

# Be[lat] the Virus



# Einleitung

## Teamvorstellung

Wir sind Team zwei aus dem Sommersemester 2020. Unser Team besteht aus Diandra Hermann, Florian Grünwald, Marie Sasse und Leander Schmidt. In dieser Dokumentation beschreiben wir unsere Simulation „Be[at] the Virus“.

## Problem

Jedes Jahr erkranken und sterben viele Menschen an Infektionskrankheiten, wie COVID-19, Diphtherie, Ebolafieber, Grippe und vielen mehr. Einfache Maßnahmen, wie das richtige Händewaschen oder Tragen eines Mund-Nasen-Schutzes, können viel bewirken. Doch Wissenschaftler fanden heraus, dass die deutsche Bevölkerung eine unzureichende Hygienapraxis verrichtet.

## Unsere Simulation

Diesem Problem wollen wir uns annehmen und klären Kinder im Alter von acht bis dreizehn Jahren durch unsere Simulation über die Wichtigkeit und Durchführung einer wirkungsvollen Hygienepraxis auf.

## Vielen Dank

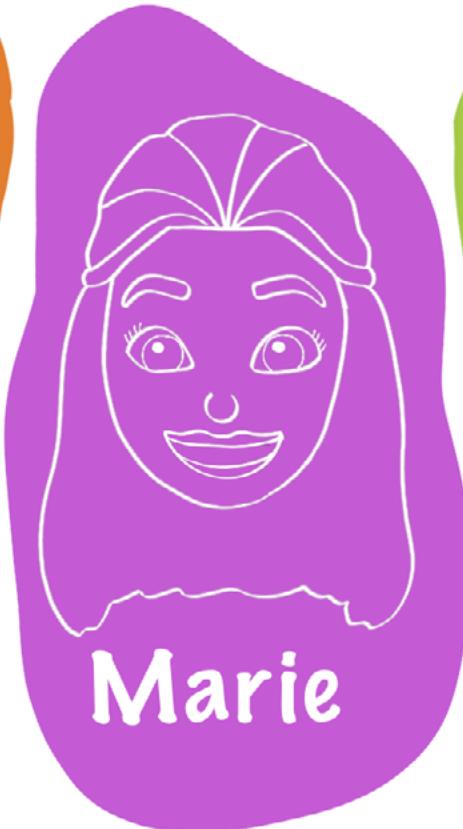
In diesem Semester drehte sich alles rund um die COVID-19 Pandemie. Aufgrund dieser hat das gesamte Semester präsenzfrei stattgefunden. Hier möchten wir einen besonderen Dank an alle Professoren, Dozenten, Tutoren und Kommilitonen ausrichten, die dieses Semester überhaupt möglich gemacht haben.



Leander



Diandra



Marie



Flo

# Inhaltsverzeichnis



# Konzept

Konzeptänderungen  
Händewaschen  
People Bouncy

# Händewaschen

## Änderungen

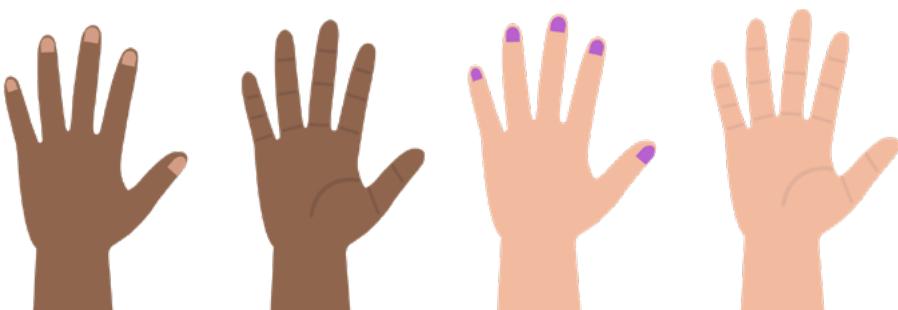
Während der Umsetzung unserer Simulation sind wir bei dem Spiel Händewaschen etwas von dem Konzept abgewichen. Wir haben gemerkt, dass der geplante Durchgang zu starr und langgezogen wirkt. Um dem entgegenzuwirken, haben wir zwei Änderungen vorgenommen.

## Wechselnde Hände

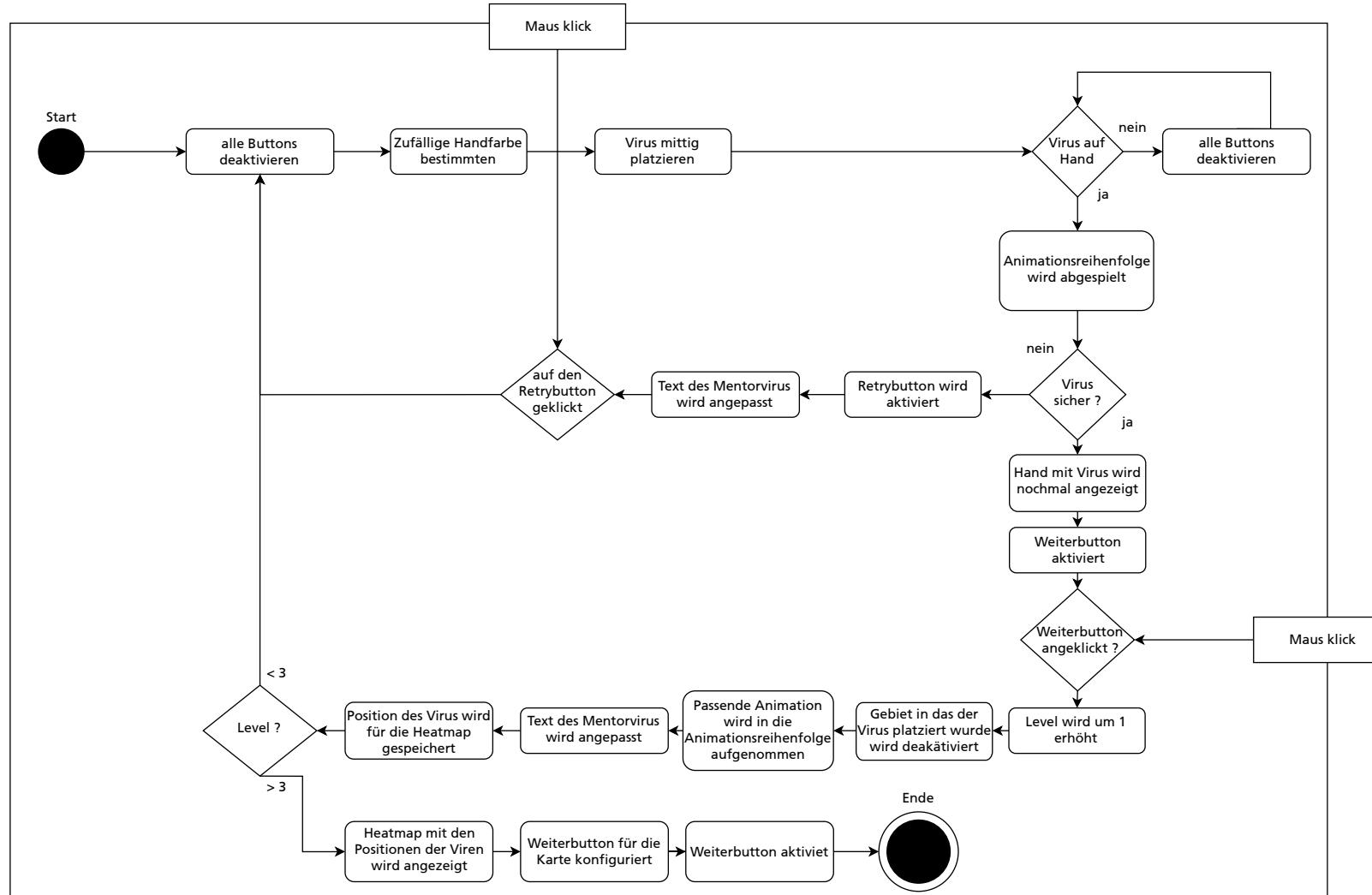
Sobald die nutzende Person in das nächste Level kommt, oder das Level wiederholen muss, wechseln sich die dargestellten Hände. So entsteht der Eindruck, dass man nun eine neue Person infizieren muss. Durch die Abwechslung bleibt die Aufmerksamkeit der Nutzenden Person erhalten. ?Aufmerksamkeitsspanne oben?

## Dynamischer Ablauf

Den wohl größten Effekte hatte die Änderung des Spielablaufs. Im Konzept beschrieben wir diesen als linear und gleichbleibend. Nun ist er jedoch dynamisch gestaltet. Das heißt, nachdem die nutzende Person den Virus platziert hat, wird immer die dazu passende Animation abgespielt. So ist die nutzende Person gezwungen aufmerksam die Schwachstellen der Händehygiene herauszufinden und das Spielerlebnis ist interaktiv und abwechslungsreich gestaltet.



# Händewaschen



# People Bouncy

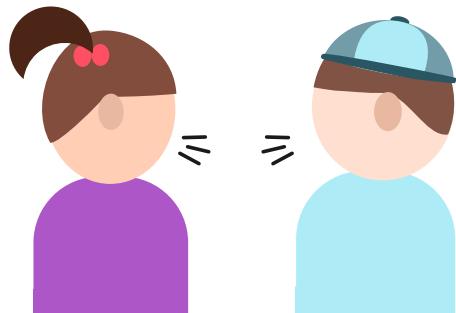
## Konzept

Während wir People Bouncy programmierten, haben wir einige Konzepte zum Spielprinzip ergänzt und andere verändert oder verworfen.

## Ergänzungen

Als erstes haben wir uns dazu entschieden, dass der Wecker Countdown beim Start des ersten Levels nicht sofort beginnt, um allen Kindern die Möglichkeit zu geben, sich zu orientieren.

