Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelor-Thesis Nr. 301

Nutzerzentriertes Design von Gamification in der Produktion mit einem Anwenderunternehmen in der Automobilindustrie

Peter Muschick

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. Albrecht Schmidt
Betreuer/in:	Dr. Oliver Korn
Beginn am:	27. November 2015
Beendet am:	31. Mai 2016
CR-Nummer:	H.5.2

Kurzfassung

In der vorliegenden Bachelorarbeit wurden verschiedene Gamificationimplementierungen in einem vorhandenen System mithilfe einer Studie bewertet und weiterentwickelt. Die Studie wurde in Zusammenarbeit mit der AUDI AG in Ingolstadt und der KORION GmbH in Ludwigsburg durchgeführt. Durch die Auswertung der Ergebnisse wurde festgestellt, dass das vorhandene System den Wünschen der potenziellen Nutzer nicht gerecht wird. Es wurde ein Konzept erstellt und umgesetzt, dass die Weiterentwicklung der Implementierungen mithilfe der Unity-Engine vorsieht.

Abstract

In this work various gamification implementations in a given systen have been evaluted with a study and further developed. The study was accomplished in cooperation with the AUDI AG in Ingolstadt and the Korion GmbH in Ludwigsburg. The evaluation of the study showed that the given system is unsatisfactory in some points. A new concept was deleoped and implemented with the Unity-Engine to match the results of the evaluation.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1. Einleitung

Gamification. Ein großes und spannendes Thema, das aktuell immer mehr an Bedeutung gewinnt. Üblicherweise wird der englische Begriff *Gamification* nicht ins Deutsche übersetzt, eine mögliche Übersetzung wäre etwa *Spielifizieren*. Der Begriff *Spielifizieren* lässt in zwei Wörter aufteilen: *Spiel* und das Wortbildungselement *-fizieren*, dessen Bedeutung von *machen* kommt [Bib16]. Aufgetrennt und wieder zusammengesetzt erhält man somit *Spiel-machen*, vereinfacht es wird ein Spiel gemacht (wo vorher keines war).

Die meisten Leser dürften bereits mit Gamificationansätzen in Berührung gekommen sein, womöglich ohne diese als solche zu identifizieren. Die Bedeutung von Gamification ist laut Sebastian Deterding et al.: *spielerische Elemente in eine spielfremde Umgebung zu transportieren, um zusätzliche Motivation und Leistung zu generieren* (U2) [DKND11].

Der gewünschte Effekt von Gamification hat zur Auswirkung, dass Tätigkeiten oder Anwendungen deren Ausführung wenig Freude bereiten, durch die Einbringung von spielerischen Elementen, wieder ansprechender werden. Dabei wird an der Tätigkeit oder Anwendung selbst nichts geändert, es werden lediglich spielerische Elemente hinzugefügt.

Bevor die Vorzüge und Umsetzungen der Gamification in dieser Arbeit erläutert werden, ist es nötig die Vorgehensweise der Bachelorarbeit aufzuzeigen.

1.1. Vorgehensweise

Die Vorgehensweise in der vorliegender Arbeit orientiert sich an der *ISO 9241-210:2010*-Norm (U1). Die Norm *9241* beinhaltet Richtlinien zur Entwicklung von Anwendungen auf dem Gebiet der Mensch-Computer-Interaktion. Der hier verwendete Teil *210* der ISO Norm, beinhaltet den Standard für den *Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme*. Die Norm enthält Standards für die Entwicklung von interaktiven Systemen. Die Norm zielt darauf ab, dass sich die Entwicklung an den Erfordernissen und Anforderungen der Benutzer orientiert, um gebrauchstaugliche und zweckdienliche Systeme zu erzeugen [ISO10].

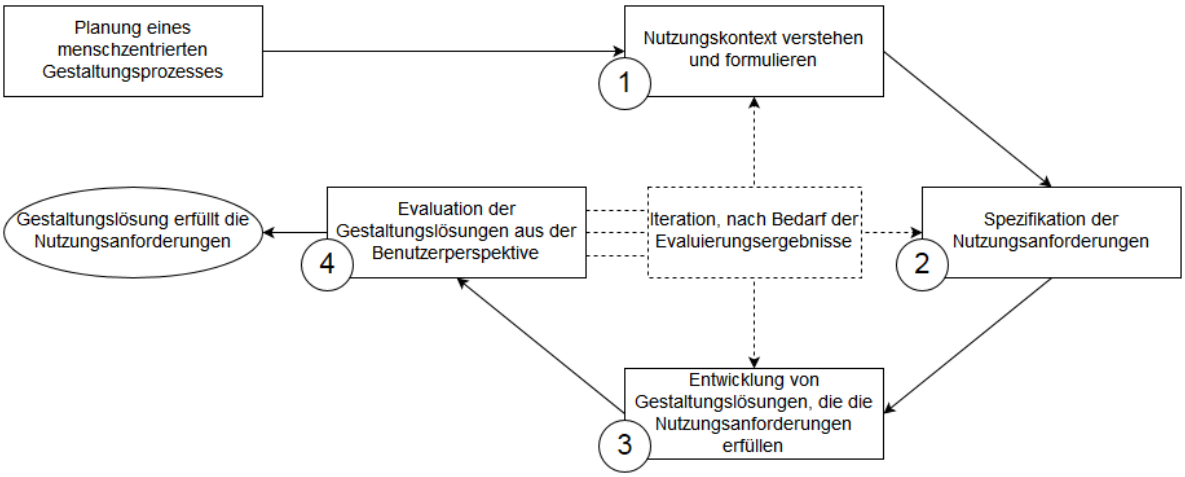


Abbildung 1.1.: Der Entwicklungszyklus eines menschzentrierten Gestaltungsprozesses anhand ISO 9241-210:2010

Zum Zweck der nutzerzentrierten Entwicklung wurde ein standardisiertes Verfahren erarbeitet (Abb. 1.1). Nachdem zu Anfang einer Systementwicklung ein menschzentrierter Gestaltungsprozess geplant wurde, beginnt der erste Schritt des Zyklus, der beinhaltet, dass der Nutzungskontext verstanden und formuliert wurde. Daraufhin werden im zweiten Schritt die Nutzungsanforderungen spezifiziert, um im dritten Schritt Gestaltungslösungen zu entwickeln, welche die Nutzungsanforderungen erfüllen. Im vierten Schritt werden die Gestaltungslösungen aus der Benutzerperspektive evaluiert. Nach dem vierten Schritt erhält man zwei Möglichkeiten, entweder die Gestaltungslösung erfüllt die Nutzungsanforderungen oder man steigt, abhängig von den Ergebnissen der Evaluierung, wiederum in einen vorigen Schritt des Iterationszyklus ein.

Diese Bachelorarbeit beginnt mit dem vierten Schritt des Entwicklungszyklus zu menschzentrierten Gestaltungsprozessen an. Nachdem in vorigen Arbeiten die Entwicklung von Gestaltungslösungen vorangetrieben worden sind, werden die Lösungen in dieser Arbeit anhand einer Benutzerstudie evaluiert. Anhand der Ergebnisse der Evaluierung wird die nächste Vorgehensweise entschieden.

1.2. Motivation

In dieser Arbeit wird an der Umsetzung von Gamification in Form der beschriebenen nutzerzentrierten Vorgehensweise geforscht. Es werden Möglichkeiten und Herausforderungen auf dem Gebiet der Gamification erarbeitet. Die Durchführung von Zyklen in der nutzerzentrierten Vorgehensweise hat den Effekt Implementierungen stetig zu verbessern. Da sich Nutzeranforderungen ändern, müssen die Implementierungen immer wieder neu angepasst werden, womit die Notwendigkeit der Durchführung von neuen Entwicklungszyklen entsteht. Mit dieser Arbeit befindet sich ein neuer Entwicklungszyklus in Arbeit.

Die Arbeit ist ein Teil des renommierter Forschungsprojekts motionEAP und beschäftigt sich somit mit zukunftsweisenden Technologien in der Produktion. Für MotionEAP wird mit dem Automobilhersteller AUDI und den Gemeinnützigen Werkstätten und Wohnstätten (GWW) zusammengearbeitet, die als Endanwender zu verstehen sind [PVM]. Für die Forschung und Entwicklung arbeitet die Universität Stuttgart mit der Hochschule Esslingen und der Korion GmbH zusammen. Das Projekt wird durch das Bundesministerium für Wirtschaft und Technologie gefördert und wurde für den *Exzellent:Kooperation*-Preis der Werkstätten, aufgrund besonders guter Zusammenarbeit mit Behindertenwerkstätten, nominiert [Oli14]. Das Vorgängerprojekt von motionEAP wurde mit dem *Gips-Schüle-Preis*-Preis für soziale Innovation ausgezeichnet [mot16a].

1.3. Gliederung

Die vorliegende Arbeit ist in sieben Kapitel gegliedert. Die ersten drei Kapitel enthalten eine Einführung ins Thema Gamification und geben einen Überblick zum Stand der Technik. Daran schließen sich die drei Kapitel Akzeptanz-Studie, Konzept und Implementierung an. Die Ergebnisse der im Kapitel Akzeptanz-Studie präsentierten Studie fließen in das Konzept ein, dessen Beschreibung im Kapitel Erweitertes Gamification-Konzept behandelt wird. Die Umsetzung des Konzepts wird anschließend im Kapitel Implementierung vorgestellt. Das letzte Kapitel enthält die Zusammenfassung der vorliegenden Arbeit und einen Ausblick zum Thema Gamification. Am Ende der Arbeit befinden sich die beiden Kapitel Anhang und *Quellen*.

1.4. Untersuchungsfragen

Die folgenden Untersuchungsfragen werden in dieser Arbeit beantwortet:

U1 An welcher Vorgehensweise orientiert sich die vorliegende Arbeit?

U2 Was ist Gamification und

U3 Wie wird Gamification in der Produktion eingesetzt?

U4 Wie ist die Einstellung der AUDI-Mitarbeiter gegenüber der aktuellen Gamification-Implementierung?

U5 Welche Ergebnisse kann man aus der Studie zu ziehen?

U6 Wie können die Ergebnisse umgesetzt werden?

U7 Wie sehen die umgesetzten Ergebnisse aus?

Die Untersuchungsfragen werden im Laufe der Bachelorarbeit beantwortet. Die Beantwortung einer Frage wird im Text mit beispielsweise (U1) signalisiert.

2. Hintergrund

Das folgende Kapitel behandelt die Grundlagen von Gamification und gibt Einblicke und Hintergrundwissen zu den Vorprojekten, auf welchen diese Arbeit aufbaut.

2.1. Spielerische Ansätze und Motivation

Die Idee von Gamification spielerische Elemente in eine Tätigkeit einzuführen um ein Plus an Leistung und Motivation zu erhalten, dürfte schon so alt sein, dass die ersten Anfänge kaum nachvollziehbar sind (U2). Bekanntlich werden schon junge Kinder von ihren Eltern mit verschiedenen spielerischen Anreizen etwa zum Essen motiviert. Ein Beispiel für die weite Verbreitung von Gamification ist das Pfadfinderabzeichen, welches schon im Jahr 1911 etabliert worden ist. Um ein derartiges Abzeichen zu erhalten muss eine Leistung, innerhalb eines reglementierten Wettbewerbs, erbracht werden. Nachdem eine Leistung erbracht worden ist, wird ein Mitglied mit dem entsprechenden Abzeichen honoriert und gegebenenfalls in eine Bestenliste ausgenommen [Ger09]. In eine neutrale und spielfremde Tätigkeit wird spielerische Elemente eingesetzt und somit *gamifiziert*.

2.1.1. Geschichte der Gamification

Die moderne Geschichte von Gamification beginnt im Jahre 1984, als *Charles Coonradt* das Buch *Game of Work* schrieb und darin verschiedene Vorgehensweisen auf diesem Gebiet der Mitarbeitermotivation zusammengefasst hat [Coo84]. Er erkannte die Missstände von vielen Mitarbeitern in Bezug auf die Bereitschaft und Motivation auf dem Arbeitsplatz und hat erste Motivationshilfen in Form von spielerischen Elementen erwähnt und zusammengefasst. Dadurch gilt er als Wegbereiter für die Gamification.

Ein Viertel Jahrhundert später hat Nick Pelling, in den Anfängen der 00er Jahre, den Begriff Gamification definiert. Ein enormer Zuwachs an Popularität von Gamification ist ab dem Jahr 2010 erfolgt. Etwa ab diesem Zeitpunkt hat die Technologie immer mehr Verbreitung in digitalen Anwendungen gefunden. Der Begriff *Gamification* wird ab dieser Zeit hauptsächlich in einem digitalen Kontext genutzt.

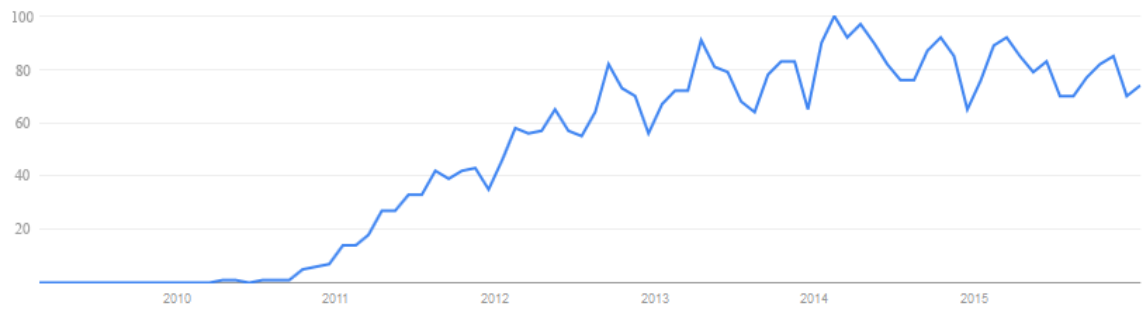


Abbildung 2.1.: Google Trends zu Gamification [Goo16]

Die Behauptung, dass sich der Begriff *Gamification* ab etwa 2010 durchgesetzt hat, lässt sich mithilfe von Google Trends dokumentieren. Google Trends ist ein Tool zur Visualisierung von Suchanfragen bei Google. Anhand der Ergebnisse lassen sich Trends und Interessen der Internetbevölkerung ab dem Jahre 2004 dokumentieren. Die Google-Suchmaschine ist für über 90% der weltweiten Suchanfragen zuständig, was eine gewisse Relevanz mit sich bringt.

Die X-Achse der Abbildung 2.1 gibt die Jahreszahl von Ende 2009 bis 2016 an. Die Y-Achse der genannten Abbildung gibt die relative Popularität der Suchanfrage an, d.h. der höchste Punkt des Graphen gibt den Zeitpunkt an, an welchem die meisten Suchanfragen des Begriffs getätigt wurden. In diesem Fall wurden Anfang 2014 die meisten Suchanfragen, die den Begriff *Gamification* beinhalteten, durchgeführt. In der Abbildung werden dabei nicht die absoluten Suchanfragen dargestellt, sondern die relative Anzahl der Suchanfragen zum Höchstwert. Daraus folgt, dass sobald ein neuer Höchstwert erreicht worden ist, die vorherigen Werte angepasst werden. Die angezeigten Werte sind auch nicht in Relation zur Häufigkeit von anderen Begriffen zu verstehen, sondern das Diagramm steht für sich selbst.

Man kann auf der Abbildung einen ersten Anstieg am Ende des Jahres 2010 erkennen, was die Behauptung stützt. Anschließend setzt sich eine etwa drei Jahre andauernde Steigerung der Popularität des Begriffs fort, die in einer Stagnation endet. Nach einer kurzen geschichtlichen Übersicht des Begriffes, wird im nächsten Abschnitt erläutert, was mit dem Begriff ausgesagt wird.

2.1.2. Motivation

Gamification soll motivieren Dinge zu tun, die man ohne Gamification vermeiden würde. Weshalb man durch Gamification motiviert wird und welche psychologischen Theorien dahinter stehen, folgt als nächstes.

Es werden zwei psychologischen Theorien, die Erklärungsversuche für Motivation geben, vorgestellt. Die Theorie der intrinsischen und extrinsischen Motivation sagt aus, dass Motivationen verschiedene Gründe haben, die sich in die beiden Gruppen, der intrinsischen und extrinsischen Gründe, aufteilen lassen. Die zweite Theorie, die Flow-Theorie von *Mihály Csíkszentmihályi (2006)* gibt einen tieferen Einblick in die Entstehung der Gründe für die Gruppe der intrinsischen Motivationen.

Intrinsische und Extrinsische Motivation

Die Theorie der intrinsischen und extrinsischen Motivation baut zu einem großen Teil auf dem Erklärungsansatz der *drei großen Motive*(1984) von *David McClelland* auf. Der Ansatz von McClelland besagt, dass der Ursprung von Motivation auf ein Macht-, Zugehörigkeits- oder Leistungsmotiv zurückgeht [McC84]. Laut McClelland soll es insgesamt fünf Motivationsquellen geben, die auf die drei genannten Motive zurückgehen. Die Quellen lassen sich in die beiden Gruppen der intrinsischen oder extrinsischen Motivation einteilen:

Die intrinsische (*intrinsecus* (lat.): *inwendig* oder *hineinwärts*) Motivation geht darauf zurück, dass Motivation zur Tätigkeit von der Tätigkeit selbst ausgeht. Die Sache wird um ihrer selbst willen getan ohne bewusst eine Belohnung zu erwarten. Die Belohnung ist Spaß oder auch die eigene Zufriedenheit. Standards und Maßstäbe die ein Mensch unbewusst verinnerlicht hat, können ebenso intrinsische Motivationsfaktoren darstellen [McC84].

Die Gründe für eine extrinsische (*extrinsecus* (lat.): *von außen her (angeregt)* oder *nicht aus eigenem Antrieb erfolgend*) Motivation liegen außerhalb der Tätigkeit. Sachwerte, die Anerkennung des Umfelds oder von außen zugeführte Ideale oder übernommene Ziele, also konkrete zugeführte Belohnungen, können extrinsische Motivationsfaktoren sein. Die Belohnungen können in die drei Kategorien, der Wunsch nach Macht, Zugehörigkeit oder Leistung, eingeordnet werden. Extrinsische Ziele und Ideale, können nach einer gewissen Zeit so weit verinnerlicht werden, dass diese zu intrinsischen Motivationsquellen werden [McC84].

Großangelegte Studien sprechen für die Theorie von *McClelland et al.*. Es konnte nachgewiesen werden, dass je nachdem ob intrinsische oder extrinsische Motivationsquellen aktiv sind, verschiedene Neurotransmitter ausgeschüttet werden. Weiterhin wurde festgestellt, dass verschiedene Menschen unterschiedlich stark auf extrinsische Motivationshilfen reagieren und intrinsische Motive unregelmäßig ausgeprägt sind. Daraus kann man schließen, dass jeder Mensch unterschiedliche Belohnungen benötigt um nachhaltig motiviert zu werden [MPSB87].

Die Flow-Theorie

Die Flow-Theorie von *Mihaly Csikszentmihalyi, 2006* führt die Theorie der intrinsischen Motivation weiter und bietet zusätzliche Erklärungsversuche. Als Flow (*flow* (engl.): *Fließen, Rinnen, Strömen*) bezeichnet *Mihaly Csikszentmihalyi* das Gefühl einer völligen Vertiefung in eine Tätigkeit und ein sich selbst am Laufen erhaltendes Aufgehen in einer Sache. Man könnte das auch als Schaffens- oder Tätigkeitsrausch bezeichnen. Eine wichtige Voraussetzung für das Flow-Erleben ist die Abwesenheit von Unter- bzw. Überforderung. Daraus folgt, dass die Tätigkeit keinen Stress, aber auch keine Langeweile auslösen sollte. Ein Mensch kann im Flow aufgehen, wenn eine Handlung eine gewisse Kreativität oder eigenen Handlungsspielraum zulässt, ohne dass der Handelnde die Kontrolle verliert oder in bedrohliche Situationen abdriftet. Der Flow ist ein Zustand, dessen Erreichen nicht ohne Weiteres erzwungen werden kann. Am ehesten kann ein Flow-Zustand erreicht werden, indem Störfaktoren, wie Ablenkungen oder Überforderung, beseitigt werden [Csi95].

2.1.3. Mechanismen

Nach einem kurzen Ausflug in die psychologischen Erklärungsversuche von Motivationsfaktoren, werden die gängigsten Gamificationmechanismen aufgezählt. Durch Einsetzen von ein oder mehreren der unten angeführten Spielmechanismen wird eine spielfremde Anwendung gamifiziert [KO12]:

Sichtbarer Status
: Dem Spieler wird ein erreichtes Niveau mithilfe von Statussymbolen wie Titeln, Badges bzw. Abzeichen oder Leveln, angezeigt.

Fortschrittsanzeige
: Dabei handelt es sich um eine dynamische Anzeige die den Benutzern den Fortschritt während einer Tätigkeit anzeigt. Dabei werden abgeschlossene und noch zu erledigende Teile visualisiert.

Rangliste
: Durch die direkte Gegenüberstellung der erwähnten Statussymbole wird das Element des Wettbewerbs gestärkt.

Quests/Spezielle Aufgaben
: Bestimmte Aufgaben deren Erledigung eine Verbesserung des eigenen Status als Ergebnis hat. Die Aufgaben können optional gestellt werden und zum eigentlichen Spielziel nichts beitragen.

Transparenz des Resultats
: Dem Spieler sind die Resultate seiner Aktionen bekannt. Beispielsweise können Erfahrungspunkte dafür eingesetzt werden dem Spieler zu signalisieren, dass seine Aktionen zum Erreichen eines Statussymbols beitragen.

Direkte Rückmeldung
: Die Rückmeldung nach einer abgeschlossenen Aktion folgt unmittelbar danach.

Tieferer Sinn
: Es können Ziele definiert werden, die von den Benutzern ein zielstrebiges Handeln erfordern. Innerhalb eines Spiels können besonders großartige Errungenschaften erkämpft werden.

Gruppenarbeit
: Den Spielern werden Aufgaben gestellt die einzeln nicht zu bewältigen sind. Das Spiel fordert die Spieler dazu auf, sich zusammenzutun und eine Lösung zu finden in der nicht der einzelne die Verantwortung trägt, sondern die gesamte Gruppe.

Cascading Learning
: Dem Spieler werden nur Informationen mitgeteilt die zur Erledigung der aktuellen Teilaufgabe nötig sind.

Die Mechanismen haben den Effekt, dass jede Aktion des Spielers eine unmittelbare, eindeutig identifizierbare Bewertung erhält. Die Spielwelt bleibt dadurch immer leicht fassbar und nachvollziehbar, dadurch wird Stress der durch Unverständnis oder Überforderung erzeugt werden könnte, vermieden. Durch einen variablen Schwierigkeitsgrad werden Spieler aller Geschicklichkeitsstufen gleichermaßen gefordert. Die erwähnten Mechanismen zielen darauf ab eine für alle Benutzer gleichermaßen bewältigbare Aufgabe unmittelbar zu belohnen.

Dabei fällt auf, dass die Gamification-Mechanismen wie von der Flow-Theorie inspiriert wirken. Die vorangegangenen psychologischen Modelle erklären erstaunlich deutlich, welche Auswirkungen die Gamification, bei richtiger Anwendung, auf die Motivation eines Menschen haben kann. Durch einen variablen Schwierigkeitsgrad, kann eine Über- oder Unterforderung vermieden werden. Zusätzlich wird durch die unmittelbare Rückmeldung auf alle Handlungen dem Spieler eine vollständige Kontrolle über die Geschehnisse vermittelt. Die beiden Punkte wurden als wichtige Voraussetzungen für ein Flow-Erleben identifiziert. Über die Gamification-Merkmale des sichtbaren Status (Abzeichen), der Fortschrittsanzeige und der Rangliste können extrinsische Motivationsfaktoren, wie Anerkennung oder Zugehörigkeit nahezu perfekt ausgelöst werden. Durch die Verteilung von Belohnungen in Form von Punkten oder Spielgeld können zusätzlich extrinsische Motivationsfaktoren angesprochen werden. Falls ein Spieler eine längere Zeit in einem Spiel verbringt, können intrinsische Motive aktiviert werden, indem sich der Spieler unbewusst mit dem Spiel identifiziert.

