FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Human Maschine Interface

Hausarbeit

Simon Game

Wintersemester 2017/ 2018

Gruppe 16

7. Februar 2018

Inhalt

1. Einleitung

1.1 Mensch-Computer-Interaktion

Die Erforschung der Schnittstelle zwischen Mensch und Maschine.

Nehmen wir einmal auf unserer Reise durch dieses Thema die Definition der gängigen Plattform Wikipedia[1], so handelt es sich demnach bei Human-Computer-Interaction (Abk. HCI) um die Erforschung des „Designs und der Verwendung von Computer-Technologie“[1]. Ferner beschäftigt sich die Forschung „mit der Art und Weise, wie Menschen mit Computern und Design-Technologien interagieren“[1].

Weiterhin wird erwähnt, worauf wir an dieser Stelle nicht weiter eingehen, dass „Erkenntnisse der der Informatik auch solche aus der Psychologie (vor allem der Medienpsychologie), der Arbeitswissenschaft, der Kognitionswissenschaft, der Ergonomie, der Soziologie und dem Design herangezogen“[1] werden.

Auch wenn heutzutage die Benutzung von Computern wie Notebooks, Maschinen, Smartphones und viele weitere Geräte das Leben der Menschen vereinfachen soll, kann es dazu kommen dass die Kommunikation zwischen Mensch und Maschine den Menschen überfordert oder unverständlich ist.

Das Ziel ist es, dies zu vermeiden und eine passende Kommunikationsmöglichkeit zu erstellen oder gar über einen manchmal längeren Prozess maximal zu optimieren und zu verbessern.

Hierbei spielt natürlich auch der Kostenfaktor eine Rolle. Eine individuelle Schnittstellenoptimierung für ein bestimmtes Individuum oder eine Menschengruppe (z.B. junge/ alte Menschen, körperlich oder auf Sinnesorganebene eingeschränkte Menschen, Farbblindheit etc.) verursacht Entwicklungskosten die manchmal bei Projekten mit bedacht werden sollen und das ganze muss auch noch auf Funktionsfähigkeit getestet werden.

Die zwei Kommunikationsunterschiede zwischen „Mensch/Mensch“ und „Mensch/ Maschine“ werden in den Vorlesungsfolien von Prof. Dr. -Ing. Matthias Deegener im Modul “Human Maschine Interface” aufgezeigt. Hier wird vermittelt, dass der Mensch eine natürliche Intelligenz aufweist Weltwissen und die Fähigkeit zur Abstraktion, Interpolation und Anpassung während der Computer nur aus einer Programmierung besteht, folglich aus stark eingeschränktem Spezialwissen, ohne Möglichkeit einer Abweichung vom Programm.

Und hierfür muss ein Kompromiss beziehungsweise eine Lösung hergestellt werden, der für beide auf seine Art verständlich ist.

1.2 Grundlagen der HMI

Ein wichtiger Fokus der HMI fällt auf die 5 Sinneswahrnehmungen des Menschen. Wir führen hier die Sensorik aus der Vorlesung noch einmal zusammengefasst in unseren Worten auf.

Die sensorischen 5 Sinne des Menschen nehmen die Reize der Umgebung auf und leiten sie als Nervenimpulse weiter.

- Sehen 80% der Datenmenge kann erfasst werden
- Hören 15% der Datenmenge kann erfasst werden
- Tasten, Fühlen
- Schmecken
- Riechen

1.3 Das Simon Spiel als interessantes HMI Modell

Im Vorhinein lässt sich sagen, dass das Simon Spiel eine interessante Wahl war, um mehrere Schnittstellenansprüche zwischen einem Menschen (Benutzer) und Computer (Spiel) zu untersuchen und mit mehreren Themen der Vorlesung genauer vertraut zu werden und diese zu verinnerlichen.

Auch eignet sich das Simon Spiel aus der Vergangenheit, nun in digitaler Form hervorragend zum erweitern, um verschiedene Daten über Reaktionszeit oder Erinnerungsvermögen des Menschen und weiteres z.B. in einer Textdatei zu erfassen und diese weiter mit anderer Software auszuwerten oder graphisch darzustellen. Dies war damals vermutlich nicht so leicht zu erfassen, wie es heute der Fall ist, anhand einer Simon Software.

