Audio-Video-Programmierung

Prof. Dr. Plaß & Sudau

WiSe 17/18

https://github.com/irenabecker/AVPrg\_DIY3DSoundkulisse

SoundSpace

Takes you wherever you want to be.

Carina E. Krafft | 2269579  
Jordanis Lazaridis | 2142772  
Irena Becker | 2238833  
Kevin Hagen | 2270985

17.01.20178  
Hochschule für angewandte Wissenschaften Hamburg  
Fakultät Design, Medien und Information  
Department Medientechnik

Inhalt

[Einleitung 2](#_Toc504060981)

[Back-End Entwicklung 2](#_Toc504060982)

[Form-Erkennung 2](#_Toc504060983)

[Farb-Erkennung 2](#_Toc504060984)

[Kalibrierung und Positionsbestimmung in 3 Dimensionen 3](#_Toc504060985)

[Front-End Umsetzung 3](#_Toc504060986)

[Interface 3](#_Toc504060987)

[WebAudio 3](#_Toc504060988)

[Schnittstellenprogrammierung 4](#_Toc504060989)

[Herausforderungen 4](#_Toc504060990)

[Reflexion: Umsetzung der User-Stories 4](#_Toc504060991)

# Einleitung

„SoundSpace“ ist ein tangible-Sound Projekt, bei dem es dem Nutzer möglich ist durch Objekte verschiedener Formen und Farben seinen eigenen Sound Space zu erzeugen. Dafür können Würfel, Kugeln und dreiseitige Pyramiden in den Farben Rot, Blau und Grün frei in einem kalibrierten Bereich platziert werden. Hierbei steht jedes Objekt für einen eigenen Sound, der vom Benutzer beliebig über ein Webinterface angepasst werden kann.

* Verschiedene Shapes gleicher Farbe und Form müssen voneinander unterschieden werden
* UML-Klassen- und Sequenzdiagramm am Ende

Das Projekt „SoundSpace“ wurde im Rahmen der Vorlesung Audio-Video-Programmierung realisiert. Eine Projektvoraussetzung war daher die Verwendung der, in der Vorlesung behandelten, Programmiersprachen C++, unter Verwendung der Entwicklungsumgebung QT und der Bibliothek openCV, und JavaScript, in Kombination mit der WebAudio API.

Die Entwicklung des Projektes wurde grundlegend in die Bereiche Backend- und Frontend- und Schnittstellen- Entwicklung unterteilt, sodass an allen Bereichen unabhängig voneinander gearbeitet werden konnte.

# Back-End Entwicklung

Das Back-End von „SoundSpace” wurde in C++, unter Verwendung der Entwicklungsumgebung QT und der Bibliothek openCV, umgesetzt.

Ziel des Back-Endes war es ein eingespeistes Kamerabild zu analysieren und dadurch die dreidimensionalen Positionen aller einzelnen Objekte, sowie deren Farbe und Form, erkennen und ausgeben zu können.

## Form-Erkennung

Da es für „SoundSpace“ wichtig war mehrere Objekte gleicher Farbe, und möglicherweise gleicher Form, voneinander unterscheiden zu können, musste die Erkennung der einzelnen Objekte primär über die Form geschehen. Hierfür war es notwendig, im zu analysierenden Bild, möglichst eindeutige Formen darzustellen. Um dies zu gewährleisten, hat die Kamera aus der Vogelperspektive das Geschehen gefilmt und als Objekte wurden eine Kugel, eine dreiseitige Pyramide und ein Würfel gewählt.

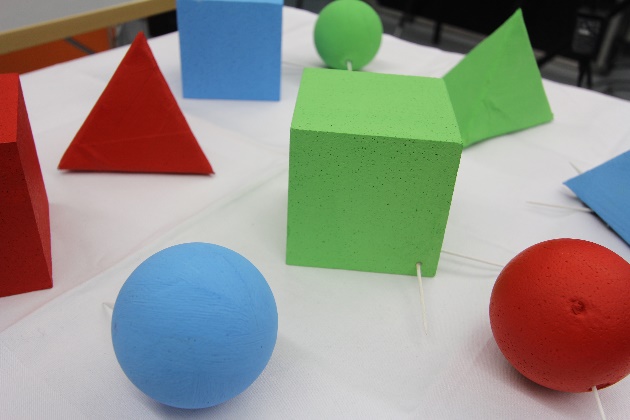
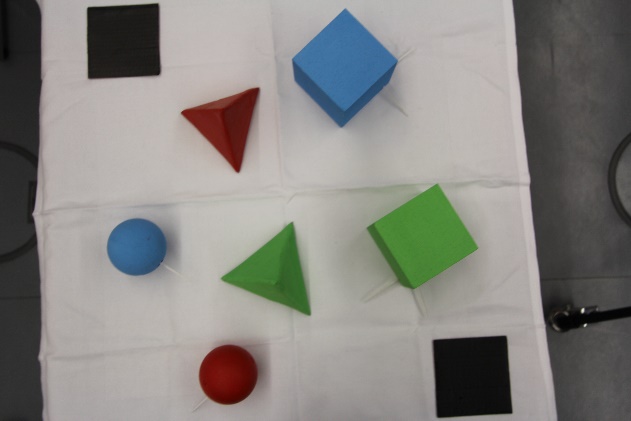
 