Weiterhin haben wir kleine Striche an den Mündern der Figuren beim Husten ergänzt, da die Aktion nur mit dem Soundeffekt nicht richtig beim Spielenden angekommen ist.



## Veränderungen

Nach einigen Testläufen entschieden wir uns dazu, die Reihenfolge und den Sinn der Level zu ändern.

Im ersten Level werden die People nun vergrößert dargestellt, um den Raum um sie herum kleiner wirken zu lassen. Auch haben die People so weniger Platz sich zu bewegen und es ist einfacher sich als Erreger zu verbreiten. Durch die Vergrößerung der People, sind auch der Virus über deren Kopf und die Striche beim Husten größer und somit besser zu erkennen.

Im zweiten Level haben die People nun zur Hälfte Masken an und befinden sich auf dem Pausenhof. Hier ist die Verbreitung schwieriger, da die Masken die Flugweite reduzieren und die kleineren Personen deutlich mehr Platz haben, sich zu bewegen.

# People Bouncy

Hier soll nun deutlich werden, dass Masken nicht besonders viel bewirken, wenn kein Abstand eingehalten wird und zu viele die Maske nicht tragen.

Im dritten Level haben alle People Masken auf und halten, so gut es geht, einen Abstand ein. Jetzt sollte es fast unmöglich sein, sich als Erreger zu verbreiten. So soll letztendlich deutlich werden, dass nur durch Einhalten von Abständen und das Tragen von Masken eine Ausbreitung wirklich eingedämmt werden kann.

## Verbreitung trotz Maske

Wir haben bei der Entwicklung natürlich auch andere Ideen ausprobiert, unsere Botschaften zu vermitteln. So haben wir getestet, wie es sich spielt, wenn die Masken die Wahrscheinlichkeit ändern, mit der ein Virus beim Drücken der

Leertaste erzeugt wird. Dies fanden wir aber wenig befriedigend, da es sich komisch anfühlt, wenn ein Drücken der Taste keine sichtbare Auswirkung hat. Selbiges gilt für den Fall, wenn Masken den People eine gewisse Wahrscheinlichkeit geben, bei einem Treffer nicht angesteckt zu werden. Der/die Spielende wird sich eher fragen, ob er/sie nicht richtig getroffen hat oder, ob das Spiel fehlerhaft sei, als die eigentlich Botschaft zu verstehen.

Letztendlich war eine Flugweiten-Minimierung die Option, welche sich am besten angefühlt hat und auch ohne Erklärung am verständlichsten ist.

Im letzten Level reduzierten wir die Personen Anzahl auf Hannah und einen People für unsere Hauptperson. Dies taten wir, um die Dramaturgie der Szene zu erhöhen und, um die Zerreißten Animation einheitlich darstellen zu können.



# People Bouncy

Diese war am Ende ein Problem, da die People sonst immer zufällig verteilt wurden und wir so nicht einfach ein Video einbauen konnten, in welchem der Screen zerreißt. Daher entschieden wir uns für die oben genannte Szene, um eine immer gleiche Ausgangslage für das Video zu schaffen.

## Verworfen

Einige Konzepte haben wir bei unserem Prototyp erstmal ausgelassen oder verworfen, da diese entweder technisch für uns in der Zeit nicht umsetzbar waren, oder doch nicht in die Simulation sollten.

Grafisch entschieden wir uns dazu die Richtungspfeile an den Figuren wegzulassen, da die Figuren bei uns nur in vier Hauptrichtungen schauen können und diese klar zu erkennen sind.

Beim Anstecken entfernten wir die Bedingung, dass die People sich gegenseitig anschauen müssen, um den Erreger zu übertragen, da die Figuren sich zu zufällig zueinander bewegen und drehen. Auch sind wir bei dieser Einschränkung auf das Problem gestoßen, was überhaupt passieren soll, wenn der Virus jemanden von hinten trifft. Wenn der Erreger einfach verschwindet, sieht das zum einen komisch aus und fühlt sich zum anderen auch falsch an. Daher haben wir uns entschieden, diese Beschränkung zu entfernen.



# Technik

[View Hierarchy](#)

[Action Hierarchy](#)

[Technische Umsetzung](#)

[Klassendiagramm](#)

# View Hierarchy

Wir haben uns am Anfang der Umsetzung dazu entschieden, eine View Hierarchy umzusetzen. Nach dieser strukturierten wir unsere Klassen und Screens. Wir setzten diese so um, dass wir jedem Objekt beliebig viele Unterobjekte zuweisen können.

So haben wir ein „großes“ World Objekt erstellt, welches dann wiederum unsere Szenen als Unterobjekte beinhaltet. Die Szenen wiederum beinhalten alle Unterobjekte, welche zu ihnen gehören und so weiter.

Dies erleichterte uns die Arbeit enorm, da die Position der Unterobjekte immer relativ zu ihren Überobjekten definiert ist. So konnten wir zum Beispiel den Virus über den Kopf der jeweils aktiven Figur in People Bouncy sehr einfach programmieren.

Weiterhin wird immer erst das Überobjekt gezeichnet, darüber dann die Unterobjekte. So konnten wir bestimmen, welches Bild über welches, gezeichnet wird.

Auch werden nur die Objekte aktualisiert, deren Überobjekte aktiviert sind. Dies ermöglichte uns einen einfachen Szenenwechsel und sorgte allgemein für einen sehr strukturierten Code mit erleichterter Fehlersuche.

Wir haben uns entschieden, dass alle Klassen ab InteractiveObject Unterobjekte beinhalten können, da in unserer Anwendung nur InteractiveObjects Unterobjekte haben müssen.

Alles in allem ist unsere View Hierarchy eine der wichtigsten Grundlagen, auf welcher unser Prototyp beruht.

# View Hierarchy

Konzept

Technik

Design

Management

## Unterobjekt

Szene Händewaschen

## Unter-Unterobjekt

Hände

## Unter-Unterobjekt

Button Weiter

# Action Hierarchy

## Weiterentwicklung

Als wir die View Hierarchy programmiert haben, ist uns aufgefallen, dass wir die Struktur dieser auch nutzen könnten, um Inputs und Updates zu verwalten.

### Inputs

So haben wir beschlossen, dass alle Inputs (zum Beispiel ein Klicken mit der Maus), die ein Überobjekt treffen, an dessen Unterobjekte weitergegeben werden. Dies verbesserte die Performance, da die Seite nicht mehr alle Objekte durchgehen und dabei testen muss, ob diese aktiv sind und vom Mausklick getroffen werden, sondern nur die Unterobjekte, deren Überobjekte auch aktiv sind und angeklickt wurden. Weiterhin ist so eine klare Reihenfolge vorgegeben, in der die Objekte ihre Aktionen durchführen.

### Update und init()

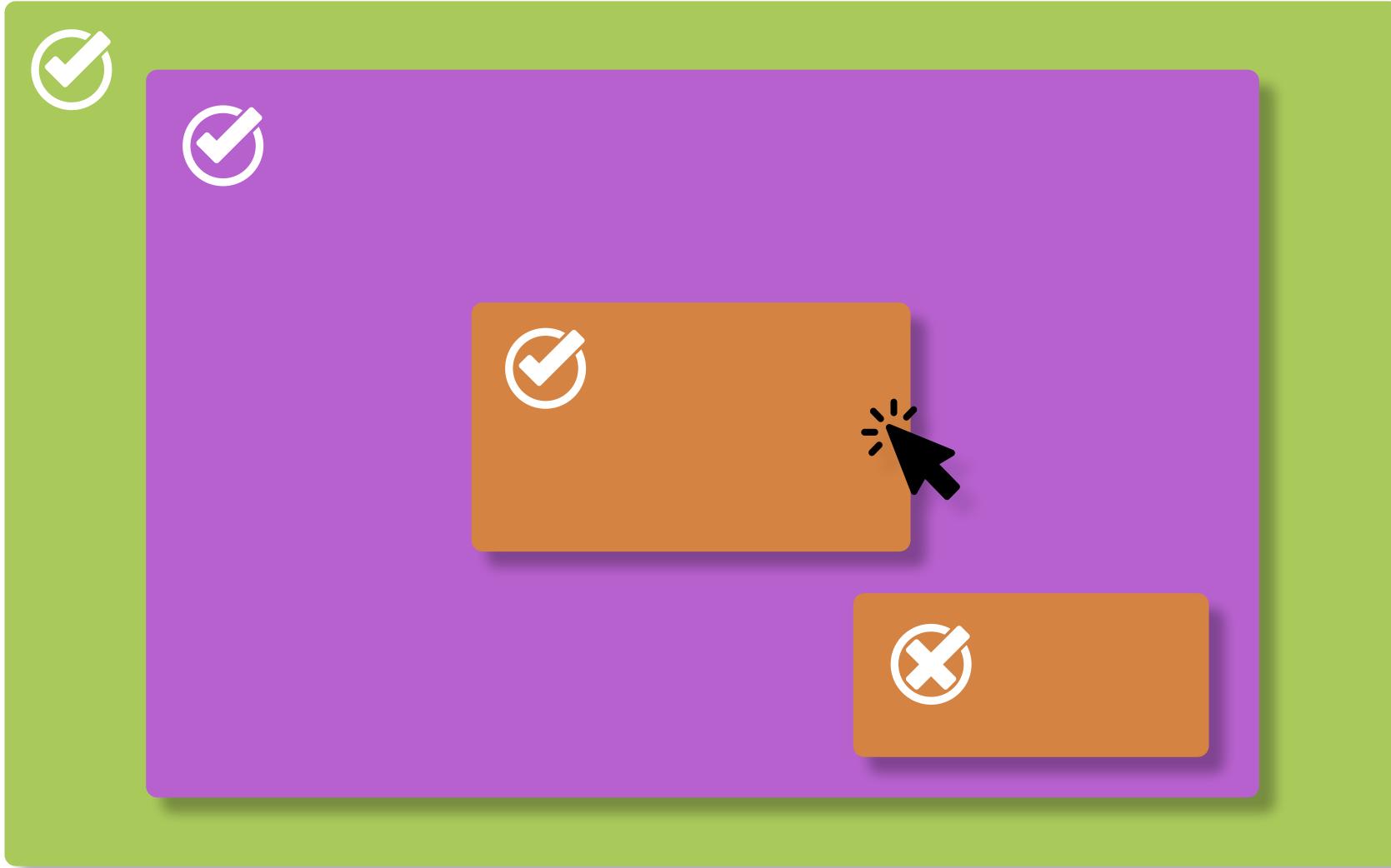
Ähnlich ist es mit den Funktionen update() und init() welche auch in einer festen Reihenfolge von den Überobjekten bei den Unterobjekten aufgerufen werden.

