CATCH ME IF YOU CAN

ITS & MOSY
Sommersemester 2019

Jette Heinsohn 2320987
Max Bauer 2323129
Dominik Jordan 2318526
Nele Schuster 2319509
https://github.com/FunkyMax/CatchMeIfYouCan

Jette Heinsohn, Max Bauer
Dominik Jordan, Nele Schuster

Inhaltsverzeichnis

Jette Heinsohn, Max Bauer
Dominik Jordan, Nele Schuster

Konzept

Projektziel

Wir konzipieren ein Katz-und-Maus-Spiel. Dabei übernehmen zwei Moving Heads die Rolle der Charaktere, wobei die „Maus“ sich zufällig über die Spielfläche bewegt. Die „Katze“ wird per Joystick über eine App von einem Spieler gelenkt. Die Moving Heads projizieren auf eine Wand, die als Leinwand dient. Das Spiel ist vorbei, wenn die Maus in einer vorgegebenen Zeit nicht gefangen wird oder nach einem festgelegten Zeitrahmen, in welchem die Maus möglichst oft gefangen werden muss.

Anforderungsanalyse

Das Spiel soll unterhaltsam, leicht verständlich und intuitiv bedienbar sein. Die Hauptfunktion ist es, die Koordinaten der Moving Heads auf der Wand genau berechnen zu können, um damit sicher Treffer erkennen zu können. Unser Plan ist es, dass die Berechnung aller Positionen und Bewegungen in der App abläuft. So sollen die DMX-Werte direkt aus der App über die Bluetooth-Verbindung in den Arduino gegeben werden. Dadurch versuchen wir Latenzen möglichst gering zu halten. Die Bewegungen werden auf einer festen Fläche skaliert, die nicht von den Moving Heads verlassen werden kann.

Falls es die Zeit und das Material zulassen, überlegen wir, ein Mehrspieler-Modus einzubinden. In diesem haben beide Spieler ein Smartphone und können jeweils einen Moving Head steuern, sodass die Maus sich nicht zufällig bewegt, sondern auch von einem Spieler gesteuert wird.

Eine weitere Option ist es, mehrere Moving Heads als Mäuse einzubauen und daraus eine Art Coop-Spiel zu bauen, in welchem die zwei Spieler in einer bestimmten Zeit möglichst viele Mäuse fangen müssen. Das hängt von den uns zur Verfügung stehenden Moving Heads und der verbleibenden Zeit ab.

So ist die Anzahl der umgesetzten Modi am Ende abhängig von der Zeit, die uns noch bleibt.

Auch werden wir uns zunächst auf eine feste Positionierung der Moving Heads festlegen, in welcher die Positionsberechnungen sicher funktionieren. Wenn es zeitlich umsetzbar ist, soll es eine Möglichkeit geben, die Positionen der Moving Heads zur Wand und die Größe der Wand individuell einstellen zu können.

Technische Rahmenbedingungen

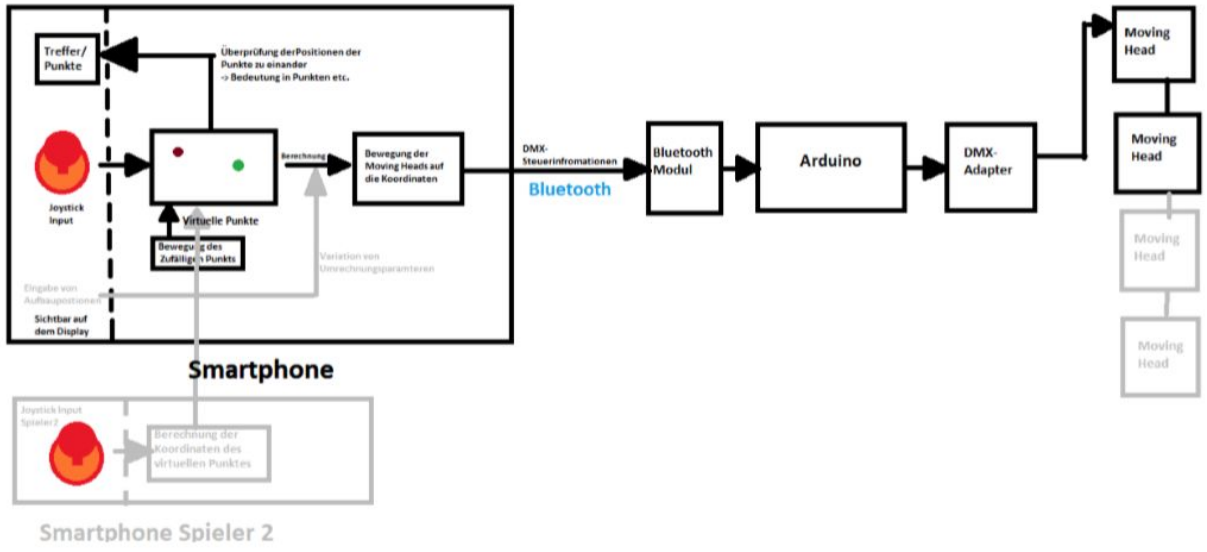
Wir benötigen:

