

PROJEKTABSCHLUSSBERICHT

Multimediasysteme



Guess it!

eingereicht von

Bahara MURADI, Dmitry MAKLAKOV, Elias BÜRGER,
Sebastian WINDSPERGER, Richard HOANG

Datum der Fertigstellung

14. Juni 2020

Übungsleiterin: Frau Manuela POLLAK

INHALTVERZEICHNIS

Kapitel 1: Filter und Kompression	3
<i>Main-Programm und Kommunikation</i>	<i>3</i>
<i>Bildbearbeitung</i>	<i>3</i>
<i>Textbearbeitung</i>	<i>5</i>
<i>Audiobearbeitung</i>	<i>6</i>
<i>Herausforderung</i>	<i>6</i>
Kapitel 2: Backend	7
<i>Kurzbeschreibung</i>	<i>7</i>
<i>Ablauf</i>	<i>7</i>
<i>Schwierigkeiten</i>	<i>7</i>
Kapitel 3: Applikation	9
<i>Kommunikation</i>	<i>9</i>
<i>Design</i>	<i>9</i>
Herangehensweise	9
Implementierung	10
Assets	10
Animationen	10
Retrospektive	11
Reflexion	12
Literaturverzeichnis	13

KAPITEL 1: FILTER UND KOMPRESSION

MAIN-PROGRAMM UND KOMMUNIKATION

Das Kompression und Filter Programm haben wir aufbauend auf die Idee von Microservices¹ entwickelt. Grundidee dabei war, dass der Server ein externes Programm aufruft, welches sich um die Bearbeitung der Daten kümmert. Da wir dieses Kompression-/Filtersprogramm „stateless“ implementiert haben, hat es nur mehr eine Funktionseigenschaft die aufgerufen und nach der Bearbeitung terminiert wird. Dies entspricht der UNIX-Philosophie „Do One Thing and Do It Well“²

Das Java-Programm haben wir als Maven-Projekt entwickelt. Das Main-Programm bekommt CLI Befehle. Diese werden dann mittels CommandLineParser abgearbeitet indem die gewünschten Filter ausgeführt werden. Ein Beispiel für einen solchen Command Aufruf samt Parameter schaut wie folgt aus:

```
java -jar MMS_Project_CompressionFilter-0.1.0-SNAPSHOT.jar -t image -i test.png -o newFilteredTest.png  
-f1 blur -l1 3 -f2 zoom -l2 7 -f3 noise -l3 6
```

(-t soll den Typ der Quelldatei, -i der Eingabedatei, -o die Ausgabedatei, -f die gewünschten Filter und -l den gewünschten Schwierigkeitsgrad bestimmen).

BILDBEARBEITUNG

Zur Entwicklung der Bildfilter ist es empfehlenswert eine Library auszuwählen, um schnell eine Vielzahl an Filtern zu entwickeln. Für dieses Projekt haben wir uns für ImageJ³ entschieden. ImageJ ist ein Open-Source-Bildverarbeitungsprogramm für mehrdimensionale Bilddaten mit dem Schwerpunkt auf wissenschaftlicher Bildverarbeitung (imagej, 2021).

Für die Bildfilter haben wir ein Interface *Filter* erstellt, um eine Grundstruktur vorzugeben. Dieses Interface hat zwei Methoden. Die Methode *getFilterName()* liefert den Namen des Filters zurück. Die zweite Methode *runFilter()* ist um einiges wichtiger. Diese dient dazu, um den Filter auf ein Bild anzuwenden. Die Methode hat zwei Parameter. Der erste ist vom Typ ImagePlus (das Bild, dass zu bearbeiten ist). Um ein Bild mit dem Typ ImagePlus zu erhalten, muss als ein solches eingelesen werden. Dies kann mit dem Befehl *IJ.openImage(path)* gemacht werden. Der zweite Parameter dient dazu, die Schwierigkeit festzulegen, also wie stark das Bild verändert werden soll. Es wird kein Rückgabewert benötigt, da das übergebene Bild direkt verändert wird.