Für die init() Funktion haben wir uns entschieden, da der Konstruktor eventuell aufgerufen werden könnte, bevor die p5 draw ausgeführt wird. Dies könnte dazu führen, dass die Bilder noch nicht geladen sein könnten, da nicht sicher gestellt ist, dass das preload() zu Ende bearbeitet wurde.

Deshalb führten wir die init() Methode ein, welche erst mit dem Start der draw() Funktion aufgerufen wird.

Allerdings stießen wir auf das Problem, dass die Kopplung von Über- und Unterobjekt init() zu Problemen in der Definitionsreihenfolge führen kann. Für dieses Problem haben wir leider noch keine gute Lösung gefunden.

# Action Hierarchy



# Technische Umsetzung

## Verlauf der technischen Umsetzung

Gestartet haben wir unsere Umsetzung in der Technik damit, dass wir uns nochmal unser Klassendiagramm angeschauten und daraus die Oberklassen gebaut haben.

So begannen wir damit, erstmal DisplayObject und InteractiveObject zu erstellen und zu testen. Bei den Tests bemerkten wir schnell, dass wir einige wichtige Variablen und Methoden in unserem ersten Klassendiagramm vergessen hatten und ergänzten diese nachträglich.

So arbeiteten wir uns von oben nach unten durch unser Klassendiagramm durch, bis wir Szenen mit bewegten Objekten und Buttons erstellen konnten. Bevor wir allerdings diesen Teil fertigstellen konnten, kamen noch zwei größere Probleme auf.

# Probleme

## Globale Variablen

Das erste war, die Bilder, Schriften oder Sounds, welche in der p5setup.js geladen wurden den anderen Dateien zur Verfügung zu stellen. Als erstes lösten wir dies über ein ENUM Objekt, welches wir an window anhängten, um es überall aufrufen zu können. Später kam dann der Tipp, dass wir dies auch über Import und Export lösen konnten, was bei uns aber zu zahlreichen Bugs geführt hat. Im Nachhinein könnten diese Bugs durch einen anderen verzweigteren Bug mit der Libarie p5sound zu tun gehabt haben. Auf Grund dieses Bugs, welchen wir erst kurz vor Ende mit Hilfe lösen konnten, verwendeten wir aber bis zum Schluss unser ENUM Obejekt. Dieses erwies sich generell als sehr praktisch, da wir dort auch andere Konstanten, wie die ID der verschiedenen Hitboxformen speichern und unter einem Namen zugängig machen konnten. So wurde unser Code deutlich lesbarer, da nirgendwo eine Zahl übergeben werden musste, welche keine Koordinate oder Zeit war.

# Globale Variablen

Button.js

window.ENUM.SHAPEROCT

People.js

window.ENUM.SHAPEROUND

window.ENUM.IMAGEPEOPLE

window

ENUM.

SHAPE.

RECT: 1

ROUND: 0

IMAGE.

PEOPLE: img,

BUTTON: img,

...

# Probleme

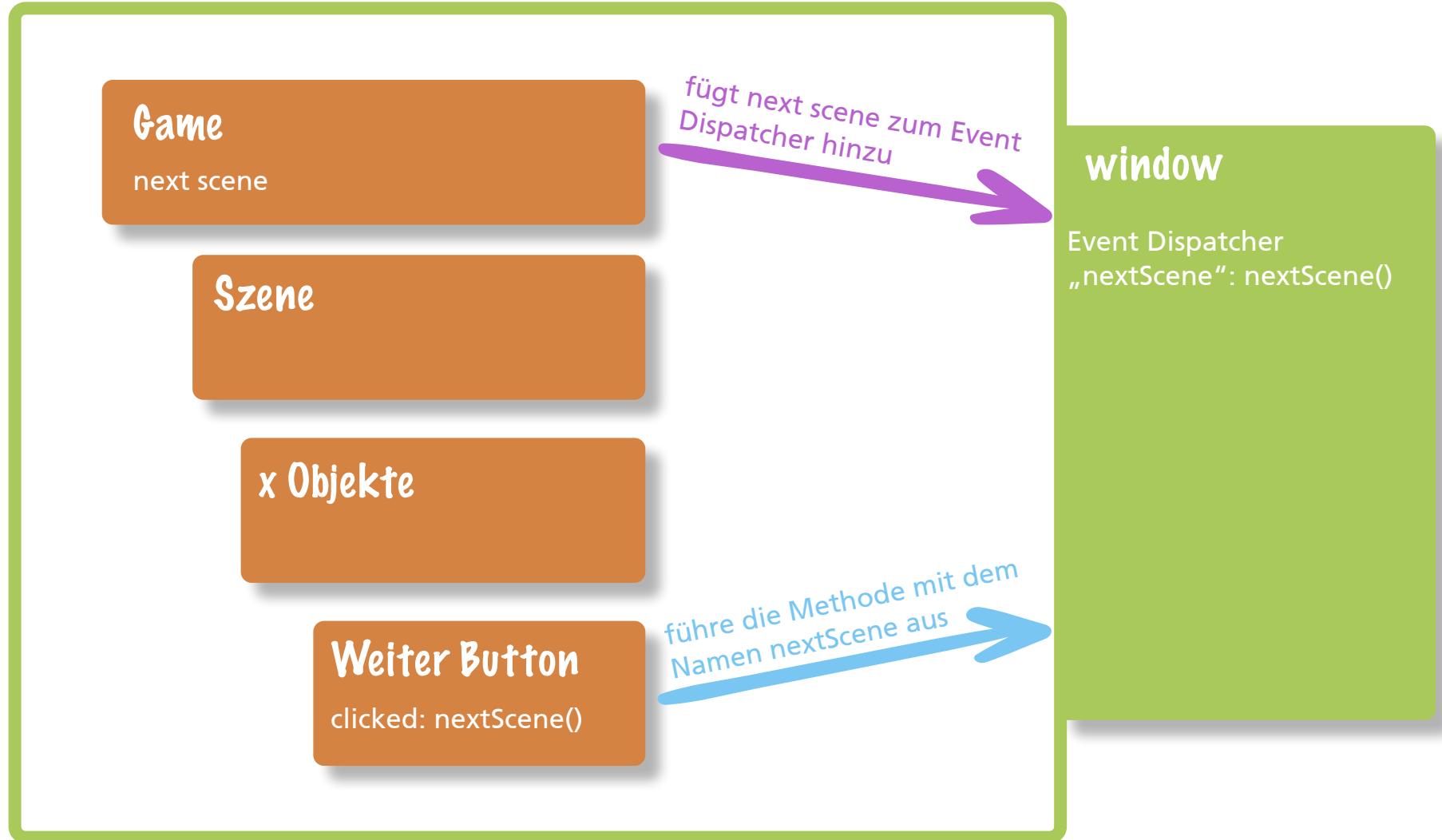
## Methoden aus anderen Objekten aufrufen

Das zweite große Problem auf welches wir stießen war, wie wir eine Funktion von einem Übergeordneten Objekt von einem beliebig weit Untergeordneten Objekt aufrufen können.

1. Wir stießen dabei auf einen simplen EventDispatcher, welchen wir an unser window anhängen konnten. Dieser sammelt praktisch Funktionen aus verschiedenen Objekten und speichert diese unter einem bestimmten Namen ab. Nun kann man aus egal welchem Objekt dem EventDispatcher Objekt mitteilen, dass dieser eine gewisse Funktion in einem anderen Objekt ausführen soll. Diese Lösung war auf alle Fälle die beste, welche wir gefunden haben.
2. Die Alternative wäre gewesen, auf das X. Überobjekt zu zugreifen, was allerdings die Wiederverwendbarkeit der Klassen beeinträchtigt hätte, da diese dann immer als X. Unterobjekt des entsprechenden Überobjektes instanziiert werden müssten. Theoretisch hätten wir auch die jeweilige Funktion durch die Unterobjekte übergeben können, was aber wiederum zu noch mehr Verwirrung geführt hätte.

# Event Dispatcher

1.



# Parent Aufruf

Konzept

Technik

Design

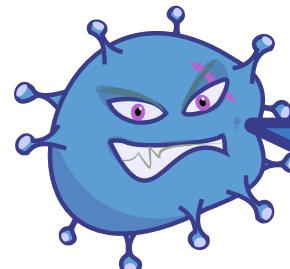
Management

2.