Aus den aufgezählten Beispielen lässt sich ableiten, dass digitale Gamification verschiedene Belohnungen bereithält und es somit möglich ist Menschen unterschiedlichster Charaktere zu motivieren. Hinzu kommt, dass Belohnungen durch digitale Gamification im Gegensatz zu Belohnungen in der realen Welt, keine Kosten verursachen. Die aufgezählten Punkte liefern eine Erklärung für die Funktionsweise von Gamification, ebenso kann damit die steigende Popularität und Bedeutung begründet werden.

2.1.4. Ausgangslage für Gamification in der Produktion

Mit dem Beginn der Industrialisierung wurden Maschinen immer flächendeckender eingesetzt, was den Wechsel der Produktion von Qualität auf Quantität angestoßen hat und die Massenproduktion zur Folge hatte. Dadurch sind viele Arbeitsplätze entstanden, die ausschließlich aus wenigen, kurzen und anspruchslosen Handgriffen bestehen, um die geforderte Menge an Produkten bereitstellen zu können. Auch wenn in den letzten hundert Jahren viele dieser Arbeiten durch weitere Maschinen ersetzt werden konnten, gibt es heute noch eine enorme Masse an Arbeitsplätzen, die an eine von Menschenhand gefertigte Stückzahlproduktion gebunden sind. Die Arbeitsplätze setzen sich beispielsweise aus den Bereichen der industriellen Fertigung zusammen, in welchen noch keine passende Maschine erfunden wurde, die Anschaffung einer Maschine zu teuer wäre oder die Anpassungen der Produkte an Kundenwünsche oder die Flexibilität des Marktes so hoch ist, das es sich nicht lohnt Maschinen einzusetzen [Klu11].

Die hier angesprochenen Arbeiten haben folgende Eigenschaften gemeinsam, es sind einfache manuelle, auf hohe Stückzahlen ausgerichtete Tätigkeiten, die meist nur durch überdurchschnittlich hohe Bezahlung durchgeführt werden. Es gibt viele Untersuchungen darüber, dass körperliche Beschwerden bei dieser Art von Arbeit auftreten können, die beiden Paper beziehen sich auf Unterarmbeschwerden bei Bandarbeitern in der Automobilindustrie [BHWA95] [HsB97]. Vieles wurde im Hinblick auf den Abbau der körperlichen Strapazen dieser Arbeiten getan. Ein passendes Beispiel ist die Luftqualität in großen und kleineren Produktionsstätten, die sich kontinuierlich verbessert hat [KZa08]. Die psychischen Belastungen sind jedoch seit vielen Jahrzehnten die gleichen geblieben, an der Schichtarbeit und den immer gleichen Arbeitsschritten hat sich wenig bis nichts geändert. Wie erwähnt ist die Bezahlung oft der tragende Motivationsfaktor für die Mitarbeiter um dem Arbeitsplatz nicht den Rücken zu kehren [JS07].

Seit Jahren versuchen Arbeitgeber monotone Arbeiten abwechslungsreicher zu gestalten oder Motivationshilfen zu finden. Das Rotationsprinzip, so dass ein Mitarbeiter nie länger als einen bestimmtem Zeitraum eine Arbeit durchführen muss, Bonuszahlungen oder zusätzliche Urlaubstag für besonders schnelle, präzise oder unangenehme Arbeiten, sind nur einige Beispiele aus der gängigen Praxis. An der eigentlichen Arbeitstätigkeit kann aber meist wenig geändert werden [WG52].

In dieser Ausgangslage befinden sich die nachfolgenden Projekte, die sich mit Assistenz- und Gamificationansätzen in der Produktion beschäftigen.

2.2. MotionEAP

Nachdem die psychologische Erklärung für die Funktionsweise und Popularität von Gamification abgeschlossen wurde, wird im folgenden Abschnitt das Projekt motionEAP, das sich mit der Bewegungserkennung von Arbeitern in der Produktion beschäftigt, vorgestellt. An dem Projekt wird derzeit am Institut für Visualisierung an der Universität Stuttgart geforscht. MotionEAP ist aus dem Projekt ASLM entstanden, das zuerst vorgestellt wird.

2.2.1. Das Vorprojekt ASLM

MotionEAP ist aus dem Vorläuferprojekt ASLM (Assistenzsysteme für leistungseingeschränkte Mitarbeiter in der manuellen Montage), das am 01.10.2011 begonnen wurde und am 31.01.2013 zu Ende ging, hervorgegangen.

Im Fokus des Projekts stand die Aufwertung von manuellen Montageplätzen von leistungsgeminderten Mitarbeitern mittels Bewegungserkennung und die Projektion von Text, Bild und Video direkt im Arbeitsbereich. Neben einer anleitenden Funktion über die Projektion, ist auch eine Möglichkeit der Qualitätskontrolle durch die Bewegungserkennung verfügbar. Nach zahlreichen Studien konnte nachgewiesen werden, dass durch das System die Performance von leistungsgeminderten Mitarbeiter verbessert und die Montagedauer der Bauteile verringert wurde [KFS15b].

2.2.2. MotionEAP Vorstellung

Nach den positiven Ergebnissen und entdeckten Möglichkeiten die ASLM geliefert hat, wurde das Nachfolgeprojekt motionEAP ins Leben gerufen. MotionEAP ist ein *System zur Effizienzsteigerung und Assistenz bei Produktionsprozessen in Unternehmen auf Basis von Bewegungserkennung und Projektion* [mot16b]. Das Ziel des Projektes ist die Erforschung von Assistenzsystemen in Produktionsprozessen anhand von Prototypen und deren Evaluation. Es wird Wert darauf gelegt, die Forschung eng zusammen mit den zukünftigen Anwendern zu betreiben. Der Projektbeginn war am 01.01.2013 und wird nach aktuellem Stand voraussichtlich bis etwa Ende 2016 laufen.

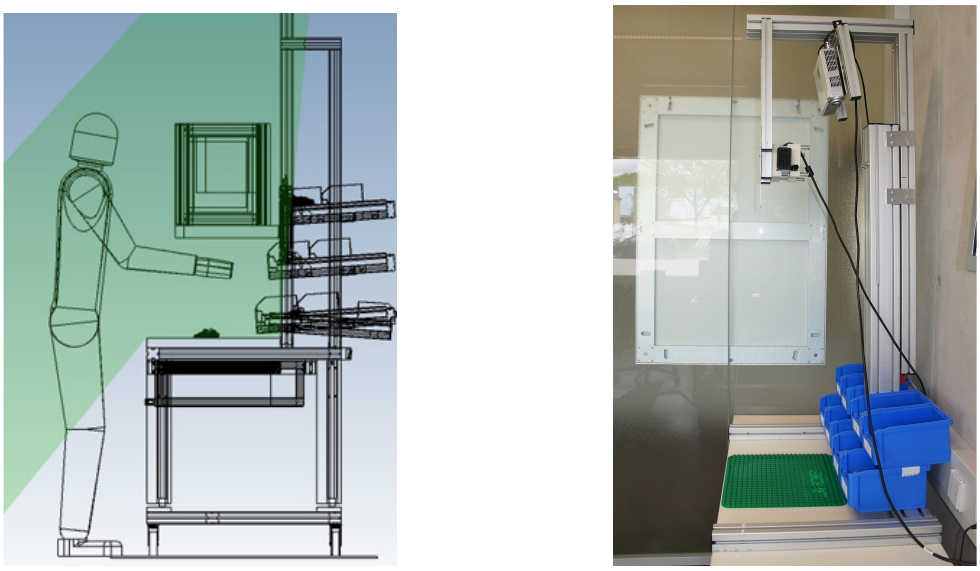


Abbildung 2.2.: Aufbau des Arbeitsplatzes in motionEAP. Links schematischer Aufbau, rechts tatsächlicher Aufbau mit Bewegungserkennung und Projektor über der Arbeitsfläche

Auf der schematischen Zeichnung (Abb. 2.2 (linke Abbildung) [mot16b]) kann man einen Arbeitsplatz für die manuelle Fertigungsproduktion erkennen. Der Mitarbeiter steht vor einem Arbeitstisch und bedient sich aus den Fächern auf der rechten Seite. In den Fächern befinden sich Einzelteile die zur Fertigung des gesamten Bauteils benötigt werden. Das Bauteil wird auf der Arbeitsfläche (grüne Fläche auf dem Tisch, rechte Abb.) vom Mitarbeiter zusammengesetzt. Soweit unterscheidet sich der Aufbau nicht von herkömmlichen Arbeitsumgebungen. Für motionEAP wird dem Aufbau eine Kamera mit Bewegungserkennung und ein Projektor hinzugefügt. Die beiden Geräte werden direkt über der Arbeitsfläche angebracht, erkennbar auf dem rechten Bild der Abbildung 2.2. Das Gerät auf der linken Seite vor dem weißen Hintergrund ist eine X-Box Kinect mit Bewegungserkennung, auf der rechten Seite ein Stück weit darüber befindet sich der Projektor. MotionEAP übersteigt in den folgenden Punkte den Stand der Technik:

- Bewegungserkennung mit unmittelbarer Rückmeldung
- Projektion direkt im Arbeitsbereich
- Implementierung motivierender Elemente (Gamification)

Besonders um den letzten Punkt *Implementierung motivierender Elemente (Gamification)* handelt es sich in der vorliegenden Arbeit. Die Gamification-Implementierung nimmt Bezug auf die Bewegungen des Benutzers. Das Projekt beschäftigt sich neben möglicherweise zukunftsweisenden Technologien im Bereich von Informatik, Maschinenbau und Elektrotechnik, zusätzlich mit arbeitsethischen und psychologischen Fragen. Das ist der Grund, weshalb an dem überwiegend technischen Projekt auch Psychologen und Philosophen aktiv beteiligt sind. Im nächsten Kapitel wird der aktuelle Stand des Projekts genauer erläutert.

3. Stand der Technik

In diesem Abschnitt wird der aktuelle Stand von Gamification und von motionEAP behandelt. Das Kapitel beginnt mit der Beschreibung der momentanen Implementierung von motionEAP.

3.1. Technische Besonderheiten von Gamification in der Produktion

Die Gamification in der Produktion enthält mehrere Besonderheiten im Gegensatz zu herkömmlichen Gamification-Implementierungen. Der Benutzer hat in der Produktion ein Ziel, nämlich die Produktion von Gütern. Von diesem Ziel sollte er so wenig wie möglich abgelenkt werden. Das hat zur Folge, dass die Interaktion zwischen dem Benutzer und der Gamification-Anwendung gering gehalten werden sollte. Im Gegensatz dazu steht das Ziel Motivation durch Gamification zu generieren. Dadurch wird es nötig ein System zu schaffen, dass durch implizite Interaktion gesteuert werden kann. Die Bedienung darf nicht durch zusätzliche Handgriffe erfolgen, wie es die Eingabe durch Maus, Tastatur oder Controller voraussetzen würden, sondern das Gamification-System muss auf die Arbeitsgeschwindigkeit, -qualität oder andere Faktoren Bezug nehmen.

Um die Ablenkung gering halten zu können, darf die grafische Benutzeroberfläche des Systems nicht zu komplex werden. Bewegende Elemente, komplexe Animationen oder ausufernde Grafikeffekte sollten nach Möglichkeit vermieden werden. Ferner spielt es eine große Rolle an welcher Stelle die Projektion des Systems stattfindet. Der Benutzer bzw. der Mitarbeiter sollte die Informationen ohne Anstrengung ablesen können. Das kann am besten durch eine Integration der Visualisierung in den Arbeitsplatz erreicht werden, sodass zum Ablesen der Implementierung keine zusätzlichen Bewegungen erforderlich sind. Der Verzicht auf zusätzliche Bewegungen gilt auch für die Steuerung einer Gamification-Anwendung in der Produktion. Bei Gamification in der Produktion ist es zusätzlich besonders wichtig, dass die Gamification unmittelbar und in unterstützender Weise auf Benutzereingaben reagiert. Durch die stark eingeschränkten Eingabemöglichkeiten ist es wichtig, dass der Benutzer zu jeder Zeit das Gefühl hat die Kontrolle über das System zu besitzen [KFS15a].

3.1.1. Unterschiedliche Gamificationimplementierungen in motionEAP

In diesem Abschnitt wird der aktuelle Stand der Gamification des im Kapitel 2 vorgestellte Projekt motionEAP beschrieben.

Bevor die einzelnen Implementierungen aufgezählt werden, wird zunächst die Funktionsweise von motionEAP erklärt. MotionEAP verfolgt den Bewegungsablauf eines Mitarbeiters beim Fertigen eines Bauteils. Die Bewegungen sind in einem vorher angefertigten Ablauf gespeichert. Dafür wird die genaue Reihenfolge und Fertigungszeit von jedem Handgriff benötigt. Zusätzlich ist im System

vermerkt, wie das zu fertigende Bauteil nach jedem Handgriff aussehen soll. Dem Mitarbeiter ist es somit nicht möglich die Handgriffe lediglich vorzutäuschen. Der gespeicherte Arbeitsverlauf eines Bauteils wird *Workflow* genannt. Sobald eine Abweichung des Mitarbeiters zu den eingespeicherten Handgriffen erfolgt, wird ein Fehler ausgegeben.

Über die Bewegungserkennung und den Projektor ist es möglich dem Mitarbeiter Informationen über seinen Arbeitsverlauf anzuzeigen. Dem Mitarbeiter könnte beispielsweise auf der Arbeitsfläche angezeigt werden wie viele Bauteile er im Verlauf des Arbeitstages oder Jahres gefertigt hat. Es wäre denkbar, dass die verbleibende Zeit zur Mittagspause, dem Feierabend oder dem nächsten Urlaub angezeigt wird. Man könnte einen begangenen Fehler durch eine Textausgabe, die das Wort *FEHLER* enthält signalisieren. Das wären alles mögliche Optionen, die dem Mitarbeiter durch das System auf der Arbeitsfläche angezeigt werden könnten. Die Intention des Systems ist jedoch die Produktivität des Mitarbeiters zu erhöhen. Das hängt bei der Fertigung von Bauteilen primär von zwei Variablen ab, nämlich der Arbeitsgeschwindigkeit und der Fehlerquote. Es stellt sich die Frage, wie man Mitarbeiter durch eine Visualisierung auf der Arbeitsfläche zu einer Verbesserung der beiden Variablen und einem erhöhten Arbeitseinsatz motivieren kann. Gamification enthält, wie im Kapitel 2 – Hintergrund beschrieben, eine Vielzahl an Möglichkeiten, die für dieses Ziel geeignet sind.

Falls es durch Gamification gelingt den Mitarbeiter zu höheren Leistungen zu motivieren, wurde ein Ziel von motionEAP, die Effizienzsteigerung von Produktionsprozessen, erreicht (U3). Die Erhöhung der Produktivität der Mitarbeiter wird derzeit über drei verschiedenen Gamification-Implementierungen getestet. Die Implementierungen werden in den folgenden Abschnitten detailliert vorgestellt. Zunächst werden Elemente aufgezählt, die alle Implementierungen gemeinsam haben. In allen Implementierungen wird das Element der Zeit- und der Fortschrittsanzeige verwendet. Der Benutzer soll auf einen Blick erkennen, ob die Fertigung des Bauteils innerhalb der Fertigungszeit liegt oder nicht. Dafür wird der zeitliche Verlauf in allen Implementierungen durch den farblichen Verlauf von grün über gelb über orange zu rot visualisiert. Das bedeutet, so lange das Element für die Zeitanzeige grün ist, befindet sich der Mitarbeiter innerhalb der Fertigungszeit. Gelb bzw. orange stehen für eine befriedigende bzw. ausreichende Arbeitsgeschwindigkeit. Falls sich das Element rot färbt, wird dem Mitarbeiter signalisiert, dass die Fertigungszeit überschritten wurde.

Ein weiteres gemeinsames Element ist die Fehleranzeige. Falls das Element für die Anzeige eines Fehlers grün dargestellt wird, arbeitet der Mitarbeiter fehlerlos. Sobald sich das Element rot färbt, ist dem Mitarbeiter ein Fehler unterlaufen. Momentan wird ein Fehler angezeigt, wenn der Mitarbeiter die Fertigungsreihenfolge nicht einhält. Es folgt die Vorstellung der ersten der drei Gamification-Implementierungen in motionEAP.

Tetris

Abbildung 3.1.: Skizze der vorhandenen Tetris Gamification

Für die Tetris-Implementierung ist an dieser Stelle eine kurze Erläuterung des Tetris Prinzips notwendig. Das Spiel *Tetris* wurde im erstmals im Jahre 1984 auf einem digitalen Medium, einem Elektronika-60-Rechner, veröffentlicht. Es hat sich seitdem etwa 100 Millionen Mal verkauft, wurde vielfach mit Auszeichnen überhäuft und kann somit gut als einer der größten Klassiker der Computergeschichte bezeichnet werden. Das Spielfeld ist in den gängigsten Varianten etwa 20 Zeilen hoch und zehn Spalten breit. In der ursprünglichen Version fallen sieben verschiedene Steine vom oberen Bildschirmrand herunter. Der Spieler muss die Steine noch bevor diese den Boden erreichen, in der Luft so positionieren, dass keine Lücken entstehen. Falls der Spieler es geschafft hat, die Spielsteine so anzuordnen, dass in einer Zeile keine Lücke entsteht, wird diese Zeile gelöscht und es werden Punkte gutgeschrieben. Das Ziel ist es, so viele Punkte wie möglich zu erhalten. Es gibt somit kein wirkliches Spielende. Die Spieler versuchen sich in der Anzahl ihrer Punkte gegenseitig zu übertrumpfen [Kar07].

Auf diesem Prinzip baut die erste Implementierung auf (Abb. 3.1), die auch chronologisch als erste Gamification-Implementierung in motionEAP ihren Einzug fand. Pro Arbeitsschritt fällt ein Stein herunter, der Mitarbeiter braucht diesen allerdings im Gegensatz zum klassischen Spiel nicht selbst anzuordnen, sondern der Stein fällt in einem festgelegten Ablauf herunter. Der Stein färbt sich in der

Luft im oben erwähnten Farbschema, hier kennzeichnet der farbliche Verlauf die Arbeitsgeschwindigkeit. Im Falle eines Fehlers wird der Stein komplett rot eingefärbt. Falls ein Bauteil abgeschlossen ist, wird das Spiel neu gestartet.

Kreis & Balken

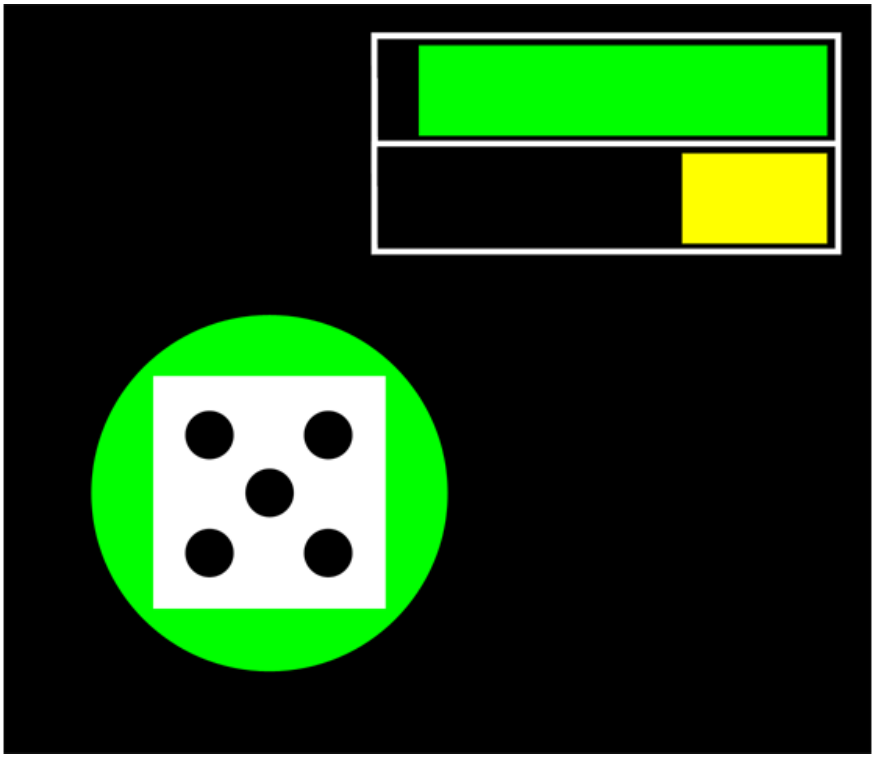


Abbildung 3.2.: Skizze der vorhandenen Kreis & Balken Gamification

Diese Implementierung ist aus zwei Teilen aufgebaut, der Balken und der Kreis (Abb. 3.2). Der Kreis färbt sich abhängig von der Arbeitsgeschwindigkeit im gewohnten Farbschema von grün zu rot. Der Würfel im Kreis zeigt dabei die Nummer des Arbeitsschrittes an. Falls man den ersten Schritt durchführt, wird eine Eins angezeigt, für den zweiten eine Zwei, für alle weiteren Schritte die entsprechende Schrittzahl. Nach einem Arbeitsschritt wird der untere Balken mit der Farbe des Kreises aufgefüllt. Der obere Balken gibt dabei den Verlauf der letzten Bauteile an. Dieser Balken fungiert somit als Fortschrittsbalken über die vergangenen Schritte. Falls der Balken eine deutlich grüne Färbung aufweist, kann der Proband mit einem Blick ablesen, dass das aktuelle Bauteil in einer sehr guten Zeit gefertigt wurde. Ist der Balken hingegen überwiegend rot eingefärbt, wird dem Mitarbeiter signalisiert, dass er entweder sehr langsam ist oder viele Fehler macht. Hier steht eine rote Färbung ebenso wie in der Tetris-Implementierung für einen Fehler. Falls ein Bauteil abgeschlossen

ist, wird die Komponente zurückgesetzt, der Würfel zeigt wieder eine Eins an und der untere Balken ist leer.

Pyramide

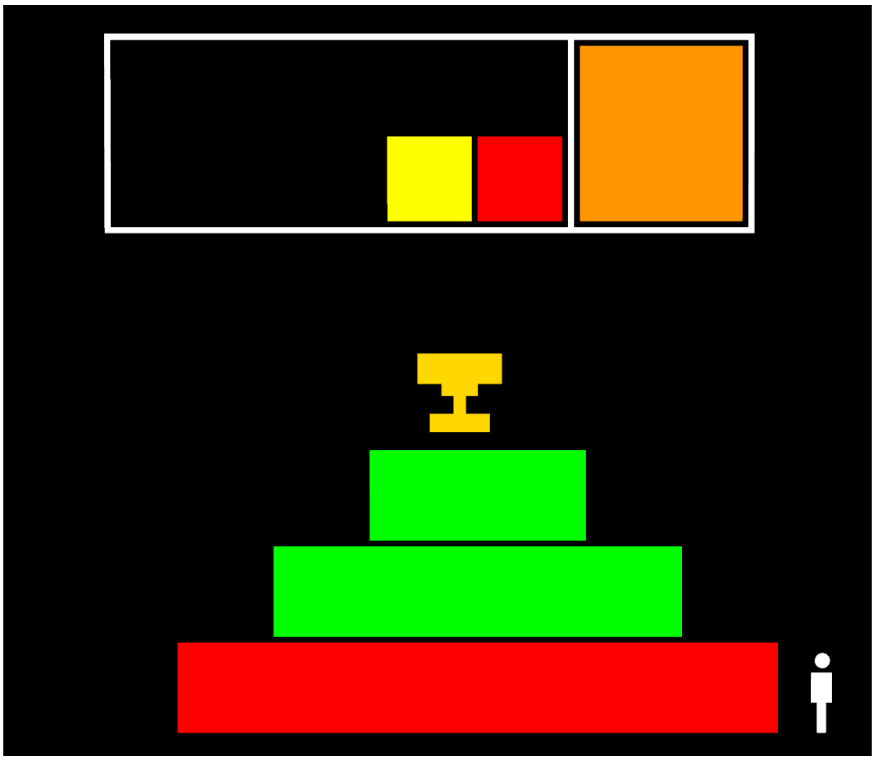


Abbildung 3.3.: Die vorhandene Pyramiden Gamification

Dieser Ansatz besteht aus einer Pyramide, einer Spielfigur und einer Verlaufsanzeige (Abb. 3.3). Die Anzahl der Stufen der Pyramide zeigen die Gesamtanzahl der Arbeitsschritte pro Bauteil an. Besteht die Pyramide beispielsweise aus vier Blöcken, dann braucht man vier Arbeitsschritte um ein Bauteil herzustellen. Die Spielfigur befindet sich zu Beginn am Fuße der Pyramide und steigt stufenweise auf. Falls man am Ende des Bauteils angelangt ist, erwartet die Spielfigur ein Pokal. Den Pokal erhält der Spieler nur dann, wenn das gesamte Bauteil ohne einen Fehler gefertigt wurde. Die einzelnen Stufen der Pyramide färben sich wieder nach dem gewohnten Farbschema von grün zu rot ein, abhängig von der Arbeitsgeschwindigkeit der Mitarbeiters. Falls dem Proband ein Fehler unterläuft, wird die entsprechende Pyramidenstufe komplett rot eingefärbt. Nachdem ein Bauteil gefertigt wurde, wird über der Pyramide, ähnlich Balken & Kreis Komponente, ein Verlauf angezeigt. Der Verlauf wird in dieser Implementierung mithilfe unterschiedlich gefärbter Blöcke visualisiert. Es werden maximal zehn Blöcke angezeigt, diese geben somit eine Übersicht über die vergangenen zehn Bauteile. Die Blöcke werden, falls kein Fehler unterläuft, im Farbdurchschnitt der Pyramide eingefärbt. Falls die

eine Hälfte der Pyramide grün und die andere dunkel orange oder rot ist, wird der Block demnach gelb. Hier gilt dasselbe Prinzip wie bei der vorigen Implementierung auch, falls ein Fehler passiert, wird der Verlaufsblock, unabhängig davon ob die Pyramide sonst nur grün ist, rot eingefärbt.