Unsere mobile Simon App spricht bis zu drei der fünf Sinne des Menschen an, das Sehen, das Tasten und das Hören.

Der Sinn des Spiels besteht darin, sich eine visuell dargestellte Farbkombination zu merken. Es beginnt mit der Berührung von „Start“ und fängt mit einer Farbe an, die aufleuchtet, die zu merken und anschließend über Berührung auf die gleiche Farbe zu wiederholen ist. Ist vom Benutzer eingegebene Farbe richtig, so steigt das Spiel in den nächsten Level hoch und es ist eine zusätzliche Farbe zu der bereits gezeigten Farbreihenfolge zu merken und über Berührung zu rekonstruieren. Ist eine Eingabe der Farbreihenfolge falsch, so ist das Spiel zu Ende und kann über Start neu begonnen werden.

Weiterhin waren wir in unseren Gruppenmeetings darum bemüht eine Lösung zu finden, wie wir die App einfach und benutzerfreundlich gestalten können, damit diese auch für Anwender ohne Vorkenntnisse verständlich zu bedienen ist.

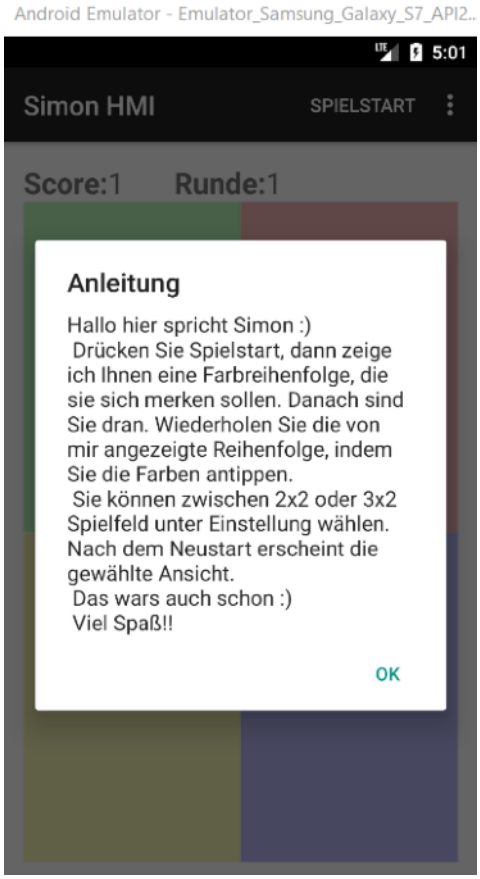


Abbildung 1

Eine sympathische, persönliche Benutzeranleitung wurde zusätzlich als Hilfe mit eingebaut und wurde mit Bedacht auf „schnell durchlesen und loslegen“, somit leichtes Verständnis dem Spiel hinzugefügt (Abbildung 1).

Weitere Details zum Simon Spiel, vor allem dem Sinn und Zweck dahinter, weshalb wir dieses Spiel gewählt haben und am Ende begeistert waren, wird in der Projektbeschreibung genauer erläutert, da an dieser Stelle nur einleitend das Grobe zum Spiel angesprochen werden soll.

2. State of the art

Simon erschien mit dem Namen Senso/Simon zum ersten mal im Jahr 1978 als Spiel unter einem vom Jahr 1977 angemeldeten US-Patent (US4207087)[3]. 1979 wurde es zum Spiel des Jahres gekürt.

In den 1980er Jahren war Simon mehr und mehr beliebt. Es konnte als eine Art mobile Spielkonsole (Siehe Abbildung 2) gesehen werden.

Heutzutage ist das Simon Spiel sogar mobil für Android/Apple Smartphones/Tablets erhältlich. So kann man in der heutigen Zeit Simon überall zum Spielen mit dabei haben.