Abbildung 1: 3D Objekte in Frontansicht Abbildung 2: 3D Objekte in Vogelperspektive

Für die Formerkennung wurde die Klasse ShapeRecognition geschrieben, welche von VideoProcessor erbt. In der process-Methode wird aus dem übergebene Kameraframe in mehreren Schritten die Form, absolute Pixelkooridinaten und absolute Größe jedes Objektes erkannt und die Daten in einem Array von Objekten gespeichert. Als erstes wird dafür der Kameraframe in ein schwarz-weiß Bild umgewandelt, wobei es wichtig ist in diesem Schritt die Schatten der Objekte herauszufiltern, um die Form nicht zu verfälschen. Da in unserer Anwendung für die Kalibrierung die Farbe Schwarz verwendet wird, können die Schatten erst nach der Kalibration herausgefiltert werden. Aus dem so entstandenen schwarz-weiß Image werden dann in einem zweiten Schritt, mithilfe der findContours- und approxPolyDP - Methoden genauere Informationen über die Konturen der einzelnen Objekte, sowie deren Position und Größe, gewonnen. Durch Betrachtung der Seitenanzahl und Winkelbestimmung kann zwischen einem Rechteck, Dreieck und Kreis unterschieden werden. Jedes auf diese Weise ermittelte Objekt wir nun mit den dazugehörigen Daten in einem statischen Array gespeichert.

## Farb-Erkennung

Nachdem die Bestimmung der Formen einzelner Objekte abgeschlossen ist und die so gewonnenen Daten gespeichert wurden, wird die Farberkennung der Objekte durchgeführt. Dies geschieht in der Klasse ColorProcessor, die ebenfalls von VideoProcessor erbt. In der process-Methode werden nun die zuvor ermittelten Daten ausgelesen und für jedes Objekt eine Farbe ermittelt. Hierfür wird jeweils an der absoluten Pixelposition des Objektes die Farbe aus dem Kameraframe ausgelesen und analysiert. Ist die Farbe im Farbbereich Rot, Blau, Grün, oder Calibration-Color (in unserem Fall Schwarz), so wird die jeweilige Farbe dem Objekt zugeordnet. Anderenfalls ist ein undefiniertes Objekt gefunden worden und die Daten für dieses Objekt werden gelöscht. Nach der Kalibrierung werden ebenfalls alle Objekte, die als Calibration-Color erkannt werden gelöscht.

## Kalibrierung und Positionsbestimmung in 3 Dimensionen

Um den Aufbau des Projektes so flexibel und individuell anpassbar zu machen, wie möglich, muss der zu analysierende Raum bei jeder Anwendung neu kalibriert werden. Hierfür platziert der Benutzer je ein schwarzes Rechteck an zwei sich diagonal gegenüberliegenden Ecken des Raumes und startet die Kalibrierung. Während der Kalibrierung sucht das Programm über einen bestimmten Zeitraum gezielt nach zwei schwarzen Rechtecken. Sobald diese gefunden wurden wird daraus der Nullpunkt des Raumes in der oberen linken Ecke festgelegt, und die Größe des Raumes wird berechnet. Ebenso wird die Größe der Rechtecke als Standartgröße festgelegt, und daraus die Brennweite der Kamera berechnet, was später für die Bestimmung der Z-Achse der Figuren notwendig ist. Wichtig ist zu beachten, dass hierbei mit relativen Werten und nicht mit der realen Größe der Objekte gerechnet wird. Die errechnete Brennweite ist also ein fiktiver Wert und stimmt nicht mit der tatsächlichen Brennweite der verwendeten Kamera überein.

Bilder als Erklärung?