Alle Filterklassen implementieren das Interface. Sie erstellen mit *image.getProcessor()* ein ImageProcessor des aktuellen Bilds. Auf diesen können die Methoden zur Bearbeitung des Bildes angewandt werden. Die Schwierigkeit der Filter wird meist mit einem *switch*-Abfrage oder einer *while*-Schleife (mehrmaliges anwenden desselben Filters) eingestellt.

Insgesamt haben wir 14 Filterklassen gemacht. Als Beispiel stellen wir einige dieser Klassen vor:

¹ <https://de.wikipedia.org/wiki/Microservices>

² <https://de.wikipedia.org/wiki/Microservices>

³ <https://imagej.nih.gov/ij/index.html>

Der PartsOfPic-Filter dient dazu zufällige Teile des Bildes mit Blöcken zu überdecken. Dazu werden mit einem *switch*-Abfrage die Anzahl an Blöcken festgelegt. Mit Random werden die Positionen der Blöcke auf dem Bild definiert.

Der EdgeToMid-Filter ist ähnlich zum PartsOfPic-Filter. Es wird zentral in der Mitte des Bildes ein Block platziert, sodass anfangs nur die äußeren Bereiche des Bildes erkennbar sind, nach und nach zieht sich dieser Bereich zusammen und es werden mehr Teile sichtbar.

Der Draw-Filter dient dazu Linien quer über das Bild zu zeichnen und dadurch das Bild schwerer zu erkennen zu machen. Die Linien beginnen und enden an den Außenkanten des Bildes. Entweder sie werden von oben nach unten oder von links nach rechts über das Bild gezogen.

Der MedianFilter ersetzt jedes Bildpixel durch den Median der Pixel in der aktuellen Filterregion R. (Burger & Burge, 2016)



Abbildung 1: MedianFilter mit Stufe 3

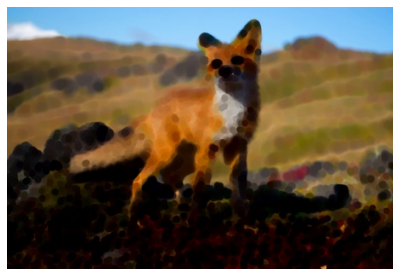


Abbildung 2: RankFilter mit Stufe 3



Abbildung 3: ColorInvert

4s

Chat

Funny Turtle guessed correctly

Brave Unicorn guessed correctly


Funny Snake guessed correctly

Silent Panda guessed correctly

Funny Turtle panda

Funny Turtle guessed correctly

Round 2 / 10



your guess here

	Name	Points
	Funny Turtle	18 Points
	Brave Unicorn	7 Points
	Funny Snake	6 Points
	Silent Panda (you)	3 Points

Abbildung 4: DrawFilter, ZoomFilter und NoiseFilter kombiniert

TEXTBEARBEITUNG

Auch für die Text Filter haben wir ein Interface angelegt. Dieses ist sehr ähnlich zu dem der Bildfilter, jedoch wird ein Text-String übergeben und ein bearbeiteter String zurückgegeben. Die Implementierung schaut wie folgt aus:

ShuffleWords-Filter mischt die Buchstaben eines Wortes. Die 3 verschiedene Schwierigkeitsstufen entscheiden, welche Mischung durchgeführt werden soll. Die ersten zwei Stufen zufällig, einmal wird der Text in Kleinbuchstaben und einmal in Großbuchstaben zurückgegeben. Die letzte Stufe ordnet die Buchstaben aufsteigend.

FillInTheBlanks-Filter ist nach dem Prinzip des Galgenmann programmiert. Die Schwierigkeit ist in zwei Stufen geteilt. Das klassische Weggeben von Lauten bis das Weglassen von Buchstaben, die in der englischen Sprache am häufigsten vorkommen.