- Min. 1 Smartphone
- Min. 2 Moving Heads des Types Spot (max. 5)
- 1 Bluetooth Modul
- Arduino - DMX-Adapter
- Abschlusswiderstand

Technisches Konzept

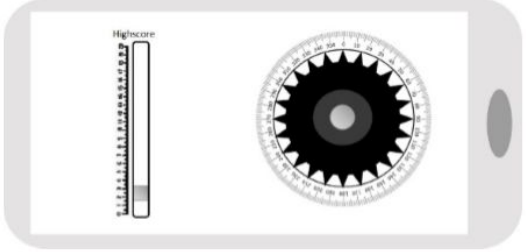
Da die mobile Anwendung auf einem Android Gerät laufen soll, wird die App mithilfe der Entwicklungsumgebung Android Studio entwickelt. Dabei wird die neuartige Programmiersprache Kotlin zum Einsatz kommen. Die App übernimmt dabei zwei grundlegende Aufgaben. Wie bereits erwähnt, dient die grafische Oberfläche der App als Steuerungseinheit. Ein virtueller Joystick wird in 360° Grad bewegbar sein. Dieser Input wird als DMX taugliche Information vom Handy an das Bluetooth Modul des Arduinos gesendet, welcher die Signale dann an die Moving Heads weiterleitet.

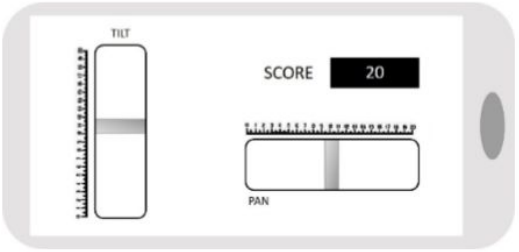
Daneben simuliert die App die Positionen und Bewegungen der realen Moving Head-Lichtkegel in einem eigenen 2D-Koordinatensystem. Dabei steuert der virtuelle Joystick die Lichtkegel in diesem Koordinatensystem (und nicht die realen Moving Heads an sich). Pro Frame werden diese Koordinaten in reale Weltkoordinaten umgerechnet, sodass darauf basierend der Winkel der realen Moving Heads eingestellt wird. Falls die Koordinaten beider Moving Head-Lichtkegel im simulierten Koordinatensystem übereinstimmen, sollen die realen Moving Heads einen identischen Punkt auf der Leinwand anleuchten. So korrespondieren die Moving-Head-Koordinaten mit den simulierten Werten.



Bedienkonzept

Wie schon erwähnt wird die Steuerung des Spiels über einen virtuellen Joystick ablaufen. Ob es sich hierbei um einen 360° oder zwei separate Zweiweg-Joysticks handelt, ist noch offen. Auf dem Userinterface soll ebenfalls eine Art Punkteskala der bisherigen Treffer sichtbar sein. Unser Ziel ist es, dass die Moving Heads möglichst in Echtzeit bewegt werden.





