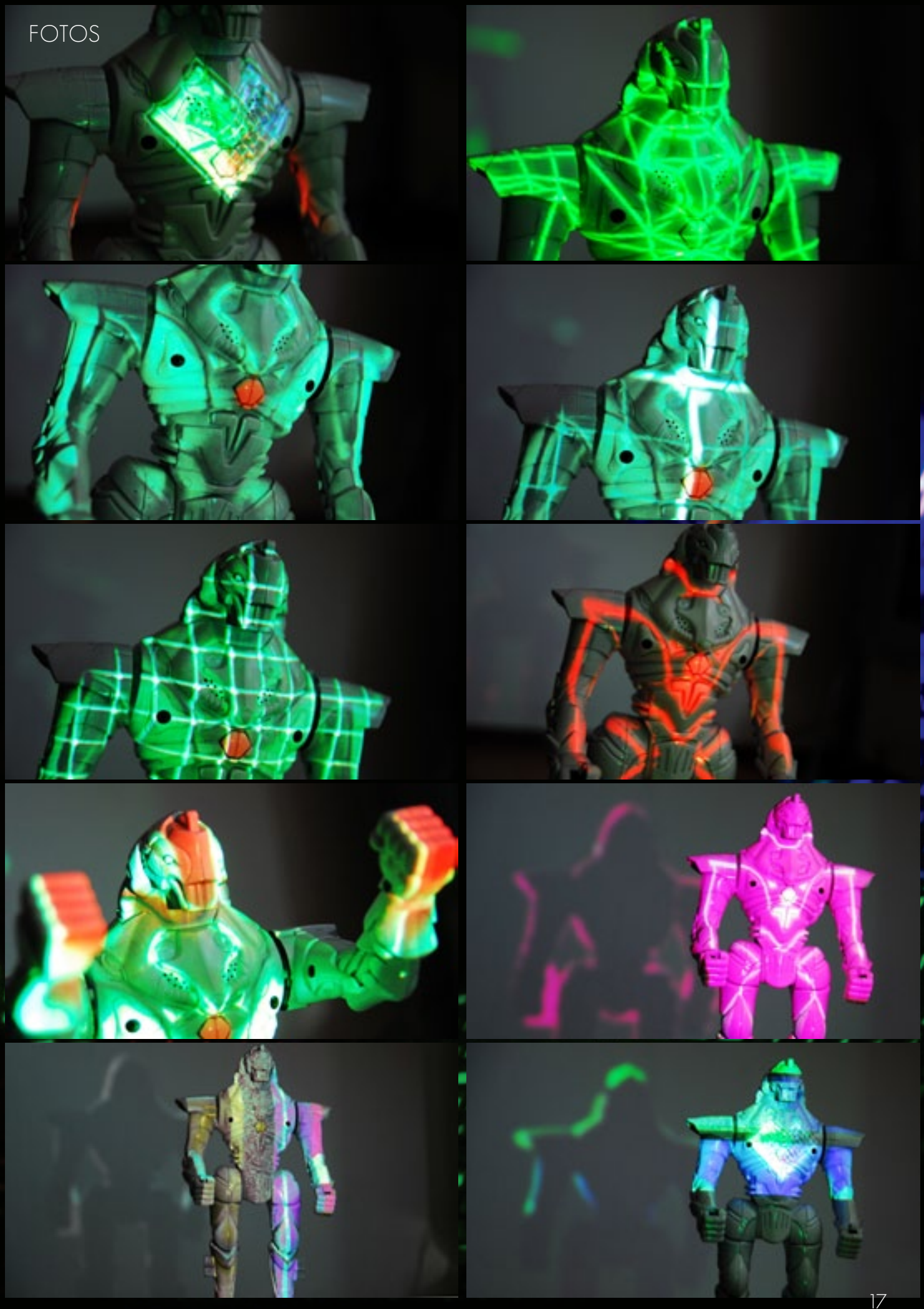
PROJECTIEMAPPING

In de studio

BRING YOUR OWN BEAMER #2

compilatie

FOTOS



REFLECTIE

Ik hoop dat dit verslag duidelijkheid verschaft over het proces dat ik doorlopen heb en de oplossingen die ik gevonden heb. Wat ik het leukst vond aan het project was de combinatie van vakgebieden en de ontdekking dat ik voor veel problemen een simpele techniek kon vinden die vaak met een goede tutorial uitgelegd kon worden.
Wat me tegenviel was de onbetrouwbaarheid van mijn eigen installatie. Niet zozeer de techniek want ik heb bijna elke dag wel geschaafd en getweaked tot het deed wat ik wil, maar de positie van de beamer ten opzichte van de robot. Dit en de positie van de robot op het draaipunt hebben veel invloed op de werking van de mapping zoals je kan zien in het filmpje.
Het mooiste zou een vaste opstelling zijn waarop de beamer en robot vast staan ten opzichte van elkaar. Ik had bijvoorbeeld thuis een perfecte opstelling bereikt maar voor het transport heb ik de robot los gemaakt van de voet en ondanks nauwgezet weer monteren valt het toch weer tegen. Ook de hele constructie die ik bewust licht gehouden heb had wel wat degelijker gekund.
In mijn projectvoorstel heb ik al kort beschreven dat ik graag een projectiemapping realtime zou willen maken met verschillende inputs (via Arduino) en Unity. Hier zou ik me dus verder in willen verdiepen. Een projectiemapping die als een soort VJ set fysiek reageert op de midi input, audio of licht.

