Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Medien
Faculty of Media

„What the Hack?“

Konzeption und Implementierung eines erweiterbaren und adaptiven Serious Game zur Verbesserung von Information Security Awareness

Masterarbeit

im Studiengang Medieninformatik
zur Erlangung des akademischen Grades
Master of Science

Fachbereich Medien
Hochschule Düsseldorf

Julian Geywitz
[redacted]

Datum: April 2019

Erst- und Zweitprüfer

Prof. Dr.-Ing. Holger Schmidt
Prof. Dr.-Ing. M.Sc. Markus Dahm

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Masterarbeit selbständig und ohne unzulässige fremde Hilfe angefertigt habe. Die verwendeten Quellen sind vollständig zitiert. Diese Arbeit wurde weder in gleicher noch ähnlicher Form einem anderen Prüfungsamt vorgelegt oder veröffentlicht. Ich erkläre mich ausdrücklich damit einverstanden, dass diese Arbeit mittels eines Dienstes zur Erkennung von Plagiaten überprüft wird.

Ort, Datum *Julian Geywitz*

Kontaktinformationen

[redacted]

Zusammenfassung

„What the Hack?“

Konzeption und Implementierung eines erweiterbaren und adaptiven Serious Game zur Verbesserung von Information Security Awareness

Julian Geywitz

Diese Masterarbeit beschreibt den Entwurf und die Neuimplementierung von *What the Hack*, einem digitalen Serious Game zur Verbesserung von Information Security Awareness. *What the Hack* ist ursprünglich in einem Studentenprojekt entstanden und soll in der Unity Game Engine neu implementiert, erweitert und verbessert werden.

Nach einer Untersuchung gängiger Literatur auf Merkmale eines guten Serious Games wird der aktuelle Markt von Serious Games zum Thema Information Security Awareness untersucht. Es zeigt sich, dass es dem Markt an Spielen für eine fortgeschrittene, technische Zielgruppe mangelt. Ebenfalls ersichtlich wird, dass sich wenige Spiele spezialisieren oder anpassen lassen. Es folgt eine Bestandsaufnahme der originalen Implementierung. Mit den Ergebnissen der theoretischen Untersuchungen wird die originale Implementierung von *What the Hack* auf Kriterien eines guten Serious Games untersucht. Anschließend werden Anforderungen für die Neuimplementierung von *What the Hack* gestellt. Es folgt eine Beschreibung des Game Designs und der Implementierung wichtiger Eigenschaften des Spiels, wie die Unterstützung von Schwierigkeitsgraden und der Erweiterbarkeit.

Abschließend folgt eine Analyse der Neuimplementierung. Es zeigt sich, dass das Spiel die gestellten Anforderungen erfüllt und *What the Hack* sich zu einer, auf eine technische Zielgruppe fokussierte, Serious Game Plattform entwickelt hat.

Abstract

„What the Hack?“

Concept and implementation of an extensible and adaptive serious game to improve information security awareness

Julian Geywitz

This thesis describes the design and re-implementation of *What the Hack*, a digital serious game which teaches information security awareness. *What the Hack* has its roots in a student project. In this thesis the game is re-implemented, extended and improved utilizing the Unity Game Engine.

A research about characteristics of a good serious game is followed by an investigation of available serious games that teach information security awareness. The findings indicate that only a few games target advanced and technical people. Also, only a few of them can be specialized or adapted. Thereafter, the original implementation is examined and documented. Using the results of the theoretical investigation, requirements for the re-implementation of *What the Hack* are defined. Then a description of the game design and important game properties follows. This includes the support for different difficulty levels and the support of Plugins.

Finally, the re-implementation is investigated. It becomes apparent that the game fulfills the given requirements.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
BSI	Bundesamt für Sicherheit in der Informationstechnik
GPL	GNU General Public License
GUID	Globally Unique Identifier
JSON	JavaScript Object Notation
Mod	Modifizierung
SDK	Software Development Kit
UI	User Interface

Kapitel 1

Einleitung

1.1 Kontext

Informationssicherheit ist ein Thema wichtiger denn je. In Zeiten, in denen mehr und mehr Daten zu jedem von uns gesammelt werden, wird es immer wichtiger, diese vor falschen Händen zu schützen[1]. Häufig wird bei Informationssicherheit hauptsächlich an den technischen Schutz von Daten gedacht, wie z. B. die Verschlüsselung zur Verhinderung von unerwünschtem Mitlesen. Allerdings sind technische Maßnahmen allein in vielen Fällen nur bedingt wirkungsvoll (Hagen, Albrechtsen & Hovden, 2008). Eine weit verbreitete technische Schutzmaßnahme ist z. B. das Passwort. Es wird in vielen Bereichen zur Authentifizierung von Menschen wie etwa bei Betriebssystem- oder anderen Accounts genutzt (International Organization for Standardization, 2017) und findet auch bei der Verschlüsselung bspw. bei Key Wrapping oder Password-based Encryption (van Tilborg & Jajodia, 2011) seine Verwendung. Allerdings ist das Passwort auch anfällig für menschliche Fehler. Schlecht gewählte Passwörter, im Sinne von zu kurz oder leicht zu erraten, oder leichtsinnige Fehler, etwa wenn ein Passwort auf ein Blatt Papier geschrieben wird, sind keine Seltenheit.

Dies ist nur eines von vielen Beispielen, in denen der Mensch das schwächste Glied der Kette ist. Weiterhin existieren zahlreiche andere Angriffe, um gewünschte Informationen von Menschen zu bekommen. Dies ist oft einfacher als ein technischer Angriff und wird *Social Engineering* genannt. Bursztein zeigte auf der *Black Hat USA* 2016, wie effektiv ein Angriff mittels willkürlich verteilten USB Sticks, sogenannten „Honeypots“, sein kann. Er verteilte knapp 300 präparierte USB-Sticks an verschiedenen Stellen auf dem Campus der Universität Illinois. 98% der USB-Sticks wurden mitgenommen und 45% haben sich bei einem Server zurückgemeldet, als Dateien auf dem Stick geöffnet wurden (Bursztein, 2016). Bei einem echten Angriff können Hardware und Dateien so präpariert werden, dass Tastendrücke aufgenommen werden (*Keylogging*) oder Daten entwendet werden können (van Tilborg & Jajodia, 2011). Um sich vor solchen Angriffen und leichtsinnigen Fehlern zu schützen muss der Nutzer auf mögliche Angriffe und die Relevanz des Themas aufmerksam gemacht werden. Das Ziel ist es *Security Awareness* beim Nutzer zu schaffen (Helisch & Pokoyski, 2009, Kapitel 2.1). Besitzt ein Nutzer Security Awa-

[1] https://resource.alibabacloud.com/whitepaper/data-security-is-now-more-important-than-ever_219, aufgerufen am 27.03.2019.

reness, ist er sich der Gefahren bewusst und handelt dementsprechend, um sie zu vermeiden (Wolek, 2008).

Um Security Awareness zu schaffen, bieten Unternehmen oft Schulungen oder Präsentationen zum Thema an, die selbstständig z. B. einmal im Jahr bearbeitet werden müssen. Eine weitere Möglichkeit, dies zu erreichen, ist *Serious Gaming*. Hierbei werden die Lehrinhalte spielerisch vermittelt (Strahringer & Leyh, 2017). Im Bereich der Security Awareness existieren bereits einige Serious Games wie *The Weakest Link*[2] der Firma *IS Decisions* oder das Kartenspiel *Control-Alt-Hack*[3].

1.2 Motivation

Ob Unterhaltungselektronik, Kassensysteme, Buchhaltung oder kritische Infrastruktur – ein Ausfall oder eine Kompromittierung dieser Systeme kann weitreichende Folgen für Einzelne, Unternehmen oder die Gesellschaft haben. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) erstellt daher jedes Jahr in Kooperation mit der *Allianz für Cyber-Sicherheit* die *Cyber-Sicherheits-Umfrage*. Im aktuellen Bericht von 2017 zeigt sich, dass 2016 und 2017 ca. 70% der befragten deutschen Institutionen Opfer eines Cyberangriffs geworden sind. Bei knapp über der Hälfte aller Betroffenen hatte ein Angriff einen Produktions- oder Betriebsausfall zur Folge. Technische Schutzmaßnahmen sind mittlerweile bei den meisten Betrieben im Einsatz. So betreiben ca. 89% aller befragten Unternehmen eine Netzwerkabsicherung. Die Prävention durch Schulung der Mitarbeiter ist allerdings erst bei 52% der befragten Unternehmen im Einsatz, bei 19% in Planung und bei 29% weder geplant noch im Einsatz (siehe Abb. 1.1) (BSI, 2017).

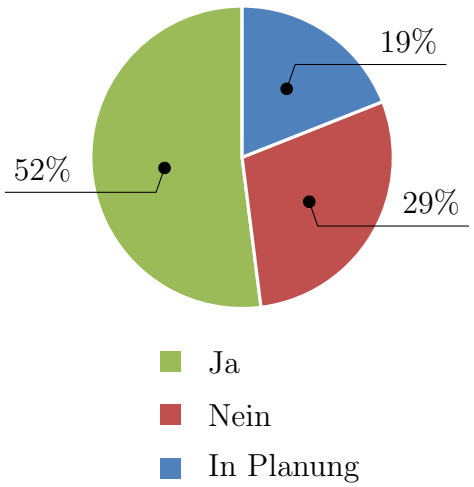


Abbildung 1.1: Wird das Personal in regelmäßigen Abständen in Bezug auf IT-Sicherheit geschult? (BSI, 2017)

Proofpoint, ein Unternehmen, welches laut eigener Angabe Cybersicherheitslösungen an über die Hälfte der *Fortune 100* Unternehmen ausliefert, verfasst

[2]https://www.isdecisions.com/user-security-awareness-game/, aufgerufen am 18.02.2019.
[3]http://www.controlalthack.com/, aufgerufen am 18.02.2019.

jährlich einen Bericht speziell zum Faktor Mensch in der IT-Sicherheit auf Basis der Angaben der eigenen Kunden. Laut diesem Bericht ist der Faktor Mensch aktuell die größte Ursache für gelungene Cyberangriffe (Proofpoint, 2018). Daher ist es für Unternehmen besonders wichtig, in diesen Bereich zu investieren und Sicherheitsbewusstsein zu schaffen.

1.3 Wissenschaftlicher Beitrag

In Abschnitt 2.4 wird eine Analyse des Markts zu Serious Games zum Thema Security Awareness durchgeführt. Dort zeigt sich, dass die meisten Spiele für die Allgemeinheit sowie für Anfänger ausgelegt sind. Um umfassend Security Awareness zu schaffen, müssen allerdings alle Menschen in einer Organisation geschult werden. Die verschiedene Wissensstände der Mitarbeiter, die unterschiedlichen Umgebungen in denen sie arbeiten und die Kultur sowie Anforderungen des eigenen Unternehmens müssen beachtet werden. Zur Schulung dieser Mitarbeiter werden Serious Games benötigt, die für das Unternehmen, den Arbeitsplatz und die Umgebung spezialisiert werden können. Um Mitarbeiter mit jedem Wissensstand schulen zu können, müssen die Serious Games adaptiv sein und mehrere Schwierigkeitsgrade unterstützen. Dem Markt mangelt es an solch spezialisierten, adaptiven Serious Games, die für technische oder wirtschaftliche Zielgruppen ausgerichtet sind. Der Beitrag dieser Arbeit ist ein solches adaptives Serious Game zur Verbesserung der Security Awareness. Das Spiel ist mit einem theoretischem Hintergrund zur Serious Gaming Theorie erstellt. Es unterstützt mehrere Schwierigkeitsstufen, ist adaptiv und erweiterbar. Ein Application Programming Interface (API) sowie ein Software Development Kit (SDK) erleichtern die Erstellung von Erweiterungen. Weiterhin besitzt das Spiel zwei Spielmodi, Echtzeit und klassisches Spiel, welche für zukünftige Evaluationen zur Effektivität von Serious Games interessant sind. Das Spiel wird unter einer Open Source Lizenz veröffentlicht.

1.4 Ziele

Das Ziel dieser Arbeit ist es, das in einem Studentenprojekt entwickelte Android-Spiel mit dem Titel *What the Hack* auf umgesetzte Serious-Gaming-Aspekte zu untersuchen und neu zu implementieren. Das Spiel ist eine Echtzeit-Wirtschaftssimulation, bei der der Spieler eine Firma, die Hacking-Aufträge ausführt, leitet. Er kann Arbeiter einstellen, Aufträge abarbeiten lassen und lernt dabei Verschiedenes über mögliche Angriffe auf Computersysteme. Das Spiel wurde mit Java und der Grafikbibliothek `libgdx`[4] entwickelt. Die Neuimplementierung soll über ein Plugin-System zur Erweiterung des Spiels verfügen und mit der Spiele-Engine `Unity`[5] entwickelt werden. Die Adaptivität des Spiels wird durch die Unterstützung mehrerer Schwierigkeitsgrade gegeben. Eine Auflistung der Ziele folgt in Tabelle 1.1:

[4] https://libgdx.badlogicgames.com, aufgerufen am 11.03.2019.
[5] https://unity3d.com, aufgerufen am 11.03.2019.

Ziel Nr.	Beschreibung	Kapitel
Z1	Herausarbeiten des aktuellen Stands der Technik von Serious Games im Umfeld von Security Awareness. Dabei soll auch untersucht werden an welche Zielgruppen sich die Spiele richten und welche Vor- und Nachteile Serious Games besitzen.	2.2.3, 2.4
Z2	Die Erstellung einer Liste von Kriterien, die ein Serious Game erfüllen sollte. Das ursprüngliche Spiel *What the Hack* wird anschließend auf diese Kriterien untersucht. Aus den Ergebnissen werden Anforderungen an die Neuimplementierung formuliert.	2.2.2, 2.6
Z3	Der Quellcode der Neuimplementierung soll unter einer geeigneten Open Source Lizenz veröffentlicht werden, welche ausgewählt und angewendet werden soll.	4.2
Z4	Auch wenn sich Serious Gaming zur Erzeugung von Security Awareness eignet, ist nach einmaligem Spielen von einem geringem Erfolg auszugehen. Es kann allerdings von einem kurzfristigem positiven Einfluss ausgegangen werden (Hendrix, Al-Sherbaz & Bloom, 2016, S. 7). Ziel ist es, das Verhalten des Menschen in sicherheitskritischen Umgebungen zu ändern. Erfolgversprechender für langfristige Security Awareness könnte ein Spiel sein, welches zum regelmäßigen Spielen motiviert. Dadurch könnte eine größere Präsenz des Themas bei dem Spieler geschaffen werden. Diese These soll weiter recherchiert werden. Trifft die These zu, soll eine neue Spielmechanik entwickelt werden, bei der das Spiel an die reale Uhrzeit gebunden wird. Das ermöglicht ein Spielprinzip, wie es z. B. beim mobilen Spiel *Clash of Clans*[6] umgesetzt ist. Der Spieler spielt hier meistens öfter am Tag für kurze Zeit, da die Aktionen im Spiel oft mehrere Echtzeit-Stunden benötigen, bis sie abgeschlossen sind.	3.3

[6]https://supercell.com/en/games/clashofclans/, aufgerufen am 18.02.2019

Ziel Nr.	Beschreibung	Kapitel
Z5	Das Spiel soll großteils dem Original entsprechen, allerdings soll es mit spielerischen und inhaltlichen Plugins erweiterbar sein. Diese Plugins sollen das Spiel auf eine bestimmte Ziel- oder Berufsgruppe maßschneidern, indem sie etwa neue Spielmissionen mit passendem Lehrinhalt einführen. Zu Beginn soll das Maß der Erweiterbarkeit festgelegt sowie ein Mechanismus zum Laden von Plugins auf allen Plattformen gefunden werden. Die Entwicklung der Plugins soll in Unity erfolgen und durch verschiedene grafische Werkzeuge, eine API und Dokumentation ermöglicht werden.	3.2, 4.4
Z6	Das Spiel soll von Grund auf mit der Unity Game Engine neu implementiert werden. Dabei werden ein Großteil der bereits vorhandenen Assets, wie Grafiken und Sounds, weiterverwendet, während die Software-Architektur komplett neu gestaltet und dokumentiert wird. Auch am Spielprinzip oder an der Bedienung sind Änderungen möglich. Die Zielplattformen sind Desktop (Linux, macOS und Windows) im Falle der originalen Spielmechanik sowie mobile Geräte, insbesondere Android.	4
Z7	Die Unterstützung mehrerer Schwierigkeitsgrade soll implementiert werden. Dadurch wird die benötigte Adaptivität für verschiedene Wissensstände erreicht. Dies kann sich einerseits auf den Lehrinhalt sowie auch auf die Schwierigkeit der Spielmechanik auswirken.	3.4
Z8	Zur Dokumentation sowie zum finalen Test soll ein Beispiel-Plugin mit Lehrinhalt für eine bestimmte technische Zielgruppe entwickelt werden, welche im Verlauf der Arbeit festgelegt wird. Das Spiel eignet sich hierfür besonders, da es von dem Humor dieser Branche lebt.	4.9
Z9	Abschließend soll die Neuimplementierung auf die in Z2 festgelegten Kriterien und Anforderungen analysiert werden. Optional ist auch eine Evaluation möglich.	5

Tabelle 1.1: Ziele dieser Arbeit

1.5 Vorgehen

Ziel Nr.	Vorgehen
Z1	Recherche, siehe Abschnitt 2.2.1.
Z2	Die Kriterien werden aus grundlegender Literatur herausgearbeitet.
Z3	Zuerst werden Kriterien für die Lizenz aufgestellt. Aufgrund dieser werden verschiedene, populäre Open Source Lizenzen gegeneinander abgewägt.
Z4	Recherche zum Thema, siehe Abschnitt 3.3.1.
Z5	Das Maß der Erweiterbarkeit wird durch eine Abwägung zwischen Zeit und Relevanz zur Adaptivität des Spiels festgelegt.
Z6	Zu Beginn wird eine Architektur der Software erstellt, welche auf Erweiterbarkeit ausgelegt ist. Anschließend wird die Architektur implementiert und Schritt für Schritt werden Features zum Spiel hinzugefügt. Begonnen wird dabei mit der Erweiterbarkeit, dem Kern des Spiels.
Z7	Implementierung der Schwierigkeitsstufen.
Z8	Zuerst wird eine Recherche zu einem speziellen Thema aus der IT-Sicherheit, das den Lehrinhalt des Spiels darstellt, durchgeführt. Danach folgt die Implementierung des Beispiel-Plugins. Dabei wird auf eine gute Dokumentation geachtet.
Z9	Überprüfung des fertigen Spiels auf die gestellten Anforderungen.

Tabelle 1.2: Vorgehen bei den einzelnen Zielen

1.6 Aufbau

Zu Beginn dieser Arbeit wird der aktuelle Stand der Technik von Serious Games zum Thema Security Awareness herausgearbeitet (siehe Kapitel 2.1.2). Dies beinhaltet eine kurze Erklärung der Begrifflichkeiten, gefolgt von einer Recherche zu Merkmalen eines lehrreichen Serious Games. Anschließend werden erhältliche Serious Games zum Thema Security Awareness auf die Kriterien aus Abschnitt 2.4 untersucht. Es folgt eine Bestandsaufnahme der originalen Implementierung von *What the Hack*, welches auf die zuvor festgelegten Kriterien hin untersucht wird (siehe Kapitel 2.5). Aus den Ergebnissen der Marktanalyse und der Analyse der originalen Implementierung werden Anforderungen an die Neuimplementierung gestellt (siehe Kapitel 2.6).

In Kapitel 3 wird das Game Design behandelt. Anschließend wird in Kapitel 4 die Architektur des Spiels behandelt. Es folgt eine Analyse der Neuimplementierung auf die in 2.1.2 erstellten Kriterien und Anforderungen sowie die Ziele der Arbeit (siehe Abschnitt 5). Abschließend ist in Kapitel 6 ein Fazit sowie ein Ausblick formuliert.

Kapitel 2

Stand der Technik

Dieses Kapitel untersucht den aktuellen Stand der Technik von Serious Games zum Thema Security Awareness. Nach einer Erklärung von Begrifflichkeiten folgt in Abschnitt 2.2 die Ausarbeitung von Merkmalen eines guten Serious Games. Daraufhin wird die Eignung und Effektivität von Serious Games zu diesem Thema in Abschnitt 2.3 untersucht. Es folgt eine Marktanalyse vorhandener Spiele und ihrer Zielgruppen in Abschnitt 2.4. Abschließend wird das originale Spiel *What the Hack* auf seine Tauglichkeit als Serious Game untersucht (siehe Abschnitt 2.6.1).

2.1 Begrifflichkeiten

2.1.1 Informationssicherheit und der Faktor Mensch

Informationssicherheit hat nicht zwingend etwas mit Computersystemen zu tun. Viel mehr geht es um die allgemeine Sicherheit von Informationen und ihren Schutz – ob analog oder digital. Ein System ist *informationssicher*, wenn es die Schutzziele der *Vertraulichkeit*, *Verfügbarkeit* und *Integrität* erfüllt (Eckert, 2009, Seite 5). Diese Ziele können mit den drei „Ps“ der Sicherheit nach Helisch und Pokoyski erreicht werden (Helisch & Pokoyski, 2009, Kapitel 2.2):

> **Products**: Es wird Technologie wie z. B. Verschlüsselung oder eine Firewall eingesetzt und es exisitert eine Sicherheitsinfrastruktur.
>
> **People**: Die Mitarbeiter verhalten sich sicherheitskonform.
>
> **Processes**: Es existieren Sicherheitsrichtlinien und Prozesse sind auf Sicherheit ausgerichtet.

Die drei „Ps“ bilden die **Sicherheitskette**. Um die Sicherheit zu gewährleisten, darf kein Glied der Kette versagen (Helisch & Pokoyski, 2009, Kapitel 2.2). Produkte und Prozesse können mit gut messbaren Faktoren auf Sicherheit ausgerichtet werden. So kann beispielsweise eine Firewall eingerichtet werden, die unberechtigte Zugriffe auf ein Netzwerk messbar verhindert. Bei dem zweitem Glied, dem Faktor Mensch, ist dies nicht so leicht möglich, da das Verhalten von Menschen nicht immer vorhersagbar ist und auf Faktoren wie etwa Emotionen basiert.

Der Faktor Mensch gilt schon seit geraumer Zeit als „schwächstes Glied“ der Sicherheitskette (Parsons, McCormac & Butavicius, 2010, Seite 1). Stanton, Stam,

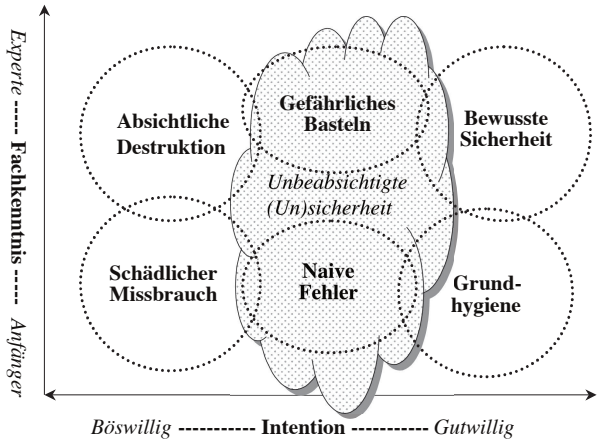


Abbildung 2.1: Klassifizierung von Fehlern beim Faktor Mensch (Stanton, Stam, Mastrangelo & Jolton, 2005)

Mastrangelo und Jolton haben 2005 eine Klassifikation für den Faktor Mensch und seine Verhaltensweisen in der Informationssicherheit erstellt. Die Klassifikation ordnet die Verhaltensweisen auf zwei Skalen ein: Fachkenntnis und Intention (siehe Abb. 2.1).