Um die Position eines Objektes im dreidimensionalen Raum zu bestimmen werden als erstes die bisher bestimmten absoluten Pixelkoordinaten in relative X- und Y-Raumkoordinaten im Wertebereich von 0 bis 100 umgerechnet. In diesem Schritt werden zudem alle Daten von Objekte außerhalb des kalibrierten Raumes gelöscht. Als nächstes wird die Z-Koordinate des Objektes berechnet. Hierfür wird zuerst über die Brennweite und Standartgröße der Abstand des Objektes zur Kamera berechnet, und anschließend mithilfe der X- und Y-Koordinaten über eine Dreiecksberechnung die Z-Position des Objektes bestimmen.

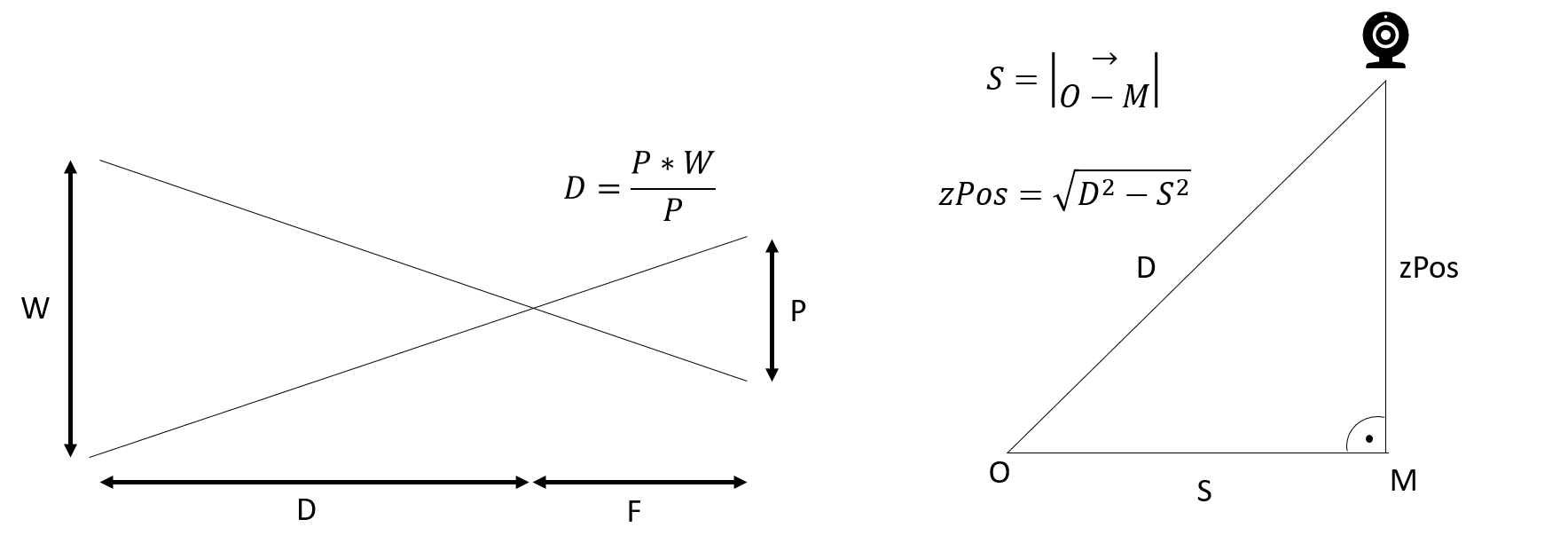


Abbildung 3: Bestimmung der z-Position eines Objektes im 3-dimensionalen Raum

# Front-End Umsetzung

## Interface

Durch ein intuitives Web Interface, soll es dem User möglich sein, verschiedene Einstellungen anhand von Karten zu ändern. So kann dieser zwischen verschiedenen Themen auswählen, die Auswirkung auf dem bereits vorhandenen Soundclip haben. Auch kann die Lautstärke jedes einzelnen Objektes eingestellt werden. Durch die Nutzung der Web Kameras des Gerätes, erhält der Benutzer ein Feedback welche Objekte sich zurzeit im Raum befinden. Diese erscheinen nach dem erfolgreichen Tracking in einer Hierarchie, in der zur Laufzeit die Lautstärke geändert bzw. auf die bereits in der Karte festgelegt Werte zurückgesetzt werden kann.

## WebAudio

* Erstellung eines virtuellen Raumes
* Bestimmung der Raumkoordinaten aller Objekte anhand der empfangenen Daten
* Einstellung der einzelnen Klangquellen mit Soundclip und Lautstärke, vorgegeben durch platzierte Objekte.
* Wiedergabe der Klangquellen an den durch die Objekte festgelegten Positionen

UML?

