Luft-Piano

Tobias Braun 2365827,
Kira Hansen 2364148,
Jacob Heindorf 2364947

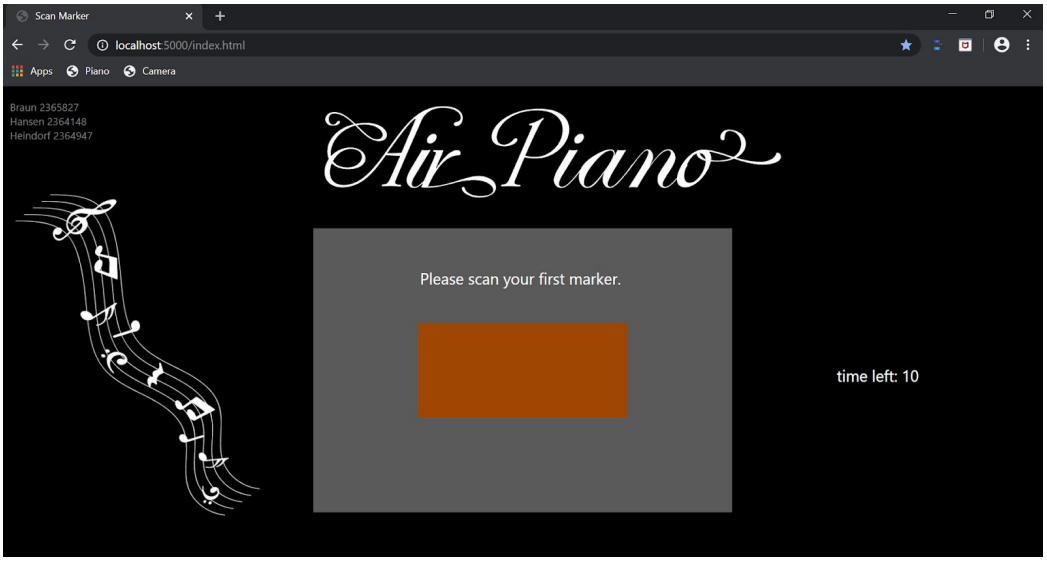
05.01.2020

Inhalt

1. Einleitung

Das Ziel des Projektes ist ein fertiges Programm, mit dem der Benutzer „Luft-Klavier“-spielen kann. Der Anwender benutzt zwei visuelle Marker, um auf der virtuellen Klaviatur die Tasten anzuschlagen. Um die Anwendung zu starten, muss vorher die Project.py Datei per Terminal gestartet werden. Danach kann dann die index.html im Browser über localhost:5000/ aufgerufen werden.

2. Die Weboberfläche mit HTML und CSS



Im Laufe des Projekts wurde das vorher angedachte Design überarbeitet.
Auf der Startseite ist der Hintergrund schlicht in schwarz gehalten und die Texte jeweils in weiß. Sowohl auf der Start- als auch auf der Hauptseite finden sich unsere Namen mit

Matrikelnummern links oben. Diese sind in grauer Schrift, damit sie nicht herausstechen. Darunter ist ein Bild von einem Notenschlüssel (Bildquelle: https://de.pngtree.com/element/down?id=NDI2MzM5Nw==&type=1), um die Seite dem Thema unseres Programms besser anzupassen. Das oben mittige Logo “Air Piano” hat Tobias Braun erstellt.
Das Farbfeld in der Mitte der Webseite wird durch einen grauen Hintergrund vom restlichen schwarzen Hintergrund hervorgehoben. Rechts wird der Countdown angezeigt, welcher jeweils für beide Marker abläuft. Mit Hilfe der counter.js wird der Python übermittelte Countdown und Farbwert entgegengenommen und der Countdown-Text und die Farbe der Farb-Box geändert. Nach Ablauf des zweiten Countdowns wird die piano.html aufgerufen.

Auf der Hauptseite ist das Logo “Air Piano” mittig und größer als auf der Startseite dargestellt. Der Hintergrund ist weiterhin schwarz und links oben sind nach wie vor unsere Namen und Matrikelnummern aufgelistet. Die eingescannten Marker und dessen Farben werden als runde Kreise (span-Element in HTML) dargestellt. Die Klaviatur ist durch einen grauen Hintergrund vom restlichen schwarzen Hintergrund hervorgehoben. Alle Tasten sind div-Elemente und in der Klasse “keys”. Um jedoch jede einzelne Taste positionieren und skalieren zu können, sind alle jeweils mit ihren eigenen ID’s gekennzeichnet. Diese werden in CSS aufgerufen. Um die Tasten realistischer wirken zu lassen, haben alle eine image-Komponente zugewiesen. Wenn eine Taste von einem Marker “gedrückt” wird, so wird dessen Farbe durch die synth.js Datei gräulich gefärbt.