Game  
nextScene()

Szene

Weiter Button  
clicked: nextScene()



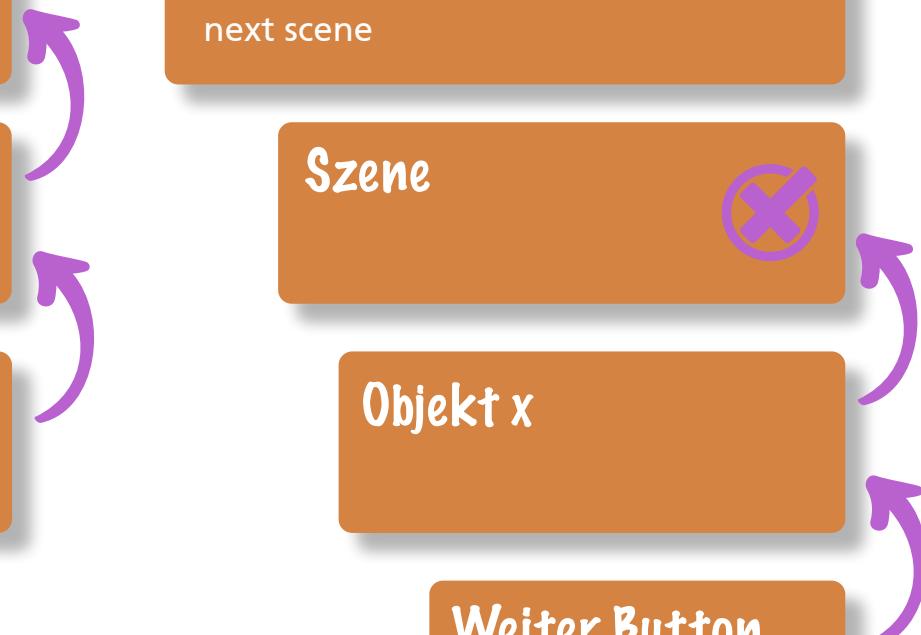
So nicht!

Game  
next scene

Szene

Objekt x

Weiter Button  
clicked: nextScene()



# Szenenverwaltung

Als wir diese zwei großen Probleme gelöst hatten, konnten wir endlich mit der eigentlichen Umsetzung beginnen.

## Erste Idee

1. Als erstes hatten wir den Gedanken, dass jedes Spiel eine eigene Unterklasse von Game wird und die einzelnen Level dann als Szenen an diese angehängt werden. Dies erwies sich aber als unnötig, da sich unsere Level nicht so stark von einander unterschieden, um für diese wirklich einzelne Klassen zu erstellen.  
Der Code in den Szenen hätte sich wahrscheinlich relativ schnell gedoppelt, was wir unbedingt

mit unserer View Hierachy und mit der Objektorientierung vermeiden wollten.

## Bessere Lösung

2. Nach dieser Erkenntnis, kam uns aber die Idee, dass wir diese Dopplungen vermeiden können, wenn wir nur ein Game Objekt als oberstes Objekt erstellen und an dieses dann die einzelnen Simulationen und Screens als Szenen anhängen. So entstand letztendlich unsere Szenenverwaltung und jede Simulation/jeder Screen wurde als eine Unterklasse von Szene geschrieben.

2.

Game

Händewaschen

Flappy Mask

People Bouncy

# Szenenverwaltung

1.

Händewaschen

Level 1

Level 2

Level 3

Flappy Mask

Level 1

Level 2

Level 3

People Bouncy

Level 1

Level 2

Level 3



# Untersimulationen

Als die Szenen Struktur nun klar war, konnte es wirklich mit unseren Untersimulationen losgehen.

Als erstes programmierten wir die Untersimulation Händewaschen, da uns diese am einfachsten erschien und wir die oben genannten Konzepte erstmal an einem echten Teil unserer Simulation testen wollten.

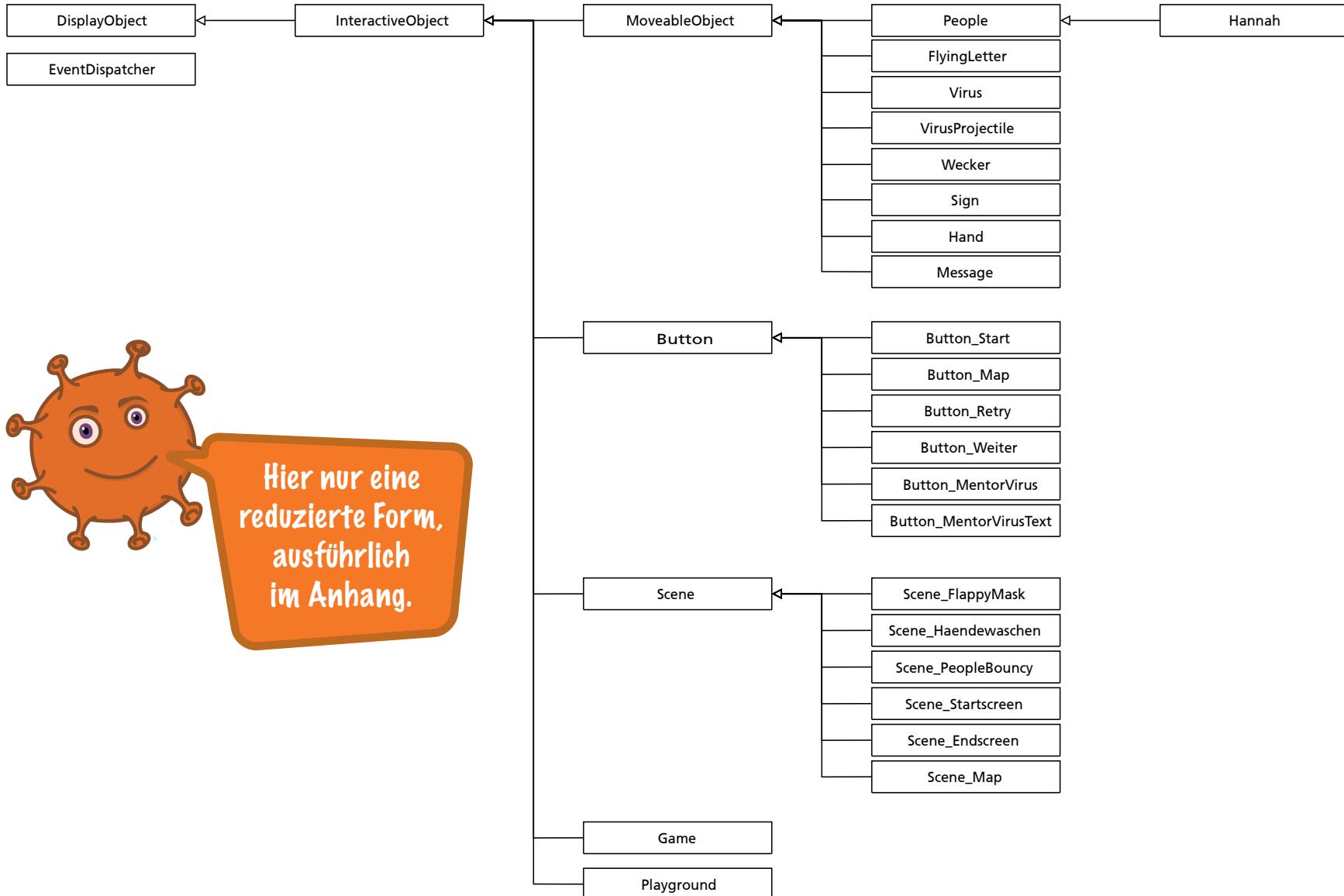
Nachdem unsere Struktur den ersten Test gut überstanden hat, setzen wir eine erste Version der Karte und des Startscreens um.

Schließlich begannen wir als letztes mit People Bouncy und überarbeiteten währenddessen auch nochmal die anderen Szenen.

Durch unser modulares Arbeiten konnten wir für People Bouncy viele Objekte aus Händewaschen übernehmen, wie z.B. den Mentorvirus oder den Weiterbutton.

Als letztes erstellten wir dann die Webseite und alles, was um den Canvas zu sehen sein sollte. Ein kleines Problem blieb ungelöst, leider entsteht beim Überblenden zu oder nach den Videos ein kurzes flackern.

# Klassendiagramm



# Design

Designprozess  
Elemente  
Animationguide  
Sound

# Designprozess

## Vorgehensweise

Die assets für unseren Prototypen haben wir stets nach Coding Anspruch gestaltet. So war jede Woche klar, welcher Teil als nächstes kommt. Dadurch war auch sofort klar, welches Bild als nächstes gestaltet werden musste. Deswegen haben sich meistens ein Gestalter und ein Programmierer im Zweierteam zusammengetan.

Beispielsweise haben wir beim Start der Programmierung von Händewaschen parallel den Hintergrund geschaffen. Als die Animationen der Hände im Code eingebunden wurden, hat parallel jemand anders die Animationen dafür erstellt.

So konnten wir gründlich von Sprint zu Sprint alles wichtige gestalten und es direkt im Team bewerten. Außerdem haben wir dadurch einen gestalterischen "Flickenteppich" vermieden.

## Eigene Elemente

Bei Gestaltung der Screens nutzten wir lizenzenfreie Vektorgrafiken von freepik.com und änderten diese so ab, dass sie einheitlich aussehen und zum Stil der Simulation passen.

Einige Elemente haben wir auch neu erstellt, um ein noch einheitlicheres Bild zu schaffen. Dabei nutzten wir unsere Hausfarben, sowie einen flachen 2D Stil ohne Outlines.