3.2. Assistenzsysteme

Die in den vorangegangenen Abschnitten erwähnte Implementierungen wird in der Publikation *Design Approaches for the Gamification of Production Environments. A Study Focusing on Acceptance* (O. Korn et al., 2015) auf Akzeptanz überprüft. Erprobt wurden die Implementierungen Kreis & Balken und Pyramide, mithilfe leistungsgeminderten Mitarbeiter. Die Tätigkeit der Mitarbeiten besteht darin, größere Bauteile aus kleineren Einzelteilen zusammenzubauen. Es wurde überprüft, ob diese Tätigkeit mit einer Gamification-Visualisierung, am Rande der Arbeitsfläche eine Verbesserung erfährt. Dabei wurde das Pyramidenkonzept von den Mitarbeitern favorisiert [KFS15a].

Um Gamification in ein Produktionssystem zu implementieren, ist es nötig bestimmte Voraussetzungen an einem Arbeitsplatz zu schaffen (U3). Die Arbeitsplätze und Arbeitsumgebungen müssen so verändert werden, dass es möglich ist Gamification zu integrieren. Oft endet dies darin, dass Arbeitsumgebungen neu aufgebaut werden müssen, um den Systemen eine Basis zu ermöglichen. Welche Voraussetzungen dafür im Speziellen nötig werden beschreibt die Ausarbeitung *Assistive system experiment designer ASED: A toolkit for the quantitative evaluation of enhanced assistive systems for impaired persons in production* (O. Korn et al., 2012) [KSHK12].

Überwiegend der Teilaspekt der Projektion von Informationen in Arbeitsumgebungen in der Produktion wird in der Ausarbeitung *Assisitive Augmentation at the Manual Assembly Workplace using In-Situ Projection* (O. Korn et al., 2014) behandelt. Es werden Möglichkeiten der in-situ Projektion und der Bewegungserkennung angesprochen und eine Empfehlung zu verschiedenen Systemen gegeben. Dieses Projekt wurde überwiegend mit leistungsgeminderten Mitarbeitern durchgeführt, allerdings wurde das Ziel formuliert, die Systeme allen Menschen zugänglich zu machen [KFS14].

Eine umfassende Erklärung für kontextbewusste Assistenzsysteme in der Produktion wird in der Doktorarbeit *Contextaware assistive systems for augmented work. A framework using gamification and projection* von Dr. Oliver Korn beschrieben. Darin wird auch eine erste exemplarische Implementierung eines kontextbewussten Systems in einer Produktionsumgebung realisiert. Die Arbeit bildet einen Anfangspunkt auf dem Feld der kontextbewussten Assistenzsysteme [Kor14].

3.3. Gamification

In diesem Abschnitt werden aktuelle Entwicklungen zum Thema *Gamification* vorgestellt.

In der Veröffentlichung *Assistive Systems in Production Environments: Exploring Motion Recognition and Gamification* wird festgestellt, dass sich Systeme mit Bewegungserkennung in der Produktion für Gamification eignen könnten. Die Kombination aus Bewegungserkennung und Gamification wird darin in ersten Implementierungsversuchen vorangetrieben [SKH12].

Das Buch *Gamification by Design* (G. Zicherman, C. Cunningham, 2011) gibt eine übersichtliche Einführung zum Thema Gamification. Es bietet eine Übersicht der verschiedenen Gamification Methoden an und führt Neueinsteiger in das Thema ein. Die letzten Kapitel beschreiben aktuelle Implementierungen und Vorgehensweisen von Gamificationelementen in Mobile Apps und Webanwendungen [ZC11].

Eine Bestandsaufnahme der aktuellen Entwicklung mit Ausblick auf die kommende Zukunft hat ein Autorenkollektiv im Paper *Gamification: Using Game Design Elements in Non - Gaming Contexts* auf einer Konferenz, im Rahmen eines Workshops, erarbeitet. Die beiden angesprochenen Themen, die Feststellung der aktuellen Entwicklung und der potentiellen Möglichkeiten der Zukunft, geben einen aktuellen Überblick [DOS+11].

Zahlreiche Studien, die sich mit den Auswirkungen von Gamification auf die Motivation und das Verhalten der Benutzer beschäftigen, wurden im Paper *Does Gamification Work?* (J. Hamari et al., 2014) analysiert. Es wurde festgestellt, dass in den Jahren 2010 bis einschließlich 2013, die Anzahl der Paper die sich mit dem Thema Gamification beschäftigten, kontinuierlich stieg. Diese Metastudie vergleicht, nachdem verschiedene Auswahlkriterien besprochen wurden, die Ergebnisse von insgesamt 24 Studien. Das Ergebnis der Untersuchung zeigt, dass es eindeutige Indizien für eine positive Wirkung von Gamification gibt, wobei es sehr stark auf die untersuchten Teilnehmer und den Kontext ankommt. Besonders deutlich wird an dieser Stelle, wie jung das gesamte Thema ist, so haben die Forscher im Jahre 2010 gerade einmal etwa hundert Veröffentlichungen gezählt, im Jahre 2013 waren es hingegen an die 2300 [HKS14]. Eine Ergänzung zur vorhergehenden Studie *Does Gamification Work?* (J. Hamari et al., 2014), hinsichtlich des Alters und der Herkunft von Gamification bietet die Arbeit *Soviet and American Precursors to the Gamification of Work* (M. J. Nelson, 2012). Der Autor untersucht die Vorgänger der modernen Gamification an den Beispielen der Sowjetunion und den Vereinigten Staaten von Amerika im 20. und Anfang des 21. Jahrhundert. Er beschreibt darin den sozialistischen und im Gegensatz dazu den amerikanischen Versuch Wettbewerbe zwischen Fabriken zu entfachen und somit die Grenze zwischen Arbeit und *Spiel* zu verwischen. Beide Ansätze werden erforscht und auf ihre Vor- und Nachteile untersucht [Nel12].

Mithilfe einer Studie an Studenten wurde in der Arbeit *An empirical study of gamification impact on e-Learning environment* (A. Amriani et al., 2013) versucht herausfinden, wie sich Gamification auf das Lernverhalten auswirkt. Für die Studie wurde ein soziales Netzwerk und ein Forum mit Gamificationelementen programmiert. Die Studenten wurden daraufhin in drei Gruppen aufgeteilt: klassischer Lernansatz ohne spezielles Netzwerk oder Gamification, mit Gamification und mit sozialem Netzwerk. In dieser Studie schnitt die Gruppe, welche sich regelmäßig im Forum mit Gamification ausgetauscht hat besser ab, als mit dem klassischen Ansatz. Voraussetzung dafür war allerdings, dass die Studenten regelmäßig die Lehrveranstaltungen besucht haben [AAUJ13].

Eine wichtige Frage stellt sich *Scott Nicholson* im Abschnitt *Exploring the endgame of gamification* im Buch *Rethinking Gamification.* Die Frage lautet, was passiert mit den Benutzern eines Gamificationsystems, welche einen Spielzyklus durchschritten haben. In Anlehnung an das *Endgame* (Fortgeschrittener Spielstand in Computerspielen), stellt er Überlegungen an, wie man die Benutzer auf lange Sicht motivieren kann. Er sieht eine Möglichkeit der Motivation in der Perspektive, dass die Benutzer im realen Leben für ihre Erfolge belohnt werden [FFRS14a].

Eine kurze Einführung in die Geschichte der Gamification und in das Punkte- bzw. Tokensystem im Speziellen, gibt das Paper *Making points the point: towards a history of ideas of gamification* (M. Fuchs et al., 2014). Computerspiele werden seit jeher mit Punkten, Gold oder ähnlichen Belohnungen in Verbindung gebracht, woher das kommt und weshalb die Spiele vermutlich gerade dadurch so erfolgreich geworden sind wird darin erklärt. Eine mögliche Erklärung für den Erfolg von Punkten und ähnlichen Belohnungen, ist die Imitation von extrinsischer Motivationsfaktoren [FFRS14b].

Eine der wenigen kritischen Stimmen zu Gamification, die in einer wissenschaftlichen Abhandlung geäußert wurden, findet sich in *Gamification and Post-Fordist Capitalism* (S. Walz et al., 2015). Darin wird die Frage aufgeworfen ob es moralisch vertretbar ist, dass Arbeiter bei ausbeuterischen Tätigkeiten Spaß empfinden sollen. Gamification wird darin als lächelnder Dieb bezeichnet. Es wird zum Ausdruck gebracht, dass Gamification im Kapitalismus als moralisch nicht unbedenklich gewertet werden kann [WD15].

4. Akzeptanz-Studie

Im Rahmen dieser Arbeit wurde eine Studie in Ingolstadt bei der AUDI AG durchgeführt. In der Studie wurden den Teilnehmern die aktuellen Gamification-Implementierungen vorgeführt und anschließend ihre Meinungen dazu abgefragt. Die 21 Teilnehmer, davon vier Personaltrainer und 17 Montagemitarbeiter, haben einen Fragebogen erhalten, welchen sie vor und nach dem Experiment bearbeiten sollten. Das Experiment bestand daraus, den Mitarbeitern Videoausschnitte zu zeigen, welche die Funktionsweisen der in motionEAP verwendeten Gamification-Implementierungen demonstrieren. Aktuell werden die Implementierungen Tetris, Kreis & Balken und Pyramide genutzt. Die Videos wurden den vier Personaltrainern in Einzelgesprächen im Februar 2016 gezeigt. Den Montagemitarbeitern hingegen wurden die Videos in Kleingruppen, der Größe von durchschnittlich vier Personen, vorgeführt. Das Experiment mit den Montagemitarbeitern wurde etwa zwei Monate später durchgeführt.

4.1. Vorstellung und Durchführung der Studie

Die Studie ist unter Berücksichtigung der Richtlinie ISO 9241-210:2010 erstellt und durchgeführt worden. Die Norm enthält Anforderungen und Empfehlungen für nutzerzentrierte Design-Prinzipien von computergestützten, interaktiven Systemen [ISO10].

4.1.1. Durchführung des Experiments

Im Rahmen der Studie wurde ein Experiment durchgeführt, das daraus bestand den Teilnehmern ein Video vorzuführen und sie anschließend nach ihrer Einstellung dazu zu befragen. Die Befragung wird mithilfe eines Fragebogens durchgeführt der in Anhang – A abgedruckt ist. Das Ziel der Befragung ist die Abfrage der Meinungen der Teilnehmer gegenüber den Videos. Die insgesamt 32 Fragen des Fragebogens gliedern sich in zwei Teile, dem quantitativen und dem qualitativen Teil. Im erstgenannten Teil werden 18 Fragen gestellt, deren Antwortmöglichkeiten mithilfe einer Likert-Skala beantwortet werden können. Die Likert-Skala geht auf den amerikanischen Psychologen Rensis Likert zurück und wird genutzt um Einstellungen von Studienteilnehmern zu erfassen. Üblicherweise besitzen die Skalen fünf, sieben oder elf Abstufungen. In dieser Studie wurde eine fünfgliedrige Skala genutzt. Jeder Antwortmöglichkeit wird eine Punktzahl zugewiesen, wobei hier eine besondere Abneigung mit einem Punkt und besondere Zuneigung mit fünf Punkten gewertet wird. Durch die Vorgehensweise, die Meinungen der Teilnehmer auf der Likert-Skala festzuhalten, wird die Möglichkeit geschaffen Ergebnisse berechnen zu können, aus welchen Schlussfolgerungen gezogen werden können [ABB12]. Die Ergebnisse und das Fazit der vorliegenden Studie wird am Ende des Kapitel vorgestellt.

Das hier verwendete Beantwortungsmuster sieht wie folgt aus: stimmt überhaupt nicht (1), stimmt nicht (2), neutral (3), stimmt (4) und stimmt genau (5). Die erste Frage kann mit den folgenden Möglichkeiten beantwortet werden, sehr gut (1), gut (2), normal (3), nicht gut (4) und schlecht(5). Darauf folgen wiederum zwei Fragen die vom erstgenannten Muster folgendermaßen abweichen, gar nicht (1), <2 Stunden (2), 2-5 Stunden (3), 5-10 Stunden (4) und über 10 Stunden (5). Die Zahlen in der Klammer hinter den Antwortmöglichkeiten geben die Wertung in der Likert-Skala an.

Der Qualitative Teil besteht aus sechs offenen Fragen, welche die Teilnehmer in eigenen Worten beantworten. Der Fragebogen für die Montagemitarbeiter ist etwas kürzer, als der Fragebogen für die Personaltrainer, welche Fragen speziell gekürzt wurden, wird im Folgenden behandelt.

4.1.2. Vorstellung des Fragebogens

Zu Anfang des Fragebogens (Abb. Anhang A.1) werden die Teilnehmer gebeten ihr Alter anzugeben. Für die vier Personaltrainer ist vorgesehen ihr genaues Alter anzugeben, für die Montagemitarbeiter gibt es drei Sparten: <30, 30-40, >40. Die Änderung wurde gemacht, um dem Wunsch nach höherer Anonymität nachzukommen.

Der Fragebogen wird mit einer Frage zum aktuellen Gemütszustand (Frage 01 A) fortgesetzt. Der Punkt unterscheidet sich zu den restlichen Fragen in den Antwortmöglichkeiten. Es ist möglich die Frage mit: sehr gut, gut, normal, nicht gut, und schlecht zu beantworten. Die Skalierung wurde so gewählt, dass das gesamte Befindlichkeitsspektrum abgedeckt wird.

Anschließend wird abgefragt, wie viele Stunden der Teilnehmer im Schnitt pro Woche vor dem Computer in der Arbeit und im privaten Umfeld verbringt. Auch diese beiden Punkte unterscheiden sich zu den üblichen Fragen in der Antwortmöglichkeit, die möglichen Antworten sind folgende: gar nicht, <2 Stunden, 2-5 Stunden, 5-10 Stunden, über 10 Stunden. Um einen Eindruck zu erhalten ob sich die Probanden gar nicht oder nur sehr selten vor dem Computer befinden wurde mit den ersten beiden Antwortmöglichkeiten abgedeckt. Im Falle, dass die Teilnehmer länger am Computer arbeiten bzw. sich damit in ihrer Freizeit länger beschäftigen wird mit den verbleibenden Antworten abgedeckt.

Als letztes wird in diesem Abschnitt die allgemeine Haltung zu Computerspielen und die Offenheit spielerische Elemente in Arbeitsprozesse bzw. speziell in die Montage bei der AUDI AG einzubinden untersucht. Nach den allgemeinen Fragen, beginnt das Experiment den Teilnehmern werden die Videos gezeigt.

Visuelle Gestaltung

Die Abbildung (Abb. Anhang A.2) zeigt den Abschnitt des Katalogs, welcher erst nach dem Experiment gestellt wird. Die Fragen 07T - I3 beziehen sich auf die visuelle Gestaltung und die Spielmechanik der Gamification-Implementierungen. Anhand dieser Fragen wird primär die technische Akzeptanz abgefragt. Die Fragen 07T, 08K und 09P, unterscheiden sich nur dadurch, dass die Haltung zu den unterschiedlichen Implementierungen abgefragt wird, die Frage bleibt dieselbe. Die Fragen sind nummeriert und an einigen Stellen in dreier Blöcken gestellt. Dabei steht der Buchstabe T hinter der

Nummer für die Implementierung Tetris, der Buchstabe K für Kreis & Balken und der Buchstabe P für die Pyramide. Dieses Muster wird im weiteren Verlauf des Fragebogens beibehalten. Im Folgenden wird einzig der Hintergrund der ersten Frage des dreier Blocks erläutert, da die verbliebenen zwei vom Sinn her gleich sind. Die Frage I1 beinhaltet die erste offene Frage. Hier sollten die Teilnehmer mit eigenen Worten beschreiben welche geeignete Visualisierungen sich wünschen würden. In den Fragen 10A und 11A wird abgefragt an welchem Ort die Implementierungen dargestellt werden sollten. Wünschen sich die Teilnehmer beispielsweise eine Projektion direkt auf ihre Arbeitsfläche oder sollte eine zusätzliche Fläche für die Projektion geschaffen werden? Die Implementierung Tetris wurde im Video auf einem Bildschirm neben dem Arbeitsbereich gezeigt. Kreis & Balken wurden im Video direkt auf der Arbeitsfläche angezeigt. Die Pyramide wurde auf einem Brett direkt vor der Arbeitsfläche projiziert. Die Teilnehmer hatten somit die Chance, verschiedene Orte der Projektion im Experiment zu betrachten.

Spielmechanik

Die Frage 12T, 13K und 14P beziehen sich auf die Verständlichkeit des Bewertungsmechanismus der einzelnen Implementierungen. Die Teilnehmer können hier zum Ausdruck bringen ob sie die Bewertungsmechanismen intuitiv fassbar waren oder ob das Gegenteil der Fall ist. In diesem Punkt wurde bewusst die Formulierung *leicht verständlich* gewählt, da eine einfache Verständlichkeit der Implementierungen unerlässlich ist. Die Probanden sollen auf einen Blick erkennen ob sie positiv oder negativ bewertet werden, dabei soll eine unnötige Ablenkung vermieden werden. Nach der offenen Frage I2, in der sich die Probanden verschiedene Bewertungsmechanismen ausdenken und notieren dürfen, wird in den Fragen 15T - 17P, die mögliche Motivation durch eine Gamification-Implementierung abgefragt. Diese wichtige Fragen, könnten Aufschluss über die mögliche Zusatzmotivation der Implementierungen geben. Die Bedeutung der Motivation in Gamification wurde im Kapitel 2 – Hintergrund erörtert. Die Frage I3 ist auf die vorhanden drei Gamification-Implementierungen bezogen, allerdings können die Befragten auch eine neue Art der Gamification nennen.

Akzeptanz

Die Fragen 18T - I6 behandeln die Frage nach der Akzeptanz der gezeigten Implementierungen, siehe Abb. Anhang A.2. Falls die Befragten von einer Implementierungen überzeugt sind und diese einem Kollegen empfehlen würden, kann das in den Fragen 18T - 20P zum Ausdruck gebracht werden. Die Frage ob sie es aus Begeisterung empfehlen würden oder nur deshalb weil Gamification nichts für sie selbst ist, sie aber einen Kollegen kennen, dem sie es empfehlen würden, kann mithilfe der Beantwortung der offenen Fragen geklärt werden. Wie die Gamification aussehen sollte um Empfehlenswert zu sein, wird in der anschließenden Frage geklärt. Welche Komplexität der Gamification angemessen ist und ob die Implementierungen zu simpel sind oder gar von der Arbeit ablenken würden, kann in den folgenden sieben Fragen beantwortet werden. Abschließend ist in Frage I6 gefragt, welcher Grad der Ablenkung für vertretbar gehalten wird. Die offene Frage I6 ist im gekürzten Fragebogen für die Montagemitarbeiter nicht vorhanden.

Nach der Präsentation: Veränderung der Haltung

Am Ende des Fragebogens (Abb. A.3) werden die Fragen 05A und 06A wiederholt. Es könnte sein, dass sich durch das Experiment, der Eindruck von Gamification verändert hat. Dies kann bei Punkt I7, der wiederum für die Montagemitarbeiter gekürzt wurde, nochmals ausführlich geäußert werden. In der letzten offenen Frage, wird nach Hemmnissen und Ansatzpunkten gefragt. Bei dieser Frage können die Teilnehmer Sorgen oder Ideen zum Thema notieren. Abschließend wird der Punkt erörtert ob für die Teilnehmer Gamification in der Produktion prinzipiell denkbar wäre, falls es ein passendes Design geben würde.

Der Unterschied der beiden Fragebögen, für Personaltrainer bzw. für die Montagemitarbeiter, äußert sich in den drei Fragen 00A, I6 und I7. Die offenen Fragen I6 und I7 wurden im Fragebogen für die Montagemitarbeiter gekürzt. Die Frage 00A betrifft das Alter. Es wurden Sorgen zur Wahrung der Anonymität von Seiten der AUDI AG geäußert. Dadurch wurde die absolute Angabe des Alters abgeschafft und Alterskategorien eingeführt.

Nachdem die einzelnen Punkte des Fragebogens vorgestellt und die Durchführung und der Verlauf der Studie präsentiert wurden, folgen im kommenden Abschnitt die Ergebnisse.

4.2. Quantitativer Ergebnisteil

Die Fragen der Studie teilen sich in quantitative und qualitative Fragen. Den Anfang bildet die Auswertung der quantitativen Befragung. Die Untersuchung der Ergebnisse beginnt mit der Frage, ob bestimmte Korrelationen oder Abhängigkeiten innerhalb oder zwischen bestimmten Gruppen festgestellt werden können.

4.2.1. Auswertungsmethoden

Die Auswertung wurde mithilfe einer Varianzanalyse (ANOVA) durchgeführt. Wenn der hierbei ermittelte p-Wert kleiner als 0.05 (α = 5 %) ist, kann die Nullhypothese verworfen werden. Die Nullhypothese besagt, hier in allen Fällen, dass sich die Mittelwerte der Antwortmöglichkeiten der Gruppen nicht unterscheiden [Bor05]. Im Folgenden wird untersucht ob es signifikante Unterschiede in den Antworten von bestimmten Gruppen gibt. Es wird untersucht ob beispielsweise das Alter oder die Affinität zu Computerspielen einen signifikanten Einfluss auf die Antworten hat. Dafür werden in der Untersuchung nach dem Einfluss des Alters, die Antworten der jüngsten und ältesten Teilnehmer betrachtet. Die Antworten der beiden Gruppen (jung/ alt) werden mithilfe einer ANOVA untersucht, wodurch sich ein signifikanter Unterschied feststellen lässt. Die Konsequenzen der Auswertung werden im entsprechenden Abschnitt erläutert. In den folgenden Abschnitten werden vier Hypothesen aufgestellt.

4.2.2. Hypothesen

Die erste Unterteilung wird anhand des Alters der Teilnehmer gemacht. Die erste Hypothese lautet: *Die jüngeren Teilnehmer werden positiver auf die Fragen Antworten als die älteren Teilnehmer.* Dadurch, dass die jüngere Generation größtenteils mit Computer- und Konsolenspielen aufgewachsen ist und somit eine Akzeptanz und Affinität dazu aufgebaut hat, wird die deutlich offener und positiver auf die Fragen antworten.

Die nächste Unterteilung wird anhand der Tätigkeit gemacht. Die Antworten der Montagemitarbeiter und die der Personaltrainer werden gesondert untersucht. Die Hypothese dazu lautet: *Die Personaltrainer werden negativer auf die Fragen Antworten als die Montagemitarbeiter.* Durch die monotone Arbeit der Montagemitarbeiter werden diese positiv auf die Einführung einer neuen Technologie reagieren. Die Montagemitarbeiter erhoffen sich durch die Gamification eine Aufwertung ihres Arbeitsplatzes. Dadurch, dass die Personaltrainer im Arbeitsalltag mit der neuen Gamification-Technologie nicht in Berührung kommen werden diese eher zurückhaltend und negativer reagieren.