Weiterhin können heutzutage sogar interessante Eingabestatistiken von den Apps aufgezeichnet und Auswertungen über das Merkvermögen und die Reaktionen des Benutzers ausgewertet werden. Somit ist dieses Spiel interessant für HMI und erweiterbar.

Man kann zum Beispiel akustische Störfaktoren einbauen, zu verschiedenen Uhrzeiten Spielverhalten auswerten, die Geschwindigkeit der Wiedergabe der Farbreihenfolge verändern und anschließend das Spielverhalten untersuchen. Dies haben wir bereits in Abschnitt 1.3 auch angesprochen. Dadurch ist vermutlich auch eine analoge Optimierung für andere Software abzuleiten, wie die Reaktionszeit von Benutzern von unterschiedlichem Alter und einiges mehr. Vieles wäre damals vermutlich sehr kostspielig zu erfassen und undenkbar. Mit den heutigen Smartphones hat fast jeder Mensch einen mobilen Computer bei sich, der dies ohne kosten (bis auf die Softwareimplementierung und Stromverbrauch) erlaubt.

Abbildung 2

3. Projektbeschreibung

3.1 Analyse

Die verwendete Entwicklungsumgebung für die Implementierung war Android Studio 3.0.1, da wir schon im Semester vorher in Mobile Devices den Umgang erlernt haben.

Um die Applikation testen zu können benötigten wir ein Android Smartphone mit API 22.

Eine Frage, die sich stellt, ist, welche Altersgruppe und Benutzerklasse das Simon Spiel ansprechen soll und auch welchen Zweck die Benutzung des Spiels mit sich bringt?

Das Simon Spiel ist sowohl für Kinder als auch für Erwachsene ein Spiel welches die Reaktion, Gedächtnis, Konzentration, Wahrnehmung trainiert. Das Simon Spiel zeigt zufällige Farben (beginnend mit einer Farbe) in einer bestimmten Reihenfolge an. Diese Farben muss der Benutzer sich merken und versuchen in der gleichen Reihenfolge über Berührung (Interaktion) wiederzugeben. Sind die Eingaben des Benutzers richtig, wird die Schwierigkeit des Levels erhöht und eine weitere Farbe wird in der nächsten Runde der vorigen Farbreihenfolge zusätzlich angezeigt.

Das Simon Spiel sowohl ein Memory- als auch ein Konzentrationsspiel.

Somit sieht der Benutzer in jeder Spielstufe die vorige Farbreihenfolge (merken durch Wiederholung) und additiv eine neue Farbe.

Vermutlich ist es auf diese Art einfacher viele Farben zu merken, als eine plötzliche lange Reihenfolge ohne jeglichen Wiederholungcharakter, denn wir erinnern uns an die Vorlesung, dass der Mensch sich eigentlich nur 5-9 Elemente merken kann.

In der Vorlesung lernten wir, dass das Hören von bestimmter eingespielter Musik, Tönen oder Lärm die Aufmerksamkeit und somit die Konzentration und das abspeichern im Gedächtnis beeinträchtigt, was wir beim verwenden mit verschiedenen Melodien als Hintergrundmusik bestätigen konnten. Zusammengefasst lässt sich festhalten, plötzliche Schallsignale lenken die Aufmerksamkeit auf sich.

Die Geschwindigkeit der abgespielten Farbreihenfolge ist an Reaktionszeit des Benutzers anzupassen, da ältere Menschen Sinnesreize nicht so schnell verarbeiten und darauf reagieren können. Aber auch junge Menschen können mit einem extremschellen Abspielen nichts anfangen.

Im Zentrum kann das Auge ungefähr bis 25 Bilder einzeln wahrnehmen, und im Augenwinkel sogar mehr.

Der Versuch von uns mit einem Fidget spinner hat bestätigt, dass es einen Unterschied in der Wahrnehmung der Rotationsbewegung in verschiedenen Augenwinkeln gibt und kann gut zur Vorführung dieses Effektes verwendet werden.