Die absichtliche Destruktion und der schädliche Missbrauch sind beides Verhaltensweisen, die mit einer böswilligen Intention betrieben werden und daher für das Thema Security Awareness uninteressant sind. Das am häufigsten auftretende Fehlverhalten sind naive oder versehentliche Fehler (Parsons et al., 2010, Seite 2). Beispiele hierfür sind fehlendes Wissen über Sicherheitsmechanismen und die daher rührende Fehlbedienung, bspw. die Auswahl von schlechten Passwörtern oder das Anklicken eines Phishing-Links. Diese Fehler können durch eine Sensibilisierung der Nutzer und durch das Schaffen von Security Awareness (siehe folgenden Abschnitt 2.1.2) reduziert werden.

2.1.2 Security Awareness

Security Awareness, oder auch Sicherheitsbewusstsein, ist ein kontextabhängiger Begriff. Im Kontext einer Privatperson bezieht er sich meist auf den sicheren Umgang im Internet, während er im Unternehmenskontext auch Themen wie Risikomanagement oder Informationssicherheit mit einbezieht. Allgemein betrachtet handelt es sich bei Security Awareness um das Wissen über mögliche Angriffstechniken und die Einstellung einer Person zu diesem Thema. Helisch und Pokoyski beschreiben Security Awareness als Zusammenspiel aus drei Faktoren (Helisch & Pokoyski, 2009, Kapitel 2.1.2):

> **Können**: Ein sicherheitskonformes Verhalten ist in dem gegebenen Umfeld möglich.
>
> **Wollen**: Der Wille sich sicherheitskonform zu Verhalten ist vorhanden.
>
> **Wissen**: Ein Sicherheitsproblem kann von der Person erkannt und verstanden werden. Das Wissen für ein dementsprechendes Handeln ist vorhanden.

Um Security Awareness zu schaffen nutzen Unternehmen verschiedene Medien. Dazu gehören z. B. Poster, E-Mails, Präsentationen, Meetings oder Serious Games.

Der Erfolg dieser Bemühungen reicht von einem sehr geringen Einfluss bis zu einer erfolgreichen Sensibilisierung der Mitarbeiter. Aspekte, welche von vielen als essentiell für einen Erfolg gesehen werden, sind die Einbindung der Unternehmenskultur in den Lernprozess. Dabei kann auch eine so genannte Sicherheitskultur geschaffen werden (Helisch & Pokoyski, 2009, Kapitel 2.4.2). Da Angriffe stetig im Wandel sind, ist das Schaffen Security Awareness ein andauernder Prozess.

2.1.3 Lernen

Bevor wir uns den Serious Games widmen, ist es hilfreich, einen Blick auf das Lernen zu werfen. Es existieren eine Vielzahl an Theorien und Definitionen zu diesem psychologischen Vorgang. Grundlegend lässt sich das Lernen in zwei Bereiche aufteilen: Das Aneignen von Wissen, der sogenannte **Wissenserwerb**, mit den Teilprozessen Verstehen, Speichern und Abrufen (Krapp & Steiner, 2001, Seite 164) sowie der Aufbau von Verhaltensweisen und die **Verhaltensänderung**. Gerhard Steiner hat Lernen unter diesen zwei Gesichtspunkten wie folgt definiert:

> „Lernen im Sinne von **Wissenserwerb** kann als der Aufbau und die fortlaufende Modifikation von Wissensrepräsentation definiert werden.“ (Krapp und Steiner, 2001, Seite 164)

> „Lernen ist als derjenige Prozess zu verstehen, der ein Individuum – aufgrund eigener, meist wiederholter Aktivität – zu relativ überdauernden **Verhaltensänderungen** führt.“ (Krapp und Steiner, 2001, Seite 140)

In vielen Fällen sind die Bereiche allerdings nicht klar zu differenzieren, da Verhaltensweisen auf zuvor gelerntem Wissen basieren können. Als Beispiel für solch einen Fall nennt Steiner das Lernen des Autofahrens. Ohne Wissen über die Funktion der Maschine können sich auch die Verhaltensweisen nicht aufbauen (Krapp & Steiner, 2001, Seite 139). Auch zum Schaffen von Security Awareness sind beide Bereiche notwendig. Um Bedrohungen zu erkennen, muss zuerst das Wissen über mögliche Bedrohungen und wie sie zu erkennen sind vorhanden sein. Das Ziel jedoch ist es, das Verhalten einer Person zu ändern, damit sie sich im Alltag des Themas bewusst ist und automatisch dementsprechend handelt.

2.1.4 Serious Games

Der Begriff **Serious Game** wurde zum ersten Mal 1970 von Clark C. Abt in seinem gleichnamigen Buch definiert:

> „We are concerned with serious games in the sense that these games have an explicit and carefully thought-out educational purpose and are not intended to be played primarily for amusement.“ (Abt, 1987, Seite 9)

Diese Definition stimmt mit dem heutigen Gebrauch des Begriffs immer noch überein. Das Hauptziel eines Serious Games ist es, etwas zu lernen. Wichtig ist, dass sich das Lernen nicht nur auf Wissen beschränkt. Ein Serious Game kann informativ sein, Reflexe, Koordination oder Verhalten trainieren (Ritterfeld, Cody & Vorderer, 2009, Seite 3). Eine andere Definition aus 2005 fasst dies gut zusammen:

> „A serious game is a game in which education (in its various forms) is the primary goal, rather than entertainment.“ (Michael und Chen, 2005, Seite 17)

Das bedeutet aber keinesfalls, dass Serious Games langweilig sein sollen. Im Gegenteil, spielerische Elemente wie Humor, Spannung, Herausforderung oder Glück sind wichtige Elemente eines Serious Games. Ein Spiel zu spielen, dass Spaß macht, ist eine *intrinsisch* motivierte Aktivität. Das bedeutet, dass die Motivation nicht von außen kommt, sondern bereits ohne externe Einflüsse vorhanden ist (Bryant & Vorderer, 2006). Zwei Eigenschaften dieser sind besonders interessant für das Lernen: Solche Aktivitäten werden oft bewusst ausdauernder ausgeführt und sie besitzen eine höhere Wahrscheinlichkeit der Wiederholung (Oerter, 1997).

Eine weitere Stärke von Serious Games ist die Kombination von Theorie und Praxis. Spiele bieten eine sichere (und teilweise sehr komplexe) Umgebung, in der Lösungen zu einem Problem entworfen und ausprobiert sowie Entscheidungen ohne Einfluss auf die reale Welt getroffen werden können. Die Spieler erhalten schnelles Feedback und können die Konsequenzen ihrer Entscheidungen erforschen (Abt, 1987, Seite 13).

Diese Eigenschaften machen Serious Games zu einem nützlichen Werkzeug in verschiedenen Branchen und für verschiedene Zielgruppen. Eines der ersten digitalen Spiele, das als Serious Game bezeichnet werden kann, ist das Spiel *Air Defense Simulation* aus dem Jahr 1948, welches an der Johns Hopkins University entwickelt wurde (Smith, 2010). Sie finden auch in vielen anderen Bereichen wie dem Gesundheitswesen, der Industrie oder der Bildung ihren Einsatz.

2.2 Serious Gaming Theorie

Dieser Abschnitt gibt einen Überblick über die Theorie hinter Serious Games. Die Recherche erfüllt das Ziel Z2.

2.2.1 Vorgehen bei der Recherche

Die Recherche hat mit durch den Betreuer gestellte Papers und einem Buch begonnen. Durch eine Stichwortsuche wurden auf den Suchmaschinen Google, Google Scholar und Yahoo sowie auf den Plattformen Researchgate und Springer weitere deutsch- und englischsprachige Quellen gesucht. Es wurde u. a. nach folgenden Stichworten und Kombinationen (UND, ODER) dieser gesucht:

- Serious Gaming, Serious Games, Serious Game
- Gamification
- Security Awareness, Human Factor, Sicherheitsbewusstsein, Faktor Mensch, IT-Sicherheit, IT Security, Information Security, Informationssicherheit
- Spielend lernen
- Habit forming, Habits, Verhaltensänderung, Verhalten, Routine, Angewohnheit, Gewohnheiten

Ebenfalls mit einbezogen wurden die Datenbanken der Universitäts- und Landesbibliothek der Heinrich-Heine-Universität Düsseldorf und der Bibliothek der Hochschule Düsseldorf. Alle Quellen wurden mit dem Schneeballsystem betrachtet. Das Ziel war es, die Primärquellen oder Standardwerke der Literatur zu den gesuchten Themen herauszufinden. Als Primärquellen wurde häufig zitierte Literatur ausgewählt. Ebenfalls ausgewählt wurden spezifischere Paper oder Bücher, die speziell zum Thema Serious Gaming und Security Awareness passen. Für fachbereichsübergreifende Themen wie z. B. das Lernen wurde nach Standardliteratur im entsprechenden Gebiet, in diesem Fall Psychologie, gesucht.

Die Recherche wurde terminiert, als durch die Stichwortsuche und durch das Schneeballsystem keine weitere Literatur gefunden werden konnte und zu jeder Literatur mindestens eine Alternative in Betracht gezogen wurde.

2.2.2 Merkmale eines lehrreichen Serious Games

Ein besonders lehrreiches Serious Game ist ein Spiel, welches Wissen und Verhaltensweisen effizient vermitteln kann. James Paul Gee hat 2009 sechs Eigenschaften formuliert, die ein Serious Game besitzen sollte, um einen starken Lerneffekt und eine Bindung des Spielers zum Spiel zu erreichen.

Eigenschaft 1: Ziele

> „Gaming as psyching out how rules can be used for one's advantage to accomplish goals to which one is personally and emotionally attached.“ (Ritterfeld et al., 2009, Seite 68)

In den meisten Spielen geht es im Prinzip um Problemlösungen. Jedes Spiel besitzt seine eigenen Regeln, welche mehr oder weniger stark ausgeprägt sein können. Manche dieser Regeln sind explizit vorgegeben, andere sind eher regelmäßig auftretende Verhaltensweisen des Spiels, die vom Spieler entdeckt werden. Diese Regeln beschränken die Handlungsmöglichkeiten, um ein bestimmtes Problem zu lösen (Ritterfeld et al., 2009, Seite 68).

Die Lösung eines Problems, etwa ein Rennen zu gewinnen, ist ein Ziel, dass oft von dem Spielentwickler festgelegt und vom Spieler akzeptiert wird. Dadurch entsteht noch keine tiefe Bindung zwischen dem Spieler und einem Ziel. Dies geschieht erst, indem der Spieler das Ziel auf sich bezieht – so möchte er etwa ein Level in einer bestimmten Zeit schaffen, ohne ein Leben zu verlieren, ein Level unentdeckt bestehen oder in seinem eigenen Stil bewältigen. In manchen Spielen, wie *Die Sims*[1], setzt sich der Spieler fast alle Ziele selbst. Dadurch oder durch den Selbstbezug wird das Problemlösen zu etwas persönlichem und emotionalen. Dies kommt dem Lernen zugute: aufgrund Hirnforschungen ist bekannt, dass Menschen besser und tiefer lernen, wenn eine emotionale oder persönliche Bindung mit dem Lernstoff oder der Problemlösung involviert ist (Ritterfeld et al., 2009, Seite 69).

Eigenschaft 2: Microcontrol

> „Gaming as microcontrol that gives rise to either embodied intimacy or a reach of power and vision.“ (Ritterfeld et al., 2009, Seite 69)

[1] https://www.ea.com/de-de/games/the-sims, aufgerufen am 19.02.2019.

Microcontrol bedeutet die feine Steuerung eines oder mehrere Elemente in einem Spiel, wie etwa ein Charakter in einem Rollenspiel oder viele Einheiten in einem Strategiespiel. Durch Microcontroling erfährt der Spieler das Gefühl einer *verkörperten Macht* innerhalb der Spielwelt. Diese Macht ist traditionellerweise die nahe Umgebung unseres Körpers, mit dem wir auf die Welt Einfluss nehmen können. Durch digitale Spiele kann dieses Gefühl auf die Spielwelt erweitert werden. Bei einem Rollenspiel kann dies so weit gehen, dass der Spieler sich eins mit seinem Charakter fühlt, wodurch die Wahrnehmung im Spiel verkörpert ist. Es hat sich herausgestellt, dass Menschen am besten lernen, wenn das Problemlösen und Denken als verkörpertes Erlebnis geschieht. Dafür muss es sich nicht um die Verkörperung als einzelner Charakter handeln, auch Simulationsspiele wie *Anno*[2], in denen der Spieler eine Art göttliche Macht über die Welt besitzt, haben diese Eigenschaft in abgewandelter Form (Ritterfeld et al., 2009, Seite 69-70).

Eigenschaft 3: Experimentelles Lernen

> „Gaming as experiential learning with all the right conditions for learning from experience met.“ (Ritterfeld et al., 2009, Seite 70)

Neuere Forschungen zeigen, dass das menschliche Gehirn vor allem durch Erfahrungen und nicht durch abstrakte Berechnungen lernt. Es gibt allerdings einige Voraussetzungen an eine Erfahrung, die sie erst lehrreich macht. Diese Anforderungen werden oft von Spielen erfüllt (Ritterfeld et al., 2009, Seite 70-71).

Zuerst muss die Erfahrung einem oder mehreren Zielen zugeordnet sein, was bei Spielen meist der Fall ist. Ob das Ziel erreicht wurde ist dabei unerheblich, da auch das Versagen eine wichtige Rolle im Lernprozess spielt (Ritterfeld et al., 2009, Seite 70).

Der zweite Schritt ist die Interpretation der Erfahrung. Dies geschieht schon während dem Spielvorgang und zeigt sich z. B. im strategischen Denken des Spielers (Ritterfeld et al., 2009, Seite 70-71).

Die dritte Voraussetzung ist ein möglichst sofortiges Feedback. Das Feedback wird selten in direkter Form vom Spiel gegeben, sondern erfolgt eher indirekt bspw. über Diagramme wie eine Lebensanzeige. Feedback wird auch oft in der Community von anderen Spielern, auch bei Einzelspielen, gegeben. Wichtig ist, dass der Spieler versteht, warum seine Aktionen zu bestimmten positiven oder negativen Ereignissen im Spiel geführt haben (Ritterfeld et al., 2009, Seite 71).

Als Nächstes folgt die Anwendung des Gelernten in neuen Situationen. Spieler müssen Gelerntes kombinieren und generalisieren. Viele Spiele erfüllen diese Voraussetzung, indem etwa das Spiel mit dem Spielfortschritt schwieriger wird und neue Herausforderungen bietet (Ritterfeld et al., 2009, Seite 71).

Die fünfte und letzte Voraussetzung ist der Austausch mit anderen Leuten. Dazu gehört die Unterstützung von Anfängern, das Lernen von fortgeschrittenen Spielern und die Diskussion über Lösungswege. Viele Spiele erfüllen diese Voraussetzung nicht direkt. Allerdings gibt es zu vielen Spielen eine Community, in der genau dieser Austausch geführt wird (Ritterfeld et al., 2009, Seite 71).

[2]http://anno.de.ubi.com, aufgerufen am 19.02.2019.

Eigenschaft 4: Affordanz und Effektivität

> „Gaming as finding and using effectivity-affordance matches between bodies or tools and worlds.“ (Ritterfeld et al., 2009, Seite 71)

Gee beschreibt **Affordanz** als ein nutzbares Feature einer Welt (Ritterfeld et al., 2009, Seite 72). Als Beispiel hierfür kann die *Batman*[3] Spielreihe verwendet werden, bei der der Spieler in die Rolle des Superhelden schlüpft. Die fiktive Stadt kann durch Laufen, Klettern oder in der Luft gleitend erkundet werden. Eine Hauswand, die durch Klettern bewältigt werden kann, stellt eine Affordanz dar.

Batman besitzt die Fähigkeit, solche Wände empor zu klettern oder einen Enterhaken zu benutzen. Solche Fähigkeiten werden von Gee **Effektivität** genannt (Ritterfeld et al., 2009, Seite 72). Der Spieler erkennt den Zusammenhang zwischen der Wand, die bezwungen werden kann, und den Fähigkeiten von Batman, um dieses Ziel zu erreichen. Er betrachtet das Spiel abstrakt, indem er in der Welt häufig nach Möglichkeiten sucht, den Enterhaken einzusetzen, um vor Gegnern zu entkommen oder sich schnell über die Karte zu bewegen. Damit tritt die visuelle Darstellung in den Hintergrund und Batman wird zum Werkzeug des Spielers, um das Spiel zu gewinnen. Spiele stellen also Verbindungen zwischen Affordanz und Effektivität her.

Dieses Prinzip kann auch auf andere Spielarten wie Strategiespiele übertragen werden. In der Spielreihe *Anno* stehen dem Spieler eine Vielzahl an Werkzeugen zur Verfügung, um eine Stadt in der Spielwelt aufzubauen. Hier stellt der Spieler die Verbindung zwischen den Werkzeugen (Haus oder Straße bauen, Schiff bauen) und den Möglichkeiten (Flaches Terrain kann bebaut werden, Handelsrouten zwischen Häfen können erstellt werden) in der Spielwelt her.

Das abstrakte Betrachten der Spielwelt und die Problemlösung in der virtuellen Welt können auch in der realen Welt angewendet werden (Ritterfeld et al., 2009, Seite 73). Ein Beispiel dafür sind Simulatoren. Der Spieler findet Affordanz-Effektivitäts-Paare in der virtuellen Welt, die in der realen Welt ebenfalls funktionieren.

Eigenschaft 5: Modelling

> „Gaming as modeling and using models to make learning from concrete experience more general and abstract.“ (Ritterfeld et al., 2009, Seite 74)

Modelle sind für das Lernen wichtig, da sie einen Schritt von der Erfahrung zur Generalisierung einer Problemlösung sind. Außerdem können durch Modelle komplexe Sachverhalte auf die nötigsten Informationen reduziert werden, was das Lernen vereinfacht (Ritterfeld et al., 2009, Seite 77). Daher sind Modelle auch in der realen Welt, beispielsweise in den Wissenschaften, weit verbreitet. In der *Anno* Spielreihe finden sich viele Modelle. Zur Produktion von Brot wird ausschließlich Mehl benötigt. In der realen Welt wären noch einige weitere Zutaten nötig – das Spiel enthält hier also das Modell einer Bäckerei. Schiffe benötigen keine Besatzung und keine Kanonenkugeln um zu schießen – auch dies ist ein Modell. Ebenfalls ist die Interaktion mit Mitspielern ein Modell, sie beschränkt sich auf Handel, Schutzgeldzahlungen und Kriegshandlungen.

[3] https://www.batmanarkhamknight.com, aufgerufen am 19.02.2019.

Eigenschaft 6: Die Geschichte des Spielers

> „Games as player-enacted stories or trajectories.“ (Ritterfeld et al., 2009, Seite 77)

Viele Spiele besitzen eine Geschichte, welche vom Hersteller erstellt wurde. Die Geschichte, die Gee mit der letzten Eigenschaft meint, ist eine andere. Es ist die persönliche Geschichte des Spielers, die er während des Spiels erlebt. Am Beispiel von Batman ist dies gut zu erkennen: das Spiel besitzt eine recht lineare Geschichte, allerdings gibt es schwierige Kämpfe und andere Hindernisse, die vom Spieler überwunden werden müssen. Kann er etwa einen schweren Boss-Kampf nicht gewinnen, hat er die Möglichkeit, die Welt weiter zu erkunden, Nebenmissionen zu erledigen und seine Fähigkeiten zu trainieren. Nach einer gewissen Zeit kehrt der Spieler dann zu dem Kampf zurück und besiegt den Gegner nun. Solch ein Erfolgsereignis kann für den Spieler zum persönlichen Highlight seiner Spielgeschichte werden. Möglicherweise spielt der Spieler das Spiel auch mehrmals durch, vielleicht auch mit unterschiedlichen Schwierigkeitsgraden. In jedem Fall wird sich die Geschichte jedes Mal etwas unterscheiden, so, wie sie sich auch von der anderer Spieler unterscheidet. Diese Eigenschaft erzeugt eine emotionale Bindung zu dem Spiel (Ritterfeld et al., 2009, Seite 77-79).

2.2.3 Vor- und Nachteile

Dieser Abschnitt erfüllt Ziel Z1. Im Verlauf dieser Arbeit wurden bereits einige Vorteile und Chancen von Serious Gaming aufgezählt:

- Serious Games motivieren langweilige, trockene oder sehr komplexe Themen zu lernen (Prensky, 2000, Kapitel 1).
- Sie sind für jede Altersgruppe geeignet (Prensky, 2000, Kapitel 1).
- Sie fördern Problemlösung und Kreativität (Prensky, 2000, Kapitel 5).
- Sie sind adaptiv (Prensky, 2000, Kapitel 5).
- Sie liefern sofortiges Feedback (Prensky, 2000, Kapitel 5).
- Sie bieten eine sichere Umgebung zum Lernen und Ausprobieren von Lösungen (Abt, 1987, Seite 13).
- Sie können Reflexe, Koordination und Verhalten trainieren (Ritterfeld et al., 2009, Seite 3).

Prensky sieht digitales, spielbasiertes Lernen als die Zukunft für Personen jeden Alters. Als Basis zu dieser These legt er die technologische Entwicklung, welche in den letzten Jahrzehnten geschehen ist. Da Menschen bereits in sehr jungem Alter viel Kontakt mit digitalen Medien und Unterhaltung haben, würde das neue Anforderungen und Möglichkeiten an das Lernen stellen (Prensky, 2000, Seite 6). Es gibt aber auch Limitationen und Nachteile:

- Das Erstellen eines Serious Games erfordert mehr Arbeit als bei klassischen Lehrmitteln.[4]
- Das Lernen kann zeitintensiver sein.[5]
- Serious Games können vom Lehrinhalt ablenken.[6]

Abschließend betrachtet kann davon ausgegangen werden, dass Serious Games in vielen Bereichen ein großes Potential für erfolgreiches Lernen bieten.

2.3 Serious Games zum Thema Security Awareness

Zum Thema Eignung und Effektivität von Serious Games zur Vermittlung von Security Awareness existieren nur unzureichende Untersuchungen und wenige Studien. In diesem Abschnitt werden zwei Untersuchungen kritisch betrachtet.