Ik ben tevreden over het huidige project omdat ik goede reacties kreeg op mijn robot op Bring your own Beamer en ik de gestelde doelen bereikt heb. Tegelijkertijd ben ik kritisch op mijn werk want het was zeker niet perfect. Genoeg geleerd en genoeg nieuwe dingen ontdekt om mooiere dingen te maken in de toekomst.

BRONNEN

Varspeed servo library met snelheidsregeling.
http://Arduino.cc/forum/index.php?topic=61586.0
In deze thread meer info over de library.
http://Arduino.cc/forum/index.php/topic,68474.0.html

Om varspeedservo te gebruiken in Arduino 1.0 en hoger moet een waarde aangepast worden in een library file als onderstaand beschreven.
"I've used the varspeedservo library in Arduino 1.0. I just changed WProgram.h to Arduino.h in one of the library files."
http://Arduino.cc/forum/index.php/topic,61586.msg734800.html#msg734800

Onderstaande link heb ik gebruikt om de snelheid van de servo mee aan te passen. Dit werkte helaas niet zo goed als de varspeed servo maar was wel leerzaam.
http://www.bajdi.com/Arduino-servo-sweep/

De code voor de timer in de Arduino heb ik uit een oud schoolproject physical computing gehaald en is dus van Kasper Kamperman.
www.kasperkamperman.com

De timer in Flash die het geheel laat herhalen heb ik gemaakt aan de hand van de volgende tutorial:
http://www.foundation-Flash.com/tutorials/timer/

Communicatie tussen Flash en Arduino heb ik met een tutorial van Mike Chambers gemaakt.
http://www.mikechambers.com/blog/2010/08/04/getting-started-with-Flash-and-Arduino/#more-2170

De audio expressions in After Effects heb ik uit onderstaande tutorial.
http://library.creativecow.net/articles/tompkins_matthew/AE-Expressions_EQ-Bar/video-tutorial

Voor verdere filmpjes verwijs ik je naar mijn vimeo kanaal
https://vimeo.com/projekt86