3.2 Modell

Es wurde das GOMS Modell verwendet.
“GOMS zerlegt die Benutzerinteraktion mit einem Computer in elementare Aktionen. Diese Aktionen können physisch, kognitiv oder wahrnehmend sein. Mittels der elementaren Aktionen als Bezugssystem können Benutzerschnittstellen untersucht werden. Durch verschiedene Varianten von GOMS lassen sich auch verschiedene Aspekte einer Nutzerschnittstelle genau untersuchen.”[2]

- Goals sind die Ziele des Benutzers und meist in kleinere Aktionen zerlegbar.
- Operators sind die Aktionen, die der Benutzer tätigen darf, sie werden durch die Software festgelegt.
- Methods sind Ketten von Etappenzielen und Operatoren, die zusammen zum Erreichen des Ziels führen können.
- Selection Rules werden von Benutzer eingesetzt, um zu entscheiden, in welcher Situation welche Methode zur Anwendung kommt.

In Bezug auf die Applikation sieht das Modell wie folgt aus:

Goal: Start Game

- Hands bewege Finger über Start-button
- Hands berühre Start

Goal: View color combination

- Look Farbreihenfolge wird abgespielt
- Mental Der Spieler merkt sich die ausgegebene Farbreihenfolge

Goal: Enter color combination

- Mental überlege welche Farbtaste
- Hands bewege Finger über aktuelle Farbabfrage
- Hands berühre die aktuelle Farbe

Rule 1 eingegebene Farbtaste korrekt

- Mental überlege welche nächste Farbtaste
- Hands bewege Finger über nächste Farbabfrage
- Hands berühre die nächste Farbe

Rule 2 eingegebene Farbtaste falsch

- Wait Spiel zuende

- Look Punktestand (Score) wird angezeigt

Rule 3 Ganze Reihenfolge

- Wait Nächster Level startet mit View color combination +1 Farbe zusätzlich

3.3 Implementierung

Der Code ist auch dem Anhang beigefügt, auf einige Punkte der Implementierung möchten wir jedoch hier eingehen.

Durch Aufruf der Funktion anfang() zeigt, dass es nicht einfach ist, ohne Sprünge beim erklären durch den code erklärend zu gelangen. Hier wird zum Beispiel eine neue Arrayliste für die Farbenreihenfolge angelegt. Dem spiellistenzaehler wird hier der Wert der Liste reihenfolgeAbspiel übergeben und anschließend wird die Funktion zufallZusatzNotieren() aufgerufen, in der dann die Funktion zufallsfarbeHinzufuegen() aufgerufen wird.

```
protected void anfang()
{
    /* Spiel startet und beobachtet*/
    spielAn = true;
    beobachterScore.anfrageSenden(BeobachterScore.SCORE_RUECKSETZEN);
    reihenfolgeAbspiel = new ArrayList<>();
    spiellistenZaehler = reihenfolgeAbspiel.iterator();
    zufallZusatzNotieren();
    mainHost.abspiel(reihenfolgeAbspiel);
}
```

Quellcode 1

Um diese Sprünge von Methode zu Methode etwas zu meiden, möchten wir einige Aspekte mit Designvorschau hier aufgreifen, damit der Leser einen visuellen Eindruck des Designvorgehens bei der App auch erhalten kann.

Wie zuvor in Kapitel 3.1 erwähnt ist es wichtig, dass das Tempo der angezeigten Farben dem des Benutzers enstpricht, ob junger oder älterer Benutzer, unerfahrener oder Experte mit Erfahrung. Man hierfür muss ein Kompromiss für alle finden oder es als zusätzliche Einstellungsmöglichkeit implementieren.

```
public static class mainFragment extends Fragment implements View.OnClickListener, Simon.mainHost
{
    final int DAUER_ABSPIELEN_IN_MS = 600;
    final int DAUER_PAUSE_IN_MS = 200;
```

Quellcode 2

Hierfür initialisieren wir die Abspieldauer einer Farbe und die Pausendauer zwischen den Farben in Millisekunden.

