Karlsruher Institut für Technologie

Fraunhofer IOSB

Smarte digitale Lernspiele mit interoperabler Kopplung und Adaptivität

Bachelorarbeit von

Andreas Lobstedt

an der Fakultät für Informatik
Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung IOSB

Erstgutachter:	Prof. Dr.-Ing. J. Beyerer
Zweitgutachter:	Prof. Dr.-Ing. T. Längle
Betreuender Mitarbeiter:	Dipl.-Inf. Alexander Streicher
Zweiter betreuender Mitarbeiter:	Dipl.-Inf. (FH) Daniel Atorf, M.Sc.

01. August 2020 – 30. November 2020

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Änderungen entnommen wurde.

Karlsruhe, 27.11.2020

. . . . . .

(Andreas Lobstedt)

Zusammenfassung

Um das Lernen mit Hilfe von digitalen Lernspielen zu verbessern gilt es, den Spieler im sogenannten Flow-Kanal zu halten, in der Balance zwischen Herausforderung und seinen Fähigkeiten. Dazu wurde am Fraunhofer IOSB das Adaptivität-Framework „E-Learning A.I.“ (ELAI) entwickelt.

Diese Arbeit untersucht, wie sich eine interoperable, netzwerkbasierte Kopplung des Lernspiels „Lost Earth 2308“ mit der ELAI realisieren lässt, um anhand von zu entwickelnden Beispielszenarien die Mehrwerte einer Adaptivität zu demonstrieren. Damit soll untersucht werden, in wiefern ELAI, ein externes Adaptivität-Framework, ein Lernspiel adaptieren kann.

Aufbauend auf zu entwickelnden Konzepten nach dem Stand der Forschung und Technik sowie Szenarien zur Kopplung und Adaption werden Lost Earth und ELAI technisch und inhaltlich angepasst. Eine technische Verifikation anhand von Beispielszenarien zeigt die Adaptivitätsfähigkeit und die Mehrwehrte.

Abgrenzend zu einer echten automatischen K.I.-Adaptivität liegt der Fokus in dieser Arbeit auf der eigentlichen Kopplung und der Demonstration mittels der L.I.S.A. Die Adaptivität kann mittels einfachen heuristischen Regelsätzen simuliert werden.

Inhaltsverzeichnis

1 Einführung

E-Learning ist ein wichtiges Forschungsthema am Fraunhofer IOSB. 2016 wurde das Adaptivität-Framework E-Learning A.I. (ELAI) entwickelt mit dem Ziel digitale Lernspiele zu adaptieren. Bei digitalen Lernspielen ist es sehr wichtig die Balance zwischen den Fähigkeiten des Spielers und der Schwierigkeit der aktuellen Aufgabe zu halten. Dieser Zustand wird auch als Flow-Zustand bezeichnet.

In Abbildung 1.1 ist dies leicht zu erkennen. Ein Spieler kann schnell frustriert werden, wenn die Aufgaben nicht seinen Fähigkeiten entsprechen. Ebenso kann ein Spieler sich langweilen, sollten die Aufgaben trivial für ihn sein. Um dies zu verhindern ist es wichtig ein Spiel an die Bedürfnisse des Spielers anzupassen.

Ein Lernspiel oder auch Serious Game genannt hat viele Definitionen, jedoch sind die meisten sich einig, dass Lernspiele im Kern Spiele sind, welche andere Zwecke als Entertainment verfolgen [Tarja Susi 2007]. Für Lernspiele lässt sich dieser Zweck auch leicht beschreiben. Lernspiele versuchen Entertainment mit der Wissensvermittlung zu verbinden. Um diesen Zweck zu erreichen werden Computerspiele genutzt, damit der Nutzer vor Faszination gefesselt ist und somit das Erlangen neuer Fertigkeiten vereinfacht wird [Corti 2006].

Um ein Lernspiel an die Bedürfnisse der Spieler anzupassen ist Adaptivität notwendig, da ein vordefinierter Spielablauf von herkömmlichen Spielen sich nicht während des Spielens auf den Spieler anpassen kann [Mohammad u. a. 2016]. Hierbei geht es jedoch nicht nur um die Schwierigkeit einer gegebenen Aufgabe. Mit Hilfe von Adaptivität kann auch die Art und Weise wie das Wissen vermittelt wird an den jeweiligen Spieler angepasst werden [David Hauger 2007; F. Kareal 2006]. Es ist deshalb auch wichtig sinnvoll zu adaptieren. Um sinnvoll zu adaptieren ist es wichtig viele Informationen über den momentanen Spieler zu haben, um möglichst genau auf die einzelnen Bedürfnisse des Spielers eingehen zu können. Man sollte, wenn möglich, verallgemeinerte Adaptierungen vermeiden, denn diese würde zwar ebenfalls das Spiel adaptieren, aber im Endeffekt nicht für jeden Spieler nützlich sein.

In dieser Arbeit wird das Lernspiel Lost Earth 2308 betrachtet. Lost Earth 2308 basiert auf den Ansätzen des Game Based Learning [Prensky 2003] und der Immersiven Didaktik [Bopp 2005]. Ziel des Spiels ist es den Ausbildungsprozess von Bildauswertern zu unterstützen, um Kenntnisse der Bildauswertehandlung, des Auswertezyklus und des Einsatzes geeigneter Sensoren zu vermitteln. Im Spiel geht es darum das Schicksal einer fiktiven galaktischen Fraktion zu bestimmen. Der Spieler muss durch geschickte Bildauswertung und geeigneten Ressourceneinsatz versuchen immer mehr verloren gegangene Kolonien der Menschheit zu finden und in die eigene Fraktion eingliedern. Währenddessen steht der Spieler in direkter Konkurrenz zu anderen feindlichen Fraktionen[Atorf 2020].

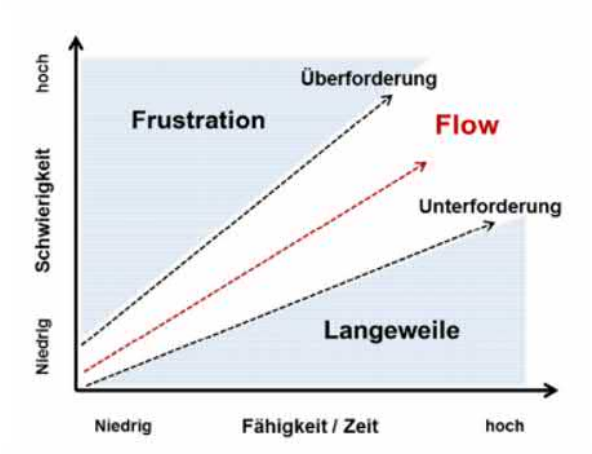

Abbildung 1.1: Flow-Kanal [Doujak 2015]

1.1 Problemstellung

In dieser Thesis soll gezeigt werden wie sich grundsätzlich Unity basierte Lernspiele an REST basierte Adaptivitätsframeworks anbinden lassen. Unter REST versteht man eine Web Architektur die genutzt wird um Web Services zu erstellen. Durch RESTful Webservices wird die Interoperabilität über das Netzwerk garantiert [W3C Working Group 2004].

Diese Fragestellung wird hier anhand von Lost Earth 2308 als Unity basiertes Lernspiel und der ELAI als REST basiertes Adaptivitätsframework untersucht. Lost Earth 2308 kann momentan noch nicht von ELAI adaptiert werden. Somit ist es momentan noch nicht möglich den Spielablauf in Lost Earth 2308 auf die unterschiedlichen Bedürfnisse unterschiedlicher Spieler anzupassen. Die Frage mit der sich diese Arbeit befasst ist, wie lässt sich ein digitales Lernspiel standardbasiert und interoperabel an die ELAI anbinden, damit das Lernspiel adaptiert werden kann und somit der Lernfortschritt verbessert werden kann. Um die Interoperabilität, also die nahtlose Zusammenarbeit verschiedener Systeme, zu gewährleisten wird die Schnittstelle auf vorhandenen Standards basieren und nicht auf proprietären Lösungen aufbauen. Außerdem ist die Frage an welchen Stellen kann ein Lernspiel sinnvoll adaptiert werden um den Spielablauf an den einzelnen Spieler anzupassen.

1.2 Zielsetzung

Ziel dieser Arbeit ist die Kopplung zwischen dem Serious Game Lost Earth 2308 und der ELAI zu implementieren. Dabei soll diese Kopplung möglichst alleine stehen und interoperabel sein, damit sie zukünftig für andere Serious Games verwendet werden kann. Mit Hilfe der Kopplung soll es der ELAI ermöglicht werden den digitalen Assistenten LISA in Lost Earth 2308 zu adaptieren.

In nachfolgenden Arbeiten können die Adaptivitätsmethoden, welche hier nur grundsätzlich gezeigt werden, mit Hilfe einer künstlichen Intelligenz dargestellt werden.

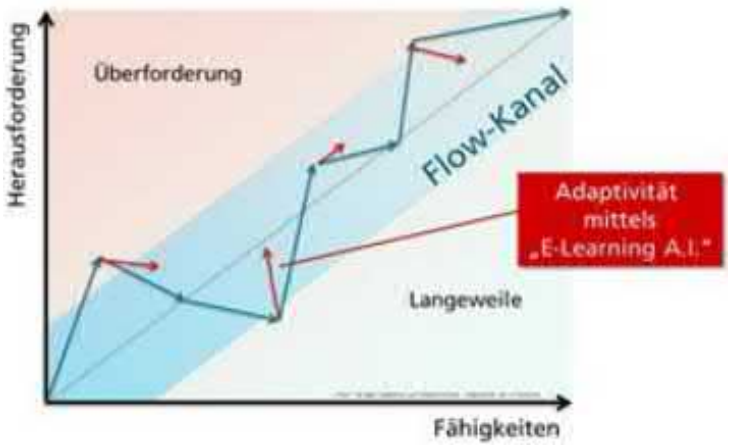

Abbildung 1.2: ELAI Reaktion im Flow-Kanal [Streicher u. a. 2016]

1.3 Projektumfeld

Die Abteilung Interoperabilität und Assistenzsysteme am Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung in Karlsruhe erforscht wie es möglich ist E-Learning mit digitalen Lernspielen interoperabel und adaptiv zu gestalten. Das Projektumfeld ist das Projekt Intelligente bildgestützte Aufklärung in verteilten Sensorsystemen (IBAS). Dieses Projekt ist im Bereich E-Learning für die Bildauswertung angesiedelt. Speziell bezieht sich dieses Projekt auf die Luft- und Satellitenbildauswertung.

Das Projekt dient dazu Bildauswerter die nötigen Fähigkeiten zu vermitteln, um ihre Aufgabe in der vorgegebenen Zeit erfolgreich zu erfüllen. Mit Hilfe von Lehrgängen oder auch arbeitsbegleitenden computergestützten Lernszenarien werden die notwendigen Fähigkeiten an den Lernenden vermittelt.

Hierfür wird die E-Learning A.I. (ELAI) genutzt. ELAI ist ein interoperabler externer Tutoring-Agent, welcher eine didaktisch geprägte Adaptionslogik beinhaltet, um das Lernen mit digitalen Lernspielen zu begleiten und zu adaptieren. Die ELAI versucht, wie in Abbildung 1.2 gezeigt, den User damit immer im Flow-Kanal zu halten, in dem sie misst wo sich der Spieler gerade in dem Flow-Kanal befindet und darauf entsprechend reagiert.

1.4 Aufbau und Gliederung

Im Folgenden wird der Aufbau der Arbeit erklärt.

Kapitel 1 In diesem Kapitel wird der Leser mit Hilfe einer Motivation dem Thema nahe gebracht. Darüber hinaus wird das Projektumfeld, die Problemstellung und die Ziele dieser Bachelorthesis vorgestellt.

Kapitel 2 Der aktuelle Stand der Forschung und Technik zu den Themen E-Learning Interoperabilität und E-Learning Adaptivität wird hier dargestellt.

Kapitel 3 Hier werden einige Grundlegende Konzepte für diese Thesis erklärt. Dazu gehört die Erklärung von Serious Games, die Unity Game Engine und die Experience API. Für diese Arbeit besonders relevant sind die E-Learning A.I. ELAI mit ihrere Adaptivitätsantwort, der pädagogische Agent LISA und das Serious Game Lost Earth 2308.

Kapitel 4 Kapitel vier stellt das grundlegende Konzept des adaptiven Assistenten LISA und des ELAIControlServices dar.

Kapitel 5 In dem Implementierungskapitel wird zunächst der erste Prototyp dieser Thesis dargestellt, welcher genutzt wurde um das zugrundeliegende Konzept zu verifizieren. Danach wird auf die interoperable Schnittstelle und den adaptiven Assistenten LISA eingegangen. Zum Schluss wird noch die Implementierung des ELAIControlServices gezeigt.

Kapitel 6 In der Verifikation wird mit Hilfe von Szenarien verifiziert, dass die Ziele dieser Thesis erfüllt wurden.

Kapitel 7 Das letzte Kapitel fasst die Thesis zusammen und es wird ein Fazit gezogen. Dazu wird ein Ausblick für zukünftige Arbeiten geliefert.

2 Stand der Forschung und Technik

Der Fokus dieser Arbeit liegt auf dem Themenbereich E-Learning, insbesondere der E-Learning Interoperabilität und der E-Learning Adaptivität. In diesem Kapitel wird auf Forschungsarbeiten zu diesen Teilbereichen eingegangen und verschiedene Konzepte dargestellt.

2.1 E-Learning Interoperabilität

Laut Collier und Robson (2002) ist die Interoperabilität der Schlüssel zu einer erfolgreichen E-Learning Umgebung, denn diese bringt unter anderem Vorteile wie eine einfachere Entwicklung oder auch die Bereitstellung von Lernmaterialien in einem standardisierten Format mit sich [Collier u. a. 2002]. Um diese Interoperabilität zu gewährleisten benötigt es Standards. Diese Standards werden im E-Learning nicht nur zum gewährleisten der Interoperabilität, sondern auch für die Portabilität und Wiederverwendbarkeit genutzt [Friesen 2005].

Es existieren mehrere große Organisationen die Standards für den E-Learning Bereich entwickeln [Friesen 2005]. Darunter fallen IMS Global Learning Consortium [IMS Global Learning Consortium 2020], IEEE Learning Technology Standard Committee [IEEE LTSC 2004] und ISO/IEC JTC 1/SC36 [ISO/ICE 2004].

IMS GLC arbeitet daran basierend auf den Anforderungen ihrer Mitglieder Standards zu entwickeln. Seit der Entstehung 1997 hat IMS GLC bereits um die 50 Standards entwickelt von denen fünf weit verbreitet sind [Bakhouyi u. a. 2017]. Das Ziel von IMS GLC ist es die Interoperabilität zwischen Systemen, welche sich mit E-Learning auseinandersetzen, zu gewährleisten [Bakhouyi u. a. 2017; IMS Global Learning Consortium 2020].

Das Ziel des IEEE LTSC ist es, mit Hilfe von vereinbarten Standards, Implementierungsempfehlungen und anderen Dokumentationen, es dem Schüler einfach zu machen so viel wie möglich durch den Computer zu lernen. Das LTSC verfolgt einen offenen und transparenten Entwicklungsprozess für neue Standards und arbeitet dabei mit anderen Organisationen zusammen [Bakhouyi u. a. 2017; IEEE LTSC 2004].

Die ISO ist eine international anerkannte Organisation, welche sich in vielen verschiedenen Bereichen mit der Standardisierung beschäftigt [Friesen 2005]. Wohingegen sich die IEC gezielt auf die Standardisierung von Elektrogeräten, Elektronik und ähnlichen Technologien fokusiert. Zusammen haben sie ein Joint Technical Committee (JTC1) gegründet, welche sich auf IT Standards fokusiert. SC36 ist ein Unterkomitee der JTC1 und spezialisiert sich auf die Implementierung und Förderung von qualitativen E-Learning Standards [Friesen 2005; ISO/ICE 2004].

In seiner Publikation beschäftigt sich Stracke mit der Wichtigkeit der Interoperabilität und der Qualitätsentwicklung für den E-Learning Bereich [Stracke 2006]. Er schreibt, dass

die Interoperabilität eine Vorraussetzung für die Qualitätsentwicklung ist. Ein Problem, dass Stracke aufweist ist, dass es unmöglich ist für ein externes System den inneren Lernprozess eines Schülers zu beobachten und zu verfolgen. Der Lernfortschritt, das Wissen und die Kompetenzen werden vom Schüler selbst erreicht [Stracke 2006]. Stracke definiert Interopabilität wie folgt: „Interoperability means the ability of exchange and reuse of every kind of information and resources in any way within or between different systems."[Stracke 2006]. Basierend auf dieser Definition gibt es laut Stracke vier Bereiche der Interoperabilität. Diese Bereiche wären „Internal", bei dem die Interoperabilität nur innerhalb eines Systems stattfindet, „Directional", bei dem der Datenaustausch nur von einem zu einem anderen System stattfindet, „Mutual", welcher den Austausch zweier Systeme in beide Richtungen darstellt, und „General", wobei hier der Austausch zwischen allen Systemen in jegliche Richtung stattfindet [Stracke 2006].

Aroyo u.a. beschreiben in ihrer Publikation von 2004 was der nächste Schritt des E-Learnings sein sollte [Aroyo u. a. 2004]. Hierbei stellen sie zwei wichtige Herausforderungen dar. Zum einen die vollständige Interoperabilität zwischen verschiedenen Lernsystemen zu erreichen und zum anderen die automatisierte, strukturierte und einheitliche Hilfe beim Erstellen von Lernsystemen zu erzielen. Um die vollständige Interoperabilität zu erreichen, gilt es sich auf die semantische Konzeptionalisierung, die standardisierte Kommunikationssyntax und die umfangreiche Service-basierte Integration von Lernmittel zu stützen. Laut der W3C Working Group ist das Semantic Web eine Erweiterung des Internets, durch welche Informationen eine klare Bedeutung erhalten, womit es Computern und Menschen einfacher gemacht wird zusammen zu arbeiten [W3C Working Group 2020]. Aroyo u.a. beschreiben wie das Semantic Web dabei helfen kann die oben genannten Ziele zu erfüllen. Durch das Semantic Web ist es möglich einen einfacheren Zugang und besseres Management von Informationen zu erhalten. Darüber hinaus ist es auch möglich eine semantisch reichhaltigere Modellierung von Anwendungen und derer Nutzer zu erzielen [Aroyo u. a. 2004].

Um Interoperabilität zu gewährleisten muss die Datensammlung im E-Learning Bereich ebenfalls plattformübergreifend und standardbasiert sein. Hierfür gibt es die Experience API (xAPI), welche ältere Standards wie AICC und SCORM ersetzt [Benedek 2013]. Die xAPI ist eine Service API für den Umgang mit Activity Streams [Benedek 2013]. Hierbei werden sogenannte Statements aufgezeichnet und versendet. Diese bestehen in der Regel aus einem Akteur, einem Verb und einem Objekt, können aber auch weitere Informationen, wie z.B. den Kontext und die Ergebnisse, beinhalten [Kevan u. a. 2016]. Vorteile der xAPI sind, dass es lesbar für den Menschen ist und somit einfach verstanden werden kann [Kevan u. a. 2016] und dass es einfach ist die Datensammlung einer Lernaktivität zu pausieren, eine andere Aktivität zu starten und danach die Datensammlung der ersten Lernaktivität wieder fortzusetzen [Benedek 2013].