3. Der Server

Wir verwenden das von Prof. Plaß vorgestellten flask_socketio. Allerdings unterstützt Flask nur die Kommunikation von Python zu JavaScript. Daher verwenden wir einen Timer um die Marker-Kalibrierung auf der Startseite zu initiieren.

4. Die Bildanalyse mit Python

Da die Kommunikation über den Server nur von Python zu Javascript funktioniert, haben wir im ersten Teil des Programms, in dem die beiden Marker eingescannt werden, in Python

einen Countdown gestartet, nachdem die Marker erfasst werden. Während der Countdown läuft, werden die Farbdaten für das Farbfeld an die Web-Anwendung gesendet.
Der Hauptzweck des Python Programm Teils ist es das Kamerabild auszulesen und die Marker Positionen zu finden. Hierzu werden die Farbwerte des Bildes mit den Farben des zuvor eingescannten Markers verglichen und eine Bitmaske erstellt. Aus dieser kann dann die Position des Markers anhand der größten Konzentration übereinstimmender Pixel gefunden werden.
Hierbei werden einige Fehlerkorrekturen genutzt, um zufällige Pixel im Hintergrund auszusortieren und um zu verhindern, dass der Marker dabei zu große Sprünge macht.
Diese Positionsdaten werden mit den Positionen der Tasten abgeglichen und es wird getestet, ob der Marker auf einer dieser Tasten liegt.
Letztendlich werden die Positionsdaten und die angeschlagene Taste (0 falls es keine gibt) an die Javascript Applikation weitergeleitet.

5. Die Soundausgabe mit JavaScript

Am Anfang werden in der Anzahl der Tasten und der zwei Marker die Gains in einem Array erzeugt und in die Audio-“Pipeline” integriert, diese fortgeführt und letztendlich mit dem Kontext verbunden. Es werden zu jeder Taste zwei Event-Listener hinzugefügt. Mouseover startet einen Sound, Mouseout beendet einen Sound. Dies dient der Bedienung mit der Maus.
Die Datei synth.js horcht mit unterschiedlichen Listenern auf den Server und nimmt die Tasten-Schlüssel mit Marker-Position zu den beiden Markern entgegen. Wird vom Server eine 0 übertragen, so wird kein Sound ausgegeben. Ein valider Tastenschlüssel führt zur Tonausgabe. Wird der gleiche Schlüssel entgegengenommen, so wird keine neue Tonausgabe gestartet. Wechselt der Schlüssel, so wird eine neue Ausgabe gestartet. Am Ende wird die Marker-Position neu gesetzt. Mit eigenen Listenern werden die Marker-Farben entgegengenommen und zum Einfärben der Marker verwendet.
Beim Starten eines Sounds wird ein zum Marker und Schlüssel gehörender Sägezahn-Oszillator mit entsprechender Frequenz erzeugt. Dazu wird die Frequenz aus einem Array mit allen verwendeten Frequenzen gelesen, mit der Audio-”Pipeline” verbunden und der Oszillator gestartet.
Um einen Klavier-Charakter zu erzeugen werden zwei Töne ausgegeben. Ein Ton wird länger gehalten. Beim Beenden einer Ausgabe werden die beiden Töne ebenfalls mit unterschiedlichen Releases ausgeblendet.
Zudem wird zum Angleichen an ein Klavier noch ein Convolver und eine Distortions-Kurve genutzt, die in die Pipeline integriert sind.

5. Fazit

Da Flask_SocketIO nur die Kommunikation von Python zu JavaScript unterstützt, konnten wir leider keinen Startbutton zum Initiieren des Marker-Scannens benutzen, sondern haben es nun über einen Countdown gelöst. Wir konnten daher dem Benutzer auch keine Möglichkeit bieten, selbst Farb- und Sättigungs-Variablen einzustellen.

Der Kamera-Input konnte nur einmal verwertet werden. Dadurch konnten wir beim Einscannen der Marker auf der Webseite kein Kamerabild anzeigen, da das Kamerabild von Python zur Analyse gebraucht wird. Um dem Nutzer Feedback über die derzeitig eingescannte Farbe zu geben, haben wir ein Farbfeld eingesetzt.
Die Farberkennung in Python ist relativ unzuverlässig und lässt sich bei schwankenden Lichtverhältnissen und ungünstig gewählten Hintergrund leicht stören. Das führt dazu, dass der Marker oftmals stark umherspringt.
Den ursprünglichen Plan noch ein Notenblatt einzufügen und die gespielten Noten einzuzeichnen, haben wir aus Zeitmangel verworfen.