Die dritte Unterteilung wird über die verbrachte Zeit der Teilnehmer am Computer durchgeführt. Die Hypothese dazu lautet: *Die Teilnehmer, die viele Stunden vor dem Computer verbringen werden negativer auf die Fragen antworten, als die Teilnehmer die wenig Zeit davor verbringen.* Die Mitarbeiter die viel Zeit vor dem Computer verbringen werden ablehnend auf die Fragen antworten, da sie nicht noch mehr Zeit vor digitalen Anwendungen verbringen möchten. Die Mitarbeiter, die nur wenig Zeit vor einem Computer verbringen werden kein Problem damit haben mehr Zeit mit digitalen Technologien zu verbringen.

Die letzte Unterteilung wird aufgrund der Affinität zu digitalen Spielen gemacht. Die Hypothese lautet: *Die Teilnehmer die Computerspiele mögen werden auch Gamification positiv bewerten. Teilnehmer die Computerspiele nicht mögen, werden auch mit Gamification nichts anfangen können.* Es ist davon auszugehen, dass jeder Teilnehmer schon einmal mit Computerspielen in Berührung gekommen ist und sich somit eine Meinung zum Thema gebildet hat. Ob dem Teilnehmer Computerspiele gefallen oder nicht wird eine Auswirkung auf die Bewertung der hier vorgestellten Gamification-Technologie haben.

In den folgenden Abschnitten werden die vorgestellten Hypothesen geprüft. Nach der Präsentation der Ergebnisse folgt eine Diskussion der Hypothesen.

Alter

Da Bedenken zur Sicherstellung der Anonymität im Verlauf der Studie geäußert wurden, ist die Frage des Alters geändert worden, statt einem absoluten Wert, können die Teilnehmer eine Kategorie ankreuzen. Die Personaltrainer haben einen absoluten Wert angegeben, die Montagemitarbeiter später hingegen eine Kategorie angekreuzt. Die Antwortmöglichkeiten lassen sich als einzige nicht problemlos miteinander vergleichen. Der Altersdurchschnitt der Personaltrainer liegt bei 39,8 Jahren die Montagemitarbeiter haben im Schnitt 2,2 angekreuzt, was der Kategorie 30-40 entspricht. Jedoch war die Streuung der Personaltrainer deutlich geringer, deren Alter bewegt sich zwischen 37 und 46, die Montagemitarbeiter setzten sich aus jungen Auszubildenden und sehr erfahrenen Mitarbeitern zusammen. Aus allen Teilnehmern wurden zwei Gruppen gebildet, die Gruppe der unter 30 jährigen

und die der über 40 jährigen. Die Gruppen wurden so gebildet um zu erfahren ob zwischen den jungen und den älteren Mitarbeitern ein Unterschied in den Antworten der Umfrage besteht. Für diese Frage wurde die Gruppe der 30-40 jährigen ausgeblendet. Mithilfe einer Varianzanalyse (ANOVA) über alle Antwortmöglichkeiten, der beiden Gruppen <30 und >40 der Teilnehmer konnte festgestellt werden, dass es keinen signifikanten Unterschied zwischen den beiden Gruppen gibt ($p > .63$).

Tätigkeit

Es wurden zwei weitere Gruppen identifiziert, die der Personaltrainer und die der Montagemitarbeiter. Eine ANOVA über den Mittelwert aller Antwortmöglichkeiten zwischen den Gruppen der Personaltrainern und der Montagemitarbeiter ergab keinen signifikanten Unterschied ($p > .08$).

Verbrachte Zeit vor dem Computer

Die ersten zwei Fragen 02A und 03A beziehen sich darauf, wie viele Stunden ein Teilnehmer in der Woche privat oder geschäftlich vor einem Computer bzw. einer Konsole sitzt. Um herauszufinden ob sich die Antworten der beiden Extrema unterscheiden wurden die folgende Gruppen gebildet: Der Mittelwert aus der Beantwortung der beiden Fragen 02A und 03A für die erste Gruppe sollte im Durchschnitt nicht mehr als zwei sein. In der ersten Gruppe befinden sich somit ausschließlich Teilnehmer die angekreuzt haben, dass sie entweder gar nicht oder unter zwei Stunden in der Woche vor einem Computer oder einer Konsole verbringen. Im Minimalfall der ersten Gruppe beschäftigt sich der Teilnehmer gar nicht mit einem Computer oder einer Konsole im Maximalfall unter 4 Stunden in der Woche. Im Gegensatz dazu werden in die zweite Gruppe nur Teilnehmer aufgenommen, die im Schnitt mehr als 3,5 angekreuzt haben, d.h. insgesamt mindestens 10 bis über 20 Stunden in der Woche vor einem der beiden Geräten verbringen. Eine ANOVA über alle Antwortmöglichkeiten der beiden Gruppen ergab keinen signifikanten Unterschied ($p > .058$).

Affinität zu digitalen Spielen

Eine letzte Überprüfung auf bestimmte Gruppen wurde mit der der Beantwortung der Frage *04A - Ich mag Computerspiele* ermittelt. Die Teilnehmer werden folgendermaßen eingeteilt:

- Gruppe 1: Die Frage 04A wurde mit *stimmt* oder *stimmt genau* beantwortet.
- Gruppe 2: Die Frage 04A wurde mit *stimmt überhaupt nicht* oder *stimmt nicht* beantwortet.

Die Teilnehmer die diese Frage mit *neutral* beantwortet haben, werden hier nicht berücksichtigt, da die Frage ermittelt werden soll, ob bei einer Affinität bzw. Abneigung gegenüber Computerspielen, das Experiment unterschiedlich aufgenommen worden ist. Eine neutrale Position ist an dieser Stelle nicht aussagekräftig genug. Ein ANOVA-Test über alle Antworten der beiden Gruppen hat einen signifikanten Unterschied ergeben ($p < .03$). Aufgrund des ermittelten Unterschiedes, werden im Folgenden die Antworten der beiden Gruppen vorgestellt.

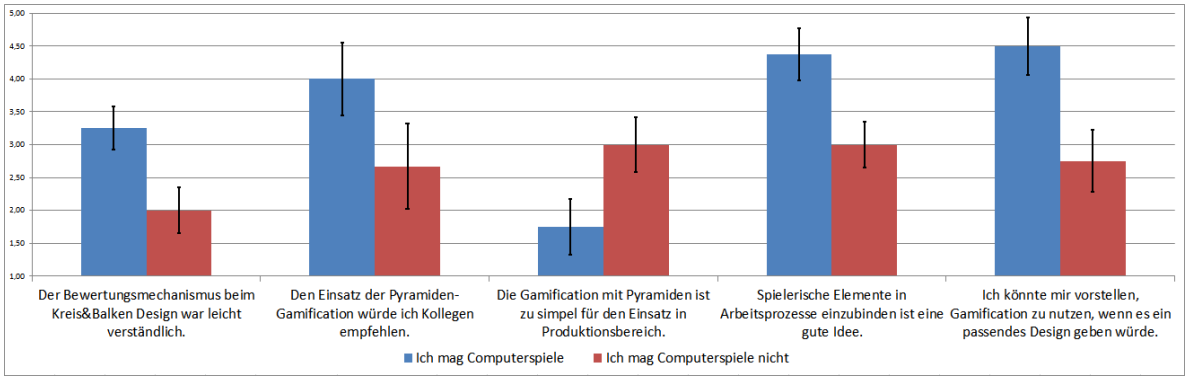


Abbildung 4.1.: Gruppenvergleich: Affinität zu Computerspielen. Der blaue Balken zeigt den Durchschnitt der Antworten der Gruppe 1 an, der rote Balken die Antworten der Gruppe 2. Die Y-Achse zeigt die verwendete Likert-Skala an.

Über die gesamte Befragung haben die Teilnehmer der Gruppe 1 (höhere Affinität zu Computerspielen), höhere Bereitschaft und Zustimmung gegenüber den Fragen nach der Gamification gezeigt. Eine Ausnahme bilden hier die Fragen nach der Simplizität und der Ablenkung der Implementierungen (21T - 26P), ein niedriger Wert zeugt davon, dass sich die Befragten durch die Implementierung nicht gestört fühlen. Die Abbildung zeigt nur die Fragen, die mit einem deutlichen Unterschied beantwortet wurden, allerdings ist dieses Muster, wenn auch in abgeschwächter Form, über den gesamten Fragenkatalog erkennbar. In Bezug auf die Fragen nach der visuellen Akzeptanz unterscheiden die beiden Gruppen sich nicht, beide geben im Schnitt dieselbe Antwort ($\overline{x}$ = 3,17). Besonders stark unterscheiden sich die Antworten der beiden Gruppen in den abgebildeten Antworten, siehe Abb. 4.1. Der Mittelwert der Antworten der ersten Gruppe beträgt 3,2 (SD: 0,78), während der Mittelwert der zweiten Gruppen 2,8 (SD: 0,47) beträgt.

Eine weitere Hypothese wird aufgestellt: *Es existiert eine Korrelation zwischen dem Alter der Probanden und deren Affinität zu Computerspielen.* Die Hypothese, je älter der Mensch, desto weniger Affinität zu Computerspielen, wird untersucht. Die Untersuchung der Korrelation des Alters und die Antwort auf die Frage *Ich mag Computerspiele*, hat r = 0.25 ergeben. Falls r = 1 gilt ein perfekter Zusammenhang, bei r < 0.19 gilt hingegen kein Zusammenhang. Das Ergebnis beträgt r = 0,25 somit ist nur ein schwacher Zusammenhang gegeben [Bor05].

4.2.3. Diskussion der Gruppenergebnisse

Nachdem vier unterschiedliche Hypothesen geprüft worden sind, konnten in drei der vier Fälle keine Unterschiede festgestellt werden (U4). Die Hypothese, dass die jüngeren Teilnehmer positiver auf die Fragen antworten konnte nicht bestätigt werden. Das Alter spielt offenbar bei der Bewertung von Gamification keine Rolle. Die Tatsache ob ein Teilnehmer Personaltrainer oder Montagemitarbeiter von Beruf ist hat keinen Einfluss auf die gemachten Antworten. Die Hypothese, dass Montagemitarbeiter positiver als die Personaltrainer auf das Experiment reagieren konnte nicht bestätigt werden. Die

verbrachte Zeit vor dem Computer hat nach Auswertung der Ergebnisse ebenso keinen Einfluss auf die Reaktion gegenüber dem Experiment. Einzig die Hypothese, dass Teilnehmer die Computerspiele positiv bewerten auch positive Rückmeldungen zum Experiment abgeben würden, konnte bestätigt werden.

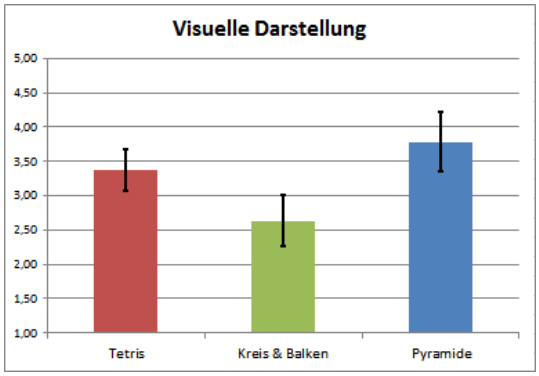
Vor allem die Ergebnisse der Altersuntersuchung, dass die jüngeren Teilnehmer keine höhere Offenheit und Affinität gegenüber dem Experiment zeigen ist überraschend.

Im nächsten Abschnitt wird die Meinung aller Teilnehmer gegenüber den drei Konzepten geprüft. Die Konzepte werden nicht innerhalb bestimmter Gruppen untersucht, da die Akzeptanz aller Teilnehmer gefragt ist.

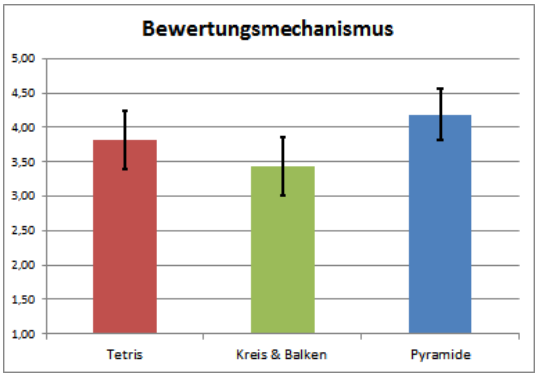
4.2.4. Untersuchung der Konzeptergebnisse

Die Frage welches Konzept in allen Disziplinen am besten abgeschnitten hat, wird in den folgenden Abschnitten geklärt. Die Konzepte werden in insgesamt drei Themen bewertet. Die Bewertungen beziehen sich auf die visuelle Gestaltung, die Spielmechanik und die allgemeine Akzeptanz. Die Fragen in den jeweiligen Themengebieten werden jeweils pro Konzept gestellt. Somit werden zu jedem Konzept dieselben Fragen gestellt, woraus eine abschließende Bewertung entsteht. Die Fragen werden in derselben Reihenfolge gestellt, in welcher sie auf dem Fragebogen zu finden sind.

In den nachfolgenden Abbildungen wird das Ergebnis einer Frage zu den drei Konzepten dargestellt. Die Y-Achse der Diagramme beschreibt die verwendete Likert-Skala. Die Y-Achse beginnt immer bei 1, die entsprechende Antwortmöglichkeit *stimmt überhaupt nicht* und endet bei 5, der Antwortmöglichkeit *stimmt genau*. In den Balkendiagrammen übernimmt der rote Balken die Anzeige für das Ergebnis der Tetris-Implementierung, der grüne Balken steht für die Kreis & Balken-Implementierung und der blaue Balken gibt das Ergebnis der Pyramiden-Implementierung wieder. Die Standardabweichung wird über die Markierung am oberen Ende des Balkens dargestellt.



(a) Die visuelle Darstellung beim ...-Design hat mir gefallen

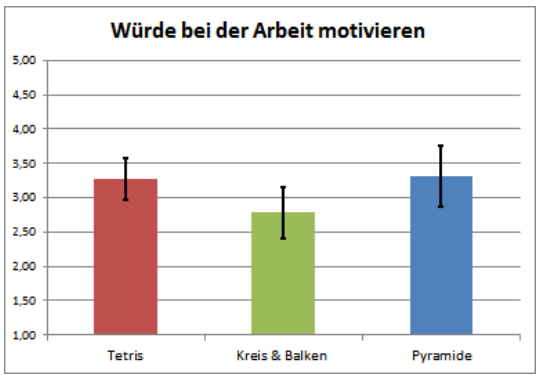


(b) Der Bewertungsmechanismus beim ...-Design war leicht verständlich

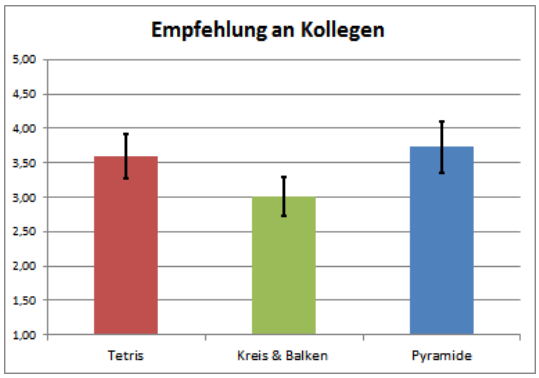
Abbildung 4.2.: Ergebnisse der Studie Teil I. Visuelle Darstellung und Bewertungsmechanismus.

Die ersten Fragen 07T - 09P (Abb. 4.2.a), beziehen sich auf die Akzeptanz der visuellen Darstellung der Implementierungen. Die ANOVA über die Akzeptanz der drei Darstellungen hat einen stark signifikanten Unterschied ergeben ($p < .0001$). Bei der Pyramiden-Implementierung ($\overline{x} = 3{,}8$) ist im Gegensatz zu Tetris ($\overline{x} = 3{,}15$) und Kreis & Balken ($\overline{x} = 2{,}42$) die Akzeptanz am höchsten.

Die Fragen 12T - 14P (Abb. 4.2.b) wurden einer ANOVA unterzogen. Es wurde ein stark signifikanter Unterschied festgestellt ($p < .003$). Die Akzeptanz der Pyramide ($\overline{x} = 4{,}15$) ist im Gegensatz zu Tetris ($\overline{x} = 3{,}55$) und Kreis & Balken ($\overline{x} = 3{,}11$), am höchsten.



(a) Eine Gamification mit dem ...-Design würde mich beim Arbeiten motivieren

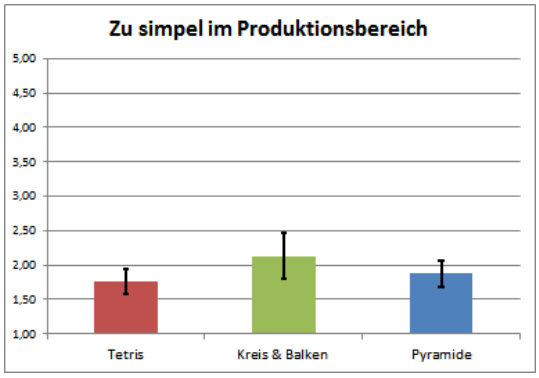


(b) Den Einsatz der ... Gamification würde ich Kollegen empfehlen

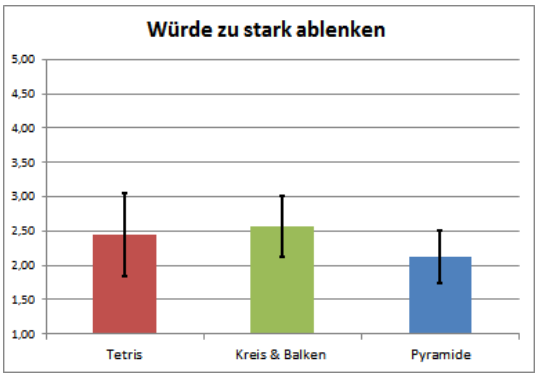
Abbildung 4.3.: Ergebnisse Studie Teil II. Mittelwert in der Likert-Skala

Die Fragen 15T - 17P, ob Gamification einen Mitarbeiter bei der Arbeit motivieren würde (Abb. 4.3.a) wurden mithilfe einer ANOVA ausgewertet. Es wurden keine signifikanten Unterschiede festgestellt.

Nach einer ANOVA über die Fragen 18T - 20P (Abb. 4.3.b) wurde ein stark signifikanter Unterschied festgestellt ($p < .03$). Das Pyramidenkonzept erhält mit ($\overline{x} = 3{,}42$) im Gegensatz zu Tetris ($\overline{x} = 3{,}21$) und Kreis & Balken ($\overline{x} = 2{,}58$) ein weiteres Mal die beste Bewertung.



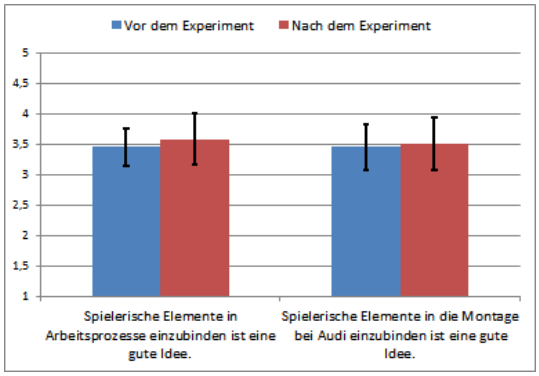
(a) Die Gamification mit ... ist zu simpel für den Einsatz in Produktionsbereich



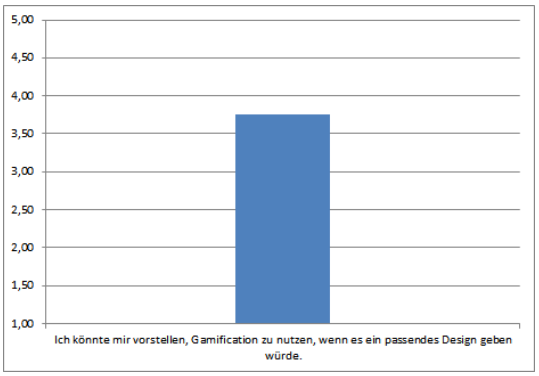
(b) Die Gamification mit ... würde mich zu stark von der Arbeit ablenken

Abbildung 4.4.: Ergebnisse Studie Teil III

Es scheint so, dass in den letzten beiden Themengebieten die Kreis & Balken-Implementierung präferiert wird, jedoch wird das Bewertungsschema unterbrochen: Da die Frage negativ gestellt ist, wird ein hoher Wert als Ablehnung der Implementierung gewertet. Die beiden letzten Themen haben gemeinsam, dass mithilfe einer ANOVA kein signifikanter Unterschied in den Antworten festgestellt worden ist.



(a) Einstellung vor und nach dem Experiment



(b) Allgemeine Haltung gegenüber Gamification

Abbildung 4.5.: Ergebnisse Studie Teil IV

Ein Fragepaar wurde einmal vor und einmal nach dem Experiment gestellt. Die beiden Fragen 05A und 06A wiederholen sich in den Fragen 27A und 28A (Abb. 4.5.a). Eine ANOVA konnte keinen Unterschied feststellen.

Die letzte Frage 29A wurde mit dem Durchschnitt ($\overline{x}$ = 3,75, SD: 1,26) beantwortet. Die Präsentation der Ergebnisse ist abschlossen. Im nächsten Abschnitt werden die erhaltenen Ergebnisse diskutiert.

4.2.5. Diskussion der Konzeptergebnisse

Die visuelle Präferenz und die Verständlichkeit des Bewertungsmechanismus liegt eindeutig auf Seiten der Pyramiden-Implementierung (U4). Der Tetris-Ansatz erhält auf den beiden Gebieten durchaus Zustimmung, wohingegen Kreis & Balken weit abgeschlagen auf dem dritten Platz liegt.

Die Probanden haben sich nicht eindeutig für die Pyramide der Tetris in Bezug auf die Motivationsfähigkeit der Konzepte, entschieden. Eindeutig geht hier jedoch hervor, dass Kreis & Balken nicht als besonders ansprechend empfunden worden ist. In diesem Teil wurden die Implementierung am kritischsten begutachtet, d.h. die Zustimmungsrate ist im Schnitt am niedrigsten. Die Einsicht in die qualitativen Antworten gibt einen interessanten Einblick, weshalb dies so sein könnte. An dieser Stelle wurde vor allem ein mögliches Problem mit der Langzeitmotivation gesehen. Die Sorgen sind besonders deutlich, in der Gruppe der Montagemitarbeiter erkennbar, im Schnitt erhielten die Implementierungen nur eine Bewertung von $\overline{x}$ = 2,5, die Bewertung der Personaltrainer hingegen ist mit $\overline{x}$ = 3,75 deutlich höher.

Ein weiteres Mal wird die Pyramide gegenüber den beiden anderen Implementierungen, in Bezug auf die Empfehlung an die Kollegen, vorgezogen. Die Beantwortung der Frage nach der Empfehlung an die Kollegen, wird abermals als stark signifikant gewertet. Was somit eine Präferenz erahnen lässt.

Durch die Ergebnisse zur Simplizität kann darauf geschlossen werden, dass die Probanden die Implementierungen allgemein als nicht zu simpel oder als zu ablenkend empfinden. Eine Besonderheit wird in deutlich, die Standardabweichung ist im Gegensatz zu den vorigen Fragen, in welchen die Abweichung nicht über ein durchschnittliches Maß hinausgeht, deutlich höher. Die Teilnehmer waren an dieser Stelle, statistisch gesehen, besonders uneins. Das kann daran liegen, dass Ablenkungen sehr stark individuell gewertet werden.

Die letzten beiden Fragen deuten darauf hin, dass eine allgemeine Bereitschaft der Belegschaft, in Bezug auf die Einführung von Gamification, vorhanden ist. Das ist insofern wichtig und interessant, da es erkennen lässt, dass Gamification von nicht leistungsgeminderten Personen positiv aufgenommen wird.

Nachdem alle quantitativen Fragen vorgestellt wurden, sind einige Tendenzen zu erkennen. Das Pyramiden-Konzept hat insgesamt in allen Disziplinen am positivsten abgeschnitten, mal mehr, mal weniger dicht gefolgt vom Tetris-Konzept. Etwas abgeschlagen, mit teils nicht überzeugenden Ergebnissen, besonders in der visuellen Darstellung (Abb. 4.2.a), bildet das Kreis & Balken-Konzept das Schlusslicht. Es konnte eine insgesamt positive Resonanz auf Gamification in der Produktion ermittelt werden.

