# Projekt: Hexaposer

Projektmitglieder:

MS: Bager Bingöl (2239338), Wladimir Nabok (2238778), Klaus Bochmann (2250591)

# Ziele und Aufgaben der Anwendung

## Einleitung

Im Studienfach Audio- Video- Programmierung soll unser Projekt “Hexaposer” erstellt und bei einer Abschlussausstellung präsentiert werden. Von den Studierenden wird verlangt, erste Erfahrung mit C++ und der WebAudio API zu erlangen und diese in unserem Projekt umzusetzen und vorzustellen. Unser Projekt wird mit der Programmiersprache C++ und JavaScript realisiert. Zur Visualisierung wird eine Webseite, mit HTML und CSS, erstellt.

## Projektziel

Dem Besucher der Ausstellung werden verschieden bedruckte Hexagone und ein Spielplan vorgelegt. Der Benutzer ordnet, ohne weitere Instruktionen, die Hexagone zu einem Bild an. Im Anschluss wird das zusammengesetzte Bild mit einer Kamera aufgenommen, gespeichert und an unsere Webseite übermittelt. Auf der Webseite wird das zusammensetzte Hexagonbild angezeigt. Nach einer Benutzerfreigabe wird ein Musikstück wiedergegeben, welches sich aus der Anordnung der Plättchen auf dem Spielplan generiert. Auf der rechten Seite kann man sich über einen Button eine Legende der im aktuellen Bild verwendeten Hexagone einblenden lassen. Ziel dabei ist es den Nutzer zum ausprobieren und spielen zu animieren.

## Anforderungsanalyse

Es muss eine Software entwickelt werden, welche in der Lage ist auf eine Kamera zuzugreifen, eine Momentaufnahme zu generieren und anschließend die von uns erstellten Spielsteine zu erkennen. Idealerweise kann sie dabei Farbe, Position und Formen auf einem definierten Spielfeld interpretieren. Die Software muss über eine Netzwerkschnittstelle, die verarbeiteten Informationen an eine Webseite übermitteln. Diese wiederum muss aus der gesendeten Interpretation mittels verschiedener Bausteine einen abspielbaren Audiostream erstellen. Außerdem muss die Webseite anhand der übermittelten Daten in der Lage sein eine flexible Legende zu den verwendeten Spielsteinen zu generieren.

# Technische Umsetzung

Aus den Vorgaben des Kurses AVPRG ergibt sich die Anforderung in unserem Projekt sowohl die WebAudio API zu verwenden als auch zumindest einen Teil der Funktionalität in C++ umzusetzen. Dabei werden wir auf (verwendete Frameworks hier ggf mit Abwägungen und Begründung. openCV ) Hardwareseitig werden wir (verwendete Hardware hier: Win10, 8.1, Desktop-PC, Webcam, selbst erstellte Spielsteine, Spielplan aus Papier)

## Frontend

Die Webseite besteht aus drei Hauptkomponenten:

1. Hexagone

   Es werden, eine noch nicht festgelegte Anzahl, an verschiedenfarbigen Hexagone auf der Webseite abgebildet.
   Jedes Hexagon steht für eine Funktionalität. So könnte z.B. das rote Hexagon für die GainNode stehen. Je weiter rechts das Hexagon steht, desto lauter wird die abgespielte Musik.
   Nodes die wir benutzen können: OscillatorNode, GainNode, ConvolverNode, DelayNode, BiquadFilterNode, PannerNode, StereoPannerNode, WaveshaperNode,..
2. Information für die Funktionalität jedes Hexagons

   Bei einem Klick auf ein Hexagon soll das Hexagon einmal umgedreht werden und es sollen Information über das bestimmte Hexagon angezeigt werden.
3. einen Button der eine Musik oder Töne abspielt