2.3.1 Eignung

Zuerst soll betrachtet werden, inwiefern sich Serious Games zur Vermittlung von Security Awareness eignen. Dieses Thema wurde von Hendrix et al., 2016 in einem Paper behandelt, indem die Autoren zum Thema Literatur gesucht und ausgewertet haben. Es hat sich gezeigt, dass zu keinem Serious Game eine aussagekräftige Evaluation durchgeführt wurde, zu manchen gar keine. Durchgeführte Evaluationen wurden mit kleinen Stichproben oder schlecht messbaren Ergebnissen erstellt. Bei allen zeigt sich allerdings durchweg ein positives Feedback der Spieler und der Autoren (Hendrix et al., 2016, Kapitel 3.1). Die Autoren kamen zum Schluss, dass sich Serious Games für dieses Thema eignen, dass aber ein Mangel an wissenschaftlichen Untersuchungen zu dazu existiert.

2.3.2 Effektivität

2011 wurde ein Paper von Kahn und anderen veröffentlicht, welches die Effektivität verschiedener Methoden zur Vermittlung von Security Awareness theoretisch untersuchte. Dabei wurden fünf Aspekte aufgestellt, die eine Methode zur Security-Awareness-Vermittlung besitzen sollte (Khan, 2011):

1. Wissen
2. Haltungsänderung
3. Soziale Normen
4. Intention
5. Verhaltensänderung

[4] http://edutechwiki.unige.ch/en/Serious_game, aufgerufen am 07.12.2018.
[5] ebd.
[6] ebd.

Die Lernmethoden Präsentation, E-Mail, Gruppendiskussion, Newsletter, Videospiel, Computerbasiertes Training und Poster wurden auf diese fünf Aspekte untersucht. Der Gewinner der Untersuchung war die Gruppendiskussion mit fünf Punkten. Videospiele waren zusammen mit Computerbasiertem Training, Newslettern und Postern auf dem letzten Platz mit zwei Punkten. Dabei wurden die Punkte für Videospiele an Aspekt zwei und vier vergeben. Als Begründung führen die Autoren auf, dass Computerspiele nur schlecht in der Lage seien, Wissen zu vermitteln (Khan, 2011). Der Spieler benötige das Wissen schon im Voraus, bevor er das Spiel spielen könne. Auch könne ein Computerspiel keine firmenspezifischen Inhalte vermitteln. Zwar würden sie helfen, den Spieler zu motivieren, aber aufgrund der schlechten Wissensvermittlung würden sie zu einer geringen Verhaltensänderung führen (Khan, 2011). Im Folgenden wird untersucht, ob diese Beurteilung angemessen ist.

Wissen

Es gibt zahlreiche Beispiele von Computerspielen, die das Argument der Wissensvermittlung widerlegen. Simulationen eignen sich durchaus zur Wissensvermittlung, da mit ihnen beispielsweise die Funktion komplexer Maschinen gelernt werden kann und somit auch Wissen vermittelt wird. Manche Rollenspiele besitzen ausgeklügelte und ausführliche Geschichten, die vom Spieler gelernt werden. Und durch Strategiespiele kann z. B. Wissen über komplexe Zusammenhänge vermittelt werden. Ebenfalls kann, auch wenn es nicht besonders intuitiv ist, Wissen mittels einfacher Texte, Grafiken oder Videos in den Spielen vermittelt werden. Diese Form der Wissensvermittlung entspricht mindestens der eines Posters oder eines Newsletters, da sogar noch weitere Medien genutzt werden können. Poster und Newsletter wurden beide als valides Mittel zur Wissensvermittlung angesehen und daher sollte für ein Videospiel das Gleiche gelten. Firmenspezifische Elemente werden zwar oft von Serious Games außen vor gelassen, allerdings ist das keine Beschränkung eines Serious Games per se, wie die Neuimplementierung von *What the Hack* durch seine Erweiterbarkeit geigen soll. Somit ist die erste Komponente *Wissen* auch bei Videospielen vorhanden.

Soziale Normen

Die einzige Methode, welche diesen Aspekt in der Untersuchung erfüllt, ist die Gruppendiskussion. Durch die Interaktion und Diskussion in der Gruppe können soziale Normen geändert werden (Khan, 2011). Mit einem einfachen Einzelspieler-Videospiel ist dies in diesem Maße nicht möglich. Allerdings gibt es auch einige Serious Games, wie das *Kaspersky Industrial Protection Simulation Business Game*, die betreut und in einer Gruppe gespielt werden. Durch die Gruppeninteraktion ist dabei eine Änderung der sozialen Normen möglich. Dieser Aspekt ist also von der Art des Spieles abhängig, kann aber auch mit Serious Games erreicht werden.

Haltungsänderung und Intention

Wie Khan selbst schreibt, ist eine Haltungsänderung und eine positive Intention mit einem Serious Game zu erreichen (Khan, 2011). Mit positiver Intention ist gemeint, dass die Person einen Willen zeigt, sich zum Thema richtig zu verhalten.

Verhaltensänderung

Um eine Verhaltensänderung beim Nutzer zu bewirken gibt es verschiedene Modelle, die beim Serious Gaming genutzt werden (Tanenbaum, Antle & Robinson, 2013). Diese können abhängig vom Lehrinhalt genutzt und kombiniert werden.

Das *Information Deficit* Model besagt, dass ein Fehlverhalten aufgrund eines Informationsmangels stattfindet (Tanenbaum et al., 2013). Durch das Lernen neuer Informationen werden Denkprozesse angestoßen, die auch zur Verhaltensänderung führen können.

Für manche Thematiken, wie z. B ein umweltbewusstes Verhalten, ist dies nicht immer ausreichend (Tanenbaum et al., 2013). Für solch soziale Verhaltensweisen können die Modelle *Procedural Rhetoric* und *Emergent Dialogue* bessere Ergebnisse erzielen. Das *Procedural Rhetoric* Modell besagt, dass Interaktionen im Spiel überzeugender sind, als Informationen, die dem Nutzer gezeigt werden (Tanenbaum et al., 2013). Der Nutzer soll den Lehrinhalt selbst „erleben“ und ausprobieren anstatt nur über ihn zu lesen.

Das *Procedural Rhetoric* Modell ist speziell auf Gruppenverhalten ausgelegt und setzt den Dialog in den Mittelpunkt. Es besagt, dass nicht nur Informationen und Ansichten zu neuem Verhalten führen, sondern auch umgekehrt das Verhalten die Ansichten einer Person prägen kann. Durch den Dialog werden die Spieler motiviert, sich mit der Thematik zu beschäftigen und eigene Ansichten zum Thema zu bilden (Tanenbaum et al., 2013).

Zusammenfassung

Zusammenfassend können also, abhängig nach Art des Spiels, vier bis fünf Punkte an ein Videospiel vergeben werden. Damit gehören Videospiele nach dieser Beurteilung zusammen mit der Gruppendiskussion (fünf Punkte) und einer Präsentation (vier Punkte) zu den effektivsten Methoden. Insgesamt kann allerdings keine aussagekräftige Antwort zu der Frage der Effektivität aufgrund des Mangels an aussagekräftigen Evaluationen gegeben werden.

2.4 Erhältliche Spiele

Dieser Abschnitt erfüllt das Ziel Z1. Es gibt einen Überblick über verfügbare Serious Games, die Security Awareness lehren.

2.4.1 Vorgehen bei der Recherche

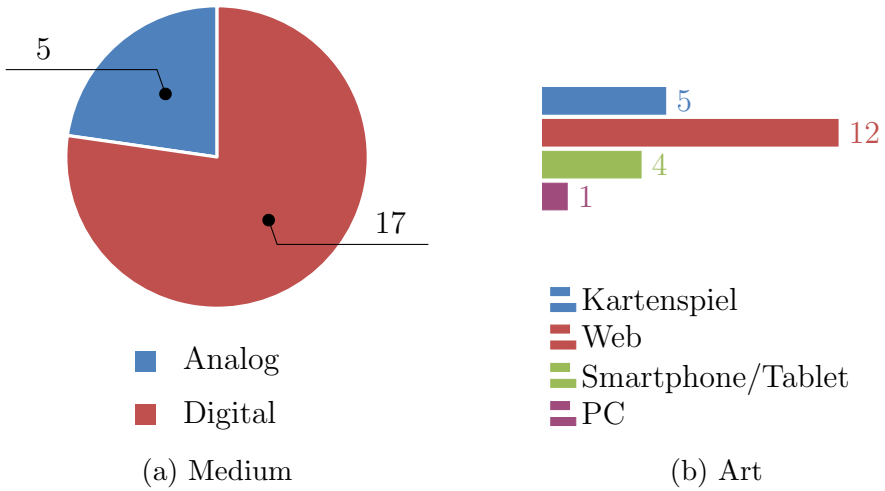
Zu Beginn der Recherche wurden alle Serious Games zum Thema Security Awareness, die in der gefundenen Literatur referenziert wurden, in eine Liste aufgenommen. Des Weiteren wurde eine Stichwortsuche u. a. mit folgenden Stichworten und Kombinationen (UND, ODER) dieser auf Google und Yahoo durchgeführt:

- Serious Game(s), Serious Gaming
- Security Awareness, Human Factor, Faktor Mensch, IT-Sicherheit, IT Security, Information Security, Informationssicherheit
- Games to raise Security Awareness

Die Recherche wurde terminiert, als keine weiteren Spiele gefunden wurden. Dies ist bewusst keine vollständige Recherche und soll als Überblick über die verbreiteten und somit leichter zu findenden Serious Games zum Thema dienen. Spiele, die nicht mehr erhältlich zu sein schienen, wurden ausgelassen. Die Spiele wurden auf mehrere Kriterien untersucht. Spiele ohne ausreichende Beschreibung wurden angespielt, um sie auf die Kriterien hin zu untersuchen.

2.4.2 Marktanalyse der erhältlichen Spiele

Durch die Recherche wurden 22 Spiele gefunden, welche im Anhang unter A.1 tabellarisch aufgelistet sind. Die Liste umfasst frei erhältliche und kommerzielle Spiele. Diese wurden teilweise von Studenten, aber auch von professionellen Spieleherstellern, Beratungsfirmen, IT-Security-Trainingsfirmen, staatlichen Einrichtungen oder dem Militär entwickelt. Somit ist eine breite Diversität des Datensatzes gegeben.



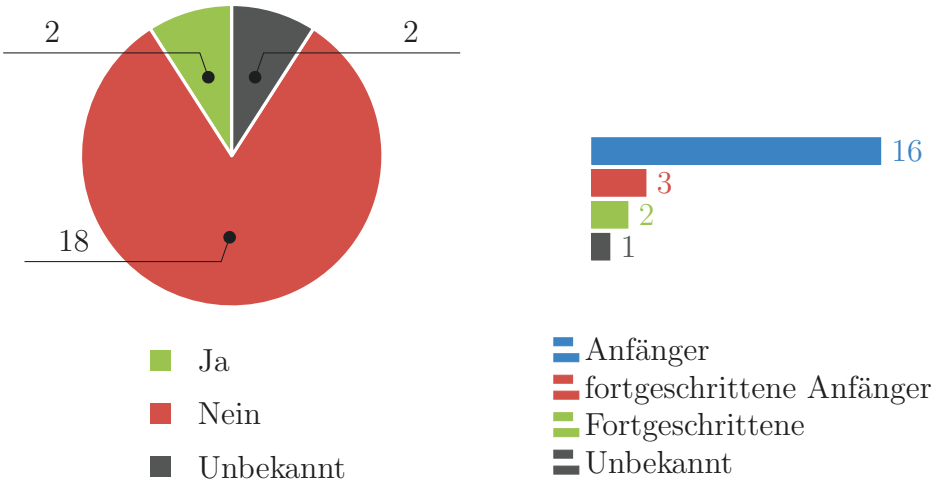
(a) Medium (b) Art

Abbildung 2.2: Ausprägung der Spiele

Fünf der untersuchten Spiele sind Kartenspiele und damit die einzigen analogen Spiele des Datensatzes (siehe Abb. 2.2a). Die am häufigsten vertretene Art sind Browsergames (siehe Abb. 2.2b). Das Spiel *BeOne VR*[7] sticht als einziges Smartphone-VR-Spiel heraus. Zum Zeitpunkt dieser Arbeit war das Spiel bisher nur angekündigt, weshalb im Datensatz die Fragen zu Szenarien und Schwierigkeit nicht beantwortet werden konnten.

Ein wichtiges Thema bei Serious Games ist die Adaptivität an den Spieler, welche etwa durch verschiedene Schwierigkeitsstufen gegeben werden kann. Ein leichtes Spiel ist nur in der Lage Anfängern etwas beizubringen. Personen, die sich im Thema bereits besser auskennen, benötigen auch einen entsprechenden Lehrinhalt im Spiel. Ebenfalls sollte die Schwierigkeit zu Spielbeginn einstellbar sein. Sind die Aufgaben für den Spieler zu trivial, kann ihn dies langweilen und davon abschrecken, weiter zu spielen. Von den untersuchten Spielen unterstützen nur zwei

[7] https://www.beonedevelopment.com/de/digitale-lernloesungen/vr-game/, aufgerufen am 19.02.2019.



(a) Ist das Spiel adaptiv? (b) Schwierigkeit des Spiels

Abbildung 2.3: Adaptivität im Sinne von Schwierigkeitsstufen

explizit mehrere Schwierigkeitsstufen, 18 unterstützen keine und bei zwei ist diese Eigenschaft unbekannt (siehe Abb. 2.3a). Allerdings können durch verschiedene Szenarien ebenfalls verschiedene Schwierigkeitsstufen implementiert werden (siehe folgender Absatz). Auffällig ist, dass sich die meisten Spiele an Anfänger richten (siehe Abb. 2.3b), nur drei fokussieren fortgeschrittene Anfänger und zwei Fortgeschrittene. Bei den untersuchten Spielen für Fortgeschrittene handelt es sich ausschließlich um Spiele, die vom Militär oder staatlichen Einrichtungen (*CyberCIEGE*[8], *Operation Digital Chameleon*[9] und *CyPhinx*[10]) sowie Cybersecurity-Unternehmen (*Kaspersky Industrial Protection Simulation Business Game*[11] und *Game of Threats*[12] von *pwc*) entwickelt wurden.

14 der 22 Spiele bieten nur ein Szenario, dass der Spieler spielen kann, an (siehe Abb. 2.4). Diese Spiele haben eine beschränkte Einsatzmöglichkeit und behandeln oft gut generalisierbare Themen. Sieben der Spiele bieten mehrere Szenarios oder sind durch eigene Szenarios erweiterbar. Erweiterbare Szenarios sind hierbei besonders interessant, da sie sich z. B. speziell an Firmen anpassen können und einen variablen Lehrinhalt bieten. Für das Spiel *CyberCIEGE* lassen sich etwa mittels eines *Scenario Development Kits* eigene Szenarien beliebiger Komplexität erstellen. Spiele dieser Art besitzen zwar möglicherweise keine Schwierigkeitseinstellung, allerdings werden hier die verschiedenen Schwierigkeitsstufen durch verschiedene Szenarien gegeben.

Insgesamt wurden sieben verschiedene Lehrinhalte bei den Spielen gefunden. Die am häufigsten vertretene Kategorie ist die allgemeine IT-Sicherheit (siehe

[8]https://my.nps.edu/web/c3o/cyberciege, aufgerufen am 19.02.2019.
[9]https://dl.acm.org/citation.cfm?id=2957800, aufgerufen am 19.02.2019.
[10]https://www.cybersecuritychallenge.org.uk/competitions/play-demand-cyphinx, aufgerufen am 19.02.2019.
[11]https://media.kaspersky.com/en/business-security/enterprise/KL_SA_KIPS_overview_A4_Eng_web.pdf, aufgerufen am 19.02.2019.
[12]https://www.pwc.co.uk/issues/cyber-security-data-privacy/services/game-of-threats.html, aufgerufen am 19.02.2019.

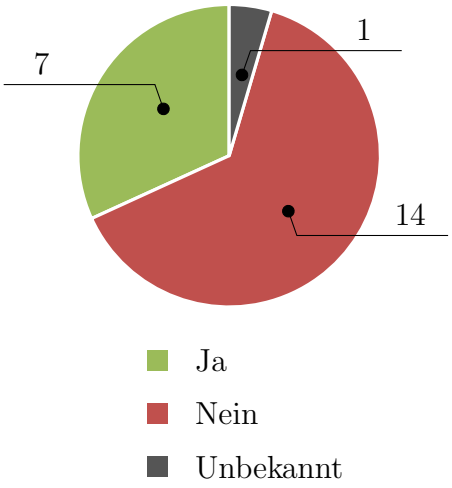


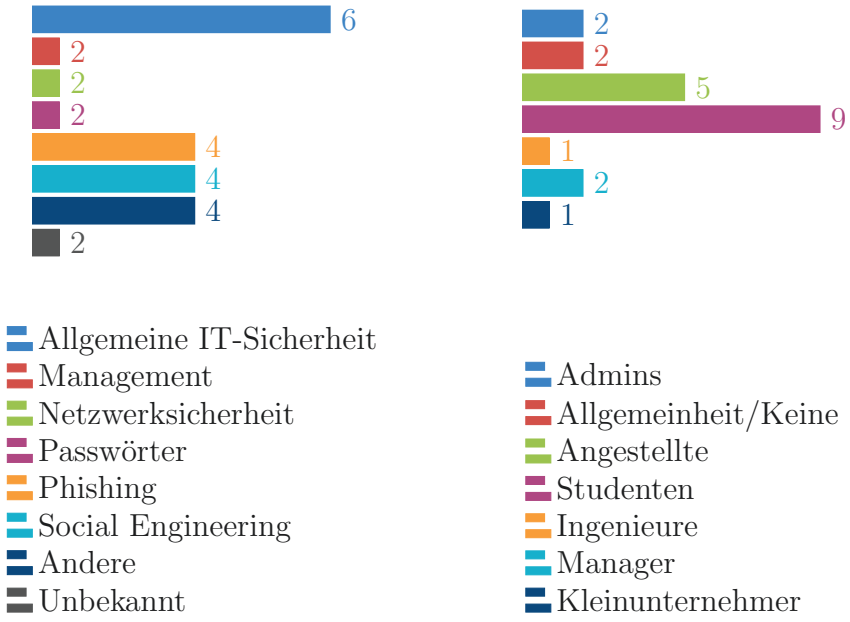
Abbildung 2.4: Unterstützung mehrerer Szenarien

Abb. 2.5a). Das Thema wird von kleinen und großen Spielen wie z. B. *CyPhinx* behandelt. Ebenfalls häufig vertreten sind die Themen Phishing, Social Engingeering und Andere. Darunter sind die Themen Forensik, Geräteschutz, Dokumentenklassifizierung oder Video Downloads zusammengefasst. Zwei Spiele beinhalten auch Themen zum Management wie Kosten eines Cyberangriffs, ökonomische Aspekte von IT-Sicherheit und die Kooperation zwischen dem Management und IT.

Am häufigsten sind die Spiele an Studenten gerichtet (siehe Abb. 2.5b). Dies rührt allerdings daher, dass in dem Datensatz eine sechsteilige Spielreihe der Universität Adelaide enthalten ist, welche alle Studenten und Universitätsmitarbeiter als Zielgruppe haben. Die zweite Zielgruppe sind Angestellte von Firmen. Diese Spiele sind darauf ausgerichtet, allgemeine IT-Sicherheit im Unternehmen zu schulen. Weitere Zielgruppen sind IT-Admins (2), die Allgemeinheit (2), Ingenieure (1) und Manager (2).

Zusammenfassend können einige Beobachtungen, die für diese Arbeit relevant sind, aus den Ergebnissen abgeleitet werden. Bei den Berechnungen wurden unbekannte Ergebnisse außen vor gelassen:

1. Nur 10% der untersuchten Spiele unterstützen mehrere Schwierigkeitsgrade und 33% der Spiele unterstützen mehrere Szenarien.
2. 76% der Spiele sind für Anfänger ausgelegt.
3. Die Lehrinhalte allgemeine IT-Sicherheit, Social Engineering und Phishing sind in 58% der Spiele vertreten. Spezifischere Themen wie z. B. Netzwerksicherheit (8%) sind in der Minderheit.
4. Der Großteil der Spiele ist für eine allgemeine Weiterbildung, unabhängig von Tätigkeit im Unternehmen, ausgelegt. Ingenieure sind nur in 5% der Spiele die Zielgruppe, Manager und IT-Admins jeweils mit 9%.
5. 14% der Spiele sind erweiterbar oder an eigene Bedürfnisse anpassbar.
6. Es existieren kaum stark spezialisierte Spiele wie das *Kaspersky Industrial Protection Simulation Business Game*.



(a) Lehrinhalt der Spiele (b) Zielgruppen

Abbildung 2.5: Inhalte und Zielgruppen

Diese Ergebnisse decken sich mit denen von Hendrix et al., 2016, Kapitel 3. Die Autoren haben ebenfalls eine Marktanalyse zu Serious Games zum Thema Security Awareness aufgestellt. Sie sind zum Ergebnis gekommen, dass die meisten Spiele an Endnutzer gerichtet sind und das Thema „Sicherheit Online“ behandeln. Es existiert ein deutlicher Mangel von Spielen mit Zielgruppen wie Management oder IT Experten (Hendrix et al., 2016, Kapitel 4).

2.5 Bestandsaufnahme von *What the Hack*

Dieser Abschnitt widmet sich der originalen Implementierung von *What the Hack*. Es bietet einen Überblick über die Funktionen des Spiels.

2.5.1 Das Spiel

What the Hack ist eine Wirtschaftssimulation, bei der Spieler in die Rolle des Managers einer White-Hat-Hacking Firma schlüpft. **White-Hat-Hacking** bedeutet ethisches Hacking, wie etwa Penetration-Testing.[13] Seine Aufgabe ist es, Mitarbeiter auszuwählen, einzustellen und mit ihnen Aufträge zu erfüllen. Wie in Abb. 2.6 zu sehen, ist die isometrische Spielwelt das Büro der Firma. Das Spiel ist im Pixel-Art Stil gehalten.

[13]https://us.norton.com/internetsecurity-emerging-threats-what-is-the-difference-between-black-white-and-grey-hat-hackers.html, aufgerufen am 11.03.2019.

Abbildung 2.6: Hauptspielschirm des originalen *What the Hack*

Interaktion

Links neben der Statusanzeige, mit einer Übersicht über das Kapital, Zeit und weiteren Informationen, finden sich Buttons, um die wichtigsten Funktionen im Spiel zu öffnen (siehe Abb. 2.6). Diese öffnen Fenster zur Verwaltung von Mitarbeitern, Aufträgen und Inventar (siehe Abb. 2.7). Wurde ein Auftrag angenommen, können Mitarbeiter auf dem Spielfeld ausgewählt und durch Tippen einem Arbeitsplatz zugewiesen werden. Nachdem eine Aufgabe für den Mitarbeiter gewählt wurde, begibt sich dieser zu dem Arbeitsplatz und beginnt seine Arbeit.

