13. Juni 2016

Heute haben wir uns Gedanken zu unserem HCI-Projekt gemacht und dazu eine Ideensammlung verfasst. Nachdem wir weitere Ideen im Laufe der Diskussion wieder verworfen haben, haben wir uns schließlich dafür entschieden, dass wir über einen Python-Server Sounds abspielen wollen und je nach Input zum Beispiel ein Schlagzeug nachstellen. Dabei sollen die verschiedenen Interaktionen mit dem Zwerg dazu führen, dass unterschiedliche Töne abgespielt werden und so ein Rhythmus bzw. ein Lied gespielt werden kann. Als Interaktionen mit dem Zwerg haben wir auch die Ergebnisse aus dem Bodystorming verwendet, so dass wir zu den Interaktionen „Streicheln“ und „Schlagen“ gekommen sind. Diese wollen wir über einen Haptic Sensor messen und unterscheiden. Eine weitere Idee ist, dass die Lautstärke der abgespielten Sounds mit Hilfe eines Potentiometers verändert werden kann. Außerdem möchten wir einen Motor einsetzen, mit dem die Zipfelmütze des Zwergs aufgerichtet werden kann, um so die Freude des Zwerges ausdrücken zu können. Zu guter Letzt möchten wir noch einen Pulse simulieren, indem wir einen Vibrationsmotor und eine rote LED Lampe einsetzen, um damit dem Zwerg mehr Leben zu verleihen.

20. Juni 2016

Endlich geht es so richtig los mit dem Projekt! Nachdem wir heute weitere Details zu unserem Projekt bekommen haben, haben wir unsere Ideensammlung noch einmal überarbeitet und verfeinert. Das Ergebnis schaut jetzt so aus, dass wir den Zwerg in ein interaktives Spiel verwandeln möchten, bei dem der Spieler über den Zwerg ein Töne eines Schlagzeugs produzieren kann und so den Rhythmus zu einem Lied spielen kann. Dabei wurde die Affordanz, die Hände und Füße des Zwerges anzufassen, dazu genutzt Töne abzuspielen. Um dem Spieler anschließend ein Feedback zu geben, wie gut er abgeschnitten hat, wird die Zipfelmütze des Zwerges verwendet, die sich entsprechend des Ergebnisses höher oder niedriger aufrichtet. Damit der Spieler weiß, welche Töne er spielen soll, also welche Hand bzw. Fuß er anfassen soll, wird ein Neopixelring verwendet, der an der Nase des Zwerges angebracht wird. Dabei wird das natürliche Mapping sowohl zwischen den Farben als auch der physikalischen Anordnung der leuchtenden LEDs und der Hände bzw. Füße beachtet.

27. Juni 2016

In der heutigen Stunde ging es dann technisch zur Sache. Zuerst habe ich bei mir Python installiert, das in unserem Projekt für das Backend verwendet wird. Nachdem dies geschafft war, musste ich mich erst einmal wieder in die Syntax von Python einarbeiten, da das letzte Mal als ich was mit Python entwickelt habe, schon sehr lange zurück gelegen ist. Bei der Einteilung der weiteren Aufgaben habe ich die Aufgabe für das QLearning bekommen. Die erste Frage, die es zu klären galt, war es herauszufinden, wo überall maschinelles Lernen in unserem Kontext angewendet werden kann. Dabei gab es zahlreiche Möglichkeiten, wie zum Beispiel bei der Erstellung der Referenzfiles, die dazu benötigt werden, dem Spieler anzuzeigen, welche Töne er bei einem Lied wann spielen soll. Eine weitere Idee war es, das QLearning im Rahmen der Songauswahl zu verwenden, wofür wir uns dann auch nach einer kurzen und informativen Diskussion entschieden haben. Die restliche Zeit habe ich dann damit verbracht, ein Konzept für die Verwendung und Integration von QLearning zur Songauswahl in unser Backend aufzustellen und mir die dazu benötigten Kenntnisse in Python anzueignen.

29. Juni 2016

Heute habe ich das Modul des QLearning Algorithmus in Python fertig entwickelt und getestet. Damit ist es nun möglich, dass verschiedene Lieder registriert werden und der Algorithmus ein passendes nächstes Lied auswählt. Ich möchte an dieser Stelle nicht näher in die Implementierungsdetails des Algorithmus eingehen, aber dennoch einen groben Einblick dazu geben. Zuerst einmal ging es bei der Songauswahl darum, dass der Spieler weder überfordert noch unterfordert werden sollte, da dies die Motivation des Spielers negativ beeinflusst und damit auch weniger Spaß macht, was für ein solches interaktives Spiel sehr ungünstig wäre. Außerdem sollte die Auswahl der Lieder abwechslungsreich sein und es sollten auch die Biosignale des Spielers mit einbezogen werden, um so ein noch besseres Bild der Emotionen des Spielers zu bekommen und diese dann auch zu berücksichtigen.
In unserem Beispiel besteht die QMatrix nun aus einem Zustand und einer Aktion für jedes registrierte Lied, nämlich dieses auszuwählen. Die Werte dieser Matrix sollten nun die Schwierigkeit des Liedes darstellen, damit so eine passende Auswahl getroffen werden kann. Dazu gehen wir von einem bestimmten Referenzwert aus, den wir für die Score des Spielers für ein Lied als optimal annehmen. Jede Abweichung des Scores von diesem Referenzwert wird nun negativ für die Belohnung angerechnet. Außerdem wird der durchschnittliche Puls des Spielers beim Spielen in der Belohnung berücksichtigt, so dass diese zum Beispiel geringer ausfällt, wenn der Spieler einen zu niedrigen Puls hatte, also nicht genug gefordert wurde. Die Auswahl des nächsten Liedes geschieht nun so, dass in 10% der Fälle ein Lied komplett zufällig gewählt wird und in den anderen 90% ein Lied zufällig aus den X am besten geeigneten Lieder gewählt wird, wobei X für eine in jedem Kontext neu zu definierende Zahl steht.