Für den QuotedPrintable-Filter sind Teile aus der Übung übernommen worden. Dieser Filter muss jedoch auch verändert werden, da verschiedene Schwierigkeitsstufen benötigt werden. Dazu wird ein Prozentanteil festgelegt, der angibt, wie viele der Buchstaben, welche den Wert 127 überschreiten, durch ihre andere Form ausgetauscht werden sollen. Da, wenn die Schwierigkeit verringert wird, nicht andere Buchstaben decodiert & codiert werden sollen, muss eine allgemeine Folge festgelegt werden.

Der Telephone-Filter verändert den Text so, dass man nur die Nummern sieht, die gedrückt werden müssen, sodass ein Buchstabe auf einem Tastentelefon eingegeben wird.

Der BLang-Filter bringt Texte in die B-Sprache bzw. B+D-Sprache. Ist das aktuelle Zeichen ein Konsonant und BLang ist aktiviert, wird an dieses Zeichen b und nochmal der Konsonant angehängt. Bsp.: Er -> Eber -> Ebeder (B+D Lang).

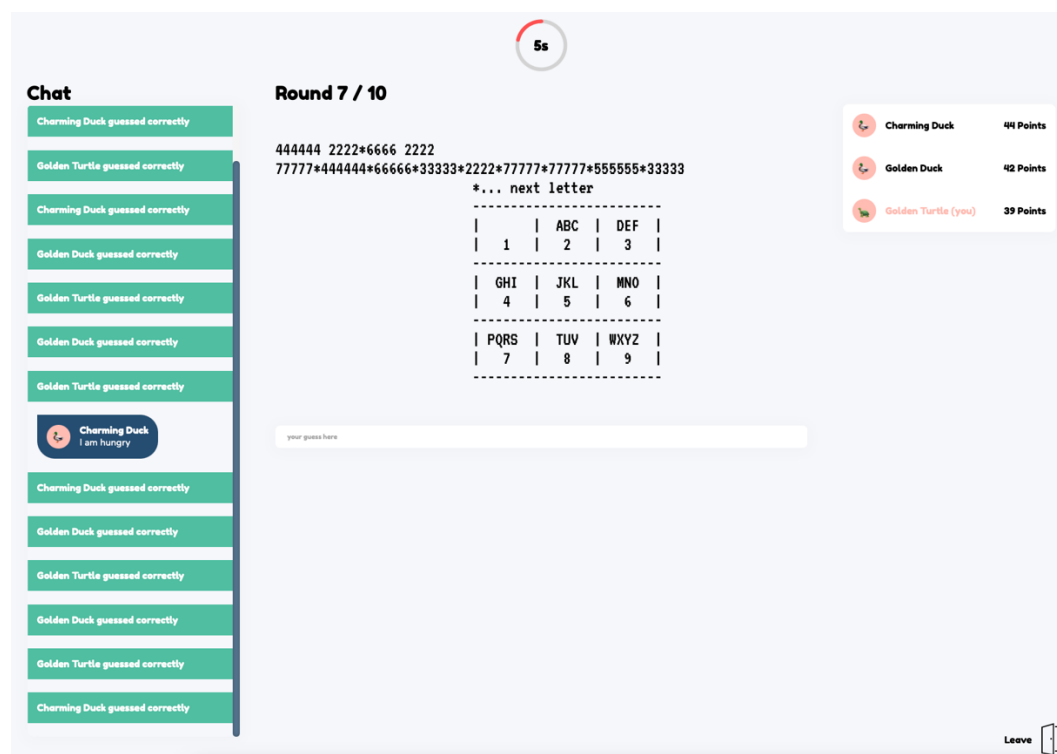


Abbildung 5: TelefonFilter mit der höchsten Schwierigkeitsstufe

AUDIOBEARBEITUNG

Für die Audiotbearbeitung haben wir uns entschieden, ein anderes Programm zu verwenden, die das Filtern von Audios übernimmt. Eine passende Möglichkeit hat uns SoX⁴ angeboten. SoX ist ein plattformübergreifendes (Windows, Linux, MacOS X usw.) Befehlszeilenprogramm, das verschiedene Formate von Computer-Audiodateien in andere Formate konvertieren kann. Es kann auch verschiedene Effekte auf diese Audiodateien anwenden, und als zusätzlichen Bonus kann SoX Audiodateien auf den meisten Plattformen abspielen und aufnehmen.