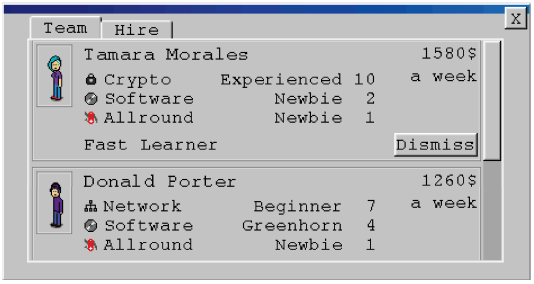
Mitarbeiter

Die meisten Mitarbeiter sind zufällig generiert. Das beinhaltet ihren Namen, ihre Fähigkeiten (im Folgenden `SkillSet`), Gehalt, Level und Aussehen. Es gibt aber auch einen `SpecialEmployee` mit eigenen Sprites. Die Skills der Mitarbeiter verbessern sich mit der Zeit. Weiterhin gibt es verschiedene Eigenschaften wie `Fast Learner` oder `Unreliable`, welche erlernt oder bereits vorhanden sein können. Jeder Mitarbeiter kann nur an einem Auftrag gleichzeitig arbeiten.

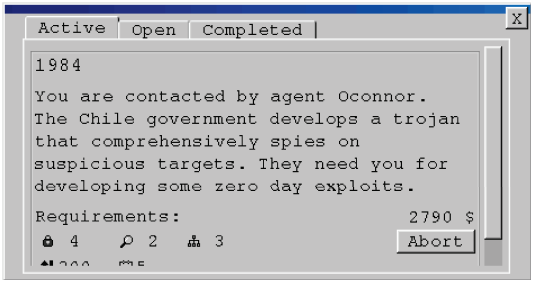
Aufträge

Aufträge bestehen aus einem Titel, einer Beschreibung, einer Deadline und einem Preis, der bei erfolgreichem Abschluss an den Spieler ausgezahlt wird. Die Aufträge erfordern immer eine Auswahl aus 1–3 Fähigkeiten (z. B. `Hardware` und `Social Engineering`). Falls ein Mitarbeiter einen Auftrag bearbeitet, der eine Fähigkeit benötigt, die er nicht hat, kommt die `General Purpose` Fähigkeit zum Einsatz. Diese ist weniger effektiv als eine spezielle Fähigkeit. Es können mehrere Mitarbeiter an einem Auftrag arbeiten.

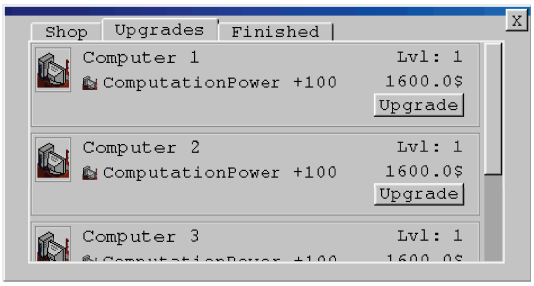
Die Aufträge behandeln alle das Thema IT-Sicherheit und sind in einer JavaScript Object Notation (JSON) Datei definiert. Um dem Spiel eine gewisse Ab-



(a) Mitarbeiterverwaltung



(b) Auftragsverwaltung



(c) Itemverwaltung

Abbildung 2.7: Wichtige Fenster im original *What the Hack*

wechslung zu bieten, können dort im Text Platzhalter für Namen oder Firmennamen eingesetzt werden, welche aus einer separaten Liste bei der Generierung der Aufträge eingesetzt werden. Listing 2.1 zeigt ein Beispiel für eine Definition:

```
1   {"name": "C'mon Honey",
2     "description": "The %TOWN% %INSTITUTION%
        frequently gets hacked by some bad guys.
        Install a honeypot in their IT system to catch
        those script kiddies.",
3     "onSuccess": "You setup a sandboxed server with
        some fake data that looks like an easy target.
        In fact, every access is logged and carefully
        analysed. Two day later the fish gets caught in
         the net.",
4     "onFail": "",
5     "skill": [
6       {"type": "Network", "value": 0},
7       {"type": "Software", "value": 0},
8       {"type": "Search", "value": 0}
9     ],
10    "risk": 0,
11    "hardness": 1.2,
12    "duration": 8,
13    "minLevel": 0,
14    "maxLevel": 0}
```

Listing 2.1: Definition eines Auftrags

Spielmechanik

Ein Spieltag ist in neun Zeitschritte z unterteilt. Ein Zeitschritt entspricht 4,4 realen Sekunden. Zu jedem Zeitschritt wird der Fortschritt von Mission F berechnet. Dies geschieht, indem für jede Fähigkeit eines Auftrags und für jeden Mitarbeiter E, der an ihm arbeitet, eine Zufallszahl zwischen Null und Eins bestimmt wird. Mit diesem Wert wird das aktuelle Level der Fähigkeit des Mitarbeiters E_s sowie eine Konstante k verrechnet (siehe 2.1).

$$F = \sum_{E=1}^{n} \text{random}(0.0, 1.0) \cdot k \cdot E_s \cdot \frac{1}{z} \tag{2.1}$$

Setzt man die Konstanten ein, erhält man folgende Formel:

$$F = \sum_{E=1}^{n} \text{random}(0.0, 1.0) \cdot 0.9 \cdot E_s \cdot \frac{1}{9} \tag{2.2}$$

Ist eine Spielwoche vergangen, werden alle Angestellten von dem Kapital des Spielers ausbezahlt. Sinkt das Kapital unter Null, dann ist das Spiel beendet.

Technische Details

Das Spiel wurde in `Java` mit der Game-Engine `libgdx`[14] erstellt. Es ist unter *Linux*, *macOS*, *Windows* und *Android* lauffähig. Der Spielstand wird mit Hilfe der `protocol-buffers`[15] Bibliothek automatisch gespeichert und geladen.

Zielgruppe

Der Lehrinhalt des Spiels ist auf eine Allgemeinbildung zum Thema IT-Sicherheit ausgelegt. *What the Hack* lebt allerdings von seinem „Nerd-Humor“. Dadurch ist das Spiel besonders interessant für technisch versierte Menschen. Dies konnte auch bei der Ausstellung des Spiels beim Tag der offenen Tür der Hochschule Düsseldorf beobachtet werden.

2.5.2 Probleme

Das Spiel wurde nicht vollständig fertig implementiert. Am auffälligsten ist die Spielmechanik, welche nicht feinjustiert wurde. Dadurch ist das Spiel sehr leicht zu gewinnen und bietet wenig Herausforderungen. Die unterschiedlichen Fähigkeiten machen keinen spielerischen Unterschied, da die `All Purpose` Fähigkeit stark genug für alle Aufgaben ist. Ebenfalls existiert im Spiel die Ressource „Bandbreite“, welche keinerlei spielerischen Einfluss besitzt.

Die Game-Engine `libgdx` bringt auch Probleme mit sich, die allerdings die Entwicklung betreffen. Im Gegensatz zu Unity existiert hier kein Editor, mit dem die visuellen Komponenten leicht implementiert werden können. Stattdessen wird bei `libgdx` z. B. die Oberfläche im Quellcode programmiert. Für unerfahrene Programmierer ist dies eine Hürde und erfordert einige Einarbeitungszeit. Auch während

[14] https://libgdx.badlogicgames.com, aufgerufen am 11.03.2019.
[15] https://developers.google.com/protocol-buffers/, aufgerufen am 11.03.2019.

der Entwicklung führt dies zu vielen „Trial and Error“ Momenten, die viel Zeit kosten.

2.6 Anforderungen an die Neuimplementierung

In diesem Abschnitt werden die Ergebnisse aus Abschnitt 2.2.2 und 2.4.2 mit den Zielen aus Abschnitt 1.4 verglichen und Anforderungen an die Neuimplementierung formuliert. Dieser Abschnitt erfüllt Ziel Z2.

2.6.1 Bewertung von *What the Hack* als Serious Game

Das Spiel und auch der Lehrinhalt wurden mit nur wenig Recherche zum Thema Serious Games erstellt. Daher wird in diesem Abschnitt die Tauglichkeit von *What the Hack* als Serious Game mit den in Abschnitt 2.2 festgelegten Merkmalen eines guten Serious Games überprüft.

Eigenschaft 1: Ziele

What the Hack besitzt kein Spielziel. Vielmehr ist das Ziel nicht zu verlieren. Der Spieler kann sich selbst eigene Ziele stellen: möglichst wirtschaftlich zu spielen, Mitarbeiter möglichst lange zu behalten, so viele Aufträge wie möglich in kürzester Zeit zu erledigen – hier sind keine Grenzen gesetzt. Es fehlt dem Spiel aber noch an motivierenden, langfristigen Zielen. So könnten dem Spieler z. B. attraktivere Missionen angeboten werden, wenn die meisten Aufträge erfolgreich oder besonders schnell abgeschlossen werden. Ein größeres Büro mit mehr Arbeitsplätzen und größeren Aufträgen wäre ebenfalls ein attraktives Ziel. Hier gibt es viele Möglichkeiten. Diese Verbesserung wird als Anforderung A1 in Tabelle 1.1 definiert.

Eigenschaft 2: Microcontrolling

Microcontrolling ist die Hauptinteraktionsart von *What the Hack*. Dies beginnt mit dem Einstellen und Kündigen von Mitarbeiten, dem Kauf von Equipment und dem Annehmen von Aufträgen. Der Spieler entscheidet auch, welcher Mitarbeiter an welchem Auftrag arbeitet. Das Microcontrolling selbst beinhaltet auch einige Aspekte des Lehrinhaltes. Aufträge benötigen bestimmte Fähigkeiten der Mitarbeiter, welche passend herausgesucht werden müssen. Ein Softwarespezialist wird sich bspw. mit einem Social Engineering Auftrag schwertun. Der Spieler lernt auch, welche Fähigkeiten für bestimmte Angriffe oder Aufträge benötigt werden. Insgesamt steht allerdings der wirtschaftliche Aspekt beim Microcontrolling im Vordergrund. Durch gutes Microcontrolling wird die Firma wachsen und interessantere Aufträge können angenommen werden.

Würde der Spieler durch Microcontrolling keine wirtschaftlichen, sondern auf IT-Sicherheit bezogene Aspekte steuern, kann eine bessere Lernleistung erzielt werden. Da dies allerdings die Grundidee des Spiels ist, soll dies auch nicht verändert werden.

Eigenschaft 3: Experimentelles Lernen

In diesem Abschnitt wird untersucht, ob die fünf Voraussetzungen an eine lehrreiche Erfahrung von *What the Hack* erfüllt werden.

1. **Zuordnung zu einem Ziel:** *What the Hack* besitzt keine langfristigen Ziele. Die am meisten präsenten Ziele für den Spieler sind das erfolgreiche Bearbeiten von Aufträgen und das Bezahlen der Mitarbeiter. Durch Anforderung A1 werden bereits neue Ziele für die Neuimplementierung entwickelt.
2. **Interpretation:** *What the Hack* fördert durch das Microcontrolling strategisches Denken und bietet somit eine Grundlage für Interpretationen durch den Spieler.
3. **Sofortiges Feedback:** Wird ein Auftrag abgeschlossen, bekommt der Spieler Feedback mit einer Beschreibung, weshalb der Auftrag erfolgreich abgeschlossen wurde oder fehlgeschlagen ist. Diese bezieht sich allerdings nur auf den Lehrinhalt. Nützlich wäre auch ein Feedback, weshalb der Auftrag aus Sicht der Spielmechanik nicht erfolgreich abgeschlossen wurde. Diese Verbesserung wird als Anforderung A2 in Tabelle 2.1 definiert.
4. **Anwenden des Gelernten:** Durch den zyklischen Verlauf des Spiels besitzt der Spieler viele Möglichkeiten, Gelerntes über das Spiel anzuwenden. Sein Wissen über IT-Security findet aber keine Möglichkeit der Anwendung. Daher sollen interaktive Aufträge in *What the Hack* implementiert werden. Diese Verbesserung wird als Anforderung A3 in Tabelle 2.1 definiert.
5. **Austausch:** Der Austausch mit anderen Spielern ist im Spiel nicht vorgesehen. Möglich wäre ein Austausch innerhalb des Unternehmens, das das Spiel einsetzt.

Eigenschaft 4: Affordanz und Effektivität

Affordanz- und Effektivitäts-Paare existieren in *What the Hack*. Ein Beispiel sind Arbeitsplätze, welche von Mitarbeitern genutzt werden können. Diese Paare besitzen nur einen geringen Zusammenhang mit dem Lehrinhalt. Dies liegt dem Design des Spiels zugrunde, weshalb dieser Aspekt nicht ohne eine grundlegende Änderung des Spiels verbessert werden kann. Es folgen zwei Spielideen, die diesen Aspekt erfüllen würden:

- Eine Wirtschaftssimulation ähnlich zu *What the Hack*, bei der der Spieler sich als Manager um die IT-Sicherheit im eigenen Unternehmen kümmert. Das Spiel würde sich an Manager in Unternehmen richten. Die Affordanz- und Effektivitäts-Paare des Spiels können ebenfalls im realen Unternehmen genutzt werden, um die IT-Sicherheit zu verbessern.
- Ein Rollenspiel bei dem sich der Spieler sicher in einem Unternehmen verhalten muss. Hier wäre die Zielgruppe Angestellte in einem Unternehmen. Auch bei dieser Idee könnten die Affordanz- und Effektivitäts-Paare im realen Leben genutzt werden.

Eigenschaft 5: Modelling

What the Hack ist das vereinfachte Modell einer IT-Sicherheitsfirma. Auch die Missionen und Fähigkeiten im Spiel sind Modelle komplexer Angriffe und Fähigkeiten im echten Leben. Der Lehrinhalt des Spiels wird durch ein Modell dargestellt und erleichtert dadurch dem Spieler das Lernen.

Eigenschaft 6: Die Geschichte des Spielers

What the Hack besitzt keine zusammenhängende Geschichte. Die Geschichte wird durch den Spielverlauf geschrieben. Ein Spiel könnte bspw. so verlaufen: Der Spieler beginnt ein neues Spiel und stellt seinen ersten Mitarbeiter ein. Er erledigt ein paar Aufträge und stellt seinen zweiten Mitarbeiter ein. Durch die Annahme eines zu schweren Auftrags gelangt er an die Grenze zum Bankrott, weshalb einer seiner Mitarbeiter entlassen werden muss. Der Spieler hat seine Fehler bemerkt und schafft es sich vor dem Bankrott zu retten. Nach einiger Zeit floriert sein Unternehmen wieder. Dies ist die persönliche Geschichte des Spielers, welche eine Bindung zum Spiel erzeugt.

Zusammenfassung

Die Eigenschaften „Modelling“ und „Die Geschichte des Spielers“, sowie Teile der Eigenschaften „Microcontrolling“ und „Experimentelles Lernen“ werden von der originalen Implementierung erfüllt. Insgesamt zeigt sich aber ein Verbesserungsbedarf am Game Design von *What the Hack*. Es fällt auf, dass das Spiel nicht mit einem wissenschaftlichen Hintergrund zu Serious Gaming entwickelt wurde.

Das Spielprinzip bietet aber eine sehr gute Basis für eine Weiterentwicklung. Durch die formulierten Anforderungen an die Neuimplementierung können alle Eigenschaften, bis auf Eigenschaft „Affordanz und Effektivität“, vollständig erfüllt werden. Nach diesen Bewertungskriterien ist *What the Hack* mit den Verbesserungen ein gut geeignetes Serious Game zur Verbesserung von Security Awareness.

2.6.2 Marktanalyse

In Abschnitt 2.4.2 sind vier Aspekte besonders aufgefallen: Die meisten Spiele richten sich an Anfänger, besitzen nur eine Schwierigkeitsstufe, sind nicht erweiterbar oder anpassbar und sind wenig spezialisiert. Diese Aspekte sollen in der Neuimplementierung von *What the Hack* beachtet werden. Diese Anforderungen sind bereits in den Zielen formuliert.

- Die Unterstützung mehrerer Schwierigkeitsstufen: Ziel Z7
- Die Erweiterbarkeit und Spezialisierung: Ziel Z5

2.6.3 Anforderungen

Anforderung Nr.	Beschreibung	Ziel Nr.
A1	Entwurf sinnvoller und langfristiger Ziele für *What the Hack*.	Z2
A2	Implementierung von Feedback beim Fehlschlag einer Mission. Das Feedback soll spielerische Aspekte beinhalten.	Z2
A3	Implementierung einer Schnittstelle für interaktive Missionen.	Z2
A4	Unterstützung mehrerer Schwierigkeitsstufen.	Z7
A5	Implementierung von Schnittstellen, einem Plugin-Lade-Mechanismus, eines SDKs und einer Dokumentation zur Erweiterung des Spiels.	Z5

Tabelle 2.1: Anforderungen an die Neuimplementierung

Kapitel 3

Game Design

Dieses Kapitel behandelt die spielerischen Entscheidungen, die für die Neuimplementierung getroffen wurden. In Abschnitt 3.1 wird dabei zunächst auf die Zielgruppe des Spiels eingegangen. Darauf wird in Abschnitt 3.2 die Erweiterbarkeit behandelt. Es folgt die Behandlung der Spielzeit in Abschnitt 3.3, wie in Ziel Z4 formuliert. Ziel Z7 besagt, dass das Spiel mehrere Schwierigkeitsstufen unterstützen soll. Dies wird in Abschnitt 3.4 beschrieben. Abschließend werden die Spielziele in Abschnitt 3.5 betrachtet.

3.1 Zielgruppe

Das originale *What the Hack* hat eine klare Zielgruppe – den privaten Endanwender. Bei der Neuimplementierung ist dies wegen der Erweiterbarkeit nicht der Fall. Das Spiel bietet eine Plattform, mit der Inhalte für beliebige Zielgruppen erstellt werden können. Dennoch eignet sich *What the Hack* hauptsächlich für technische Zielgruppen. Das Spiel nutzt verschiedene „Nerd-Elemente“, um Humor zu erzeugen. Dieser findet hauptsächlich bei technisch versierten Menschen anklang. Mögliche Inhalte sind z. B. SQL-Injections[1], Security bei Docker Containern[2], Spectre und Meltdown[3], Zertifikate[4] oder viele weitere. Auch Themen wie IT-Sicherheit für das Management sind denkbar.

3.2 Erweiterbarkeit

Die Erweiterbarkeit ist eines der wichtigsten Features des Spiels. Es erlaubt die Anpassung und Spezialisierung auf verschiedene Zielgruppen sowie die Einbindung von Unternehmenskultur. Außerdem lässt es sich so auf verschiedene Wissensstände anpassen. Dieser Abschnitt erfüllt Ziel Z5 und die Anforderung A5.

Pro Spiel kann immer nur eine Erweiterung geladen werden. Die Aspekte Mitarbeiter, Fähigkeiten, Missionen und Namen des Spiels sind erweiterbar.

[1] https://www.w3schools.com/sql/sql_injection.asp, aufgerufen am 23.03.2019.
[2] https://docs.docker.com/engine/security/security/, aufgerufen am 23.03.2019.
[3] https://meltdownattack.com/, aufgerufen am 23.03.2019.
[4] https://letsencrypt.org/, aufgerufen am 23.03.2019.

(a) Laufen

(b) Idle

(c) Arbeiten

Abbildung 3.1: Sprites eines Special Employees

3.2.1 Mitarbeiter

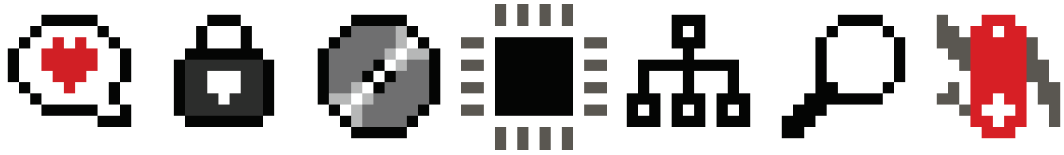
Die meisten Mitarbeiter werden zufällig erzeugt. *What the Hack* unterstützt aber auch sogenannte `Special Employees`. Folgende Eigenschaften können bei solchen Mitarbeitern festgelegt werden:

- Name
- Sprites für Laufen-, Idle- und Arbeiten-Animation mit der Größe **32x64 Pixel**. Siehe Abb. 3.1
- `EmployeeSpecials` wie z. B. `Fast Worker`
- Bedingungen für das Auftauchen als verfügbarer Mitarbeiter

Durch diese Erweiterbarkeit können Unternehmen z. B. den Chef im Spiel als Figur einbinden.

3.2.2 Fähigkeiten (Skills)

Eine Erweiterung hat die Möglichkeit das Standard-Skill-Set durch ein eigenes zu ersetzen. Je nach Lehrinhalt können damit die Fähigkeiten der Mitarbeiter und Missionen genau auf das Thema abgestimmt werden. Der `General Purpose` Skill ist in jedem Fall vorhanden. Jeder Skill benötigt einen Namen und ein Sprite mit der Größe **13x13 Pixel**. In Abb. 4.4 ist das Standard-Skill-Set zu sehen.



Social Engineering | Cryptography | Software | Hardware | Network | Search | General Purpose

Abbildung 3.2: Sprites des Standard Skill-Sets

3.2.3 Missionen

Die Missionen bzw. Aufträge beinhalten den Lehrinhalt des Spiels. Sie benötigen folgende Inhalte:

- **Name**
- **Beschreibungen** für die Mission, einen Erfolg und einen Fehlschlag
- **Dauer** in Tagen
- Liste benötigter **Skills**
- Liste aller **Interaktionen** (**MissionHooks**, siehe folgender Abschnitt)
- **Schwierigkeitsstufe**
- **Hardness**: Faktor, welcher die Schwierigkeit einer Mission innerhalb der Schwierigkeitsstufe festlegt.

Zusätzlich kann das Erscheinungsverhalten einer Mission definiert werden:

- **Benötigtes Level** eines Mitarbeiters
- **Benötigte Anzahl** an Mitarbeitern
- Anzahl **vergangener Tage**
- Liste an **Missionen**, die **abgeschlossen** sein müssen
- **Story-Mission**: Zeige die Mission immer, wenn die Erscheinungsbedingungen erfüllt sind. Ansonsten erfolgt eine zufällige Auswahl. Story-Missionen werden auch nur einmal im Spiel generiert, können aber wiederholt werden, wenn sie nicht erfolgreich beendet wurden.

Wie in der Anforderungstabelle 2.1 unter A3 definiert wurde, sollen die Missionen eine Möglichkeit zur Interaktion bieten. Interaktionen haben folgende Möglichkeiten und Limitationen:

- Interaktionen besitzen eine eigene Oberfläche.
- Die Mission wird während einer Interaktion pausiert.
- Bei erfolgreicher Lösung der Interaktion schreitet die Mission weiter voran. Bei einem Fehlschlag ist auch die ganze Mission fehlgeschlagen.
- Im Echtzeitspielmodus wird der Spieler benachrichtigt, wenn er eine Interaktion durchführen muss. Das Spiel läuft weiter, aber die Mission ist pausiert.

3.2.4 Namen

Eine Erweiterung kann eine eigene Liste an Vor-, Nach-, Firmen- und Städtenamen besitzen. Diese Namen können zusätzlich zu den bereits vorhandenen Namen oder alleinig im Spiel verwendet werden. Dies wird vom Erweiterungsersteller festgelegt.

3.3 Spielzeit

Dieser Abschnitt untersucht die Möglichkeiten verschiedener Mechaniken für die Spielzeit und erfüllt damit Ziel Z4.