De Penning u.a. stellen in ihrem Paper von 2008 die SimSCORM Plattform vor, welche Trainingssimulatoren und E-Learning interoperabel und standardbasiert kombinieren soll [Penning u. a. 2008]. Durch die Durchsetzung von Standards, wie z.B. HLA und SCORM, sei es nun möglich diese Kombinatierung kosteneffektiv durchzuführen. SimSCORM ist dadurch flexibel und kann sich mit praktisch jedem HLA-konformen Trainsingssimulator mit einem SCORM-konformen Learning Management System (LMS) verbinden. Die ELAI,

welche in dieser Arbeit als zentrale Adaptionseinheit genutzt wird, basiert ebenfalls auf HLA und ersetzt SCORM durch den neueren xAPI Standard.

Für diese Arbeit ist Interoperabilität sehr wichtig, da es hierbei darum geht ein System zu entwickeln, welches größtmöglich eingesetzt werden kann. Dazu muss es auf Standards aufbauen wie z.B. HTTP und xAPI (Abschnitt 3.4). Über diese Standards kann sichergestellt werden, dass systemübergreifend Daten erfasst und genutzt werden können. Im Beispiel dieser Arbeit wird z.B. xAPI genutzt um ausgelesene Daten aus einem Serious Game für die letztendliche Entscheidung der Adaptivität zu nutzen. Durch das standardbasierte Format xAPI lassen sich somit Daten aus vielen verschiedenen Anwendungen nutzen und dadurch kann eine genauere, auf den Nutzer zugeschnittene Adaptivität vollzogen werden. Im Falle dieser Arbeit wird der „Mutual"Bereich der Interoperabilität genutzt, denn hier können jegliche Komponeneten bidirektional mit der ELAI interagieren, jedoch nicht direkt untereinander.

2.2 E-Learning Adaptivität

In der Publikation „Towards an interoperable adaptive tutoring agent for simulations and serious games"von Streicher und Roller beschreiben sie, dass Adaptivität genutzt werden muss um die intrinsische Motivation zum Lernen und Interagieren auszunutzen [Streicher 2015]. Dies lässt sich leicht anhand des Flow-Kanals in Abbildung 1.2 belegen. Das von Streicher und Roller beschriebene Ziel ist auch für Prensky von hoher Bedeutung, denn er schreibt ebenfalls darüber Engagement anstatt Frustration beim Spieler hervorrufen zu wollen [Prensky 2005]. Weiter schreiben Streicher und Roller, dass sich die meisten E-Learning Nutzer heutzutage mit Spielen und Simulationen bereits auskennen, jedoch gibt es bislang noch kaum bis gar keine didaktische Adaptivität im E-Learning Bereich, welche von dem Vorwissen der Nutzer profitieren kann [Streicher 2015]. Um diese Adaptivität zu gewährleisten wird empfohlenen eine externe Adaptivitätseinheit zu nutzen, da diese über unterschiedliche Anwendungen hinweg einen Nutzer bei dem Lernvorgang begeleiten kann und somit sich besser an den Nutzer adaptieren kann. In Abbildung 2.1 ist ein vorgeschlagenes Modell für die Entkopplung der Adaptivitätseinheit und des Serious Games von Streicher und Roller zu sehen. Hierbei wird ein Game Engine Adapter, eine externe Adaptivitätseinheit und eine dazwischen liegende Kommunikationsschicht genutzt. Der Game Engine Adapter dient dazu Informationen des Spielers zu sammeln und diese mit Hilfe der Kommunikationsschicht an die Adaptivitätseinheit weiterzuleiten, damit diese dann die Daten analysieren kann und basierend darauf dann Adaptivitätsentscheidungen trifft [Streicher 2015].

Alshammari u.a. schreiben, dass das traditionelle Game Design fokusiert auf einen generischen Nutzer ist [Alshammari u. a. 2014]. Der Lernprozess, also das Erlangen von Wissen und Fähigkeiten, ist jedoch bei vielen Nutzern verschiedenen. Um das Lernen so einfach wie möglich zu machen, muss das Spiel auf die Bedürfnisse des einzelnen Nutzers adaptiert werden. Dabei müssen das momentane Wissen, der Lernstil und der affektive Zustand des Nutzers berücksichtigt werden. Der Lernstil ist hierbei die wichtigste Charakteristik die es zu berücksichtigen gilt [Bostrom u. a. 1990; Felder u. a. 1988; Keefe 1979].

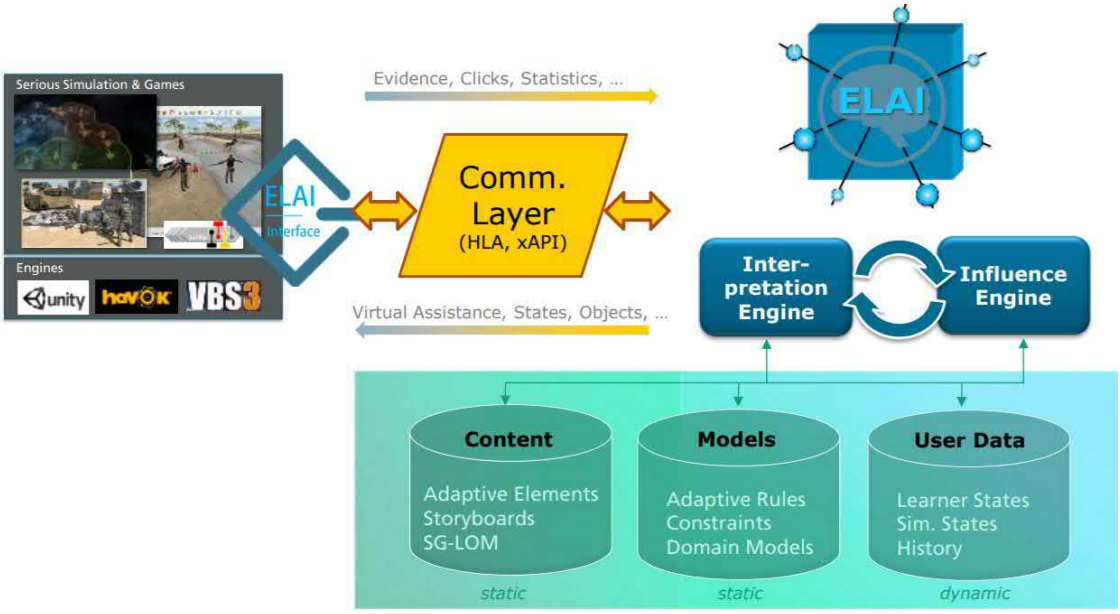

Abbildung 2.1: Beispiel für Entkopplung der Adaptivitätseinheit und des Serious Games [Streicher 2015]

Karael u.a. sind ebenfalls der Meinung, dass verschiedenen Schüler unterschiedliche Informationen in verschiedenen Formen benötigen und das der Grund für ein adaptives Lernsystem ist [F. Kareal 2006]. Darüber hinaus sind sie auch der Meinung, dass es bei der Adaptivität darauf ankommt den Lernprozess auf den einzelnen Schüler anzupassen. Darüber hinaus beschreiben sie, wie Adaptivität die Anzahl an Nutzer erhöhen kann, da auch weniger erfahrene Nutzer sich somit leichter mit einer Thematik auseinandersetzen können [F. Kareal 2006].

Hauger u.a. sind der Meinung dass Adaptivität sehr hilfreich für das Bildungswesen ist [David Hauger 2007]. Darüber hinaus nennen sie zwei Gründe, weshalb Adaptivität so wichtig ist für den E-Learning Bereich. Der erste Grund ist, dass ein System von unterschiedlichen Nutzern benutzt wird. Diese Nutzer unterscheiden sich dabei in Hinblick auf ihre Ziele, Lernstile, Präferenzen, Hintergründe und ihr Wissen. Darüber hinaus zeigen sie auch noch auf, dass sich das Wissen eines Nutzers mit der Zeit verändert [David Hauger 2007]. Der zweite Grund den Hauger u.a. aufzeigen ist, dass ein System einem Nutzer durch Nutzer-spezifische Pfade helfen kann.

Danz hat in seiner Masterarbeit sich die Aufgabe gestellt, zu testen ob ein digitaler Agent einem Nutzer beim Lernen helfen kann [Danz 2018]. Sein Ziel war es zu überprüfen in wie weit er das Lernspiel Lost Earth 2307 dahingehend adaptieren kann einen pädagogischen Agenten namens LISA anstatt herkömmlichen Tutorials zu nutzen. Er konnte feststellen, dass Nutzer, welche das Lernspiel mit LISA gespielt haben, in allen Metriken, vor allem aber in der Missionsdauer, etwas besser abgeschnitten haben. Damit konnte er zeigen, dass es einen kleinen Vorteil hat einen pädagogischen Agenten wie LISA zu nutzen, denn er konnte einen signifikanten Effekt zwischen Nutzern mit und ohne Erfahrung in Strategiespielen feststellen [Danz 2018]. In dieser Arbeit wird auf seiner Arbeit aufgebaut und LISA um eine Adaptivitätskomponente erweitert.

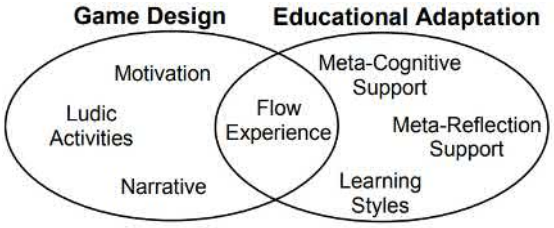

Abbildung 2.2: Trennung von Game Design und didaktischer Adaption [Peirce u. a. 2008]

Peirce u.a. beschreiben in diesem Paper von 2008, dass Lernspiele das Potential haben die intrinsische Motivation zum Lernen von Nutzern zu steigern, jedoch von den generischen Ansätzen des Lernvorgangs zurückgehalten werden [Peirce u. a. 2008]. Sie gehen weiter darauf ein, dass adaptive Lernspiele dieses Problem lösen können, in dem sie auf eine persönliche Lernerfahrung bauen. In diesem Paper schlagen Peirce u.a. eine Architektur namens ALIGN („Adaptive Learning In Games through Non-invasion")[Peirce u. a. 2008] vor, welche es ermöglicht Lernspiele unaufdringlich zu adaptieren. Die vorgestellte Architektur separiert Spieldesign und die didaktische Adaption klar mit Ausnahme von der Flow Erfahrung (Abbildung 1.1), welche für beide Aspekte sehr wichtig ist. Die Unterteilung der Architektur ist in Abbildung 2.2 deutlich zu sehen. Bei der ALIGN-Architektur werden Spielmomente auf abstrakte Lernkonzepte und abstrakte Adaptionen auf Veränderungen der Spielwelt übertragen. Ein Beispiel nennen Peirce u.a. sei das Erfassen eines Fehlers des Spielers während einer bestimmten Aufgabe und das dahereingehende Reduzieren der Spielerfähigkeit für diese bestimmte Aufgabe [Peirce u. a. 2008]. Dieses Konzept wird auch in dieser Arbeit genutzt, denn hier werden die Fehler und die Erfolge des Spielers zu unterschiedlichen Aufgaben aufgezeichnet, um später diese Informationen für die Adaption des Spiels zu nutzen.

In dem Paper „An adaptive Games-Based exercising framework"stellen Silva und El Saddik ein Framework für die Entwicklung einer adaptiven Game Engine für Exercising Spiele vor [Silva u. a. 2011]. Dieses Framework basiert darauf eine Game Engine zu entwickeln, welche jegliche Daten über den Spieler, wie z.B. Alter, Geschlecht und Fitnesszustand, weiss und diese benutzen kann das Spiel basierend für diesen Spieler anzupassen. Dies ist hier wichtig, denn es geht hier um ein physisches Trainingsspiel, welches zwischen unerfahrenen und erfahrenen bzw. fitten Spieler und Spielern die das nicht sind unterscheiden muss. Hierbei wird eine Adaptionsstrategie verwendet, welche basierend auf den Informationen die gesammelt werden über den Spieler während des Spiels, wie z.B. die physische Anstrengung, Adaptivitätsentscheidungen trifft [Silva u. a. 2011]. Das Ziel dieses Frameworks ist es Spieler in einer gewissen physischen Anstrengung zu halten, damit sie optimal Trainieren können. Das Framework wurde so konzipiert, dass es modular erweiterbar ist [Silva u. a. 2011].

Kickmeier u.a. gehen auf Probleme von Lernspielen ein, wie z.B. die hohen Kosten und die dadurch auftretende technologische Diskrepanz zwischen Lernspielen und professionellen, kommerziellen Spielen oder aber der fehlende psychologische, pädagogische oder didaktische Hintergrund [Kickmeier u. a. 2006]. In ihrem Paper stellen sie daher das ELEKTRA („Enhanced learning experience and knowledge transfer") Projekt vor, welches die

genannten Probleme beheben soll, in dem ELEKTRA einen interdisziplinären Ansatz für Kognitionswissenschaft, Neurowissenschaft, Pädagogik, Spieldesign und Spieleentwicklung nutzt [Kickmeier u. a. 2006]. ELEKTRA will deshalb die Vorteile von traditionellen Spielen nutzen, wie z.B. die Immersion des Spielers durch den hohen Realismus und den vielen Interaktionsmöglichkeiten, welche das Spiel wichtig für den Spieler erscheinen lässt, um dadurch die Lernerfolge von Spielern in Lernspielen zu erhöhen [Kickmeier u. a. 2006].

In dieser Arbeit wird viel Bezug auf den Flow-Kanal genommen, denn er bildet die Basis des Adaptivitätsverhaltens welches hier genutzt wird. Durch die konstante Datenerfassung und Berechnung der Adaptivitätsantwort (Abschnitt 3.7) wird dauerhaft darauf geachtet, ob der Nutzer über- oder unterfordert ist. Wie hier beschrieben wird dadurch sicher gestellt, dass der jeweilige Nutzer eine auf ihn zugeschnittene Erfahrung erhält mit dem Ziel sein Engagement und Lernerfolg zu steigern. Durch die Nutzung von xAPI ist es leicht sich auf einzelne Nutzer zu fokusieren und die gesammelten Daten effektiv zu trennen für die Adaptivität unterschiedlicher Nutzer.

3 Grundlegende Konzepte

In diesem Kapitel werden die grundlegenden Konzepte, welche für diese Arbeit genutzt wurden, erklärt. Darunter fallen Serious Games insbesondere Lost Earth 2308 mit dem pädagogischen Agenten LISA, die Unity Game Engine, die Experience API, und die E-Learning A.I. ELAI mit ihrer Adaptivitätsantwort.

3.1 Serious Games

Laut Susi u.a. gibt es viele verschiedene Definitionen für Serious Games [Tarja Susi 2007]. Im Allgemeinen lässt sich jedoch sagen, dass es bei Serious Games nicht nur um das Entertainment geht. Dem stimmt auch Kolja Bopp zu, in dem er sagt, dass Serious Games genutzt werden können um jegliche Lerninhalte zu vermitteln [Bopp 2009]. Serious Games werden in vielen Anwendungsbereichen wie z.B. dem Gesundheitswesen oder dem Militär eingesetzt [Tarja Susi 2007].

Bopp sagt außerdem, dass es möglich ist eine bandbreite von Lerneffekten zu vermitteln, darunter fallen unter anderem feinmotorische Fähigkeiten, wie die Hand-Augen Koordination, kognitive Fähigkeiten, wie z.B. Strategien zur Problemlösung, und soziale Fähigkeiten, wie die Stärkung der Zusammenarbeit und Kommunikation mit Mitmenschen [Bopp 2009]. Mitchell u.a. schließen sich dieser Aussage an, in dem sie der Meinung sind, dass man durch Serious Games sowohl analytische Fähigkeiten als auch das strategisches Denken verbessern kann [Alice Mitchell 2004]. Als Beispiel für das Erlernen von Fähigkeiten zeigte Enochsson u.a., dass Medizinstudenten durch Videospiele besser bei Simulationen mit einem Endoskop abgeschnitten hatten. Was darauf zurück zu führen war, dass die Studenten sich besser im drei-dimensionalen Raum der Simulation zurechtgefunden haben [Enochsson u. a. 2004]. Ein Beispiel für die Förderung der sozialen Kompetenzen von Spielern durch Videospiele ist das Zusammenschließen von Spielern in virtuellen Welten, in denen sie zusammenarbeiten um komplexe Aufgaben zusammen zu erfüllen und voneinander zu lernen [Daniel Livingstone 2006].

Solche Fähigkeiten können laut Bopp jedoch auch von normalen Computerspielen vermittelt werden. Er stellt klar, dass der wesentliche Unterschied zwischen normalen Computerspielen und Serious Games sei, dass ein Serious Games diesen Lernfortschritt anstrebt [Bopp 2009]. Darüber hinaus sagt Susi u.a. außerdem noch, dass es wichtig ist, dass Serious Games den Unterhaltungsfaktor nicht vernachlässigen, da es, wie auch durch den Flow-Kanal in Abbildung 1.1 zu sehen ist, leichter fällt etwas zu lernen, während man unterhalten wird.

Ein Vorteil von Serious Games ist, dass es dem Lernenden ermöglicht Szenarien zu erleben die sonst nicht möglich wären, sei es auf Grund von einer zu großen Gefahr oder

zu hohen Kosten. Dies ermöglicht z.B. die Weiterbildung von Piloten in Simulatoren ohne den enormen Aufwand der durch einen realen Flugzeugstart entstehen würde.

3.2 Lost Earth 2308

Lost Earth 2308 „ist ein 4X Strategie-Spiel für die Luft- und Satellitenbildauswertung“[Atorf 2020]. 4X steht für explore, expand, exploit und exterminate. Lost Earth 2308 basiert auf dem Konzept des Digital Game Based Learning [Atorf 2020]. Das bedeutet, dass der Spieler beim Eintauchen in die digitale Welt alle Lernziele so vermittelt bekommt, wie sie in der Realität benötigt werden. Somit bildet Lost Earth 2308 eine Brücke zwischen einem Unterhaltungsspiel und den Lernzielen. Basierend darauf erreicht Lost Earth 2308 „eine hohe Lernmotivation und nachhaltige Lernerfolge “[Atorf 2020].

Das Spiel ist modular aufgebaut, um sich an die einzelnen Lernziele anzupassen [Atorf 2020]. Die Module des Frameworks beinhalten: Missions, Story, Setting, Assets und Gameplay. Im Falle der Missionen würde man für andere Anwendungsbereiche die Bildauswertemissionen austauschen gegen passende Missionen des neuen Anwendungsbereiches [Atorf 2020].