Die Filterart wird mit einem If-Else abgefragt, dann wird die passende Klasse aufgerufen, diese liefert dann die richtigen Werte für den Filter und dessen Schwierigkeit zurück. Diese Klassen führen folgende Funktionen aus.

Downsample-Filter verringert die Qualität des Filters unterschiedlich stark, sodass es immer schwieriger wird das Audio zu erkennen. Fast-Filter beschleunigt und Slow-Filter verlangsamt das Audio. Treble-Filter fügt ein Störgeräusch über das Audio. Reverse: Spielt das Audio rückwärts ab.

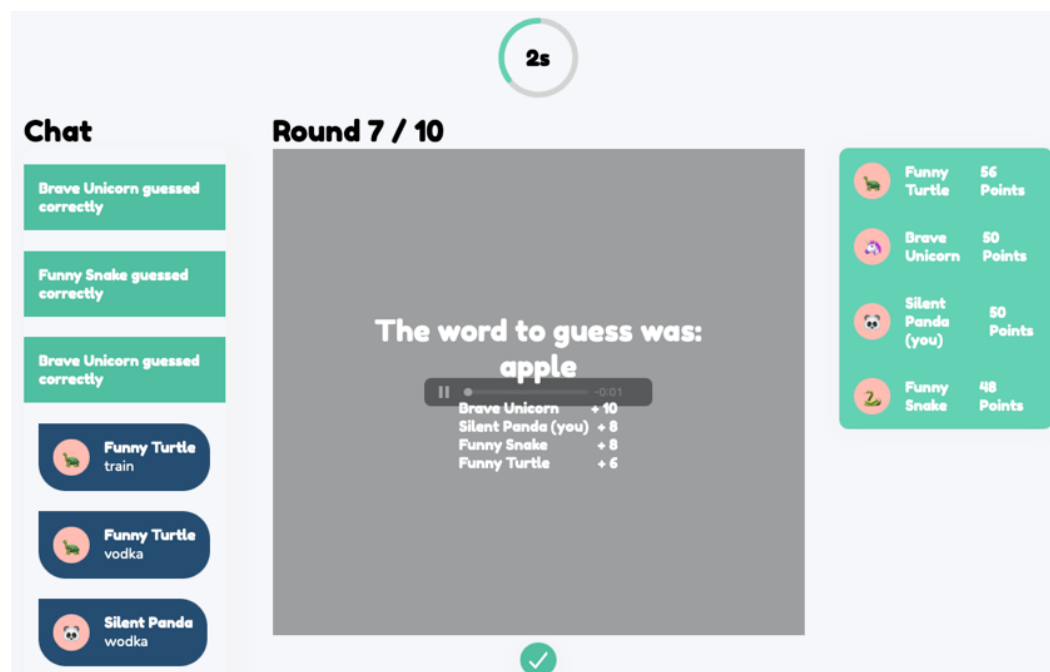


Abbildung 6: AudioFilter

HERAUSFORDERUNG

Eine große Sache bei der Implementierung des Filter-und-Kompressionsprogramms war die Recherche nach guten und schnellen Libraries, „sich-in-der-Kommunikation-und-in-den-Bibliotheken-zurecht-zu-finden“ und ein Microservice zur Verfügung zu stellen, welches weitgehend entkoppelt von den restlichen Modulen ist. Damit konnten wir eine parallele Entwicklung des Projekts gewährleisten.

⁴ <http://sox.sourceforge.net>

KAPITEL 2: BACKEND

KURZBESCHREIBUNG

Das Ziel vom Backend ist es, verschiedenen Clients die Verbindung zum Server zu ermöglichen und Spiele zu managen. Dabei werden über Websocket-Verbindungen Nachrichten mit den Clients ausgetauscht.