3.3.1 Vorgehen bei der Recherche

Zu Beginn der Recherche wurden die bereits gefundenen Quellen nach Informationen zu diesem Thema durchsucht. Zusätzlich wurde eine Stichwortsuche mit folgenden Stichworten und Kombinationen (UND, ODER) dieser auf Google, Google Scholar und den Bibliotheksdatenbanken durchgeführt:

- Serious Game(s), Serious Gaming
- Behaviour (change), habit(s)
- Verhalten, Verhaltensänderung, Gewohnheit(en), Gewohnheitsänderung

Das Schneeballsystem wurde angewendet. Die Suche wurde terminiert, als keine neue Literatur mehr gefunden werden konnte.

3.3.2 Klassisches Spiel und Echtzeit

Wie in Ziel Z4 in Tabelle 1.1 beschrieben ist die Verhaltensänderung ein längerer Prozess, welcher von einmaligem Spielen nicht erreicht werden kann. Diese Tatsache findet seine Grundlagen in der Psychologie. In diesem Abschnitt soll untersucht werden, ob ein Spiel, welches häufig, aber dafür kürzer gespielt wird, sich besser zur Verhaltensänderung eignet als ein selten, dafür länger am Stück gespieltes Spiel. Ein Beispiel für ersteres ist *Clash of Clans*[5]. Das Spiel ist ein Online-Multiplayer-Strategiespiel für mobile Geräte. Der Spieler schlüpft in die Rolle des Oberhaupts eines Dorfes. Mit verschiedenen Ressourcen kann er sein Dorf aufbauen, Soldaten ausbilden und andere Dörfer angreifen.

Clash of Clans ist ein Echtzeitspiel. Das bedeutet, dass alle Aktionen auch weitergeführt werden, wenn der Spieler das Spiel geschlossen hat. Der Bau eines Gebäudes bspw. benötigt eine bestimmte Dauer (von wenigen Sekunden bis mehrere Stunden). Nach Ablauf dieser Dauer bekommt der Spieler eine Benachrichtigung. Ebenfalls kann der Spieler zu jeder Zeit angegriffen werden, auch wenn er das Spiel aktuell nicht spielt. Durch diese Spielmechanik entsteht ein anderes Spielverhalten als bei einem klassischen Spiel. Die Spieler verbringen häufig mehrmals am Tag, dafür jeweils kürzer, Zeit im Spiel. In Abb. 3.3 ist ein Dorf zu sehen bei dem die Kaserne ausgebaut wird. Da nur eine begrenze Anzahl an

[5]https://supercell.com/en/games/clashofclans/, aufgerufen am 13.12.2018.

Abbildung 3.3: Clash of Clans: Anfängliches Dorf in, bei dem die Kaserne ausgebaut wird

Gebäuden gleichzeitig gebaut werden kann, erhält der Spieler 30 Minuten später eine Benachrichtigung (siehe Abb. 3.4). Er kann nun ein weiteres Gebäude bauen.

Um zu untersuchen, ob solch ein Spiel als Serious Game zur Verhaltensänderung geeignet ist, wurde eine Recherche durchgeführt. Allerdings wurde zu dieser Thematik keine Literatur, Veröffentlichungen oder Untersuchungen gefunden. Daher werden in der Neuimplementierung von *What the Hack* beide Varianten integriert, was zukünftige Untersuchungen zu diesem Thema ermöglicht. Es kann eine Spielmechanik für beide Varianten verwendet werden. Durch einen Parameter kann die Geschwindigkeit des Spiels eingestellt werden. In Tabelle 3.1 sind die Unterschiede beider Varianten aufgezählt.

	Klassisches Spiel	**Echtzeit**
Dauer eines Zeitticks	4,5 Sekunden	10 Minuten
Benachrichtigungen	Nein	Die Dauer eines Auftrags wird im Voraus berechnet, Benachrichtigungen werden beim Schließen der App in eine Warteschlange eingereiht.
Erneutes öffnen	Laden des alten Spielstands	Laden des alten Spielstands und Berechnung des Spielfortschritts seit dem letzten Laden.

Tabelle 3.1: Spielzeit: Unterschiede der Spielmechanik

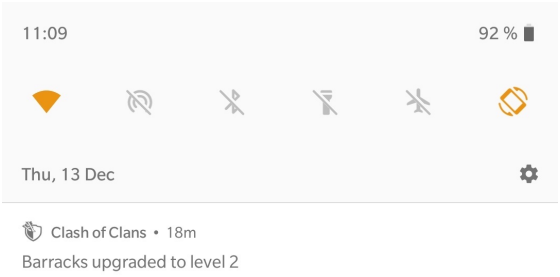


Abbildung 3.4: Clash of Clans: Benachrichtigung über abgeschlossenen Aufbau

3.4 Schwierigkeitsgrade

Dieser Abschnitt beschreibt die Erfüllung von Ziel Z7 und Anforderung A4. Es gibt zwei Arten von Schwierigkeiten, die festgelegt werden können. Zum einen gibt es vier Schwierigkeitsstufen, welche zu Beginn eines neuen Spiels ausgewählt werden: **Easy**, **Normal**, **Hard** und **Guru**. Diese Stufen legen fest, welche Missionen der Spieler zur Auswahl bekommt. Sie sind dafür gedacht, um den Spieler an seinem aktuellen Wissensstand abzuholen. Des Weiteren gibt es den Faktor **Hardness** einer Mission. Er beschreibt die Schwierigkeit einer Mission innerhalb der Schwierigkeitsstufe. Mit diesem Faktor kann z. B. eine besonders leichte Mission erstellt werden, welche dafür auch eine geringere Belohnung hat.

3.5 Ziele

In Anforderung A1 der Tabelle 2.1 wurde festgelegt, dass motivierende und langfristige Ziele für *What the Hack* entwickelt werden sollen. Es folgen drei Ideen für Ziele, die diese Aufgabe erfüllen:

- Bei der originalen Implementierung erreicht man schnell den Punkt, an dem alle vier Arbeitsplätze belegt sind. Daher wäre der Kauf eines größeren Büros, welches mehr Arbeitsplätze und Mitarbeiter beschäftigen kann, ein motivierendes Ziel, um weiter zu spielen. Um die Implementierung und das grafische Design zu vereinfachen, könnte man im selben Haus ein weiteres Stockwerk kaufen.
- Bisher gibt es nur einen Typ von Arbeitsplatz. Verschiedene, spezialisierte Arbeitsplätze (z. B. für Hardware), welche für bestimmte Aufträge benötigt werden, wären eine sinnvolle Erweiterung. Sie fördern zum einen das Verständnis, dass nicht jedes Hacking am Computer geschieht, andererseits bietet es auch Abwechslung. Diese Arbeitsplätze könnten auch von einer Erweiterung bereit gestellt werden.

- Mitarbeiter könnten, anstatt an Aufträgen zu arbeiten, Tools entwickeln, welche im Unternehmen genutzt werden können. Diese Tools könnten Voraussetzungen von Aufträgen sein oder bei der Arbeit an späteren Aufträgen helfen, um sie schneller zu bearbeiten. Mitarbeiter könnten auch Forschung betreiben, um die Firma populärer zu machen, was dazu führt, dass interessantere Aufträge erscheinen.

Kapitel 4

Software Architektur

Dieses Kapitel beschreibt die technische Umsetzung der Neuimplementierung von *What the Hack* und somit auch die Erfüllung von Ziel Z6. Zu Beginn werden allgemeine Themen, wie die Auswahl einer Game Engine und einer Open Source Lizenz, beschrieben. Es folgt eine Übersicht über die Architektur des Spiels. Anschließend wird erläutert, wie die in Ziel Z5 festgelegte Erweiterbarkeit umgesetzt wurde. Daraufhin folgt eine Beschreibung der Herausforderungen des Savegame-Mechanismus, sowie der Besonderheiten bezüglich *Android*- und *iOS*-Support. Abschließend wird das entwickelte Beispiel-Plugin, festgelegt in Ziel Z8, beschrieben.

4.1 Auswahl der Game Engine

Obwohl in den Zielen die `Unity` Game Engine bereits festgelegt wurde, wird im Folgenden die Entscheidung dazu begründet.

4.1.1 Anforderungen

Es werden folgende Anforderungen an die Game Engine gestellt:

- Grafischer Editor für die Entwicklung
- Kostenlose Verfügbarkeit
- Unterstützung von 2D und Pixel Art
- Spiele müssen durch Code und Assets erweitert werden können
- Unterstützung von *Linux*, *macOS*, *Windows*, *Android*, *iOS*

4.1.2 Auswahl

Zur Auswahl standen zwei Game Engines: `Unity`[1] sowie `Godot`[2]. Beide Engines erfüllen alle Punkte der Anforderungen. In Tabelle 4.1 folgt eine Auflistung der Vor- und Nachteile, bezogen auf dieses Projekt.

[1] https://unity3d.com/de, aufgerufen am 21.02.2019.
[2] https://godotengine.org/, aufgerufen am 21.02.2019.

	`Unity`	`Godot`
Vorteile	• Große Community • Verfügbarkeit einer Bibliothek zur Erstellung von Plugins • Eigene Editor-Fenster können erstellt werden • Etabliert innerhalb der Hochschule	• Open Source: Bugs können selbst gefixed werden • Performance des Editors und der Builds (ausgelegt auf schwache Hardware) • Unterstützung vieler Programmiersprachen (*Python*, *C#*, *GDScript*, *Visual Script*, *C++*, ...)
Nachteile	• Mittelmäßiger Support für Nutzer der kostenlosen Version • Closed Source: Bugs können nicht selbst gefixed werden	• Kleine Community, junges Projekt • Kein Plugin Support vorhanden, muss selbst implementiert werden

Tabelle 4.1: Vergleich der Game Engines Unity und Godot

Aufgrund der großen Community, der Verfügbarkeit einer getesteten Bibliothek für Spiel-Plugins und der Etablierung an der *Hochschule Düsseldorf* ist die Entscheidung auf `Unity` gefallen.

4.2 Auswahl der Open Source Lizenz

Das Spiel soll unter einer Open-Source-Lizenz veröffentlicht werden. Im Folgenden wird der Entscheidungsprozess für die Lizenz geschildert. Dieser Abschnitt erfüllt das Ziel Z3.

4.2.1 Anforderungen

Die Open Source Lizenz muss folgende Anforderungen erfüllen:

- Kompatibilität mit verwendeten Bibliotheken (*MIT*[3], *Creative Commons Attribution 3.0 Unported*[4], *BSD Licence*[5])
- Problemlose Weiterentwicklung an der Hochschule und von Dritten
- Nutzung und Veränderungen sollen auch in einer Firmenumgebung und kommerziellen Umgebung möglich sein

[3]https://opensource.org/licenses/MIT, aufgerufen am 22.02.2019.
[4]https://creativecommons.org/licenses/by/3.0/, aufgerufen am 28.03.2019.
[5]https://opensource.org/licenses/BSD-3-Clause, aufgerufen am 28.03.2019.

4.2.2 Auswahl

Es existiert eine große Anzahl an Open Source Lizenzen. Um die Auswahl zu beschränken, wurden die drei meistgenutzten Lizenzen ausgewählt. Diese erfüllen auch die Anforderungen aus dem vorherigen Kapitel:

1. MIT[6]
2. GNU General Public License (GPL)[7]
3. Apache[8]

Die *Apache* und *MIT*-Lizenz ähneln sich stark in ihrem Inhalt. Beide erlauben eine Änderung des Codes ohne Veröffentlichung dessen – sie sind also keine Copyleft[9]-Lizenzen. Die *MIT*-Lizenz ist um einiges kürzer und simpler formuliert als die *Apache*-Lizenz, weshalb die *MIT*-Lizenz der *Apache*-Lizenz für dieses Projekt bevorzugt wird.

What the Hack soll eine offene Plattform bleiben. Die *GPL* forciert dies, indem der Quellcode bei Veröffentlichung des Spiels ebenfalls veröffentlicht werden muss (Copyleft). Dies ist bei der *MIT*-Lizenz nicht der Fall. Eine unter *GPL* veröffentlichte Software erlaubt auch eine kommerzielle Nutzung, wie sie in den Anforderungen definiert wurde. Aus diesem Grund wird *What the Hack* unter der *GPL-3.0* veröffentlicht.

4.3 Übersicht

Dieser Abschnitt gibt einen Überblick über die wichtigsten Komponenten und Assets von *What the Hack*. Auf die Komponenten, welche die Erweiterbarkeit zur Verfügung stellen, wird in Abschnitt 4.4 eingegangen.

4.3.1 Quellcode

Der Quellcode und die Assets, ausgenommen des Beispiel-Plugins, befinden sich in einem `git`-Repository[10]. Alle von Unity verwendeten Dateien befinden sich im Ordner `Assets/`. Die Dateien können in mehrere funktionale Kategorien eingeteilt werden:

Main Assembly

Die `Main Assembly` beinhaltet alle Klassen, welche nicht für Plugin-Entwickler zur Verfügung stehen sollen.

Wth.ModApi

In der `Wth.ModApi` sind alle Klassen enthalten, welche zur Erstellung von Plugins, im Folgenden Modifizierungen (Mods), benötigt werden.

[6] https://opensource.org/licenses/MIT, aufgerufen am 22.02.2019.
[7] https://opensource.org/licenses/gpl-3.0, aufgerufen am 22.02.2019.
[8] https://opensource.org/licenses/Apache-2.0, aufgerufen am 22.02.2019.
[9] https://www.gnu.org/licenses/copyleft.de.html, aufgerufen am 22.02.2019.
[10] https://git-scm.com/, aufgerufen am 28.03.2019.

ModTool

`ModTool`[11] ist eine Bibliothek für Unity. Sie ermöglicht es, Spiele durch Szenen, Assets und Code zu erweitern. Dazu dient ein Plugin System, welches in spezifizierten Ordnern nach kompatiblen Mods sucht und sie ggf. lädt. Damit Mod-Entwickler nicht auf sich alleine gestellt sind, bietet die Bibliothek eine Quellcodevalidierung und eine Konflikterkennung. Die Bibliothek unterstützt die Plattformen *Linux*, *macOS*, *Windows* und *Android*.

UnityEnhanced

Bei `UnityEnhanced`[12] handelt es sich um eine Bibliothek für Unity, welche eine State Machine, ein Event System und Erweiterungen zu Unity-Funktionen enthält. Die State Machine und das Event System basieren auf `ScriptableObjects`[13], einem Datencontainerformat von Unity, und werden von *What the Hack* genutzt.

BuildSystem

What the Hack kann von einem GitLab CI-System[14] automatisch getestet und kompiliert werden. Die Datei `.gitlab-ci.yml` enthält die dazugehörige Definition. Der Build-Vorgang wird von der Methode `BuildCommand.PerformBuild` durchgeführt.

Android Plugin

Zur Verwaltung von Addon-Apps dient eine native Android-Bibliothek (auch AAR genannt)[15], welche in Unity als Plugin geladen wird. Die Funktionalität wird genauer in Abschnitt 4.8.3 beschrieben.

Tests

Einige Bereiche des Spiels werden mit Tests getestet. Unity unterscheidet generell zwischen zwei Varianten von Tests:

Mode	Anwendungsfall
Editor Mode	Der *Editor Mode* wird genutzt, um funktionale Bestandteile des Spiels zu testen. In diesen Tests können keine `GameObjects`[16] oder ihr Verhalten in der Szene getestet werden. Dafür sind die Tests in der Durchführung um einiges schneller, als im *Play Mode*.

[11] http://hellomeow.net/modtool/documentation/html/96083abe-78cf-43e9-a3f6-59c4e717b7e7.htm, aufgerufen am 05.03.2019.
[12] https://github.com/hendrik-schulte/UnityEnhanced, aufgerufen am 11.03.2019.
[13] https://docs.unity3d.com/Manual/class-ScriptableObject.html, aufgerufen am 28.03.2019.
[14] https://about.gitlab.com/product/continuous-integration/, aufgerufen am 28.03.2019.
[15] https://developer.android.com/studio/projects/android-library, aufgerufen am 28.03.2019.
[16] https://docs.unity3d.com/ScriptReference/GameObject.html, aufgerufen am 28.03.2019.

Mode	Anwendungsfall
Play Mode	Im *Play Mode* können komplexere Zusammenhänge in einer Szene getestet werden. `GameObjects` können instantiiert und manipuliert werden.

Tabelle 4.2: Testen in Unity

Zusätzlich wird die Bibliothek `NSubstitute`[17] für komplexere Tests verwendet.

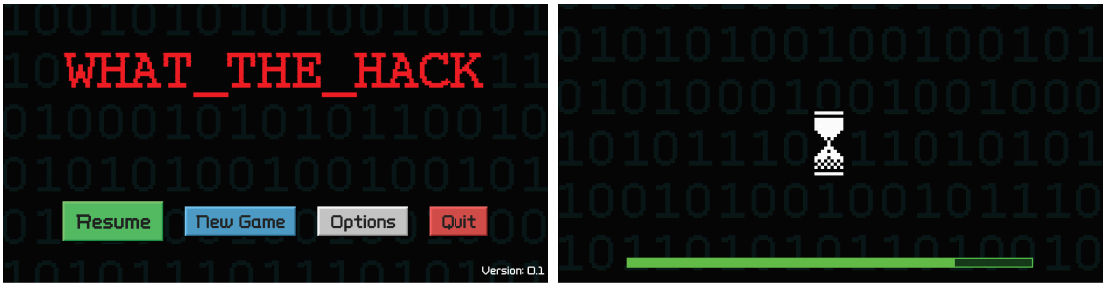
4.3.2 Sprites

Zur Steigerung der Performance werden die Sprites in einem Sprite-Atlas zusammengefasst. Das bedeutet, dass die einzelnen Sprites in einer großen Bilddatei untergebracht werden. Dafür wird das Tool *TexturePacker*[18] verwendet. Das Tool bietet ein Plugin für Unity zur nahtlosen Integration.

Das Spiel nutzt eine dynamische Beleuchtung. Um Sprites, etwa das Spielfeld, einen räumlichen Eindruck zu verschaffen, können *Normal Maps* erstellt werden. Diese beinhalten Informationen über die Ausrichtung der Normalen, welche für die Beleuchtung genutzt werden können.[19] Für einige der Sprites wurden Normal Maps mit dem Tool *SpriteIlluminator*[20] erstellt. Diese werden vom *TexturePacker* in einen separaten Sprite-Atlas gepackt, welcher in Unity mit einem entsprechenden Shader genutzt werden kann.

4.3.3 Unity

Dieser Abschnitt widmet sich der Strukturierung des Spiels aus Sicht der Unity Game Engine.



(a) Die Menü Szene von *What the Hack* (b) Die Lade-Szene von *What the Hack*

Abbildung 4.1: Unity-Szenen des Projekts

[17] https://nsubstitute.github.io/, aufgerufen am 12.03.2019.
[18] https://www.codeandweb.com/texturepacker, aufgerufen am 05.03.2019.
[19] https://docs.unity3d.com/Manual/StandardShaderMaterialParameterNormalMap.html, aufgerufen am 28.03.2019.
[20] https://www.codeandweb.com/spriteilluminator, aufgerufen am 05.03.2019.

Szenen

Das Spiel besteht aus drei Szenen. Die `Menu`-Szene (siehe Abb. 4.1a) wird zum Spielstart angezeigt. Von dort kann der Spieler ein neues Spiel starten, ein Begonnenes fortsetzen und Einstellungen vornehmen.

Hat der Spieler ein Spiel gestartet, gelangt er vorübergehend in die `Loading`-Szene (siehe Abb. 4.1b). Hier sieht der Spieler den Fortschritt des Ladevorgangs. Diese Szene lädt die `MainGame`-Szene asynchron, welche das eigentliche Spiel enthält.

States

Die Menü- und Spielszene besitzen jeweils eine eigene State Machine, welche den aktuellen Zustand der User Interfaces (UIs) darstellen. Für jedes geöffnete Fenster existiert ein State. Wird der State einer State Machine geändert, reagiert die Oberfläche darauf und blendet die zum State gehörenden Elemente ein. So wird vermieden, dass es zu ungewollten UI-Zuständen kommen kann.

Events

Das Event System wird für globale Spielevents verwendet. Dazu gehört etwa das Einstellen oder Entlassen eines Mitarbeiters, das Annehmen oder Abschließen einer Mission, das Fortschreiten der Spielzeit oder die Änderung des Kontostands.

Daten

What the Hack wird mit einem Basisspiel ausgeliefert, welches den Spielinhalt der originalen Implementierung enthält. Dieser Spiel-Content, dazu gehören Missionen, Mitarbeiter, Skills und Namenslisten, wird mithilfe von ScriptableObjects gespeichert. Plugins verwenden die selben ScriptableObject-Klassen, um ihren Content zu speichern.

4.4 Erweiterbarkeit

Dieser Abschnitt beschreibt die Implementierung von Ziel Z5 unter Verwendung des in Abschnitt 3.2 festgelegten Rahmen der Erweiterbarkeit.

Die Erweiterbarkeit des Spiels basiert auf der Bibliothek `ModTool` (siehe Abschnitt 4.3.1). Zu Beginn wird der Aufbau eines Mods aus Sicht der `ModTool`-Bibliothek (siehe Abschnitt 4.4.1) beschrieben, anschließend aus der Sicht von *What the Hack* (siehe Abschnitt 4.4.2). Es folgt eine Beschreibung des Plugin-Mechanismus zum Laden eines Mods in Abschnitt 4.4.3. Mod-Entwicklern wird ein Paket bereitgestellt, welches alle Assets und Klassen enthält, die für die Erstellung eines Mods benötigt werden. Dieser Vorgang wird in Abschnitt 4.4.4 beschrieben. Die Funktionen der Mod-API wird in Abschnitt 4.4.5 erläutert. Dem Mod-Entwickler steht auch eine grafische Oberfläche zur Verfügung, welche in Abschnitt 4.4.6 beschrieben wird. Abschließend folgt in Abschnitt 4.4.7 eine Beschreibung, wie ein Mod für *What the Hack* exportiert werden kann.

4.4.1 Aufbau eines Mods

Ein Mod der `ModTool`-Bibliothek wird als Ordner ausgeliefert und besteht aus drei Bestandteilen.

1. Die Datei `<modname>.info` enthält Informationen wie den Namen, die Version, eine Beschreibung und den Autor eines Mods sowie die Unity Version, welche zur Erstellung des Mods verwendet wurde.

2. Die DLL `<modname>-<version>-Assembly-CSharp.dll` enthält die Klassen, welche von einem Mod definiert werden.

3. Für jede der unterstützen Plattformen wird ein eigenes Asset Paket erstellt, welche in separaten Ordnern untergebracht sind. Dort enthalten sind etwa eigene Sprites oder Sounds.

4.4.2 Bestandteile eines *What the Hack*-Mods

Zentraler Bestandteil eines Mods ist das ScriptableObject `ModInfo`. Es dient als Sammelpunkt für den Inhalt eines Mods. In Tabelle 4.3 sind die einzelnen Bestandteile aufgelistet.