In diesem Serious Game geht es darum fremde Planeten zu erkunden, zu befreien und währenddessen optische, Infrarot- und Radarbilder auszuwerten [Atorf 2020]. Dabei werden bei der Ausbildung von Bildauswertern folgende Lernziele vermittelt:

1. Grundkenntnisse in der Luft- und Satellitenbilderauswertung (Bildannotationen, Reporting) [Atorf 2020]
2. Verständnis bei den Abläufen des Reconnaissance-Cycle [Atorf 2020]
3. Verständnis der Unterschiede von einzelnen Sensoren [Atorf 2020]
4. Verständnis der Vor- und Nachteile von unterschiedlichen Sensorplattformen, wie Drohnen, Flugzeuge und Satelliten [Atorf 2020]

3.3 Der pädagogische Agent LISA

LISA ist der pädagogische Agent, welcher in Lost Earth 2307 von Marcel Danz implementiert wurde [Danz 2018]. Ein pädagogischer Agent ist ein autonomer Charakter, welcher mit Schülern in einer Lernumgebung lebt, um die Lernerfahrung reichhaltig zu gestalten [W. Lewis Johnson 2015]. LISA bietet dem Spieler hilfreiche Hinweise, um dem Spieler dabei zu helfen die geforderten Aufgaben erfolgreich zu erfüllen. LISA wurde implementiert, um zu überprüfen, ob ein pädagogischer Agent dem Spieler besser helfen kann als ein herkömmliches Tutorial.

LISAs Funktionen umfassen als Beispiel „Fading“, was dafür sorgt, dass im Verlaufe des Spiels über Missionen hinweg immer weniger Hinweise angezeigt werden, außer wenn für neue Aufgaben neues Wissen vermittelt werden muss, und „Modeling“, wodurch LISA dem Spieler die nächsten Schritte hervorheben kann [Danz 2018]. Dabei wird zwischen

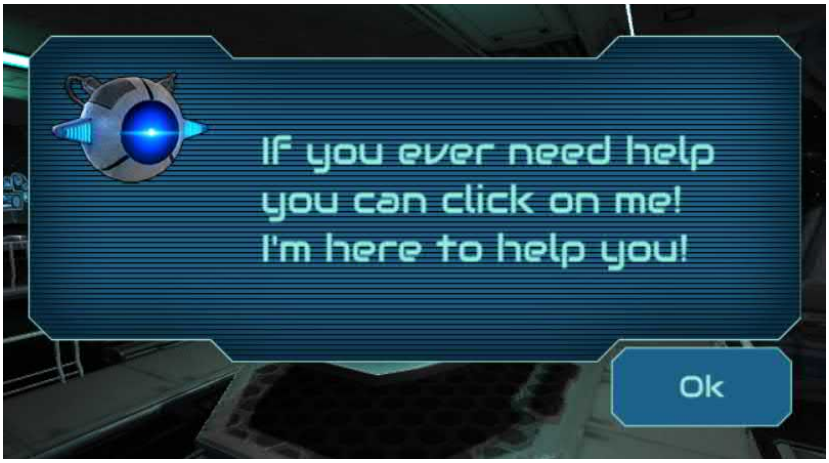

Abbildung 3.1: LISA redet mit dem Spieler über die Message Box

Abbildung 3.2: LISA redet mit dem Spieler über die Sprechblase

blockierenden und nicht-blockierenden Aktionen unterschieden. Bei blockierenden Aktionen wird der Spielfluss unterbrochen um dem Spieler wichtige Informationen mitzuteilen, welche er für das Weiterspielen benötigt. Nicht-blockierende Aktionen hingegen sind beiläufige, während das Spiel voran geht, Hinweise, die dem Spieler lediglich Informationen bieten, welcher er bereits wissen könnte [Danz 2018].

LISA kann darüber hinaus mit dem Spieler interagieren. Zum Interagieren kann sie wie in Abbildung 3.1 dargestellt ist mit dem Spieler über eine Message Box reden, welche eine blockierende Aktion ist. Um einzelne Hinweise dem Spieler zu präsentieren ohne den Spielablauf zu blockieren, kann LISA eine Sprechblase nutzen wie in Abbildung 3.2 zu sehen ist. Letztlich ist es LISA auch möglich verschiedene Spielobjekte für den Spieler hervorzuheben um dem Spieler Hilfe zu bieten beim Erfüllen der Aufgaben. Dies kann man in Abbildung 3.3 sehen.

3.4 Experience API (xAPI)

Die xAPI ist eine Service API für den Umgang mit Activity Streams [Benedek 2013]. Hierbei werden Aktivitäten in sogenannten Statements an ein Learning Record Storage (LRS) übermittelt. Diese Statements setzen sich aus Actor, Verb und Object zusammen,

Abbildung 3.3: LISA hebt die Missionskonsole hervor

können jedoch auch noch weitere Informationen übermitteln, wie z.B. der Kontext oder die Ergebnisse der aktuellen Situation [Kevan u. a. 2016]. Der grundlegende Aufbau eines Statements spiegelt die einfache Frage “Wer macht Was?“wieder. Der Actor beschreibt den Handlungstragenden, welcher eine Aktion ausführt. Hierbei kann es sowohl ein Lebewesen oder eine technische Komponente, z.B. ein System, sein. Das Verb hingegen beschreibt die Aktivität, also was der Actor in diesem Statement mit dem Objekt gemacht hat. Das Verb hilft bei der xAPI zu unterscheiden welche Aktion gerade durchgeführt wurde, um diese sinnvoll zu analysieren. Das Objekt beschreibt eine Komponente auf die sich der Actor bezieht. Dieses einheitliche Format der Statements ermöglicht es Daten über verschiedenste Anwendungen, wie z.B. Serious Games oder Smartphone Apps, zu sammeln [xAPI.com 2020].

Das Ziel von xAPI ist es AICC und SCORM, beides ältere Standards, abzulösen [Benedek 2013]. Der xAPI Standard wurde, durch den Auftrag der Advanced Distributed Learning (ADL) Inititate, von Rustici Software als REST-basierte Open-Source API entwickelt [Benedek 2013; xAPI.com 2020]. Durch den eingesetzten REST Standard wird gewährleistet, dass xAPI durch die Nutzung von HTTP-Aufrufen interoperabel zwischen verschiedenen Anwendungen kommunizieren kann [W3C Working Group 2004]. Die Statements werden im JavaScript Object Notation (JSON) Format versendet. Dabei besteht ein JSON Objekt aus einer Kollektion von Name/Value Paaren und wird wie in Abbildung 3.4 aufgebaut. Ein JSON Objekt startet und Endet mit geschwungenen Klammern. In dem Objekt werden nun durch Kommas getrennte Name/Value Pairs definiert, welche durch einen Doppelpunkt getrennt werden. Values können Strings, Nummern, Objekte, Arrays, Booleans oder null sein [JSON.org 2020]. In Abbildung 3.5 ist ein Beispiel JSON Objekt zu sehen für die Adaptivitätsantwort (Abschnitt 3.7) der ELAI. Hieran kann man leicht erkennen, dass es möglich ist weitere Objekte innerhalb des JSON Objekts zu definieren.

Vorteile der xAPI sind unter anderem, dass es schnell von Entwicklern etabliert wurde, der Fokus auf dem Schüler ist und die Statements lesbar für einen Menschen sind [Kevan u. a. 2016]. Darüber hinaus lassen sich mit xAPI einfach große Datenmengen sammeln, welche es ermöglichen eine genaue Datenanalyse durchzuführen, um dadurch die effek-

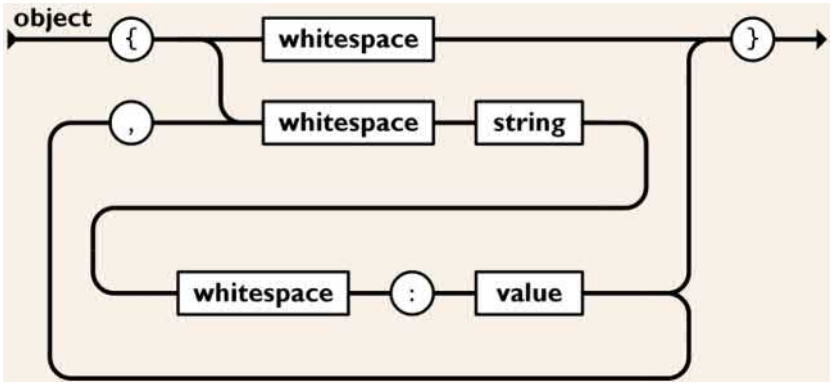

Abbildung 3.4: JSON Objekt Aufbau [JSON.org 2020]


```
{
  "performance": {
    "result": {
      "score": {
        "scaled": 0.9993312177727969
      }
    },
    "display": {
      "en-US": "Performance Score"
    },
    "id": "https://www.wikidata.org/wiki/Q35140"
  },
  "skill": {
    "result": {
      "score": {
        "scaled": 0.45373368796352864
      }
    },
    "display": {
      "en-US": "Skill Level"
    },
    "id": "https://www.wikidata.org/wiki/Q205961"
  },
  "assistance": {
    "result": {
      "score": {
        "scaled": 0.5876524966799679
      }
    },
    "display": {
      "en-US": "Assistance Level"
    },
    "id": "https://www.wikidata.org/wiki/Q1179505"
  }
}
```


Abbildung 3.5: Beispiel JSON Objekt für die Adaptivitätsantwort der ELAI

tivste Art des Lernvorgangs für den aktuellen Nutzer zu bestimmen [Benedek 2013]. Ein weiterer Vorteil ist es, dass die Datenermittlung auch offline durchgeführt werden kann [xAPI.com 2020]. Außerdem ist es einfach die gesammelten Daten eines Nutzers von einem LRS auf ein anderes zu transferieren [xAPI.com 2020].

3.5 Unity Game Engine

Game Engines bilden den Grundstein für Entwickler um ihre Spiele zu erstellen [Game Designing 2020]. Im Allgemeinen bietet eine Game Engine die Funktionalität spiel-relevante Arbeiten, wie Rendering oder Physikberechnung, für die Entwickler zu abstrahieren [Ward 2008; Game Designing 2020]. Game Engines bieten wiederbenutzbare Komponenten, wie z.B. das Laden, Anzeigen und Animieren von Modellen oder auch die Kollisionserkennung. Wieviel und welche einzelnen wiederverwertbaren Komponenten eine Game Engine besitzt und in welchem Ausmaße diese vorhanden sind, variiert zwischen einzelnen Game Engines [Ward 2008; Game Designing 2020]. Game Engines bieten den Vorteil, dass sie es einfacher machen die Visionen der Entwickler Realität werden zu lassen.

Laut Ward existieren drei verschiedene Arten von Game Engines.

Die „roll your own“, bei der der Spieleentwickler seine eigene properitäre Engine entwickelt. Das ist in der Regel mit sehr hohen Kosten und Aufwand verbunden. Diese basieren oftmals auf öffentlichen Bibliotheken wie z.B. DirectX oder OpenGL [Ward 2008]. Ein populäres Beispiel hierfür wäre die Frostbite Engine von Electronic Arts [Electronic Arts 2020]. Ein großer Vorteil von diesen internen Game Engines ist, dass es die größtmöglichste Flexibilität für die Entwickler bietet, da diese genau die Elemente in ihre Game Engine implementieren können, welche sie auch benutzen [Ward 2008] wohingegen bei lizensierten Game Engines leicht zusätzliche Funktionalitäten enthalten sein können, welche ein Entwickler niemals nutzen wird für sein Videospiel.

Die zweite Kategorie von Game Engines laut Ward sind die „mostly-ready“Game Engines, bei denen viele Funktionen bereits implementiert sind. Es wird zwar immernoch ein großer Teil an Eigenprogrammierung erfordert, jedoch im Bezug auf das tatsächliche Spiele, nicht auf die Game Engine selbst [Ward 2008]. Hierrunter fällt z.B. die Unreal Engine [Epic Games 2020].

Als letzte Form der Game Engine gibt es noch die „point-and-click “Game Engine [Ward 2008]. Hierunter fallen Game Engines, bei denen man kaum eigene Programmierkenntnisse benötigt, sondern das meiste per Mausklicks erstellen kann. Hier ist es sogar möglich gesamte Spiele ohne eine einzelne eigene Zeile Programmcode zu erstellen. Laut Ward ist ein populäres Beispiel hierfür Unity3D [Unity 2020].

In dem von GamesRadar publizierten Artikel von OXM Staff [OXM Staff 2018] wurden einige Technikabteilungen von großen Publishern und Spieleentwicklern befragt. Viele der Aussagen von Ward wurden auch hier bestätigt. Darüber hinaus gehen sie auch noch auf sogenannten Middleware ein, welche es den Game Engines ermöglicht modular einzelne Funktionen auszutauschen, wie z.B. die native Physikberechnung gegen die NVidia PhysX Bibliothek zu tauschen um eine bessere Physikberechnung zu erlangen [OXM Staff 2018].

Die Unity Game Engine gehört heutzutage zu den bekanntesten Game Engines [OXM Staff 2018; Game Designing 2020] und bietet Entwicklern die Möglichkeit einfach platt-

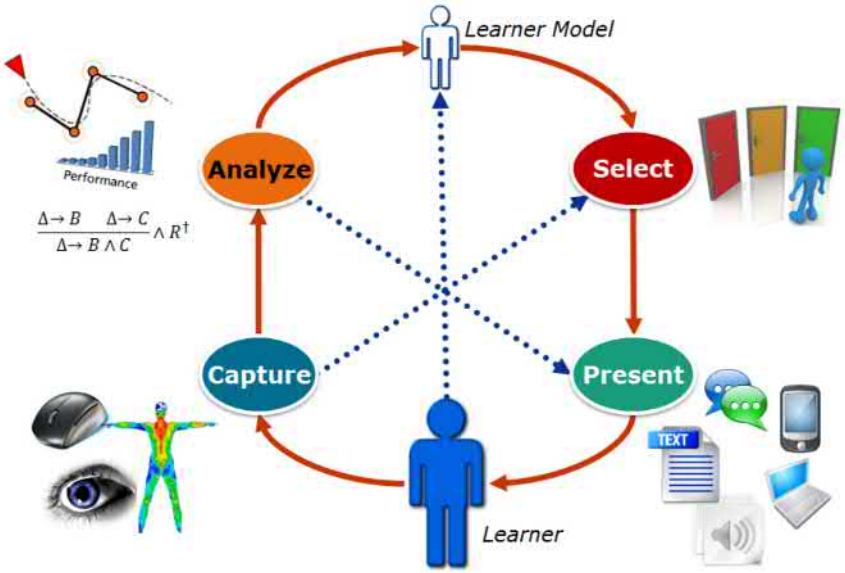

Abbildung 3.6: 4-stufiger Adaptivitätszyklus [Streicher u. a. 2016]

formübergreifende interaktive 2D/3D Videospiele zu entwickeln [Game Designing 2020]. Gerade Indie (Independent, Unabhängig) Entwickler benutzen bevorzugt die Unity Engine für ihre exzellente Funktionalität und der Möglichkeit so gut wie jede Art von Spiel in ihr zu entwickeln [Game Designing 2020]. Darüber hinaus gibt es eine riesige Community, welche zahlreiche informative Tutorials bietet für neue Unity-Entwickler [OXM Staff 2018]. Einige der bekanntesten Unity-Spielen sind Kerbal Space Program [Game Designing 2020], Cuphead und Ori: Will of the Wisp [OXM Staff 2018]. Unity wird sogar von Firmen genutzt, welche direkt nichts mit Spieleentwicklung zu tun haben. Darunter fallen Disney, Coca-Cola, NASA und Lego [OXM Staff 2018].

3.6 E-Learning A.I. (ELAI)

Bei der ELAI handelt es sich um eine Künstliche Intelligenz, welche basierend auf dem in Abbildung 3.6 dargestellten 4-stufigen Adaptivitätszykluses arbeitet. Der 4-stufige Adaptivitätszyklus besteht aus den 4 Phasen Capture, Analyze, Select und Present.

In der Capturephase geht es darum mit Hilfe des ELAI-Adapters, welcher in der grundlegenden ELAI-Architektur in Abbildung 3.7 zu sehen ist, Daten des Spielers während der Interaktion mit dem Spiel zu erfassen und diese weiterzuleiten an die ELAI [Biegemeier 2016]. Die hier gesammelten Daten können unter anderem einfache Tastenbenutzungen oder die Dauer zur Bewerkstelligung einer Aktivität sein.

Die in der Capturephase gesammelten Daten werden in der Analyzephase von der ELAI verarbeitet und aufbereitet [Biegemeier 2016]. Hier wird mit einem zugrundeliegenden Machine-Learning Mechanismus ermittelt wie sich der Spieler aktuell basierend auf den gesammelten Daten zurechtfindet in dem Spiel. In dieser Phase kann es allerdings zum sogenannten Cold-Start Problem kommen, welches die Problematik von fehlenden Daten

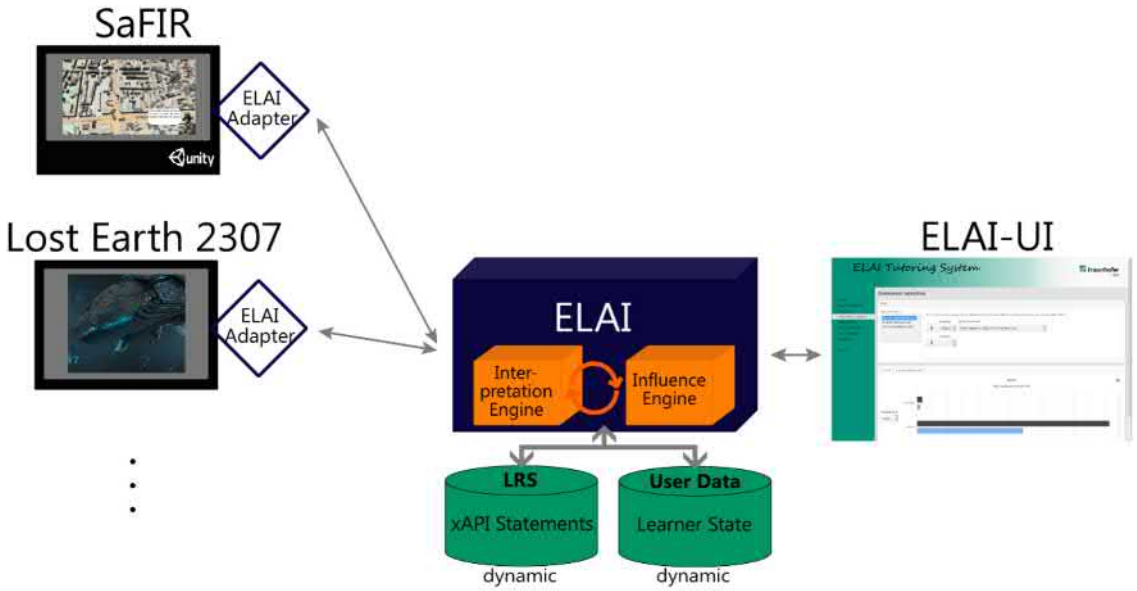

Abbildung 3.7: Grundlegende ELAI Architektur [Biegemeier 2016]

beim ersten Adaptivitätsdurchlaufs beschreibt, weswegen keine qualitative Entscheidung ermittelt werden kann [Biegemeier 2016]. Nachdem die Daten analysiert wurden, werden die neu gefundenen Informationen zum passenden Nutzermodell gespeichert, welche dann in zukünftigen Phasen weiterverwendet werden [Biegemeier 2016].

