Nutzerzentrierte Entwicklung einer Game-based Learning Anwendung für Unity

Vincent Schiller
New Work Design Lab
Fachhochschule Dresden
01067 Dresden, Deutschland
v.schiller@fh-dresden.eu

Maximilian Liebscher
New Work Design Lab
Fachhochschule Dresden
01067 Dresden, Deutschland
m.liebscher@fh-dresden.eu

Marius Brade
New Work Design Lab
Fachhochschule Dresden
01067 Dresden, Deutschland
m.brade@fh-dresden.eu

ABSTRACT

In einer Zeit der Digitalisierung nimmt die Relevanz des Programmierens in Echtzeit-Entwicklungsumgebungen immer mehr zu. So auch in der Unity Engine. Zeitgleich werden Game-based Learning Anwendungen immer häufiger zum Erlernen neuer Inhalte eingesetzt. Mit ‚ENC#YPTED' wurde das erste frei zugängliche Unity-Programmierlernspiel entwickelt. ENC#YPED hat das Ziel, ein positives Lernerlebnis bei der Aneignung von Programmierfertigkeiten zu unterstützen, indem Lernende in eine Spielwelt eintauchen.

Die entstandene Anwendung wurde entsprechend einer nutzerzentrierten Entwicklung mithilfe von User-Experience Tests iterativ verbessert und weiterentwickelt. Im Ergebnis zeichnet sich ENC#YPED durch eine hohe Gebrauchstauglichkeit aus. Diese wird insbesondere durch folgende Merkmale erreicht: Nutzung bereits bekannter Funktionalitäten, Reiz-Reduzierung, Vorbeugung von Missverständnissen, ausführliche Feedbacks auf Fehler, genaue Erklärungen der Features und barrierefreie Lösungen. Es wird angenommen, dass durch den Einsatz von ENC#YPED motivationale Aspekte beim Erlernen von Programmierfertigkeiten positiv beeinflusst werden, da motivationsfördernde Techniken der Spielgestaltung zum Einsatz kommen, z. B. Aufbau einer emotionalen Bindung zwischen den Spielenden, der Story und der Spielfigur sowie Schaffung von Verbesserungsmöglichkeiten. Dieser Beitrag beschreibt das methodische Vorgehen bei der Umsetzung von ENC#YPTED und beschäftigt sich mit der Frage, wie beim spielerischen Programmieren ein positives Lernerlebnis geschaffen werden kann.

Die entstandene Game-based Learning Anwendung stellt einen Ausgangspunkt dar für weitere Forschung und Entwicklung zur Verbesserung des Fertigkeitserwerbs im Bereich der Programmierung.

MuC'20 Workshops, Magdeburg, Deutschland © Proceedings of the Mensch und Computer 2020 Workshop on «Gam-R – Gamification Reloaded». https://doi.org/10.18420/muc2020-ws103-311

CCS CONCEPTS

• Human-centered computing → User centered design; Social and professional topics → Software engineering education; Software and its engineering → Interactive games

KEYWORDS

Game-based Learning, GBL, User Experience Design, UXD, Unity, Programmieren Lernen

1 Einleitung

Studien zeigen, dass sich ein Großteil der Bundesbürger ein Leben ohne Smartphones gar nicht mehr vorstellen kann [1, p. 3] und täglich bzw. mehrmals pro Woche einen PC, Laptop oder ein Notebook im Haushalt benutzt [2, p. 54].

Laut der Bundesregierung sind wir „längst in einer digitalisierten Welt angekommen. Die Digitalisierung betrifft nicht mehr nur klassische IT-Unternehmen, sondern Unternehmen quer durch sämtliche Branchen und Sektoren" [3].

Dieser ständige Kontakt mit moderner Technik erfordert digitale Kompetenzen bei jedem Anwender. Gleichzeitig sind schon heute Menschen mit hoher digitaler Kompetenz und Fertigkeiten im Bereich der Programmierung auf dem Arbeitsmarkt sehr gefragt [4]. Dies zeigte sich beispielsweise 2018 an 82.000 freien Stellen im IT-Bereich [5]. Die Wissensvermittlung und Ausbildung im Bereich der Programmierung ist demnach von großer Bedeutung.

Die Software Unity ist als weltweit „meistgenutzte Realtime-3D-Entwicklungsplattform" [6] hochrelevant für vielfältige Einsatzbereiche da sie die Etablierung von Echtzeit-3D nicht nur in der Spielewelt, sondern auch in Bereichen der Architektur, Automobilbranche, des Films, der Medizin, der Raumfahrt und darüber hinaus vorantreibt [6] [7]. Sie umfasst Werkzeuge, mit

denen man sowohl klassische 2D- und 3D- als auch interaktive AR- und VR-Erfahrungen entwickeln kann.
Diese Werkzeuge helfen bei der Erstellung und Gestaltung von Partikeleffekten, Landschaften und Animationen sowie bei der Simulation physikalischer Begebenheiten. Obwohl Unity bereits eine Vielzahl an Komponenten für klassische Aufgaben liefert, wie beispielsweise in den Bereichen Audio, Physik und Rendering, muss vieles noch vom Entwickler selbst programmiert werden. Vor allem das Herzstück – die spezifische Spiellogik – erfordert die Erstellung sogenannter Skripte mithilfe der Programmiersprache C# [8, p. 37].
Es gibt viele Möglichkeiten und Angebote, das Programmieren (insbesondere in Unity) zu erlernen. Neben Suchmaschinen, Foren, Videos, Büchern und Unterricht beinhalten Spiele ein besonderes Potential bei der Wissensvermittlung im Bereich der Informatik. Digital Game-based Learning kann hierbei als die Verbindung von pädagogischen Inhalten mit Computerspielen verstanden werden [9, p. 145] und den Einstieg in das für viele Menschen „offenbar unheimliche" Thema, welches Berührungsängste in allen Altersstufen hervorruft [10], erleichtern. Es kann zudem helfen, genügend Motivation aufzubringen, um sich intensiv damit zu beschäftigen [11, p. 92].
Ein wichtiger Bestandteil ist dabei Gamification, also die Nutzung von Spielelementen in spielfremden Kontexten [12, p. 2], da dies sowohl einen positiven Effekt auf die Effektivität des Lernens als auch auf die erlebte Motivation hat [13]. Zudem können Spiele Unsicherheiten (z.B. Sorgen, etwas durch falsche Programmierung kaputt zu machen) beseitigen, da Umgebungen geschaffen werden können, die das Prinzip von Ursache und Wirkung spielerisch erlebbar machen.
Lernspiele gibt es zu allen gängigen Programmiersprachen, allerdings ist auf keinem der üblichen digitalen Marktplätze eine Anwendung zu finden, welche als Unity-Programmierlernspiel bezeichnet werden kann [11, p. 20].
Um dies zu ändern, wurde eine Simulationsanwendung für das Erlernen der Unity-spezifischen Programmierparadigmen namens ‚ENC#YPTED' entwickelt.
Hierbei wurde mithilfe eines User-Centered Design Ansatzes [14] mit User-Experience Tests erforscht, wie eine Digital Game-based Learning Anwendung zum Erlernen der Programmierung in der Unity Engine gestaltet werden kann, sodass sie gebrauchstauglich, motivierend und erweiterbar ist.