Feld	Erforderlich	Beschreibung
`Id`	✓	Einzigartiger Name des Mods, z. B `com.abc.myWthMod`.
`Scriptable-Object-Dictionary`	✓	Ein Verzeichnis aller verwendeten ScriptableObjects. Siehe Abschnitt 4.6 für eine detaillierte Erklärung.
`Banner`	✓	Ein Banner-Bild, welches in der Mod-Auswahl angezeigt wird (siehe Abb. 4.2).
`SkillSet`	✗	Ein eigenes Skill-Set.
`EmployeeList`	✗	Eine Liste aller Employee-Defintionen.
`MissionList`	✗	Eine Liste aller Missions-Definitionen.
`NameLists`	✗	Listen für eigene Mitarbeiter-, Firmen- und Ortsnamen.

Tabelle 4.3: Bestandteile der ModInfo-Klasse

Abbildung 4.2: Auswahl eines Mods

4.4.3 Plugin-Mechanismus

Wird das Spiel gestartet, durchsucht die Bibliothek `ModTool` festgelegte Ordner nach kompatiblen Mods. Folgende Ordner werden auf den unterschiedlichen Betriebssystemen durchsucht:

Plattform	Mod Pfad
Android	`<Externer Speicher>/Android/data/com.github.geigi.wth/files/Mods/`
Desktop	`<Installationspfad>/Mods/`

Tabelle 4.4: Plattformabhängige Speicherorte für Mods

Gefundene Mods können vor dem Spielstart betrachtet werden (siehe Abb. 4.2). Nur kompatible Mods werden in dieser Übersicht angezeigt. Mods sind dann kompatibel, wenn sie mit derselben Unity Version wie das Hauptspiel gebaut wurden, eine `ModInfo`-Instanz enthalten und die erforderlichen Felder befüllt wurden (siehe Abschnitt 4.4.2). Das Laden eines Mods geschieht, sobald er im Menü ausgewählt wurde.

Der Inhalt eines Mods wird über die Klasse `ContentHub` abgerufen. Für jedes der Bestandteile der `ModInfo` existieren `get`-Methoden, welche überprüfen, ob der Mod das gewünschte Element enthält oder nicht. Enthält ein Mod bspw. keine eigenen Mitarbeiter, werden von der Methode `GetEmployeeLists` die vom Basisspiel definierten Mitarbeiter zurückgegeben.

4.4.4 Bereitstellung eines Mod-SDKs

Die `ModTool`-Bibliothek kann für ein Unity-Projekt ein `Unity Package`[21] bereitstellen, welches alle Assets und Klassen enthält, die für die Erstellung eines Mods benötigt werden (im Folgenden Mod SDK). Diese Funktion ist über das Menü `Tools > ModTool > Create Exporter` erreichbar.

[21] https://docs.unity3d.com/Manual/AssetPackages.html, aufgerufen am 13.03.2019.

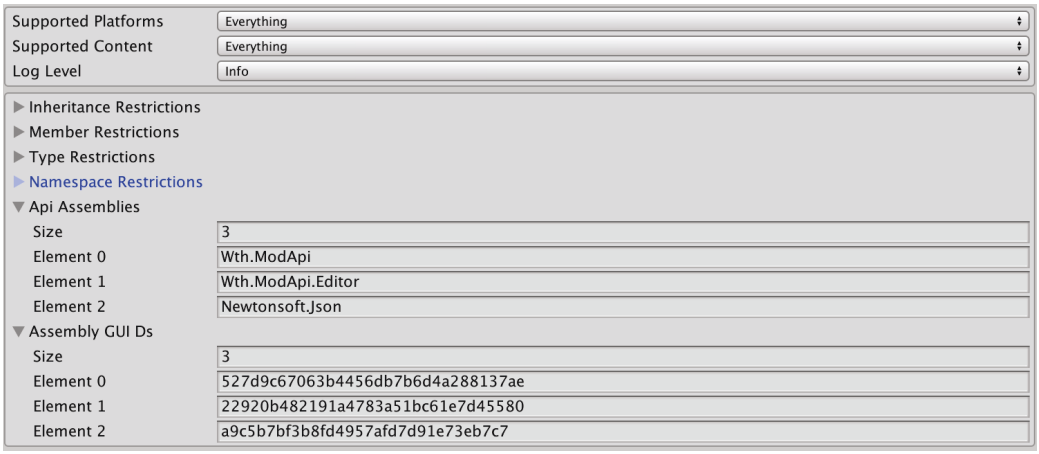


Abbildung 4.3: Einstellungen der Bibliothek `ModTool`

Bevor das SDK exportiert werden kann, müssen einige Einstellungen festgelegt werden. Diese sind über das Menü `Tools > ModTool > Settings` erreichbar. In den Einstellungen können Beschränkungen für Mods festgelegt werden. Außerdem wird festgelegt, welche Assemblies mit in das SDK inkludiert werden sollen (siehe Abb. 4.3). In diesem Fall sind es die Assemblies `Wth.ModApi`, `Wth.ModApi.Editor` und `Newtonsoft.Json`. Diese Assembly-DLLs besitzen einen von Unity fest zugewiesenen Globally Unique Identifier (GUID). Die GUIDs müssen zwischen dem Mod-Projekt und dem Basisspiel konsistent bleiben. Daher wurde die Bibliothek `ModTool` um eine Liste an GUIDs erweitert, welche im Mod-Projekt übernommen werden (siehe Abb. 4.3).

Nach dem erfolgreichen Export des SDKs kann dieses per Drag&Drop zu einem neuen Unity-Projekt hinzugefügt werden.

4.4.5 API

Die API ist in zwei Teile aufgeteilt, welche als separate DLLs kompiliert werden: `Wth.ModApi` und `Wth.ModApi.Editor`. Letztere wird im folgendem Abschnitt 4.4.6 beschrieben. In der `Wth.ModApi` sind alle Klassen enthalten, die benötigt werden, um Inhalte für *What the Hack* zu erstellen. In Tabelle 4.5 werden die wichtigsten Klassen kurz aufgelistet. Jede dieser Klassen beinhaltet Felder für die in Abschnitt 3.2 festgelegten Informationen.

Klasse	Funktion
`EmployeeDefinition`	Definition eines `SpecialEmployee`
`MissionDefinition`	Definition einer Mission
`MissionHook`	Definition einer Missions-Interaktion
`SkillDefinition`	Definition eines Skills
`NameLists`	Definition von verschiedenen Namen, die im Spiel zufällig verwendet werden

Tabelle 4.5: Wichtige Klassen zur Content-Definition

4.4.6 Unity Editor Tools

Ein Mod kann erstellt werden, ohne dass der Entwickler Code schreiben muss. Nur wenn interaktive Missionen erstellt werden, muss der Entwickler ggf. eine Logik für diese Interaktion implementieren.

Für alle weiteren Aufgaben wurden Unity Editor-Fenster[22] implementiert. Diese befinden sich in der DLL `Wth.ModApi.Editor.dll`. Die Fenster können über das Menü `Tools > What_the_Hack ModApi` erreicht werden. Da die Oberflächen alle eine ähnliche Grundfunktion besitzen, leiten sie von einer Basisklasse mit dem Namen `BaseEditor` ab.

Employee Creator

Der *Employee Creator* (siehe Abb. 4.4a) wird genutzt, um eigene Mitarbeiter zu erstellen. Hier werden alle Informationen wie der Name, die Animationen und wann der Mitarbeiter erstellt werden soll, festgelegt. Er besitzt eine direkte Verknüpfung zum Animation Editor (siehe Abschnitt 4.4.6), um dort die Animationen für laufende, untätige und arbeitende Mitarbeiter zu erstellen.

Mission Creator

Missionen werden im *Mission Creator* (siehe Abb. 4.4b) erstellt. Die dort enthaltenen Texte zu Missionen können Platzhalter enthalten, welche im Name Editor (siehe Abschnitt 4.4.6) definiert werden können. Des Weiteren werden hier auch die Erscheinungsbedingungen für die Mission festgelegt. Alle in Abschnitt 3.2.3 definierten Einstellungen können hier vorgenommen werden.

Skill Creator

Skills werden im *Skill Creator* (siehe Abb. 4.4c) erstellt. Alle dort erstellten Skills werden zu einem Skill-Set zusammengefasst. Ein Skill besteht immer aus einem Namen und einem Sprite (siehe auch Abschnitt 3.2.2).

Name Editor

Im *Name Editor* können Listen von Namen erstellt werden, die im Spiel für Mitarbeiternamen oder Platzhalter in Missionen verwendet werden. In Tabelle 4.6 sind die unterstützen Namen und ihre zugehörigen Platzhalter aufgelistet.

Name	Platzhalter
Weiblicher Vorname	`%CONTACT_F%`
Männlicher Vorname	`%CONTACT_M%`
Nachname	`%CONTACT_L%`
Firmenname	`%COMPANY%`
Passwort Applikation	`%PW_APPLICATION%`

[22] https://docs.unity3d.com/ScriptReference/EditorWindow.html, aufgerufen am 28.03.2019.

Name	Platzhalter
Bildungseinrichtung	`%UNIVERSITY%`
Web Service	`%WEBSERVICE%`
Software	`%SOFTWARE%`
Stadt	`%TOWN%`
Land	`%COUNTRY%`
Institution	`%INSTITUTION%`

Tabelle 4.6: Namen und Platzhalter in What the Hack

Animation Editor

Der *Animation Editor* erlaubt die Erstellung einfacher Animationen. Unity bietet diese Funktionalität bereits von Haus aus. Dieser Editor ist einfacher zu bedienen, bietet dafür aber auch weniger Funktionalität als der Animationseditor von Unity. Dafür lässt er sich bspw. in den Employee Creator integrieren.

4.4.7 Exportieren eines Mods

Ein Mod kann über das Menü `Tools > ModTool > Export` exportiert werden.

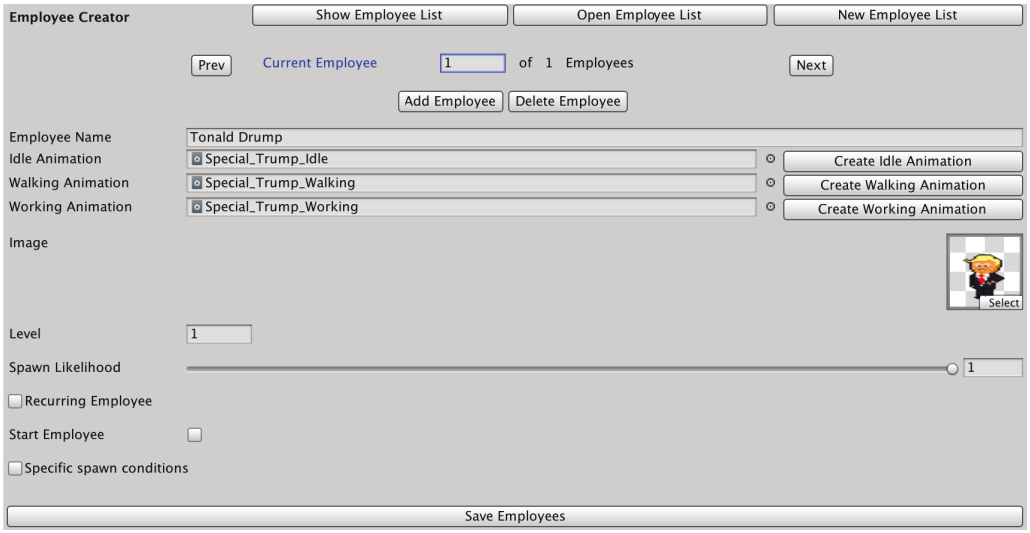
4.5 Spielmechanik

Dieser Abschnitt beschreibt die Umsetzung der in Abschnitt 3.3 festgelegten Funktionsweise für die Spielzeit.

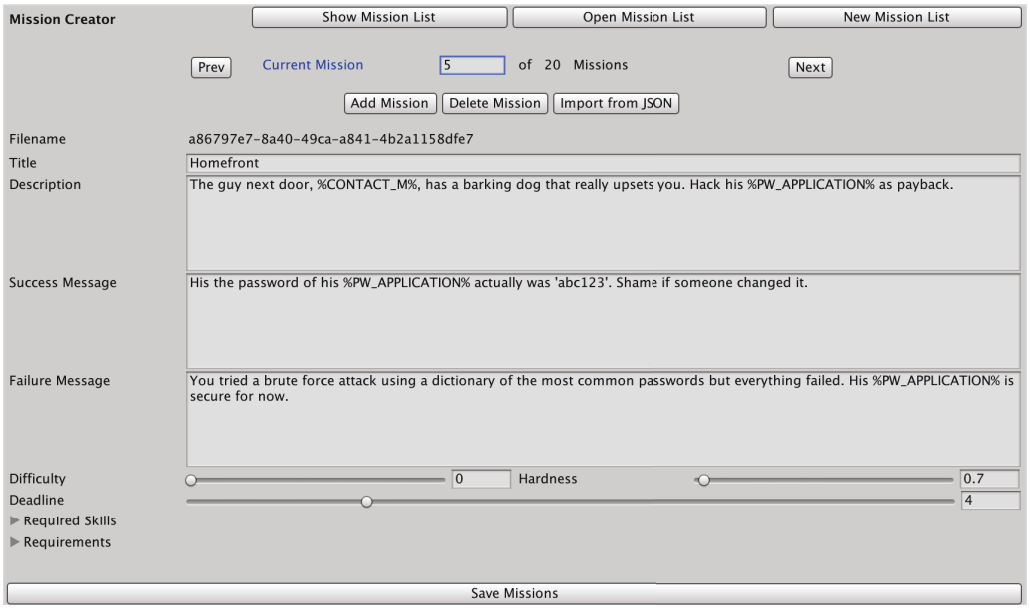
Wie in Abschnitt 4.3.3 beschrieben, wird die Spielzeit von einem Event-System getrieben. Es existieren zwei Events, die ausgelöst werden: zu jedem neuen Zeitschritt eines Spieltages und zu jedem neuen Spieltag. Im klassischen Spielmodus werden diese Events wie in Abschnitt 4.3.3 definiert kontinuierlich ausgelöst. Auch im Echtzeitspielmodus wird dies so gehandhabt, solange das Spiel geöffnet ist. Wird das Spiel nach einer längeren Zeit erneut geöffnet, muss der Spielfortschritt um die entsprechende Zeit nachberechnet werden.

Die Nachberechnung geschieht, indem die entsprechende Anzahl der vergangenen Events nacheinander ausgeführt wird. Wurde das Spiel bspw. eine Stunde nicht geöffnet, werden sechs Zeitschritt-Events ausgelöst. Da die Spielmechanik an diese Zeitschritte angekoppelt ist, wird der Spielfortschritt dementsprechend nachberechnet. Der Vorteil dieser Variante ist es, dass sehr wenig Code geschrieben werden muss, um beide Varianten zu unterstützen. Die Klasse `GameTime` ist für diese Aufgabe zuständig.

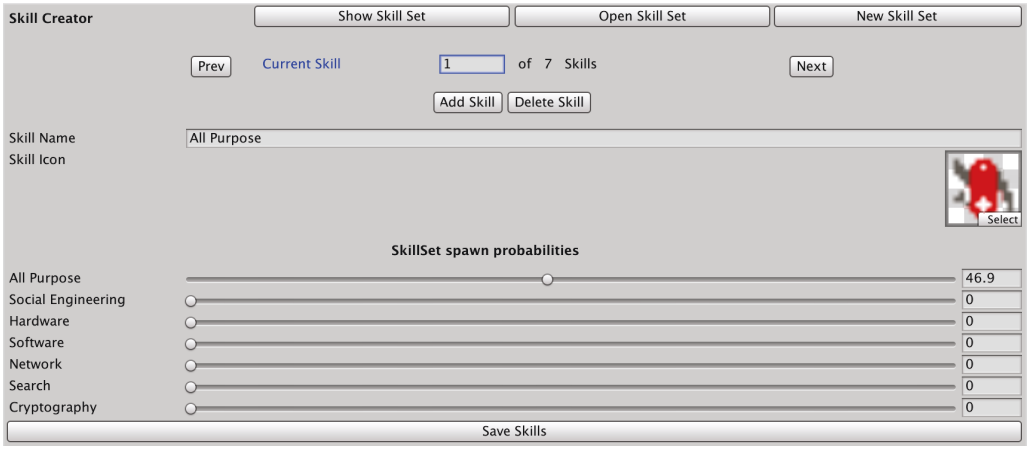
Wird das Spiel mit interaktiven Missionen gespielt, wird die Nachberechnung nicht vollständig durchgeführt. Hat der Spieler eine interaktive Mission angenommen und tritt diese Interaktion auf, wird die Nachberechnung ab diesem Zeitpunkt gestoppt. Wäre das nicht der Fall, hätte der Spieler nach einer längeren Spielpause (wie etwa über Nacht) keine Chance, die Mission erfolgreich abzuschließen.



(a) Der Employee Creator des SDKs



(b) Der Mission Creator des SDKs



(c) Der Skill Creator des SDKs

Abbildung 4.4: Oberflächen zur Erstellung eines *What the Hack*-Mods

4.6 Savegames

Dieser Abschnitt beschreibt kurz den Savegame-Mechanismus des Spiels und geht auf eine Besonderheit bei der Verwendung von ScriptableObjects ein.

In *What the Hack* sind Daten strikt von der Logik getrennt. Daten, welche gespeichert werden müssen, befinden sich in separaten Klassen, die mit dem `Serializable`-Attribut[23] versehen sind.

Die Klasse `SaveGameSystem` beinhaltet die Logik zum Laden und Speichern des Spielstandes. Sie verwendet den .NET `BinarySerializer`[24]. Eine Besonderheit dessen ist, dass er nicht nur die Daten eines Objektes speichert, sondern auch welche Klasse und ggf. welche Spezialisierung einer Klasse diese Daten repräsentieren. Es wird also, im Gegensatz zum `XmlSerializer`, der genaue Typ abgespeichert.

Der `BinarySerializer` kann die meisten Datentypen ohne weitere Konvertierung serialisieren. Manche Datenklassen, wie z. B die `Mission`-Klasse, besitzen Referenzen auf ein ScriptableObject. Im Falle einer Mission ist dies die Klasse `MissionDefinition`. Ein ScriptableObject ist eine Instanz einer Klasse und wird von Unity selbst serialisiert. Der `BinarySerializer` weiß nicht, wie er mit solch einer Instanz umgehen soll. Die Klasse `ScriptableObjectDictionary` löst dieses Problem.

Für jedes ScriptableObject existiert ein Eintrag in diesem Dictionary. Es verknüpft solch ein Objekt mit einer GUID. Der `BinarySerializer` kann diese GUID anstelle des ScriptableObjects serialisieren. Bei der Deserialisierung wird das zur GUID gehörende Objekt zurückgegeben.

Da Mods eigene ScriptableObjects definieren können, besitzt jeder Mod ein eigenes `ScriptableObjectDictionary`. Die Klasse `ScriptableObjectManager` wird genutzt, um auf alle Dictionaries zuzugreifen.

Der `BinarySerializer` wurde mit einem `ISerializationSurrogate` erweitert, damit er dieses Dictionary nutzen kann (siehe Listing 4.1). Diese Klasse heißt `ScriptableObjectSerializationSurrogate` und besitzt zwei Methoden: `GetObjectData` zur Serialisierung (Zeile 1) und `SetObjectData` (Zeile 7) zur Deserialisierung.

```
1 public void GetObjectData(object obj, SerializationInfo
    info, StreamingContext context)
2 {
3     var so = (ScriptableObject) obj;
4     info.AddValue("key",
        ScriptableObjectManager.Instance.GetKey(so));
5 }
6
7 public object SetObjectData(object obj,
    SerializationInfo info, StreamingContext context,
    ISurrogateSelector selector)
```

[23] https://docs.microsoft.com/de-de/dotnet/api/system.serializableattribute?view=netframework-4.7.2, aufgerufen am 28.03.2019.

[24] https://docs.microsoft.com/de-de/dotnet/standard/serialization/binary-serialization, aufgerufen am 28.03.2019.

```
8  {
9      return ScriptableObjectManager.Instance
           .GetObject((string)info.GetValue("key",
           typeof(string)));
10 }
```

Listing 4.1: Serialisierung und Deserialisierung von ScriptableObjects

4.7 iOS

What the Hack unterstützt generell iOS, da Unity auch iOS als Plattform unterstützt. Die Bibliothek `ModTool` unterstützt iOS allerdings nicht. Apple verbietet es, dass eine App externen Code ausführt.[25] Daher unterstützt auch Unity unter iOS dies, im Gegensatz zu den anderen Plattformen, nicht. Soll *What the Hack* also mit speziellem Inhalt unter iOS Geräten ausgeliefert werden, muss dieser in die Basisapp integriert und neu kompiliert werden.

4.8 Android

Nur wenige native Android-Features können direkt von Unity genutzt werden. Aufgaben, wie etwa das Erstellen einer Benachrichtigung, müssen von Java-Code aus erledigt werden. Unity besitzt aber die Möglichkeit, Java-Klassen von speziellen Java-Plugins aufzurufen. Der Abschnitt beschreibt, wie diese Integration umgesetzt ist.

Android wird auch von der Bibliothek `ModTool` unterstützt. Im folgenden Abschnitt wird die Kommunikation zwischen den Unity-Komponenten und nativen Android-Schnittstellen beschrieben. Um Mods einfach installieren zu können, wird anschließend in Abschnitt 4.8.3 beschrieben, wie Mods als separate Apps installiert werden können.

4.8.1 Kommunikation zwischen Unity und Android

Als Schnittstelle zwischen Android und Unity dient ein Java-Projekt mit dem Namen `AndroidUnityBridge`, welches als AAR Bibliothek kompiliert wird. In Unity können die Klassen dieser Bibliothek genutzt werden. Die abstrakte Klasse `AndroidPluginInteraction` dient als Wrapper dafür.

4.8.2 Benachrichtigungen

Dieser Abschnitt beschreibt die Umsetzung der in Abschnitt 3.3 festgelegten Funktionsweise für die Benachrichtigungen.

Im Echtzeitspielmodus von *What the Hack* soll das Spiel auch im Hintergrund weiter laufen und den Spieler benachrichtigen, sobald wichtige Ereignisse im Spiel

[25] https://download.developer.apple.com/Documentation/License_Agreements__Apple_Developer_Program/Apple_Developer_Program_License_Agreement_20181019.pdf, siehe Abschnitt 3.3.2, aufgerufen am 27.03.2019.

vorgefallen sind oder seine Interaktion gefragt ist. Folgende Benachrichtigungen sind im Spiel implementiert:

- Eine Mission wurde abgeschlossen.
- Eine Mission erfordert die Interaktion des Nutzers.
- Der Zahltag der Mitarbeiter steht bevor.
- Kein Mitarbeiter arbeitet mehr.

Weitere Benachrichtigungen sind denkbar und sollen einfach implementiert werden können. Dieses System kann auf verschiedene Art und Weise implementiert werden, wovon hier zwei vorgestellt werden.

Benachrichtigungen mit einem Timer

Unter Android können Benachrichtigungen mit einem Erscheinungsdatum erstellt werden. Diese Benachrichtigungen werden auch angezeigt, wenn die App, welche sie erstellt hat, bereits geschlossen ist. Dieses Feature kann für *What the Hack* genutzt werden. Da die Spielmechanik nur zu einem kleinen Teil zufällig berechnet wird, kann mit großer Genauigkeit bestimmt werden, wann eine Mission abgeschlossen wird und ob sie erfolgreich abgeschlossen wird oder nicht. Wird *What the Hack* geschlossen, können diese Zeitpunkte berechnet und die Benachrichtigungen erstellt werden. Startet der Spieler das Spiel erneut, können die eingereihten Benachrichtigungen abgebrochen und gelöscht werden.