# Elemente

## Hände

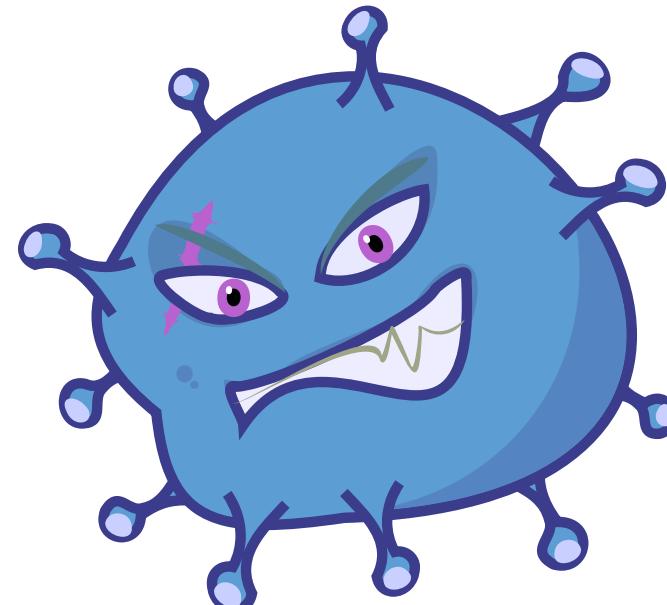
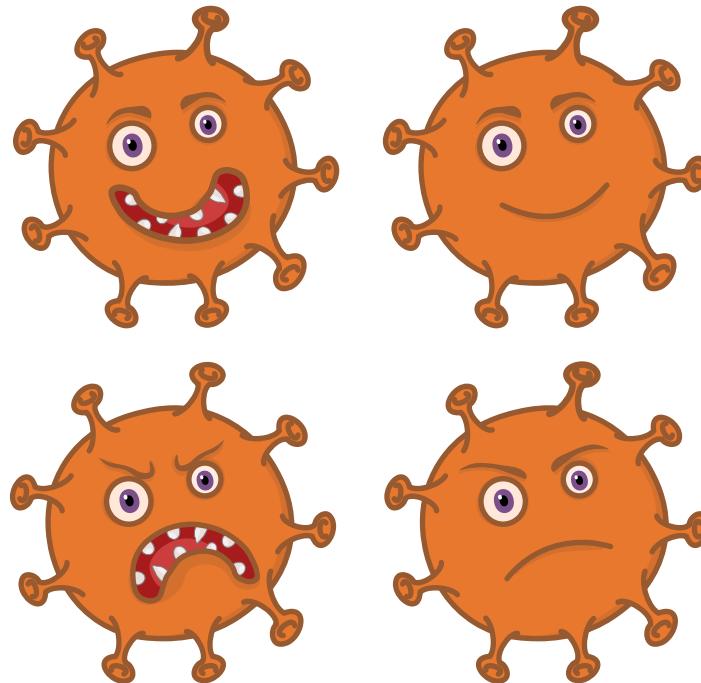
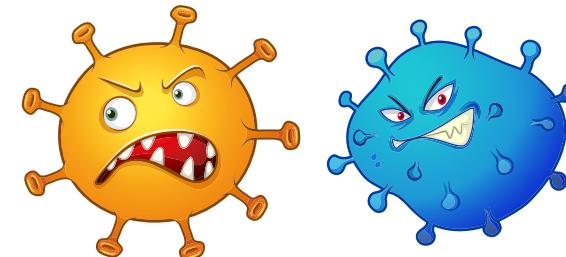
Marie gestaltete die Hände, auf die man den Virus platzieren muss, neu, da sie realistischer aussehen sollten und so besser zu den Händewasch-Animationen passen.



# Elemente

## Viren

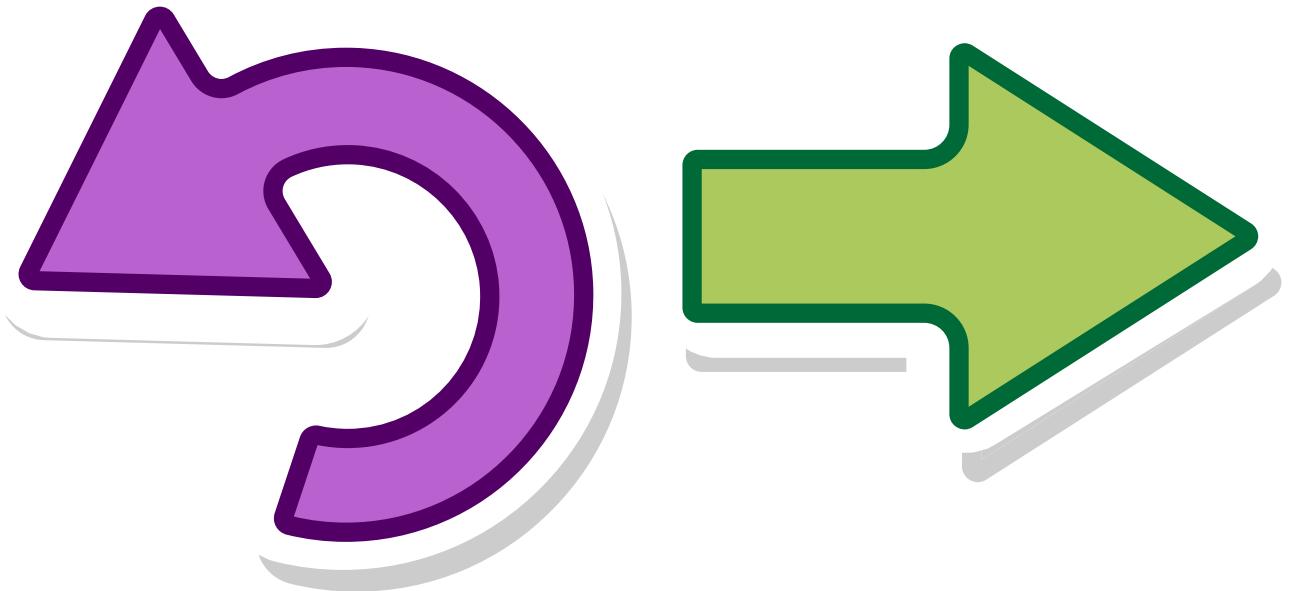
Flo gestaltete unsere beiden Viren neu, da sie nicht zu unserem Stil passten. Sie haben nun weniger Glanz und neue Details, wie eine Narbe am Auge und verschiedene Gesichtsausdrücke.



# Elemente

## Button

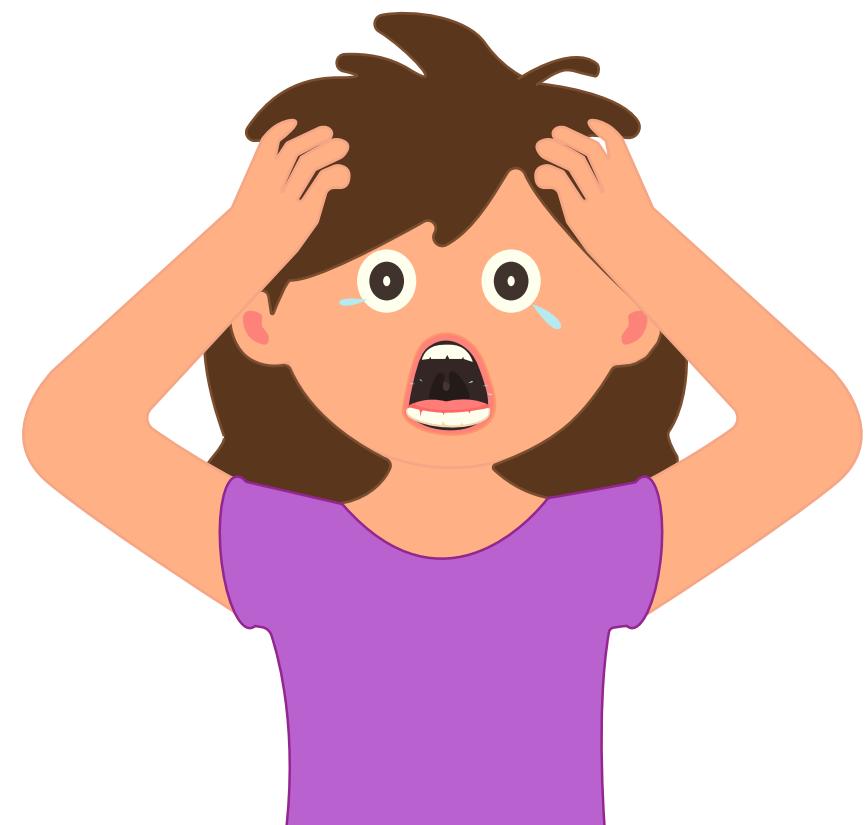
Leander bearbeitete den Weiter und Retry Button. Sie waren zuvor zu rundlich und nicht einheitlich gestaltet. Sie haben eine deutliche Kontur, da sie sich so vom Spielfeld abheben und besser erkennbar sind.



# Elemente

## Mädchen

Diandra erstellte das Mädchen, welches in unserer Anwendung den Traum hat. Die erste Version war zu jung und passte nicht zur Simulation. Außerdem erstellten sie auch auf der Karte ein Mädchen, um konsistent zu bleiben. Zuvor wechselt der Charakter zwischen männlich und weiblich.



# Animationguide

## Easing

Bei der Erstellung der Animationen haben wir besonderen Fokus auf ein passendes Easing gelegt und versucht möglichst keine linearen Bewegungen in die Simulation einzubauen. Dazu zählen neben allen Buttons auch die Bewegungen der People in People Bouncy.

## EaseOut

Bei den Animationen der interaktiven Objekte (z.B. Button), wird ein EaseOut Easing genutzt, damit die Nutzenden ein möglichst schnelles Feedback erfahren. Dies haben wir vor allem in Hinblick auf unsere Zielgruppe bedacht.

## Programmierung

Die meisten Animationen wurden auf eigene Weise in Java Script umgesetzt. Die Animationsgeschwindigkeit entspricht dem linearen Zeitfortschritt pro Frame, dabei ist 1 das Ende der Animation. Die Animationsgeschwindigkeit wird durch die angegebene Easing Funktion in einen nicht-linearen Fortschritt umgewandelt. Diese mathematischen Umwandlungs-Funktionen haben wir von der opensource Plattform easings.net bezogen.