Basierend auf dem in der Analyzephase ermittelten Nutzermodell wird in der Selectphase entschieden, wann die Adaption des Spiels statt findet und in welcher Form [Biegemeier 2016].

Zuletzt gibt es noch die Presentphase, in der die gewählte Art der Adaption zum gewählten Zeitpunkt im Spiel angewendet wird [Biegemeier 2016]. Hierbei kann es sich um für den Spieler ersichtliche Adaptionen handeln, wie z.B. das Hervorheben von Objekten oder Anzeigen von Hilfetexten durch einen Assistenten. Darüber hinaus können auch unsichtbare Dinge angepasst werden, wie z.B. das Zeitlimit zum Bestehen der Mission kann erhöht werden.

In Abbildung 3.7 ist die grundlegende Architektur der ELAI zu sehen. Der ELAI-Adapter wird zur Kommunikation zwischen den Serious Games und der ELAI verwendet, um Nutzerdaten des Spielers zu sammeln [Biegemeier 2016]. Diese Nutzerdaten werden in der Form von Statements, welche sich aus Actor, Verb und Object zusammensetzen, gesammelt und in einem Learning Record Storage (LRS) gespeichert. In Abbildung 3.9 sind die von der ELAI definierten Verben und in Abbildung 3.8 die von der ELAI definierten Aktivitäten zu sehen.

Basierend auf diesen definierten Verben und Aktivitäten unterscheidet die Interpretation-Engine die einzelnen Statements die von den Adaptern an die ELAI gesendet werden [Biegemeier 2016]. Die Interpretation-Engine beschreibt somit einen Wortschatz, „welcher die Aktionen des Lerners Maschinen verständlich beschreibt“[Biegemeier 2016].

Name (en-US)	URI
Game	http://iosb.fraunhofer.de/ias/ibis/elai/xapidefinitions/activity/game/
Task	http://iosb.fraunhofer.de/ias/ibis/elai/xapidefinitions/activity/task/
Help	http://iosb.fraunhofer.de/ias/ibis/elai/xapidefinitions/alert/help/
Alert	http://activitystrea.ms/schema/1.0/alert/

Abbildung 3.8: Definierte Aktivitäten [Biegemeier 2016]

Name (en-US)	URI
started	http://iosb.fraunhofer.de/ias/ibis/elai/xapidefinitions/verb/started
terminated	http://activitystrea.ms/schema/1.0/terminated/
launched	http://adlnet.gov/expapi/verbs/launched/
completed	http://adlnet.gov/expapi/verbs/completed/
played	http://iosb.fraunhofer.de/ias/ibis/elai/xapidefinitions/verb/played
asked	http://adlnet.gov/expapi/verbs/asked/
performed	http://iosb.fraunhofer.de/ias/ibis/elai/xapidefinitions/verb/performed

Abbildung 3.9: Definierte Verben [Biegemeier 2016]

Die Influence-Engine kann mit Hilfe des, durch die Interpretation-Engine erstellten, Wortschatzes Verhaltensregeln definieren, welche es ermöglichen das Spiel situationsbedingt zu adaptieren [Biegemeier 2016].

Dazu gibt es noch die ELAI-UI, welche eine „systemübergreifende Webschnittstelle für die Analyse von Nutzungsdaten und Adaption von Spielelementen [ist] “[Biegemeier 2016].

3.7 Adaptivitätsantwort

Für die ELAI wurde ein eigenes Format der „Antwort “von Alexander Streicher entwickelt basierend auf der Vorarbeit von Herrn Biegemeier [Streicher u. a. 2019; Biegemeier 2016], welche sich stark am Format von xAPI orientiert. Das Ziel dieses Formates ist es eine größtmögliche Generalität mit anwendungsspezifischen Erweiterungen zu kombinieren, damit andere Assistenzsysteme diese Antwort schnell und einfach verwenden können, um die Interoperabilität zu maximieren. Die Adaptivitätsantwort besteht aus drei quantitativen Bewertungsgrößen: dem Performance Score, Assistenzlevel und dem Skill Level. Alle diese Werte werden quantitativ zwischen null und eins (als Intervall: [0;1]) und für einen gegebenen Kontext zu einem bestimmten Zeitpunkt angegeben.

1. Der Performance Score gibt den Fortschritt des Spielers an und wird basierend auf vorherigen Beobachten dynamisch berechnet.

2. Das Assistenzlevel gibt den Unterstützungsbedarfs des Spielers an, welcher manuell von einem Tutor eingestellt werden kann oder von der ELAI berechnet wird. Dieser Wert orientiert sich am Performance Score für die Berechnung des Assistenzlevels.

3. Das Skill Level gibt den geschätzten Kompetenzlevel des Spielers an, welcher unabhängig vom Performance Score und Assistenzlevel berechnet werden kann. Die Berechnung basiert auf langfristigen und qualifizierenden Beobachtungswerte.

Ein Beispiel für eine Adaptivitätsantwort ist in Abbildung 3.5 zu sehen.

4 Entwurf

In dieser Arbeit besteht das Grundkonzept aus zwei Komponenten. Die erste Komponente ist der adaptive Assistent LISA innerhalb von Lost Earth 2308, welcher die Kommunikation direkt mit der ELAI durchführt und basierend auf dessen Antworten das Lernspiel adaptiert. Im Kontext dieser Arbeit ist ein adaptiver Assistent, eine Entität, welche dem Spieler zur Hilfe steht und basierend auf bestimmten Faktoren das Spiel anpasst, um dem Spieler das Erfüllen der Aufgaben zu vereinfachen. Die zweite Komponente bildet der sogenannte ELAIControlService, welcher die ELAI dabei unterstützt eine passende Entscheidung basierend auf den vorliegenden Daten zu treffen.

In Abbildung 4.1 ist der grundlegende Entwurf dieser Arbeit zu sehen. Dabei sendet Lost Earth 2308 xAPI Statements (Abschnitt 3.4) an den Learning Record Storage. Der adaptive Assistent LISA verwaltet den Adaptivitätsteil von Lost Earth 2308, in dem LISA eine Anfrage nach einer neuen Adaptivitätsantwort startet, welche von der ELAI an den ELAIControlService weitergeleitet wird. Der ELAIControlService wird nun basierend auf den Informationen von LISA und den gesammelten xAPI Statements des LRS das neue Assistenzlevel berechnen und zurücksenden.

In Abbildung 4.2 ist ein vollständiges Aktivitätsdiagramm für eine Spieleranfrage zu sehen. Das Aktivitätsdiagramm wurde in drei getrennte Abschnitte unterteilt, welche in Abbildung 4.5, Abbildung 4.6 und Abbildung 4.9 einzeln dargestellt sind. Die Aktivitätsdiagramme in Abbildung 4.5 und Abbildung 4.6 werden in Abschnitt 4.1 und das Aktivitätsdiagramm in Abbildung 4.9 wird in Abschnitt 4.2 genauer erklärt.

In Abbildung 4.3 ist ein vollständiges Sequenzdiagramm für eine Spieleranfrage zu sehen. Die linke Hälfte wird in Abschnitt 4.1 und die rechte Hälfte in Abschnitt 4.2 erklärt.

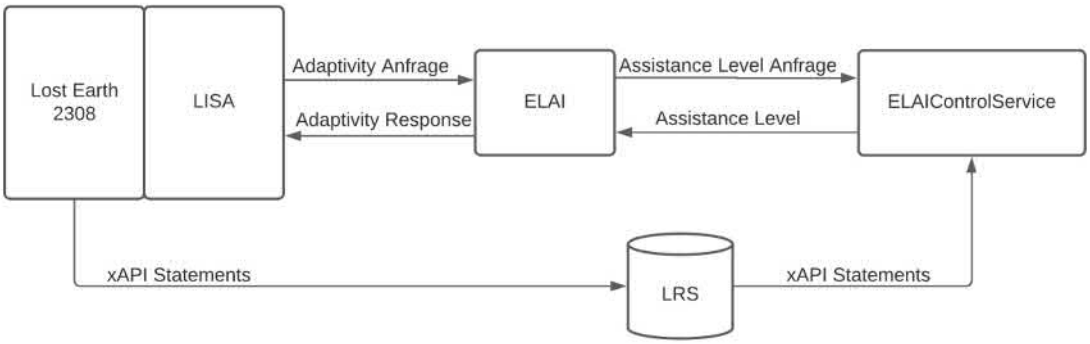

Abbildung 4.1: Grundlegender Entwurf

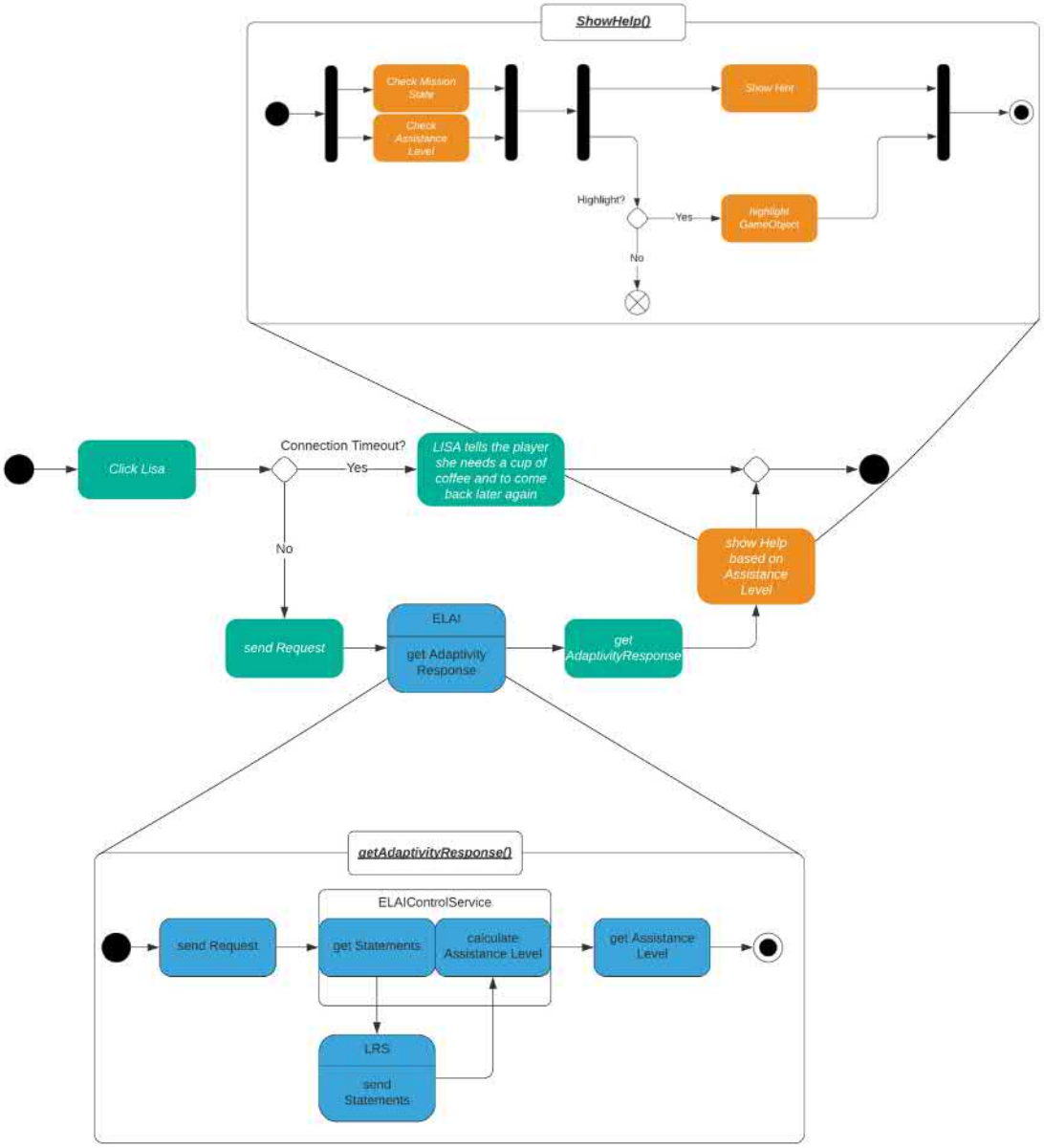

Abbildung 4.2: Komplettes Aktivitätsdiagramm einer Spieleranfrage. Detailiert zu betrachten in Abbildung 4.5, Abbildung 4.6 und Abbildung 4.9

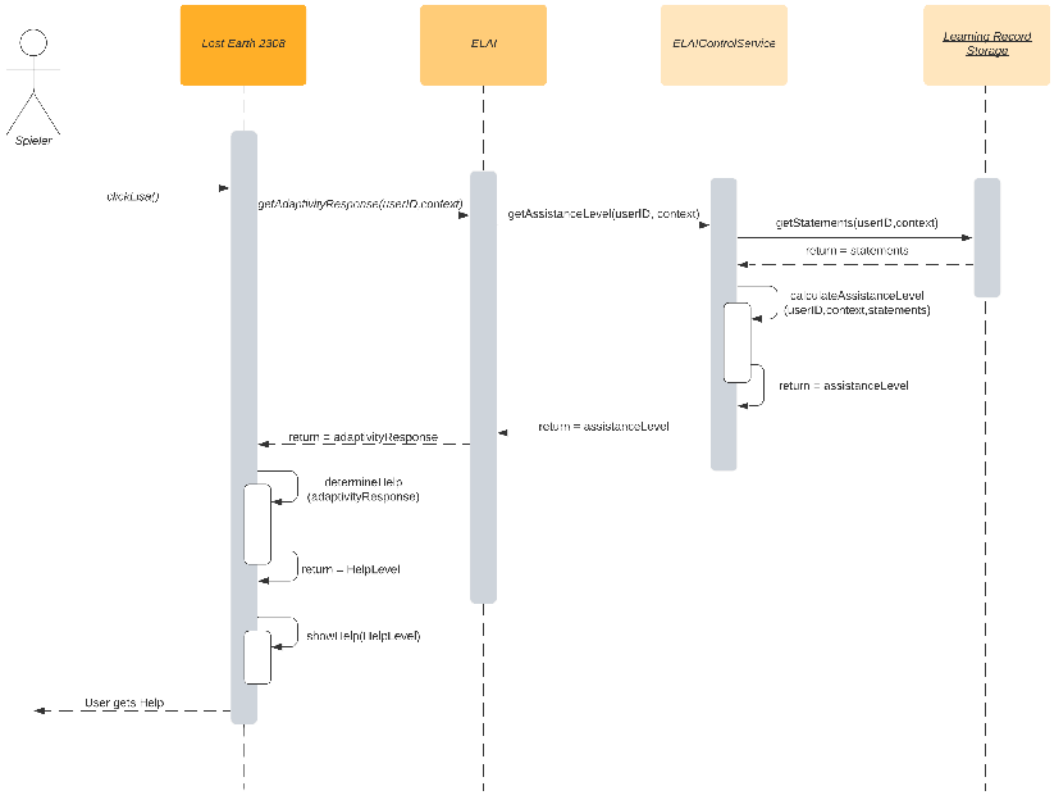

Abbildung 4.3: Gesamtes Sequenzdiagramm für eine Spieleranfrage

4.1 Entwurf des adaptiven Assistenten LISA

Der adaptive Assistent LISA wird in Unity 5.6.7f1 entwickelt. Dazu wird auf vorherige Arbeiten zurückgegriffen und diese um die weitere Funktionalität erweitert. Die Programmierung findet in C# statt. Um den gewünschten Effekt mit Hilfe der adaptiven LISA zu erfüllen, werden diverse neue Funktionalitäten benötigt. Darunter fallen:

1. das Aufzeichnen und Speichern von Fehlschlägen des Spielers, welche später dafür genutzt werden das neue Assistenzlevel zu berechnen
2. die Herstellung einer Verbindung zur ELAI, um eine Adaptivitätsantwort zu erfragen
3. Das Assistenzlevel aus der Adaptivitätsantwort zu extrahieren
4. Darstellen von Hinweisen, welche variieren basiernd auf dem momentanen Assistenzlevel
5. Hervorheben von Objekten basierend auf dem momentanen Assistenzlevel

Die adaptive LISA ist so konzipiert, dass diese die Anfrage an ELAI und die Adaptierung des Spiels eigenständig vornimmt. Für die Anfrage an die ELAI sammelt sich LISA während des Spiels Informationen über den Spieler. Darunter fällt die UserID, welche beim Start des Spiels festgelegt wird, der momentane Missionszustand und wie oft der Spieler bereits in diesem Zustand war und die Aufgabe nicht korrekt durchgeführt hat. Diese Daten werden dann wie in Abbildung 4.4 dargestellt an die ELAI gesendet und für die Weiterleitung an den ELAIControlService genutzt (s. Abschnitt 4.2).

Eine Spieleranfrage wird durch einen Klick auf LISA, wie in Abbildung 4.5 zu sehen ist, gestartet. Dadurch wird versucht eine Verbindung mit der ELAI aufzubauen und die Anfrage, welche den momentanen Missionszustand, die Anzahl der Fehler für die momentane Aufgabe und den Nutzer übermittelt, nach einer Adaptivitätsantwort gestellt. Wie die ELAI die Adaptivitätsantwort ermittelt können Sie in Abschnitt 4.2 nachlesen. Die erhaltene Adaptivitätsantwort wird wie in Abbildung 4.4 zu sehen ist nun von LISA genutzt um das neue Assistenzlevel zu ermitteln.

Basierend auf diesem Assistenzlevel und dem momentanen Missionszustand wird nun, wie Sie Abbildung 4.6 entnehmen können, ermittelt wie LISA reagieren muss. Dabei kann LISA nun Hinweise darstellen und gleichzeitig Spielobjekte hervorheben. LISA wurde so entworfen, dass sie je nach Höhe des Assistenzlevels unterschiedliche Textnachrichten ausgibt, die einen variierenden Grad an Hilfe bieten. Darüber hinaus werden Spielobjekte nur hervorgehoben, wenn das Assistenzlevel hoch genug ist. Basierend auf dem Missionszustand werden unterschiedliche Arten von Hinweisen gegeben. Dabei regelt der Missionszustand, was für Hinweise gegeben werden und das Assistenzlevel wie viel Hilfe diese bieten. In Abbildung 4.7 können sie 4 unterschiedliche Hinweise sehen, welche LISA während der Aufgabe des Sensorstarts anzeigen kann. Diese Hinweise werden basierend auf dem Assistenzlevel ausgewählt, wobei der 1. Hinweis bei einem sehr hohen Assistenzlevel ausgegeben wird, da der Spieler hier am meisten Hilfe benötigt, und der 4. Hinweis ausgegeben wird, wenn der Spieler sich bereits gut auskennt und damit ein sehr niedriges Assistenzlevel vorhanden ist.