4.3. Qualitativer Ergebnisteil

Durch die folgende Besprechung der offenen Fragen (qualitativer Teil) kommen noch einige Aspekte zu den Ergebnissen des quantitativen Teils hinzu. Die Fragen werden entweder gekürzt zusammengefasst wiedergegeben oder nach Möglichkeit wird ein Original Zitat verwendet. Eine Visualisierung der Antworten bietet sich an dieser Stelle nicht an (U4).

Im Gegensatz zu den quantitativen Fragen, wo kein Unterschied der beiden Gruppen Personaltrainer und Montagemitarbeiter festgestellt wurde, lässt sich in den offenen Antworten eine gewisse Tendenz herauslesen.

Über alle offenen Fragen hinweg, lassen die Personaltrainer eine sehr positive und aufgeschlossene Grundstimmung gegenüber der Gamificationidee, in ihren Antworten erkennen. Es werden viele Vorschläge zur Weiterentwicklung der Gamification gemacht. Ein Studienteilnehmer schreibt, dass er sich Belohnungen in der Realität für eine besonders erfolgreiche Individualleistung wünschen würde. Gehaltsaufschläge oder Ermäßigungen bei Firmeneigenen Mitfahrzeugen, wurden als Beispiele genannt, ein ähnlicher Vorschlag wurde von einem anderen Trainer, sofort als *großes Motivationsproblem* gedeutet. Was passiert mit der Motivation des Mitarbeiters, wenn *alle Kollegen um ihn herum mehr Geld bekommen* aber er selbst nicht?

Ein weiterer Studienteilnehmer äußerte die einmalige Idee, dass die Einbeziehung von technischen Hilfsmitteln bei der Fertigung mitberücksichtigt werden sollte. Bei AUDI in der Produktion gibt es verschiedene Hilfsmittel für die Mitarbeiter, die eine körperschonende Arbeitsweise unterstützen. Diese Hilfsmittel werden, aber nach Meinung des Mitarbeiter zu selten genutzt, da ihre Benutzung oft zeitaufwendiger ist, als eine kurze Verdrehung oder Mehrbelastung des Körpers. Dadurch kommt es, laut Mitarbeiteraussage, in manchen Fertigungsbereichen zu hohen Ausfallquoten aufgrund von Bandscheibenbeschwerden. Es wäre wünschenswert, wenn die Mitarbeiter diese Hilfsmittel öfter benutzen würden. Dieser Punkt könnte seiner Meinung nach in eine neue Gamification implementiert werden.

Die Frage nach der grafischen Gestaltung wurde stark diskutiert. Für einen Mitarbeiter, war es absolut undenkbar die aktuellen Implementierungen, aus Gründen der Ästhetik, zu nutzen. *Ein Autobauer im Premiumsegment, muss auch intern Premiumqualität bieten, und das vor allem in Bezug auf die grafischen Qualitäten*, war die Aussage. Es wurden allerdings auch von Studienteilnehmer geäußert, die mit der aktuellen Grafik zufrieden waren und meinten, dass durch die einfache Grafik Ablenkungen vermieden werden.

Uneinig waren sich speziell die Personaltrainer in der Veröffentlichung der Einzelergebnisse und der Belohnung bzw. Bestrafung der Benutzer. Ein Trainer empfand schon die Ausblendung des Pokals bei einem Fehler als zu *hohe Fallhöhe* und vermutete eine hohe Wahrscheinlichkeit der Demotivation dahinter, wohingegen ein anderer die Aussage tätigte: *Die Ausblendung (des Pokals) ist sehr gut.* Er meinte, dass die Einblendung einer Bestenliste aller Mitarbeiter den Wettbewerbscharakter stärken würde und somit die Motivation besser als die anderen zu sein steigern würde. Die Idee, dass neue Mitarbeiter ihre Fertigungszeiten spielerisch mit Bestzeiten vergleichen könnten, um einzuschätzen wie schnell sie sind, wurde insgesamt als positiv erachtet.

In einem Punkt haben ausnahmslos alle Personaltrainer starke Bedenken geäußert es dürfen keine Einzelleistungen der Mitarbeiter aufgezeichnet werden und den Führungskräften zugänglich gemacht werden. Der Punkt der Anonymität im regulären Betrieb sei dem Betriebsrat außerordentlich wichtig. Falls sich eine Gamification in ebendiese Richtung entwickeln sollte, ist eine praktische Realisierung undenkbar. Dieser Punkt hat weitreichende Folgen. Die Gamification muss Einzelleistungen aufzeichnen und bewerten können, jedoch ohne ein Archiv zu erstellen. Eine von vielen gewünschte Highscoreliste, ist somit schon nur aus Fragen der Anonymität undenkbar es wäre zwar möglich

Benutzernamen anstatt den echten Namen zu verwenden, allerdings besteht dann immer die Gefahr der Enttarnung, was ein zusätzlicher Stressfaktor ungeahnten Ausmaßes erschaffen könnte.

In der Gruppe der Montagemitarbeiter war die Sorge um die Meinung des Betriebsrats zu Gamification nicht so präsent wie ind er Gruppe der Personaltrainer. Insgesamt waren die Antworten etwas zurückhaltender und vorsichtiger formuliert. Einige der Mitarbeiter gaben mündlich an, bei einem Vorprojekt beteiligt gewesen zu sein, in der die vorgestellten Implementierungen über einen längeren Zeitraum (mehrere Tage) getestet wurden. Aus Gründen der Wahrung der Anonymität ist es leider nicht möglich, die Antworten zurückzuverfolgen. Es kann sein, dass sich ebendiese Mitarbeiter besonders deutlich mit der Frage nach der Langzeitmotivation auseinander gesetzt haben, da Antworten in dieser Richtung oft geäußert wurden. *Was passiert nach ein oder zwei Monaten (mit den Visualisierungen), die werden doch langweilig*, war eine derartige Aussage. Da stellt sich natürlich die Frage, wie eine Implementierung auch noch nach Monaten neue Anreize und Motivation schaffen kann.

Viele Mitarbeiter hatten die Sorge durch die Gamification überfordert oder zusätzlich gestresst zu werden. Sei es dadurch, dass sie die Gamification nicht verstehen werden oder, dass sie durch eine unverständliche Visualisierung verwirrt werden würden.

Hinzu kommt die Frage nach der Sichtbarkeit der Gamification. In den Produktionshallen wird großer Wert auf eine ausreichende Beleuchtung gelegt was passiert aber im Falle, dass eine nicht optimale Ausleuchtung möglich ist, sei es durch Überbelichtung oder Unterbelichtung des Teils des Arbeitsplatzes an dem die Visualisierung dargestellt wird. Laut einigen Mitarbeitern sollte eine Sicherstellung einer optimalen Ausleuchtung eine hohe Priorität zugewiesen bekommen.

Eine weitere Äußerung an der erkennbar wird, dass der Mitarbeiter mit den Implementierungen womöglich schon einmal in Kontakt gekommen ist, war der Wunsch nach der Möglichkeit verschiedene Lösungspfade bei der Fertigung nutzen zu können. Momentan darf ein Mitarbeiter nur einen einzigen Fertigungsablauf abarbeiten, jedoch gibt es viele Mitarbeiter die nach Jahren eigener Erfahrung, entweder die Reihenfolge des Einbaus von Einzelteilen variieren oder, was sehr häufig vorkommt, einzelne Schrauben immer wieder in unterschiedlichen Reihenfolgen montieren. Sobald nicht die Schraube als erstes montiert wird, die im Fertigungsablauf eingespeichert, gibt das System einen Fehler aus. Dieses Problem kann frustrierend wirken, da kein wirklicher Fehler begangen wurde.

Die gewonnenen Erkenntnisse zusammen mit der technischen Realisierbarkeit werden im Kapitel Erweitertes Gamification-Konzept abschließend verarbeitet.

5. Erweitertes Gamification-Konzept

Das Konzept der zukünftigen Implementierung wird auf Grundlage der Studienergebnisse und den technischen Möglichkeiten von motionEAP erarbeitet. Der Vorschlag, der in den offenen Fragen von einem Teilnehmer gemacht wurde, die Benutzung von Hilfsmitteln für ein körperschonendes Arbeiten, in die Gamification mit einfließen zu lassen, scheitert an den technischen Grenzen von motionEAP. Das motionEAP-System verarbeitet ausschließlich Bewegungen über der Arbeitsfläche. Da die Hilfsmittel größtenteils dort zum Einsatz kommen, wo Mitarbeiter an einer nicht fest installierten Arbeitsfläche tätig werden, ist das ein anderer Anwendungsfall. Es ist in motionEAP nicht möglich den gesamten Körper des Mitarbeiters aufzunehmen und die Haltung zu bewerten. Dieser Punkt birgt Anknüpfungspunkte für ein etwaiges Nachfolgeprojekt.

Viele der Antworten, in welchen es um Sorgen und Ängste bezüglich eines erhöhten Stressfaktors oder erhöhter Ablenkung durch die Gamification geht, sind schon in die Vorprojekte mit eingeflossen (U5). Die Gamification ist beispielsweise nur indirekt steuerbar, es müssen also keine zusätzlichen Bewegungen gemacht werden, um das System zu steuern. Auch ist der Projektionsort, die Arbeitsfläche, so gewählt, dass der Mitarbeiter ohne Mehraufwand die Gamification-Anwendung einsehen kann. Die Bewertungsmechanismen, die Darstellung und die Animationen wurden so gewählt, dass eine Ablenkung vermieden wird. Dieser Punkt ist sehr wichtig und hier sollte nicht nach einer ungefähren Mehrheit entschieden werden, sondern danach, ob es Individuen gibt, die mit der Visualisierung überfordert sind. Denn nur so, kann sichergestellt werden, dass eine weite Bandbreite an Mitarbeiter die Lösung nutzen kann. In der Studie gab es einige kritische Bemerkungen zu diesem Thema, allerdings hat kein Teilnehmer eine völlige Überforderung geäußert. Da an einigen Stellen in den offenen Fragen Sorgen zu einer möglichen Überforderung durch Gamification geäußert wurden, sollte die Möglichkeit bereitgestellt werden, die Visualisierung auszublenden.

Ein eindeutiges Problem mit der visuellen Darstellung der aktuellen Gamification-Konzepten ist identifiziert worden.

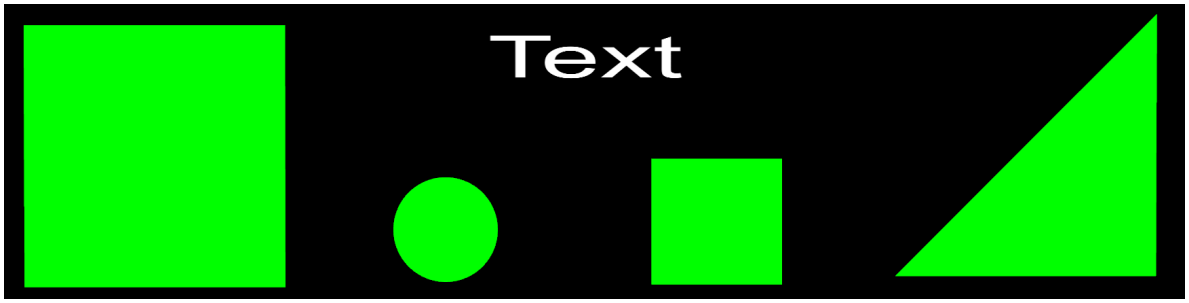


Abbildung 5.1.: Visuelle Darstellung von motionEAP

Die aktuelle visuelle Darstellung im motionEAP-System beschränkt sich auf sehr einfache zweidimensionale Strukturen wie Kreise, Dreiecke und Vierecke. Es ist tatsächlich nicht möglich Formen die mehr als vier Ecken besitzen anzeigen zu lassen (Abb. 5.1). Es fällt auf, dass das System primär nicht für *ästhetische* visuelle Anwendungsfälle geschrieben wurde.

Zusätzlich ist es nicht möglich flüssig wirkende Verläufe zu erstellen. Es kann nur eine Abfolge von mühsam erstellten Momentaufnahmen ausgegeben werden. Ein *Farbverlauf* verkommt dabei zu einer Abfolge von wenigen Farben. Um einen Farbverlauf zu erhalten, muss zuvor eine Reihenfolge von exakt definierten Farben angelegt werden. Das ist nicht nur zeitaufwendig und umständlich für den Entwickler, sondern wird zudem auch nicht als visuell angenehm fürs Auge wahrgenommen. Die Unmöglichkeit einen flüssigen Ablauf zu Erhalten macht sich nicht nur an einem farblichen Verlauf bemerkbar, sondern auch an einem Bewegungsablauf. Ein Element kann nicht *flüssig* bewegt werden, sondern der Entwickler ist gezwungen, das Element an einer Stelle auszublenden und an einer anderen Stelle wieder einzublenden. Im Falle einer Spielfigur wäre eine flüssige Bewegung nicht nur sinnvoll, sondern auch visuell ansprechend (siehe dazu die Ergebnisse *visuelle Darstellung* im Kapitel Akzeptanz-Studie). Die beschränkten visuellen Darstellungsmöglichkeiten haben eine Verringerung der Akzeptanz gegenüber dem gesamten Gamification-Konzept zur Folge.

Die Erkenntnis ist gereift, dass der Wunsch nach einer Verbesserung der visuellen Darstellung von motionEAP nicht befriedigt werden kann. MotionEAP ist mit der Darstellung der vorhandenen Gamification-Konzepte an die Grenzen der technischen Möglichkeiten gelangt. Um Nachfolgeprojekten eine geeignete Grundlage für die Weiterentwicklung der Gamification-Implementierungen zu schaffen, ist der Entschluss gefasst worden, dass ein neues System für die visuelle Darstellung benötigt wird. MotionEAP soll als Gesamtsystem für die Bewegungserkennung und Projektion erhalten bleiben, lediglich die visuelle Darstellung der Gamification-Komponente soll ein Zweitsystem übernehmen. Somit sollen die Möglichkeiten der Darstellungsweise der Gamification-Konzepte in motionEAP verbessert werden.

Von den drei vorhandenen Konzepten hat sich die Pyramide eindeutig als bevorzugte Darstellungsweise herausgestellt. Nachdem ein neues Darstellungssystem implementiert worden ist, soll das Pyramiden-Konzept in dem neuen System verwendet werden.

5.1. Gamification Modul

Aufgrund der eben genannten Bedingungen, wird ein neues System mit folgenden Eigenschaften gesucht (U6). Das neue System sollte mit der Programmiersprache C# umgehen können. MotionEAP ist in C# geschrieben. Die Einführung einer weiteren Programmiersprache eignet sich aus Gründen der Wartbarkeit und Weiterentwicklung nicht. Um geeignete Fachkräfte für ein etwaiges Folgeprojekt oder für kleinere Weiterentwicklungen zu finden, ist eine einheitliche Programmiersprache der beiden Anwendungen zu bevorzugen. Da die Menge der C# Entwickler immer gleich groß oder größer sein wird, als die Menge der C# Entwickler die zusätzlich noch die Programmiersprache des Zweitsystems beherrschen.

Die neue Mechanik sollte eine deutliche überlegene Möglichkeit der grafischen Visualisierung besitzen, als das derzeitige System (U6). Es ist nicht notwendig aktuelle grafische Topleistungen zu erbringen. Ein solides System, welches dreidimensionale Grafiken in angemessener Form zum Ausdruck bringt genügt dabei.

Das neue Programm sollte nach Möglichkeit kostenlos sein oder eine kostenlose Lizenz für Studenten enthalten. Im Falle einer Weiterentwicklung müssten komplizierte Lizenzfragen erörtert werden, die zusätzlich kostspielig werden könnten. Das ist im Rahmen einer studentischen Abschlussarbeit nicht zumutbar.

Das Zweitsystem soll über motionEAP gesteuert werden können. Das neue System muss beispielsweise Informationen über den Bewegungsablauf des Benutzers erhalten. Für diese Art der Kommunikation, zwischen motionEAP und dem Zweitsystem, ist die Kommunikation über UDP geeignet. Die Möglichkeit dieser Übertragung sollte im neuen System reibungslos ablaufen können. UDP (User Datagram Protocol) ermöglicht Anwendungen den Versand von Informationen in einem IP-basierten Netzwerk [Pos80].

Es wurden somit folgende Punkte für ein Zweitsystem identifiziert:

- Bevorzugte Programmiersprache ist C#
- Verbesserte visuelle Darstellungsmöglichkeit
- Kostenlos
- Unterstützung der UDP-Kommunikation

Nachdem die wichtigsten Punkte identifiziert wurden, kann nach einem geeigneten Zweitsystem gesucht werden. Ein aktuelles System, das die Merkmale erfüllen könnte ist die Unity-Engine.

Die Unity-(Spiel)-Engine ist eine plattformübergreifende Entwicklungsumgebung für digitale Spiele und wird derzeit auch stark für Anwendungen in den Themengebieten der Virtuellen- und Erweiterten-Realität genutzt. Unity wurde im Juni 2005 veröffentlich und seitdem wurde es kontinuierlich weiterentwickelt. Die aktuellste Version ist von Januar 2016 und der letzte kleinere Patch wurde im April 2016 veröffentlicht [Haa14].

Unity kann u.a. mit den beiden Programmiersprachen C# und Unityscript (Abwandlung von JavaScript) umgehen. Die visuelle Darstellungsmöglichkeit von Unity ist dem aktuellen System weit überlegen.

Da es in der aktuellen Spielentwicklung genutzt wird, bietet es in dieser Hinsicht für dieses Projekt, praktisch keinerlei Grenzen. Neben den sehr guten grafischen Voraussetzungen und der passenden Programmiersprache lassen sich auch Animationen erstellen und Musik bzw. Töne verarbeiten.

Durch die Unterstützung der Programmiersprache C# wird die geforderte UDP-Schnittstellen Unterstützung gewährleistet. Unity ist für private und nicht-kommerzielle Zwecke oder zur Weiterbildung im schulischen und universitären Umfeld vollkommen kostenlos. Alle an dieser Bachelorarbeit beteiligten Personen haben durchweg positive Erfahrungen mit Unity gemacht. Das ist nicht nur ein überzeugender Punkt für Unity, sondern somit würde eine Einarbeitungsphase auch größtenteils wegfallen.

Unity ist ein aktuelles und fest auf dem Markt verankertes Produkt. Dies äußert sich nicht zuletzt durch eine große, aktive und zuvorkommende Community. Sehr viele Fragen und Problemstellungen wurden bereits erfolgreich beantwortet. Auch für neue Fragen sind die Foren groß genug, dass innerhalb von maximal zwei Tagen eine (in den meisten Fällen) weiterbringende Antwort zur Verfügung steht. Dies darf als ein weiterer Pluspunkt für Unity gewertet werden.

Da die Unity-Engine in allen Punkten voll überzeugen kann, fällt die Entscheidung auf Unity. Es wird somit zum neuen Zweitsystem für die visuelle Darstellung in motionEAP.

5.2. Technische Details von motionEAP

Im Folgenden wird erst das vorhandene motionEAP-System beschrieben, daraufhin wird das Konzept der Unity-Engine vorgestellt und abschließend wird die Zusammenarbeit der beiden erläutert.

In diesem Kapitel werden die technischen Details von motionEAP beleuchtet, eine Beschreibung des System ist im Kapitel Stand der Technik enthalten. MotionEAP ermöglicht es mittels einer Bewegungssensorik, manuelle Fertigungsabläufe zu verfolgen und mit einem Projektor gewünschte Bewegungsabläufe darzustellen.

5.2.1. Beschreibung der motionEAP Geräte

Um das motionEAP-System nutzen zu können ist ein Projektor, eine Xbox-Kinect Kamera und ein Computer notwendig. Die Software läuft auf dem Computer und erhält Informationen von der Xbox-Kinect Kamera, verarbeitet diese und projiziert die Ergebnisse, per Projektor, auf eine Arbeitsfläche.

Die Arbeitsfläche ist eine Tischplatte über welcher die Xbox-Kinect und der Projektor montiert sind. Die Kamera erfasst die Bewegungen, die auf der Arbeitsfläche getätigt werden. Die Xbox-Kinect ist eine Kamera die Bewegungen und Veränderungen, in einer Entfernung von 2m, auf der X- und Y-Achse (Höhe, Breite)auf 3mm und der Z-Achse (Tiefe) auf etwa 1 cm genau, erkennen kann [Mic16a].

MotionEAP kann über den angeschlossenen Projektor verschiedene Bereiche auf der Arbeitsfläche einfärben. Die Größe des Bereiches der Projektion ist etwa 1,5 m lang und 1,5 m breit. Die Größe deckt sich mit dem Bereich der Bewegungserkennung.

5.2.2. Steuerungsbefehle von motionEAP an die Unity-Anwendung

Die Fertigung eines Bauteils wird in einzelne Arbeitsschritte unterteilt. Ein Arbeitsschritt kann dabei eine Bewegung oder das Ablegen eines Bauteils in einen Bereich sein. Nachdem die für den Arbeitsschritt notwendige Bewegung erkannt wurde, ist der Schritt abgeschlossen. Daraufhin beginnt der nächste Arbeitsschritt. Falls alle nötigen Bewegungen bzw. Arbeitsschritte für ein Bauteil erfolgt sind. Beginnt die Fertigung des neuen Bauteils beim ersten Arbeitsschritt.

Um erkennen zu können wann ein Bauteil abgeschlossen wird, sind die folgenden Informationen erforderlich:

- Gesamtanzahl der Arbeitsschritte
- Fertigungszeiten der einzelnen Arbeitsschritte
- Start- und Endzeitpunkt eines Arbeitsschrittes
- Aktueller Arbeitsschritt
- Fehlerausgabe (hier: die Schrittreihenfolge wurde fehlerhaft durchgeführt)
- Vergangene Schrittzeiten

Eine Gesamtanzahl der Arbeitsschritte ist notwendig um zu erkennen, ab welchem Arbeitsschritt ein Bauteil abgeschlossen ist. Die Fertigungszeiten werden derzeit aus einer Kalibrierungsphase, die jeder Benutzer individuell durchführt, ermittelt. Die Durchschnittszeit der ersten drei, vollständig gefertigten, Bauteile wird als Fertigungszeit genutzt. Der Start- und Endzeitpunkt eines Arbeitsschrittes, gibt an ob sich die Geschwindigkeit der Fertigung innerhalb der Fertigungszeiten befindet. Die Information über den aktuellen Arbeitsschritt ist notwendig um den Fortschritt festzuhalten.

Die vorhandenen Gamification-Implementierungen erhalten intern die angeführten Informationen. Damit ein Zweitsystem die Aufgabe der Visualisierung der Gamification-Implementierungen übernehmen kann, ist es notwendig die erwähnten Informationen über die UDP-Schnittstelle zu erhalten. Da bereits ein laufendes Gamification-System existiert, ist es nicht nötig die erwähnten Informationen in einem neuen Verfahren zu ermitteln. Für das neue Zweitsystem müssen die Informationen, statt intern übergeben zu werden, über die UDP-Schnittstelle übermittelt werden.

5.2.3. Gamification Steuerungsklasse in motionEAP

In motionEAP übernimmt die Steuerung der vorhandenen Gamification-Implementierungen hauptsächlich eine Klasse. Die sich in *motionEAPAdmin\Scene* befindliche *GamificationKo* Klasse, verarbeitet die oben erwähnten Informationen und generiert daraus die einzelnen Gamification-Konzepte.

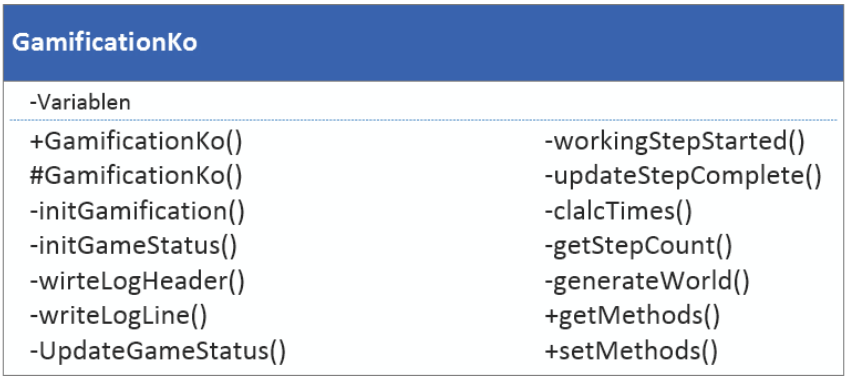


Abbildung 5.2.: GamificationKo Klassendiagramm

Das abgebildete Klassendiagramm der GamificationKo-Klasse (Abb. 5.2), zeigt alle Methoden, die für die Steuerung und Generierung der Gamification benötigt werden. Die Variablen werden in der Abbildung nicht einzeln notiert, da die Übersichtlichkeit darunter leiden würde. Die Variablen werden an den benötigten Stellen angeführt. Die ersten beiden Einträge der linken Spalte, beschreiben die Konstruktoren der Klasse. Der erste Konstruktor *+GamificationKo()* wird bei der Erstellung der Klasse auf der Arbeitsfläche aufgerufen.*-GamificationKo()* wird beim Laden eines JSON-Objektes aufgerufen. Das ist dann der Fall, wenn ein zuvor gespeicherter Workflow geladen wird, der eine Gamification enthält.