2 Verwandte Arbeiten

ENC#YPTED hat das Ziel, beim Anwender Wissen aufzubauen. Dabei soll eine gute User Experience erreicht werden. Um bei der Entwicklung beiden Aspekte zu genügen, wurde auf Grundlagenarbeiten zurückgegriffen, die darlegen, wie das Gehirn auf äußere Spieleinflüsse reagiert [15, p. 220] :

- Wahrnehmung ist subjektiv.
- Das Erinnerungsvermögen lässt nach, umso mehr Zeit vergeht.
- Aufmerksamkeit ist nur auf wenige Ressourcen beschränkt.
- Motivation beeinflusst unser Verhalten entscheidend.
- Emotionen beeinflussen unsere Erkenntnisse bzw. werden von ihnen beeinflusst, zudem steuern sie unser Verhalten.

Ebenso wurden vorhandene Spielkonzepte analysiert, um entsprechende Erkenntnisse für das eigene Konzept abzuleiten. Die Spiele ‚Pony Island' [16], ‚SQL Island' [17], ‚SoloLearn' [18], ‚Swift Playgrounds' [19] und ‚CodinGame' [20] wurden in den Kategorien ‚Zielgruppe', ‚Lernziele', ‚Story', ‚Design', ‚Features', ‚Plattform' und ‚Spieldauer' miteinander verglichen.
Insbesondere Pony Island und Swift Playgrounds inspirierten das Spielkonzept von ENC#YPTED in folgenden Punkten:

- Das Spielsetting des Indie-Horrorgames Pony Island bietet durch die Verbindung von visuellen Effekten mit dramatischen Geräuschen eine effektive Spannungserzeugung mit einfachen Mitteln.
- Im Gegensatz zu den untersuchten Spielen SQL Island und SoloLearn setzt Swift Playgrounds auf Belohnungen in Form von unterhaltsamen Animationen des zu steuernden Avatars. Die Kombination der emotionalen Bindung zur Spielfigur mit dem spielerischen Ausprobieren von unterschiedlichen Lösungsansätzen zeichnet die Anwendung aus.

3 User-Centered Design

Die gewählte Methode eines User-Centered Designs bestand in diesem Fall aus den mehrfach wiederholten Phasen Analyse, Konzeption, Umsetzung und Evaluierung.
Die Analyse umfasste vor allem Erkenntnisse aus den zuvor beschriebenen Kapiteln ‚Einleitung' und ‚Verwandte Arbeiten'. Der entscheidende Faktor für die erfolgreiche Umsetzung einer solchen Methode ist die frühzeitige Beteiligung der potenziellen Benutzer*Innen am Entwicklungsprozess. Die regelmäßige Evaluierung und anschließende iterative Verbesserung des Konzeptes und der Umsetzung sorgt für die Formung eines nutzerzentrierten Spielerlebnisses anstelle einer vom Entwickelnden festgelegten Vorstellung [14].
Während der Umsetzung von ENC#YPTED wurden zehn User Experience Tests mit insgesamt 16 Personen durchgeführt. Sie fanden in unterschiedlichen Entwicklungsstadien statt und dienten der Evaluierung der zuvor hinzugefügten, veränderten oder erweiterten Features. Das Alter der Probanden lag zwischen 19 und 61 (M: 26,75, SD: 10,59).
Allen wurde im Vorhinein die Story erklärt, sowie der Hinweis gegeben, dass sie eine Aufgabe lösen müssen, welche in einem Spielfenster zu sehen ist und, dass ein Code-Editor für die Lösung dieser Aufgabe genutzt werden muss.
Im Anschluss wurden die Testpersonen ohne weitere Informationen gebeten, das Level zu lösen. Hierbei wurden sie bei ihrem Umgang mit der Anwendung beobachtet und nach ihrer User Experience befragt. Wenn Fragen aufkamen, wurden diese beantwortet und die Kernprobleme notiert, um mit der Zeit intuitive Lösungen für diese zu finden. Die Dauer und der

Umfang der Tests waren stets unterschiedlich, da sie sich individuell und nutzerorientiert entwickelten.
Für die Dokumentation der User Experience-Tests (im Folgenden ‚UX-Tests') wurden Protokolle aufbereitet, in welchen sowohl Probleme als auch mögliche Lösungen zu diesen notiert sind.
Auf Grundlage der gewonnenen Erkenntnisse wurde die Anwendung demnach iterativ verbessert und weiterentwickelt.

