Hochschule für angewandte Wissenschaften Würzburg - Schweinfurt
Studiengang: Wirtschaftsinformatik

Sommersemester 2019

Projektarbeit

Dynamische Lichtshows für DMX Moving Lights

vorgelegt von: Benjamin Böhm, Simon Janssen, Elias Riedel
Matrikelnummer: 5015005, 5115102, 5115070

Abgabetermin: 30.09.2019

I. Inhalt

II. Abbildungsverzeichnis

1. Einführung

Ziel dieser Arbeit war es eine Software zu programmieren, die es ermöglicht eine Lichtshow virtuell in einer eigenen 3D-Umgebung zu erstellen. Diese erstellte Show soll dann die Befehle in der richtigen Zeitlichen Reihenfolge an die echten Hardwarekomponenten also die Lichter/Fixtures senden.

Um die Komplexität und um den zeitlichen Rahmen der Aufgabenstellung nicht zu sprengen, haben wir uns dafür entschieden im Rahmen dieser Arbeit nur die Fixtures zu unterstützen die auch in unserer Testumgebung dem Raum H.1.5 vorkommen.

Der User soll folgende Möglichkeiten haben eine Show zu erstellen. Es soll möglich sein die Bühnenumgebung exakt nachzubauen mit freiplatzierbaren Traversen und Lichtern/Fixtures. Der User soll in einer Timeline die einzelnen Befehle für die Lichter platzieren können. Zu den Befehlnooen zählen die Farbauswahl, sowie Funktionen. Auch soll ein Gruppierungsfunktion unterstützt werden.

2. Projektmanagement

Als Projektmanagementmethode verwendeten wir Scrum, allerdings nahmen wir einige Anpassungen an der Scrum-Vorgehensmethode vor, die uns für unsere Gruppenkonstellation sinnvoll erschienen. Wir wählten Scrum als Vorgehensmethode da diese es uns ermöglicht die Anforderungen während des Projektverlaufes einfach anzupassen.

Wir wählten zudem keine dem Scrum üblichen Rollenverteilung aus Product Owner, Entwicklungsteam und Scrum Master. Wir bestimmen lediglich einen Projektverantwortlichen, der Rest der Rollen wurde gemeinsam übernommen.

2.1 Projektmanagement Tool und Versionskontrolle

Als Projektmanagement Tool entschieden wir uns für „GitHub.com“, wir entschieden uns für dieses Projektmanagement Tool da dieses uns ermöglichte jeden Product-Backlog Eintrag bzw. jedes Feature als eigenen Issue anzulegen, diese Issues wurden dann mit passenden Labels(bug, concept, enhancement, help wanted, orga, question, TIME SENSITIVE) versehen und konnten einzelnen Personen oder

Personengruppen zugewiesen werden. Auch war es möglich Issues zu kommentieren und in anderen Issues zu referenzieren. Dies bot die Möglichkeit größere Issues auf mehrere Personen aufzuteilen und eine einfache Kommunikation über die Kommentarfunktion.

Auch ist es möglich Issues in einem Meilenstein zusammenzufassen, dieses Feature war essenziell für unsere Zeit- und Sprintplanung.

Zudem benutzten wir die Möglichkeit unseren aktuellen Vorschritt mittels eines Kanbanboards dazustellen.

Für die Versionskontrolle benutzten wir das von Unity mitgebrachte Tool: UnityCollab, da die Software in Unity programmiert wird, ist UnityCollab die einfachste und zuverlässigste, sowie fehlerfreiste Lösung.

Link zu unserem GitHub: https://github.com/simonHJanssen/DMX-UI

2.2 Product-Backlog / Featureliste

Die Anforderungen an unsere Software hielten wir im Product-Backlog fest bzw. nannten wir es auch Featureliste.

Wir benutzten hier keine Userstorys oder ähnliches, ein Eintrag besteht hier aus: Featurename, Featurebeschreibung und Testprotokoll, sowie einer von Github vergebenen ID. Diese Beschreibungen finden sich alle im GitHub wieder, da dort jeder Anpassungen vornehmen konnte.

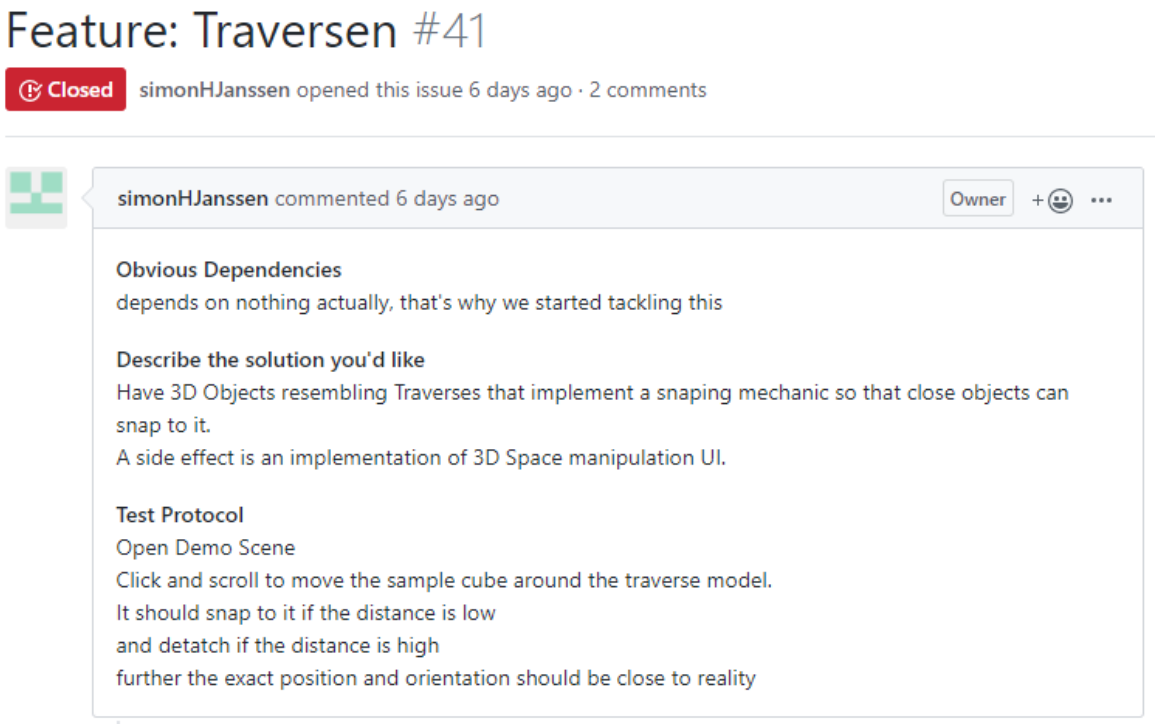

Abbildung 1: Beispielfeature

Für das Product-Backlog sammelten wir zuerst Anforderungen. Diese wurden dann zusammen in Gruppen kategorisiert und mit einem geschätzten Arbeitsaufwand bewertet. So entstanden 4 Gruppen.

Die Features aus der Gruppe „Absolute minimum“ wurden alle übernommen und sollten in der fertigen Software enthalten sein.

Aus der Gruppe „Actually a nice product“ sollten die Features:

- Room traverses snap function for fixtures
- Help button tutorial (Funktion, Wiki, Videos)

Umgesetzt werden.

Aus den restlichen Gruppen sollten keine Features übernommen werden.