Auch die Kalibrierung der Moving Heads ist eine Idee, die wir oben schon erwähnt haben. Dabei könnten die Einstellungen der Entfernung der Moving Heads und der Wand angepasst werden. Es bleibt offen, ob wir genügend Zeit haben, so etwas umzusetzen.

Das Spiel läuft unter dem Namen “Catch me if you can”, eine Anspielung auf den Film, der sich mit dem Leben von Frank Abagnal beschäftigt. Darin wird der Hochstapler Abagnal von der Polizei verfolgt. Diese Parallele versuchen wir auch in unserem Spiel widerzuspiegeln. Gerade auch bei unserem Poster werden wir hierauf zurückgreifen.

Zeitplan

April
- Joystick, der Punkt in zweidimensionalem Raum bewegt (nur in der App)
- Joystick, der Moving Head bewegt (Horizontal/Vertikal)
- Bewegung des Punktes in der App mit der des Moving Heads an der Wand synchronisieren

Mai
- Zweiten Punkt, der sich zufällig im zweidimensionalen Raum bewegt, integrieren Zweiten Moving Head integrieren
- Spielmodus einbauen, welcher Überlagerung der Punkte kontrolliert *Basisfunktion erreicht*

Juni
- Kalibrierungsmodus integrieren
- Unterschiedliche Schwierigkeitsgrade einbauen / Balancing
- Weiter Moving Heads integrieren Juli
- Weitere Spielmodi umsetzen
- Zweispielermodus bauen

Hardware

Benötigte Hardware:

1.) Arduino Uno
2.) DTS Jack (4x)
3.) Bluetooth Modul HC-06
4.) DMX Transceiver MAX485
5.) Adroid Smartphone

1.) Arduino Mega
Der Arduino Mega ist ein microcontroller board.
-54 digital input/output pins
-16 analog inputs, 4 UARTs (hardware serial ports)
-16 MHz crystal oscillator
-USB connection
-ICSP header
-reset button

2.) DTS Jack
Der DTS Jack ist ein high power compact lightweight moving head.

- Beam, Spot und Wash in einem Gerät
- 1 Goborad mit 9 rotier- und austauschbaren Gobos
- 1 Goborad mit 10 festen Gobos
- Rotierbares 4-fach-Prisma
- Motorisierter Zoom von 1° bis 46°
- 1 Farbrad mit 18 Farben
- 50.000 Lux (5m)
- Nur 230W Leistungsaufnahme
- Schnelle Pan/Tilt-Fahrten (2,5s/1,5s)

3.) Bluetooth Modul HC-06
· Arduino Bluetooth Terminal
· BlueTerm
· Blue Control

4.) DMX Transceiver MAX485
· Anbindung von Mikrocontrollern (z. B. Arduino MEGA) an den RS485 Bus
· Umwandlung von „normalen“ Digitalen Signalen in „differentielle“ Signale

PINS
· RO: Leseleitung (zum Arduino)
· DI: Schreibleitung (vom Arduino)
· RE: Read Enable [low aktiv] (vom Arduino)
· DE: Data Enable [high aktiv] (vom Arduno)

Jette Heinsohn, Max Bauer
Dominik Jordan, Nele Schuster

Technische Umsetzung

An dem Arduino Mega werden das Bluetooth Modul HC-06 sowie der DMX Transceiver MAX485 angeschlossen. (siehe Schaltplan S.?) Um diesen dann mit unseren DTS Jacks zu verbinden haben wir die Kabel an den MAX485 angeschlossen und an einen DMX Stecker (3-pol) gelötet. Dieser wird dann an den ersten DTS Jack angeschlossen. Die weiteren drei DTS Jacks schleifen wir dann mit normalen DMX Kabeln (3-pol) durch. Über das androidfähige Smartphone senden wir per Bluetooth die Daten an unseren Arduino, der diese dann über das MAX485 Modul an unsere DTS Jacks weiterleitet. Die DTS Jacks werden unter dem standardmäßigen 25 Kanal Modus eingestellt. Die vier DTS Jacks beginnen dann jeweils auf Kanal 1, 26, 51 und 76. Diese werden dann im Arduino Code angesteuert. In den 25 Kanälen liegen die verschiedenen Steuerelemente des Moving Heads, z.B. Pan, Tilt, Farbe, Helligkeit, Motorgeschwindigkeit.(Aus dem DTS Jack Manual, siehe Anhang) Diese Kanäle belegen wir im Code mit den von uns gewünschten Werten, bzw. mit dem vom Benutzer generierten Input.