Der dritte *initGamification()* und vierte *initGameStatus()* Eintrag in der linken Spalte wird zum Starten der Gamification genutzt. Alle Starteinstellungen werden durch den Aufruf der beiden Methoden geladen und die Variablen werden initialisiert. Dadurch werden grafische Objekte erstellt und der Kalibrierungsmodus aktiviert.

Die Methoden *writeLogHeader()* und *writeLogLine()* schreiben den Verlauf der Fertigung in eine Datei. Darin wird die in Anspruch genommene Zeit der Arbeitsschritte, die Fehler und die Kalibrierungszeiten protokolliert.

Nachdem der *GameStatus* in *initGameStatus()* initialisiert wurde, wird in *UpdateGameStatus()*, ebendieser auf dem Laufenden gehalten. Dazu gehört neben dem Aufbau der Spielumgebung nach dem Abschließen der Kalibrierung, auch die Erstellung der Verlaufsanzeige nach jedem gefertigten Bauteil.

Die Übermittlung des aktuellen Arbeitsschritts wird in *workingStepStarted()* realisiert. Die Methode wird zu Beginn eines Arbeitsschrittes aufgerufen, wohingegen *updateStepCompleted()* am Ende aufgerufen wird und die benötigte Zeit speichert.

Die Kalibrierungszeiten werden in *calcTimes()* berechnet und in der erwähnten Methode *writeLogLine()* protokolliert. Der spezielle *Getter*: *getStepCount()*, enthält die Anzahl der benötigten Arbeitsschritte pro Bauteil.

Um die vorhandenen Implementierungen visuell auf der Arbeitsfläche entstehen zu lassen, wird die Methode *generateWorld()* benötigt. Mit den am Schluss angeführten *get*- und *set*-Methoden werden die Variablen gesetzt bzw. zurückgegeben.

5.2.4. Aufbau eines Steuerungsbefehls

Die vorgestellten Methoden enthalten alle benötigten Informationen zur Erstellung einer Gamification-Implementierung. Möchte man ein Zweitsystem implementieren, genügt es die vorhandenen Informationen nach außen hin zu versenden. Das Zweitsystem wird ähnlich der vorhandenen Implementierung mit den erhaltenen Informationen aufgebaut und man erhält das Pyramiden-Konzept im Zweitsystem.

Intern werden die Informationen als Variablen verarbeitet. Eine Variable besteht aus einem Namen und ihrem Wert. Damit ein Zweitsystem die Informationen unterscheiden kann, genügt es nicht, nur den Wert der Variable zu übermitteln. Damit wäre eine Unterscheidung der Informationen unmöglich. Es ist zusätzlich nötig den Namen der Variable in einer Nachricht mit zu übermitteln. Damit eine erhaltene Information im Zweitsystem nutzbar wird, ist folgender Aufbau notwendig:

[Name der Variable] + [Trennzeichen] + [Wert der Variable]

Für eine erhöhte menschliche Leserlichkeit wird ein Trennzeichen zwischen den beiden Informationen eingeführt. Somit erhält jede Nachricht denselben dreiteiligen Aufbau und wird dadurch zu einem Steuerungsbefehl. Ein möglicher Befehl, der die Gesamtanzahl der Arbeitsschritte enthält, könnte folgendermaßen lauten:

getStepCount:3

In diesem Fall wird die Variable *getStepCount* mit dem Wert *3* übermittelt. Ein solcher Befehl wird an alle Stellen gesetzt, wo sich eine Veränderung einer benötigten Variable befindet. Sobald eine Variable geändert wird, wird der entsprechende Änderung als Befehl an das Zweitsystem geschickt. Das Zweitsystem überschreibt die eigenen Variablen mit der erhaltenen Information und baut daraufhin das Konzept auf. Diese Arbeitsweise erfordert vom Zweitsystem eine Verarbeitung der Befehle. Die Erklärung der Arbeitsweise und die Aufzählung weiterer notwendiger Eigenschaften des Zweitsystems folgen im nächsten Abschnitt.

5.3. Konzept der Unity-Anwendung

Das Konzept sieht vor, dass die Unity-Anwendung, die von motionEAP geschickten Befehle empfängt und damit das Pyramiden-Konzept darstellt. Neben den im Abschnitt 5.1 angeführten grundlegenden Informationen, welche die Unity-Anwendung zur Erstellung der Pyramide benötigt, sind weitere Informationen erforderlich. Es handelt sich um die Größe und die Positionierung des Anwendungsfensters. Es müssen vier weitere Variablen, die Länge und Breite des Fensters sowie die X- und Y-Koordinate der Position übermittelt werden. Mit diesen Informationen ist es möglich, dass Fenster an der Position darzustellen, an welcher die vorhandene Implementierung dargestellt wird. Die vorhandene Implementierung wird daraufhin ausgeblendet und die Visualisierung der Gamification übernimmt

die neue Unity-Anwendung. Die Steuerung wird weiterhin vollständig über motionEAP ablaufen. Die Unity Anwendung bekommt somit keine eigenen Menüs oder Steuerungsmöglichkeiten. Die gesamte Steuerung wird über die Befehle erfolgen, die über die UDP-Schnittstelle empfangen werden.

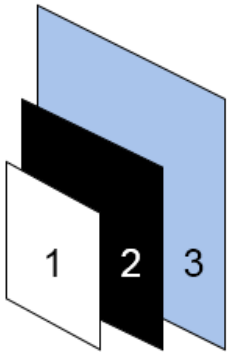


Abbildung 5.3.: Skizzierung der Überlappung der Unity-Anwendung, (1) Unity Anwendung, (2) MotionEAP Arbeitsflächen-Anwendung, (3) Desktop

Die Unity-Anwendung muss zu jeder Zeit über der motionEAP Arbeitsflächen-Anwendung ausgeführt werden (Abb. 5.3. Die vorhandene Implementierung wird in der motionEAP Arbeitsflächen-Anwendung (Abb. 5.3 (2)), über dem Desktop (Abb. 5.3 (3)), ausgeführt. Die Unity-Anwendung (Abb. 5.3 (3)) muss immer im Vordergrund laufen und darf nicht von der motionEAP Anwendung (2) überdeckt werden, denn sonst wäre sie nicht mehr zu sehen.

Die Unity-Anwendung darf keine üblichen Windowsfensterrahmen besitzen. Zusätzlich darf die Anwendung keinen eigenen Hintergrund besitzen. Außer den gewünschten Objekten, beispielsweise der Pyramide, muss alles transparent sein. Das Ausblenden der Fensterrahmen und die Transparenz des Hintergrundes würden Teile der Arbeitsfläche verdecken, was die Darstellung beeinträchtigen würde.

5.3.1. Klassendiagramm der Unity-Anwendung

Das Klassendiagramm des Konzepts der Unity-Anwendung ist in Abb. 5.4 zu sehen. Am oberen Abbildungsrand ist motionEAP dargestellt. MotionEAP schickt über die UDP-Schnittstelle Befehle an Unity. Die Befehle werden in der Unity-Anwendung in der *UDPReceive*-Methode empfangen der *UDPReceiveController*-Klasse empfangen. Die Methode übergibt die empfangenen Daten in Form einer *String*-Variable an die *Controller*-Methode weiter. In der Controller Methode wird über einen regulären Ausdruck abgefragt, welcher Befehl übermittelt wurde. Falls die übergeben *String*-Variable Zahlen enthält, werden diese hier extrahiert und als *Integer*-Variablen abgespeichert. Anschließend werden die Informationen als entsprechende *Player.Prefs* abgespeichert. Die Erklärung der Speicherung und der Funktionalität der *Player.Prefs* erfolgt im nächsten Abschnitt nachdem das Klassendiagramm vollständig beschrieben wurde.

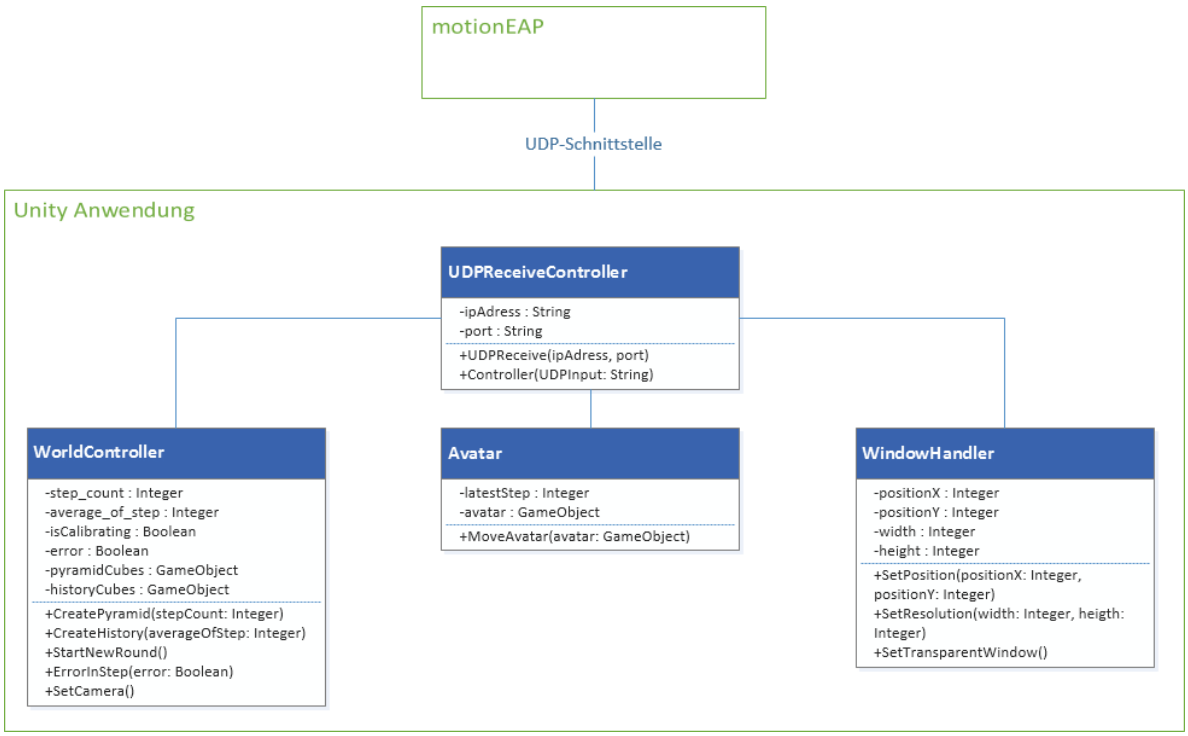


Abbildung 5.4.: Unity-Konzept Klassendiagramm

Die Klasse *WorldController* (Abb. 5.4) greift auf die abgespeicherten Informationen in den *Player.Prefs* zu und erstellt daraufhin die Spielumgebung. Die Spielumgebung enthält das Pyramiden-Konzept. Die Pyramide wird analog zur vorhandenen Implementierung aufgebaut. Das Pyramiden-Konzept wurde im Kapitel 3 – Stand der Technik im Abschnitt Pyramide bereits beschrieben.

In dieser Umgebung interagiert ein Avatar bzw. eine Spielfigur, der durch die *Avatar*-Klasse gesteuert wird. Das Verhalten der Spielfigur verhält sich analog zu der vorhandenen Pyramiden-Implementierung die im Kapitel 3 – Stand der Technik im Abschnitt Pyramide bereits beschrieben wurde.

In der Klasse *WindowHandler*, wird die Fensterposition in der Methode *SetPosition()* und die Fenstergröße in der *SetResolution()*-Methode eingestellt. In dieser Klasse werden zudem die Standard-Windowsfensterrahmen und der Hintergrund der Unity-Anwendung in der Methode *SetTransparent()* ausgeblendet. In der *SetTransparent()*-Methode wird zusätzlich eingestellt, dass die Unity-Anwendung immer im Vordergrund läuft.

5.3.2. Die Player.Prefs in Unity

Die *Player.Prefs* bieten in Unity eine Möglichkeit Informationen über die gesamte Laufzeitlänge eines Programms abzuspeichern. Diese Option ist vergleichbar mit einem Objekt, welches Variablen

beinhaltet mit dem Unterschied, dass sich der Entwickler um die Lebendigkeit der *Player.Prefs* nicht weiter kümmern muss. Variablen die in den Player.Prefs gespeichert sind, werden in Unity gesondert gespeichert. Diese Möglichkeit wurde von Unity bereitgestellt um Informationen über mehrere Stages hinweg zu sichern. In Unity bezeichnen Stages verschiedene Spielabschnitte (Levels). Falls der Spieler eine Spielumgebung (Level) verlässt und in eine nächste Spielumgebung (Level) wechseln möchte, werden normale Objekte und Variablen neu initialisiert. Es gibt Informationen, die beim Wechsel nicht verloren gehen sollen, dafür sind die Player.Prefs geeignet. Die Player.Prefs werden über die gesamte Laufzeit der Anwendung gespeichert. Falls in weiteren Versionen eine weitere Spielumgebung zum Pyramiden-Konzept hinzukommen sollte, könnte man ohne große Umstände eine neue Spielumgebung (Level) erstellen und auf die vorhandenen Player.Prefs-Variablen zugreifen. Falls Informationen in einem selbst definierten Objekt gespeichert werden würden, wären sie nur für das Pyramiden-Konzept verfügbar. Durch die Speicherung der Informationen in den Player.Prefs ist die Erweiterung der Unity-Anwendung, um ein weiteres Konzept, mit nur einem geringen Mehraufwand verbunden.

```
1 Player.Prefs.SetString('InfoVariable', 'Uni-Stuttgart');
2 Player.Prefs.GetString('InfoVariable');
```

Code 5.1: Player.Prefs Funktionen

Der Befehl Player.Prefs.SetString('InfoVariable', 'Uni-Stuttgart') (Abb. 4.3, Z: 1) erstellt eine neue String-Variable mit dem Namen *InfoVariable* und dem Inhalt Uni-Stuttgart in den Player.Prefs. Der Befehl Player.Prefs.GetString('InfoVariable') (Abb. 4.1, Z:2) gibt die angelegte Variable zurück und kann äquivalent zu einer String-Variable genutzt werden. Die *Player.Prefs* werden in den verschiedenen Betriebssystemen in einer speziellen Datei an unterschiedlichen Orten abgespeichert. In der beigefügten Quelle werden die Speicherorte und die Funktionsweise der Player.Prefs ausführlich beschrieben [Uni16b].

5.3.3. Eingliederung der Unity-Anwendung in motionEAP

Nachdem die technischen Details von motionEAP und das Konzept der Unity Anwendung vorgestellt wurden, wird anschließend die Eingliederung der Unity-Anwendung in motionEAP erläutert.

An zwei Stellen wird die motionEAP Steuerung Einfluss auf die Unity-Anwendung haben. Den Anfang bildet die Erstellung der Gamification-Komponente auf der Arbeitsfläche. Zum Erstellen einer Gamification-Implementierung in motionEAP muss auf der Arbeitsfläche der Button Gamification geklickt, gehalten und an die gewünschte Stelle auf der Arbeitsfläche gezogen werden. Sobald sich die Maus über der Arbeitsfläche befindet, verändert sich der Mauszeiger in ein Richtungskreuz, mit welchem signalisiert wird, dass die Gamification platziert werden kann. Um die Position der Komponente nach dem Platzieren zu verändern, kann auf die Gamification nochmals geklickt werden, dabei ist der Klick zu halten, währenddessen ist das Verschieben möglich. Auf der Arbeitsfläche wird die Gamification durch ein blaues Rechteck dargestellt. Eine weitere Möglichkeit das Gamification-Modul zu verschieben ist die Modifikation des Eintrags Position X bzw. Y in der Tabelle auf der linken Seite. Die beiden Koordinaten geben die genaue Position der Komponente auf dem Bildschirm an. Es ist möglich die Position durch eine Eingabe einer Zahl oder durch das Klicken auf die beiden Schaltflächen

höher und niedriger zu verändern. Die Angaben sind beim Starten der Unity-Anwendung für die Positionierung nötig.

Nachdem ein Workflow, erstellt worden ist, kann dieser geladen werden. Sobald ein Workflow eine Gamification-Komponente enthält, soll jene gestartet werden. Jedes Mal, wenn ein Workflow die gewünschte Information enthält, wird der Konstruktor der GamificationKo-Klasse aufgerufen. Um die Unity-Anwendung bei jedem Aufruf mit starten zu lassen, wird ein Befehl, der die Unity-Anwendung ausführt in den Konstruktor der Klasse geschrieben.

Vorausgesetzt ein Workflow wurde geladen und der Benutzer hat auf den Button-*Start Assembly* geklickt, beginnt daraufhin die Kalibrierungsphase der Gamification. Die Anzahl der Kalibrierungsdurchgänge ist in der *Integer*-Variable *calibrationAmount* gespeichert. Die Anzahl von drei Durchgängen hat sich in den Testläufen bewährt. Nach drei absolvierten Durchgängen, wird die Gamification-Komponente angezeigt. Die Geschwindigkeit der Einfärbung der einzelnen Pyramidenstufen orientiert sich an der Durchschnittszeit der drei Kalibrierungsdurchgängen.

Falls der Button-*End Assembly* geklickt wird, wird die gesamte Darstellung ausgeblendet. Nach einem Klick auf den Button-*Start Assembly*, wird die Darstellung wieder eingeblendet. An dieser Stelle benötigt die Unity-Anwendung mehrere Informationen, welcher Button wurde geklickt und wurde die die Kalibrierungsphase davor abgeschlossen. Jeweils eine Variable für die Übertragung des Klicks auf den *Start Assembly*- bzw. *End Assembly*-Button. Und eine Variable für die Beendigung der Kalibrierungsphase. In der GamificationKo-Klasse ist für das Erkennen der Kalibrierungsphase die Boolean-Variable *isCalibrating* zuständig. Die Variable ist *true* solange die Kalibrierung nicht abgeschlossen wurde. Nachdem die Kalibrierung durchgeführt wurde, wird die Variable auf *false* gesetzt. Die Funktion der Variable ist somit für die Steuerung der Unity-Anwendung geeignet und kann ohne Veränderungen als Befehl übermittelt werden.

In diesem Kapitel wurde aus den Studienergebnissen das Konzept für eine neue Gamification-Anwendung erarbeitet. Aufgrund der technischen Grenzen des motionEAP-System in Bezug auf die visuelle Darstellung wird ein Zweitsystem genutzt. Das Zweitsystem wird mithilfe der Unity-Engine realisiert. Anschließend wurde der Aufbau und die Steuerung des Zweitsystem beleuchtet. Die Beschreibung der Implementierung des Konzepts folgt im nächsten Kapitel.

6. Implementierung

Nachdem im vorigen Kapitel das Konzept erläutert worden ist, wird im Folgenden die Ausführung und Implementierung ebendiesem vorgestellt. Die Implementierung wird chronologisch in der, im Konzept vorgestellten Reihenfolge, implementiert. Als erstes wird die Unity Anwendung geschrieben, daraufhin werden die Befehle in motionEAP eingefügt und schließlich wird die Steuerung des gesamten Programms über motionEAP realisiert.

6.1. Unity-Editor

Im folgenden Abschnitt wird die Arbeitsweise und die einzelnen Besonderheiten des Unity-Editors vorgestellt. Im vorigen Kapitel wurden die Eigenschaften der Unity-Engine bereits beschrieben. An dieser Stelle werden die technischen Merkmale im Vordergrund stehen.

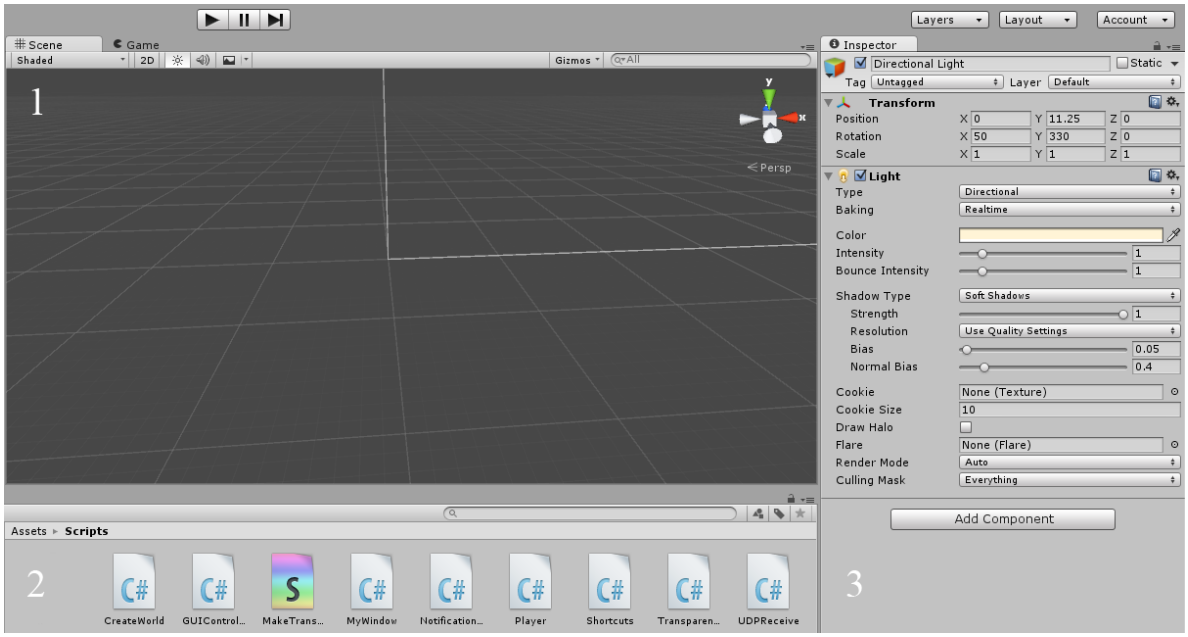


Abbildung 6.1.: Entwickleransicht der Unity-Anwendung

Unity wurde ursprünglich für die Spielentwicklung konzipiert, obwohl es heute für eine Reihe anderer Anwendungsfälle genutzt wird. Daher kommt die typische Dreiteilung der Anwendung, wie es für Editoren in der Spielentwicklung üblich ist. Die Abbildung 6.1 zeigt die Ansicht des Fensters des Unity-Editors. Die drei Teile wurde nummeriert und werden im Folgenden beschrieben. Der mit (1) markiert Abschnitt ist die Modellierungsansicht. Hier können Elemente im Raum erstellt, verschoben oder gelöscht werden. Im unteren Bereich (2) befinden sich alle, für das Projekt, benötigten Dateien, beispielsweise Skripte, Shader, Oberflächenmaterialien, 3-D Elemente und Levelabschnitte. Auf der rechten Seite (3) befindet sich das Optionenfenster, oft benötigte Einstellungen können hier verändert werden, alle weiteren Möglichkeiten müssen in den Skripten definiert werden. Ein Klick auf den oben in der Mitte befindlichen Play-Button erweckt die vorher definierte Spielwelt zum Leben und gibt dem Entwickler die Möglichkeit das Spiel zu testen. Für weitere Informationen, lohnt sich ein Blick in die sehr umfangreiche und ausführliche Unity Dokumentation [Uni16a].

An dieser Stelle wird das nachfolgend häufig genutzte Wort *Frame* erläutert. Mit *Frame* (engl.: Bild, Rahmen, Gestell) wird die Bildwiederholungsrate angegeben. Damit eine Bewegung für das menschliche Auge flüssig erscheint sind mindestens etwa 30 Bilder pro Sekunde (bzw. Frames per second) notwendig, fällt die Bildwiederholungsrate darunter, werden nur noch einzelne Bilder erkannt [CCD06].

Im Unity-Editor werden die Dateien, die den Quellcode enthalten, Skripte genannt. In den Skripten wird das Verhalten der gesamten Anwendung programmiert. Ein Skript enthält je eine Klasse. Beispielsweise enthält das Skript *Avatar* die *Avatar*-Klasse. Neben dem besonderen Umgang der Unity-Engine mit Klassen, verfügt die Engine über eine Reihe an speziellen Methoden.