Absolute minimum:

DMX control via Artnet (4)
2D Rotation (4)
Filter wheels, gobo wheels, light source colors (16)
different fixtures for different channel modes (3)
Time dynamic with scrubable timeline (100)
fixture grouping (40)
Filter Selections (4)
Alignment of the fixtures to 3D room positions (20)
Save Configs (12)

Actually a nice product:

Hotkeys for easy usage (12) Layering (32)
Position grouping (20)
Room traverses snap function for fixtures (12)
Functions on positions (depth limitation for staking of functions) (50)
Color Management (16)
Help button tutorial (Function, Wiki, Videos) (8)

Cooperation with other project group?:

Person position tracking? (X)

Stretch Goals:

Describe parameter animation by function (50)
Default function/example scene (12)
MIDI Clocks (20)
Joystick support (12)
Group effects (mirror duplicate...) (10+)

Abbildung 2: Gesammelte Anforderungen

Die daraus resultierenden Meilensteine, die Meilensteine hatten jeweils eine grobe Terminierung:

0.1 Usable

Past due by 5 days 50% complete

The software is fit for use in H.1.5 while not all features are implemented

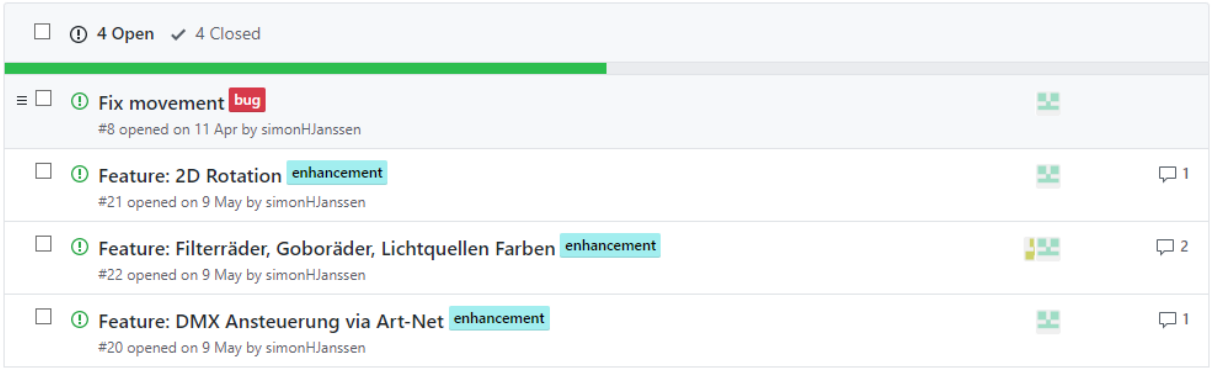

Abbildung 3: Erster Meilenstein

„0.1 Usable“ – „The software is fit for use in H.1.5 while not all features are implemented“

0.2 Good

Past due by 4 days 25% complete

On top of being fit for use for H.1.5 the product could be used to a certain extend ba a amature user, not all is documented, fixed or implemented but the main features are working and accessible

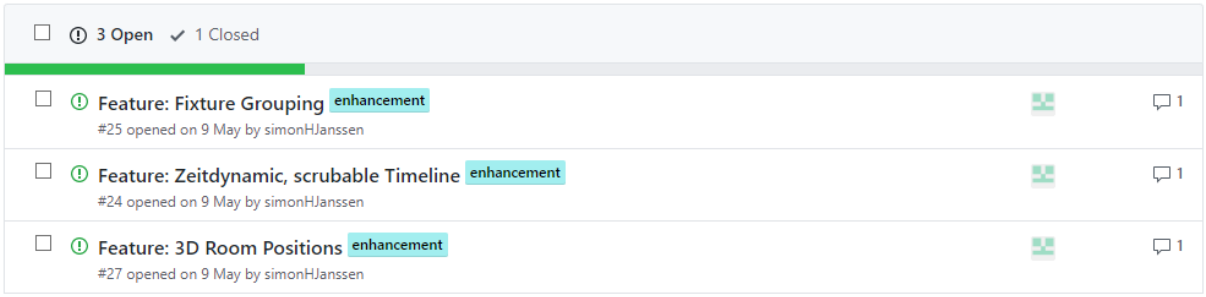

Abbildung 4: Zweiter Meilenstein

„0.2 Good“ - „On top of being fit for use for H.1.5 the product could be used to a certain extend by an amature user, not all is documented, fixed or implemented but the main features are working and accessible“

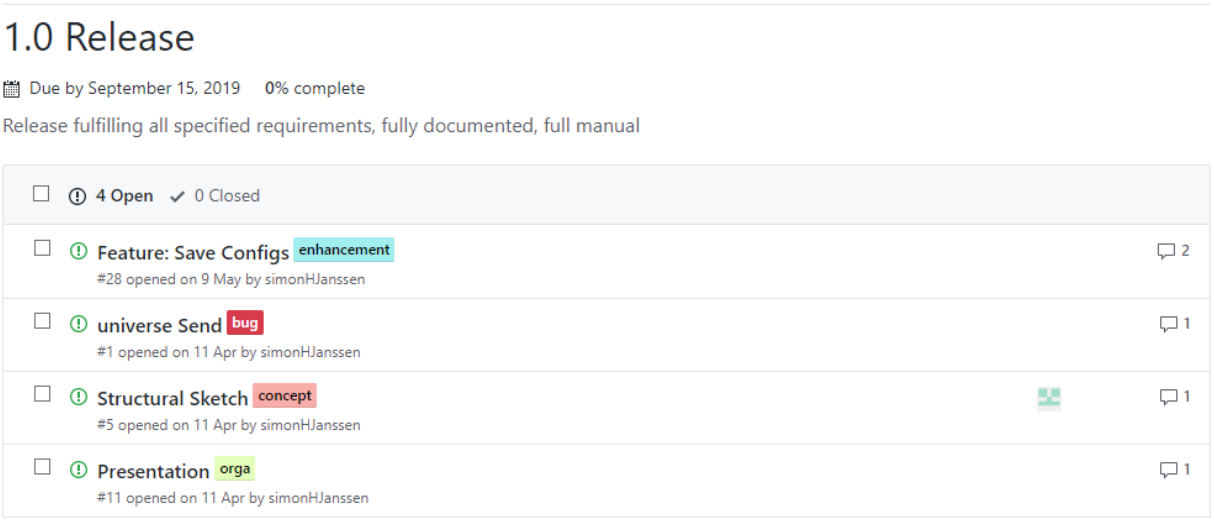

Abbildung 5: Letzter Meilenstein

„1.0 Release“ – „Release fulfilling all specified requirements, fully documented, full manual“