Logik

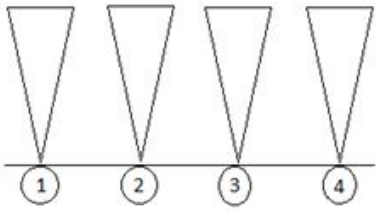
Um die Koordinaten der Punkte in der Virtuellen Ebene in DMX-Werte für die Moving Heads zu konvertieren werden folgende Schritte durchgeführt

Koordinaten in der Virtuellen Ebene in Koordinaten auf der Wand umrechnen

$$Wand\ X\ = Virtuell\ X * \frac{Wandbreite}{Displaybreit} - Offset\ des\ MH$$

Zunächst wird die X-Koordinate auf der virtuellen Ebene auf die Wand skaliert mit dem Faktor Wandbreite/Displaybreite. Dann muss die Koordinate allerdings noch verschoben werden um den sogenannten Offset des jeweiligen Moving Heads, Dieser Offset berechnet sich folgendermaßen:

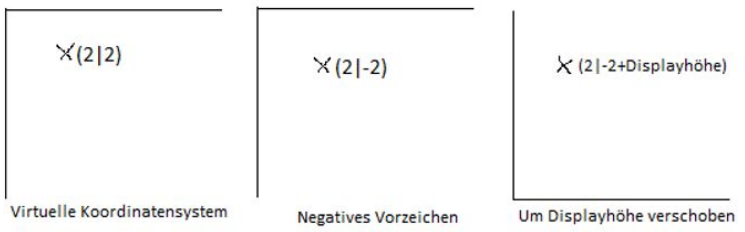
$$Offset\ des\ MH = \frac{Wandbreite}{2} + (n - 2{,}5) * AbstandMH$$



Wobei n der Position der Lampe entspricht von rechts gesehen und Die mit dem Abstand der Moving Heads Multipliziert wird, welcher zwischen allen Moving Heads gleich ist.

$$Wand\ Y\ = (-Virtuell\ Y + Displayhöhe) * \frac{Wandhöhe}{Displayhöhe}$$

Um die Y-Koordinate auf der Wand zu berechnen muss man diese allerding zunächst einmal invertieren und um die Displayhöhe nach oben verschieben. Dies liegt daran, dass das virtuelle Koordinatensystem sich im von den Positionen im unteren rechten Quadranten des Koordinatensystems befindet, weswegen das ganze um die Displayhöhe verschoben wird, jedoch nur positive Y-koordinaten ausgibt, und das negative Vorzeichen somit vorhanden

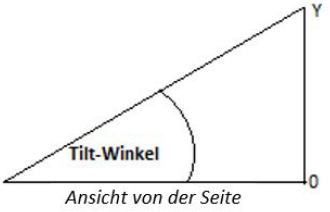


ist um die wirkliche Positionierung im unteren rechten Quadranten darzustellen. Dann kann man diesen Wert genau wie die X-Koordinate skalieren.

Pan- & Tiltwinkel des Moving Heads zu den Koordinaten berechnen

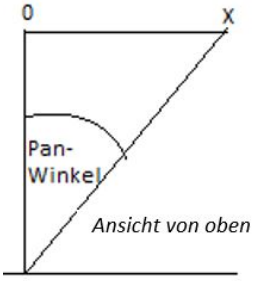
$$TiltWinkel = arcsin\left(\frac{Wand\ Y}{\sqrt{WandX^2 + WandY^2 + Entfernung\ zur\ Wand^2}}\right) * \frac{360}{2\pi}$$