4 Das Lernspielkonzept ‚ENC#YPTED'

Bei ENC#YPTED handelt es sich um ein dystopisches Computerspiel mit simulationsähnlichen Ansätzen und folgendem Nutzungsverlauf und Spielsetting:
Zum Beginn des Lernspiels betreten die Nutzer*Innen ein Windows-ähnliches Betriebssystem, dessen Funktionalitäten sie zunächst explorativ kennenlernen. Über die Nutzung des spieleigenen Internet Browsers werden sie zwangsweise zum Download eines Virus verleitet. Nachdem sich dieser selbstständig ausführt, kommt es zu einem Bluescreen, woraufhin sich das Betriebssystem mit einer düsteren Ästhetik wieder hochfährt. Von diesem Zeitpunkt an, können die Spieler*Innen nicht mehr auf die, bis dahin unbekannten Daten auf dem virtuellen Desktop zugreifen, da diese verschlüsselt wurden. Nur durch das Code-basierte Einsammeln von Dateien in der Spielwelt, können die Nutzer*Innen die Dateien auf dem Desktop wieder freischalten und öffnen. Wie sie zur Lösung dieser Aufgabe kommen, müssen sie selbst herausfinden, wobei ihre Möglichkeiten limitiert werden. Wenn die Nutzer*Innen alle Level erfolgreich gelöst und somit alle ihre Daten gerettet haben, verlässt der Virus das Betriebssystem wieder.
Im Folgenden wird das Konzept der Game-based Learning Anwendung an den Aspekten ‚Lernziele', ‚Zielgruppe', ‚Aufbau User Interface' vorgestellt, sowie ‚Spielelemente zur Gamifizierung' und ihre Entwicklung auf Basis der UX-Tests beschrieben.
Abbildung 1: ENC#YPTED - Code Editor und Spielfenster vor UX-Tests zeigt den Entwicklungsstand vor- und Abbildung 2 nach den UX-Tests.

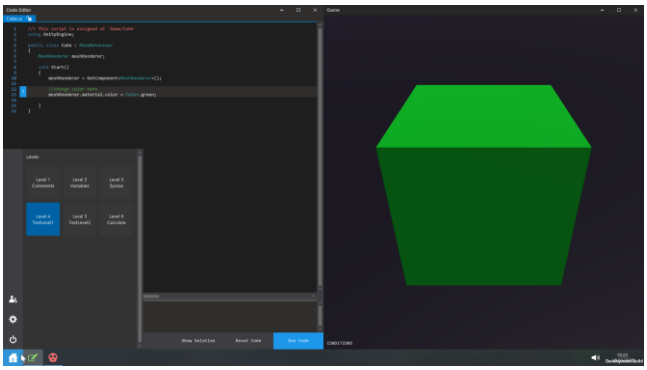

Abbildung 1: ENC#YPTED - Code Editor und Spielfenster vor UX-Tests

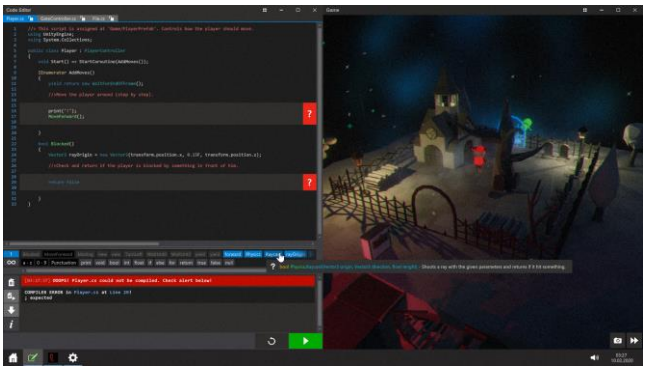

Abbildung 2: ENC#YPTED - Code Editor und Spielfenster nach UX-Tests

4.1 Lernziele

Die Anwender*Innen lernen im Spielverlauf Unity-spezifische Methoden kennen. Die drei Level beschäftigen sich jeweils mit bestimmten Themengebieten, wie Input, Physics und Animation, in Kombination mit grundlegenden C# Anwendungsfällen, wie der Verwendung von Vektoren, Schleifen und Logik. Das Lernziel ist es, Zusammenhänge zu erkennen und zu erlernen, wie man sich externe Hilfe für verschiedene Probleme suchen kann.
Die Intention der Anwendung ist es nicht, dass Benutzer*Innen nach dem Lernspiel eine umfassende Kenntnis der Unity Engine besitzen. Diese besteht mit ihrem grafischen Benutzerinterface aus weitaus mehr Facetten als dem *Coden*. Die Anwender*Innen sollten jedoch im Anschluss wenig Probleme damit haben, sich das restliche Wissen über die Engine auf anderen Wegen selbst anzueignen.

4.2 Zielgruppe

Das Spiel richtet sich vor allem an Studierende und Lehrkräfte im Bereich Informatik und Game Design, welche die Programmierung in der Unity Engine erlernen oder lehren möchten. Englischkenntnisse und grundlegende Programmierkenntnisse (optimalerweise in C#) sind Voraussetzungen, um die Lernziele erreichen zu können.
Da es sich bei der entwickelten Anwendung um einen Prototyp mit drei Leveln handelt, wird das gewählte Thema noch nicht umfassend abgedeckt. Sollten zukünftig weitere Level entwickelt werden, die anhand eines klar definierten, roten Fadens einen Komplett-Einblick in die Unity-spezifischen Paradigmen geben, besteht das Potenzial, die Anwendung nach einer einsemestrigen Grundlagenvorlesung zu objektorientierter Programmierung in der Lehre einzusetzen. Im Falle der Einbeziehung von objektorientierten Grundlagen mit der Programmiersprache C# innerhalb der Anwendung, ist sogar der Einsatz für Studierende ohne Vorkenntnisse denkbar.

4.3 Aufbau User Interface

Das Designkonzept basiert auf der Simulation eines realistisch wirkenden, abstrakten Windows Betriebssystems (vgl. Abbildung 2), um die Immersion zu verstärken. Anwendungsfenster lassen sich beliebig transformieren und geben den Spielenden damit die Freiheit, eine personalisierte Bildschirmaufteilung einzustellen. Um die geöffneten Fenster verwalten und die Simulation realitätsnah zu gestalten, wurden zudem eine Taskleiste und ein Startmenü implementiert. Eine Reduktion auf die wichtigsten grafischen Elemente dient dem Ziel, die Aufmerksamkeit von Benutzer*Innen im Spielverlauf steuern zu können. Ein Proband der UX-Tests half durch entsprechendes Feedback aufgrund seiner Rot-Grün-Sehschwäche bei der Identifizierung von Einschränkungen im User Interface.

Der eigens erstellte Code-Editor ist ästhetisch an einen Standard-Editor von Unity angelehnt, damit sich Benutzer*Innen später beim Wechsel von ENC#YPTED zur realen Unity-Anwendung aufgrund des bekannten Designs kaum umstellen müssen.