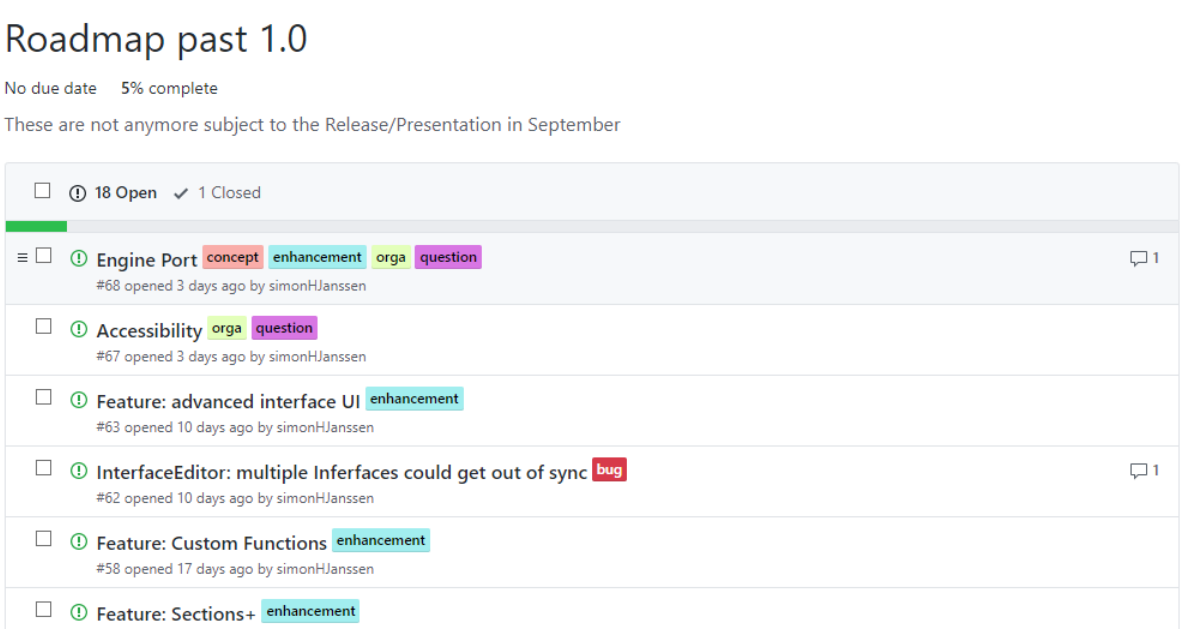

Abbildung 6: Ablage Meilenstein

In „Roadmap past 1.0“ – „These are not anymore subject to the Release/Presentation in September.“

Für den Feature Lifecycle und den Milestone Lifecycle wurden 2 Prozessabläufe erstellt. Diese Prozesse beschreiben den vorgeschriebenen Ablauf von der Erstellung eines Features bzw. Milestones in Github bis zum schließen dieses Features.