Folgend sehen Sie die Deklaration des Typs MediaPlayer um Audiodateien im nächsten Schritt abzuspielen. Hierfür haben wir im Verzeichnis „res“ ein neues Verzeichnis „raw“ angelegt, in das wir unsere Audiodateien abgelegt haben.

```
Button b;
static MediaPlayer mp;
static MediaPlayer mpzonk;
```

Quellcode 3

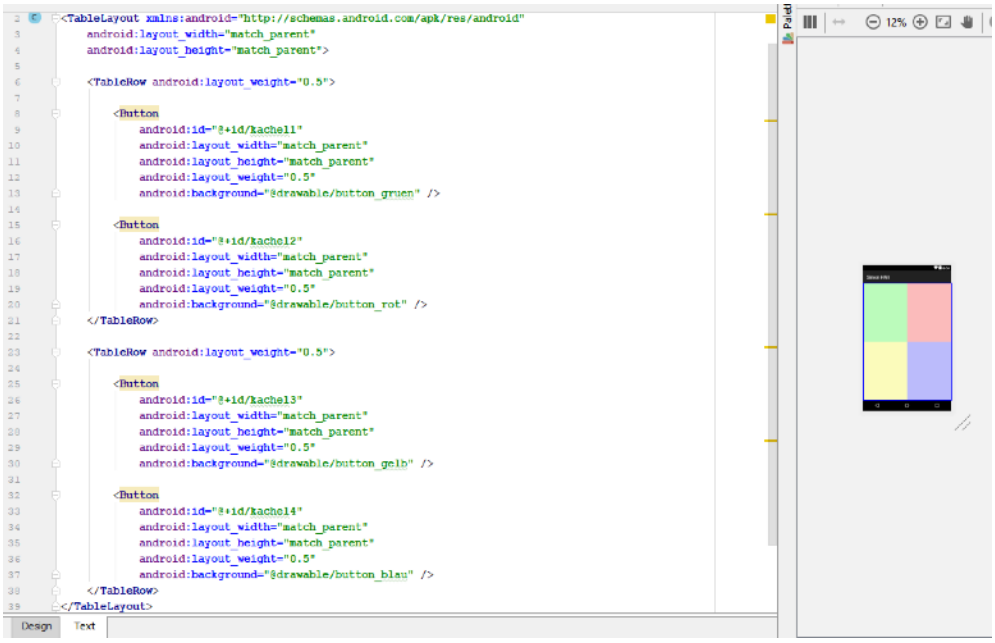
Die Audiodatei kann .wav oder mp3 Format haben und wird mit start(); an beliebiger Stelle einer beliebigen Funktion im Code abgespielt.

```
final MediaPlayer mp = MediaPlayer.create( context: this, R.raw.knightrider);
mp.start();
```

Quellcode 4

Wie man sieht, ist es nicht sehr schwierig in Android Studio auch Audio in eine App einzubinden. Wichtig ist jedoch zu bedenken, an welcher Stelle man ein Audiofile abspielt, wie lang die Dauer davon ist und ob es sich vielleicht wiederholen soll, z.B Aufruf in einer while-Schleife als endlose Hintergrundmusik oder nur einmal in einer Funktion beim Click bzw. Touch auf den Bildschirm.

Als nächstes sehen Sie in der unteren Darstellung links die .xml Datei des 2x2 Spielbretts und rechts eine Vorschau, wie es dann angeordnet auf dem mobilen Gerät aussieht. Bei der .xml für das 3x2 Spielbrett ist dies analog und es wurde noch eine TableRow unten im Code mit Kachel 5 & 6 hinzugefügt.



```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TableRow android:layout_weight="0.5">

        <Button
            android:id="@+id/kachel1"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="0.5"
            android:background="@drawable/button_gruen" />

        <Button
            android:id="@+id/kachel2"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="0.5"
            android:background="@drawable/button_rot" />
    </TableRow>

    <TableRow android:layout_weight="0.5">

        <Button
            android:id="@+id/kachel3"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="0.5"
            android:background="@drawable/button_gelb" />

        <Button
            android:id="@+id/kachel4"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="0.5"
            android:background="@drawable/button_blau" />
    </TableRow>
</TableLayout>
```