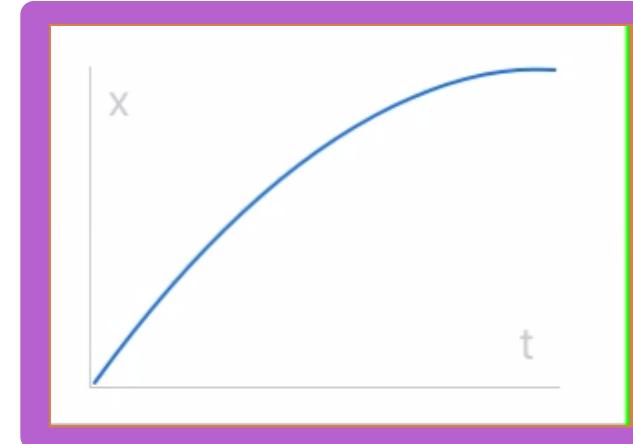
Die von uns verwendeten Easing Funktionen sind nachfolgend dargestellt.

# Animationguide

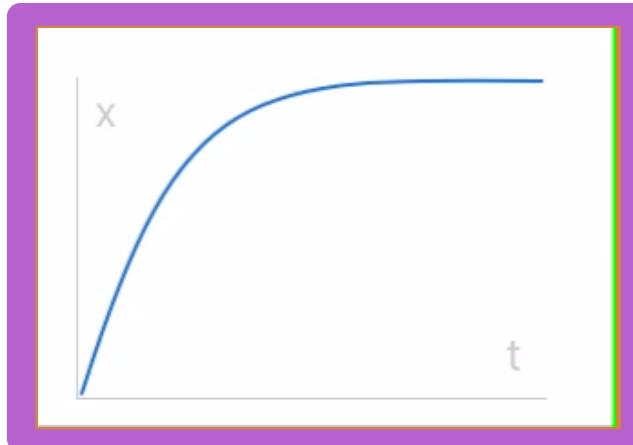
**EaseOutSine**



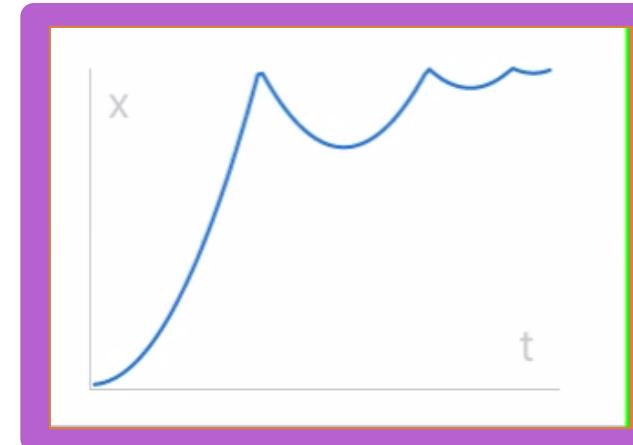
**EaseOutQuad**



**EaseOutQuint**



**EaseOutBounce**

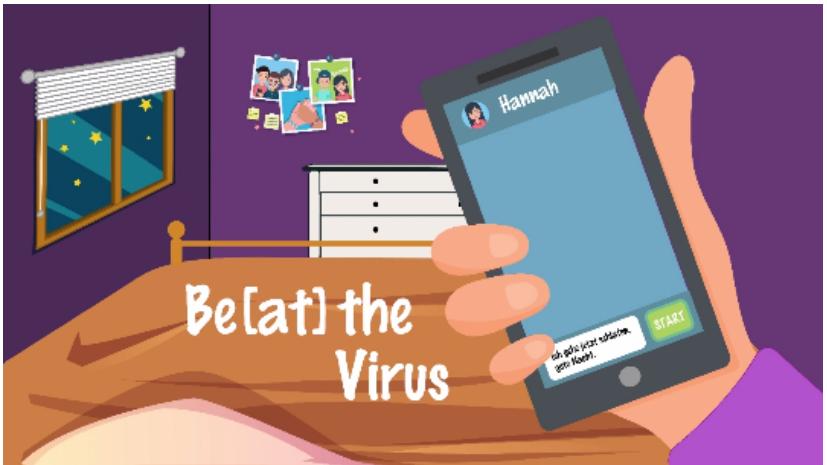


# Animationguide

## Nachrichten Startscreen

Diese Animation soll die Aufmerksamkeit der Nutzenden auf die Mitteilungen im Handy richten, die den erste Einstieg in die Rahmenhandlung geben.

- Animationstyp: EaseOutSine
- Animationsgeschwindigkeit: 0,06



## Startbutton

Diese Animation verdeutlicht, dass der Button klickbar ist.

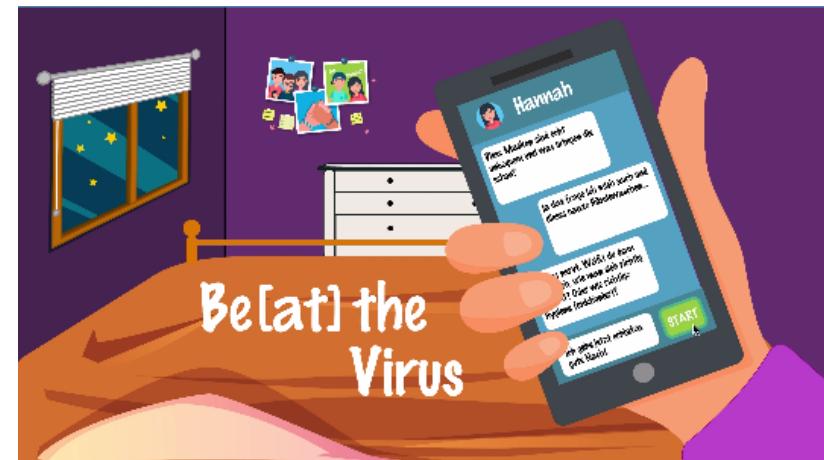
- Animationstyp: EaseOutQuint
- Animationsgeschwindigkeit: 0,06



# Animationguide

## Übergang Traum

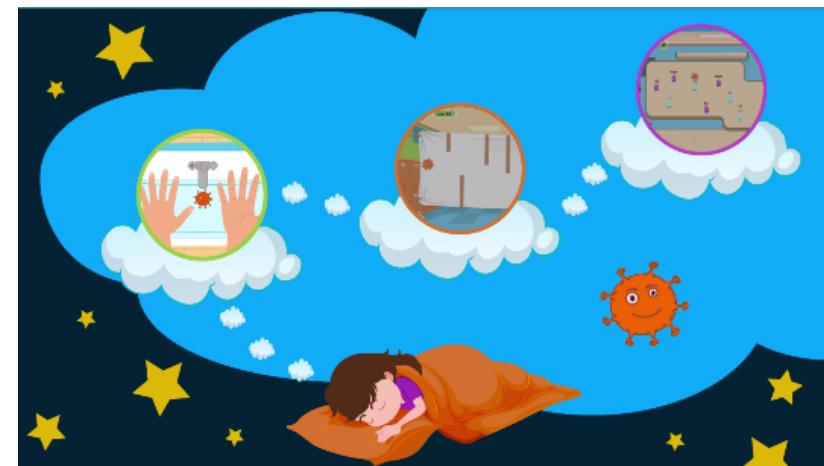
Diese Animation wurde mit Hilfe von After Effects erstellt. Der „Swirl“ Effekt verdeutlicht, dass der Charakter sich nun schlafen legt und der Nutzende sich in einem Traum befindet.



## Untersimulation Auswahl

Diese Animation verdeutlicht, dass das Bild der Untersimulation klickbar ist.

- Animationstyp: EaseOutQuint

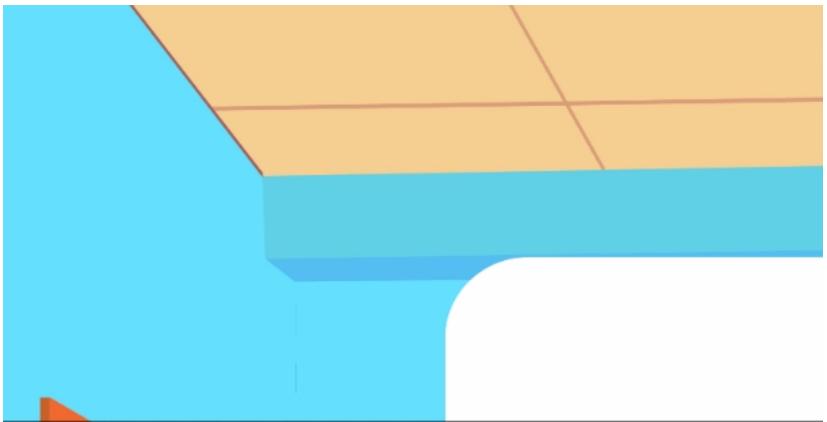


# Animationguide

## Schilder und Wecker

Die Schilder und der Wecker „fallen“ in die Spielumgebung hinein, um kurz Aufmerksamkeit auf sich zu ziehen und den Nutzenden eine Orientierung zu bieten. Der Bounce Effekt unterstreicht die optisch „an Fäden aufgehängte“ Darstellung.

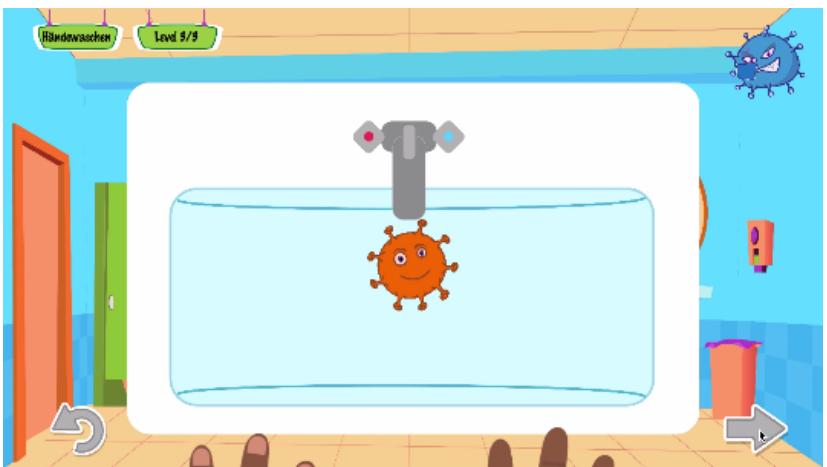
- Animationstyp: EaseOutBounce
- Animationsgeschwindigkeit: 0,04



## Hände

Diese Animation lässt die Hände, die zu infizieren sind, lebendiger wirken und verdeutlicht, nach Ablauf der Animation, dass die Simulation nun gespielt werden kann.