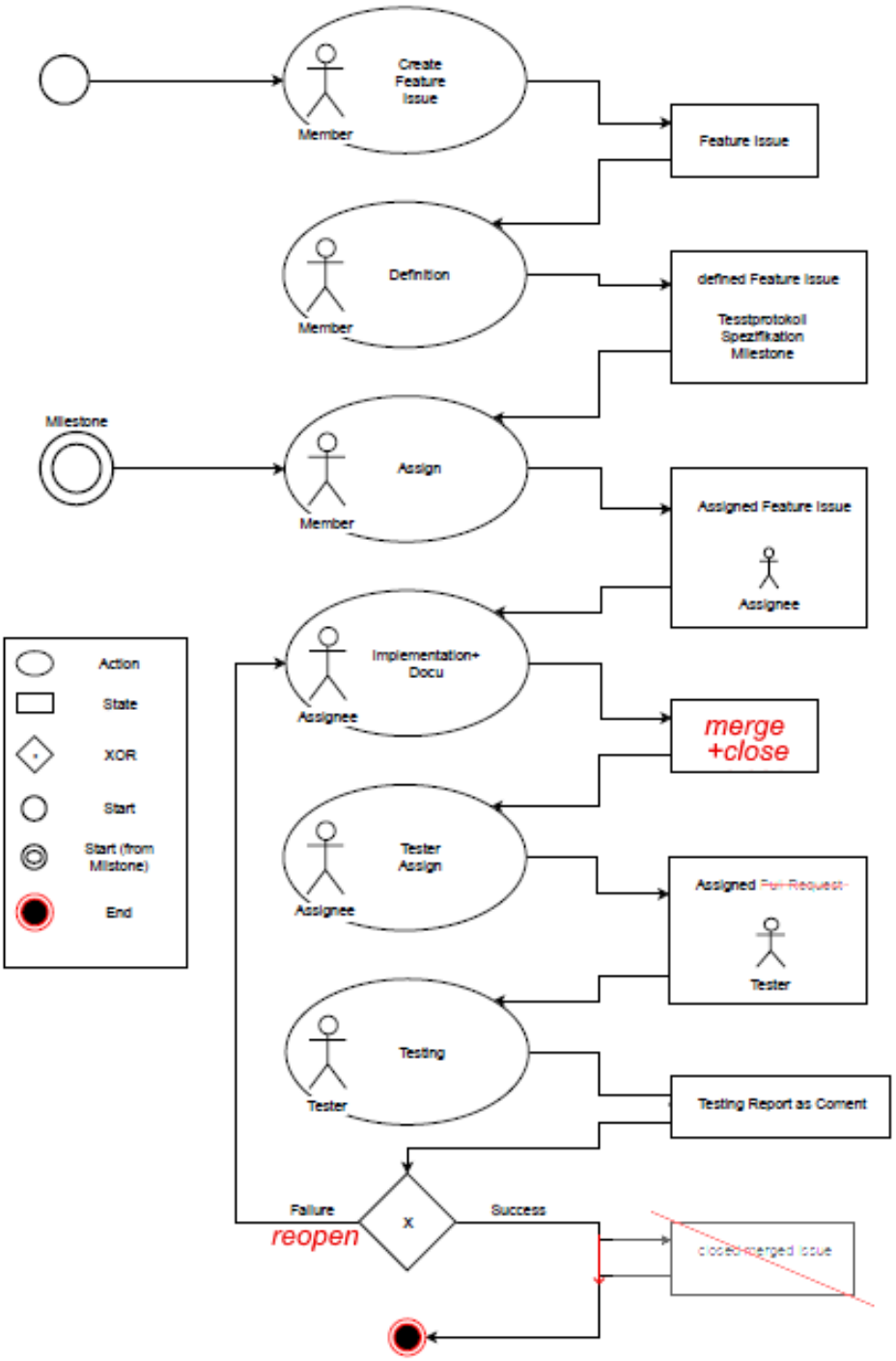

Abbildung 7: Feature Lifecycle

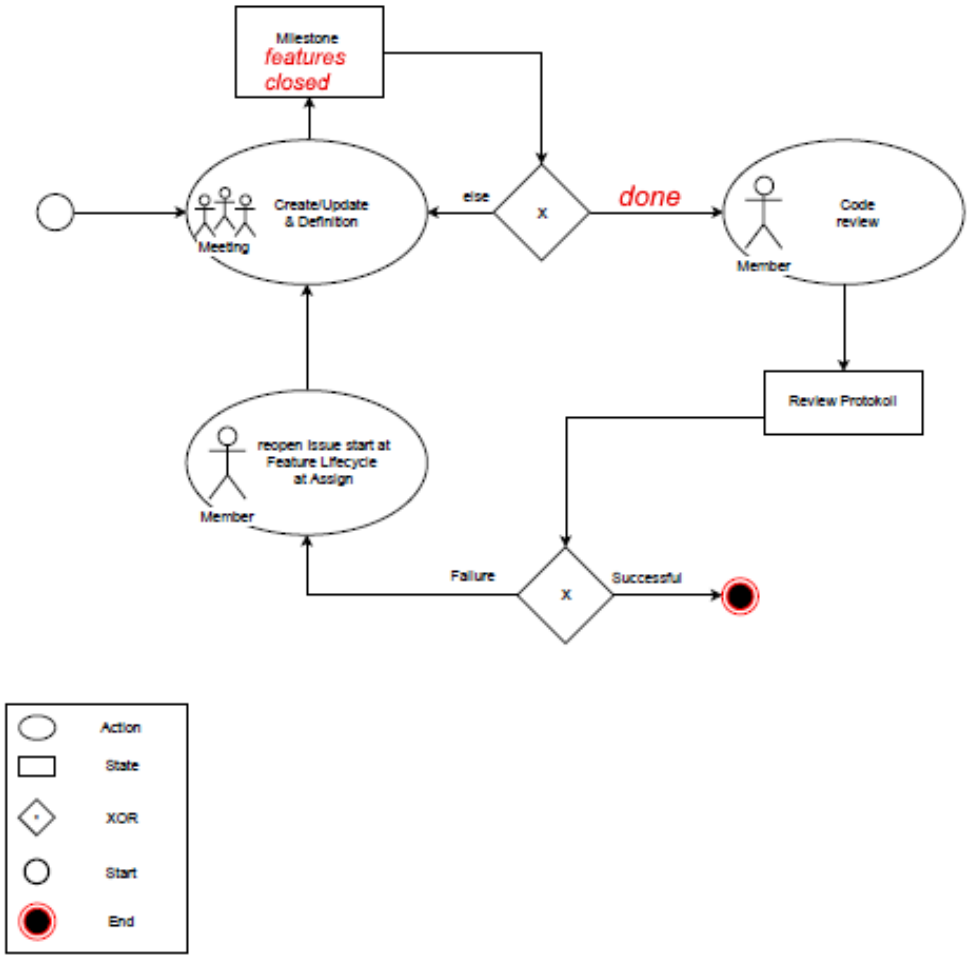

Abbildung 8: Milestone Lifecycle

Weitere Anforderungen, die wir erfüllen wollten, waren Anforderungen, die sich durch Gespräche mit Brancheninternen Leuten ergaben, hier zu erwähnen wären.

- Stabilität während dem Programmierungsprozess -> Konsequenz stabiles Framework
- Bühne semantisch zu programmieren wurde als gut befunden

Weniger wichtige Anforderungen allerdings mit weniger Priorität

- Benutzerfreundlichkeit
- Hardwarezuverlässigkeit und Format
- Kompatibilität der Fixtures

2.3 Sprint Planung

Nachdem die Vorbereitungen bzw. die Projektplanung (11.04 – 03.06) abgeschlossen waren planten wir unsere Sprints und deren Terminierung. Wir planten unsere Sprints in die Monate August und September ein. Bis zu dem Anfang der Sprints wurden noch einige Vorbereitungen (03.06 – 26.08) für die Sprints getroffen z. B. UI-Controller. Für die zeitliche Terminierung nutzen wir die Aufwandsabschätzungen. Geplant war es die Meilensteine nacheinander in Sprints abzuarbeiten, während der Vorbereitungen viel uns jedoch auf, dass es sinnvoller ist die Aufgabenverteilung pro Sprint dem Projektleiter zu überlassen, da dieser den besten Überblick hatte, wann welches Feature fertig sein musste, bzw. welcher Features zuerst realisiert werden müssen. Die dann genaue Aufgabenverteilung innerhalb eines Sprints übernahm der Projektleiter, da dieser den besten fachlichen Überblick hatte.

Den ersten Sprint planten wir vom Zeitraum 26.08 bis 08.09 ein.

Für diesen Sprint waren folgende Features geplant:

- 2D - Rotation
- Filterräder, Goboräder, Lichtquellen Farben
- 3D Room Positions
- Fixture Grouping
- Traversen snapping
- Fixtures and timeline state

Den zweiten Sprint planten wir vom 09.09 bis 15.09

Für diesen Sprint waren folgende Features geplant:

- Movement Animation
- Zeitdynamic, scrubable Timeline

Den letzten Sprint setzten wir vom 16.09-22.09 an.

- Debugging
- Testing

Die Woche vom 23.09-29.09 wurde als Notfallpuffer eingeplant.

2.4 Sprint-Backlog

Um den aktuellen Bearbeitungsstatus eines Sprints zu visualisieren benutzten wir eine Kanban-Tafel diese konnte man in GitHub erstellen.

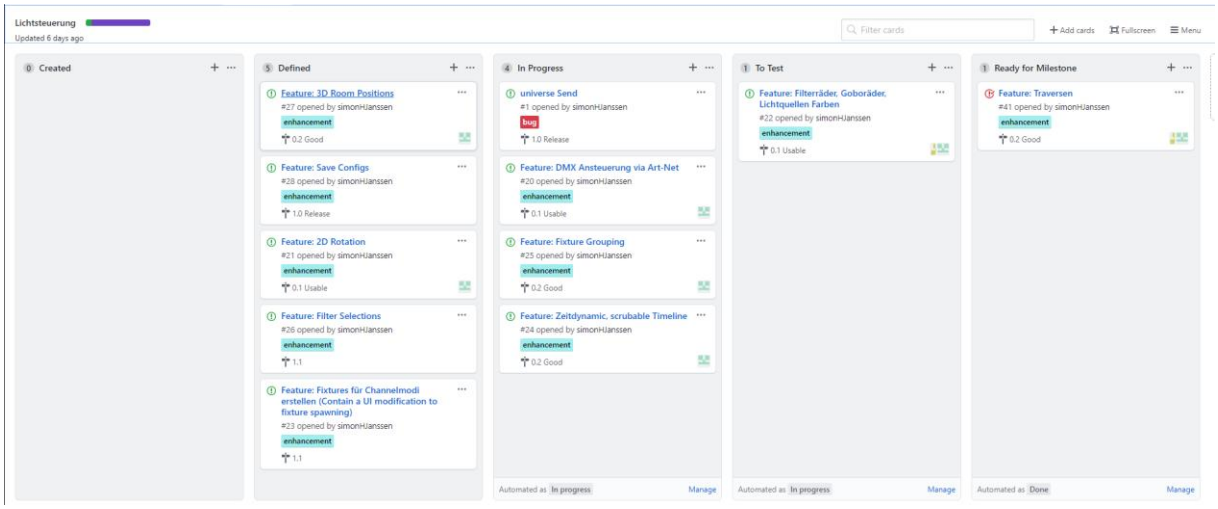

Abbildung 9: Kanbanboard

Unsere Kanban-Tafel bestand aus 5 Spalten.

- Created – Das Feature wurde als Issue erstellt.
- Defined – Das dem Sprint zugewiesene Feature wartet auf Bearbeitung
- In Progress – Das Feature wird aktuell bearbeitet
- To Test – Das Feature ist implementiert und wartet darauf anhand dem Testprotokoll getestet zu werden.
- Ready for Milestone – Das Feature ist fertig implementiert und getestet und kann geschlossen werden.

2.5 Daily Scrum und Sprint Review

Daily Scrum

Aufgrund der kleinen Gruppenkonstellation entschieden wir uns von dem üblichen Daily Scrum Meeting während eines Sprints abzuweichen und das ganze Dynamisch zu machen, trotzdem versuchten wir spätestens alle 3 Tage ein Zwischenstands-Meeting zu machen um schnell auf Probleme reagieren zu können. In den Meetings wurde der Fortschritt besprochen und über aufkommende Änderungen disskutiert.

Sprint Review

Sprint 26.08-8.09 – Rückstand von geschätzten 3 Tagen festegestellt.