BIJLAGEN

De code die de projectiemapping aanstuurt.

```
#include <VarSpeedServo.h>

#define RESET 'w'      //serial string variabele die doorgestuurd
wordt
#define START  'a'
#define EOL_DELIMITER '\n'


int incoming     = 0;       // variabele waarin de meest recente
serialstring opgeslagen wordt

boolean beamStart = false; // variable die bepaalt of het spel ge-
start is of niet
boolean isGestart = false;

unsigned long        runTime = 0; // gametime hoelang de gameStart
var is veranderd
unsigned long  currentMillis = 0; // huidige tijd
unsigned long previousMillis = 0; // vorige huidige tijd


VarSpeedServo myServo;  // create servo object to control a servo
VarSpeedServo myLarm;
VarSpeedServo myRarm;

int servoSpeeds = 30;  // snelheid van de servo
int servoMinPosition = 1;   // minimale waarde
int servoMidPosition = 90;  // midden waarde
int servoMaxPosition = 179; // max waarde


void setup()
{
    myServo.attach(10);     // fysieke verbindingen aan de pins
     myLarm.attach(12);
     myRarm.attach(11);

    Serial.begin(9600);// baudrate voor communicatie

    Serial.print("INITIALIZING");    // echo dat de Arduino leeft
en boot.
    Serial.print(EOL_DELIMITER);

    Serial.print("READY");       // echo dat het programma ingelad-
en is
    Serial.print(EOL_DELIMITER);

      myServo.slowmove(servoMidPosition,servoSpeeds) ;  // stel de
```

```
uitgangspositie in
      myLarm.slowmove(servoMidPosition-10,servoSpeeds) ; //-10 om-
dat ik de arm scheeft gelijmd heb
      myRarm.slowmove(servoMidPosition,servoSpeeds) ;


}

void loop()
{
        currentMillis = millis();           //huidige millis is
gelijk aan de gestarte timer in Arduino


     if(Serial.available() > 0){
         //read a single byte.
         incoming = Serial.read();

            if(incoming == START)    // serialstring start komt
binnen dus begin timers
            {
                digitalWrite(13, LOW);
                beamStart = true;
                if (isGestart == false){
                previousMillis = currentMillis;   // huidige millis
wordt opgeslagen in previousmillis om later gebruikt te worden in
de formule
              }
            }

            if(incoming == RESET)
            {
            digitalWrite(13, HIGH);
            beamStart = false;
            isGestart = false;
            runTime = 0;
            }
    }

 //begin sequence

    if(beamStart == true){
            runTime = currentMillis - previousMillis;
     }

    //slowrotate

     if (runTime >= 55240 && runTime <= 55300){ //na x miliseconden
doe dit

        myServo.slowmove(servoMaxPosition,servoSpeeds) ;
        myLarm.slowmove(servoMaxPosition,servoSpeeds) ;
        myRarm.slowmove(servoMaxPosition,servoSpeeds) ;
```

```
    }

   if (runTime >= 56140 && runTime <= 56200){

        myServo.slowmove(servoMinPosition,servoSpeeds) ;
        myLarm.slowmove(servoMinPosition,servoSpeeds) ;
        myRarm.slowmove(servoMinPosition,servoSpeeds) ;

    }

 if (runTime >= 58100 && runTime <= 58300){

        myServo.slowmove(servoMidPosition,servoSpeeds) ;
        myLarm.slowmove(servoMidPosition,servoSpeeds) ;
        myRarm.slowmove(servoMidPosition,servoSpeeds) ;

    }

 if (runTime >= 59100 && runTime <= 59300){