# Schnittstellenprogrammierung

Um die ermittelten Daten vom C++ Backend an das JavaScript Frontend zu senden wurde die in der Vorlesung besprochenen Variante durch MIDI Datenübertragung verwendet. Um das gesendete Datenvolumen zu minimieren wurde entschlossen lediglich alle 24 Frames die Daten der erkannten Objekte zu übertragen. Dies ermöglichte zudem die Implementation einer Durchschnittsbestimmung. Dafür werden die im Backend ermittelten Daten der erkannten Objekte über einen Zeitraum von 24 Frames gesammelt, verglichen und ausgewertet. Es wird angenommen, dass Objekte sich innerhalb dieser Zeit nur um eine bestimmte Distanz bewegen, sodass Objekte mit annähern gleicher Positionen einander zugewiesen werden. Daraufhin werden die Form- und Farbwerte einander zugewiesener Objekte verglichen und nur der am Häufigsten auftretende Wert übertragen. Auf diese Weise werden kleine Jitter im Form- oder Farberkennungsalgorithmus ausgeglichen.

Bei der Übertragung durch MIDI-Daten war schnell klar, dass nicht mit Standard MIDI-Befehlen, wie zum Beispiel noteOn() oder noteOff() gearbeitet werden kann, das die zu übertragenden Datenmengen zu groß sind. Die Überlegung einen WebSocket zur Übertragung der Daten zu verwenden wurde nach einiger Recherche und Ausprobieren schlussendlich verworfen und es wurde beschlossen die Daten in Form von MIDI System Exclusive’s zu kodieren und zu übertragen.

Darstellung unserer Verschlüsselung?

# Herausforderungen

Während der Umsetzung des Projektes wurde die Gruppe mit einigen Herausforderungen konfrontiert. Zum einen war die Einarbeitung in die Programmiersprache C++ und die Entwicklungsumgebung herausfordernd und führte zu einigen Verzögerungen. Vor allem das Einbinden externer Scripte, wie zum Beispiel „drumstick“, erzeugte verwirrende Fehlermeldungen.

Auf der Seite des Frontendes und der Webdarstellung musste festgestellt werden, dass verschiedene Auflösungen verschiedener Computer das Layout des Userinterfaces ungünstig beeinflusste. Dies zu umgehen und dafür eine gute Lösung zu finden bereitete einige Schwierigkeiten.

Eine weitere Herausforderung war das Finden der richtigen Farbe und Formen der verwendeten Objekte. Es mussten Objekte gewählt werden, die aus der Vogelperspektive eindeutig bestimmt werden konnten auch wenn sie bewegt und aus einem anderen Winkel betrachtet wurden. Zudem mussten Farben gewählt werden, die auch bei sich ändernden Lichtverhältnissen eindeutig zu bestimmen waren. Erst nach einigen Iterationen wurden die letztlich verwendeten Formen und Farben gefunden.

Die wohl größte und leider auch ungelöste Herausforderung war die Verwendung der Audioanlage im Ton Labor. Durch technische Probleme konnte der Audio-Anteil des Projektes nicht, wie geplant, über die große Anlage abgespielt werden. Als Alternative wurde das Audio über 5.1 Kopfhörer wiedergegeben. Hierbei gehen allerdings viele Klanginformationen verloren, vor allem über die Y- und Z-Position des Klanges.

# Reflexion: Umsetzung der User-Stories

Nach Beendung des Projektes ist festzustellen, dass der Großteil der vorher festgelegten User-Stories umgesetzt wurde. Der Benutzer kann seine Soundkulisse selber wählen, indem auf dem Webinterface ein Sound-Thema ausgewählt wird. Zudem können die Klangeigenschaften alle Objekte einzeln durch ein Slide-Down Menu auf der Website angepasst werden. Der User kann alle Objekte separat verwenden und bewegen, wobei eine Veränderung des Raumes in einer Veränderung des Sound Spaces resultiert. Die einzige Abweichung von den User Stories ist, dass die Farbe eines Objektes die Lautstärke vorgibt, anstatt dessen Größe. Der Grund hierfür ist, dass lediglich eine Kamera verwendet wird, um die dreidimensionale Position des Objektes im Raum zu bestimmen.

* Die letzten beiden User Stories wurden nicht umgesetzt

Brauchen wir diesen kompletten Abschnitt? Falls wir den reinmachen, müssen wir ehrlich zugeben, dass wir und nicht an die User-Stories gehalten haben.