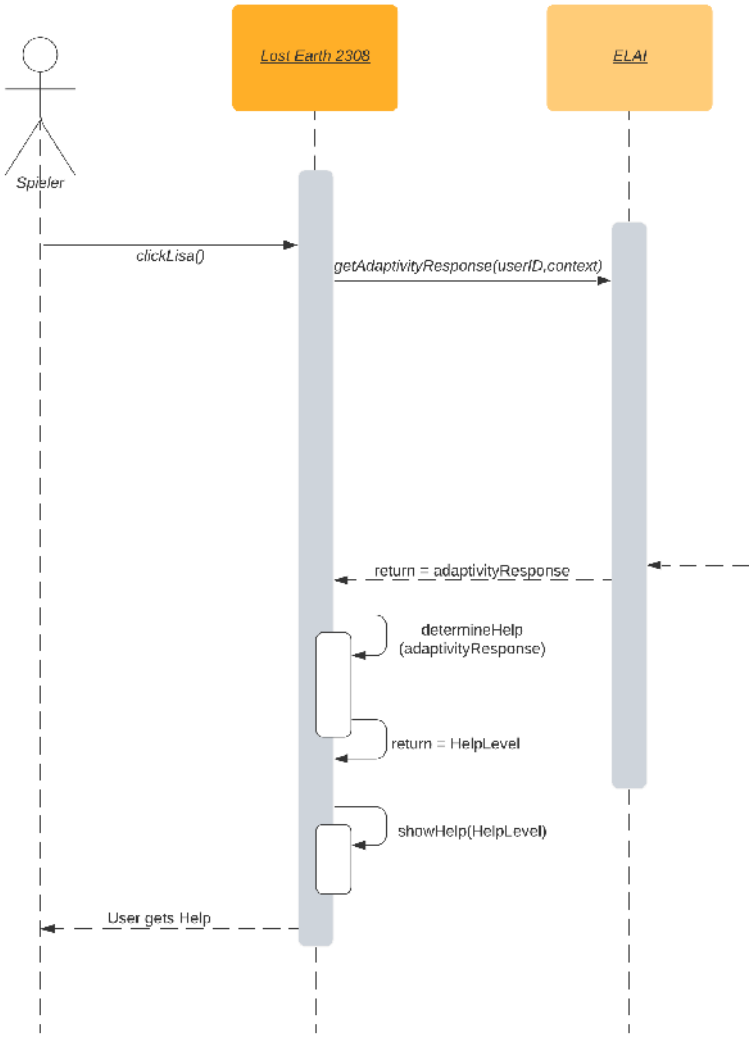

Abbildung 4.4: Ablauf einer Anfrage des Spielers um Hilfe

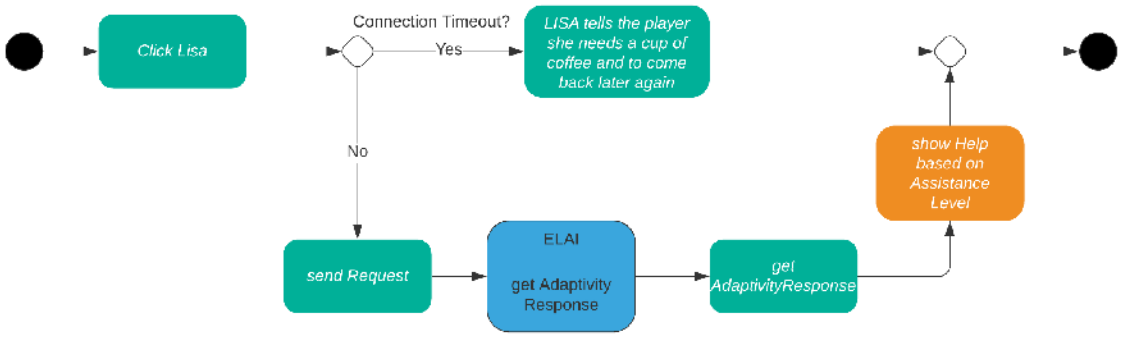

Abbildung 4.5: Aktivitätsdiagramm für eine Hilfsanfrage eines Spielers

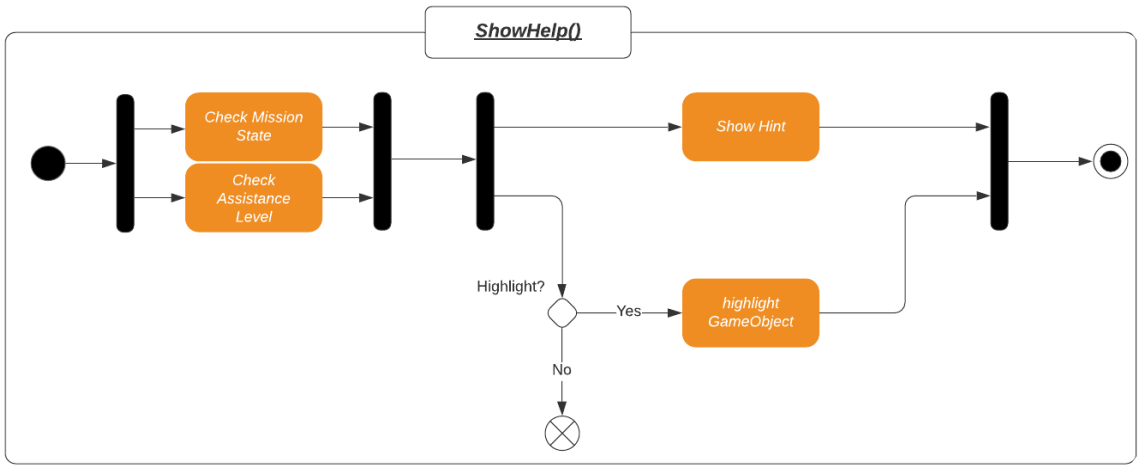

Abbildung 4.6: Aktivitätsdiagramm für das Zeigen von Hilfselementen

```
<Message>Before launching a sensor with the Task Sensor Console you should check the weather in the
         Weather Console and delay the sensor so it doesn't scan during bad weather.</Message>

<Message2>To launch a sensor you should interact with the Task Sensor Console.</Message2>

<Message3>You should launch a sensor.</Message3>

<Message4></Message4>
```

Abbildung 4.7: Beispiel für die unterschiedlichen Hinweise die LISA basierend auf dem Assistenzlevel bietet

4.2 Entwurf des ELAIControlService

Der ELAIControlService wird in PyCharm Professional 2020.2.2 als Python Flask Webservice entwickelt. Der Webservice wird mit Python 3.8.5 entwickelt,

Zu Beginn der Arbeit sollte der ELAIControlService, lediglich eine einfache Python Flask Webpage sein, welche mit Hilfe von dem Drücken unterschiedlicher Knöpfe der ELAI ein Assistenzlevel sendet, welches dann für die Adaptivitätsantwort genutzt wurde. Dieser erste Entwurf hätte somit aber wenig Interoperabilität aufgewiesen. Deshalb wurde der ELAIControlService erweitert um die Aufgaben:

1. Anfrage von ELAI erhalten mit den weitergeleiteten Werten von LISA
2. xAPI Statements aus einem Learning Record Storage holen und für den jeweiligen User mit Hilfe der UserID relevante Statements filtern
3. Basierend auf den weitergeleiteten Werten wird mit Hilfe eines Entscheidungsbaums ein Assistenzlevel ermittelt
4. Basierend auf den gefilterten xAPI Statements wird das Assistenzlevel angepasst
5. Berechnetes Assistenzlevel zurück an ELAI senden

Der ELAIControlService wurde dafür entworfen, der ELAI bei der Berechnung der AdaptivityResponse beizustehen. Momentan gibt es noch keine KI-Komponente in der ELAI. Um diese zu Ersetzen wird die ELAI die Anfrage von LISA an den ELAIControlService weiterleiten. Der ELAIControlService wird dann, wie in Abbildung 4.8 und Abbildung 4.9 zu sehen ist, alle Statements des Learning Record Storages holen und basierend auf dem Kontext und dem Nutzer einzelne Statements nutzen um mit diesen, dem Kontext und des Nutzers ein Assistenzlevel mit Hilfe eines Entscheidungsbaumes zu berechnen. Dieses neu berechnete Assistenzlevel wird dann an die ELAI zurückgesendet.

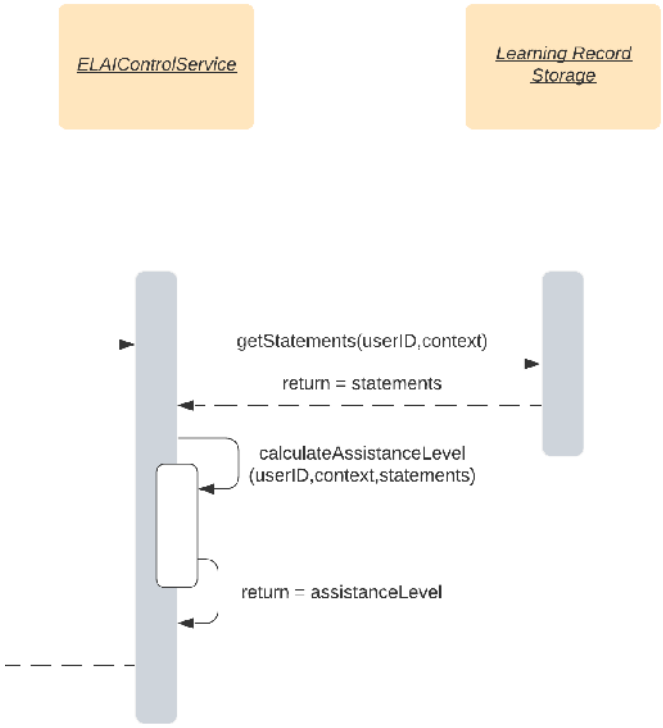

Abbildung 4.8: Berechnung des neuen Assistenzlevels

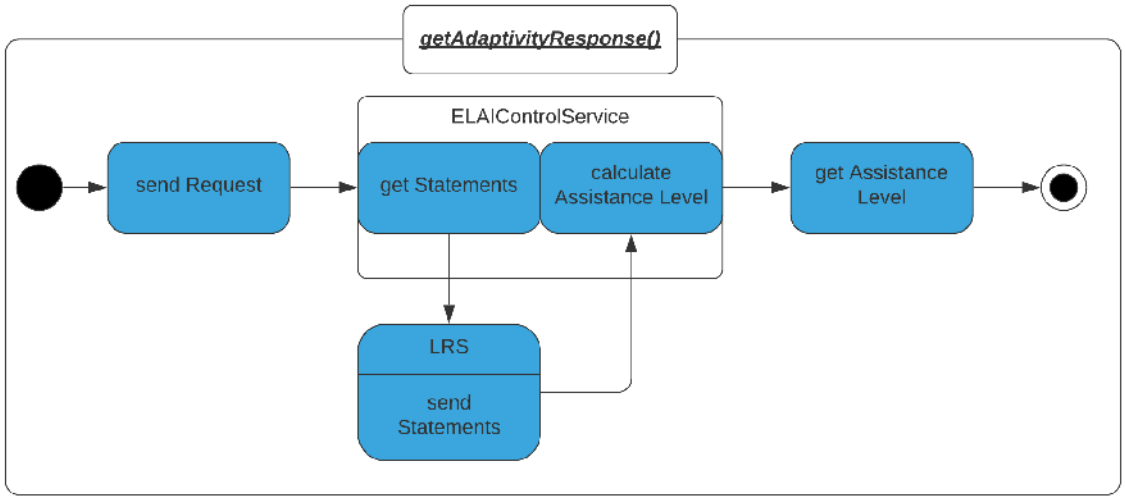

Abbildung 4.9: Aktivitätsdiagramm für das Erhalten einer Adaptivitätsantwort

4.3 Szenario

Dieses Beispielszenario wurde zu Beginn der Arbeit entworfen, um einen Richtwert für das Endprodukt zu haben. Hierbei wurde ein Durchlauf der ersten Lost Earth 2308 Mission im LISA-Modus genommen und mit Hilfe von Mockups bearbeitet. Da die erste Mission im ursprünglichen Design nur klares Wetter beinhaltet, wurde dies für dieses Beispielszenario und die Implementierung angepasst. Mit Hilfe des Szenarios wurden Rahmenpunkte ermittelt, an welchen Stellen die Adaptivität durch LISA mit Hilfe der Unterstützung der ELAI und des ELAIControlServices angewendet werden soll und in welchem Ausmaße.

Abbildung 4.10: Startszene von Lost Earth 2308

Zu Beginn des Spiels landet der Spieler auf dem Übersichtsbild der Arche (zu sehen in Abbildung 4.10. Hier kann der Spieler in einzelne Räume hinein. Der Spieler wird in diesem Bildschirm einen Hinweis von LISA erhalten, dass er sich zu der Brücke begeben und mit Kasparov reden soll, um mit seiner Mission zu beginnen.

Auf der Brücke angekommen wird Kasparov bereits mit einem gelb leuchtenden Ausrufezeichen markiert, wie in Abbildung 4.11 zu sehen ist. Der Spieler kann nun mit ihm, durch einen Mausklick auf ihn, interagieren und erfährt einige Details über das Spiel.

LISA gibt dem Spieler einen Hinweis, dass er die erste Mission von der Missionsliste aus starten kann und hebt diese hervor (zu sehen in Abbildung 4.12). In dem User Interface der Missionsliste werden einige grundlegende Informationen der Mission aufgelistet. Darunter fällt eine kurze Beschreibung, die Ziele und die Kosten der Mission. Des weiteren kann durch die einzelnen Missionen gewechselt werden. Das User Interface ist in Abbildung 4.13 zu sehen. Durch einen Klick auf Start wird die Mission gestartet.

Abbildung 4.11: Isaac Kasparov auf der Brücke in Lost Earth 2308

Abbildung 4.12: Missionsliste, welche durch LISA hervorgehoben wird

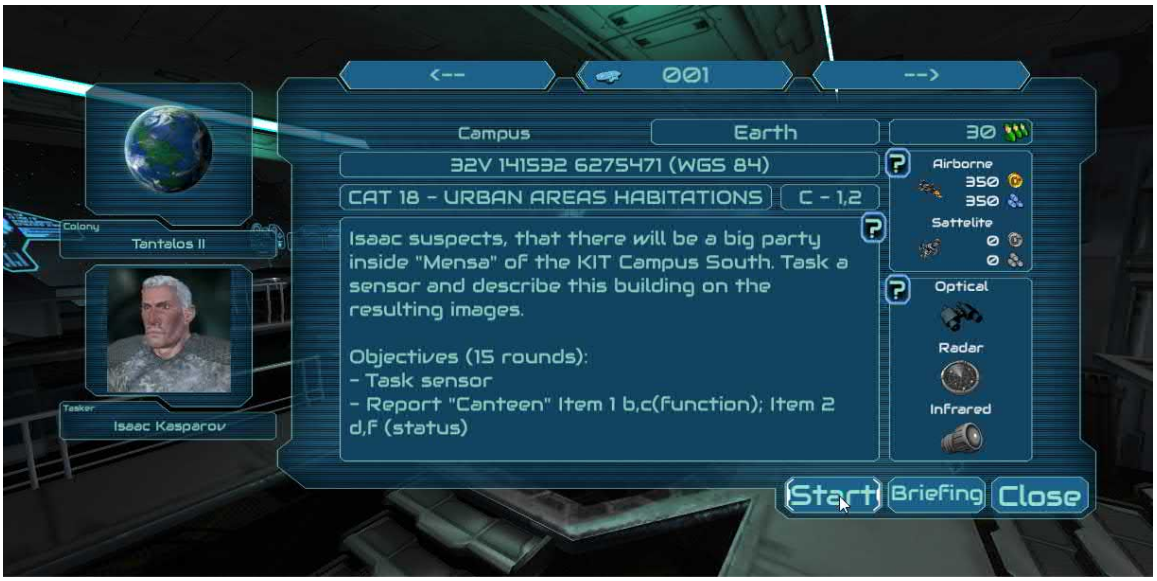

Abbildung 4.13: User Interface der Missionsliste

Abbildung 4.14: User Interface der Select und Task Sensor Konsole

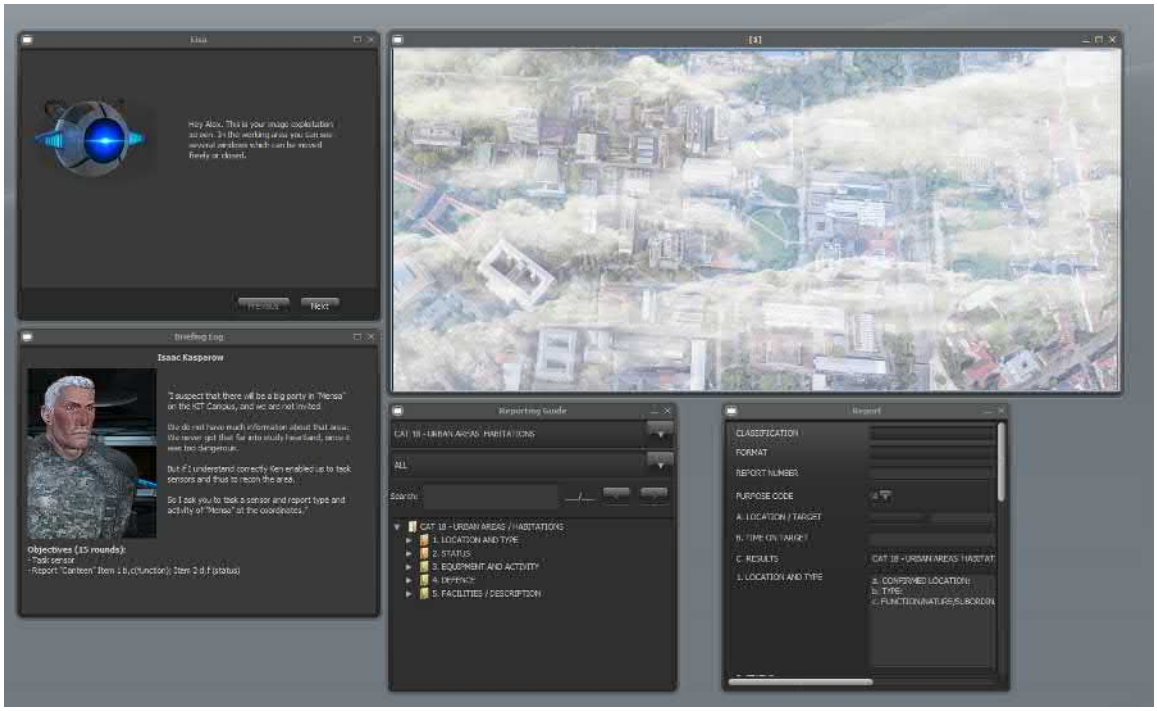

Abbildung 4.15: Auswertungsprogramm ViLand zeigt Wolkendecke über dem Sensorbild

Der Spieler begibt sich in den Hangar von dem aus er einen Sensor starten lassen kann um Satellitenbilder aufzunehmen. Durch die Unerfahrenheit des Spielers und ohne Hilfe von LISA wird der Spieler nicht wissen, dass es bei einem Sensorstart von äußerster Wichtigkeit das Wetter vorher zu prüfen. Aus diesem Grund wird der Spieler den Sensor einfach starten. Das User Interface der Select und Task Sensor Konsole kann man in Abbildung 4.14 sehen. Durch einen Klick auf Launch wird der Sensor gestartet.