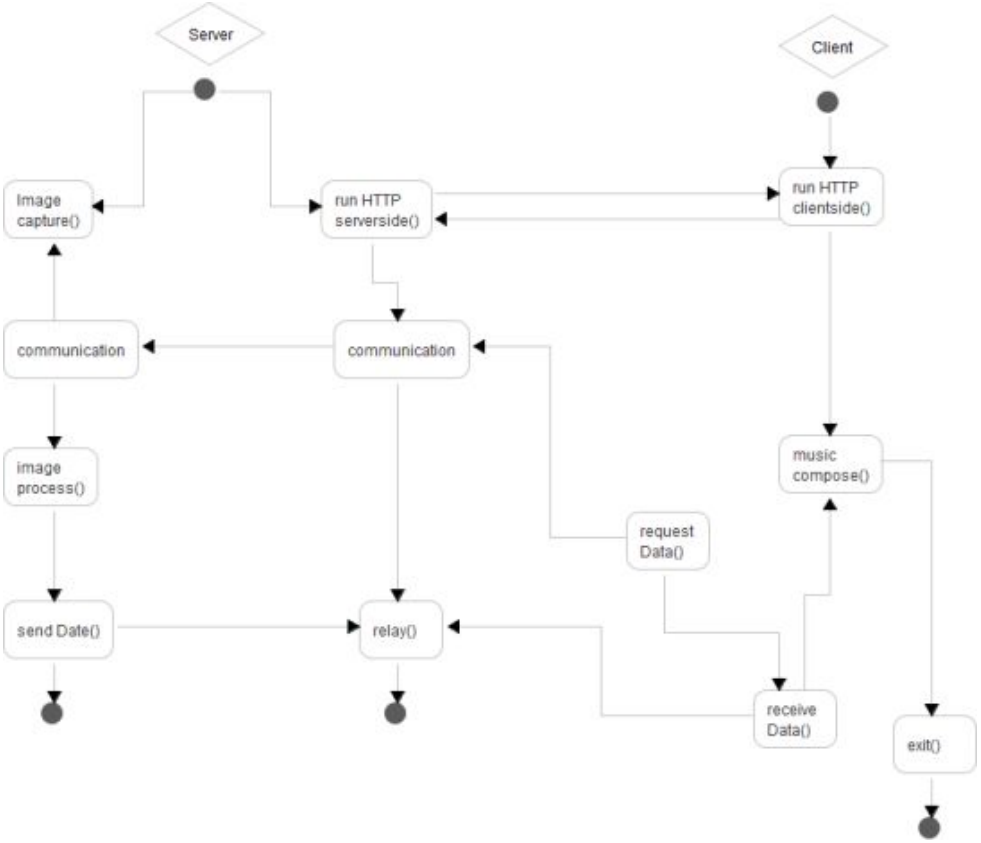
   Es wird ein Start/Pause/Stop- Button auf der Seite abgebildet.
   Dieser ist für den Start/Pause/Stop eines Tons oder einer Musik verantwortlich. Die Musik kann beliebig oft abgespielt werden.

Die Webseite wird mit HTML/CSS visualisiert und mit JavaScript funktionalisiert.

# Backend

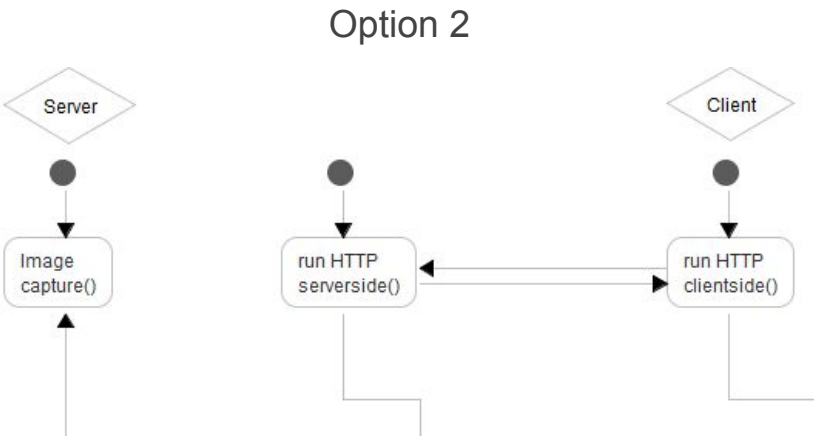
Nach dem jetzigen Stand haben wir zwei Lösungen für die angegebenen und unsrigen Anforderungen Webapplikation ausgearbeitet.
Folgernd aus einer unterschiedlichen Anzahl der Programmstarts werden unterschiedliche Backendlösungen verwendet. Diese bestehen in einem Fall aus einer WebAPI Lösung implementiert durch ASP.net mit einer angepassten OpenCV implementation für das .net framework (siehe Option 1). Im Kontrast dazu steht eine Lösung mit QT + Netzwerkkomponente, OpenCV C++ und einer Webapplikationsbackendtechnologie (siehe Option 2).

Option 1




Das Prozessdiagramm von Option 1, weist zwei Prozesse und dabei eine Unterteilung in Server und Client auf. Auf der Serverseite beginnt das Programm mit einem Programmstart, welches sich in Bildverarbeitungsthread und Hypertext Transfer Protocol (HTTP) auffächert. Der Server-subprozess welcher das HTTP ausführt, antwortet dabei auf korrekte Clientanfragen mit dem Senden der Frontendseitigen Musikkomposer App, bestehend aus HTML, CSS und Javascript (Music compose()).
Wenn Clientseitig asynchron Hexagondaten angefragt werden, wird dieses über den HTTP-Subprozess an einen Bildverarbeitungsunterprozess weitergeleitet (Request Data() und Communication).

Dieser Prozess fängt ein Bild über eine Kamera ein (Imagecapture()), verarbeitet es (Imageprocess()) und sendet die Daten über eine Weiterleitung an den Clientseitigen Prozess, wo ihn der Musikkomposer entgegennimmt.(send Data(), relay(), receive Data()). Also gibt es in dieser Option 1 eine Unterteilung in Unterprozesse aber nur zwei Prozessstarts, einen auf Seiten des Backends und dem anderen auf Seiten eines Clients.