Der Tilt-Winkel wird einfach berechnet, indem man ein Dreieck zwischen dem Moving Head seinem Auftreffpunkt auf der Wand und dem Punkt auf der X-Achse mit derselben X-Koordinate wie der Auftreffpunkt. Dann wird einfach der Arkussinus berechnet. Die Länge der Gegenkathete entspricht der Höhe also der Y- Koordinate auf der Wand. Die Länge der Hypotenuse wird aus den drei Vektorelementen des Lichtstrahls vom Lampenursprung bei zu den Auftreffpunkt auf der Wand mit Hilfe des Satz des Pythagoras berechnet. Der Wert muss dann noch vom Bogenmaß in Gradmaß umgerechnet werden.



$$PanWinkel = arcsin\left(\frac{Wand\ X}{\sqrt{WandX^2 + WandY^2 + Entfernung\ zur\ Wand^2}}\right) * \frac{360}{2\pi}$$

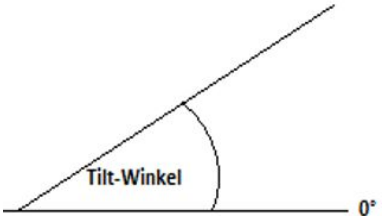
Der Pan-Winkel wird genau wie der Tilt-Winkel berechnet. Nur das für die Länge der Gegenkathete dieses Mal die Entfernung zwischen der Y-Achse und dem Auftreffpunkt in de X-Achse genutzt.



16 bit DMX-Werte aus den Winkeln berechnen

$$TiltDMX = T0 + \frac{T45 - T0}{45} * TiltWinkel$$

T0 ist der DMX-Wert, der der 0°-Position entspricht und T45 dem DMX-Wert, bei welchem einem Tilt-Winkel von 45° erreicht wird. Somit beschreibt der Bruch (T45-T0)/45 die lineare Steigung des DMX-Wertes. Und wird dann mit dem Tilt-Winkel multipliziert, um eine Lineare Veränderung zu erreichen.

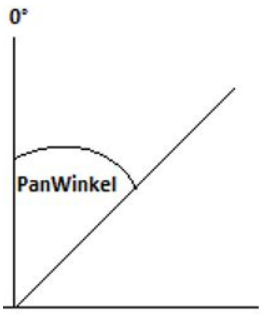


$$PanDMX = P0 + \frac{P45 - P0 + Verzerrungsausgleich}{45} * PanWinkel$$

Der Pan-DMX-Wert berechnet sich an sich genau wie der Tilt-DMX-Wert. Allerdings gibt es bei dem Pan-Wert eine Verzerrung mit steigendem Tilt-/Y-Wert. Deswegen mit zum P45 noch ein Versorgungsausgleich hinzugerechnet.

$$Verzerrungsausgleich = (DisplayHöhe - Pixel\ Y) * Streckfaktor$$

Dieser Verzerrungsausgleich berechnet sich aus der (Displayhöhe-Pixel Y), sodass der Wert linear mit der Höhe auf der Wand steigt multipliziert mit einem Streckfaktor, welchen wir ausprobiert haben (und vermutlich abhängig von der



Entfernung zur Wand ist).

16-bit DMX-Werte in zwei 8-bit für die jeweiligen Kanäle umrechnen

$$Pam\ Channel\ 1 = \frac{PanDMX}{256}\ [muss\ abgerundet\ werden]$$

Die ersten 8 bit des Pan-Wertes

$$Pan\ Channel\ 2 = PanDMX\ mod\ 256$$

Die zweiten 8 bit des Pan-Wertes

$$Tilt\ Channel\ 1 = \frac{TiltDMX}{256}\ [muss\ abgerundet\ werden]$$

Die ersten 8 bit des Tilt-Wertes

$$Tilt\ Channel\ 2 = TiltDMX\ mod\ 256$$

Die zweiten 8 bit des Tilt-Wertes

Trivial.