Nach dem der Spieler die Zeit gewartet hat bis der Sensor mit den Bildern zurückkommt, wird er zur Brücke gehen müssen um die Sensorbilder auszuwerten. Dies geschieht im Exploitation Table, welcher die externe Auswertungsanwendung ViLand startet. Das Ziel des Spielers ist es in ViLand sich die Satellitenbilder anzuschauen und den beigefügten Fragebogen auszufüllen. Da der Spieler den Sensor nicht verzögert hat, ist das Satellitenbild mit Wolken bedeckt. Dadurch ist der Spieler nicht in der Lage den Fragebogen korrekt auszufüllen. Sobald der Spieler den Fragebogen fertig ausgefüllt hat kann er diesen zurück an das Spiel senden, um diesen auszuwerten.

Wie in Abbildung 4.16 zu sehen ist, ist die Mission fehlgeschlagen, da der Spieler den Fragebogen nicht richtig ausfüllen konnte, aufgrund der Tatsache, dass das Satellitenbild mit Wolken bedeckt war. Im selben Zuge wird im Hintergrund der Fehlschlag des Spielers aufgezeichnet und der ELAI mitgeteilt. Durch diesen Fehlschlag der ersten Mission wird das Assistenzlevel des Spielers hochgestuft, weshalb LISA beginnt dem Spieler Hilfe zu geben.

Abbildung 4.16: Gescheiterte Mission

Abbildung 4.17: Adaptierte Task und Launch Sensor Konsole

Der Spieler startet die Mission somit erneut und verfolgt die selben Schritte wie zuvor, bis er in den Hangar kommt um den Sensor zu starten. Wenn der Spieler in die Task und Launch Sensor Konsole geht bekommt der Spieler nun, wie in Abbildung 4.17 zu sehen ist, einen Hinweis von LISA, dass der Spieler auf das Wetter achten sollte bevor er einen Sensor startet. Daraufhin begibt sich der Spieler zur Wetterkonsole im Hangar.

Bei der Wetterkonsole angekommen, teilt LISA dem Spieler mit, dass er keinen Sensor starten sollte, wenn dieser ein Satellitenbild, während es bewölkt ist, aufnehmen würde. Im selben Zuge werden die Tage mit bewölktem Wetter hervorgehoben, wie in Abbildung 4.18

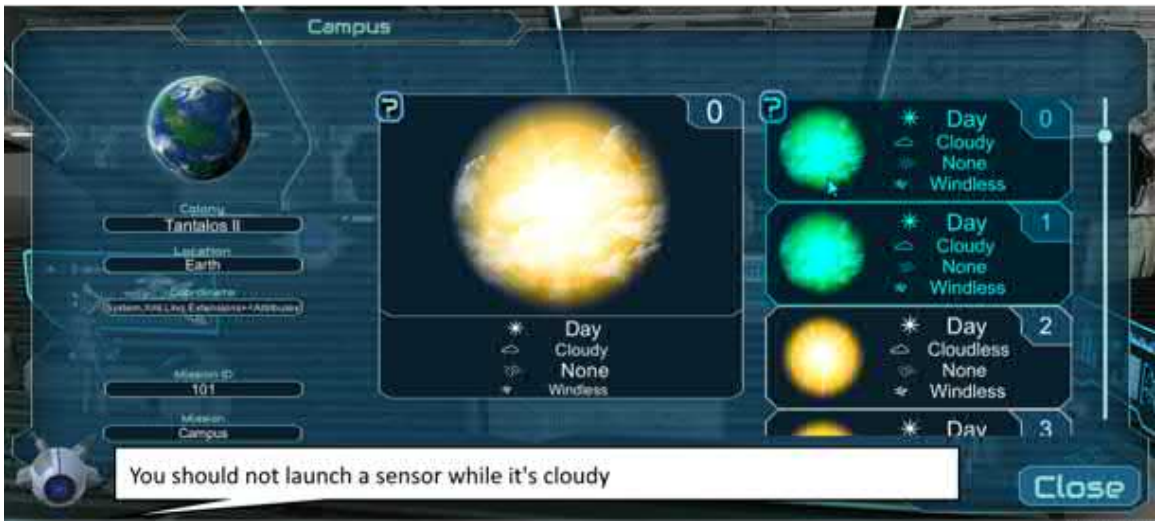

Abbildung 4.18: Adaptierte Wetterkonsole mit Hervorhebung der bewölkten Tage

zu sehen ist. Somit hat der Spieler das Verständnis erlangt, dass er einen Sensor nicht starten sollte, wenn das Wetter nicht klar ist.

Abbildung 4.19: Adaptierte Task and Launch Sensor Konsole mit hervorgehobener Verzögerung

Mit dem neu erlangten Wissen begibt sich der Spieler nun wieder zur Task und Launch Sensor Konsole. In Abbildung 4.19 ist zu sehen, dass LISA dem Spieler nun eine Nachricht mitteilt, dass dieser den Sensor verzögern sollte und dies über den hervorgehobenen Teil des User Interfaces machen kann.

Nun würde der Spieler sich wieder auf die Brücke begeben, um das Bild auszuwerten. In ViLand angekommen ist das Satellitenbild nun klar erkennbar, wie in Abbildung 4.20

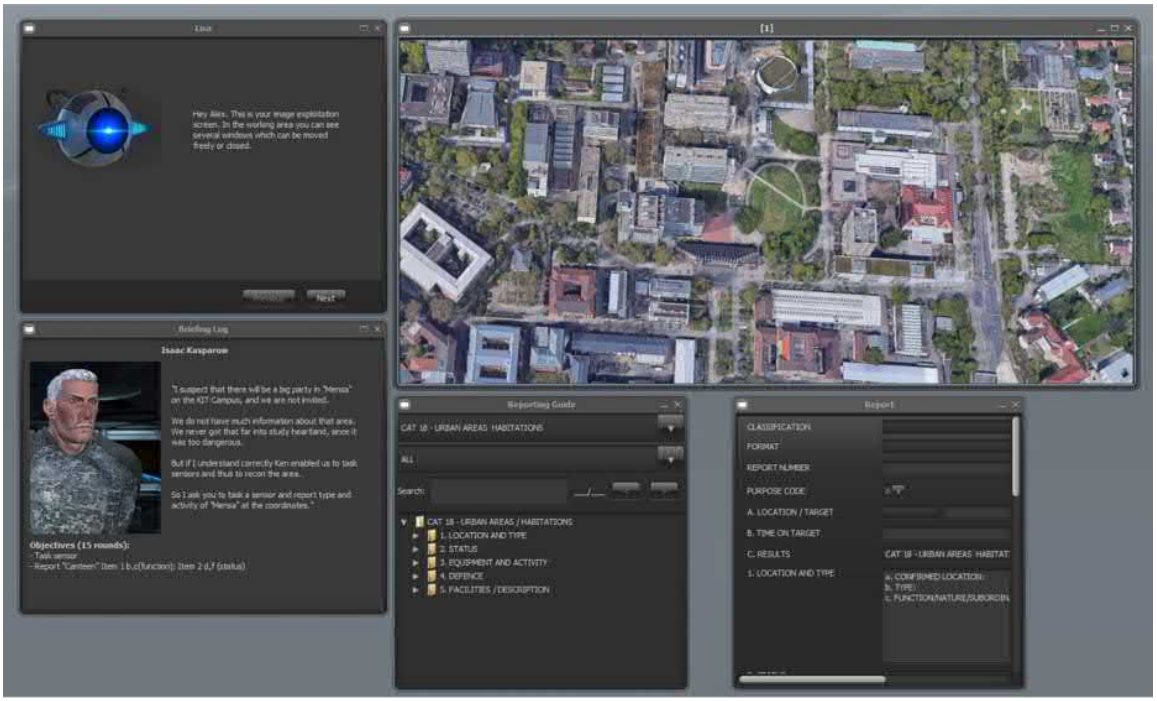

Abbildung 4.20: korrektes Satellitenbild in ViLand

zu sehen ist. Der Spieler kann somit die Aufgabe ohne Probleme erfüllen und schließt die Mission wie in Abbildung 4.21 erfolgreich ab.

Abbildung 4.21: Erfolgreiche Mission

5 Implementierung

In diesem Kapitel wird das Ergebnis der Implementierungsarbeit vorgestellt. In Abschnitt 5.1 wird der erste Prototyp dargestellt, welcher als proof of concept genutzt wurde. Danach wird der adaptive Assistent LISA in Abschnitt 5.2 und der ELAIControlService in Abschnitt 5.3 gezeigt.

5.1 Erster Protoyp

Zu Beginn der Arbeit wurde ein Prototyp entwickelt, welcher als proof of concept galt. Dieser Prototyp bestand aus drei Einzelteilen. Als erstes wurde ein neues Unity Projekt erstellt, welches einen LISA-Dummy beinhaltet. Dazu wurde dann in Java eine kleine Netzwerkschnittstelle implementiert, welche die ELAI simuliert und zu guter letzt noch eine in Python Flask realisierte Webpage, welche, wie in Abschnitt 4.2 erklärt, zu Beginn mit Hilfe von Knöpfen die Antwort der Netzwerkschnittstelle steuern sollte.

5.1.1 Unity LISA-Dummy

In Abbildung 5.1 erkennt man, dass die Unity-Szene einfach aufgebaut wurde um das Grundkonzept zu zeigen. Sie besteht aus dem LISA-Dummy, welcher eine dunkle Kugel ist, einer Textbox und einem dunkelgrauen Hintergrund. Der LISA-Dummy besitzt die Möglichkeit die Textbox anzusprechen, um darüber die Adaptivität zu zeigen. Die Adaptivität wird dadurch präsentiert, dass der Dummy basierend auf dem Assistenzlevel unterschiedliche Nachrichten darstellt. Dies soll darstellen wie LISA in der endgültigen Implementierung dem Spieler basierend auf dem Assistenzlevel varrierend hilfreiche Hinweise geben kann. Sobald der Dummy gedrückt wird, wird im Hintergrund eine Anfrage an die Netzwerkschnittstelle gestellt (in Abbildung 5.2 zu sehen) und es öffnet sich die Textbox mit variierenden Nachrichten.

Der LISA-Dummy stellt einen neuen *UnityWebRequest* an die Netzwerkschnittstelle und sendet mit der Anfrage einen String, welcher der Netzwerkschnittstelle mitteilt, dass der Dummy gerade die Anfrage schickt. Sobald die Anfrage der Netzwerschnittstelle zurückkommt, wird der Assistenzlevel ausgelesen und benutzt, um darauf basierend verschiedene Texte in der Textbox auszugeben. Damit konnte das Konzept gezeigt werden, dass es möglich ist das Unity Spiel basierend auf dem Assistenzlevel zu adaptieren. In diesem Fall wird zwischen einem Assistenzlevel von 1.0, bei dem der Spieler völlige Unterstützung benötigt, 0.0, bei dem der Spieler erfahren genug ist und keine Hilfe benötigt und Werten dazwischen, bei denen der Spieler etwas Hilfe benötigt, unterschieden.

Abbildung 5.1: LISA-Dummy in Unity

```
IEnumerator Upload()
{
    var uwr = new UnityWebRequest("http://localhost:8080/Interface_war/MyServlet", "POST");
    byte[] jsonToSend = new System.Text.UTF8Encoding().GetBytes("{sender: UnityGame, data: X}");
    uwr.uploadHandler = (UploadHandler)new UploadHandlerRaw(jsonToSend);
    uwr.downloadHandler = (DownloadHandler)new DownloadHandlerBuffer();
    uwr.SetRequestHeader("Content-Type", "application/json");

    //Send the request then wait here until it returns
    yield return uwr.Send();

    if (uwr.isError)
    {
        Debug.Log(uwr.error);
    }
    else
    {
        double value = double.Parse (uwr.downloadHandler.text, System.Globalization.CultureInfo.InvariantCulture);
        if (value == 1.0) {
            theText.text = "Your current Assistance Level is 1.0";
        } else if (value == 0.0) {
            theText.text = "You seem to be doing just fine!";
        } else {
            theText.text = "I think you still need some help";
        }
        Debug.Log(uwr.downloadHandler.text);
    }
}
```

Abbildung 5.2: Programmcode: Anfrage an die Netzwerkschnittstelle

5.1.2 Python Flask Webpage

Die in Python Flask erstellte Webpage sollte als Ersatz für die künstliche Intelligenz in der ELAI dienen. Der Zweck hierfür war, dass durch diese Webpage das Spiel Lost Earth 2308 gezielt adaptiert werden kann. Dieses Konzept der Webpage war dafür gedacht, dass bei späterer Anwendung der Tutor, welcher dem Schüler zur Seite steht, gezielte Änderungen am Assistenzlevel vornehmen konnte, basierend auf den Spielgeschehnissen des Schülers. In der fertigen Implementierung wurde der menschliche Tutor durch einen Entscheidungsbaum ersetzt.

Die Webpage wurde zu Beginn der Arbeit ELAISteeringService bezeichnet, da die Hauptfunktion der Webpage war die ELAI zu steuern. Dazu sollte die Webpage der Netzwerkschnittstelle mitteilen, welches Assistenzlevel sie zurückgeben sollte sobald sie von LISA angefragt wird. Für den Prototyp wurde ein einfaches Konzept gewählt. Dafür wurde eine einfache Webpage erstellt, wie in Abbildung 5.3 zu sehen ist. Diese Webpage beinhaltet drei Knöpfe, welche durch Drücken eine Anfrage an die Netzwerkschnittstelle senden. Je nachdem welcher Knopf gedrückt wurde, wurde ein anderes Assistenzlevel gesendet. Da dies nur als proof of concept galt, wurden feste Assistenzlevel gewählt. Für diesen Ansatz standen noch zwei Alternativen zur Auswahl. Die erste Alternative war der Gedanke über ein Textfeld ein Assistenzlevel einzutragen, welches durch das Drücken eines Knopfes an die Netzwerkschnittstelle gesendet wird. Die zweite Alternative war es eine eigene Berechnung des Assistenzlevels durchzuführen.

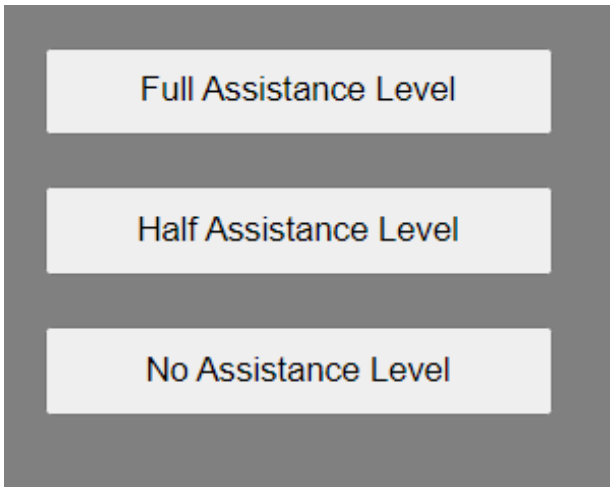

Abbildung 5.3: Benutzeroberfläche der Webpage

In Abbildung 5.4 ist der Programmcode für das Verschicken des Assistenzlevels an die Netzwerkschnittstelle zu sehen. Die Webpage wurde mit Hilfe von Python Flask schnell und einfach implementiert, in dem das *request*-Objekt genutzt werden konnte. Das Assistenzlevel und der Versender der Nachricht wird in ein JSON verpackt und dann an die Netzwerkschnittstelle gesendet.

```python
@app.route('/nohelp', methods=['GET'])
def nohelp():
    if request.method == 'GET':
        nhData = {'sender': 'flaskServer', 'data': '0.0'}
        nhRes = requests.post("http://localhost:8080/Interface_war/MyServlet", json=nhData)
        return nhRes
```

Abbildung 5.4: Programmcode: Verschicken des Assistenzlevels an die Netzwerkschnittstelle mit Assistenzlevel von 0,0

5.1.3 Java Netzwerkschnittstelle

Der Sinn hinter dieser Netzwerkschnittstelle war die Simulierung der ELAI. Es wurde eine eigene Netzwerkschnittstelle implementiert, da es mit dieser einfacher war schnelle Änderungen zu tätigen um den proof of concept zu zeigen.

Die Netzwerkschnittstelle wurde als *HttpServlet* in Java realisiert und auf einem Tomcat 9 Server gehostet. Wie in Abbildung 5.5 zu sehen ist, wurden beim Erhalten einer Anfrage zuerst die ankommenden Daten mit *private String[] getJSONString(HttpServletRequest request)* verarbeitet. Basierend auf der Versenderinformation wird dann im Falle des *flaskServer* das neue Assistentlevel gespeichert und im Falle des *UnityGame* das Assistenzlevel an LISA verschickt.

```java
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String[] data = getJSONString(request);
    if(data[0].equals("flaskServer")) {
        assistanceLevel = Double.parseDouble(data[1]);
    } else if(data[0].equals("UnityGame")) {
        response.getOutputStream().println(assistanceLevel);
    }
}
```

Abbildung 5.5: Programmcode: Abhandlung von Anfragen in der Netzwerkschnittstelle

5.2 Der adaptive Assistent LISA

Genau wie in der Arbeit von Marcel Danz [Danz 2018] wurde auch hier die Funktionalität des Klickens auf LISA wieder verwendet und erweitert. Die bisherige Funktionalität des Klickens hat ermöglicht darüber einen Hinweis zu erfragen. Zu dieser Funktionalität wird nun jedes Mal bei einem Klick zuerst eine Anfrage an die ELAI für eine neue Adaptivitätsantwort geschickt. Sobald diese ankommt wird daraus das Assistenzlevel geparst. Um zu garantieren, dass der jetzt angefragte Hinweis auf dem neuen Assistenzlevel basiert, wird nun mit Hilfe eines Events ein Hinweis angefragt. Der adaptive Assistent LISA verfügt über einige neue Funktionen im Vergleich zu ihrem Vorgänger. LISA ist nun in der Lage folgende Aufgaben zu erfüllen:

```
IEnumerator getDataFromELAI()
{
    currentMission = missionController.ActiveMission;
    StageTypes currentStage = currentMission.CurrentStage;

    int fails = getFails(currentStage);

    string context = currentStage.ToString() + "," + fails;
    string responseString = "context=" + context + "&userid=" + userName + "&engine=ElaiSteeringService";
    string url = "http://localhost:8080/ELAI2019/api/v1/adaptivity/response?" + responseString;
    var request = new UnityWebRequest(url, "GET");
    request.downloadHandler = (DownloadHandler)new DownloadHandlerBuffer();

    yield return request.Send();

    if (request.isError)
    {
        Debug.Log(request.error);
    }
    else
    {
        assistanceLevel = float.Parse(parseAssistanceLevel(request.downloadHandler.text));
        Debug.Log(assistanceLevel);
        if (OnRequestHint != null)
        {
            isUpdated = true;
            OnRequestHint();
        }
    }
}
```

Abbildung 5.6: Programmcode: Anfrage an die ELAI

Aufzeichnen der Fehlschläge LISA ist es nun möglich einige Fehlschläge des Spielers aufzuzeichnen. Dies wurde über ein Event System implementiert, welches an verschiedenen Stellen des Spiels, wie z.B. einem Missionfehlschlag oder dem Verlassen einer Szene während einer Aufgabe triggert. Auf dieses Event reagiert der *ELAIConnectionController* und speichert die Daten, welche für die spätere Anfrage an die ELAI wiederverwendet werden. Um mehr Szenarien zu erfassen, können einfach neue Events implementiert werden, welche die zugehörigen Werte der Fehlschläge erhöhen.