Quellcode 5

Das Scorelayout wurde ebenfalls in einer .xml Datei erfasst und sieht wie folgt aus. Hier wird später, wie rechts leider schwer zu erkennen ist, nach Runde die Rundenzahl übergeben und während des Spiels angezeigt (der Punktescore neben Score ebenfalls).

```
<?xml version="1.0" encoding="utf-8"?>


<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:orientation="horizontal">

    <TextView
        style="@style/scoreleiste"
        android:layout_gravity="right"
        android:text="Score:" />

    <TextView
        android:id="@+id/scoreid"
        android:layout_width="50dp"
        style="@style/scoreleiste"
        android:freezesText="true"
        android:layout_gravity="right" />

    <TextView
        style="@style/scoreleiste"
        android:layout_gravity="right"
        android:text="Runde:" />

    <TextView
        android:id="@+id/rundeid"
        android:layout_width="40dp"
        style="@style/scoreleiste"
        android:freezesText="true"
        android:layout_gravity="right" />
```

Quellcode 6

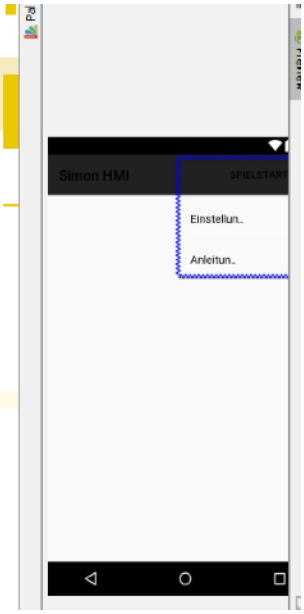
Die Hauptleiste der main.xml ist hier auch noch einmal rechts als Vorschau genauer zu sehen und die Logik dafür befindet sich in den .java Dateien. Spielstart wird durch „always“ immer angezeigt, während Einstellung und Anleitung ausgeblendet („never“) werden und per Touch auf die drei vertikal eingeblendeten Punkte erscheinen.

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.example.orhankrc.simonhmi.MainActivity">

    <item
        android:id="@+id/start_spiel"
        android:title="Spielstart"
        app:showAsAction="always" />

    <item
        android:id="@+id/konfigurieren"
        android:title="Einstellung"
        app:showAsAction="never" />