4.4 Spielelemente zur Gamifizierung

4.4.1 Dynamiken

4.4.1.1 Beschränkungen / Regeln. Im Code-Editor existieren in jedem Level verschiedene Skripte, welche die Nutzer*Innen bearbeiten können. Es wurde von Beginn an dafür gesorgt, dass es geschützte Bereiche innerhalb der vorbereiteten Skripte gibt, um diese nicht aus Versehen zu verändern. Zudem wird der geschriebene Code vor der Ausführung mit einer sog. Allowlist (Liste erlaubter Klassen und Methoden) abgeglichen. Die Relevanz und Entwicklung dessen wird im Folgenden beschrieben:

Beim klassischen Programmieren haben Programmierende unzählige Möglichkeiten für die Lösung der ihnen gestellten Aufgaben. Dieser Umstand lässt Raum für Kreativität und Motivation, weshalb der Ansatz ursprünglich beibehalten werden sollte. Durch Selbsttests wurde jedoch schnell festgestellt, dass ohne jegliche Art von Eingabe-Einschränkung schwerwiegende Sicherheitsprobleme herbeigeführt werden können. Aus diesem Grund wurde eine Blocklist (Liste gesperrter Klassen und Methoden) implementiert, um potenziell kritische Befehle erkennen und verbieten zu können.

Aufgrund von Beobachtungen während der UX-Tests stellte sich heraus, dass dies bei weitem nicht ausreicht, um die Anwendung sicher und verständlich gestalten zu können. Zum einen ist es fast unmöglich, alle gefährdenden Funktionen auf einer Blocklist zu vereinen, da es auch für diese unzählige andere Möglichkeiten der Umsetzung gibt. Zum anderen kann eine theoretisch korrekte Umsetzung der Aufgaben nicht zum gewollten Lernerfolg führen. Aufgrund der gegebenen Freiheiten können sich die Spieler*Innen den für sie angenehmsten Weg zur Lösung aussuchen, ohne zwingend Unity-eigene Methoden kennenzulernen.

Durch den darauffolgenden Einsatz der Allowlist wird dem Spiel zwar der Aspekt von unzähligen möglichen Lösungen eines Problems und der damit verbundenen Freiheit entzogen, allerdings zu Gunsten der Sicherheit und des Lernerfolgs. Zudem können die Spieler*Innen damit besser durch die Level geleitet und einer Überforderung vorgebeugt werden, da sie die möglichen Eingabemöglichkeiten sehen und sich die Lösungen besser erschließen lassen. Die genaue Umsetzung der Allowlist wurde durch Swift Playgrounds inspiriert.

4.4.1.2 Narrative Elemente. Die Geschichte handelt von einer außer Kontrolle geratenen künstlichen Intelligenz (KI) eines riesigen Konzerns, welche die Spieler*Innen im Rahmen eines Akquise-Programms zu Programmierenden ausbilden möchte und dabei ihre persönlichen Daten verschlüsselt. Nach erfolgreicher Lösung von Levels, werden die Dateien nacheinander wieder freigegeben. Zugleich gewinnen die Spieler*Innen zahlreiche Programmierkenntnisse dazu, womit die KI ihre Aufgabe erfüllt.

Ursprünglich sollte sich die Story damit beschäftigten, dass der Benutzende ein Spiel veröffentlicht hat, welches von einem unbekannten Konkurrenten gehackt und manipuliert wurde. Die Aufgabe der Spieler*Innen sollte demnach die Reparatur der fehlerhaften Level mithilfe von sog. Hotfixes darstellen. Hierdurch wurde versucht, eine emotionale Bindung des Spielenden zur Story aufzubauen sowie ein übergeordnetes, motivierendes Ziel zu erschaffen – die Reparatur des eigenen Spiels und das Besiegen des Hackers. Um wie bei dieser Idee vorgesehen Hotfixes für die gehackten Stellen in den Levels entwickeln zu können, müssten die Anwender*Innen bereits wissen, welche Lösung sie anstreben. Es wurde während der UX-Befragungen jedoch schnell festgestellt, dass dieses Ziel sehr fremdgesteuert wirkt und eher extrinsische Motivationen [21] hervorruft, da es sich bei den Spieler*Innen um Personen mit wenig Vorwissen handelte, die sich nicht mit der Geschichte identifizieren konnten. Deshalb wurde die Story im weiteren Verlauf abgeändert, um intrinsische Motivationen [22] zu befördern und eine möglichst schlüssige und langanhaltende Spielerbindung zu erreichen. Die Story sah von diesem Zeitpunkt an vor, dass der fiktive Computer der Spieler*Innen von einem Hacker attackiert wird, der mit der Löschung der von ihm verschlüsselten Daten droht, sollten die Spieler*Innen die gestellten Aufgaben nicht lösen. Ebenso fehlte seit Beginn eine Erklärung für die Motivation des Hackers, um die Identifizierung der Nutzer*Innen mit der Geschichte abzurunden. Es wurde sich dafür entschieden, dass ‚der Hacker' im weiteren Verlauf als eine außer Kontrolle geratene KI beschrieben wird. Durch diese Änderung wurde die Geschichte als schlüssig und von den Probanden als zukünftig denkbar empfunden, womit die Simulation noch immersiver wurde.

4.4.1.3 Emotionen. Durch das narrative Element der Datenverschlüsselung sowie die passende Ästhetik und die Unwissenheit der Spieler*Innen was in der Ausgangssituation passiert ist, wird die Neugier und der Ehrgeiz, eine Antwort herauszufinden begünstigt. Schwierige Herausforderungen programmatisch zu lösen kann für Momente der Frustration sowie Glück bei Erfolgsmomenten führen. Die emotionale Bindung zum gesteuerten Avatar wird durch die

Rahmengeschichte begünstigt. Aufgrund der Erkenntnisse aus den UX-Tests wurden unterhaltende Animationen und Sound-Effekte für die Spielfigur eingeführt. Um negative Emotionen nach einer nicht erfolgreichen Lösung eines Levels nach dem Ausführen von Code nicht überhand nehmen zu lassen, wurden alternative Spielausgänge entwickelt, die den Spieler*Innen zwar signalisieren, dass das Level nicht geschafft ist, aber ein Gefühl des spielerischen Lernens hervorrufen. Dies wurde beispielsweise durch eine unterhaltsame Hinfall-Animation umgesetzt, nach welcher der Avatar wieder aufsteht und Spieler*Innen es noch einmal probieren können. Dieser Umgang mit Fehlern wirkte sich in den UX-Tests merkbar positiv auf die Motivationserhaltung der Testpersonen aus.

