20.6 - Vorlesung

Die Rahmenbedingungen und Fragestellung des Projektes werden vorgestellt. Wir haben die Idee eines Musikspiels ähnlich GuitarHero erarbeitet. Der Spieler bekommt einen Song ohne Drums vorgespielt, Farben zeigen ihm an, was als nächstes gespielt werden soll. Basierend auf seiner Reaktionszeit wird ein Score errechnet.

Offene Fragestellung ist vor allem das miteinbeziehen von Q-Learning. Meine Idee war den Score auf den Nutzer anzupassen und zu lernen welche Fehler häufiger auftreten und dies in die Bewertung der Performance einzubauen. Insgesamt tendieren wir jedoch eher zur Auswahl des nächsten Songs auf Basis des User-Scores, da dies einen größeren Einfluss auf das Gameplay hat.

23.6

Meine Arbeit basierend auf der Projektidee bezieht sich vor allem auf die Recherche von Möglichkeiten zum Drummen bzw. zum Input der Userdaten.

Für den Prototyp sind jedoch die vorhandenen Sensoren ausreichend. Einzelne Anschläge können erkannt werden und der bereits vorhandene MPR121 Kapazitive Sensor bietet genügend Anschlussmöglichkeiten um unseren vier verschiedenen Drum Sounds abzuspielen. Nachteil des Sensors ist die fehlende Möglichkeit festzustellen, wie fest der Sensor gedrückt wurde. Dies Schränkt die Spielmechanik zwar nicht ein, aber verringert die Authentizität des Erlebnisses, da verschiedene Lautstärke bei einem normalen Schlagzeug zu erwarten ist.

27.6 - Vorlesung

Wir starten die eigentliche Entwicklungsphase unseres Spieles. Die Idee ist grundsätzlich festgelegt, sobald wir einen Prototyp haben können wir selbst spielen und verschiedene Ansätze austesten. Oberste Priorität ist somit auch der Prototyp selbst. Mein Anteil hierzu ist die Weiterentwicklung unseres Python Scripts, das einen Großteil der Spiellogik beinhaltet. Dies umfasst einerseits das Aufnehmen von Referenzsongs und das Spiel selbst. Auf Basis des Referenzsongs wird dem Spieler angezeigt wann er was spielen muss.

Eine offene Frage ist inwieweit Programmlogik auf dem Arduino sein sollte. Meiner Meinung nach so wenig wie möglich, einerseits aufgrund der begrenzten Speicherkapazität, andererseits aufgrund der fehlenden Möglichkeit dort asynchron zu programmieren. Für ein physisches Spiel ist es enorm wichtig eine geringe Latenz zu haben, da Nutzer dies bei physikalischer Interaktion nicht erwarten. Außerdem sollen viele BPM möglich sein, um auch schwerere Lieder anbieten zu können.

27.6

Ich habe unser Plüschtier und den Haupt-Arduino mitgenommen um die in der Vorlesung erarbeiteten Komponenten zu verknüpfen. Hierbei musste das Arduino Skript auf I/O

Funktionen reduziert werden und die Kommunikation zwischen Laptop und Zwerg angepasst werden. Das in der Vorlesung erarbeitete Abspielen von Songs und die Recherche nach passenden Liedern resultierte in der Iteration die im Video zu sehen ist. Die Lichter reagieren hierbei noch zufällig auf Nutzereingaben. Der Test zeigt das das spielen an sich Spaß macht, jedoch müsste an den Eingabesensoren und den abgespielten Drum-Sounds gearbeitet werden, diese klingen noch nicht wirklich authentisch. Die an diesem Nachmittag Implementierte Interaktion zwischen Licht, Touch und Sound funktioniert jedoch schon gut.

Video des Resultats: https://drive.google.com/open?id=0B5YGK82mBGafdFVNTjh4TERra3c

28.6

Ein Format für Referenzlieder wurde erstellt. Eingaben werden als Json abgespeichert, die Eingabezeiten pro Touch Komponente werden diskretisiert abgespeichert.