    <item
        android:id="@+id/hilfe"
        android:title="Anleitung"
        app:showAsAction="never" />
</menu>
```



Quellcode 7

Den Text der Anleitung haben wir im Verzeichnis „values“ unter strings.xml festgehalten. Diesen sehen Sie noch einmal in Kapitel 1.3 Abbildung 1.

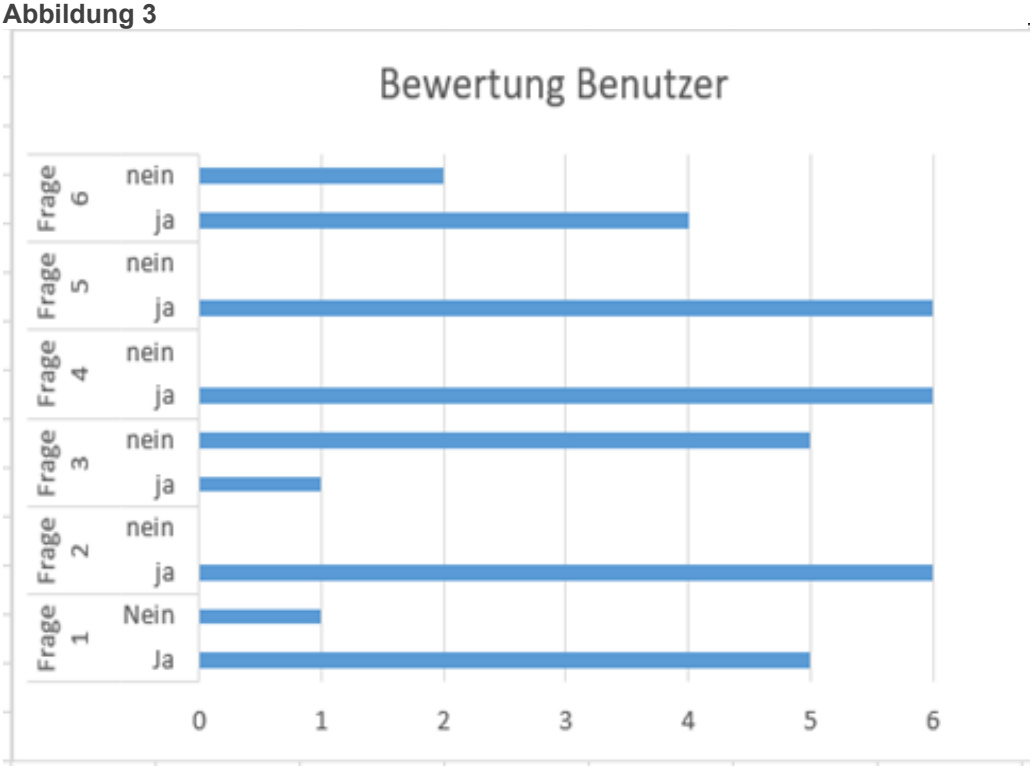
3.4 Evaluierung

Bevor Simon Spiel für die Benutzer zur Verfügung gestellt wurde, gab es ein Testprozess was von den Entwicklern nochmal getestet wurde. Dieser Testprozess dient dazu, falls es Fehler gibt, die auftreten, diese zu erfassen und zu beheben. Der Testprozess verlief so ab:

1. Startet das Spiel Fehlerfrei ohne Absturz?
2. Werden Farben nach der Reihe angezeigt? Stürzt es ab?
3. Ist die Farbeingabe von den Benutzern möglich?
4. Wird der Score korrekt angezeigt und addiert immer eine Zahl dazu?
5. Stoppt das Spiel wenn eine falsche Farbe eingegeben wird?
6. Aktualisiert sich das Display nach dem Ändern der Schwierigkeitsstufe?
7. Wird der User beim beenden des Spiels aufgefordert die App zu schließen?

Nach diesem Testverfahren konnten anfangs Fehler von uns entdeckt werden. Weiterhin gab es 6 Testpersonen aus dem Bekanntenkreis die das Spiel getestet haben. Die Testpersonen füllten anschließend einen Evaluationsbogen aus, der Angaben mit ja/nein zu den zuvor erwähnten Fragepunkten erfasste. Das Ergebnis

Abbildung 3



der Bewertung der Testpersonen haben wir im Diagramm festgehalten (siehe Abbildung 3).

4. Ergebnisse

Probanden konnten sich mehr wie nur 5-9 Farben nacheinander merken und über die Berührung rekonstruieren. Unter Punkt 5, der Diskussion finden Sie zu diesem Ergebnis unsere weiteren Gedankengänge erläutert.

Ein erwähnenswerter Faktor fürs merken war auch, dass die Farben auf programmierebene so gewählt waren, dass man sie leicht voneinander unterscheiden kann. Wählte man den Fortgeschrittenenmodus mit dem Spielfeld von 6 Farben, war es leichter die Reihenfolge zu reproduzieren, wenn die Farben einander nicht sehr ähnlich waren. Unterschiedliche Gelb/Orange Farbtöne waren schwerer zu merken, wie z.B. rot, gelb, blau, grün, braun, violett im Spielfeld, wobei zu bemerken ist, dass man auch die Position der aufgeleuchteten Farbe sich merkt.

Wir haben aber nach dem Vorlesungskonzept aber keine hell aufblendenden Farbtöne verwendet im Kontrast zu dunklen, da eine visuelle Störung durch plötzlich helles Licht oder helles Kachelbild eventuell die Konzentration beim Spielen im Dunkeln beeinträchtigt.

Auch haben wir kein Hermann Gitter erzeugt, um keine Ablenkung durch kreise wahrzunehmen, wobei dies eher bei starken hell dunkel Kontrast wahrzunehmen wäre.

Weitere Ergebnisse können erfasst werden, wenn die Outputs der abgespielten Farbreihenfolge und die Inputs vom Benutzer in einer separaten Datei aufgezeichnet