ABLAUF

Die Clients verbinden sich mittels Spiel-Codes. Dabei wird entweder für diesen Code ein neues Spiel erstellt/oder dem Spiel mit dem Code beigetreten. Danach können noch weitere Spieler beitreten.

Wird das Spiel gestartet, beginnt der Server Wörter und deren zugehörigen Bilder/Texte/Audiodateien auszuwählen. Für jedes Wort wird außerdem noch eine zufällige Menge an Filtern ausgewählt. Während des Spiels werden dann die Bilder, Texte und Audiodateien gefiltert und an die Spieler geschickt.

Ein Beispiel-Spiel als Sequence Diagram ist auf der Abbildung 1 zu sehen. Die verschickten Daten wurden in Sätze umformuliert. Der tatsächliche Datenaustausch erfolgt mittels JSON.

SCHWIERIGKEITEN

Ein großer Punkt für die Implementierung war das Anwenden der Filter während der Laufzeit. Da alle gefilterten Dateien am Anfang zu generieren zu langsam wäre, werden die Filter immer in separaten Threads während des Spiels generiert.

Ein Spiel besteht aus mehreren Runden. Eine Runde besteht aus einem Bild/Wort/Audiofile mit mehreren Levels. Das Level gibt an, mit wie vielen Filtern das Bild/Wort/Audio gefiltert wird. Mit der Zeit wird das Level kleiner und somit wird es weniger stark gefiltert und ist leichter zu erkennen. Erreicht das Level 0, wird die nächste Runde gestartet. Außerdem wird auf die nächste Runde gewechselt, falls alle Spieler richtig geraten haben.

Grundsätzlich wird während jedem Level das Bild fürs nächste Level gefiltert. Da es jedoch immer sein kann, dass alle vorzeitig richtig raten, muss zumindest 1 Bild für die nächste Runde bereit liegen. Deshalb wird von Anfang für jedes Wort ein gefiltertes Bild generiert.