Verbindung zur ELAI Die Verbindung zur ELAI für die Anfrage einer neuen Adaptivitätsantwort wurde mit Hilfe eines *UnityWebRequests* und einem *DownloadHandler* implementiert. Der *UnityWebRequest* stellt eine neue Anfrage, wie in Abbildung 5.6 zu sehen ist, und erhält über den *DownloadHandler* die Daten der Anfrage. Hierfür wird der aktuelle Missionszustand und die Fehlschläge der aktuellen Aufgabe an die ELAI übermittelt.

Parsen des Assistenzlevels LISA nimmt die Adaptivitätsantwort der ELAI entgegen und parst diese in *public string parseAssistanceLevel(string json)* (zu sehen in Abbildung 5.7). Dazu wird ein *JsonTextReader* genutzt, welcher es ermöglicht die einzelnen Zeilen des JSONs zu lesen. Es wird solange gelesen bis man zu dem *assistance*-Objekt kommt und

```
public string parseAssistanceLevel(string json)
{
    var assistanceFound = false;
    JsonTextReader jsonReader = new JsonTextReader(new StringReader(json));
    while (jsonReader.Read())
    {
        if (jsonReader.Value != null)
        {
            if(jsonReader.Value.ToString().Equals("assistance")) {
                assistanceFound = true;
            }
            if (jsonReader.Value.ToString().Equals("scaled") && assistanceFound == true)
            {
                jsonReader.Read();
                assistanceFound = false;
                return jsonReader.Value.ToString();
            }
        }
    }
    return "";
}
```

Abbildung 5.7: Programmcode: Parsen der Adaptivitätsantwort

dann wird der nächste *scaled*-Wert genommen. Sollte kein *assistance*-Objekt gefunden werden, gibt die Methode einen leeren String zurück.

Darstellen von variierenden Hinweisen Durch einfügen neuer *Messages* in die *LisaHints.xml*, welche für die einzelnen Hinweise der LISA zuständig ist, können nun variierende Hinweise angezeigt werden. Dazu werden nun für jeden Hinweis vier Nachrichten angelegt, welche beim Erstellen des Hinweises eingelesen werden. Basierend auf dem Assistenzlevel wird beim Anfragen von Hinweisen nun unterschiedliche Hinweise gegeben (s. Abbildung 5.8). Hierbei wird im Falle dass eine *Message* gleich *null* ist der normale Hinweis ausgegeben. Dies wurde implementiert, um weiterhin den normalen Spielablauf zu gewähren, sollten die Hinweise für die aktuelle Mission nicht überarbeitet sein. Des Weitern kann LISA nun basierend auf dem Assistenzlevel auch unterschiedliche Funktionen nutzen. Dazu gehören z.B. das Hervorheben von Objekten bei einem Assistenzlevel von größer als 0,5 und das Anzeigen von blockierenden Hinweisen mit Hilfe einer *MessageBox* bei einem Assistenzlevel von 1,0.

Hervorheben von Text Das Hervorheben von Objekten, wie z.B. einer Konsole oder eines Raums, wurde von der Implementierung von Marcel Danz weitergenutzt [Danz 2018]. Es wurde nun die Möglichkeit hinzugefügt ab einem Assistenzlevel von 0,5 in der Wetterkonsole den Text *Cloudy* rot hervorzuheben, um dem Spieler es einfacher zu machen die bewölkten Tage zu sehen.

```
private string getMessage(float assistanceLevel, Hint hint) {
    switch( assistanceLevel == 1.0 ? "1" :
            assistanceLevel >= 0.5 ? "High":
            assistanceLevel > 0.0 ? "Low":
            "0")
    {
        case "1":
        if(hint.Message2 != null) {
            return hint.Message2;
        }
        goto default;

        case "High":
        return hint.Message;
        case "Low":
        if(hint.Message3 != null) {
            return hint.Message3;
        }
        goto default;
        case "0":
        if(hint.Message4 != null) {
            return hint.Message4;
        }
        goto default;
        default:
        return hint.Message;
    }
}
```

Abbildung 5.8: Programmcode: Auswahl des Hinweises basierend auf dem Assistenzlevel

Leitung des Spielers durch Hinweise beim ersten Spielstart Beim ersten Spielstart wird der Spieler mit Hilfe von zwei *MessageBoxes* zum Missionsstart geführt. Dazu wird zuerst bei der Ankunft auf dem Übersichtsbildschirm der Arche eine *MessageBox* angezeigt, welche den Spieler zur Brücke bringt. Dann auf der Brücke angekommen wird eine weitere *MessageBox* angezeigt, welche dem Spieler mitteilt, dass er durch Klicken auf LISA Hilfe erhalten kann.

5.3 ELAIControlService

Der ELAIControlService wurde implementiert um die ELAI dabei zu unterstützen ein Assistenzlevel zu berechnen. Hierfür besitzt der ELAIControlService folgende Funktionen:

Laden von xAPI Statements aus einem LRS Der ELAIControlService besitzt die Fähigkeit sich alle xAPI Statements aus einem LRS zu laden. Diese Statements werden zur Bestimmung des Assistenzlevels genutzt. Das LRS kann dabei über Systemvariablen angesteuert werden (s. Abbildung 5.9). Dafür gibt es die Systemvariablen LRS_URL, bei der die URL des LRS, LRS_USER, bei der der Benutzername, und LRS_PASSWORD bei der das Passwort abgespeichert werden. Sollten keine Systemvariablen vorhanden sein wird ein Standardwert genutzt. Die Methode *get_statements_from_lrs(url, user, password)* wurde von Alexander Streicher entwickelt und zur Verfügung gestellt für diese Arbeit.

```
url = os.environ.get("LRS_URL", "Insert Standard Here")
user = os.environ.get("LRS_USER", "Insert Standard Here")
password = os.environ.get("LRS_PASSWORD", "Insert Standard Here")
content = get_statements_from_lrs(url,user,password)
```

Abbildung 5.9: Programmcode: Auswahl des LRS über Systemvariablen

Filtern der Statements Die einkommenden Statements von *get_statements_from_lrs(url, user, password)* können mit Hilfe von *check_for_userid(userid, statement)* gefiltert werden. Hierbei wird geprüft ob das angegebene Statement *statement* die erforderliche *userid* beinhaltet. Dazu wird, wie in Abbildung 5.10 zu sehen ist, überprüft ob der *name* des *actors* des aktuellen *statements* der *userid* entspricht. Dabei gibt die Methode eine Fehlermeldung aus, sofern das Statement falsch formatiert ist, also keinen *actor* beinhaltet. Mit dieser Methode kann der ELAIControlService gewährleisten, dass nur Statements die für den aktuellen Nutzer relevant sind verwendet werden.

Miteinbeziehen von Verben und Aktivitäten Der ELAIControlService beachtet bei der Berechnung des Assistenzlevels die Verben *failed* in Kombination mit der Aktivität *Mission* und *asked* in Kombination mit der Aktivität *hint*. Diese Statements treten dann auf, wenn eine Mission fehlgeschlagen ist oder der Spieler einen Hinweis erfragt hatte. Um nicht

```
def check_for_userid(userid, statement):
    try:
        if (statement["actor"]["name"] == userid):
            return True
        else:
            return False
    except:
        # occurs if the parsed statement from the lrs isn't correctly formated
        print("the current statement doesn't feature an actor or name")
```

Abbildung 5.10: Programmcode: Überprüfung der UserID im Statement

zu alte Daten miteinzubeziehen werden hier lediglich die letzten sieben Statements verwendet. Dies ist einfach anpassbar auf die Bedürfnisse des Anwenders. Außerdem ist es leicht, weitere Verben und Aktivitäten zu implementieren. Abhängig von der Anzahl der fehlgeschlagenen Missionen bzw. erfragten Hinweise wird das Assistenzlevel angepasst.

Berechnung des Assistenzlevels mit einem Entscheidungsbaum Für die Berechnung des Assistenzlevels wurde ein Entscheidungsbaum implementiert, welcher basierend auf dem Nutzer, dem aktuellen Missionszustand und der Misserfolge in dem aktuellen Missionszustand ein Assistenzlevel berechnet und dies mit Faktoren, welche durch die Statements aus dem LRS berechnet werden, kombiniert.

5.4 Diskussion

In diesem Kapitel wird darüber diskutiert, in wiefern die entworfenen Konzepte aus Kapitel 4 realisiert wurden.

Adaptiver Assistent LISA LISA ist in der Lage Fehlschläge des Spielers aufzuzeichnen und zu speichern, jedoch wurden für diese Arbeit nur einige gezielte Misserfolge aufgezeichnet. Darunter fallen das Verlassen der Szene während einer Aufgabe, wenn dies nicht ausdrücklich erforderlich ist, das Scheitern der Aufgabe und das falsche Ausfüllen des Reports in ViLand. Dies wurde über ein leicht erweiterbares *Eventsystem* realisiert, welches die Möglichkeit bietet weitere Misserfolge in der Zukunft aufzuzeichnen. Durch die Erweiterbarkeit dieses *Eventsystems* ist es durchaus vertretbar, dass in dieser Arbeit nur einige *Events* für die Demonstrierung implementiert wurden.

LISA besitzt die Funktion die ELAI anzufragen nach einer neuen Adaptivitätsantwort und diese nach dem neuen Assistenzlevel zu parsen. Dies wurde erfolgreich und vollfunktionsfähig implementiert.

Das Darstellen von Hinweisen, welche basierend auf dem momentanen Assistenzlevel variieren, wurde erfolgreich für *Mission001* implementiert. Wie bereits beschrieben, ist dieses System so entwickelt, dass die Hinweise in der *LisaHints.xml* Datei jeder Mission

hinzugefügt werden müssen. Dies erfolgt durch das Hinzufügen von *<Message2> </Message2>*, *<Message3> </Message3>* und *<Message4> </Message4>* Blöcken in der xml-Datei. Das System wurde so entwickelt, dass es weiterhin vollkommen kompatibel ist mit dem alten Format, welches nur einen einzigen *Message*-Block besitzt. Bei der Verwendung des alten Formats fallen jedoch die variierenden Hinweise weg, da diese missionsspezifisch sind. Das Hervorheben der Objekte basierend auf dem Assistenzlevel ist unabhängig davon voll funktionsfähig und funktioniert ohne eine benötigte Änderung von Dateien.

Damit sind die Aufgaben des adaptiven Assistenten LISA, wie in Abschnitt 4.1 beschrieben, erfolgreich implementiert.

ELAIControlService Der ELAIControlService verfügt über die Funktion eine Anfrage der ELAI entgegenzunehmen und mit Hilfe der weitergeleiteten Werte ein Assistenzlevel zu berechnen. Zur Berechnung des Assistenzlevels werden *xAPI Statements* aus einem LRS geladen und diese basierend auf der *UserID* gefiltert. Basierend auf diesen Statements wird das Assistenzlevel angepasst, welches durch den Entscheidungsbaum berechnet wird. Das Berechnete Assistenzlevel kann an die ELAI zurückgesendet werden.

Somit wurden alle Aufgaben die in Abschnitt 4.2 erfolgreich implementiert. Der Entscheidungsbaum jedoch beruht nicht auf *State of the Art*-Methodiken und könnte deshalb verbessert werden. Die Implementierung des Entscheidungsbaums ist leicht gehalten. Diese Komponente sollte in zukünftigen Arbeiten erweitert und mit *State of the Art*-Methodiken implementiert werden.

6 Verifikation

In diesem Kapitel wird gezeigt, dass die gewünschte Adaptivität des Lernspiels Lost Earth 2308 mit Hilfe des adaptiven Assistenten LISA erzielt wurde. In Abschnitt 4.3 wurde vorab ein Beispielszenario entworfen. Anhand von diesem Beispielszenario wird nun mit Hilfe des Anwendungsszenarios ermittelt, wie die einzelnen Adaptivitätsmechanismen umgesetzt wurden.

6.1 Anwendungsszenario

Abbildung 6.1: Startbildschirm von Lost Earth 2308 mit eingetragenem Nutzer

In diesem Anwendungsszenario spielt Andreas einen Probedurchlauf von Lost Earth 2308. Andreas startet das Spiel und hinterlegt im Startbildschirm seinen Namen und beginnt das Spiel, wie in Abbildung 6.1 zu sehen ist.

Zuerst kommt er auf die Übersichtskarte der Arche (Abbildung 6.2). Hier bekommt er eine Nachricht von LISA, dass er sich auf die Brücke begeben sollte, um die erste Mission zu starten.

Auf der Brücke angekommen bekommt er eine weitere Nachricht, welche ihm sagt, dass er durch einen Klick auf LISA um Hilfe bitten kann (Abbildung 6.3). Da Andreas nun nicht weiss was er tun muss, klickt er auf LISA und bekommt dann von ihr den Hinweis, dass

Abbildung 6.2: Übersichtsbildschirm der Arche mit Hinweis von LISA

Abbildung 6.3: Hinweis von LISA auf der Brücke

Abbildung 6.4: Hinweis von LISA auf der Brücke für den Missionsstart

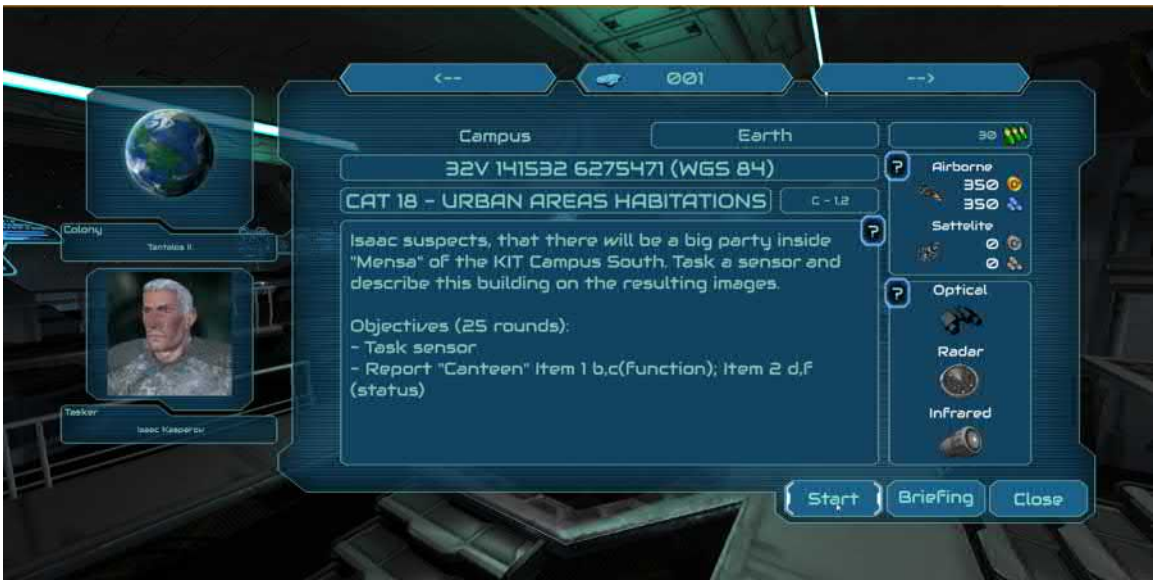

Abbildung 6.5: Missionsübersicht

Abbildung 6.6: Hinweis von LISA im Hangar für den Sensorstart

er seine Mission in der *MissionList* starten kann (Abbildung 6.4). Nachdem Andreas die Missionsbeschreibung gelesen hat (Abbildung 6.5), begibt er sich in den Hangar.

Dort angekommen schaut er sich an, was für Konsolen es gibt und er erinnert sich, dass er einen Sensor starten muss. LISA gibt ihm im selben Zuge auch den Hinweis, dass er zum Starten eines Sensors an die *Select und Task Sensor* Konsole muss (Abbildung 6.6).

Abbildung 6.7: Hinweis von LISA zum Überspringen von Zügen

Abbildung 6.8: Nachricht von LISA bzgl. des Scheiterns des Sensors

Daraufhin öffnet er die *Select und Task Sensor* Konsole und startet einen Sensor. Ohne zu wissen was zu tun ist, verlässt er den Hangar und bekommt von LISA den Hinweis, dass er die Zeit verstreichen lassen muss bis der Sensor wieder zurück kommt (Abbildung 6.7).

Abbildung 6.9: Fehlschlagen der Mission

Andreas wartet daraufhin bis LISA ihm die Nachricht gibt, dass er den Sensor falsch gestartet hat, denn der Sensor konnte durch die Wolken keine brauchbaren Bilder aufzeichnen (Abbildung 6.8). Dadurch schlägt die Mission fehl und die Mission wird neugestartet (Abbildung 6.9).

Abbildung 6.10: Hinweis von LISA auf der Brücke für den Missionsstart mit Hervorheben der Konsole

Abbildung 6.11: Hinweis von LISA auf der Brücke mit Ortsangabe

Abbildung 6.12: Hinweis von LISA im Hangar zum korrekten Sensorstart mit Beachtung der Wetterlage

Im zweiten Durchlauf bekommt Andreas viel mehr Hilfe von LISA. Auf der Brücke angekommen sieht der Spieler direkt dass die *MissionList* hervorgehoben ist und somit startet Andreas die Mission erneut (Abbildung 6.10).

Andreas bekommt direkt von LISA mitgeteilt, dass er sich zum Hangar begeben soll und Räume mit Hilfe eines Rechtsklicks verlassen kann (Abbildung 6.11).