Im Vergleich zu Option 1 gibt es in der Option 2 ingesamt drei Anwendungsstarts. Zwei auf Seiten des Servers und einen Clientseitigen. Es gibt also eine Entkopplung der beiden Prozesse. Bildaufnahme findet dabei in einer Schleife statt und falls der HTTP-Prozess eine Anfrage an den Bildverabeitungsprozess, mit entsprechender Antwort, stellt, findet dies in unterschiedlichen Programmen statt. Weitere Unterschiede sind beim Schreiben des Programmcodes vorstellbar.

Die Entscheidungsprozess findet dabei in der ersten Woche nach Vorstellung des Konzeptes statt und hängt von der Komplexität der Konfiguration des Webservers der Option 1 ab, weil eine Konfiguration auch bei Option 2 notwendig ist, aber diese auch zu einem späteren Augenblick stattfinden kann. Beim Programmieren des Backends wird also in der zweiten Woche nach dem Vorstellen des Konzepts definitiv mit OpenCV und der Bildverarbeitung gearbeitet.

# Bedienkonzept

Nach Start des Programms wird zuerst die Kamera angesprochen und die Software verlangt, unter Einblendung eines HowTo, nach der Erfassung des Spielplans. Sobald erfolgt, wechselt die Software in einen Livekamerastream und blendet einen Button mit dem Label “Kalibrierung” zur Aufnahme ein. Mit einem weiteren HowTo wird der Nutzer aufgefordert den Spielplan auszurichten, alle Spielsteine in eine Grundkonfiguration zu legen und den Button zu bestätigen. Die Software wertet die Aufnahme aus. Wenn Spielplan als auch die Ausgangskonfiguration erkannt wurde, ist die Kalibrierung abgeschlossen. Andernfalls werden Hinweise zu Kameraausrichtung, Beleuchtung, etc. eingeblendet und der Nutzer aufgefordert die Kalibrierung erneut mit einer Aufnahme abzuschließen.

Sobald die Kalibrierung erfolgt ist, ändert der Button sein Label zu “Aufnahme” (Oder “Go”...).
Erfolgt nun eine Aufnahme wechselt die Anzeige zum Bildmodus, die Software wertet die Informationen aus und der Button wechselt zu “Nochmal”. Neben dem Bild wird nun numerisch angezeigt wie viele Spielsteine erkannt wurden und wenn mindestens ein Spielstein erkannt wurde, wird zusätzlich ein “Senden” Button eingeblendet. “Nochmal” wechselt zurück in den Livestream Aufnahmemodus. “Senden” ruft im Standardbrowser die Webseite auf und übermittelt die erfassten Informationen (Spielsteine, Anordnung, Rotation)

Auf der Landingpage werden die Spielsteine in ihrer erfassten Konfiguration eingeblendet und ein Button “Play” erstellt aus der aktuellen Konfiguration über WebAudio eine Audiosource, welche über das Standardausgabegerät abgespielt wird. Nach dem Abspielen wird neben dem “Play” Button ein weiterer “Legende” Button eingeblendet. Bei Betätigung wird von rechts eine Karte eingeblendet, auf der beim Klick auf die Hexagone nähere Hinweise zu deren Funktion bereitgestellt wird. Die Audiosource kann beliebig oft neu abgespielt werden.

Überlegungen:

- Die Software könnte auch die Möglichkeit zum Speichern und Laden von Bildern beinhalten.
- Prüft die Software beim Senden ob die Webseite bereits offen ist, oder wird bei jedem Mal ein neues Browserfenster geöffnet?
- Kann die Erkennung der Spielsteine bereits im Livestream erfolgen?

# Zeit- und Ressourcenplan

## Zeitplan

| | |
|---|---|
| 14.10.18 | Ideenfindung, Diskurs und Recherche |
| 21.10.18 | Wahl zum Projektinhalt |
| 28.10.18 | **Fertigstellung Konzept** |
| | Präsentation 30.10.18 |
| 04.11.18 | |
| 11.11.19 | |
| 18.11.18 | |
| 25.11.18 | **Fertigstellung Prototyp** |
| | Präsentation am 27.11.18 |
| 02.12.18 | |
| 09.12.18 | |
| 16.12.18 | **Fertigstellung Funktionalität** |
| | Präsentation am 18.12.18 |
| 11.01.19 | **Abschluss Feintuning** |
| | Abgabe bis 13.01.19 |

## Teamplanung

| | |
|---|---|
| Wladimir Nabok | Dokumentation, Imageprocessing in C++, Backendprogrammierung |
| Bager Bingöl | Dokumentation, Gestaltung, Materialien, HTML/CSS, JavaScript |
| Klaus Bochmann | Dokumentation, Gestaltung, Materialien, HTML/CSS, JavaScript |