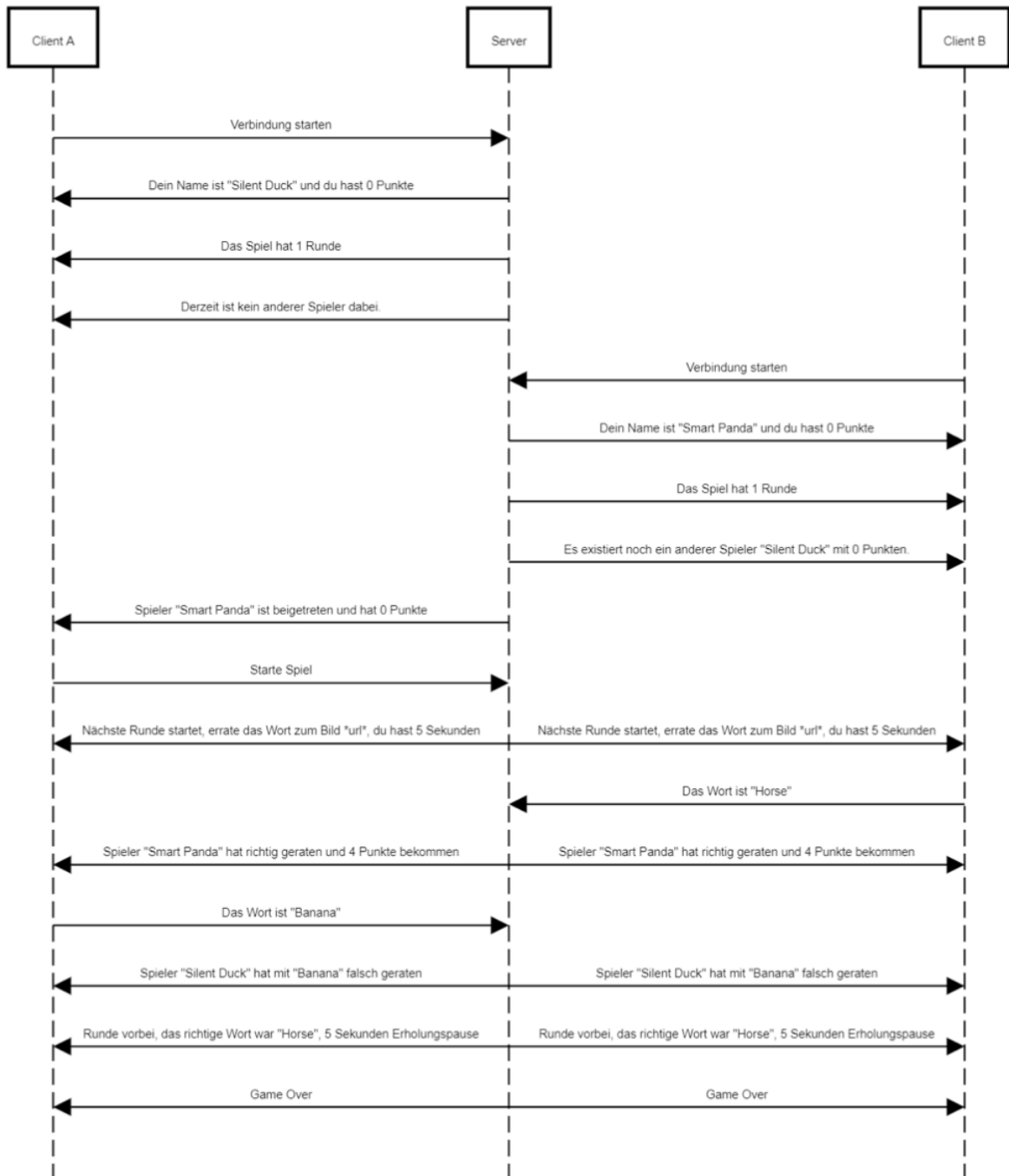


Abbildung 7: Demospiel

KAPITEL 3: APPLIKATION

KOMMUNIKATION

Für die Webapplication wurde das Framework „Angular“ verwendet. Diese Technologie, in Verbindung mit Sprachen HTML (HTML5), CSS (CSS3) und Typescript, erlaubt eine moderne Webapplication zu entwickeln.

Am Anfang kommt man auf die Startseite „/home“, wo man entweder ein neues Spiel starten kann („Start“) oder ein Spiel beitreten („Join“). In beiden Fällen kommt User in die Lobby („/lobby“), derzeit wird auch die Verbindung zum Server erstellt. Zuerst wird eine Message mit dem Gamecode geschickt, als Antwort bekommt man zwei Messages: „ASSIGN_PLAYER“ und „CURRENT_PLAYERS“. Noch zwei Messages kann man im Lobby bekommen, wenn jemand noch das Spiel beitrifft oder verlässt. Das Ganze wurde mithilfe von Subscription und Observable realisiert. Der Spieler, der das Spiel erstellt hat, kann das Spiel anfangen („Start“). Alle Spieler werden auf die Gameseite („/game“) redirected und das Spiel wird angefangen. Es gibt drei verschiedene Typen von Aufgaben, die Information darüber wird in der Message „NEXT_ROUND“ gespeichert. Um den passenden Block zu zeigen, wurde die Verzweigung *ngIf verwendet. Das Spiel besteht aus mehreren Runden (Rounds), die aus mehreren Schwierigkeitsstufen bestehen (Levels). Spieler schreiben Vorschläge in Textfeld und als Antwort bekommen „GUESSED_CORRECT“ oder „GUESSED_WRONG“. Falls die Vermutung richtig war, wird die Anzahl von Punkten erhöht. Am Ende der Runde werden die richtige Antwort und die neuen Punkte gezeigt. Nach der letzten Runde wird „GAME_END“ Message bekommen und man ins „Leaderboard“ („/leaderboard“) weitergeleitet.

Die folgende Tabelle kann man mit der Abbildung 1 gut kombinieren.