werden würden. Das Konzept ist einfach, man schneidet die Werte der Variablen bei jeder Aktion in der jeweiligen Funktion mit, wo sich der Wert ändert, und notiert hierbei die fortlaufende Systemzeit in Millisekunden in einer ASCII-Datei. Dieses Konzept haben wir bereits mit Begeisterung im letzten Semester in einer App für „Mobile Application Exercises“ verbaut, jedoch fehlte uns diesmal die Zeit, das Konzept hier zu verwenden oder gar die Werde dann zu analysieren. Das wäre aber ein entschiedener Vorteil einer Simon App gegenüber dem Konsolenspiel der 80er, deshalb erwähnen wir das hier noch einmal.

Abbildung 4

Auch haben wir zum Schluss noch nach jeder eingegebenen Farbreihenfolge ein Feedback über eine sogenannte erscheinende Toastmeldung eingebaut (Abbildung 4) „Super! Jetzt bin ich an der Reihe“, damit eine Art Kommunikationsgefühl während des Spiels mit Simon entsteht und der Spieler weiß, was gerade geschieht. Somit wirkt das Spiel sympathischer.

5. Diskussion

Mit dem Simon Spiel kann, wie wir in der Vorlesung gelernt haben, das kognitive Gedächtnis trainiert werden. Beim Spielen hat man manchmal das Gefühl, man würde die Farbreihenfolge automatisch eingeben, wie zum Beispiel das umschalten des Ganges in einem Pkw oder das Schreiben eines Textes an einer Tastatur mit dem 10 Finger System oder per Hand.

Das merken der Farbreihenfolge müsste demnach über das Langzeitgedächtnis erfolgen, welches für Fakten, Daten und Konzepte zuständig ist, aber auch für Emotionen und Gefühle. Das lässt die Vermutung aufstellen, dass man sich Dinge wie Vokabeln zum Beispiel deshalb leichter merken kann, wenn man es mit einem Gegenstand/Ort, den man bereits in Erinnerung hat, oder einer Emotion oder gar Gefühl verknüpft und anschließend assoziieren kann.

Als Stütze für unsere zuvor formulierte Deutung möchten wir für unser Spiel Simon an dieser Stelle aus der Vorlesung bezüglich Assoziation zitieren:

„-Aufnehmen neuer Informationen, Erlernen.
-Die kognitiven Prozesse versuchen, die neuen Fakten und Eigenschaften mit den bereits vorhandenen zu verbinden und zu verknüpfen.

-Je besser das gelingt, desto mehr festigt sich das bereits vorhandene Wissen und desto einfacher werden neue Informationen dem Langzeitgedächtnis dauerhaft hinzugefügt.“[4]

Der Mensch kann sich nämlich ansonsten maximal zwischen 5-9 Elemente gleichzeitig merken. Elemente sind z.B. Zahlen, Bilder oder zusammengesetzte Zeichen.

Durch das Wiederholen, werden vermutlich Verknüpfungen für kurze Zeit, aber nicht dauerhaft im Gehirn gefestigt und vermutlich als eine zusammengesetzte Zeichenfolge gemerkt, so dass sogar mehr Farben in Kombination zu merken sind.

6. Literatur

https://developer.android.com/guide/components/intents-filters.html

[1] https://de.wikipedia.org/wiki/Mensch-Computer-Interaktion

[2]https://de.wikipedia.org/wiki/GOMS

[3]https://de.wikipedia.org/wiki/Senso_(Spiel)

[4]Vorlesungsunterlagen HMI Prof. Dr. –Ing. Matthias Deegener

https://instrumentalfx.co/download-knight-rider-theme-song/

soundbible.com/524-R2D2-Again.html

Abbildung 2: https://commons.wikimedia.org/wiki/File:Simon_game.jpg