6.1.1. Besondere Methoden der Unity-Engine

Unity besitzt einige Methode, die C# erweitern und besondere Eigenschaften besitzen. Die Methode spielen in der nachfolgenden Implementierung der Gamification-Anwendung eine wesentlich Rolle, und werden deshalb näher beleuchtet.

```
1  void Awake () {}
2  void Start() {}
3
4  void Update() {}
5  void LateUpdate() {}
6  void FixedUpdate() {}
7
8  void OnGUI (){}
9
10 void OnRenderImage() {}
```

Code 6.1: Besondere Unity Methoden

Die in Code 6.1 vorgestellten Methoden enthalten ein besonderes Aufrufverhalten. Üblicherweise wird eine Methode nur dann ausgeführt, wenn sie explizit aufgerufen wird. Die hier vorgestellten Methoden wichen davon ab.

Die Methode *Awake()* wird nur einmal zu Beginn der Lebenszeit der Instanz des Skripts aufgerufen, nachdem alle Objekte initialisiert wurden und bevor der erste Frame gestartet wird.

Die Methode *Start()* wird nur einmal zu Beginn der Lebenszeit der Instanz des Skripts aufgerufen, nachdem alle Objekte initialisiert wurden und direkt bevor der erste Frame anläuft. Der Unterschied zwischen den beiden Methoden *Awake()* und *Start* liegt in der Aktivierung des Skripts. Falls ein Skript deaktiviert wurde, wird nur die *Awake()*-Methode aufgerufen, bei Aktivierung, werden beide Methoden aufgerufen. Alle *Awake()*-Methoden werden zu Beginn des Starts einer Szene aufgerufen, erst danach werden die *Start*-Methoden aufgerufen. Im Falle, dass eine *Start*-Methode auf ein vorinitialisiertes Objekt zugreifen möchte, sollte das gewünschte Objekt in der *Awake()*-Methode initialisiert werden.

Die Methode *Update* wird einmal pro Frame aufgerufen. In dieser Methode werden alle Spielmechaniken implementiert, die ein solches, sich wiederholendes Verhalten, benötigen. Die Methode *LateUpdate()* wird ebenso einmal pro Frame aufgerufen, jedoch nachdem alle Befehle in *Update* durchgeführt wurden. Es wird empfohlen Kameras die Objekte verfolgen in *LateUpdate()* zu implementieren, da sich die Objekte normalerweise in *Update* bewegen, kann es sonst zu Fehlern kommen.

OnGUI() wird mindestens einmal pro Frame aufgerufen und zusätzlich pro Event. Die Methode wird für Veränderungen im Graphical User Interface (GUI) genutzt. Um Texte anzuzeigen, die sich pro Event ändern, jedoch minimal einmal pro Frame, ist diese Methode ideal.

Nachdem das Bild pro Frame gerendert wurde, wird *OnRenderImage()* aufgerufen. Diese Methode wird genutzt um nachträglich besondere Effekte einzufügen.

Neben den hier vorgestellten Punkten enthält C# für die Unity-Engine, im Gegensatz zur herkömmlichen C# -Programmiersprache eine Reihe weiterer Besonderheiten, dazu gehören beispielsweise die Player.Prefs als Speichermöglichkeit, die speziell für den Anwendungsfall der Computerspielentwicklung geschrieben wurden. Die zusätzlichen Besonderheiten erweitern C# entweder um Zusatzfunktionen oder verkürzen umständliche Programmierarbeiten. Weitere Besonderheiten werden im Laufe des nächsten Abschnitts vorgestellt.

6.2. Unity Implementierung

In diesem Teil werden die Programmierarbeiten der Unity-Anwendung beschrieben. Es werden Codeabschnitte vorgestellt, die entweder besondere Funktionen erfüllen oder für das Verständnis der Anwendung unerlässlich sind. Die Zeilen der Codeabschnitte sind nummeriert und werden im Format (Z: Zeilennummer) referenziert. Abweichungen vom Konzept werden an den entsprechenden Stellen beschrieben. Die Reihenfolge der vorgestellten Merkmale der Unity-Anwendung orientiert sich an der chronologischen Reihenfolge der erfolgten Implementierung. Zu Beginn werden die implementierten Grundfunktionen beschrieben.

6.2.1. Empfang der Daten

Einen wichtigen Kernpunkt des Programms bildet die Klasse bzw. das Skript *UDPReceive*. In der Klasse werden die Befehle von motionEAP empfangen und innerhalb der Unity-Anwendung weitergegeben.

```
1  public class UDPReceive : MonoBehaviour{
2  private void init (){
3     receiveThread = new Thread (
4       new ThreadStart (ReceiveData));
5     receiveThread.IsBackground = true;
6     receiveThread.Start ();
7   }
8
9  private void ReceiveData (){
10    client = new UdpClient (port);
11    while (true) {
12      try {
13        IPEndPoint anyIP = new IPEndPoint (IPAddress.Any, 0);
14        byte[] data = client.Receive (ref anyIP);
15        string text = Encoding.UTF8.GetString (data);
16        lastReceivedUDPPacket = text;
17      } catch (Exception err) {
18        print (err.ToString ());
19   }}} ... }
```

Code 6.2: Ausschnitt aus der UDPReceive-Klasse. Es werden Daten über die UDP-Schnittstelle empfangen und in eine String-Variable umgewandelt

Die Methode *Init()* (Code 6.2), Z: 2) wird in der *Start()*-Methode des Skripts *UDPReceive* aufgerufen. Der Thread *ReceivedData* wird daraufhin erstellt und läuft von da an parallel zur Anwendung im Hintergrund. In Zeile 10 des eben genannten Codeabschnitts, wird ein *UdpClient()* mit dem Namen *client* und dem Parameter *port* erstellt. Die Variable *port* wird mit dem festen Wert *20000* initialisiert. Die Übertragung der Befehle wurde auf den UDP-Port 20000 festgelegt, da auf diesem Port wenig Störquellen zu erwarten sind. Falls die Unity-Anwendung wider Erwarten auf dem Port 20000 etwas, von einer anderen Quelle als motionEAP empfangen sollte, ist die Anwendung so konzipiert, dass es zu keinem Fehler kommen sollte. Es werden ausschließlich die angeführten Befehle zur Steuerung der Anwendung genutzt, andere Zeichenketten werden ignoriert. Die Wahrscheinlichkeit, dass andere Quellen als motionEAP, den exakten Wortlaut der Befehle versenden, ist so gering, dass sie nicht weiter berücksichtigt wird.

Der Befehl *IPAddress.Any* (Z: 13) stellt sicher, dass die Unity-Anwendung auf allen IP-Adressen des Computers, auf welchem die Anwendung ausgeführt wird, mithört. Da sich die IP-Adresse des Computers ändern kann, ist an dieser Stelle, im Gegensatz zur festen Porteinrichtung, eine dynamische Abfrage notwendig. Ein Computer hat in der Regel eine IP-Adresse im lokalen Netzwerk und eine IP-Adresse im Internet. Der Befehl *IPAdress.Any* erhält beide Adressen, so dass der Empfang der Steuerungsbefehle über das lokale Netzwerk und das Internet möglich ist. In der üblichen Verwendung von motionEAP wird jedoch ausschließlich die lokale Übertragung genutzt. Nachdem der Empfang sichergestellt ist, werden alle ankommenden Daten im UTF-8 Format dekodiert (Z: 15) und in der *String*-Variable *text* gespeichert (Z: 16). Somit liegt eine empfangene Zeichenkette nun intern als *String*-Variable vor und kann von der nächsten Methode als Befehl entschlüsselt werden.

6.2.2. Steuerungsbefehle der Unity-Anwendung

Alle Steuerungsbefehle der Unity Anwendung sind in der Tabelle 6.1 angeführt. In der linken Spalte steht der reguläre Ausdruck, der von der Unity-Anwendung genutzt wird um die Befehle zu erkennen. In der nächsten Spalte steht der Ort in motionEAP, an welchem der Befehl gesendet wird. In der letzten Spalte befindet sich die Kurzfassung der Funktionsbeschreibung. Der Befehl zum Schließen der Anwendung wird in der Methode *shoutDownApplication()* der Klasse *BackendControl.cs* beim Beenden von motionEAP aufgerufen. Der Befehl wird gesendet, kurz bevor motionEAP beendet wird, somit wird auch die Unity-Anwendung geschlossen.

Befehl	Aufruf in motionEAP	Funktion
x	BackendControl.cs: shoutDownApplication()	Schließt die Unity-Anwendung
error	GamificationKo.cs failCurrentStep()	Zeigt einen Fehler an
po:-?[0-9]+;-?[0-9]+	GamificationKo.cs: writeInitSettings()	Enhält die Fensterposition
sc:[0-9]+	GamificationKo.cs: initGamification()	Enthält die Gesamtanzahl der Arbeitsschritte
ls:[0-9]+	GamificationKo.cs: workingStepStarted()	Gibt den aktuellen Arbeitsschritt zurück
av:[0-9]+	GamificationKo.cs: addTimeToPreviousTries()	Enthält den Wert für den vergangenen Durchlauf
ic:[0-9]+	GamificationKo.cs: writeInitSettings() WorkflowPanel.xaml.cs: stopWorkflowButton_Click() startWorkflowButton_Click()	Blendet die Spielumgebung aus bzw. ein
me:([0-9]+;)+	GamificationKo.cs: calcTimes()	Enthält die Arbeitszeiten pro Schritt
ScreenChecker:[0-1]+	settings.txt (externe Datei)	Enthält Informationen über die Bildschirmanordnung

Tabelle 6.1.: Die Tabelle enthält alle von der Unity-Anwendung erkannten Befehle

Der Befehl *error* wird gesendet, wenn der Mitarbeiter einen Fehler in der Fertigung der Bauteile begeht. Der Befehl wird ausschließlich im eben erwähnten Fall gesendet und ist für die entsprechende Reaktion der Spielwelt vorgesehen. Es handelt sich dabei um keinen kritischen Fehler, sondern um einen Fehler der durchaus oft vorkommen kann. Die Häufigkeit des Fehlers ist von der Arbeitsqualität des Mitarbeiters abhängig. Die verbleibenden Befehle laufen nach demselben Muster ab. Als erstes wird eine Beschreibung des Befehls, gefolgt von Zahlen, gesendet. Bei mehreren Zahlen in einem Befehl werden diese durch ein Semikolon getrennt erkannt. Das Vorgehen wurde im Kapitel Erweitertes Gamification-Konzept bereits erläutert. Die Buchstabenanzahl zu Beginn der Befehle wurde

abweichend von der Konzeptvorgabe, auf zwei Buchstaben gekürzt. Durch diese Änderung enthalten alle Befehle genau drei Zeichen am Anfang, was die weitere Verarbeitung vereinfacht. Die Erklärung der Notation der Tabelle (reguläre Ausdrücke) und die Verarbeitung der Befehle folgen als nächstes.

6.2.3. Verarbeitung der Befehle mithilfe regulärer Ausdrücke

Nachdem die Nachricht in der Methode *ReceiveData()* empfangen und in einer *String*-Variablen gespeichert wurde. Ist die *ControllerMethod(string inputLine)*-Methode (Code 6.3, Z: 1) für die weitere Verarbeitung zuständig. Die Methode wird mit dem Parameter *string inputLine* aufgerufen, was der empfangenen Zeichenkette in Code 6.2, Z:15 entspricht.

```
1 public void ControllerMethod (string inputLine){
2  Regex windowPosition = new Regex (@"po:-?[0-9]+;-?[0-9]+");
3  if (windowPosition.IsMatch(inputLine){
4    string[] numbers = Regex.Split(inputLine.Remove(0, 3), @";");
5
6    Int32.TryParse(numbers[0], out x_pos);
7    Int32.TryParse(numbers[1], out y_pos);
8       PlayerPrefs.SetInt("x_pos", x_pos);
9       PlayerPrefs.SetInt("y_pos", y_pos);
10 }
```

Code 6.3: Ausschnitt aus der UDPReceive-Klasse. In der Controller-Methode werden die empfangenen Daten mithilfe eines regulären Ausdrucks zugeordnet und anschließend abgespeichert

Die Methode *ControllerMethod(string inputLine)* wird hier verkürzt dargestellt. Es wird nur ein regulärer Ausdruck gezeigt. In der Implementierung kommt für jeden genutzten Befehl, ein regulärer Ausdrücke zum Einsatz. Zur Demonstration der Funktionsweise genügt hier allerdings eine einzelne Abfrage, da die übrigen analog dazu ablaufen.

An dieser Stelle werden reguläre Ausdrücke kurz vorgestellt. Reguläre Ausdrücke werden (hier) genutzt, um eine Zeichenkette auf einen bestimmten Inhalt hin zu überprüfen. Sobald der Inhalt erkannt worden ist, kann er weiter verarbeitet werden. In dem hier genutzten Anwendungsfall, empfängt die Unity-Anwendung eine unbekannte Zeichenkette die auf verschiedene Inhalte geprüft wird. Die unbekannte Zeichenkette wird immer einen Befehl enthalten der anhand der regulären Ausdrücke zugeordnet wird. Sobald der Befehl erkannt wird, reagiert die Unity-Anwendung darauf.

Falls die Zeichenkette *inputLine* auf einen regulären Ausdruck zutrifft, wird die entsprechende if-Abfrage aktiv. Der Ausschnitt zeigt die Abfrage für den Befehl, um das Fenster zu positionieren. Die Nachricht beginnt mit *po:* gefolgt von zwei durch Semikolon getrennten ganzen Zahlen. Da es sich hierbei nicht um Benutzereingaben, sondern in motionEAP definierten Eingaben handelt, wird darauf verzichtet fehlerhafte Eingaben abzufangen. Falls motionEAP nicht mitten im Senden abstürzt, gibt es keine Möglichkeit eine Fehleingabe zu produzieren. Sollte motionEAP abstürzen wird empfohlen die Unity-Anwendung ebenfalls zu schließen und neu zu starten. Nachdem der Befehl ankommt, werden die beiden Zahlen extrahiert, in Integer umgewandelt und als *Player.Prefs x_pos bzw. y_pos* gespeichert. Die erste Zahl enthält die X-Koordinate, die zweite Zahl enthält die Y-Koordinate. Eine

erhaltene Zeichenkette wurde nun als Befehl erkannt und intern so weiterverarbeitet, dass der Befehl als nutzbare Information vorliegt.

Bei der Vorgehensweise, die Befehle mithilfe von regulären Ausdrücken zu erkennen ist eine Herausforderung aufgetreten. Dadurch, dass die Befehle nicht manuell von einem Benutzer eingegeben und verschickt werden, sondern maschinell von motionEAP versendet werden, ist die Zeitspanne zwischen den einzelnen Befehlen sehr kurz. Die Befehle folgen für die Erkennung mittels der regulären Ausdrücke teilweise zu schnell aufeinander. Dadurch wird oft nur der letzte Befehl in einer Reihe vollständig ausgeführt. Dieses Problem ist an einer Stelle aufgetreten. Die Stelle konnte so umformuliert werden, dass statt der vielen Befehle nur noch ein Befehl benötigt wird. Die Erläuterungen der einzelnen Befehle folgen in den nächsten Abschnitten.

6.2.4. Fensterpositionierung

Die Klasse *MyWindow* wird genutzt um die Fensterposition zu verändern (Code 6.4. Da keine Unity native Lösung bereitgestellt wird, das Fenster der Anwendung nachträglich zu positionieren, muss auf Windows-Bibliotheken ausgewichen werden.

```
1 [DllImport("user32.dll", EntryPoint = "SetWindowPos")]
2 private static extern bool SetWindowPos(IntPtr hwnd, int hWndInsertAfter, int x, int Y, int
      cx, int cy, int wFlags);
3
4 [DllImport("user32.dll", EntryPoint = "FindWindow")]
5 public static extern IntPtr FindWindow(System.String className, System.String windowName);
6
7 void Update(){SetPosition (PlayerPrefs.GetInt("x_pos"), PlayerPrefs.GetInt ("y_pos"));}
```

Code 6.4: Windows user32.dll Funktionen in MyWindow

In diesem Fall wird die Windows-Bibliothek *user32.dll* genutzt. Der Befehl *FindWindow* (Code **??**) sucht das gewünschte Fenster, dessen Position verändert werden soll. Hier muss der Name des Fensters der Unity-Anwendung eingegeben werden. Nachdem das Fenster gefunden wurde, kann mit *SetWindowPos* die Position des Fenster verändert werden. Die *Update()*-Methode (Z:7) enthält den Aufruf *SetPosition(x,y)*, in die beiden genötigten Parameter werden die X- bzw- Y-Koordinaten aus den *Player.Prefs* geladen. Das Fenster verändert daraufhin die Position gemäß den erhaltenen Koordinaten.

6.2.5. Bildschirmanordnung

Bei der Steuerung der Position des Fensters der Unity-Anwendung kann ein Problem mit der Bildschirmanordnung des Computers auftauchen. MotionEAP wurde ursprünglich so konzipiert, dass die Anwendung auf dem Hauptbildschirm angezeigt wird und die Arbeitsfläche auf der zweiten Anzeige links davon. Im Falle von motionEAP ist die zweite Anzeige die Projektion auf der Arbeitsfläche, das bedeutet das Bild, was auf dem zweiten Bildschirm angezeigt werden soll, wird auf die Arbeitsfläche projiziert. Die Unity-Anwendung soll sich immer über der Anzeige der Arbeitsfläche befinden.

Abbildung 6.2.: Windows Bildschirmeinstellungen

In *Systemsteuerung\Darstellung und Anpassung\Anzeige\Bildschirmauflösung* kann die Anordnung der Bildschirme in Windows beliebig verändert werden, in der gezeigten Abbildung (Abb. 6.2) sind zwei Bildschirme dargestellt, die an der oberen Kante bündig zueinander ausgerichtet sind, wobei der Hauptbildschirm links und die zweite Anzeige rechts davon ist. Das ist die Standardeinstellung, wenn an einen Windows-Computer, ein zweiter Bildschirm angeschlossen wird. Die nicht optimale Einstellung kann zudem noch verändert werden. Allerdings muss bei jeder Einstellung sichergestellt werden, dass die Unity Anwendung sich auf dem korrekten Bildschirm und, falls die Bildschirme in der erwähnten Windows-Einstellung nicht bündig eingestellt sind, auch auf der korrekten Höhe.

Alle Sonderfälle müssen in motionEAP nicht unbedingt berücksichtigt werden, da die Hauptanwendung mit der Maus positionierbar ist und die Position der Arbeitsfläche durch die Tastenkombination *[Windowstaste] + [Pfeiltaste]* steuerbar ist. Diese Möglichkeiten bietet die Unity Anwendung aufgrund der ausgeblendeten Windows-Fensterrahmen und der Transparenz nicht mehr. Durch diese Einschränkung benötigt die Unity-Anwendung einen besonderen Befehl, der die Information enthält wie genau die Bildschirme positioniert sind. Dies wird über mehrere Abfragen sichergestellt.

```
1 Screen[] screens = Screen.AllScreens;
2 System.Drawing.Rectangle bounds = screens[screenChecker].Bounds;
3 double xrect = bounds.X;
4 double yrect = bounds.Y;
```

Code 6.5: Abfrage der Bildschirmpositionierung in motionEAP. Die gewonnen Informationen werden anschließend an die Unity- Anwendung übermittelt.

Der Befehl *Screen.AllScreens* (Z:1) (Code 6.5) gibt ein Array zurück, der alle Anzeigen im System enthält [Mic16b]. In der nächsten Zeile wird mit *screens[screenChecker].Bounds* die Auflösung des mit *screenChecker* ausgewählten Screens ausgegeben. Die Variable screenChecker wird in der *settings.txt* Datei abgefragt. Die Standardanordnung (der zweite Bildschirm ist links vom Hauptbildschirm angeordnet) der Bildschirme wird mit *screenChecker:1* angegeben, falls sich die Anordnung davon unterscheiden sollte, ist es nötig den Eintrag manuell auf *screenChecker:0* zu ändern. An dieser Stelle sei angemerkt, dass der zweite Bildschirm immer ein Projektor sein wird. MotionEAP ist ausschließlich auf den Betrieb mit einem Bildschirm und einem Projektor optimiert. Hier ist die Rede von einem zweiten Bildschirm, das hat den Grund zur Ursache, dass im Menü der Bildschirmauflösungen von Windows nur von einem zweiten Bildschirm die Rede ist, jedoch nicht von einem Projektor. Die

Befehle *bounds.X* und *bounds.y* (Z: 3f), geben die (0, 0)-Koordinaten des gewünschten Bildschirms aus.

Die (0, 0)-Koordinate gibt die Koordinaten des Punktes oben links eines Bildschirms an. Dieser wird so genannt, da er, falls nur ein Bildschirm angeschlossen ist, immer die Koordinaten x,y: (0, 0) haben wird. Falls ein zweiter Bildschirm angeschlossen ist, ändern sich die (0,0)-Koordinaten der Bildschirme. Der (0, 0)-Punkt des linken Bildschirms hat immer noch die Koordinaten (0, 0), der Punkt oben links auf dem rechten Bildschirm hat jetzt allerdings die Koordinaten (Breite des linken Bildschirms, 0). Die beiden Bildschirme werden intern wie ein langer Gesamtbildschirm verarbeitet. D.h. der rechte Bildschirm fängt dort an, wo der linke zu Ende ist. Ist der linke Bildschirm beispielsweise 900 Pixel breit, ist die (0,0)-Koordinate des rechten Bildschirms: (900, 0), dies trifft allerdings nur dann zu, wenn der linke Bildschirm als Hauptbildschirm markiert wurde. Wird hingegen der rechte Bildschirm als Hauptbildschirm markiert, verändern sich die Koordinaten folgendermaßen: die (0, 0)-Koordinate des linken Bildschirm ist dann (-900, 0) und die des rechten Bildschirms (0, 0). Damit die Unity Anwendung korrekt angezeigt wird ist die Abfrage notwendig, welcher Bildschirm der Hauptbildschirm ist und auf welcher Seite er sich befindet, diese Information wird in der Variable *screenChecker* gespeichert. Dadurch können die korrekten (0,0)-Koordinaten berechnet werden, daraufhin wird die Position der Gamificationkomponente abgefragt und die Koordinaten werden hinzu addiert.

6.2.6. Transparenz des Fensters der Unity-Anwendung

In der Klasse *TransparentWindow* (Code 6.6) wird die Transparenz des Fensters der Unity-Anwendung sichergestellt. Für diese Eigenschaft existiert ebenso, wie für die zuvor erwähnte Fensterpositionierung, keine Unity native Lösung. Für die Eigenschaft muss wiederum auf Windows-Bibliotheken zugegriffen werden. Wie in der Klasse *MyWindow* wird das gewünschte Fenster über *FindWindow* gesucht. Mithilfe des Befehls *SetLayeredWindowAttributes()* wird die Transparenz festgelegt. Der Parameter *hwnd* beschreibt das aktive Fenster. *crKey* enthält den Farbwert des Hintergrund der Transparent werden soll. *bAlpha* stellt den Grad Transparenz ein. Dieser Wert wird in derselben Skala von RGB-Hexadezimalzahlen festgelegt, hierbei ist 255 der höchste Grad an Transparenz. Die Hintergrundfarbe ist somit vollständig transparent. Die Objekte die nicht transparent erscheinen sollen, dürfen nicht die exakte Farbe des Hintergrunds besitzen, sonst würden diese auch transparent erscheinen.

```
1 [DllImport("user32.dll", EntryPoint = "SetLayeredWindowAttributes")]
2 static extern int SetLayeredWindowAttributes(IntPtr hwnd, int crKey, byte bAlpha, int
      dwFlags);
3
4 void Start(){SetLayeredWindowAttributes(hwnd, 0, 255, 2);}
5
6 void OnRenderImage(RenderTexture from, RenderTexture to){
7    Graphics.Blit(from, to, m_Material);}
```

Code 6.6: Windows user32.dll Funktionen in TransparentWindow

Der Hintergrund der Spielwelt wird transparent und nur die gewünschten Objekte sind sichtbar. Zu den gewünschten nicht-transparenten Objekten gehört die Pyramide, die Spielfigur und die Verlaufsanzeige. Zusätzlich besitzt das Fenster der Unity-Anwendung keine Windows-Fensterrahmen.

Dem Fenster fehlen somit auch die drei Windows-Buttons am rechten oberen Bildschirmrand. Die vollständige Fenstersteuerung wird somit von motionEAP übernommen.