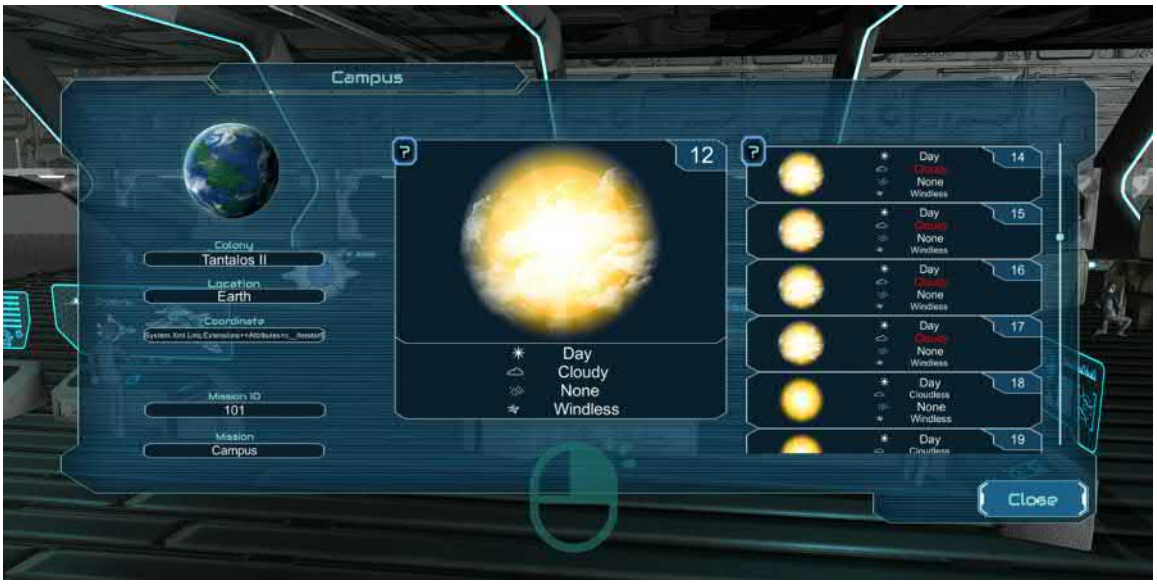

Abbildung 6.13: Hervorheben der bewölkten Tage in der Wetterkonsole

Abbildung 6.14: verzögerter Sensorstart um vier Züge

Im Hangar angekommen, bekommt Andreas dieses Mal den Hinweis von LISA, dass er bei der Wetterkonsole darauf achten muss wann es bewölkt ist und deshalb den Sensor verzögern muss (Abbildung 6.12). In der Wetterkonsole wird ihm rot hervorgehoben, an welchen Tagen es bewölkt ist (Abbildung 6.13). Mit dieser neuen Information geht Andreas wieder zur *Select und Task Sensor* Konsole und verzögert dieses Mal den Sensor (Abbildung 6.14).

Abbildung 6.15: Hilfe von LISA zum Überspringen von Zügen mit Hinweis wo der Spieler den Zug der Sensorrückkehr sehen kann

Abbildung 6.16: Nachricht von LISA bzgl. der erfolgreichen Bildaufnahme

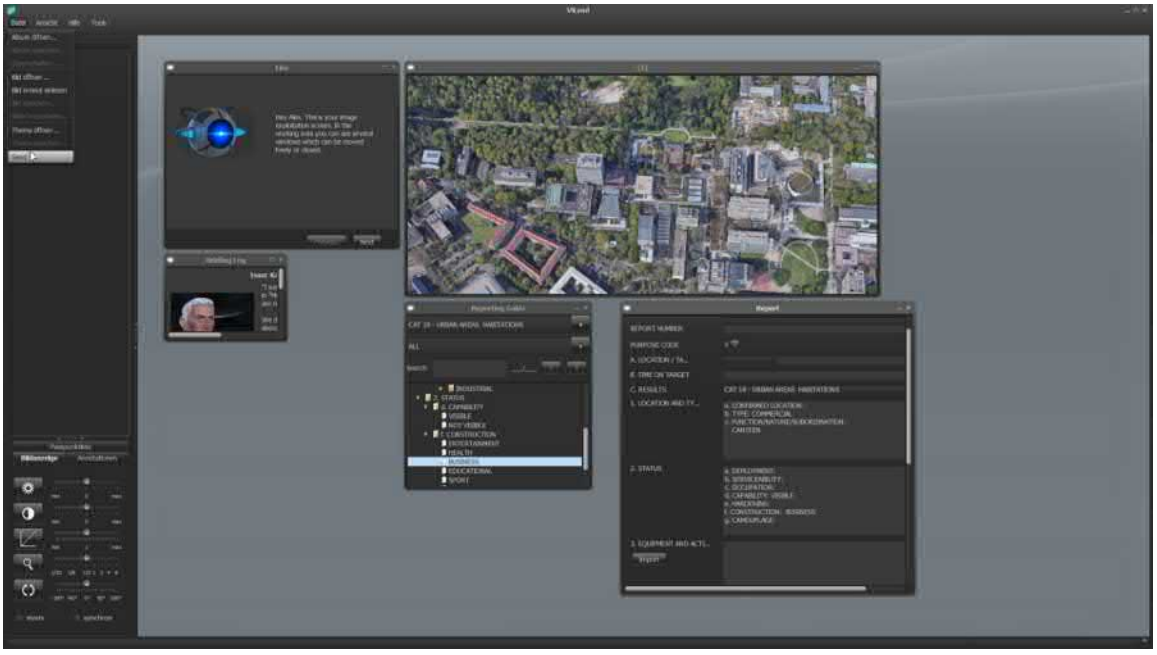

Abbildung 6.17: ViLand mit ausgefülltem Report

LISA gibt Andreas mit Hilfe eines Popups den Hinweis, dass er über das *End Turn Widget* die Zeit verstreichen lassen soll und in der *Select und Task Sensor* Konsole einsehen kann wielang er warten muss (Abbildung 6.15. Er wartet wieder ab bis der Sensor zurückkommt und bekommt von LISA mitgeteilt, dass der Sensor erfolgreich Bilder erfassen konnte und

Andreas sich auf die Brücke zum *Exploitation Table* begeben soll, um die Bilder auszuwerten (Abbildung 6.16).

Abbildung 6.18: Erfolgreicher Abschluss der Mission

Dort angekommen klickt er auf den *Exploitation Table* und begibt sich in *ViLand*. In *ViLand* kann Andreas dann die Bilder auswerten und trägt die erforderlichen Informationen in seinen Report ein, welcher er dann abschickt (Abbildung 6.17). In Lost Earth 2308 wieder angekommen, bekommt er mitgeteilt, dass er nun die Aufgabe erfolgreich abgeschlossen hat (Abbildung 6.18).

6.2 Erklärung

In diesem Szenario wird die Adaptivitätskomponente dieser Arbeit gezeigt. Hierzu wird während eines Spieldurchlaufs des Nutzers Andreas gezeigt wie das Spiel reagiert, nachdem der Spieler die Mission nicht geschafft hat.

Zu Beginn des Spiels wird der Benutzername über das Eingabefeld im Startbildschirm ermittelt. Dieser wird für die weiterführende Adaption des Lernspiels wichtig sein, um die bereits gesammelten xAPI Statements dem jetzigen Nutzer zuordnen zu können. Während des Spieldurchlaufs werden darüberhinaus Daten gesammelt, welche bei der Anfrage an die ELAI mitversendet werden. Hierzu gehören Fehler die der Spieler während des Spieldurchlaufs macht, wie z.B. das Vergessen des Wetters. Bei jeder Anfrage an die ELAI und der einhergehenden Weiterleitung an den ELAIControlService werden vergangene xAPI-Statements aus dem LRS und die übergebenen Daten genutzt, um mit Hilfe eines Entscheidungsbaums festzustellen, was für ein Assistenzlevel der Spieler aktuell benötigt. In diesem Szenario wurde nach dem Fehlschlagen der ersten Mission das Assistenzlevel neuberechnet und damit einige neue Adaptivitätsmechanismen aktiviert. In diesem Szenario wurde nach dem Fehlschlagen der Mission das Assistenzlevel auf 1,0 gesetzt, was dazu führt, dass LISA nun deutlich detailiertere Hinweise dem Spieler präsentiert und anfängt wichtige Objekte und den Text in der Wetterkonsole hervorzuheben. In Abschnitt 5.2 und Abschnitt 5.3 kann die genaue Funktionsweise nachgelesen werden.

7 Fazit und Ausblick

In dieser Arbeit wurde gezeigt wie ein Lernspiel mit Hilfe eines Adaptivität-Frameworks adaptiert werden kann. Dazu wurde das Lernspiel *Lost Earth 2308* an die *ELAI* und die *ELAI* an den neu entwickelten *ELAIControlService* angebunden. Mit Hilfe dieser Anbindung kann der adaptive Assistent *LISA* nun in *Lost Earth 2308* einzelne Spielelemente anpassen und unterschiedliche Hinweise dem Spieler präsentieren, welche dafür sorgen, dass der Spieler die Aufgaben erfolgreich abschließen kann und somit den Lernerfolg erhöht.

Über diese Arbeit hinaus gibt es Möglichkeiten den adaptiven Assistenten LISA zu verbessern, in dem man als Beispiel die Adaptierung feingranularer gestaltet. Darunter könnte fallen, dass LISA weitere Funktionen erhält, welche sie im Spiel adaptieren kann, z.B. das Hervorheben des *Start Delay*-Textes und -Eingabefeldes in der *Select und Task Sensor*-Konsole oder man könnte das gespeicherte Assistenzlevel aufteilen auf die unterschiedlichen Missionszustände, dies würde zur Folge haben, dass Spieler nur Hilfe erhalten in den Teilaufgaben der Mission, in der sie auch tatsächlich Probleme aufweisen. Die momentane Implementierung erlaubt dies nur, durch dauerhafte manuelle Anfrage eines neuen Assistenzlevels.

Darüber hinaus könnte man in ELAI eine künstliche Intelligenz implementieren, welche basierend auf den Statements im LRS das Assistenzlevel, den Performance Score und das Skilllevel berechnet. Der ELAIControlService könnte außerdem in einer weiteren Arbeit, um einen *State of the Art*-Entscheidungsbaum erweitert werden.

Des Weiteren könnte eine zukünftige Arbeit mit Hilfe einer Studie beweisen, inwiefern diese oder eine vergleichbare Implementierung den Lernerfolg eines Spielers tatsächlich steigert. Marcel Danz hat in seiner Ausarbeitung (Danz 2018) gezeigt, dass der Lerneffekt sich durch seine Implementierung eines pädagogischen Agenten nur leicht verbessert hat. Interessant wäre nun zu wissen, wie es bei dem vorgestellten adaptiven Assistenten dieser Arbeit ausfallen würde.

Abbildungsverzeichnis

Literatur

Alice Mitchell, Carol Savill-Smithh (2004). „The use of computer and video games for learning“. In: *A review of the literature*.

Alshammari, M., R. Anane und R. J. Hendley (2014). „Adaptivity in E-Learning Systems“. In: *2014 Eighth International Conference on Complex, Intelligent and Software Intensive Systems*, S. 79–86.

Aroyo, Lora und Darina Dicheva (2004). „The New Challenges for E-learning: The Educational Semantic Web“. In: *Journal of Educational Technology and Society* 7.4, S. 59–69. ISSN: 11763647, 14364522.

Atorf, Daniel (2020). *Lost Earth 2307*. besucht am 12.11.2020. URL: https://www.iosb.fraunhofer.de/servlet/is/55534/.

Bakhouyi, A. u. a. (2017). „Evolution of standardization and interoperability on E-learning systems: An overview“. In: *2017 16th International Conference on Information Technology Based Higher Education and Training (ITHET)*, S. 1–8.

Benedek, András (2013). „Learning design versus learning experience design: is the experience API making the difference“. In: *Edulearn13 Proceedings*, S. 2609–2621.

Biegemeier, Christian (2016). *Web-basierte Schnittstelle zur Analyse und Adaption von Serious Games*.

Bopp, Kolja (2009). *Serious Games Ein Literaturbericht*.

Bopp, Matthias (2005). „Immersive Didaktik: verdeckte Lernhilfen und Framingprozesse in Computerspielen“. In: *kommunikation @ gesellschaft* 6, S. 17.

Bostrom, Robert P., Lorne Olfman und Maung K. Sein (1990). „The Importance of Learning Style in End-User Training“. In: *MIS Quarterly* 14.1, S. 101–119. ISSN: 02767783. URL: http://www.jstor.org/stable/249313.

Collier, Geoff, Robby Robson u. a. (2002). *E-Learning Interoperability Standards*. White Paper. Citeseer.

Corti, Kevin (2006). „Games-based Learning; a serious business application“. In: *Informe de PixelLearning* 34.6, S. 1–20.

Daniel Livingstone, Jeremy Kemp (2006). „Massively Multi-Learner: Recent Advances in 3D Social Environments“. In: *Computing and Information Systems Journal*.

Danz, Marcel (2018). *Enhancing User Experience in a Serious Game by Revising the UI And Implementing a Pedagogical Agent for Improving Learnability*.

David Hauger, Mirjam Köck (2007). „State of the Art of Adaptivity in E-Learning Platforms“. In: *LWA*, S. 355–360.

Doujak, G. (2015). *Serious Games und Digital Game Based Learning. Spielebasierte ELearning Trends der Zukunft*. GRIN Verlag.

Electronic Arts (2020). *A software delevopment toolset for game creators - Frostbite*. besucht am 12.11.2020. URL: https://www.ea.com/frostbite/engine.

Enochsson, Lars u. a. (2004). „Visuospatial skills and computer game experience influence the performance of virtual endoscopy“. In: *Journal of Gastrointestinal Surgery* 8.7, S. 874–880. ISSN: 1873-4626. URL: https://doi.org/10.1016/j.gassur.2004.06.015.

Epic Games (2020). *The most powerful real-time 3D creation platform*. besucht am 12.11.2020. URL: https://www.unrealengine.com/en-US/.

F. Kareal, J. Klema (2006). „Adaptivity in e-learning“. In: *Current Developments in Technology-Assisted Education* 1, S. 260–264.

Felder, R. und L. Silverman (1988). „Learning and Teaching Styles in Engineering Education.“ In: *Engineering education* 78.7, S. 674–681.

Friesen, Norm (2005). „Interoperability and Learning Objects: An Overview of E-Learning Standardization“. In: *Interdisciplinary Journal of E-Learning and Learning Objects* 1.1, S. 23–31. ISSN: 1552-2237.

Game Designing (2020). *The Top 10 Video Game Engines*. besucht am 12.11.2020. URL: https://www.gamedesigning.org/career/video-game-engines/.

IEEE LTSC (2004). *IEEE Learning Technology Standards Committee*. besucht am 12.11.2020. URL: https://www.ieeeltsc.org/.

IMS Global Learning Consortium (2020). *IMS Global Learning Consortium*. besucht am 12.11.2020. URL: http://www.imsglobal.org/.

ISO/ICE (2004). *ISO/ICE JTC 1/SC36*. besucht am 12.11.2020. URL: https://www.iso.org/committee/45392.html.

JSON.org (2020). *Introducting JSON*. besucht am 12.11.2020. URL: https://www.json.org/json-en.html.

Keefe, J. W. (1979). „Learning style: An overview“. In: *Student learning styles: Diagnosing and prescribing programs* 1.1, S. 1–17.

Kevan, Jonathan M. und Paul R. Ryan (2016). „Experience API: Flexible, Decentralized andActivity-Centric Data Collection“. In: *Technology, knowledge and learning* 21.1, S. 143–149.

Kickmeier, Michael u. a. (2006). „The Elektra Project: Towards a New Learning Experience.“ In: *Interdisciplinary Aspects on Digital Media and Education, Proceedings of the 2nd Symposium of the WG HCI and UE of the Austrian Computer Society*. DOI: 10.13140/2.1.2272.8646.

Mohammad, Al-Omari, Carter Jenny und Chiclana Francisco (Jan. 2016). „A hybrid approach for supporting adaptivity in e-learning environments“. In: *The International Journal of Information and Learning Technology* 33.5, S. 333–348. ISSN: 2056-4880. URL: https://doi.org/10.1108/IJILT-04-2016-0014.

OXM Staff (2018). *The most crucial part of video-game development explained - and how it powered Fortnite's runaway success*. besucht am 12.11.2020. URL: https://www.gamesradar.com/what-is-a-game-engine-and-what-does-it-do/.

Peirce, N., O. Conlan und V. Wade (2008). „Adaptive Educational Games: Providing Non-invasive Personalised Learning Experiences“. In: *2008 Second IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*, S. 28–35. DOI: 10.1109/DIGITEL.2008.30.

Penning, Leo de, Eddy Boot und Bart Kappé (2008). „Integrating Training Simulations and e-Learning Systems: The SimSCORM Platform“. In: *Interservice/Industry Training, Simulation and Education Conference (I/ITSEC)*.

Prensky, Marc (2003). „Digital Game-Based Learning“. In: *Computers in Entertainment (CIE)* 1.1, S. 21–21.

Prensky, Marc (2005). „Engage Me or Enrage Me“. In: *Educase Review* 40.5, S. 61–64.

Silva, J. M. und A. E. Saddik (2011). „An adaptive game-based exercising framework“. In: *2011 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems Proceedings*, S. 1–6. DOI: 10.1109/VECIMS.2011.6053847.

Stracke, Christian M (2006). „Interoperability and Quality Development in e-Learning:Overview and Reference Model for e-Learning Standards“. In: *Proceedings of the Asia-Europee-Learning Colloquy*.

Streicher, Alexander und Jan D. Smeddinck (2016). „Personalized and Adaptive Serious Games“. In: *Entertainment Computing and Serious Games: International GI-Dagstuhl Seminar 15283, Dagstuhl Castle, Germany, July 5-10, 2015, Revised Selected Papers*. Hrsg. von Ralf Dörner u. a. Cham: Springer International Publishing, S. 332–377. ISBN: 978-3-319-46152-6. DOI: 10.1007/978-3-319-46152-6_14. URL: https://doi.org/10.1007/978-3-319-46152-6_14.

Streicher Alexander, Bach L. und Roller W. (2019). „Usage Simulation and Testing with xAPI for Adaptive E-Learning“. In: *European Conference on Technology Enhanced Learning*. Springer, S. 692–695.

Streicher Alexander, Wolfgang Roller (2015). „Towards an interoperable adaptive tutoring agent for simulations and serious games“. In: *International Conference on Theory and Practice in Modern Computing, MCCSIS*, S. 194–197.

Tarja Susi, Mikael Johannesson (2007). *Serious Games – An Overview*. Institutionen för kommunikation och information.

Unity (2020). *Unity - Echtzeit-Entwicklungsplattform*. besucht am 12.11.2020. URL: https://unity.com/de.

W. Lewis Johnson, James C. Lester (2015). „Face-to-Face Interaction with Pedagogical Agents, Twenty Years Later“. In: *International Artificial Intelligence in Education Society 2015*.

W3C Working Group (2004). *Web Services Architecture*. besucht am 12.11.2020. URL: https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest.

W3C Working Group (2020). *Web Services Architecture*. besucht am 12.11.2020. URL: https://www.w3.org/RDF/Metalog/docs/sw-easy.

Ward, Jeff (2008). *What is a Game Engine?* besucht am 12.11.2020. URL: https://www.gamecareerguide.com/features/529/what_is_a_game_.php.

xAPI.com (2020). *What is xAPI aka the Experience API or Tin Can API*. besucht am 12.11.2020. URL: https://xapi.com/overview/.