        myServo.slowmove(servoMinPosition,servoSpeeds) ;
        myLarm.slowmove(servoMinPosition,servoSpeeds) ;
        myRarm.slowmove(servoMinPosition,servoSpeeds) ;

    }
if (runTime >= 60200 && runTime <= 60300){

        myServo.slowmove(servoMaxPosition,servoSpeeds) ;
        myLarm.slowmove(servoMaxPosition,servoSpeeds) ;
        myRarm.slowmove(servoMaxPosition,servoSpeeds) ;

    }


 if (runTime >= 62200 && runTime <= 62300){

        myServo.slowmove(servoMidPosition,servoSpeeds) ;
        myLarm.slowmove(servoMidPosition,servoSpeeds) ;
        myRarm.slowmove(servoMidPosition,servoSpeeds) ;

    }


 //fakkeldrager

 if (runTime >= 78240 && runTime <= 78300){

        myServo.slowmove(servoMaxPosition,servoSpeeds) ;
        myRarm.slowmove(servoMaxPosition,servoSpeeds) ;
    }
```

```
 if (runTime >= 79200 && runTime <= 79300){

        myServo.slowmove(servoMinPosition,servoSpeeds) ;

    }

 if (runTime >= 81120 && runTime <= 81300){

        myServo.slowmove(servoMidPosition,servoSpeeds) ;
        myLarm.slowmove(servoMidPosition,servoSpeeds) ;
        myRarm.slowmove(servoMidPosition,servoSpeeds) ;

    }

//rocketman!

 if (runTime >= 105000 && runTime <= 105300){

        myLarm.slowmove(servoMinPosition,servoSpeeds) ;
        myRarm.slowmove(servoMaxPosition,servoSpeeds) ;

    }

 if (runTime >= 111000 && runTime <= 111300){
        servoSpeeds = 127;
        myServo.slowmove(servoMidPosition,servoSpeeds) ;
        myLarm.slowmove(servoMidPosition,servoSpeeds) ;
        myRarm.slowmove(servoMidPosition,servoSpeeds) ;

    }

 if (runTime >= 112000 && runTime <= 112300){
        servoSpeeds = 30;

 }

  //slowrotate

if (runTime >= 122120 && runTime <= 122200){

        myServo.slowmove(servoMaxPosition,servoSpeeds) ;
        myLarm.slowmove(servoMaxPosition,servoSpeeds) ;
        myRarm.slowmove(servoMaxPosition,servoSpeeds) ;

    }

   if (runTime >= 123100 && runTime <= 123200){

        myServo.slowmove(servoMinPosition,servoSpeeds) ;
        myLarm.slowmove(servoMinPosition,servoSpeeds) ;
        myRarm.slowmove(servoMinPosition,servoSpeeds) ;

    }
```

```
 if (runTime >= 125100 && runTime <= 125300){

        myServo.slowmove(servoMidPosition,servoSpeeds) ;
        myLarm.slowmove(servoMidPosition,servoSpeeds) ;
        myRarm.slowmove(servoMidPosition,servoSpeeds) ;

    }

 if (runTime >= 126100 && runTime <= 126300){

        myServo.slowmove(servoMinPosition,servoSpeeds) ;
        myLarm.slowmove(servoMinPosition,servoSpeeds) ;
        myRarm.slowmove(servoMinPosition,servoSpeeds) ;

    }
if (runTime >= 127200 && runTime <= 127300){

        myServo.slowmove(servoMaxPosition,servoSpeeds) ;
        myLarm.slowmove(servoMaxPosition,servoSpeeds) ;
        myRarm.slowmove(servoMaxPosition,servoSpeeds) ;

    }


 if (runTime >= 129200 && runTime <= 129300){

        myServo.slowmove(servoMidPosition,servoSpeeds) ;
        myLarm.slowmove(servoMidPosition,servoSpeeds) ;
        myRarm.slowmove(servoMidPosition,servoSpeeds) ;

    }


     if (runTime >= 200000 && runTime <= 201000){
            digitalWrite(13, HIGH);
            beamStart = false;
            isGestart = false;
            runTime = 0;



    }
   Serial.println(runTime);
}
```