4.4.1.4 Fortschritt. Die Verbindung des Code-Editors mit der Spielwelt wird im Hintergrund über einen implementierten Runtime-Compiler hergestellt, welcher es ermöglicht, den vom Spielenden verfassten Code innerhalb der Laufzeit der Anwendung auszuführen und die Spielwelt damit zu beeinflussen. Dies ermöglicht ein spielerisches Ausprobieren von selbst geschriebenen Programmzeilen. Der Avatar in der Spielwelt führt den durch Benutzer*Innen gewählten Code visuell aus, indem er sich entsprechend in der Spielwelt bewegt und mit den Artefakten interagiert. Entspricht das Verhalten des Avatars nicht den Erwartungen von Benutzer*Innen, so können diese das direkt mit ihrem Code in Verbindung bringen und einen neuen, iterativ verbesserten Lösungsansatz konzipieren. Hierdurch entstehen unmittelbare Verknüpfungen von Ursache und Wirkungs-prinzipien der Programmierung.

Sollte es den Spielenden nicht gelingen, sich die Lösung ohne Hilfestellung zu erschließen, können sie das ‚Instant Help Forum' (vgl. Abbildung 3, rechts) innerhalb der Anwendung nutzen, um sofort Hilfe zu möglichen Fragen zu erhalten. Hiermit wird das Lernziel adressiert, sich später Wissen selbst anzueignen und Werkzeuge wie Suchmaschinen und Foren im Programmier-Alltag effektiv für sich nutzen zu können.

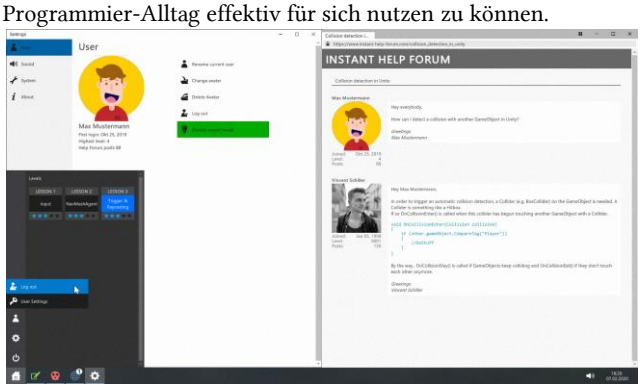

Abbildung 3: ENC#YPTED - Settings, Forum, Start Menü

In den UX-Tests fehlte den Probanden laut eigener Aussage eine Bestrafung für die Nutzung der Hilfe-Funktion. Die Intention der Einrichtung dieses Features war es, den Spieler*Innen weiterzuhelfen, jedoch erst, wenn diese nach ausführlichem Nachdenken sicher sind, dass sie nicht von selbst auf die Lösung kommen können. Da die intrinsische Motivation, die Aufgaben aus reinem Ehrgeiz ohne Hilfe zu schaffen, nicht ausreichte, wurde eine extrinsische Motivation eingeführt, bei welcher die Spieler*Innen für jede gestellte Frage einen bestimmten Anteil ihrer bis zu fünf Sterne pro Level abgezogen bekommen. Diese Vorgehensweise gilt laut Hodent zunächst als nicht empfehlenswert [15, p. 70]. Am Ende eines Levels bekämen sie demnach ihre übrig gebliebene Anzahl an Sternen angezeigt, welche auch dauerhaft gespeichert und für das entsprechende Level sichtbar bleibt, bis sie es mit einem besseren Ergebnis lösen, womit wiederum eine neue intrinsische Motivation gegeben ist, die die extrinsische überbieten könnte. Dies animiert zudem zum Wiederholen und damit besseren Verinnerlichen des gelernten Wissens [15, p. 70].

4.4.2 Mechaniken

4.4.2.1 Herausforderungen. Grundsätzlich wurde festgestellt, dass ein Tutorial-Level, in welchem der Ablauf der Aufgabe und die Herangehensweise beim Lösen erläutert wird, als dringend nötig und wünschenswert erachtet wird. Es wurde schnell festgestellt, dass das erstellte Level zu schwer für Menschen mit wenigen Vorkenntnissen zu sein scheint, auch wenn dies bei der Entwicklung dessen nicht so empfunden oder gewollt wurde. Dieser Konflikt zwischen dem, was Entwickler*Innen sich vorstellen und dem, was Nutzer*Innen wahrnehmen stellt einen Hauptgrund zur Durchführung von UX-Tests dar [15, p. 15].

4.4.2.2 Sammeln von Gegenständen. Damit sich die Anwender*Innen besser mit dem Charakter identifizieren können und, um für eine gewisse Kontinuität zwischen den einzelnen Levels zu sorgen, wurde eine Code-basierte Bewegungssteuerung für die Spielfigur entwickelt, mit dessen Hilfe die Spieler*Innen verschlüsselte Dateien auf dem Spielfeld einsammeln müssen. Dieses Ziel können sie nur mit der Lösung von Level-spezifischen Programmieraufgaben erreichen. Sowohl Code-basierte Bewegung und das Einsammeln eines Items findet man auch bei Swift Playgrounds.

4.4.2.3 Feedback. Sollten beim Versuch der Lösung bzw. bei der Ausführung von selbst geschriebenen Programmzeilen Fehler auftreten, so werden diese in übersichtlichen Konsolenausgaben dargestellt (vgl. Abbildung 2, links unten). Durch die UX-Tests wurde zudem festgestellt, dass sich einige Probanden Debug-Möglichkeiten wünschten, welche von ihnen bewusst hervorgerufen werden können. Folgend wurde das selbstgesteuerte Feedback in Form von Konsolenmeldungen für sie möglich gemacht.