- Animationstyp: beschleunigte Bewegung

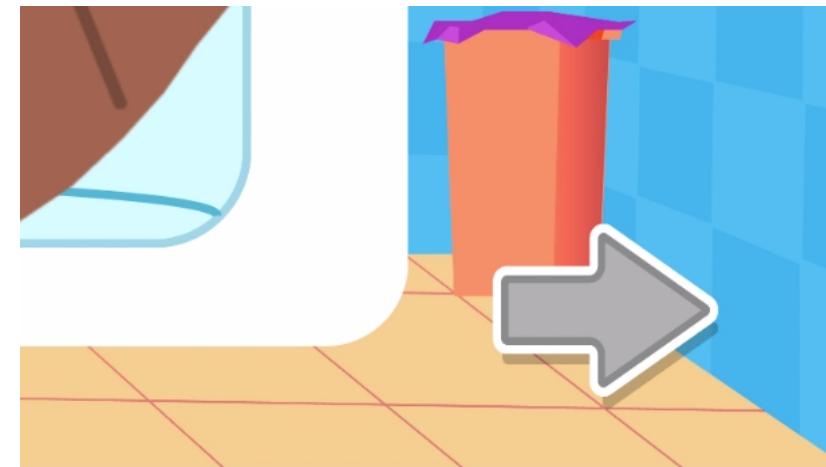


# Animationguide

## Weiter und Retry Button aktiv

Sowohl der Weiter als auch der Retry Button ändern ihre Farbe und werden größer, wenn sie aktiviert sind. Diese Animation lenkt den Fokus der Nutzenden auf die Button, damit verständlich ist, dass man sie nun klicken muss.

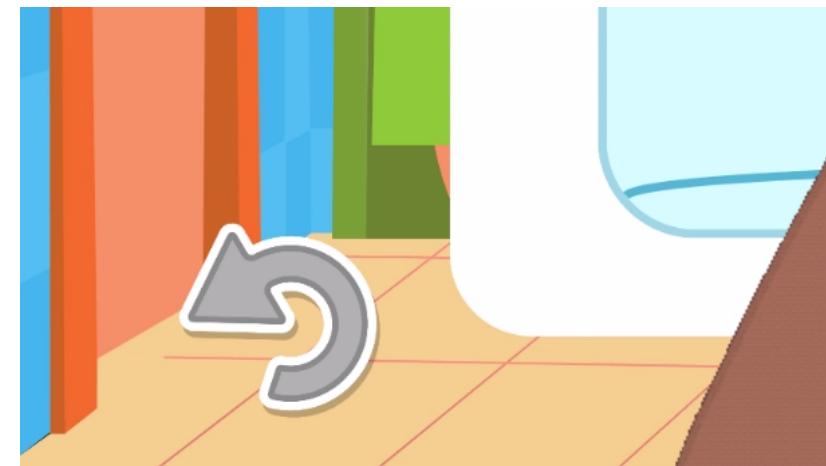
- Animationstyp: EaseOutSine
- Animationsgeschwindigkeit: 0,12



## Weiter und Retry Button hover

Diese Animation zeigt einerseits an, dass die Elemente klickbar sind. Sie verdeutlicht jedoch auch, was passieren wird. Daher dreht der Retry Button sich links herum. Der Weiter Button streckt sich nach rechts, um ebenfalls seine Funktion anzuzeigen.

- Animationstyp: EaseOutQuint
- Animationsgeschwindigkeit: 0,06



# Animationguide

## Mentorvirus Wiggle

Der Mentorvirus rotiert, um zu verdeutlichen, dass er ein klickbares Objekt ist.

- Animationstyp: EaseOutQuad
- Animationsgeschwindigkeit: 0,16



## Mentorvirus Sprechblase

Die Sprechblase des Mentorvirus fährt aus und ein, um Aufmerksamkeit auf sich zu lenken, wenn es eine neue Mitteilung gibt.

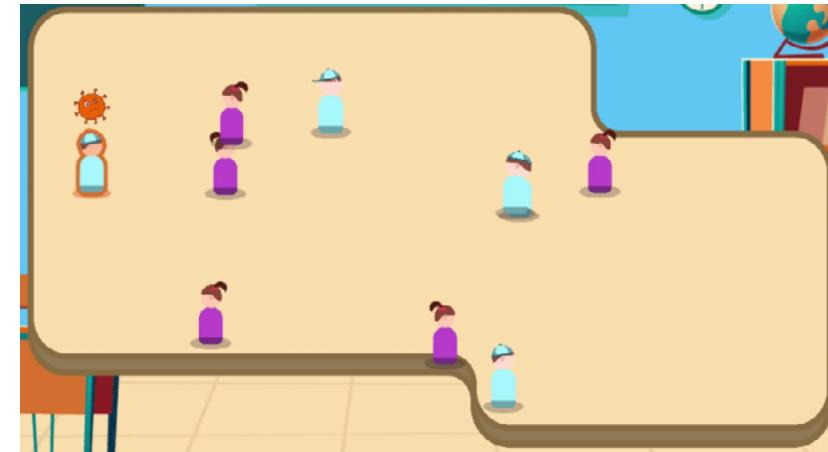
- Animationstyp: EaseOutSine
- Animationsgeschwindigkeit: 0,1

# Animationguide

## People Bewegung und fliegende Viren

Die People in People Bouncy, sowie die Viren, die sie verbreiten, bewegen sich mit einer Beschleunigung, da dies natürlicher wirkt. Dennoch konnten wir kein Easing einbauen, da sie eine unbekannte Strecke zurück legen und die Dauer der Bewegungen unterschiedlich lang sind.

- Animationstyp: beschleunigte Bewegung



## Wecker klingeln

Diese Animation ist besonders auffällig gestaltet, da sie die Aufmerksamkeit des Nutzenden auf sich ziehen soll, um ihn zu warnen, dass nur noch 10 Sekunden des Spiels verbleiben.

- Animationstyp: EaseOutSine
- Animationsgeschwindigkeit: 0,2



# Animationguide

## Zerreißen Ende

Diese Animation haben wir in XD erstellt. Sie soll die Nutzenden überraschen und schockieren, da sie anschließend ihr Handeln und ihre Hygienepraxis überdenken sollen.

- Animationstyp: Easeln
- Animationsgeschwindigkeit: 1 Sekunde



# Sound

## Soundeffekte

Wir haben unsere Soundeffekte selbst aufgenommen, da das Timing bei uns speziell war. Weiterhin entschieden wir uns allgemein für unverzerrte Soundeffekte, da sie realistischer wirken und so unseren Aufklärungscharakter unterstützen.

## Händewaschen

Gerade bei den Händewaschanimationen, sollte der Sound so gut es geht zu den gezeigten Bildern passen. Dies war uns wichtig, da der Sound so zum Einen realistischer wirkt, zum Anderen dadurch auch die Animation echter erscheint. Beim Händewaschen liegt ein besonderer Fokus darauf, dass die Animation realistisch wirkt, da diese die Anleitung zum richtigen Händewaschen darstellen soll.

## People Bouncy

In People Bouncy variieren wir die Soundeffekte „husten“ und „niesen“. Auch hat jeder von uns ein Set an Sounds aufgenommen, um noch mehr Varianz reinzubringen.

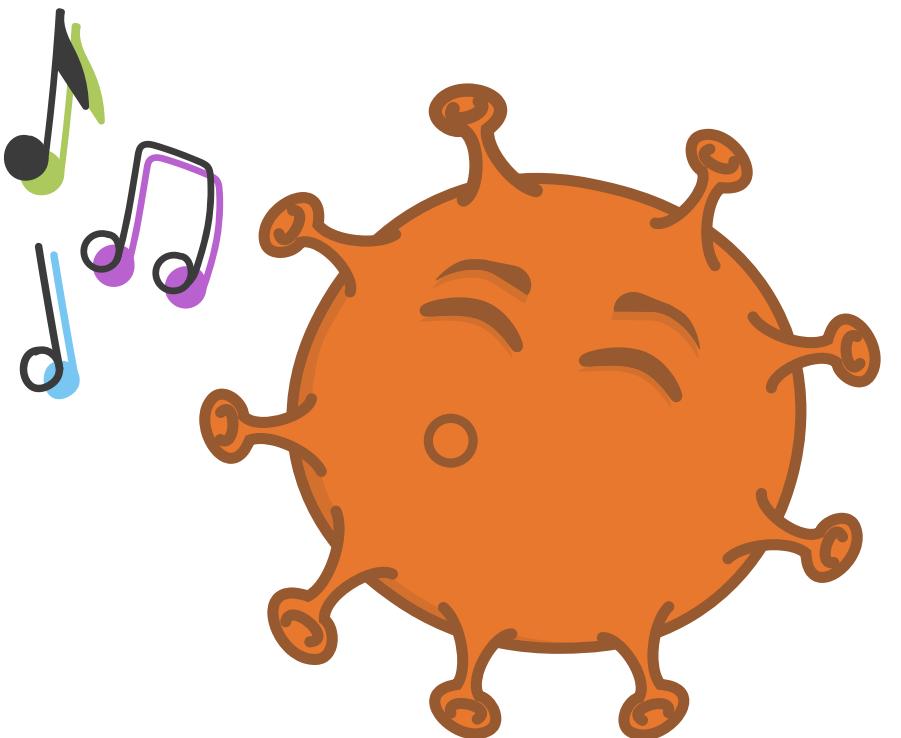
## Hintergrundgeräusche

Bei der Hintergrundgeräusche begannen wir in People Bouncy. Wir suchten Umgebungsgeräusche von einem Spielplatz und einem Klassenzimmer raus, in der Hoffnung eine Schulatmosphäre zu erzeugen. Auch bauten wir in dem 3. Level Windgeräusche ein, um die düsterere Stimmung zu unterstützen. In unserer finalen Szene steigerten wir diese Geräuschkulisse, indem wir sie um den Krach eines richtigen Gewitter ergänzten. Wir bemerkten schnell, dass uns diese Geräusche ausreichten und entschieden uns damit gegen

# Sound

eine Hintergrundmusik. Weiterhin hatten wir in der kurzen Zeit, die uns blieb keine Möglichkeit diese Geräusche selbst aufzunehmen und mussten deshalb auf Royalty-Free Aufnahmen zurückgreifen.