3. Juli 2016

Da der QLearning Algorithmus nun gut funktioniert, müssen die Lieder wiederholt gespielt werden und das dauert seine Zeit. Um jetzt aber schon zu Beginn eine recht gute Abschätzung für die Schwierigkeit eines Liedes zu bekommen, wollten wir dafür ein Modul entwickeln, was ich heute getan habe. Es ging dabei darum ein Referenzfile auszuwerten, in dem näher definiert ist, wann genau welche Töne gespielt werden sollen. Da es nicht ganz trivial ist, wie das gemacht wird, habe ich mir zuerst ein Konzept dafür überlegt. Letztendlich habe ich ein zweistufiges Verfahren gewählt. Zum einen ist für die Schwierigkeit entscheidend, wie viele Töne gleichzeitig gespielt werden sollen und zum anderen wie schnell ein Wechsel zum nächsten Ton erfolgen soll. Im ersten Schritt wird nun jeder Übergang analysiert und geschaut, wie viele Töne gleichzeitig gedrückt werden sollen, wobei gleichzeitig in einem bestimmten Intervall heißt. Dieses Ergebnis wird dann in einem Schwierigkeitswert für diesen Übergang übertragen. Im zweiten Schritt wird nun die Zeit zwischen jedem Übergang betrachtet und ebenfalls in einen Schwierigkeitsgrad für jeden Übergang gemessen. Für jeden Übergang werden nun beide Schwierigkeitsmaße zusammen verrechnet und anschließend wird der Durchschnitt gebildet. Dieser Wert ist nun ein recht gutes Maß für die Schwierigkeit des Liedes, was sich in einigen Beispielen bestätigt hat. Nachdem ich dieses Konzept entwickelt hatte, habe ich ein Modul dafür implementiert und ausführlich getestet.

4. Juli 2016

Heute haben wir in der Stunde unsere bisherigen Ergebnisse zusammengetragen und an dem Zwerg getestet. Es ist zwar noch nicht alles perfekt, aber wir sind auf einem sehr guten Weg.

Nach einer kurzen Besprechung haben wir die Aufgaben für den heutigen Tag eingeteilt. Was jetzt noch fehlt ist die Berechnung des Scores, den der Spieler für das Spielen eines Liedes bekommt und der sich an den richtig gespielten Tönen bemisst. Diese Funktionalität habe ich heute zusammen mit Martin entwickelt und in unser schon bestehendes Backend integriert. Dabei sind wir auf das Problem gestoßen, dass nicht nur die richtig getroffen Töne für den Score verantwortlich sein sollen, sondern auch die falsch gespielten und nicht gewünschten Töne, da sonst ein Spieler einfach immer alle Töne spielen könnte und dann auf diese Weise eine perfekte Punktzahl bekommen würde. Diese Erkenntnis haben wir dann daraufhin in die Score Berechnung mit einfließen lassen. Außerdem haben wir uns überlegt, wie die Pulswerte des Spielers während des Spiels berücksichtigt werden sollen und uns dafür entschieden, dass je nach durchschnittlichem Puls Bonuspunkte bzw. Maluspunkte auf die Score addiert werden.

5. Juli 2016

Nachdem wir gestern weiter am Backend gearbeitet haben und die Scoreberechnung fast vollständig entwickelt haben, habe ich heute meine bereits implementierten Module zum QLearning und zur Berechnung der Schwierigkeit eines Songs in das bestehende Backend integriert. Außerdem habe ich die Berechnung der Score fertiggestellt, in dem ich die Pulswerte des Spielers in die Berechnung der Score auf die gestern besprochene Art und Weise mit einbezogen habe. Anschließend habe ich soweit es mir möglich war getestet, ob die Integration erfolgreich verlaufen ist und dadurch nichts kaputt gegangen ist. Zudem habe ich die dabei aufgetretenen kleineren Fehler behoben.

11. Juli 2016

Heute haben wir alle Komponenten zusammengeschaltet und konnten so auf unserem Zwerg einige Lieder spielen. Diese Integration hat super funktioniert, so dass es keine größeren Probleme gab und wir mit der Aufzeichnung von Videos beginnen konnten. Außerdem haben wir den Zwerg mit echten Nutzern in Verbindung gebracht, die viel Spaß beim Spielen der Lieder hatten. Nach einer weiteren Besprechung bezüglich der Präsentation am Donnerstag haben wir die Teile für diese Präsentation aufgeteilt. Zum Ende der Stunde habe ich mich dann bereits mit der Erstellung meines Teils der Präsentation beschafft, indem es um den QLearning-Teil geht. Dazu habe ich entsprechende Folien sowohl zur Idee der Verwendung von maschinellem Lernen als auch zur Umsetzung in unserem Projekt erstellt.

12. Juli 2016

Im Laufe des heutigen Tages habe ich die einzelnen Teile der Präsentation zu einer einzigen gesamten Präsentation zusammengeführt und in ein einheitliches Layout überführt. Außerdem habe ich einzelne Folien leicht überarbeitet und Bilder hinzugefügt, so dass die Präsentation interessanter wird.

14. Juli 2016

Heute war die Abschlusspräsentation. Alles hat super funktioniert und das Projekt hat viel Spaß gemacht.