Ein weiteres Ergebnis der Befragungen war die Einführung eines Features, das Anfängern mit aufblinkenden Hotspots hilft, auf die nächsten Schritte aufmerksam zu werden, sowie Fehler einfacher lokalisieren zu können. Ebenso wurde ein auswählbarer Expertenmodus eingeführt, um Reizüberflutungen bei Unerfahrenen vorzubeugen aber Fortgeschrittenen die Möglichkeit zu geben, für sie interessante Informationen aufnehmen zu können, die bei zuvor Genannten nicht angezeigt werden. Eine weitere Konsequenz der Reizreduzierung sowie der Verbesserung der Verständlichkeit war das Hinzufügen von Tooltips (vgl. Abbildung 2, unter Cursor)., wodurch viele Texte

mit Buttons ersetzt werden konnten, die genauere Informationen beim Hovern über diesen anzeigen.

4.4.2.4 Belohnungen. Bei jedem erfolgreich abgeschlossenen Level erhalten Spieler*Innen Teile ihrer Daten als freigeschaltete Dateien lesbar zurück, wodurch sie in der Rahmengeschichte weiter voranschreiten. Zudem unterstreicht die Jubelanimation des Avatars ein Belohnungsgefühl.

4.4.3 Komponenten

4.4.3.1 Avatare / Gegner. Als Maßnahme zu der UX-basierten Story-Änderungen wurde das Spiel durch animierte Charaktere erweitert, um zum einen den ästhetischen Wert zu steigern und zum anderen eine emotionale Bindung der Spielenden mit der Anwendung zu unterstützen. Dies sollte sich außerdem positiv auf den Spielspaß und damit die Motivation zum Lernen auswirken. Einfache Konsolenausgaben oder das Ändern einer Farbe waren den Tester*Innen zu langweilig und fremd, was man auf den niedrigen Gamification-Faktor zurückzuführen könnte.

Dargestellt werden die Avatare, so wie der Rest des Level-Designs mithilfe von Low-Poly-Objekten (3D-Modellen mit minimaler Anzahl an Flächen).
Die KI wird als Datenpirat mit ‚Anonymous-Maske' (vgl. Abbildung 4, rechts) verkörpert, um den abstrakten Gegner greifbar zu machen und damit die emotionale Bindung zwischen Anwender*Innen und Spiel zu verstärken. Durch die Anonymisierung der Spielerfigur (vgl. Abbildung 4, links) mit Kapuze und Maske, soll sich jede Person mit ihr identifizieren können.

Abbildung 4: ENC#YPTED - Character Design

4.4.3.2 Level. Ziel eines jeden Levels ist es, die Datei in der 3D-Spielwelt vor den Datenpiraten durch Ausführung eines passenden Programmcodes mit dem Avatar zu erreichen.

Die spielbaren Level waren anfangs losgelöst von spezifischen Designvorstellungen und Levelzielen – manche Level sollten mit Textausgabe, andere in 2D- oder 3D-Grafik, egal ob in kindlichen oder erwachsenen Umgebungen dargestellt werden. Die Idee war, jeden Aspekt der Programmierung in C# und Unity nur so komplex wie nötig aber so ansprechend wie möglich behandeln zu können. Feedback von Probanden der UX-Tests zeigte jedoch, dass eine ästhetisch ansprechende 3D-Darstellung den meisten Zuspruch bekam.

5 Bewertung des Prototyps hinsichtlich Gebrauchstauglichkeit und Nutzungsmotivation

5.1 Gebrauchstauglichkeit

Die Tests haben ergeben, dass die Gebrauchstauglichkeit stark abhängig von der Zielgruppendefinition ist. Zudem wurde versucht, bekannte Funktionalitäten und Gewohnheiten weitgehend zu nutzen, z.B. das Windows-ähnliche Betriebssystem oder der an Visual Studio angelehnte Aufbau und das Design des Code-Editors. Dies erfolgte einerseits, um den Nutzer*Innen den Einstieg in ENC#YPTED ohne große Lernphase zu ermöglichen. Andererseits sollte darüber die Immersion erhöht werden, also das Erleben der virtuellen Umgebung als möglichst realitätsnah. Durch diesen simulationsähnlichen Zustand waren zudem alle wichtigen Bedienelemente und Features unmittelbar und durch die freie Anordnung von Fenstern personalisiert erreichbar.
Im Test wurden auch Erwartungen der Nutzenden aufgrund von Vorerfahrungen und Gewohnheiten deutlich, deren Fehlen sich unmittelbar negativ auf die User Experience auswirkte. Ein Beispiel hierfür stellt die bis zum jetzigen Stand fehlende Undo-/Redo-Funktionalität im Code-Editor dar. Es wurde im Verlauf der Entwicklung jedoch versucht, derartige Erwartungen weitestgehend zu erfüllen.
Neben der intuitiven Benutzung war es von großer Bedeutung, die Reizintensität in Übereinstimmung mit zunehmender Expertise der Nutzenden zu erhöhen. So wurde beispielsweise ein Experten-Modus eingeführt, der Erfahrenen mehr Informationen anzeigt als Nutzenden mit wenigen Vorkenntnissen, um eine Überforderung von Verarbeitungskapazitäten zu verhindern. Viel Wert wurde auch auf die genaue Beschreibung unbekannter Reize gelegt, damit diese nicht miss- bzw. gar nicht verstanden werden können. Hierbei war die Einführung von Tooltips ein wesentlicher Bestandteil. Sie ermöglichen das Anzeigen hilfreicher Informationen zu einzelnen Bedienelementen, erscheinen aber nur, wenn sich bewusst dafür entschieden wird, ihnen Aufmerksamkeit durch das Hovern über die entsprechenden Felder zu schenken. Somit konnten viele Texte durch Icons mit Tooltips ersetzt werden, wodurch das User Interface deutlich übersichtlicher wurde. Um Aufmerksamkeitsprozesse bei den Nutzenden lernförderlich zu unterstützen, wurden mehrere Indikatoren eingesetzt, die auf eine mögliche Interaktion aufmerksam machen, beispielsweise Farb-, Cursor- und Größenänderung.
Teilweise versuchten die Nutzenden auch verbotene oder nicht mögliche Aktionen auszuführen. Ohne Feedback, warum die gewollte Aktion nicht funktioniert, fühlten sich die Probanden schnell hilflos oder überfordert. Aus diesem Grund wurden für typische Nutzerfehler Hinweise in der Konsole und optische Signale in Form von aufblinkenden Feldern implementiert. Wenn die Anwendenden nun versuchen, Code auszuführen, der nicht kompiliert werden kann, erfolgt eine eindeutige und