Flash code projectiemapping software

```
import Flash.events.Event;
import Flash.display.Sprite;
import Flash.net.Socket;
import Flash.events.IOErrorEvent;
import Flash.events.ProgressEvent;
import Flash.events.SecurityErrorEvent;
import Flash.utils.Endian;
import Flash.events.MouseEvent;
import Flash.events.KeyboardEvent;


//command sent to the Arduino to toggle LED blinking state
const RESET:String = "w";   // serial string die doorgestuurd gaat
worden.
const START:String = "a";
var myTimer:Timer = new Timer(1000,0);

var Arduino_input:String;


//Character that delineates the end of a message received
//from the Arduino
const EOL_DELIMITER:String = "\n";

//socket we will use to connect to TinkerProxy
var _socket:Socket;

//Address where TinkerProxy is located. Will usually be
//localhost / 127.0.0.1
var _proxyAddress:String = "127.0.0.1";

//port TinkerProxy is listening on
var _proxyPort:uint = 5331;

//eventlisteners
stage.addEventListener(KeyboardEvent.KEY_DOWN,inPut) //luister naar
keydown
myTimer.addEventListener(TimerEvent.TIMER,timerFunction); //maak
een timerobject aan


//stop all movieclips
forward_mc.stop(); //button
backward_mc.stop(); //button
projectie.stop(); //projection

function onAddedToStage():void
{
     removeEventListener(Event.ADDED_TO_STAGE, onAddedToStage);
```

```
        _socket = new Socket();

        //Register for socket events

        //socket connected
        _socket.addEventListener( Event.CONNECT, onConnect );


        //socket closed
        _socket.addEventListener( Event.CLOSE, onClose );

        //data received from socket
        _socket.addEventListener( ProgressEvent.SOCKET_DATA, onSocket-
Data );

        //Error connecting
        _socket.addEventListener( IOErrorEvent.IO_ERROR, onIOError );

        //Security Error
        _socket.addEventListener( SecurityErrorEvent.SECURITY_ERROR,
onSecurityError );

        //need to set Endianness for Socket. THIS IS IMPORTANT
        //If this is set incorrectly, the Arduino will not be able to
understand
        //all of the data sent from Flash.
        //
        //See:
        //http://www.mikechambers.com/blog/2010/08/01/sending-multib-
yte-numbers-from-actionscript-to-Arduino/
        _socket.endian = Endian.LITTLE_ENDIAN;

        //connect
        _socket.connect(_proxyAddress, _proxyPort);
}

//called when we connect to the proxy server
function onConnect(event:Event):void
{
        /*
                note, you cannot reliably write data to the socket here.

                Im not sure if this is a Flash player issue, or a timing
issue
                with the proxy.
        */
        trace("Socket Connected");
}

/*
        This function / event handler is called when data is received
        on the socket.
```

```
     However, it is important to remember that the event is called
as
     data is received, which means that not all of the data may be
available
     when it is called.

     For example, if you sent the string "Hello World" from Ardui-
no,
     then the event handler might be called twice. Once with "Hello
Wo" and
     a second time with "rld".

     "Because of this, in most cases, you need to buffer the data,
and parse
     out messages, looking for a character (that you specify) that
delineates the
     end of a message.

     If you just want to send a single character back from Arduino,
then this
     is not necessary. However, the handler below is generic, and
already does all
     of the buffering, so you can just use it.
*/

//string to hold data as it comes in.
var buffer:String = "";

//event called when data arrives on the socket from the
//Arduino
function onSocketData(event:ProgressEvent):void
     {
          //get the string sent from the Arduino. This could be any
binary data
          //but in our case, we are sending strings.
          //In general, it is much easier to just always send
strings from
          //Arduino, and then parse then in ActionScript
          var data:String = _socket.readUTFBytes( _socket.bytesA-
vailable );

          //copy the newly arrived data into the buffer string.
          buffer += data;

          //completed message from the server
          var msg:String;
          var index:int;

          //loop through the buffer until it contains no more
          //end of message delimiter
          while((index = buffer.indexOf(EOL_DELIMITER)) > -1)
          {
               //extract the message from the beginning to where the
```

```
delimiter is
                //we don't include the delimiter
                msg = buffer.substring(0, index);

                //remove the message from the buffer
                buffer = buffer.substring(index + 1);

                //trace out the message (or do whatever you want with
it)
                trace("Message Received from Arduino : " + msg);
           }

      }


      function inPut(event:KeyboardEvent):void
           {
           if (event.keyCode == Keyboard.UP){
                beginAwesome();
           }
                if (event.keyCode == Keyboard.DOWN){
                endMadness();
           }

      }

      function Loop(){
           _socket.writeUTFBytes(Arduino_input);
           _socket.flush();
      }


      function timerFunction(event:TimerEvent) {
           trace(String((300-myTimer.currentCount)));
           if (myTimer.currentCount == 300) { //wanneer de timer
groter is dan 300 eenheden, reset
                myTimer.reset();
                myTimer.start();
                beginAwesome();
           }
      }

      function beginAwesome(){ //begin de projectie, stuur een start
serialstring en start de timer
                projectie.play();
                Arduino_input = START;
                forward_mc.gotoAndStop(2);
                myTimer.start();
                Loop();
      }

      function endMadness(){
                projectie.stop();
```

```
                Arduino_input = RESET;
                backward_mc.gotoAndStop(2);
                Loop();
        }


//called when the socket is closed
function onClose(event:Event):void
{
        trace("Socket Closed");
}

//called if an error occurs while connecting to the socket.
function onIOError(event:IOErrorEvent):void
{
        trace("IOErrorEvent : " + event.text);
}

//called when there is a security error. Usually if you try to con-
nect to a socket
//when the SWF doesn't have permission.
//See:
//http://www.adobe.com/devnet/Flashplayer/articles/fplayer9_secu-
rity_04.html
//In most cases, when testing locally, this will not be an issue.
function onSecurityError(event:SecurityErrorEvent):void
{
        trace("SecurityErrorEvent : " + event.text);
}

onAddedToStage();
```