Leider haben wir keine passenden Hintergrundgeräusche für die Untersimulation Händewaschen gefunden. Wir haben es sowohl mit Geräuschen von Wasser in allen möglichen Formen, als auch mit Umgebungsgeräuschen aus echten Toiletten. Wir haben allerdings keine Tonaufnahme gefunden, welche auch nur ansatzweise zu unserer Darstellung passt. Zum selber Erstellen hat uns am Ende die Zeit gefehlt, darum ließen wir diese Hintergrundgeräusche in unserem Prototypen aus.



# Management

Teamorganisation

Projektbericht

Ablauf

Herausforderungen

# Organisation

## Team

Für die Phase der Umsetzung behielten wir unsere Teamregeln bei. Wir legten weiterhin viel Wert auf eine offene Kommunikation, einen respektvollen Umgang miteinander und einen offenen Informationsaustausch.  
Wie in der Konzeptphase bereits erlebt, spürten wir nun auch in der Umsetzung, die Vorteil von zuvor vereinbarten Teamregeln.

## Tools

Auch die Tools zur Strukturierung und effektiven Zusammenarbeit blieben die gleichen.

Conceptboard nutzten wir weiterhin, um Mindmaps zu erstellen, Ideen zu sammeln, oder Zwischenergebnisse festzuhalten.

Wir nutzten Trello, um unsere einzelnen Sprints zu organisieren. Den Backlog befüllten wir dabei zu Beginn mit allen Themen, die zu erledigen waren, bis zur Abgabe. Wöchentlich entschieden wir uns dann, welche Aufgaben wir in der kommenden Woche bearbeiten wollten. So behielten wir über die gesamte Zeit einen guten Überblick über unsere Aufgaben.

Um unsere gesamte Arbeitsphase protokollarisch festhalten zu können, schrieben wir für jedes Meeting einen Projektbericht, in dem wir festhielten, was wir im jeweiligen Meeting bearbeitet und besprochen haben, welche Aufgaben oder Maßnahmen daraus entstanden sind und natürlich auch, wie lange eine Aufgabe dauerte und bis wann eine Maßnahme zu erledigen ist. Auf der nächsten Seite ist beispielhaft ein Auszug aus dem Projektbericht vom 15.06.2020.

# Projektbericht

Konzept

Technik

Design

Management

# Ablauf

## Testphase

Wir sind nach der Konzeptphase mit einer linearen Planung in die Umsetzungsphase gestartet. Als erstes haben wir die Grundstruktur unserer Klassen und Objekte für alle Simulationen programmiert, sodass wir etwas haben, worauf wir aufbauen konnten. Dies haben wir anhand vieler einzelner Tests mit Test-Videos/Bilder und klickbaren Objekten ausprobiert. Als wir uns sicher waren, dass alles wie geplant eingebunden werden kann, sind wir in unseren Prototypen gestartet.

## Untersimulation Händewaschen

Hier haben wir zunächst mit der Untersimulation Händewaschen begonnen. Als das Grundgerüst dieser stand, haben wir parallel mit dem Startscreen, sowie der Karte begonnen.

## Flappy Mask oder People Bouncy?

Nachdem wir Händewaschen abgeschlossen haben, wollten wir Flappy Mask beginnen. Jedoch waren wir uns unsicher, ob wir sowohl Flappy Mask als auch People Bouncy in der restlichen Zeit schaffen. Daher haben wir uns mit unserem Coding Dozenten Garrit Schaap beraten und sind zu dem Schluss gekommen, dass wir mit People Bouncy mehr lernen können als mit Flappy Mask - selbst wenn es nicht fertig wird. Außerdem fanden wir, dass People Bouncy viel mehr den Kern unserer gesamten Simulation vermittelt, als es Flappy Mask könnte.

Hier haben wir zuerst wieder die Grundstruktur programmiert, damit wir alle Level nur noch zusammenbauen müssen.

# Ablauf

## Produktvideo

Als sich People Bouncy langsam dem Ende neigte, begonnen wir mit unserem Produktvideo. Dazu erstellen wir zunächst ein Storyboard, aus dem ein Animatic entstand. Dieser war eine Art Blaupause für unsere anschließende Animation.

## Wir haben es geschafft!

Nun sind wir am Ende der Umsetzungsphase und haben sowohl den Startscreen, Händewaschen, die Karte, People Bouncy und das Ende technisch in einem Prototypen umgesetzt.

Da wir tägliche Team Meetings geplant haben, konnten wir uns gegenseitig gut helfen und unsere Stärken gleichmäßig nutzen. Wir sind Level für Level vorgegangen, was uns viel Übersicht gegeben hat. So sind wir nie im Chaos versunken, da wir wussten, was im nächsten



Sprint ansteht. Zwar lief nicht alles immer nach Zeitplan, aber die Struktur hat uns geholfen den Überblick zu wahren.

In diesem ganzen Prozess haben wir stets zwischen den Aufgaben rotiert, damit jeder an allen möglichen Stellen seinen Horizont erweitern kann. Auch wenn nicht alles immer glatt lief, haben wir doch unglaublich viel von einander gelernt und können stolz auf unseren (wenn auch nicht ganz fertigen) Prototypen sein.

# Herausforderung

## Blockade

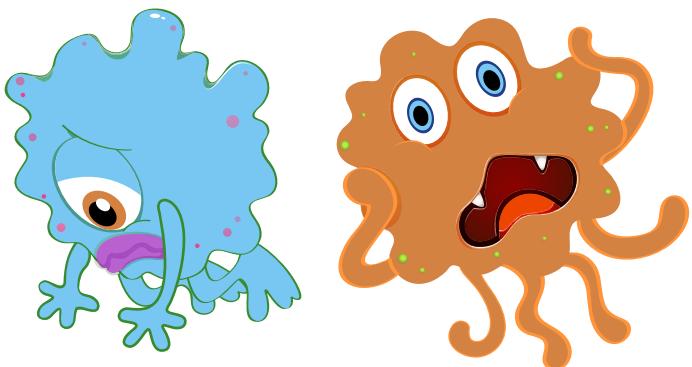
Zu Beginn der Umsetzungsphase hatten wir circa eine Woche lang Probleme uns zu organisieren. Wir fanden keinen Anfang und das gemeinsame Arbeiten in der Umsetzung fiel uns schwer.

Eine besondere Herausforderung war das interdisziplinäre Arbeiten. Aus der eigenen Komfortzone heraus zu gehen und etwas zu tun, in dem man sich noch nicht sicher fühlte.



## Lösung

Wir haben schließlich realisiert, dass wir umdenken und uns neu strukturieren müssen. Wir entschieden uns dazu in zweier Teams zu arbeiten. Beide Teams arbeiteten parallel in Discord, so waren alle immer erreichbar. Dieses Vorgehen half uns interdisziplinär zu arbeiten. Wir trafen uns jeden Tag und legten morgens die Aufgaben der Teams fest. Die täglichen Sprints gewährleisteten einen regen Austausch untereinander.



# Schluss

# Ausblick

blablablabla

# DOR & NVS

## Delegation of Rights and Non Violation Statement

Be[at] the Virus is a project, developed in the framework of the study course Interactive Media Design of Hochschule Darmstadt - Faculty of Media, during the Summer Term 2020. It is a group work of: Diandra Hermann, Florian Grünewald, Marie Sasse and Leander Schmidt. It was mentored by: Prof. Andrea Krajewski, Garrit Schaap, Dieter Stasch. We herewith delegate the non exclusive and timewise non restricted rights to publish and present the results of the project Be[at] the Virus to the Professors of Hochschule Darmstadt and to the coaches directly connected to the academic supervision of this project, named above. In the same time the student project team declares that with the project no intellectual properties rights of third parties have been harmed.

Dieburg, 08. Juli 2020

---

Diandra Hermann

---

Florian Grünewald

---

Marie Sasse

---

Leander Schmidt

# Quellen

## **Startscreen S.1**

Bettdecke „Designed by upklyak / Freepik“  
Handy „Designed by Freepik“  
Poster „Designed by Freepik“

## **Conceptboard Logo S.11**

[http://ww1.prweb.com/prfiles/2013/08/21/11049174/  
cb\\_logo\\_300ppi.png](http://ww1.prweb.com/prfiles/2013/08/21/11049174/cb_logo_300ppi.png)

## **Jitsi Logo S.11**

[https://fitsmallbusiness.com/wp-content/uploads/2019/01/  
Jitsi-logo.png](https://fitsmallbusiness.com/wp-content/uploads/2019/01/Jitsi-logo.png)

## **BigBlueButton Logo S.11**

[https://www.moonami.com/webinar/webinar-big-blue  
-button-virtual-classroom/](https://www.moonami.com/webinar/webinar-big-blue-button-virtual-classroom/)

## **Trello Logo S.11**

[https://de.m.wikipedia.org/wiki/Datei:Trello-logo-blue.  
svg](https://de.m.wikipedia.org/wiki/Datei:Trello-logo-blue.svg)