Sprint 9-15.09 – Rückstand von 1 Woche festegestellt, Testen wird in die Pufferwoche verschoben

Sprint 16-22.09 – Rückstand aufholen, beenden der Programmierarbeiten

Puffersprint 23-29.08 – Testing, Abschlussarbeiten am Projekt

Folgende Grafik veranschaulicht das Management der Features nach einem Sprint:

Features/Sprints	26.08 - 08.09	09.09 - 15.09	16.09 - 22.09	23.09 - 29.09
2D - Rotation	Geplant	Übernommen	Übernommen	Übernommen
Filterräder, Goboräder, Lichtquellen Farben	Geplant	Übernommen	Übernommen	Übernommen
3D Room Positions	Geplant	Übernommen	Übernommen	Übernommen
Fixture Grouping	Geplant	Übernommen	Übernommen	Übernommen
Traversen	Geplant			
anti delay show mode	Neu + zu Aufwenig			
RGB+ Fixtures	Neu + zu Aufwenig			
Preprogrammed	Neu + zu Aufwenig			
Mixed Fixtures	Neu + zu Aufwenig			
Animation color prevent disjoin	Neu + zu Aufwenig			
Color Space correction	Neu + zu Aufwenig			
Filter Selections	zu Aufwenig			
Fixtures für Channelmodi erstellen	zu Aufwenig			
Zeitdynamic, scrubable Timeline		Geplant	Übernommen	Übernommen
Custom Functions		Neu + zu Aufwenig		
Selections+		Neu + zu Aufwenig		
Clocks		zu Aufwenig		
Volumetric Light		Neu + zu Aufwenig		
Camera Control			Neu + bearbeitet	
3D GUI Manipulation			Neu + bearbeitet	
advanced interface UI			Neu + zu Aufwenig	
First Steps Video Series				Geplant
Save Configs				Zu Aufwendig

Abbildung 10: Sprint-Review Übersicht

Legende:	
	Feature war geplant
	Feature kam neu
	Feture kam neu hinzu und wurde als zu Aufwendig betrachtet
	Feture war geplant und wurde als zu Aufwendig betrachtet
Geplant	Feature war in diesem Sprint geplant
Neu + bearbeitet	Feature kam neu hinzu und wurde in diesem Sprint bearbeitet
Neu + zu Aufwenig	Feture kam neu hinzu und wurde als zu Aufwendig betrachtet
zu Aufwenig	Feture war geplant und wurde als zu Aufwendig betrachtet
Übernommen	Feature wurde im geplanten Sprint nicht fertig und wurde übernommen

Abbildung 11: Legende zu Abbildung 10

2.6 Test Strategie – Exploratory Testing

Während dem Projektverlauf entschieden wir uns unsere Teststrategie zu ändern. Wir entschieden uns nun für das Exploratory Testing als neue Teststrategie, da wie so unsere Software bestmöglich Testen können und sich diese Teststrategie gut mit Scrum vereinbaren lässt.

Softwaretests sind ein wichtiger Prozess in einem Softwareentwicklungslebenszyklus als Validierungs- und Verifizierungsmechanismen, um die Qualität des beabsichtigten Softwareprodukts zu gewährleisten. Exploratives Testen ist ein Software-Test, bei dem die Tester mit dem System beliebig interagieren und die Informationen, die das System liefert, nutzen können, um zu reagieren und die Systemfunktionalitäten im Allgemeinen uneingeschränkt zu untersuchen. Durch explorative Tests kann die volle Kraft des menschlichen Gehirns genutzt werden, um Fehler zu finden und die Funktionalität ohne vorgefasste Einschränkungen zu überprüfen. [1, p. 280]

2.7 Lessons Learned

Beim bearbeiten des Projektes sind uns folgende schwächen in der Planung aufgefallen:

- Aufwandsabschätzungen viel zu gering, der tatsächliche Aufwand für die Implementation eines Features war ca. 3x so hoch
- Die angelegten Features waren zu grobgranular
- Mehr bzw. bessere Meilensteine definieren, die sich mehr an den Programmierablauf richten und mehr aufeinander aufbauen

3. Anforderungen des technischen Designs

Um eine gute semantische Abstraktion für den Endnutzer zu erreichen und sowohl Erweiterbakeit als auch Verständlichkeit zu gewährleisten, muss vor Allem das technische Design der Anwendung die einzelnen Bausteine modular gut abbilden. (siehe z.B. [2])

3.1 UML Diagramm (angepasst)

Die Kernarchitektur lässt sich anhand der Abbildung 12: angepasstes UML überschauen.

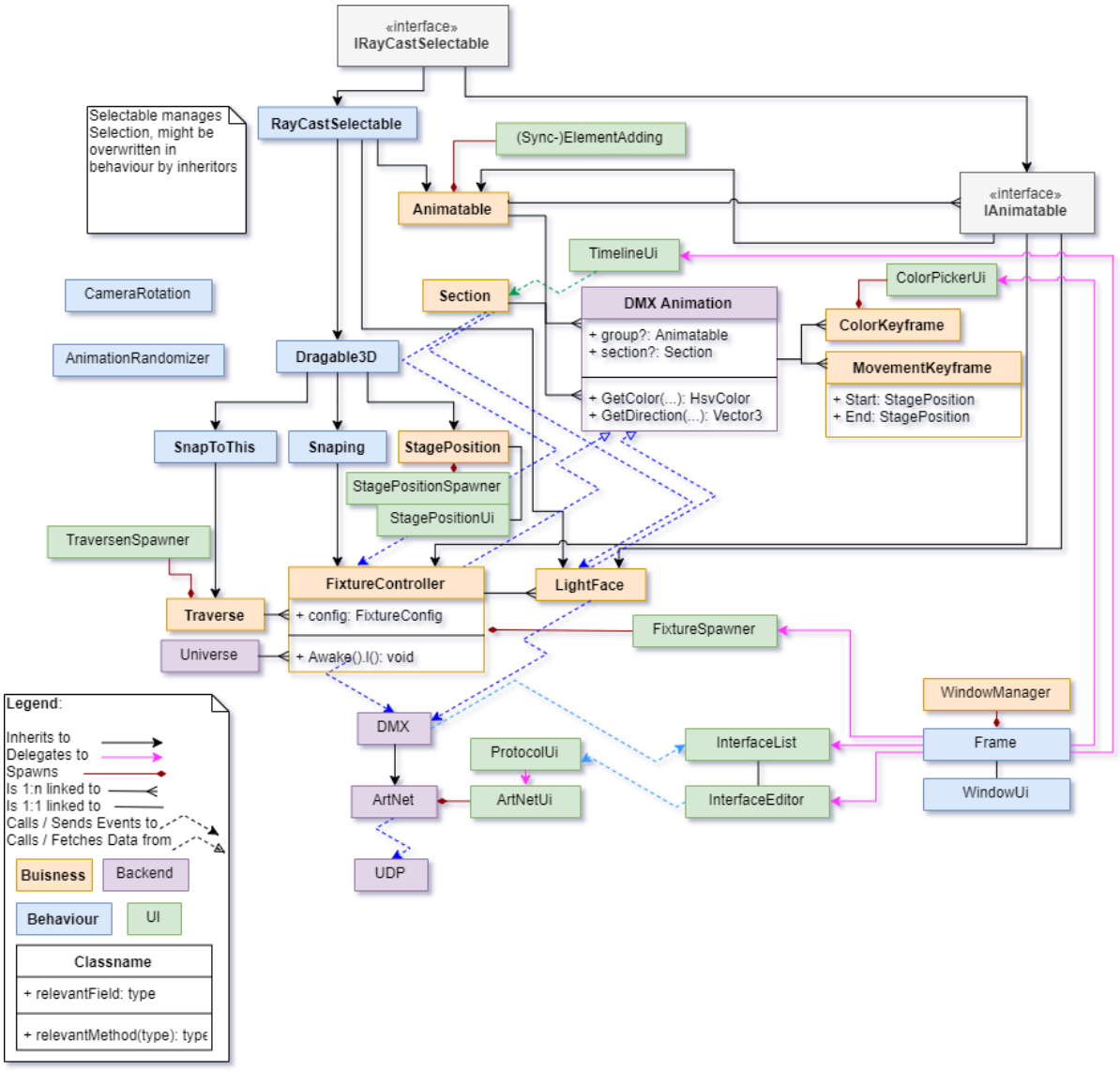

Abbildung 12: angepasstes UML

Diese zeigt sowohl den Vererbungsbaum durch den das allgemeine Verhalten im drei dimensionalen Raum implementiert ist, als auch den Ablauf der wichtigsten Ereignisse und die Beziehungen zwischen verschiedenen eigenen Datentypen.