# Abbildungen/Diagramme

Diagramme um 90 Grad drehen, um sie größer machen zu können? Zur Leserlichkeit?  
Diagramme zwischen den Texten darstellen?

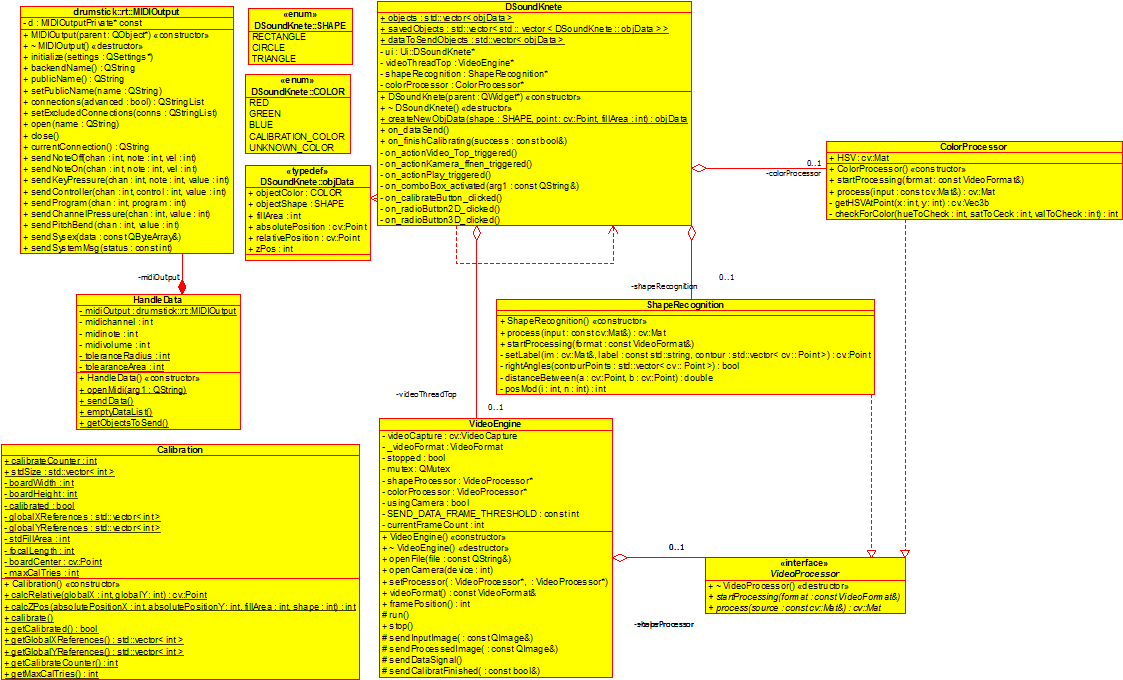


Abbildung 4:UML-Klassendiagramm des Front-End (zur Überersichtlichkeit wurde von der Darstellung weniger relevanter Klassen abgesehen)

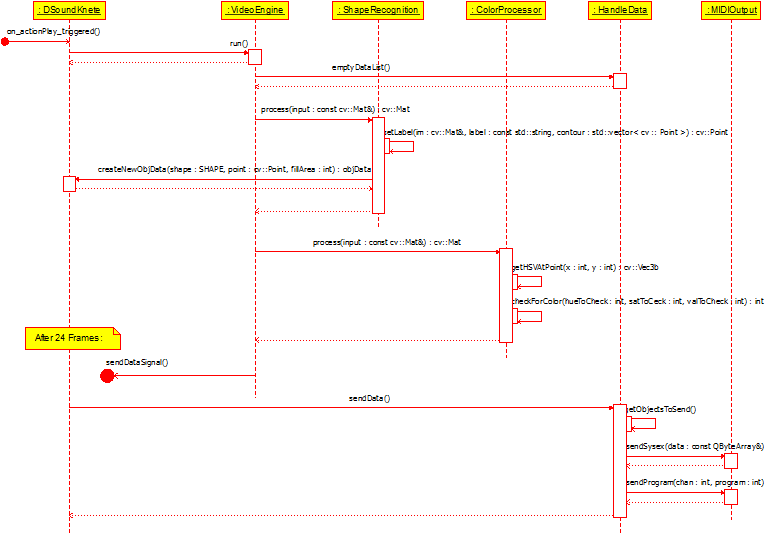


Abbildung 5: UML-Sequenzdiagramm des Front End (zur Übersichtlichkeit wurde von der Darstellung weniger relevanter Klassen und Funktionen abgesehen)