Ausgelagerte Spielmechanik

Eine weitere Möglichkeit ist es, die Spielmechanik anstatt in C# in Java oder Kotlin zu implementieren. Die Spielmechanik kann dann als Android Service im Hintergrund laufen, selbst wenn die App geschlossen ist.

Vergleich und Fazit

In Tabelle 4.7 werden die beiden Varianten auf ihre Vor- und Nachteile verglichen.

	Benachrichtigungen mit Timer	**Ausgelagerte Spielmechanik**
Vorteile	• Kein Akkuverbrauch • Simple Implementierung, wenig Fehleranfälligkeit • Die Spielmechanik muss nicht künstlich vom Spiel getrennt werden. • Zuverlässig, da erstellte Benachrichtigungen in jedem Fall gezeigt werden.	• Höhere zeitliche und inhaltliche Genauigkeit möglich. • Größere Vielfalt an Benachrichtigungen möglich.

	Benachrichtigungen mit Timer	**Ausgelagerte Spielmechanik**
Nachteile	• Durch den kleinen Faktor an Zufälligkeit in einer Mission kann es passieren, dass eine Benachrichtigung zu früh oder zu spät angezeigt wird. • Jede Benachrichtigung muss vorhersehbar sein, da ihr Erscheinungszeitpunkt berechnet werden muss.	• Unklar, ob die in Java geschriebene Spielmechanik auf jeder Plattform mit geringem Aufwand lauffähig ist. • Große Schnittstelle zwischen Java und Unity, bringt Komplexität und ist Fehleranfällig (da etwa Klassennamen als String aufgerufen werden). • Durch die Zentralität der Spielmechanik müssen weitere Komponenten, wie bspw. das SaveGame-System, in Java/Kotlin implementiert werden. • Die Performance leidet, wenn Java-Code aus Unity aufgerufen werden muss. • Höherer Akkuverbrauch, da ein Service dauerhaft im Hintergrund laufen muss.

Tabelle 4.7: Vergleich der Systeme zur Implementierung von Benachrichtigungen

Die Nachteile der *Ausgelagerten Spielmechanik* stehen nicht im Verhältnis zu den geringen Vorteilen, weshalb die *Benachrichtigungen mit Timer* als System in *What the Hack* implementiert sind. Für die Benachrichtigungen wird das Unity-Paket `Mobile Notifications` genutzt, welches von Unity selbst entwickelt wird. Es unterstützt neben *Android*- auch *iOS*-Geräte.

4.8.3 Mod APKs

Mods sollen vom Nutzer möglichst einfach installiert werden können. Eine Möglichkeit, die dem Nutzer immer zur Verfügung steht, ist die manuelle Installation. Dafür muss der Mod lediglich in einen bestimmten Ordner kopiert werden (siehe Tabelle 4.4). Dies erfordert allerdings einen Dateimanager und ist nicht für jeden Nutzer eine akzeptable Lösung. Folgende Voraussetzungen sind gegeben:

- Kompatibilität mit *Google Play Store*
- Simpler Erstellungsprozess, Distribution & Installation
- Verzicht auf einen Content-Server

Die Auslieferung von Mods als separate Android App erfüllt alle Voraussetzungen. Im Folgenden wird beschrieben, wie dieses System umgesetzt ist.

Jede Android-App besitzt die Möglichkeit, alle auf dem Smartphone installierten Apps aufzulisten. Mit diesen Informationen ist es aber noch nicht möglich, zu erkennen, welche App eine kompatible Mod-App ist. Dafür kann ein Android `Content Provider`[26] genutzt werden.

Es gibt verschiedene `ContentProvider` für verschiedene Arten von Daten. Die Bibliothek `ModTool` unterstützt allerdings nur das Laden von Mods aus dem Dateisystem; `ContentProvider` werden nicht unterstützt. Daher implementiert eine Mod-App zwar einen `ContentProvider`, dieser wird aber nur genutzt, um die App als *What the Hack* Mod-App auszuweisen. Das Basisspiel sucht dabei nach Providern mit dem Klassennamen `WTHModProvider`.

Die eigentlichen Mod-Daten befinden sich im `assets`-Ordner der Mod-App. Über den Android-`PackageManager` kann auf diese Assets zugegriffen werden:

```
1 getPackageManager()
    .getResourcesForApplication(info.applicationInfo)
    .getAssets();
```

Listing 4.2: Abrufen der Assets einer Android App

Die Asset-Daten werden dann in den spezifizierten Mod-Ordner auf dem externen Speicher installiert. Zuständig für die Installation und Deinstallation von Mods ist die Java-Klasse `ModAppManager`. Der gleichnamige C#-Wrapper stellt diese Funktionalität dann in Unity zur Verfügung.

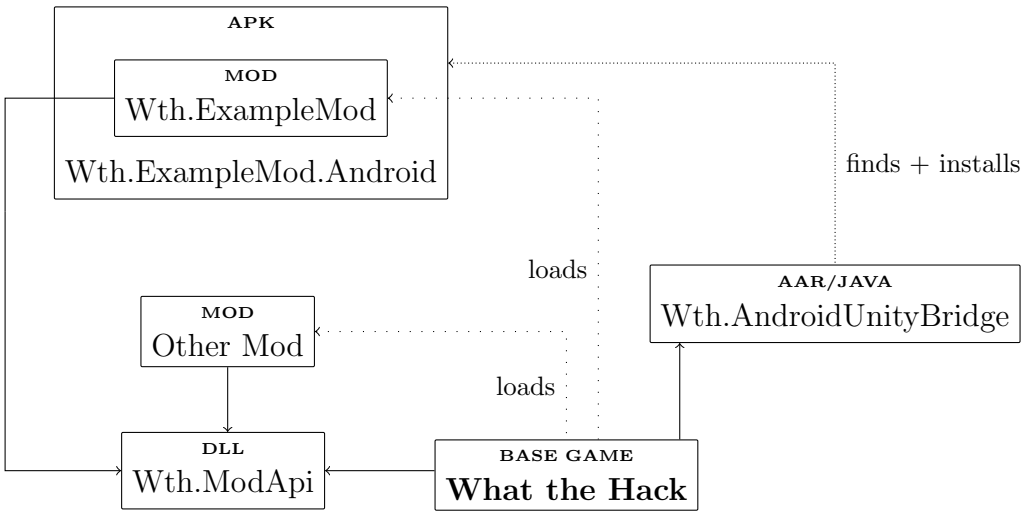


Abbildung 4.5: Übersicht des Plugin-Mechanismus

In Abb. 4.5 ist eine Übersicht dieses Mechanismus abgebildet. *Other Mod* repräsentiert einen von Nutzer manuell installieren Mod, während *Wth.ExampleMod* in einer APK Datei verpackt ist. Diese APK muss vom Nutzer installiert werden. Anschließend kann der Mod vom `ModAppManager` in der `Wth.AndroidUnityBridge`

[26] https://developer.android.com/guide/topics/providers/content-providers, aufgerufen am 28.03.2019.

erkannt und installiert werden. Ein genauer Ablauf der Installation und Deinstallation ist im Anhang in Kapitel B dargestellt. Die dort abgebildeten Prozesse werden jedes Mal, wenn das Spiel gestartet wird, ausgeführt.

4.9 Entwicklung eines Beispiel-Plugins

Dieser Abschnitt beschreibt den Lehrinhalt des Beispiel-Plugins und erfüllt Ziel Z8.

Das Beispiel-Plugin behandelt die *Java Cryptography Extension*. Diese API enthält viele Funktionen, um kryptografische Aufgaben, wie die symmetrische Verschlüsselung einer Datei, durchzuführen. Die API ist nicht besonders einsteigerfreundlich und anfällig für eine falsche Verwendung durch Entwickler. Aus diesem Grund wurde von Nadi, Krüger, Mezini und Bodden im Jahr 2016 untersucht, welche Gründe dies hat und welche Fehler am häufigsten auftreten. Die Ergebnisse dieser Arbeit wurden teilweise für den Lehrinhalt des Spiels übernommen. Eine weitere Quelle für Lehrinhalt bietet das Buch *Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs* von Long, Mohindra, Seacord, Sutherland und Svoboda, 2013, welches viele Best-Practices zur Nutzung der API enthält.

Im Plugin enthalten sind ein eigenes Set an Fähigkeiten sowie eigene Missionen, von denen die meisten mit einer Interaktion implementiert wurden. Die Interaktionen sind dabei quizartig aufgebaut. Der Nutzer muss entscheiden, wie spezifische Aufgaben mit der API korrekt durchgeführt werden. In Abb. 4.6 ist eine dieser Interaktionen abgebildet.

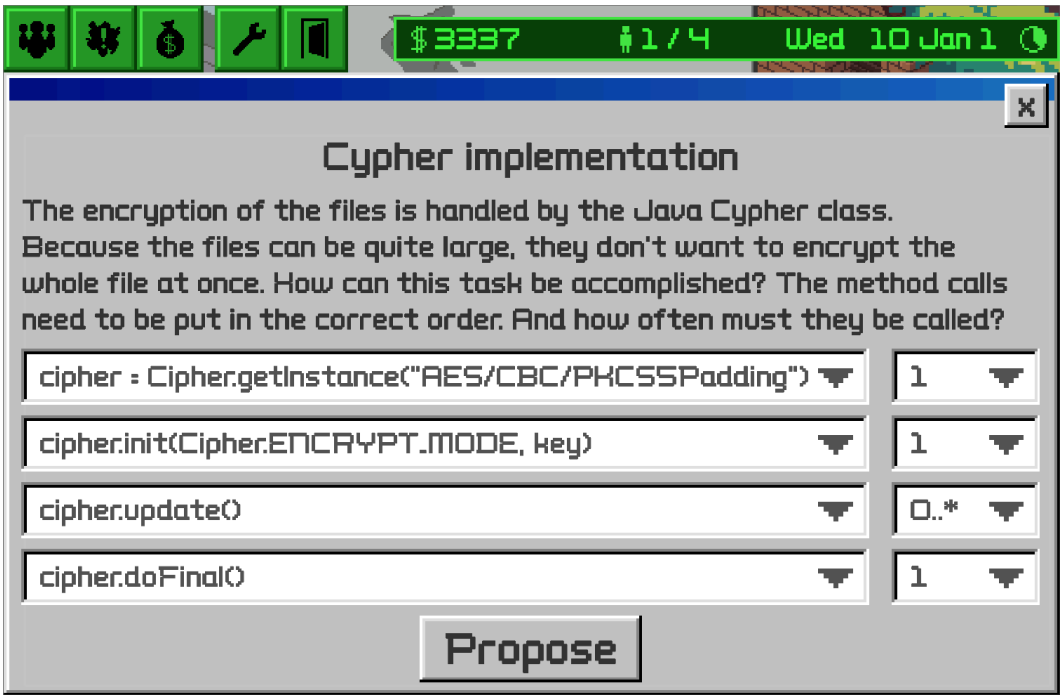


Abbildung 4.6: Interaktion einer Mission des Java Cryptography Extension Plugins

Kapitel 5

Analyse

Im Folgenden wird untersucht, ob die Neuimplementierung die Anforderungen aus Abschnitt 2.6.3 erfüllt. Hierdurch wird überprüft, inwiefern sich die Neuerungen der Neuimplementierung auf die Spielqualität und den Lernerfolg auswirken.

Das Kapitel selbst erfüllt das Ziel Z9.

Anforderung A1: Entwurf sinnvoller und langfristiger Ziele für What the Hack

Langfristige Spielziele bieten für viele Spieler eine Motivation, ein Spiel erneut zu spielen. Dadurch tragen sie auch indirekt zum Lernerfolg bei. Denn je öfter und intensiver das Spiel gespielt wird, desto mehr lernt der Spieler auch.

What the Hack bietet eine gute Basis, um verschiedenste Ideen im Spiel umzusetzen. In Abschnitt 3.5 wurden mehrere Ideen für solche Ziele entworfen. Aus Zeitgründen konnten die in Abschnitt 3.5 vorgeschlagenen Erweiterungen der Spielmechanik nicht implementiert werden. Allerdings muss ein Spiel auch nicht zwingend ein festes Spielziel und Spielende besitzen, um motivierend zu sein. Um das zu zeigen, wird erneut die *Anno* Spielreihe als Beispiel herangezogen. Hier baut der Spieler in einer sehr freien Welt Städte, ohne ein festes Ziel zu haben. Es gibt viele mittelfristige Ziele, die der Spieler annimmt, wie etwa die nächste Bevölkerungsstufe zu erreichen, Aufträge zu erfüllen oder eine weitere Insel zu besiedeln.

What the Hack unterstützt sogenannte „Story-Missionen“. Mit diesen kann hier etwas Ähnliches wie bei Anno geschaffen werden. Durch eine Reihe an Missionen, die aufeinander aufbauen, entwickelt sich eine Geschichte, auf dessen Ende der Spieler gespannt sein kann. Durch diese Steigerung wird Motivation geschaffen, das Spiel weiter zu spielen und die Story-Missionsreihe erfolgreich zu beenden. Solche Missionen bieten auch einen roten Faden, der sich durch das Spiel zieht. Das kann sich auch positiv auf den Lernerfolg auswirken. Durch die Story-Missionen kann bestimmt werden, welchen Lehrinhalt der Spieler in jedem Fall bearbeiten wird. Ohne Story-Missionen ist dies nicht zwingend gegeben, da die Missionen zufällig generiert werden. Ebenfalls kann dem Spieler dank dieser Missionen auch mitgeteilt werden, wann er das Spiel „durchgespielt“ hat, nämlich genau dann, wenn er die letzte Mission dieser Story-Missionen erreicht hat. Für Erweiterungsentwickler ist es also empfehlenswert, dieses Feature von *What the Hack* zu nutzen.

Die Anforderung A1 wurde somit erfüllt.

Anforderung A2: Implementierung von spielerischem Feedback beim Fehlschlag einer Mission

Sofortiges Feedback ist eines der Stärken von Serious Games. Diese Stärke wurde schon von der originalen Implementierung genutzt, um Lehrinhalte beim erfolgreichen oder fehlgeschlagenen Abschluss einer Mission zu vermitteln. Dieses Feedback war bisher rein inhaltlich und hat dem Spieler nicht mitgeteilt, wie er eine fehlgeschlagene Mission erfolgreich hätte abschließen können. Im Falle einer normalen Mission reduziert sich das spielerische Feedback darauf, dem Spieler mitzuteilen, welcher Skill einer Mission nicht erfüllt wurde. Bei einer interaktiven Mission erweitert sich dieses Feedback auch auf einen inhaltlichen Aspekt, da der Spieler beim Fehlschlag eine falsche Entscheidung getroffen haben kann.

Das spielerische Feedback ist vor allem für Spieler hilfreich und motivierend, die das Spiel zum ersten Mal spielen. Es erklärt, wie das Spiel funktioniert und wie sich der Spieler verbessern kann. Dies wirkt sich indirekt positiv auf die Lehreffizienz aus, da der Spieler schneller in das Spiel hereinkommt und somit auch mehr Spaß am Spiel haben kann.

Die Anforderung A2 wurde somit erfüllt.

Anforderung A3: Implementierung einer Schnittstelle für interaktive Missionen

Die Originalimplementierung von *What the Hack* kann Inhalte nur durch die Missionstexte vermitteln. Dies kann für manche Inhalte eine gute Methode sein. Auch können Methoden wie Humor genutzt werden, um den Spielspaß an diesen Texten zu steigern. Manche Inhalte können aber besser vermittelt werden, wenn der Nutzer auch interaktiv etwas entscheiden muss. Gleichzeitig wird der Nutzer dadurch auch gezwungen, den Text zu lesen und sich Gedanken zum Thema zu machen. Das steigert die Lerneffektivität.

Die implementierte Schnittstelle erlaubt eine sehr freie Gestaltung von interaktiven Missionen. Sie können mehrere Interaktionen beinhalten und es kann bestimmt werden, wann und in welcher Reihenfolge sie auftreten. Es gibt nur die Voraussetzung, dass jede Interaktion entweder erfolgreich oder nicht erfolgreich abgeschlossen wird. Diese Freiheit erlaubt es, dass kreative und auf das Thema zugeschnittene Missionen erstellt werden können.

Die Anforderung A3 wurde somit erfüllt.

Anforderung A4: Unterstützung mehrerer Schwierigkeitsstufen

Bei der großen Komplexität von technischen Themen ist es wichtig, jeden Wissensstand erreichen und fortbilden zu können. Durch verschiedene Schwierigkeitsstufen ist dies möglich. Die Schwierigkeitsstufen unterscheiden sich in *What the Hack* nur inhaltlich. So können auch Spieler, die bereits einen höheren Wissensstand besitzen, mit passendem Inhalt in das Spiel einsteigen.

Die Schwierigkeitsstufen steigern den Spielspaß, da für verschiedene Wissensstände eine passende Herausforderung gegeben ist. Gleichzeitig erweitert dies auch die geeignete Zielgruppe des Spiels.

Die Anforderung A4 wurde somit erfüllt.

Anforderung A5: Implementierung von Schnittstellen, einem Plugin-Lade-Mechanismus, eines SDKs und einer Dokumentation zur Erweiterung des Spiels

Durch die Unterstützung von Erweiterungen kann die Neuimplementierung als eine an eine technische Zielgruppe gerichtete Serious Gaming Plattform gesehen werden. Der mögliche Einsatzbereich erweitert sich dadurch enorm. Das Spielprinzip bleibt bei *What the Hack* aber immer dasselbe. Durch die Unterstützung von interaktiven Missionen ist eine weitere Spezialisierung möglich.

Das Erstellen von Erweiterungen ist dabei simpel gestaltet. Durch die Implementierung eines SDKs mit grafischer Oberfläche können, ohne Programmierkenntnisse, Erweiterungen mit einem bestimmten Lehrinhalt erstellt werden. Gleichzeitig bietet *What the Hack* aber auch die Möglichkeit, kreative und komplexere Missionen dynamisch zu laden. Dies wirkt sich nicht direkt auf die Spielbarkeit oder die Lerneffektivität aus. Allerdings ist es ein attraktives Feature für Unternehmen oder Institutionen, die das Spiel einsetzen wollen. Die Distribution der Erweiterungen ist unproblematisch. Erweiterungen können entweder durch das Kopieren in einen bestimmten Ordner oder auf Android durch die Installation einer Addon-App hinzugefügt werden.

Die Anforderung A5 wurde somit erfüllt.

Spielmodi

Das Feature des klassischen und Echtzeitspielmodus ist nicht in den Anforderungen enthalten, ist aber auf das Ziel Z4 zurückzuführen.

Sicherheitsbewusstsein zu schaffen bedeutet, das Verhalten einer Person zu ändern. Dies wird durch ein häufiges Spielen begünstigt. Der Ansatz des Echtzeitspielmodus bietet eine interessante Idee den Lernerfolg zu optimieren. Inwiefern dieser Ansatz den Spielspaß und Lernerfolg beeinflusst, kann ohne Evaluation nicht eindeutig gesagt werden. Dadurch, dass *What the Hack* diesen und einen klassischen Spielmodus bietet, ist eine Evaluationsmöglichkeit gegeben. Der Lernerfolg kann bspw. mit zwei Probandengruppen getestet werden.

Kapitel 6

Schlussfolgerung

Dieses Kapitel widmet sich einem Fazit zu dieser Arbeit und gibt einen Ausblick auf die Zukunft von *What the Hack*.

6.1 Fazit

Zusammenfassend kann man sagen, dass *What the Hack* von einem einfachen Serious Game zu einer Serious Game-Plattform für Security Awareness gewachsen ist. Durch die Unterstützung von Schwierigkeitsstufen und Erweiterungen kann eine breite Zielgruppe angesprochen werden. Gleichzeitig erlaubt die Erweiterbarkeit auch eine Spezialisierung auf Themen, Firmen und Umgebungen. Dies ist gerade bei Security Awareness und der Komplexität und Vielfalt des Umfeldes wichtig, wenn man auch fortgeschrittenere Themen durch ein Serious Game lehren möchte. Inhalte zu vermitteln muss und darf nicht immer ein trockener Vorgang sein. Spiele können motivieren und bieten die Chance, effizienter Inhalte zu vermitteln.

Eine ausführliche Literaturrecherche zum Thema Serious Gaming zu Beginn des Projekts hat eine theoretische Grundlage für das Design und die Entwicklung von *What the Hack* geschaffen. Kriterien für erfolgreiche und lehrreiche Serious Games wurden herausgearbeitet und die originale Implementierung daraufhin überprüft. Die aus dieser Untersuchung entstandenen Anforderungen wurden in die Neuimplementierung übernommen und umgesetzt. Durch diese theoretische Untersuchung ist ein erster Schritt für eine zukünftige Evaluation getan.

Die Idee des Echtzeitspielmodus konnte nicht durch Fachliteratur bestätigt werden. Sie erscheint mit einem psychologischen Hintergrund sinnvoll, allerdings wurde eine solche Spielweise weder untersucht noch evaluiert. Durch die Unterstützung des klassischen und des Echtzeitspielmodus ist eine zukünftige Evaluation möglich. Ob ein mehrfaches, verteiltes Spielen an einem Tag tatsächlich effektiver bzw. erfolgversprechender als ein klassisches Spiel ist, bleibt offen.

Das Spiel neu zu implementieren war die richtige Entscheidung. Eine Erweiterung der originalen Implementierung wäre durchaus möglich gewesen, allerdings konnte dank der Neuimplementierung die Softwarearchitektur im Hinblick auf die Unterstützung von Erweiterungen gestaltet werden. Mit Unity als Game Engine steht zukünftigen Entwicklern und Erweiterungsentwicklern nun ein mächtiges und modernes Tool zur Verfügung. Unity erlaubt nicht nur einen Teil der Entwicklung, wie etwa der grafischen Elemente, in einem Editor vorzunehmen, er kann auch

um eigene Funktionen erweitert werden. Diese Funktionalität wurde genutzt, um Oberflächen für die Erweiterungsentwicklung zu erstellen. Somit können Erweiterungen erstellt werden, ohne eine Zeile Code zu schreiben.

Die Untersuchung des Markts zu Serious Games mit dem Thema Security Awareness hat gezeigt, dass ein Mangel an spezialisierten und adaptiven Serious Games herrscht. Mit *What the Hack* ist der Markt nun um solch ein Spiel gewachsen. Da das Spiel unter einer Open Source Lizenz veröffentlicht ist, ist der Weg für eine Weiterentwicklung geebnet.

6.2 Ausblick

Aktuell ist nicht davon auszugehen, dass IT-Sicherheit und Security Awareness weniger wichtige Themen werden. Ganz im Gegenteil: im Zuge der fortschreitenden Digitalisierung werden sie mehr an Relevanz gewinnen. Eine dahingehende Schulung wird daher auch in Zukunft ein wichtiges Thema bleiben und werden. Neue und alternative Methoden zur Vermittlung dieser Inhalte sind darum interessant und wichtig.