Message	Beschreibung
ASSIGN_PLAYER	Dem User zugeordneter Spieler (Name, Bild)
CURRENT_PLAYERS	Alle Spieler im Spiel
PLAYER_JOINED	Neuer Spieler ist das Spiel beigetreten
PLAYER_LEFT	Ein Spieler hat das Spiel verlassen
NEXT_ROUND	Information für die neue Runde: Typ der Aufgabe (Text, Bild oder Audio), Schwierigkeit
WORD_SOLUTION	Die richtige Antwort für die vorherige Runde
NEXT_LEVEL	Die neue Schwierigkeit der Runde
GUESSED_CORRECT	Information, dass ein Spieler richtig geantwortet hat
GUESSED_WRONG	Ein Spieler hat falsch geantwortet
GAME_END	Das Spiel ist beendet →

Tabelle 1: Message

DESIGN

HERANGEHENSWEISE

Für das Design wurde mit dem Design-Tool „Figma“ zuerst ein Wireframe erstellt. Ein Wireframe dient dazu, ein grobes Bild zu bekommen, wie das Layout aussehen wird und welche Elemente wo hingehören, dabei wird noch nicht berücksichtigt, wie diese Elemente aussehen. Aufbauend auf diesem Wireframe wurde dann der eigentliche Entwurf erstellt mit den wesentlichen Elementen, welche die Webseite haben sollte.

Da es sich hier um ein Spiel handelt, wurde im Design-Prozess darauf geachtet die UI- Elemente passend zu gestalten. Das Thema wurde auch bei der Auswahl der Fonts und Farben entsprechend reflektiert.

IMPLEMENTIERUNG

Die Webseite wurde mit dem populären Web-Framework Angular zusammen mit SCSS für das Styling entwickelt.

Was ist SCSS?

SCSS oder auch Sass steht für „Syntactically Awesome Style Sheets“ und ist ein Superset von CSS. Das heißt jeder valide Code in CSS ist ebenfalls gültig in SCSS, da SCSS bei der Kompilierung in CSS umgewandelt wird. Die Beschreibungssprache bietet neben dem herkömmlichen CSS viele Features, die den Entwicklungsprozess um einiges vereinfachen. Ein solches Feature ist die Verschachtelung von Styles. Damit lässt sich so eine Hierarchie entsprechend der HTML Hierarchie abbilden.

BEM

Ist eine Namenskonvention für CSS-Klassen und steht für „Block Element Modifier“.

- Block: ist ein Standalone-Element, Benennung nach Zweck (z.B. container, menu)
- Element: ist Teil eines Blocks, zum Blocknamen wird der Elementname getrennt mit zwei Unterstriche hinzugefügt (z.B. container_element, menu_item)
- Modifier: kennzeichnet das Verhalten oder Aussehen eines Blocks oder Elements, zum Block- oder Elementnamen wird die Eigenschaft mit zwei Bindestriche getrennt hinzugefügt

Warum BEM?

BEM erlaubt es wiederverwendbare CSS-Klassen zu schreiben und gleichzeitig die Übersicht der Klassen beizubehalten. Außerdem wird das HTML-Template lesbar, da man durch die Klassen sofort erkennt, welches Element zu welchem gehört und was es macht.

BEM ist auch sehr kompatibel mit SCSS. Man muss in SCSS bei der Verschachtelung nicht jedes Mal die gesamte Klasse ausschreiben, sondern kann auf den Namen der Elternklasse referenzieren und diese erweitern.

ASSETS

Sämtliche Logos, Icons sowie Illustrationen wurden selbst in „Figma“ gestaltet und als SVGs exportiert. Der Vorteil von SVG ist, dass diese Vektorgrafiken frei skalierbar sind und ihre Qualität dabei beibehalten. Zusätzlich kann man mit SVG in Kombination mit CSS komplexe Animationen erstellen. So eine Animation findet man auf der Homepage der Webseite.

ANIMATIONEN

Um die Website lebendiger zu gestalten und um den Anschein zu erwecken, dass es sich hier wirklich um ein Spiel handelt, wurde sich darum bemüht sinnvolle Animationen einzubauen.