Wie Sie sehen können, gibt es bei einigen Klassen keine klare Abstraktion der Verantwortung der Datenhaltung und deren Repräsentation dem Nutzer gegenüber. Dieser Architekturstil wird duch das Unity Framework leider gefördert. Dessen modernerem Design Ansatz („ECS“) ist uns leider erst zu spät bekannt geworden, um ihn in diesem Projekt noch anzuwenden.

3.2 Semantische Abstraktion

Die Anwendung soll ein Anwendungskonzept für DMX Lichtsteuerungen vorstellen, dass in der Lage ist, komplexere Programmierungen v.A. für Licht und Bewegung durch einfache Assoziationen zu modellieren. Diese semantische Abstraktion hat einige Herausforderungen zur Folge. Die besonderen Herausforderunen der Farb und Bewegungsanimation sind in den nächsten beiden Punkten erläutert.

3.3 Farbübergänge

Eine der Problemstellungen bei der Umsetzung des Projektes war es, Farbübergänge fließend darzustellen. Ein momentaner Zustand kann auch ein Bereich von Farben sein.

Praktisch ist dies analog zu einer Gruppe Fixtures, die eine Reihe an Farben annehmen soll. Der Anfangszustand ist hier durch „color“ und „ordercoloroffset“ sowie dem Farbraum (oben links) definiert.

Dieser Farbbereich soll über die Zeit auf den neuen Bereich von „next.color“ bis „next.orderoffset“ abgebildet werden.

Die Verschiebung eines einzelnen Punktes in einer Farbgruppe ist also ein drei dimensionales Problem. Dazu hat der Farbraum in dem die Animation durchgeführt wird auswirkungen auf den jeweiligen momentanen Zustand.

Um eine nutzerfreundliche und effektive Lösung zu implementieren, versuchen wir dieses Problem auf drei zweidimensionale Probleme herunter zu brechen: den Zustand der Gruppe am Anfang und am Ende und die Verschiebung dazwischen entlang einer festgelegten Route.

Damit diese Route möglichst allgemein ästhetisch aussieht wäre der Übergang auf Abbildung 14: Saubere Animation durch den HSV Farbraumideal.

Unsere Implementierung ist dadurch limitert, dass sie für das Einzelne Element den kürzeren / längeren Pfad verwendet, statt die ganze Gruppe zu animieren. (Abbildung 13: Einschränkung unseres Ansatzes)