Spiele wie *Operation Digital Chameleon*[1] haben gezeigt, dass Serious Games zum Thema IT-Sicherheit in einem professionellen Umfeld funktionieren. Es existiert aber kein „Kochrezept“, um ein erfolgreiches Serious Game zu entwickeln. Der Erfolg und die Effektivität des Einsatzes von Serious Games ist nicht eindeutig geklärt. Es zeigt sich auch schwierig, dies allgemein zu zeigen. Da Spiele sehr unterschiedlich gestaltet sein können, wirkt sich die Qualität des Spiels maßgeblich auf den Erfolg als Serious Game aus. Aus diesem Grund ist eine zukünftige Evaluation von *What the Hack* in Anbetracht auf den Lernerfolg wünschenswert. Ohne eine aussagekräftige Evaluation ist es schwierig, Aussagen zum Lernerfolg zu treffen. Die Idee des Echtzeitspielmodus ist interessant und scheint neu in dem Feld zu sein. Auch hier wäre eine zukünftige Evaluation sehr interessant.

What the Hack selbst ist ein vollständiges Spiel. In dieser Arbeit wurden bereits Ideen aufgeführt, um das Spiel zu erweitern. Das Spielprinzip ist recht simpel, erlaubt aber einen kreativen Ausbau. Neue Spielelemente wie zusätzliche Arbeitsplätze (z. B ein Hardware-Arbeitsplatz), Items mit spielerischem Einfluss (z. B eine Kaffeemaschine) oder Mitarbeiter, welche Urlaub benötigen, krank werden und sich fortbilden können, sind Ideen, um den Spielspaß und die Motivation des Spielers zu steigern. Das Spiel auf diese Art und Weise zu erweitern ist wünschenswert, da es dem Spieler mehr Abwechslung bietet und das Spiel länger seinen Reiz behält.

Auf zukünftige Erweiterungen zu verschiedenen Themen der IT-Sicherheit kann man gespannt sein. Die implementierte Beispiel-Erweiterung bietet einen Leitfaden und zeigt, was mit dem Spiel möglich ist. Ein nützliches Feature wäre ein Shop, in dem Erweiterungen veröffentlicht und heruntergeladen werden können. So hätten Nutzer eine zentrale Anlaufstelle, um verfügbare Erweiterungen zu finden und zu installieren. Durch die Unterstützung des Google Play Store ist dies im weiten Sinne auf Android bereits gegeben.

Mobile Geräte von *Apple*, wie das *iPhone* oder *iPad*, sind in Unternehmen eine beliebte Gerätewahl. Dank Unity unterstützt *What the Hack* diese Plattform.

[1] https://dl.acm.org/citation.cfm?id=2957800, aufgerufen am 19.02.2019

Apple verbietet allerdings einen Plugin-Mechanismus mit Code-Ausführung, wie *What the Hack* ihn nutzt. Erstrebenswert wäre eine Anleitung von Start bis Ende, um *What the Hack* mit bestimmten Erweiterungen für *iOS* selbst zu kompilieren und auszuliefern. Einen Schritt weiter würde eine Unterstützung von Erweiterungen ohne Code-Ausführung unter *iOS* gehen. Dies würde eine Distribution von *What the Hack* und Erweiterungen im *Apple App Store* ermöglichen.

What the Hack wurde bereits von mehreren Beta-Testern getestet. Eine größere Testrunde mit einer darauf folgenden Optimierung der Spielmechanik wäre wünschenswert. Dabei handelt es sich um Feinheiten, die aber für manche Spieler den Grad zwischen einem erfreulichen und einem nicht erfreulichen Erlebnis ausmachen.

Anhang A

Serious Games zum Thema IT-Sicherheit

Name	Medium	Ausprägung	Zielgruppe	Inhalt	Schwierigkeit	Schwierigkeitsgrade	Szenarien	Betreutes Spielen
Control-Alt-Hack[1]	Analog	Kartenspiel	Allgemeinheit	Social Engineering, White Hat Hacking	Anfänger	Nein	Nein	Nein
CyberCIEGE[2]	Desktop	Construction and management simulation (3D)	IT Admins & Studenten	Network security	Anfänger bis Fortgeschrittene	Ja	Ja	Nein
CyPhinx[3]	Web	3D OpenWorld Role-play	IT Security Professionals	Information Security, Forensics, Network Security	Fortgeschrittene	Nein	Ja	Nein
d0x3d![4]	Analog	Kartenspiel	Keine besondere Zielgruppe	Network security	Anfänger	Nein	Nein	Nein
Data Detectives[5]	Web	Frage-Antwort	Studenten, Universitätsmitarbeiter	Document Classification	Anfänger	Nein	Nein	Nein

[1]http://www.controlalthack.com, aufgerufen am 29.11.2018
[2]https://my.nps.edu/web/c3o/cyberciege, aufgerufen am 29.11.2018
[3]https://www.cybersecuritychallenge.org.uk/competitions/play-demand-cyphinx, aufgerufen am 29.11.2018
[4]https://www.thegamecrafter.com/games/-d0x3d-, aufgerufen am 29.11.2018
[5]https://www.adelaide.edu.au/technology/secureit/games/data_detectives/story.html, aufgerufen am 29.11.2018

Name	Medium	Ausprägung	Zielgruppe	Inhalt	Schwierigkeit	Schwierigkeitsgrade	Szenarien	Betreutes Spielen
Device Defenders[6]	Web	Frage-Antwort	Studenten, Universitätsmitarbeiter	Device Protection	Anfänger	Nein	Nein	Nein
Dodgy WiFi[7]	Web	Frage-Antwort	Studenten, Universitätsmitarbeiter	Video Downloading	Anfänger	Nein	Nein	Nein
Elevation of Privilege (EoP)[8]	Analog	Kartenspiel	Entwickler (Microsoft SDL)	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege	Anfänger	Nein	Nein	Nein
Enter - IT Security Game[9]	Smartphone / Tablet	Puzzle	Mitarbeiter	Social Engineering, IT Security	Anfänger	Nein	Nein	Nein
Football Fever[10]	Web	Frage-Antwort	Studenten	Phishing	Anfänger	Nein	Nein	Nein

[6]https://www.adelaide.edu.au/technology/secureit/games/device_defenders/story.html, aufgerufen am 29.11.2018
[7]https://www.adelaide.edu.au/technology/secureit/games/dodgy_wifi/story.html, aufgerufen am 29.11.2018
[8]https://www.microsoft.com/en-us/SDL/adopt/eop.aspx, aufgerufen am 29.11.2018
[9]https://entergame.ch/en/, aufgerufen am 29.11.2018
[10]https://footballfever.tamu.edu, aufgerufen am 29.11.2018

Name	Medium	Ausprägung	Zielgruppe	Inhalt	Schwierigkeit	Schwierigkeitsgrade	Szenarien	Betreutes Spielen
Game of Threats[11]	Tablet	Decision making	Managers	IT Security for Management, Economical aspects of IT Security	Anfänger bis Fortgeschrittene	Nein	?	Ja
Hatch[12]	Analog	Kartenspiel	Mitarbeiter	Social Engineering Angriffe	Anfänger	Ja	Ja	Ja
Kaspersky Industrial Protection Simulation Business Game[13]	Tablet	Point & Click	System Experts, ITManagement	Security Threads, Costs of cyber attacks, Cooperation between management and IT	Anfänger bis Fortgeschrittene	Nein	Ja	Ja
KEEP TRADITION SECURE[14]	Web	Frage-Antwort	Studenten	Einfache Endanwender IT-Sicherheit	Anfänger	Nein	Nein	Nein

[11] https://www.pwc.co.uk/issues/cyber-security-data-privacy/services/game-of-threats.html, aufgerufen am 29.11.2018
[12] https://www.social-engineering.academy/de, aufgerufen am 29.11.2018
[13] https://media.kaspersky.com/en/business-security/enterprise/KL_SA_KIPS_overview_A4_Eng_web.pdf, aufgerufen am 29.11.2018
[14] https://keeptraditionsecure.tamu.edu, aufgerufen am 29.11.2018

Name	Medium	Ausprägung	Zielgruppe	Inhalt	Schwierigkeit	Schwierigkeitsgrade	Szenarien	Betreutes Spielen
Operation Digital Chameleon[15]	Analog	Karten und Brettspiel	IT Security Professionals	Fortgeschrittene IT-Sicherheit für kritische IT Systeme	Fortgeschrittene	Nein	Ja	Ja
Password Paladin[16]	Web	Frage-Antwort	Studenten, Universitätsmitarbeiter	Passwörter	Anfänger	Nein	Nein	Nein
Protect[17]	Web	Kartenspiel	Mitarbeiter	Social Engineering Angriffe	Anfänger	?	Ja	Nein
Real or No Deal[18]	Web	Frage-Antwort	Studenten, Universitätsmitarbeiter	Phishing	Anfänger	Nein	Nein	Nein
Sicher im Internet[19]	Web	Point & Click	Kleinunternehmer	Passwörter, Phishing, Allgemeine Sicherheit	Anfänger	Nein	Nein	Nein

[15]https://dl.acm.org/citation.cfm?id=2957800, aufgerufen am 29.11.2018
[16]https://www.adelaide.edu.au/technology/secureit/games/password_paladins/story.html, aufgerufen am 29.11.2018
[17]https://www.social-engineering.academy/de, aufgerufen am 29.11.2018
[18]https://www.adelaide.edu.au/technology/secureit/games/real_or_no_deal/story.html, aufgerufen am 29.11.2018
[19]https://www.sichere-identitaet-bb.de/sicheriminternet/, aufgerufen am 29.11.2018

Name	**Medium**	**Ausprägung**	**Zielgruppe**	**Inhalt**	**Schwierigkeit**	**Schwierigkeitsgrade**	**Szenarien**	**Betreutes Spielen**
The weakest Link[20]	Browser	Frage-Antwort	Mitarbeiter	IT Security	Anfänger	Nein	Nein	Nein
You Got Mail![21]	Web	Frage-Antwort	Studenten, Universitätsmitarbeiter	Phishing	Anfänger	Nein	Nein	Nein

Tabelle A.1: Serious Games zum Thema IT-Sicherheit

[20]https://www.isdecisions.com/user-security-awareness-game/, aufgerufen am 29.11.2018
[21]https://www.adelaide.edu.au/technology/secureit/games/youve_got_mail/story.html, aufgerufen am 29.11.2018

Anhang B

Flussdiagramme zur Android Mod Installation

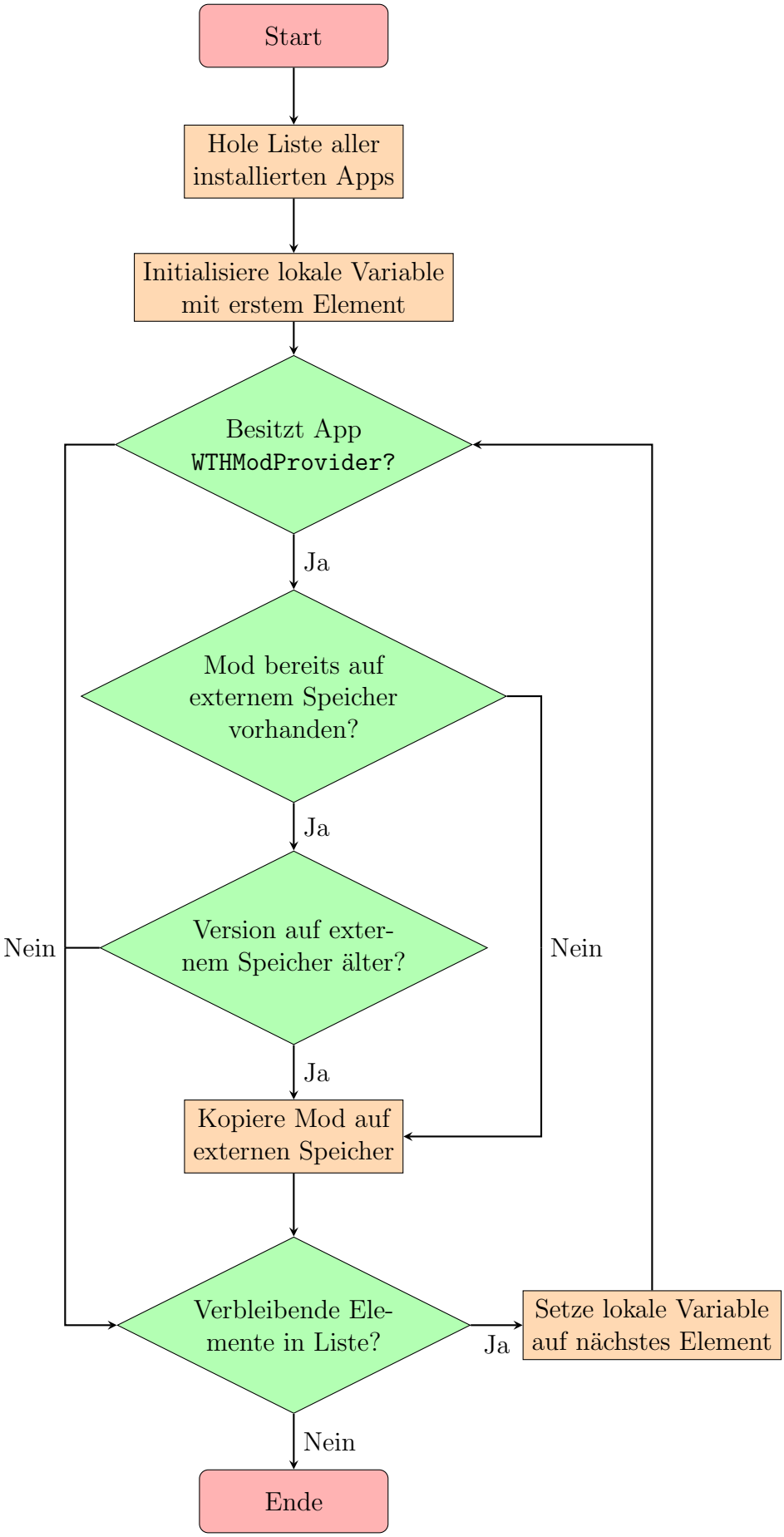


Abbildung B.1: Ablauf der Installation von Mods unter Android

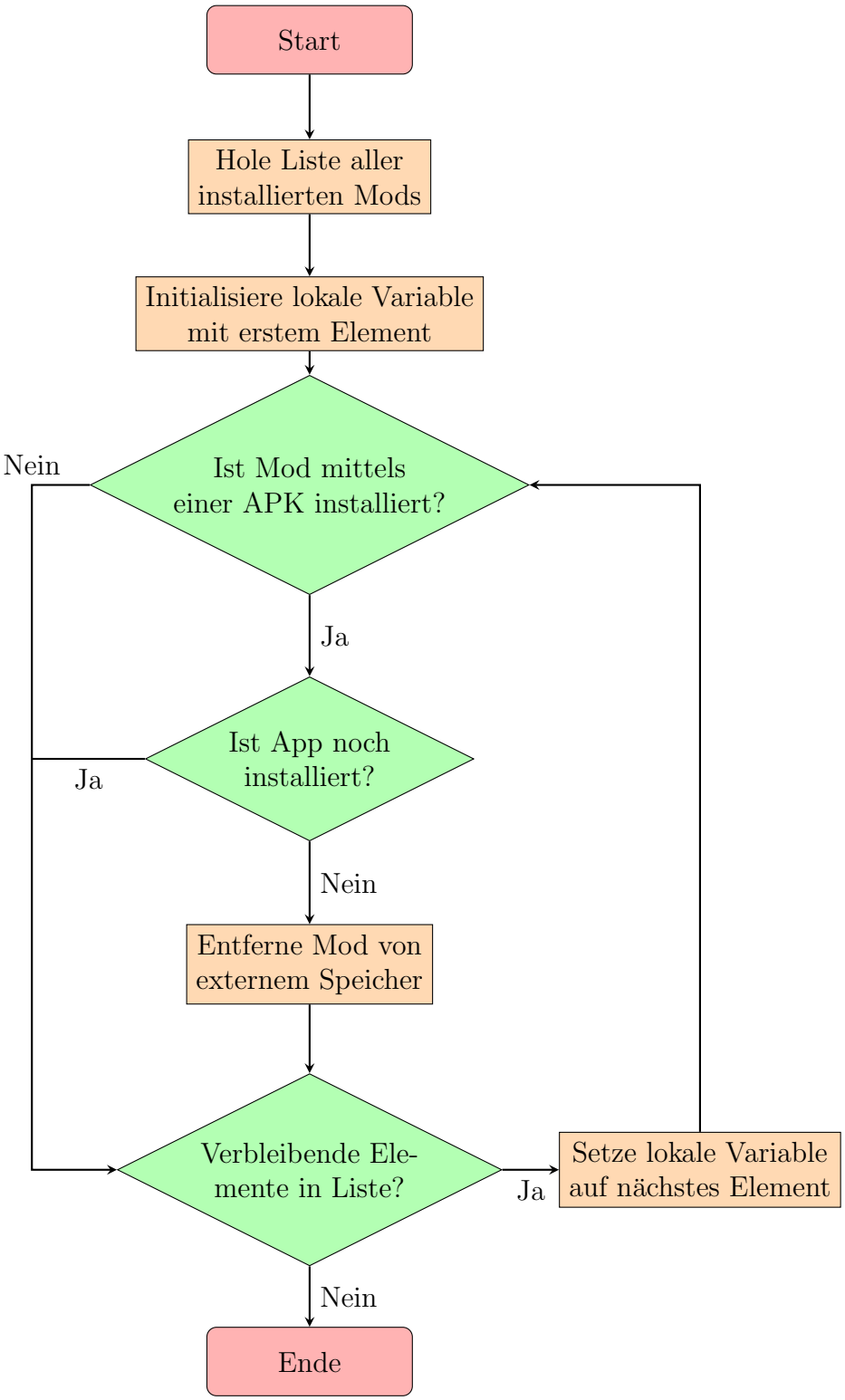


Abbildung B.2: Ablauf der Deinstallation von Mods unter Android

Literatur

Abt, C. C. (1987). *Serious Games*. University Press of America. (Siehe S. 9, 10, 14).

Bryant, J. & Vorderer, P. (2006). *Psychology of Entertainment*. Routledge Communication Series. Lawrence Erlbaum. (Siehe S. 10).

BSI. (2017). *Cyber-Sicherheits-Umfrage 2017*. Bundesamt für Sicherheit in der Informationstechnik. Aufgerufen am 29.10.2018. Zugriff unter https://www.allianz-fuer-cybersicherheit.de/SharedDocs/Downloads/ACS/cyber-sicherheits-umfrage_2017.pdf?__blob=publicationFile&v=3. (Siehe S. 2)

Bursztein, E. (2016). Does dropping usb drives really work? In *Black Hat USA 2016*. Aufgerufen am 18.02.2019. Zugriff unter https://elie.net/static/files/does-dropping-usb-drives-really-work/does-dropping-usb-drives-really-work-slides.pdf. (Siehe S. 1)

Eckert, C. (2009). *IT-Sicherheit: Konzepte - Verfahren - Protokolle* (6. Aufl.). Oldenbourg. (Siehe S. 7).

Hagen, J. M., Albrechtsen, E. & Hovden, J. (2008). Implementation and effectiveness of organizational information security measures. *Information Management & Computer Security*, *16*(4), 377–397. doi:10.1108/09685220810908796. eprint: https://doi.org/10.1108/09685220810908796. (Siehe S. 1)

Helisch, M. & Pokoyski, D. (2009). *Pokoyski, Security Awareness*. Edition kes. Vieweg+Teubner Verlag. (Siehe S. 1, 7–9).

Hendrix, M., Al-Sherbaz, A. & Bloom, V. (2016). Game Based Cyber Security Training: are Serious Games suitable for cyber security training? *International Journal of Serious Games*, *3*. doi:10.17083/ijsg.v3i1.107. (Siehe S. 4, 15, 21)

International Organization for Standardization. (2017). *Information technology – Security techniques – Information security management systems – Requirements (ISO 27001)*. (Siehe S. 1).

Khan, B. (2011). Effectiveness of information security awareness methods based on psychological theories. *AFRICAN JOURNAL OF BUSINESS MANAGEMENT*, *5*(26). doi:10.5897/ajbm11.067. (Siehe S. 15, 16)

Krapp, A. & Steiner, G. (2001). *Pädagogische Psychologie: ein Lehrbuch*. BeltzPVU. (Siehe S. 9).

Long, F., Mohindra, D., Seacord, R., Sutherland, D. & Svoboda, D. (2013). *Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs*. Pearson Education. (Siehe S. 53).

Michael, D. R. & Chen, S. L. (2005). *Serious Games: Games That Educate, Train, and Inform*. Course Technology PTR. (Siehe S. 10).

Nadi, S., Krüger, S., Mezini, M. & Bodden, E. (2016). Jumping Through Hoops: Why Do Java Developers Struggle with Cryptography APIs? In *Proceedings of the 38th International Conference on Software Engineering*. doi:10.1145/2884781.2884790. (Siehe S. 53)

Oerter, R. (1997). *Psychologie des Spiels*. Weinheim : Beltz, Psychologie Verlags Union. (Siehe S. 10).

Parsons, K., McCormac, A. & Butavicius, M. A. (2010). *Human Factors and Information Security : Individual , Culture and Security Environment*. Defence Science und Technology Group. (Siehe S. 7, 8).

Prensky, M. (2000). *Digital Game-Based Learning*. McGraw-Hill. (Siehe S. 14).

Proofpoint. (2018). *Der Faktor Mensch 2018*. Proofpoint, Inc. Aufgerufen am 29.10.2018. Zugriff unter https://www.proofpoint.com/sites/default/files/pfpt-de-tr-the-human-factor-2018.pdf. (Siehe S. 3)

Ritterfeld, U., Cody, M. & Vorderer, P. (2009). *Serious Games: Mechanisms and Effects*. Taylor & Francis. (Siehe S. 9, 11–14).

Smith, R. (2010). The Long History of Gaming in Military Training. *Simulation & Gaming*, *41*(1), 6–19. doi:10.1177/1046878109334330. (Siehe S. 10)

Stanton, J. M., Stam, K. R., Mastrangelo, P. & Jolton, J. (2005). Analysis of End User Security Behaviors. *Comput. Secur. 24*(2), 124–133. doi:10.1016/j.cose.2004.07.001. (Siehe S. 8)

Strahringer, S. & Leyh, C. (2017). *Gamification und Serious Games: Grundlagen, Vorgehen und Anwendungen*. Edition HMD. Springer Fachmedien Wiesbaden. (Siehe S. 2).

Tanenbaum, J. G., Antle, A. N. & Robinson, J. (2013). Three Perspectives on Behavior Change for Serious Games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (S. 3389–3392). CHI '13. doi:10.1145/2470654.2466464. (Siehe S. 17)

van Tilborg, H. C. A. & Jajodia, S. (2011). *Encyclopedia of Cryptography and Security*. doi:10.1007/978-1-4419-5906-5_1299. (Siehe S. 1)

Wolek, H. (2008). *Faktor Mensch in der IT-Sicherheit: Social Engineering, Sicherheitsbewußtsein, Awareness*. Books on Demand. (Siehe S. 2).