Software

Android

Softwareseitig war die Aufgabe dieses Projektes, eine Android App mithilfe der neuartigen Programmiersprache Kotlin zu entwickeln. Um den uns selbst gesetzten Projektanforderungen gerecht zu werden, sollte die App zwei grundlegende Komponenten beinhalten.
Die erste Komponente ist das Spiel, das komplett auf dem Smartphone simuliert werden sollte. Hier musste also eine Gamelogik mit Kollisionsdetektion sowie einem globalem Highscoresystem in einem ansprechendem, zeitgemäßen sowie kontextuell passenden Design realisiert werden.
Die andere Komponente beschäftigt sich mit der Bluetooth Technologie. Hier wird der Aufbau sowie der konstante Datenaustausch mit dem HM10 Bluetoothmodul garantiert. Dafür ist der eigens geschriebene BluetoothService zuständig.

Der Spielablauf lässt sich in drei Phasen gliedern. Dementsprechend existieren drei Activities, die nacheinander aufgerufen werden. Die Menu-Activity, die als Hauptmenü fungiert, zeigt dem Spieler den Titelscreen, ermöglicht einen Spielstart, lädt die Hintergrundmusik und spielt sie repetitiv ab und baut beim Starten der App automatisch im Hintergrund die Bluetoothverbindung auf. Das Design des Titelscreens ist dabei an das Logo des Films *Catch me if you can* angelehnt und selbst mittels einer Custom – View erstellt worden. Nach Betätigen des Startbuttons wird die Spiel-Activity aufgerufen.
Diese ermöglicht es dem Spieler, einen DTS Jack Moving Head, nachfolgend mit MH abgekürzt, per virtuellen Joystick anzusteuern. Die drei anderen MHs bewegen sich rein zufällig. Die Positionen der MHs auf der Leinwand werden durch selbst erstellte Views auf dem Smartphone repräsentiert, welche allerdings im Ausstellungsmodus auf unsichtbar geschaltet sind. Im Hintergrund werden die Pixelkoordinaten jeder einzelnen MH - View mehrmals die Sekunde in DMX-freundliche Werte übersetzt und anschließend via Bluetooth an den Arduino Mega gesendet. Des Weiteren läuft ein sichtbarer Timer im Hintergrund ab, der das Spiel nach 45 Sekunden beendet. Außerdem bedient sich diese Activity auch

dem selbst geschriebenen VibratorService, damit der Spieler nicht nur visuelles, sondern auch haptisches Feedback bei der Kollision mit anderen MHs erfährt. Dann wird der aktuelle Score für den Spieler sichtbar entsprechend des Kollisionspartners verändert.
Die abschließende Spielende-Activity bekommt beim Aufruf den erspielten Score aus der vorherigen Activity übermittelt und zeigt diesen groß und mittig auf dem Screen an. Falls der erspielte Score höher als der globale Highscore sein sollte, übermittelt das Abspielen einer kurzen Audiodatei sowie ein speziell auf das Audio angepasster Vibrationseffekt ein Erfolgserlebnis. Ein Druck auf den Menu Button bringt den Spieler wieder zurück ins Hauptmenü.

Arduino

Auf Seite des Arduinos passieren kontinuierlich zwei Prozesse nacheinander. Zuerst werden über Bluetooth einkommende Daten ausgewertet. Das ist deshalb notwendig, da die Daten für die MHs als JSON Objekte verpackt gesendet und sie deshalb vom Arduino aufgelöst werden müssen. Alle Daten werden dann den jeweiligen Variablen jedes MHs zugewiesen. Darauffolgend kümmert sich die DMXSerial Bibliothek darum, die Werte dieser Variablen anschließend korrekt an die jeweils entsprechenden MHs zu übermitteln.

Reflexion

In Bezug auf das Konzept gab es bei der praktischen Umsetzung doch einige Differenzen und unerwartete Komplikationen. Unsere prinzipiell Idee die Koordinaten der Moving Heads in der App zu berechnen und zu simulieren war der richtige Ansatz. Doch das perfekte Mapping von Scheinwerfer auf der Wand der Simulationspunkte in der App war bis zum Schluss eine Herausforderung. Millimeterweises Verschieben der Jacks führt zu erheblichen Unterschieden der Postion der Lichtkegel auf der Wand. Doch mit ausreichend Zeit haben wir eine annehmbare Gesamtsituation geschaffen.