Was ist mit sinnvoll gemeint? Jede Animation sollte einen bestimmten Zweck erfüllen und nicht nur als Kosmetik verwendet werden. Zum Beispiel trägt die Animation auf der Homepage zur

Erklärung bei, wie das Spiel funktioniert. Es gibt auch genügend Mikro- Animation, die das Spielerlebnis verbessern, ohne zu sehr vom eigentlichen Geschehen abzulenken.

RETROSPEKTIVE

Während der Implementierung des Designs kam es zu einigen Änderungen und ein paar Problemen. Die größte Hürde war definitiv die Synchronisation des Timers mit der Animation und dem Server.

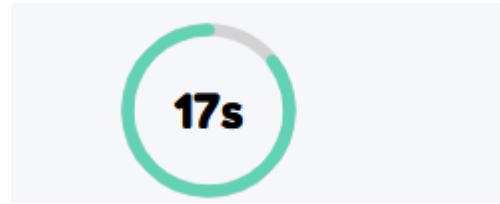


Abbildung 8: Timer

Das Problem: Der Server schickt jede Runde einen Zeitintervall, bis das nächste Medienobjekt kommt, mit. Um die Animation mit der angezeigten Zeit zu synchronisieren, muss sie eine Sekunde vorher beginnen. Würde die Animation gleichzeitig mit der Anzeige starten, würde sie nie komplett abschließen. Zusätzlich sollte der Timer die Nullte-Sekunde anzeigen, um die Spieler zu signalisieren, dass das nächste Level kommt oder eine Runde vorbei ist.

Der Lösungsansatz dazu ist relativ simpel, die mitgeschickte Zeit wird für die Anzeige um eine Sekunde verringert. Die eigentliche Schwierigkeit dabei war die Realisierung, da der Timer ein SVG-Element ist, musste mit CSS, SVG-Attributen und Angular gearbeitet werden.

REFLEXION

In unserer Projektarbeit haben wir uns mit den Themen der Multimediasystemen beschäftigt.

Da wir bereits im Rahmen der Lehrveranstaltung Multimediasysteme das Grundlegende erlernen konnten, wollten wir in dieser Arbeit auf diese Kenntnisse zurückgreifen. Daher lag es in unserem Interesse Themen der Lehrveranstaltung in der Praxis zu implementieren.

Die Implementierung hat mehr Zeit benötigt als vorgegeben (in Summe ca. 200 Stunden anstatt 100 Stunden). Abgesehen vom benötigten Zeitaufwand, verlief der Rest des Entwicklungsprozesses ohne größere Probleme.

Die Projektarbeit war für uns ein sehr spannendes und neues Erlebnis sowohl im akademischen als auch im zwischenmenschlichen Bereich. Wir konnten durch unser selbstständiges Arbeiten innerhalb des Projekts und das gemeinsame wöchentliche Treffen Problemlösungen gemeinsam bewältigen.

Rückblickend sind wir mit unserem Projekt bzw. unserer Teamarbeit sehr zufrieden.

Das Projektteam

LITERATURVERZEICHNIS

Burger, W., & Burge, M. (2016). *Digital Image Processing*. Springer-Verlag London.

imagej. (8. 06 2021). // Von <https://imagej.net> abgerufen

VERZEICHNIS DER INTERNETQUELLEN

<https://de.wikipedia.org/wiki/Microservices>

<https://imagej.net/learn/>

<https://imagej.nih.gov/ij/index.html>

<http://sox.sourceforge.net>

ABBILDUNGSVERZEICHNIS

Abbildung 1: MedianFilter mit Stufe 3.....	4
Abbildung 2: RankFilter mit Stufe 3.....	4
Abbildung 3: ColorInvert.....	4
Abbildung 4: DrawFilter, ZoomFilter und NoiseFilter kombiniert	4
Abbildung 5: TelefonFilter mit der höchsten Schwierigkeitsstufe.....	5
Abbildung 6: AudioFilter	6
Abbildung 7: Demospiel.....	8
Abbildung 8: Timer.....	11

TABELLENVERZEICHNIS

Tabelle 1: Message	9
--------------------------	---