übersichtliche Fehlermeldung. Die Gestaltung der Fehlermeldung folgt dabei dem Grundsatz, Verbesserungsmöglichkeiten aufzuzeigen, statt Fehler zu markieren, womit Frust vermieden und die Lernmotivation aufrechterhalten werden kann. Hierbei musste darauf geachtet werden, dass Fehlerquellen überschaubar blieben und die Nutzenden gar nicht erst schwerwiegende Probleme produzieren konnten. Unter anderem aus diesem Grund wurde die Allowlist entwickelt, da ansonsten Abstürze und Datenverluste nicht auszuschließen waren.
Ein weiterer, zunächst nicht bedachter, Grund für mögliches Erleben von Hilflosigkeit auf Seiten der Anwendenden stellt eine nicht-barrierefreie Umsetzung für Menschen mit Einschränkungen, wie der Farbenfehlsichtigkeit eines Probanden dar. Eine Anpassbarkeit der Signalfarben und die freie Wahl von Schriftgrößen sind mögliche Lösungen hierfür.
Auch bei Probanden ohne Einschränkung wurde festgestellt, dass Elemente, die nicht in einer typischen Signalfarbe erscheinen, wenig bis gar keine Beachtung erhielten. Selbst hervorstechende Elemente konnten je nach Größe und Position teilweise nicht wahrgenommen werden. Um deren Beachtung zu unterstützen, wurden akustische und optische Signale (wie aufleuchtende Felder) entwickelt. Vereinzelt zeigte sich, dass dennoch nicht jedes Element intuitiv verständlich sein kann. Trotz Umfärbung, Umpositionierung, Vergrößerung und Aufleuchten wurden Buttons teilweise übersehen, womit explizierte Erläuterungen in Tutorial-Levels nötig wären, um ungewohnte Features zu erläutern.

5.2 Motivation

Eine wichtige Basis der Motivationsgestaltung stellt die emotionale Bindung der Spielenden zu ihrer Spielfigur dar. Hierbei ist die Story- und Character-Entwicklung entscheidend. Alle Probanden der UX-Tests gaben an, von der Story-Idee begeistert zu sein. Gelobt wurde dabei, dass sie sich mit einer (für einige Probanden vorstellbaren) Dystopie, also einer negativen Zukunftsvision beschäftigt. Alle Testpersonen schätzten das Spiel als spannende Simulation eines zukünftig nicht unmöglichen Szenarios ein. Die aus der Story entstehende Motivation, die Daten des Spielenden retten zu wollen, ist simpel, aber gleichzeitig sehr verständlich für jeden Computernutzenden. Dies unterstützt eine Identifikation mit dem Protagonisten. Verstärkt wird diese durch Personalisierungen, wie der Festlegung des eigenen Namens und Avatar-Bildes, welche folgend für persönliche Anreden des Spielenden eingesetzt wurden.

Um diese Identifikation noch immersiver zu gestalten, wurden zwei Spielfiguren entwickelt. Bei der einen handelt es sich um einen (Daten-)Piraten. Die andere verkörperte den Spielenden.
Hierdurch erhält der psychische Konflikt zwischen KI und Anwendenden eine weitere, physische Ebene. Ähnlich wirken sich das Einsammeln der Dateien in der Spielwelt und die damit einhergehende Freischaltung der Daten auf dem virtuellen Desktop des Spielenden aus. Durch diese immersive Darstellung des Konflikts konnten weitere motivationsrelevante Features umgesetzt werden, allen voran die Animation und Ästhetik. Wie eine der Testpersonen beschrieb: „Die Magic steckt in der Figur". Durch unerwartete Spielausgänge und lustige Animationen wird das Lehr- und Lerngebiet aufgelockert. Der Nutzen für die Motivation besteht dabei im explorativen oder zufälligen Entdecken dieser Spielausgänge, welche zum Teil gewollte als auch zufällige Belohnungen durch direkten Einfluss des Spielenden darstellen und damit ein hohes Motivationspotential aufweisen. Alle Probanden gaben an, mit großer Spannung den Einfluss ihrer neuen Code-Eingabe erwartet zu haben.

Die Schwierigkeit war unerwarteterweise kaum relevant für die Motivation der Probanden. Der Großteil der Versuchspersonen konnte nur ein, nach eigenen Angaben, sehr schweres Level testen, dessen Lösung für manche auf Grund mangelnder Vorkenntnisse ohne Hilfestellungen (fast) unmöglich gewesen wäre. Dennoch gaben alle Probanden an, durchgängig motiviert gewesen zu sein. Auf Nachfrage, warum dem so sei, antworte der Großteil, dass sie „der Ehrgeiz gepackt" habe. Das von den meisten getestete Level wurde mitunter zwei Stunden lang gespielt – eine beachtliche Dauer, wenn man bedenkt, dass die untersuchten vergleichbaren Game-based Learning Spiele wenige Sekunden oder Minuten für das Lösen eines Levels veranschlagen. Bei der entwickelten Anwendung wird die Motivation laut Aussage der Tester durch Etappen-Erfolge, die sich die Spielenden selbst erarbeiten, immer wieder aufrechterhalten. Hierbei verloren die Probanden teilweise ihr Zeitgefühl und zeigten sich überrascht von der Dauer der vergangenen Spielzeit. Dies und die Aussagen zur Spiel-bzw. Lernmotivation in der vorliegenden Anwendung deuten darauf hin, dass durch ENC#YPTED ein Flow-Erlebnis [23, p. 75] auslösbar ist und dies trotz hoher wahrgenommener Schwierigkeit. Durch die lange Spieldauer eines Levels, welches sich mit verhältnismäßig wenigen Unity-spezifischen Methoden beschäftigt, kann davon ausgegangen werden, dass die Wissensvermittlung deutlich langsamer abläuft als bei herkömmlichen Methoden, wie dem Folgen von YouTube-Tutorials. Andererseits ist aufgrund des hohen Motivationspotentials und des Unterhaltungsfaktors auch eine schnellere Überwindung der Einstiegshürde denkbar.

Ein weiterer Motivationsfaktor ist die zukünftig geplante Bewertung der Leistung am Ende eines Levels, womit Spielende zur Wiederholung animiert werden. Auf diese Weise könnte das Gelernte elaborierter verinnerlicht werden.