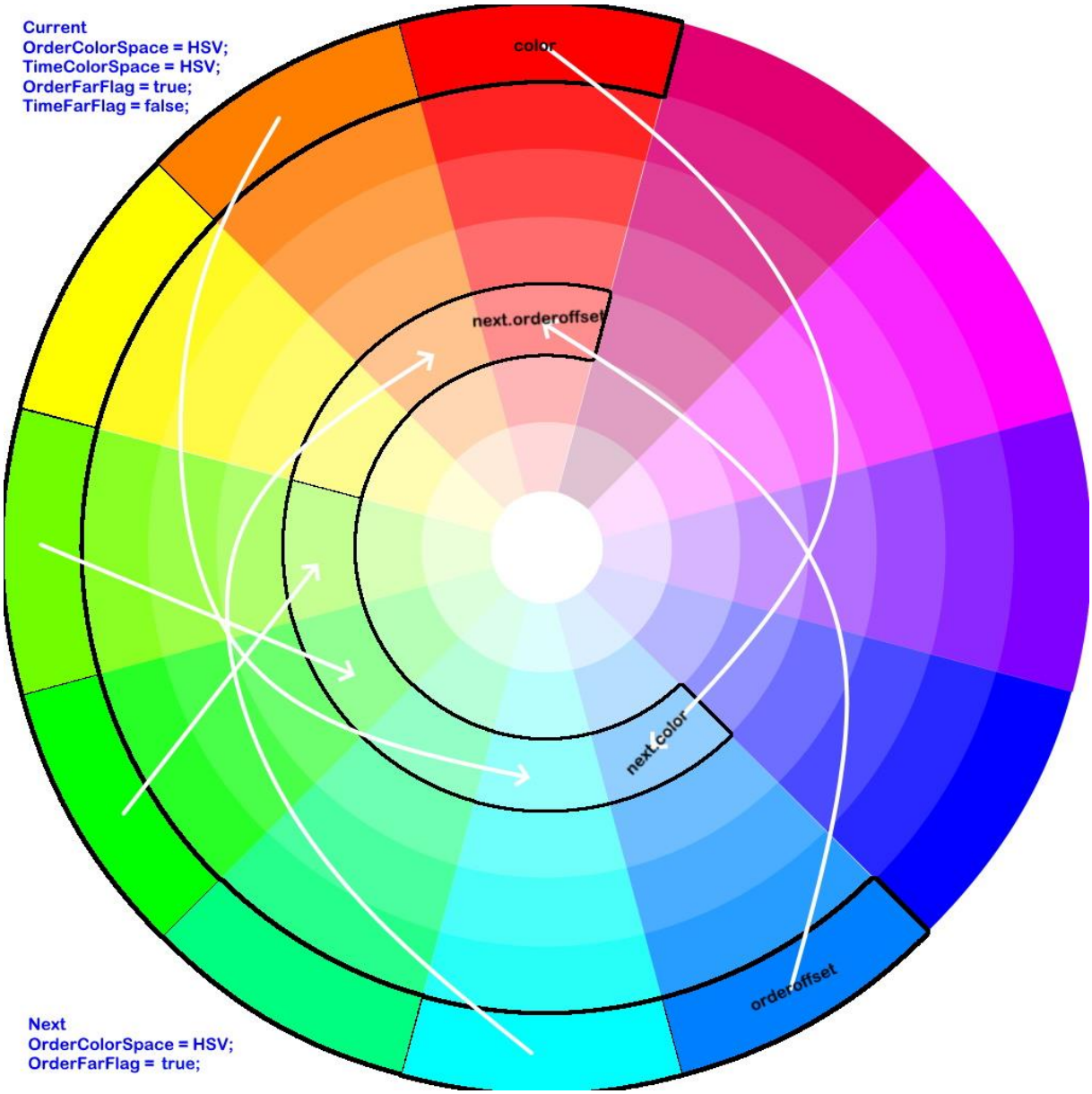

Abbildung 13: Einschränkung unseres Ansatzes

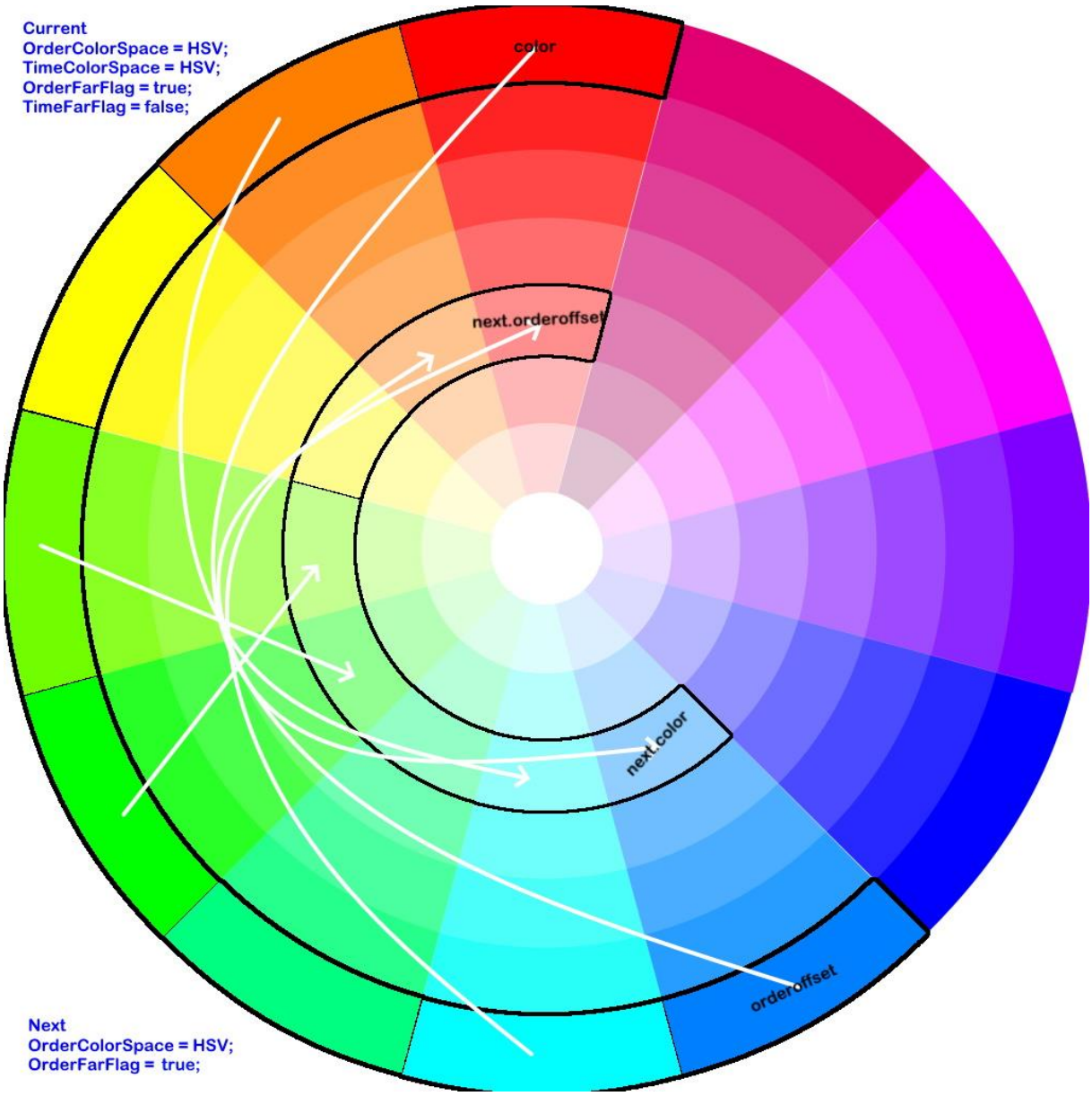

Abbildung 14: Saubere Animation durch den HSV Farbraum

3.4 Stage Positions

Um Bewegung einer Fixture zu Abstrahieren wird dem Nutzer die Möglichkeit gegeben Positionen in der Welt zu definieren, zwischen denen dann animiert werden kann.

Es wird ein neuer Typ eingeführt, um

a) eine Manipuierbare Darstellung in der Szene zu haben und
b) einen Offset der Position durch 3 Modi zu unterstützen, die oft verwendete Muster implementieren. ([3] zeigt z.B. die Inverted Mirrored Modus Abstraktion im Einsatz, Abbildung 15: Inverted Mirrored StagePosition das theoretische Modell)

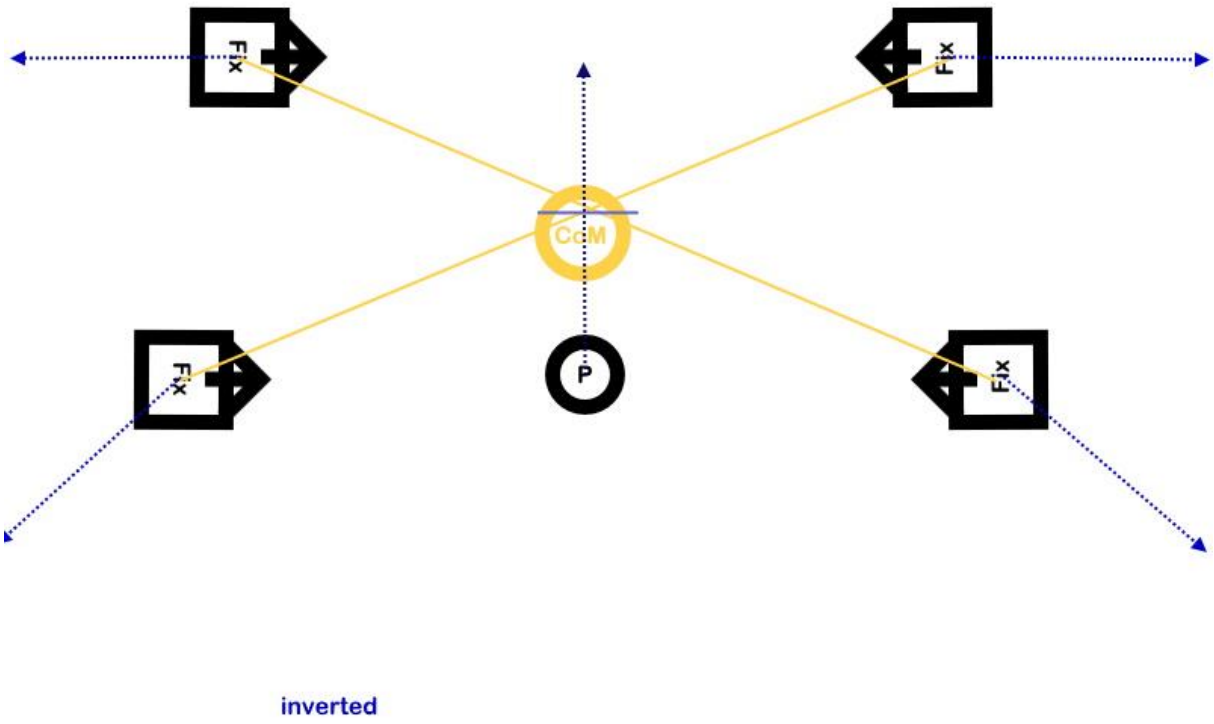

Abbildung 15: Inverted Mirrored StagePosition

4. User Interface Desgin

Für das Entwicklung des User-interfaces erstellten wir in der Planungsphase in Gruppenarbeit einen UI-Scetch:

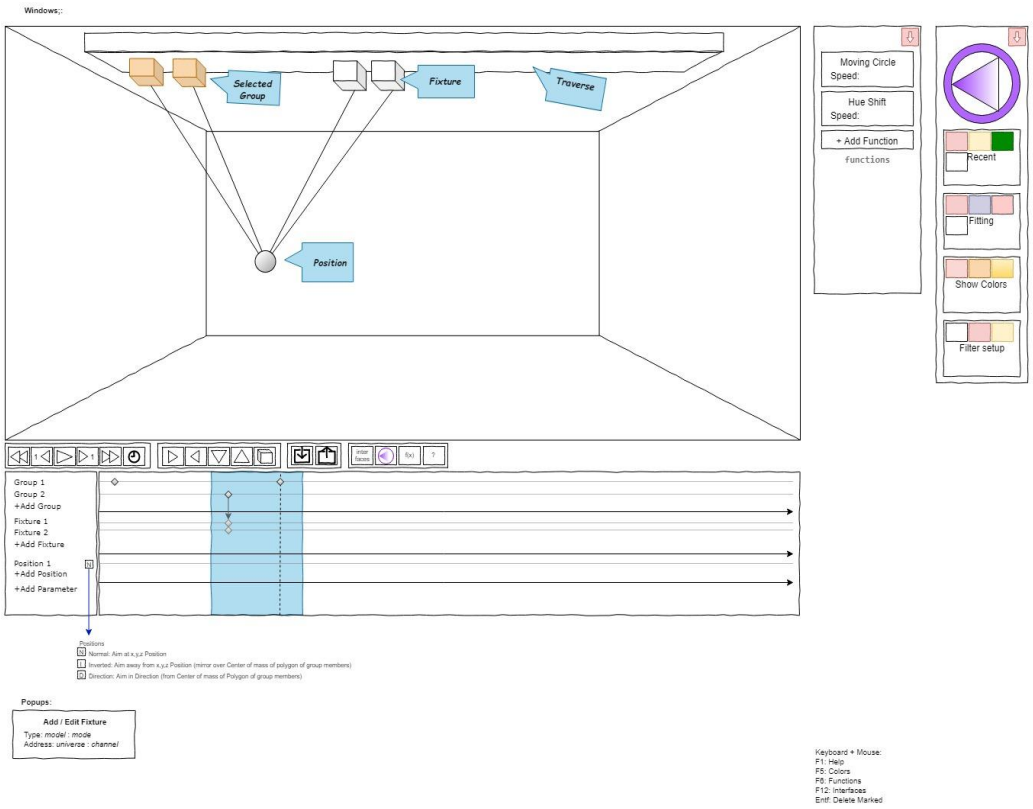

Abbildung 16: UI-Scetch

Dieser beinhaltete eine Grobe Darstellung für die Umsetzung der einzelnen Features in einer UI.

Für die Timeline sollten folgende Buttons, von links nach rechts beschrieben, erstellt werden:

Back to first frame: Auf Klick dieses Buttons wird zurück zum ersten Frame gesprungen

Back one frame: Auf Klick dieses Buttons wird zurück zum nächsten Keyframe in der Timeline gesprungen

Play: Auf Klick dieses Buttons wird die Schow abgespielt

Forward one frame: Auf Klick dieses Buttons wird in der Timeline zum nächsten Frame gesprungen

Clock (Stretchgoal): Auf Klick dieses Buttons wird ein Listener auf MIDI-Events kreiert um den Schowablauf darauf zu synchronisieren

Viewbuttos (Left, Right, Top, Dow, Default): Auf Klick dieser Buttons wird die Sicht auf die Bühne auf die jeweilige Position gesetzt

Import: Auf Klick dieses Buttons wird eine Auswahlfenster geöffnet, um eine zuvor gespeicherte Schow zu öffnen

Export: Auf Klick dieses Buttons wird eine Auswahlfenster geöffnet, um die geöffnete Schow ab zu speichern

Traverse: Auf Klick dieser Buttons wird eine neue Traverse kreiert

Colorpicker: Auf Klick dieser Buttons wird der Colorpicker für den ausgewählten Keyframe geöffnet

Funktion: Auf Klick dieser Buttons wird das Funktionsfenster für die ausgewählten Keyframes geöffnet

Help: Auf Klick dieser Buttons wird ein Tutorial für den Umgang mit dem Programm geöffnet.

3-dimensionaler Raum

Die Traversen und Fixtures sind im 3D Raum frei bewegbar.

Durch die Zuweisung zu einer Position, dargestellt durch eine Kugel im 3D Raum wird die Ausrichtung der Fixtures zu einem bestimmten Zeitpunkt beschrieben

UI – Timeline

Add Fixture: Auf Klick dieses Buttons wird ein neues Fixture erstellt

Add Group: Auf Klick dieses Buttons werden die Ausgewählten Fixtures gruppiert um synchron auf eine Position zugewiesen werden zu können.

Add Position: Auf Klick dieses Buttons wird eine neue Position für die ausgewählte Fixture oder Gruppe eine Stageposition festgelegt, auf die sie sich ausrichten

Rechtsklick auf Timeline: Ein Menü für das setzen eines Keyframes wird geöffnet

Linksklick auf Keyframemarker: Keyframe wird ausgewählt und markiert

Rechtsklick auf Keyframe: Der Keyframe wird ausgewählt und Ein Menü zum Löschen des Keyframes wird geöffnet.

Im Laufe des Projekts wurde die UI ensprechend den Anforderungen und den Umsetzungsmöglichkeiten angepasst.

Hierfür wurden Der Clock-Button mitsamt der dahinterstehenden Funktionalität entfernt aus Gründen der Umsetzung im zeitlich vorgegebenen Rahmen, ebenso wie die Buttons Export und Import.

Die Buttons Colorpicker und Funktion wurden entfernt da die enstprechenden Fenster sich immer öffnen sollten, wenn eine Gruppe oder ein Fixture ausgewählt wird.

Rechtsklick auf einen leeren Frame in der Timeline wurde entfernt da der entsprechende Workflow überarbeitet wurde. Ein Keyframe wird nur auf einen ausgewählten Frame gesetzt, wenn der Color oder der entsprechende Positionswert geändert wird.

Es wurde außerdem noch eine Scrollbar mit zwei Inputfeldern oberhalb der Timeline hinzugefügt mit zwei Inputfeldern.

Die Scrollbar ist für das horizontale Scrollen der Timeline zuständig. Das linke Inputfield gibt die Länge der Schow in Frames an. Das rechte Inputfield den derzeit ausgewählten Frame

Auch neben den Buttons Add Fixture und Add Group wurde je eine Scrollbar hinzugefügt, die für das vertikale Scrollen durch die Timeline der verschiedenen Fixtures/Gruppen zuständig ist. Dies dient dem einfachen auswählen der zu bearbeitenden Fixtures/ der Gruppe.

III. Quellenverzeichnis

[1] B. Suranto, “Exploratory software testing in agile project,” in *2015 International Conference on Computer, Communications, and Control Technology (I4CT)*, Kuching, Sarawak, Malaysia, Apr. 2015 - Apr. 2015, pp. 280–283.

[2] WIRFS-BROCK, Rebecca; WILKERSON, Brian; WIENER, Lauren. *Designing object-oriented software*. 1990.

[3] Foto: Igor Bezborodov https://www.freiepresse.de/sport/starladder-cs-go-major-in-berlin-geht-in-entscheidende-phase-artikel10598591 2019