Die Idee eines Mehrspieler-Modus haben wir relativ schnell verworfen. Hierzu hätten wir ein zweites Smartphone und ein weiteres Bluetooth-Modul benötigt. Da es mit dem Bluetooth-Modul anfangs deutlich Schwierigkeiten gab, haben wir uns dann auf die Version mit mehreren zu fangenden Moving Heads („Mäuse“) konzentriert. Dies wurde auch in das finale Spiel eingebaut.

Wir hatten mehrere Ideen, den „Katzen“-Moving Head in der App zu steuern. Am Ende war der 360 Grad Joystick die intuitivste Steuerung. Der eigene Score wird ebenfalls auf dem Screen angezeigt. So hatten wir es uns schon im Konzept vorgestellt.

Auch ein einfacher Kalibrierungsmodus wurde eingebaut. Allerdings nur um die Überlagerung der Lichtkegel zu überprüfen. Dabei haben die Moving Heads die Ecken des Spielfelds angesteuert. Bei optimaler Ausrichtung haben sich die Lichtkegel in jeder Ecke überlagert. Bei Ungenauigkeiten musste die Korrektur der Scheinwerfereinstellungen danach händisch ausgeführt werden. Jedoch ist dieser Modus nicht direkt in die App implementiert sondern läuft in einem gesonderten Arduinocode. Für einen in die App integrierten Kalibrierungsmodus, welcher sich an neue Gegebenheiten anpasst und das Ausmessen des Aufbaus überflüssig macht, gab es theoretische Ansätze, die aber relativ schnell aufgrund ihrer Komplexität verworfen wurden.

Für das Plakat haben wir uns von dem Intro-Score des Films inspirieren lassen. Sowohl Farbe als auch Grafik und Schriftart wurden daran angelehnt. Diese Idee hatten wir auch im Konzept aufgenommen.

Unserem Zeitplan konnten wir nicht ganz wie geplant einhalten. Gerade die Bluetoothverbindung hat uns am Anfang viel Zeit gekostet.

Auch die mathematische Umsetzung der virtuellen Koordinaten in Lichtkegelkoordinaten musste einige Male optimiert werden, bevor es funktioniert hat. Anfangs hat zusätzlich die DMX-Ansteuerung Probleme gemacht, was sich aber nach einer recht einfachen Fehlerbehebung wieder korrigiert hat. Außer das für immer das Rätsel bestehen wird, warum man einmal im Setup einen Kanal, der höher ist als alle genutzten, einmal ansteuern muss.

Insgesamt haben wir dennoch alles geschafft, was wir angestrebt hatten und haben sogar eine Zusatzidee (mehr als 2 Scheinwerfer) eingebaut. Auch am Rundgang wurde das Spiel von einer großen Zahl Besucher gespielt mit positivem Feedback, was uns sehr erfreut hat. Wir denken, das Projekt ist uns gelungen und wir haben viele praktische Dinge daraus mitgenommen.

Anhang

Kanalbelegung

25-Kanal Modus (Standard)

1 PAN MSB
2 PAN LSB
3 TILT MSB
4 TILT LSB
5 SPEED MOVEMENT (Motorgeschwindigkeit)
6 PAN FAR (nur bei JACK FPR, Art.Nr. 03.MS012.EBLFP)
7 DIMMER
8 SHUTTER
9 FARBE
10 FARBMODUS
11 GOBO
12 GOBOMODUS
13 GOBO ROTATION / INDEX – PRISMA ROTATION
14 GOBO INDEX FEIN
15 GOBO SHAKE
16 FIXED GOBO
17 FIXED GOBO SHAKE
18 IRIS
19 MAKROS
20 MAKROS MODUS
21 PRISMA
22 FOCUS
23 FOCUS FEIN
24 ZOOM
25 RESET + LAMPENZÜNDUNG

Komplettes Handbuch und ausführliche Kanal erklärung unter:
http://light-control.de/lc-produktdatenbank/manuals/manualJACKde.pdf

Jette Heinsohn, Max Bauer
Dominik Jordan, Nele Schuster

Schaltplan und Steckplan