6 Diskussion und Ausblick

Der in Unity entwickelte, sehr fortgeschrittene Prototyp der Anwendung ist ein innovatives und zum jetzigen Stand einzigartiges Produkt für das Erlernen der Programmierung in Unity.
Die gewählte Methode hat sich als äußerst effektiv herausgestellt. Durch die nutzerbasierte Untersuchung konnten viele unbedachte UX-Probleme identifiziert und gelöst werden.

Bisherige UX-Tests verfolgten nicht das Ziel, den Prototyp in der Breite unter gleichen Bedingungen und Fragestellungen zu evaluieren, sondern dienten der iterativen Entwicklung. Vor diesem Hintergrund schien die explorative Herangehensweise sehr geeignet, Schwachstellen des entwickelten Systems aufzudecken und iterativ zu verändern. Die UX-Tests zeigten eine Diskrepanz in der Schwierigkeitseinschätzung zwischen den Nutzer*Innen und dem Entwickler, die im Rahmen bisheriger Iterationsprozesse nicht behoben werden konnte.
Wenngleich durch diesen Umstand viele UX-Probleme identifiziert werden konnten, welche auf Grund der fehlenden Features in einfacheren Levels nicht aufgefallen wären, wäre eine Implementierung von einfacheren Levels zukünftig wünschenswert. Diese würde einen breiteren Einsatz des Systems im Rahmen von standardisierten UX-Tests an einer großen Stichprobe sowie eine gezieltere Untersuchung des Lernerfolgs ermöglichen. Hierbei wäre auch zu untersuchen, ob es geschlechtsspezifische Unterschiede in der Nutzung von ENC#YPTED gibt.
Die gewonnenen Erkenntnisse beziehen sich zwar auf die Entwicklungsplattform Unity, können aber stellenweise auch für andere Engines adaptiert werden.

REFERENCES

[1] Bitkom, "Smartphone-Markt: Konjunktur und Trends," 2019.

[2] VuMA, "Berichtsband VuMA 2019," 2019.

[3] BMWi, "Digitalisierung," o.J.. [Online]. Available: https://www.bmwi.de/Redaktion/DE/Dossier/digitalisierung.html. [Accessed 03 Februar 2020].

[4] J. Boie, "Warum man Coden lernen sollte," 2014. [Online]. Available: https://www.sueddeutsche.de/digital/programmieren-warum-man-coden-lernen-sollte-1.2063091. [Accessed 03 Februar 2020].

[5] Bitkom, "82.000 freie Jobs: IT-Fachkräftemangel spitzt sich zu," 2018. [Online]. Available: https://www.bitkom.org/Presse/Presseinformation/82000-freie-Jobs-IT-Fachkraeftemangel-spitzt. [Accessed 03 Februar 2020].

[6] Unity Technologies, "Public Relations," 2020a. [Online]. Available: https://unity3d.com/public-relations. [Accessed 01 Februar 2020].

[7] Unity Technologies, "Echtzeitlösungen, endlose Möglichkeiten," 2020b. [Online]. Available: https://unity.com/de/solutions. [Accessed 31 Januar 2020].

[8] C. Seifert and J. Wislaug, Spiele entwickeln mit Unity 5: 2D- und 3D-Games mit Unity und C# für Desktop, Web & Mobile, Hanser, 2017.

[9] M. Prensky, Digital Game-Based Learning, PARAGON HOUSE PUBL, 2007.

[10] J. Bernewasser, "Informatik für alle," *ZEIT ONLINE*, 2019.

[11] V. Schiller, Digital Game Based Learning am Beispiel der Programmierung in der Unity Engine: Entwicklung eines Simulationsspiels für das Erlernen der Unity-spezifischen Programmierparadigmen, Dresden, 2020.

[12] S. Deterding et al., "Gamification: Toward a definition," in *CHI 2011 Gamification Workshop Proceedings*, 2011.

[13] J. Hamari, J. Koivisto and H. Sarsa, "Does Gamification Work? A Literature Review of Empirical Studies on Gamification.," in *IEEE 47th Hawaii International Conference on System Sciences.*, Hawaii, 2014.

[14] J. Zimmermann and S. Gerstl, "Eine Einführung in den User Centered Design Process (UCDP)," 02 01 2019. [Online]. Available: https://www.embedded-software-engineering.de/eine-einfuehrung-in-den-user-centered-design-process-ucdp-a-785958/. [Accessed 01 06 2020].

[15] C. Hodent, The Gamer's Brain: How Neuroscience and UX Can Impact Video Game Design, CRC Press, 2017.

[16] M. Böhm, "Pony Island im Test: Seltsames Meta-Spiel für den Computer," 2016. [Online]. Available: https://www.spiegel.de/netzwelt/games/pony-island-im-test-seltsames-meta-spiel-fuer-den-computer-a-1073700.html. [Accessed 05 Februar 2020].

[17] J. Schildgen and . S. Deßloch, "SQL-Grundlagen spielend lernen mit dem Text-Adventure," 2015.

[18] SoloLearn, "SoloLearn: Learn to Code for Free," 2020b. [Online]. Available: https://www.sololearn.com/. [Accessed 04 Februar 2020].

[19] Apple Inc., "Swift Playgrounds Apple (DE)," 2020. [Online]. Available: https://www.apple.com/de/swift/playgrounds/. [Accessed 05 Februar 2020].

[20] CodinGame, "Play with Programming - CodinGame," 2020. [Online]. Available: https://www.codingame.com/. [Accessed 05 Februar 2020].

[21] R. M. Ryan and E. L. Deci, "Intrinsic and extrinsic motivations: Classic definitions and new directions.," *Contemporary educational psychology*, pp. 54-67, 2000.

[22] E. L. Deci, Intrinsic motivation., New York: Plenum Press, 1975.

[23] M. Csikszentmihalyi, Das flow-Erlebnis: jenseits von Angst und Langeweile: im Tun aufgehen, Klett-Cotta, 1987.

[24] BMBF, "Wissenswertes zum DigitalPakt Schule," 2019. [Online]. Available: https://www.bmbf.de/de/wissenswertes-zum-digitalpakt-schule-6496.php. [Accessed 03 Februar 2020].