29.6

Mit Markus vormittags am Recording für Referenzsounds gearbeitet. Gute Lieder ohne Drums sind schwer zu finden, das abspeichern als diskretisierte Zeitschritte funktioniert gut, Hauptproblem bei der Implementierung war das threading und shared variables ordentlich hinzukriegen. Am Ende hat die Aufnahme von Referenzsounds soweit funktioniert, siehe auch das Video (mit Tierstimmen als Lied).

Video: https://drive.google.com/open?id=0B5YGK82mBGafUTRaN0lmdmdFVUE

Nachmittags an der Implementierung des Anzeigens des Referenzlieds gearbeitet. Frage ist hier insbesondere wie viel im voraus eine Eingabe angezeigt werden muss. Der Neopixel ist durch die 12 LEDs beschränkt, jeder der 4 Sounds erhält somit nur 3 LEDs zum anzeigen. Man könnte damit drei Zeitschritte im voraus pro Farbe anzeigen, was jedoch mMn zu wenig ist. Es wäre sinnvoll vom Neopixel auf 4 unabhängige Strips zu wechseln, wenn die Zeit bzw. der Takt eine größere Rolle spielen soll. Dann ist man auch näher an den Referenzspielen wie Guitar Hero und ermöglicht eine höhere Schwierigkeit von Liedern. Die Kommunikation zwischen Arduino und Python funktioniert gut, der Gameloop ebenfalls, das einfache Anzeigen von baldigen Eingaben auch. Limitierung ist aktuell noch die Beschränkung des Abspielens von Tönen auf OS X, für Windows müssten noch zusätzliche Libraries hinzugefügt werden. Das ist jedoch für mich schwer zu testen/ implementieren.

Video: https://drive.google.com/open?id=0B5YGK82mBGafbnMzVUs2cXJ3eG8

4.7 - Vorlesung

In der Vorlesung habe ich mit Thomas größtenteils an der Verknüpfung des Spiel-Python Servers und dem QLearning gearbeitet. Wir haben uns eine Berechnung des Scores überlegt und Schnittstellen zwischen unseren Modulen definiert. Prinzipiell werden während des Spiels die Nutzereingaben und das Referenzlied verglichen. Dabei werden Berührungen des Sensors (korrekt und inkorrekt) gespeichert. Mithilfe des Referenzliedes wird ein

Koeffizient aufgestellt anhand dessen der Score berechnet wird. Video1 und Video2 sind Mitschnitte der Zugfahrt nach Hause und nicht für die Entwicklung relevant.

Video1: https://drive.google.com/open?id=0B5YGK82mBGafSXFZaC1IcUFuWW8
Video2: https://drive.google.com/open?id=0B5YGK82mBGafNXlpb1FfeHZiemc

5.7

Refactoring des Python Servers, Code Review für QLearning. Ich habe mir zudem ein paar Alternativen für den Neopixel angeschaut, um mehr Anzeigefläche zu haben und die Zeitkomponente besser darstellen zu können. Zusätzlich habe ich nach Songs ohne Drums und Drum Samples gesucht.

6.7 - 13.7

Wir haben Videos gemacht, die Leitfäden vom Markus getestet/implementiert was super funktioniert hat - der Prototyp funktioniert. Ich habe neue Songs runtergeladen (nicht Lizenzfrei) und die Drum Sounds angepasst. Eigentlich würde ich gerne den Code rewriten und seperate Prozesse für Anzeige und Inputverarbeitung nutzen und Timestamps statt den diskretisierten Schritten verwenden, aber dafür ist keine Zeit (Prüfungen). Lukas und Stefan haben netterweise ein paar Referenzlieder eingespielt und den Prototyp soweit getestet. Meinen Teil der Präsentation habe ich auch noch vorbereitet.

14.6

Präsentation. Hat alles funktioniert.
Wenn es Probleme mit dem Python Code oder sonstwas gibt, bitte ne Mail schreiben. ;)