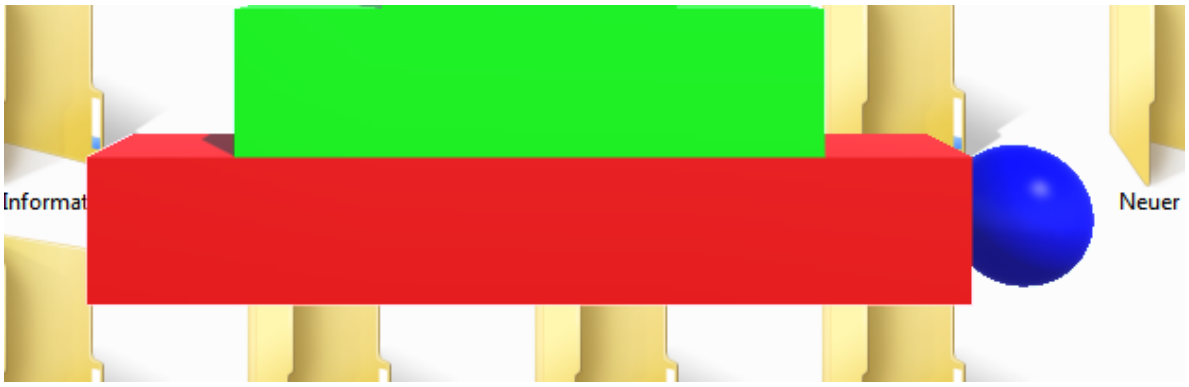


Abbildung 6.3.: Die Unity-Anwendung befindet sich über einem beliebigen Windowsordner, dadurch ist die Transparenz des Hintergrunds der Anwendung ersichtlich

Der Befehl *Graphics.Blit* (Code 6.6, Z:7) nimmt die exakte Hintergrundfarbe, die transparent erscheinen soll, entgegen. Dafür muss im Unity-Editor ein neues Material erstellt werden. Dem Material wird dieselbe Farbe zugewiesen wie dem Hintergrund, um diesen transparent erscheinen zu lassen (Code 6.3). Sind die Farben nicht exakt dieselben, erscheint der Hintergrund nicht vollständig transparent.

Zusammen mit dem vorangegangenen Absatz wurden alle Grundlegenden Funktionen der Unity-Anwendung vorgestellt. Es werden Zeichenketten entgegengenommen und als Befehle interpretiert. Der Hintergrund des Fensters erscheint transparent und es kann neu positioniert werden. In den nachfolgenden Absätzen wird der Aufbau und die Erstellung der Spielwelt und das Verhalten der Spielfigur erläutert.

6.2.7. Erstellung und Aufbau der Spielwelt

Die Klasse *CreateWorld* ist für die Erstellung der Spielwelt zuständig. In der Klasse werden die Objekte: Pokal, Pyramide und Verlaufsanzeige gesteuert. Sobald ein neuer Workflow geladen wird, prüft die Methode *createWorld()* ob eine vorhandene Spielumgebung existiert, ist das der Fall wird die Umgebung gelöscht und eine neue erstellt. Sollte keine Spielumgebung existieren wird der Löschvorgang übersprungen und es wird direkt eine neue aufgebaut. Der Aufbau erfolgt in den Methoden *createWorld()* und *createPyramid()*.

```
1 cube = GameObject.CreatePrimitive(PrimitiveType.Cube);
2 cube.transform.localScale += new Vector3(i * 2 + 1, 0 , 0);
3 cube.transform.position = new Vector3(-0.5F, -i, 0);
4 cube.GetComponent<Renderer>().material.color = Color.green;
```

Code 6.7: Aufbau von Spielobjekten in der Unity-Anwendung. Von oben nach unten Typ Größe Position Farbe des Objekts

Die Erstellung der Objekte erfolgt immer auf dieselbe Weise (Code 6.7). Die Pyramide ist aus Würfeln (engl. *Cubes*) aufgebaut. Im Codeabschnitt 6.7 (Z: 1) wird ein primitiver Würfel erstellt. Der gezeigte Codeabschnitt kann in einer Schleife durchgeführt werden, dadurch kann mühelos eine beliebige Anzahl an Würfel erschaffen werden. Die Größe eines Würfels, kann mithilfe des Befehls *transform.localScale* (Z: 2 des Codeabschnitts) eingestellt werden. In diesem Fall wird der Würfel in X-Richtung um ein vielfaches einer Variable gestreckt. Die Variable wird in einer Schleife genutzt und wird am Ende des Codeabschnitts hochgezählt, die Würfel werden somit für den Betrachter immer breiter. Der Codeabschnitt zeigt die Pyramidenerstellung, die nach unten hin immer breiter wird. Als Nächstes wird die Position des Würfels festgelegt. Die Y-Position der Würfel nimmt mit jedem Schleifendurchlauf ab. Die Würfel werden somit immer weiter nach unten versetzt. Danach wird jedem Würfel eine Farbe zugeteilt (Z: 4). Schließlich werden die Objekte (nicht mehr im Codeabschnitt zu sehen) in einem Array gespeichert. Die Speicherung der Objekte in einem Array erleichtert die folgende Funktion.

6.2.8. Ein- und Ausblenden der Objekte

Sollte sich das Programm im Kalibrierungsmodus befinden, wird die Spielwelt ausgeblendet *hideAll()*. Nach dem Abschluss der Kalibrierung wird die Spielwelt mithilfe von *showAll()* wieder eingeblendet.

```
1 public void showAll() {
2   foreach(Transform child in GameObject.FindGameObjectsWithTag ("Holder")[0].transform)
3     { child.GetComponent<MeshRenderer>().enabled = true; }}
```

Code 6.8: Einblenden von Objekten in der Methode showAll() der Klasse createWorld

Alle Objekte werden über dieselbe Funktion ein- oder ausgeblendet. Diese Ein- und Ausblendung betrifft nicht die Transparenz bzw. die Ausblendung des Hintergrunds. Die Ausblendung der Spielobjekte und die Ausblendung des Hintergrunds, sind grundverschiedene Funktionen und finden an verschiedenen Stellen des Codes statt. Die Ausblendung der Spielobjekte ist nur während der Kalibrierung erwünscht. Die Ausblendung des Hintergrunds erfolgt dauerhaft. Um ein Spielobjekt auszublenden wird der *MeshRenderer* ausgeblendet. Der *MeshRenderer* beinhaltet den sichtbaren *Render*-Teil des Objekts. Sobald der MeshRenderer auf *true* gesetzt wird, erscheint das Objekt wieder (6.8, Z.3). Durch die Speicherung der Objekte in einem Array, kann über das Array iteriert werden (6.8, Z.2) und eine gewünschte Funktion ausgelöst werden. In diesem Fall ist es die Ausblendung aller Objekte.

Um beispielsweise alle Spielobjekte der Pyramide zugreifen zu können, werden diese in einem eigenen Array gespeichert. Es gibt Arrays für die Pyramide, die Verlaufsanzeige, den Pokal und die Spielfigur. Zusätzlich werden alle Spielobjekte in einem Gesamt-Array gespeichert. Wird eine Änderung für die

gesamte Pyramide gewünscht, kann über das Pyramiden-Array iteriert werden, um die Änderung zu realisieren. Wird eine Änderung nur für eine einzelne Pyramidenstufe nötig, kann über den entsprechenden Index im Pyramiden-Array darauf zugegriffen werden. Eine häufig genutzte Funktion die auf die Array-Speicherung zugreift ist die Einfärbung der Spielobjekte, die im nächsten Abschnitt vorgestellt wird.

6.2.9. Einfärben der Objekte

Das Einfärben der Objekte ist eine wichtige Signalfunktion der Gamification-Implementierung, um dem Benutzer Informationen mitzuteilen. Die einzelnen Stufen der Pyramide werden in einer bestimmten Geschwindigkeit eingefärbt. Die Zeitspanne in der sich die Stufen von rot zu grün einfärben, ist von den Kalibrierungszeiten abhängig. Falls in der Kalibrierung ermittelt wurde, dass ein Arbeitsschritt zehn Sekunden dauert, wird der Farbverlauf der entsprechenden Pyramidenstufe, die genannte Zeitspanne benötigen, um von grün zu rot zu wechseln. Sobald der Benutzer länger als zehn Sekunden für den Arbeitsschritt braucht, bleibt die Stufe weiterhin rot.

```
1 void Update(){
2   cube.GetComponent<Renderer> ().material.color = Lerp3 (Color.green, Color.yellow, Color.red,
        currentTime / timeToMove * 0.32F);}
```

Code 6.9: Einfärben der Pyramidenstufen in der Update-Methode der Klasse CreateWorld.

Der in Codeabschnitt 6.9 (Z:2) genutzte Befehl *Lerp3(color, color, color, time)* wird verwendet um ein Material über einen dreifarbigen Verlauf in einer bestimmten Zeit einzufärben. Der hier gezeigte Befehl benutzt die Farben *Color.green* (grün), *Color.yellow* (gelb) und *Color.red* (rot) als Farbverlauf. Die Abstufungen zwischen den drei angegebenen Farben erfolgen automatisch. So erfolgt der Verlauf von *Color.yellow* (gelb) zu *Color.red* (rot) automatisch über orange. Der letzte Parameter bestimmte die benötigte Zeit, die der Farbverlauf benötigt, um das Objekte in den drei Farben einzufärben. Die Zeit wird hier 6.9 (Z:2) über den Quotienten der beiden Variablen *currentTime* und *timeToMove* berechnet. Beide Variablen enthalten den Zeitstempel ab wann ein Arbeitsschritt angefangen wurde. Ab diesem Zeitpunkt fängt die Einfärbung bei grün und verläuft zu gelb. Die Variable *timeToMove* behält den Zeitstempel, die Variable *currentTime* zählt die vergangene Zeitspanne seitdem Beginn des Arbeitsschrittes hoch. Durch eine Anpassung der Geschwindigkeit der Einfärbung über die Konstante *0.32F* wird die gewünschte Zeitspanne der Einfärbung berechnet. Im Beispiel des vorigen Abschnitts braucht ein bestimmter Arbeitsschritt zehn Sekunden um abgeschlossen zu werden. Nach fünf Sekunden ist die entsprechende Pyramidenstufe gelb-orange eingefärbt. Somit kann der Benutzer anhand der Farbgebung auf einen Blick erkennen wie viel Zeit vergangen ist. Wurde vom Benutzer ein Fehler begangen wird die Pyramidenstufe unabhängig der aktuellen Farbe rot eingefärbt und der Pokal wird ausgeblendet *hideCup()*. Nachdem ein Bauteil vollständig gefertigt worden ist, wird die gesamte Pyramide wieder grün eingefärbt und die Fertigung eines neuen Bauteils kann beginnen.

Die vorgestellte Einfärbung bezieht sich auf die Einfärbung der Pyramidenstufen, die als einzige einen zeitlichen Farbverlauf erhalten. Alle anderen Objekte ändern ihre einmal zugewiesene Farbe nicht mehr.

Die zeitliche Einfärbung der vorhandenen Pyramiden-Implementierung in motionEAP erfolgt über einen zehnstufigen Farbverlauf. Alle angezeigten Farben müssen zuvor einzeln definiert werden, dabei

existieren beim Farbverlauf keine Farben zwischen den zuvor definierten. In Unity wird der Farbverlauf automatisch über die Hexadezimaldefinition der Farben gestaltet. Somit kann ein feinerer Farbverlauf erreicht werden. Für das menschliche Auge sind die Farbabstufungen des Unity Farbverlaufs, im Gegensatz zum zehnstufigen Farbverlauf der vorhandenen Implementierung, nicht zu unterscheiden. Da die Einfärbung der Pyramidenstufen kontinuierlich über die gesamte Laufzeit der Anwendung stattfindet, ist das ein Punkt der sich auf die visuelle Akzeptanz der Anwendung merklich auswirkt.

6.2.10. Verlaufsanzeige

Die Verlaufsanzeige vermittelt dem Benutzer einen Eindruck über die Arbeitsgeschwindigkeit und -qualität seiner letzten gefertigten Bauteile. Sie befindet sich über der Pyramide und zeigt maximal letzten zehn Durchgänge an (Abb. 6.4. Die Anzeige besteht aus eingefärbten Würfeln. Durch die Verlaufsdarstellung kann der Benutzer auf einen Blick den Status der letzten zehn Bauteile erkennen. Die Implementierung befindet sich in der Klasse *CreateWorld*.

Abbildung 6.4.: Die Verlaufsanzeige zeigt den Status der vergangenen sechs Bauteile an

Die Abbildung 6.4 zeigt den Status der vergangenen sechs Bauteile an. In der Abbildung wurden bereits sechs Teile gefertigt, wobei das jüngste Teil durch den grünen Würfel unten rechts angezeigt wird. Für die farbliche Gestaltung der Würfel wird eine Array aus zehn Farben in den Abstufungen von grün über gelb über orange zu rot in der Methode *initializeColorRange()*, erstellt. Um die Farbe eines Würfels zu bestimmen, wird der farbliche Durchschnitt der Pyramide berechnet. Der Würfel wird daraufhin in der durchschnittlichen Farbe der Pyramide eingefärbt. Sind beispielsweise in einer vierstufigen Pyramide die ersten beiden Stufen rot und die letzten beiden grün, wird der Durchschnitt etwa gelb-orange sein. Der Würfel erhält dann eine gelb-orange Farbe. Sobald der Benutzer einen Fehler begeht, wird der Würfel rot eingefärbt unabhängig davon, ob die Pyramide sonst vollkommen grün ist. Die Funktionsweise der verlaufsanzeige wird in der Methode *createHistory()* durchgeführt. Sobald mehr als zehn Bauteile gefertigt wurden, wird der Status des ältesten Bauteils ausgeblendet.

Nachdem die vorangegangenen Klassen implementiert wurden, ist das Ergebnis auf Abb. 6.6 entstanden. Die Stufen besitzen bisher noch keine Einfärbung. Der Pokal (pinker Würfel) wird bei einem Fehler ausgeblendet. Die Spielfigur (gelber Würfel) bewegt sich zwischen den grünen Pyramidenstufen. Da die Spielfigur hier auf der ersten Stufe startet, es aber insgesamt drei Arbeitsschritte gibt, befindet sich die Spielfigur im letzten Arbeitsschritt weit über der Spitze der Pyramide. Im Falle eines Fehlers werden nicht die Stufen sondern die Spielfigur rot eingefärbt.

6.2.11. Positionierung der Kamera

Die Kamera in der Unity-Engine zeigt dem Benutzer einen Ausschnitt der Spielwelt. In dieser Implementierung ist die Kamera frontal zur Pyramide ausgerichtet und zeigt dabei die Spielobjekte Pyramide, Pokal, Spielfigur und Verlaufsanzeige. Die Anforderung an die Kamerapositionierung ist, dass es zu jeder Zeit möglich ist alle Spielobjekte zu betrachten. Die Kameraposition ist somit abhängig von der Pyramidenhöhe, da keine anderen Objekte ihre Größe bzw. Höhe ändern.

```
1 int distanceZ = -(PlayerPrefs.GetInt("totalStepCount") + 4);
2 if (distanceZ > -9) {
3    distanceZ = -9;}
4 camera.transform.position = new Vector3(-0.5F ,-(( PlayerPrefs.GetInt("totalStepCount") - 4)
      / 2 ), distanceZ);
```

Code 6.10: Einstellen der Kameraposition in der Methode setCamera

Dafür stellt die Methode *setCamera()* in der Klasse *CreateWorld*, die Kameraposition ein (Code 6.10. Der Befehl *camera.transform.position = new Vector3 (x,y,z)*, Positioniert die Kamera auf der X-, Y- und Z-Achse. Die X-Koordinate der Kamera wird nicht verändert. Die Y-Achsenkoordinate der Kamera skaliert mit der Höhe der Pyramide. Die Kamera befindet sich immer etwa auf der Hälfte der Höhe der Pyramide. Die Position auf der Z-Achse muss ebenso angepasst werden, bei hohen Pyramiden muss die Kamera nach hinten ausweichen, damit die gesamte Pyramide angezeigt werden kann.

6.2.12. Verhalten der Spielfigur

Die Spielfigur wird in der Klasse *Player* gesteuert. Zu Beginn wird die Kugel erstellt und am unteren Ende der Pyramide platziert. Nachdem der Befehl *ls:[0-9]+*, der den aktuellen Arbeitsschritt enthält, empfangen wird, bewegt sich die Spielfigur auf die Pyramidenstufe des entsprechenden Arbeitsschrittes. Die Bewegung geht insgesamt über drei Punkte: Startposition, Zwischenpunkt und Endposition. Über den Unity-Befehl *cube.transform.position = new Vector3 (x,y,z)*, wird die Figur auf der X-, Y- bzw. Z-Achse positioniert. Es wird ein neuer Vector3 mit dem Namen *destination* erstellt, der die Koordinaten des Zwischenpunktes übernimmt, daraufhin wird das Objekt solange mit *transform.position* nach oben bewegt bis die Y-Koordinate des Objekts mit der Y-Koordinate des Vektors *destination* übereinstimmt. Sobald dies der Fall ist, übernimmt der Vektor *destination* die Werte des Endpunktes und die Spielfigur wird so lange nach links bewegt bis die beiden X-Koordinaten übereinstimmen, wenn die Konstellation erreicht ist und somit die X- und Y-Koordinate der Spielfigur mit den Koordinaten des Endpunktes übereinstimmt, ist die Bewegung abgeschlossen.

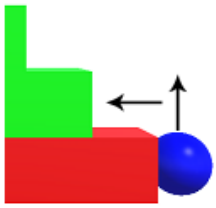


Abbildung 6.5.: Die Bewegung der Spielfigur wird in der Methode Update() der Player-Klasse definiert. Als erstes Bewegt sich die Spielfigur nach oben, sobald die entsprechende Höhe erreicht wurde, fängt die seitwärts Bewegung an.

Der Endpunkt wird anhand des aktuellen Arbeitsschrittes ermittelt. Falls ein Arbeitsschritt abgeschlossen wird, bevor die Spielfigur ihre vorige Endposition erreicht, wird der Endpunkt überschrieben und um eine Stufe weiter oben angesetzt. Die Figur bewegt sich ab dem Zeitpunkt, an welchem der neue Arbeitsschritt begonnen wurde, solange nach oben in Y-Richtung bis die gewünschte Stufenhöhe erreicht ist, ab diesem Zeitpunkt beginnt die Bewegung in X-Richtung.

Sobald die Figur, die Spitze der Pyramide erreicht hat und eine neue Runde beginnt, wird sie ohne eine flüssige Bewegung direkt wieder am Fuß der Pyramide platziert. Es wurden Versuche unternommen die Spielfigur in einer flüssigen Bewegung nach unten zu bewegen. Falls jedoch der erste Arbeitsschritt zu schnell getätigt wird, erreicht die Spielfigur nicht schnell genug die Startposition. Damit fehlt ein eindeutiges Zeichen des Rundenbeginns, was verwirrend wirkt und aus diesem Grund nicht realisiert wurde.

6.2.13. Zwischenstand der Implementierung

Nachdem die Funktionen der erwähnten Abschnitte realisiert wurden, befindet sich die Implementierung in der Hälfte der Bachelorarbeit auf dem Stand der Abb. 6.6.

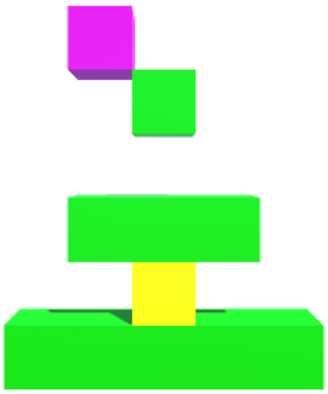


Abbildung 6.6.: Erste Version der unity-Anwendung

Die unfertige Version der Implementierung besteht aus der Pyramide, der Spielfigur (gelber Würfel) und dem Pokal (pinker Würfel) (U7). Die Farben sind fest vergeben worden und die Verlaufsanzeige fehlt. Die Spielfigur befindet sich zu Beginn auf und nicht neben der ersten Pyramidenstufe. Im letzten Arbeitsschritt schwebt die Spielfigur, um einen Schritt zu weit oben (ist auf der Abbildung nicht zu sehen). Nach vielen weiteren Anpassungen, ist das Endergebnis entstanden, das in Abschnitt 6.2.16 vorgestellt wird.

6.2.14. Startverhalten der Unity-Anwendung

Ein Punkt des Konzepts sieht vor beim Laden eines Workflows, der die Gamification enthält, die Unity-Anwendung zu starten. Sobald ein Workflow geladen wird, der keine Gamification enthält, soll die Unity-Anwendung beendet werden. Die Unity-Anwendung soll also nach Bedarf gestartet und beendet werden. Dieser Punkt erwies sich als besonders anspruchsvoll. Es ist das Problem aufgetaucht, dass der Workflow, der als JSON-Objekt gespeichert wird, fehlerhaft ist. JSON (JavaScript Object Notation) ist ein Datenformat, welches zur Speicherung von beliebigen Informationen genutzt wird. JSON-Objekte haben die Eigenschaft, dass sie für Menschen in einer einfach lesbaren Form verfügbar sind [jso16].

Der Workflow (JSON-Objekt) enthält die Anzahl der, auf der Arbeitsfläche verwendeten Objekte, deren Größe, Position, Rotation und noch einige andere Informationen. Das Problem besteht in der falschen Information zur Anzahl der Objekte. Die Gamification-Klasse: *GamificationKo.cs* wird, auch wenn sie sich nur einmal auf der Oberfläche befindet, fälschlicherweise mindestens zweimal in das JSON-Objekt geschrieben, in einigen Situationen auch öfter. Dadurch wird der Konstruktor der *GamificationKo.cs*-Klasse unregelmäßig oft aufgerufen. Falls die Unity-Anwendung im Konstruktor gestartet wird, wird die Anwendung somit immer mehrmals ausgeführt, was zu einer Reihe an kritischen Fehlern führt.

Es wurden verschiedene Lösungswege getestet, von welchen sich schließlich einer als annehmbar erwiesen hat. Da anfangs bekannt war, dass der Konstruktor immer zweimal aufgerufen wird, wurde

im Konstruktor eine Schleife implementiert die bis zwei zählt, daraufhin würde die Unity-Anwendung nur bei jedem zweiten Aufruf gestartet werden. Dieser Ansatz erwies sich in einer Reihe von Versuchen als unbrauchbar, da im JSON-Objekt die Anzahl der gespeicherten Gamification-Objekte keinem nachvollziehbaren Muster entspricht. Ein ähnlicher Ansatz mit einer *Boolean*-Variable, die weitere Ausführung der Unity-Anwendung blockierte sollte, sobald eine Unity-Instanz offen war, erwies sich in der Praxis als nicht realisierbar. Es konnte nicht zwischen absichtlich und unabsichtlich geöffneten Anwendungen unterschieden werden.

Eine weiterer Lösungsansatz sah vor, dass die Unity-Anwendung selbst erkennt ob sie schon einmal geöffnet ist und falls dies der Fall sein sollte, wird ein weiterer Aufruf blockiert. Dieser Versuch scheiterte jedoch an den sehr kurz aufeinander folgenden Aufrufen der Gamification-Klasse im JSON-Objekt. Die zweite und dritte Unity-Anwendung wurde bereits in Auftrag gegeben, ohne, dass die erste überhaupt wirklich geladen war. Um weitere Ausführungen zu blockieren müsste die Unity-Anwendung allerdings vollständig geladen werden und eine kurze Zeit lang ausgeführt werden. Außerdem kommt hier erwähnte Problem hinzu, dass nicht unterschieden werden kann, ob die Anwendung absichtlich oder unabsichtlich aufgerufen wurde.

Es wurde festgestellt, dass das Problem an einer veralteten Version von motionEAP liegt. Da durch eine Abspaltung des Gamification-Projektes von dem motionEAP-Hauptprojekt vor einiger Zeit zur Folge hat, dass keine Updates der vergangenen Zeit aufgespielt wurden, ist der Fehler im aktuellen motionEAP-Hauptprojekt behoben wurden, jedoch nicht mehr im Gamification-Projekt. Nach einem Gespräch mit den Verantwortlichen des motionEAP Teams, ist man zur Lösung gekommen, dass nach Abschluss dieser und noch einiger anderen Abschlussarbeiten, alle abgespaltenen Projekte vom motionEAP-Hauptprojekt zusammenführt werden. Dadurch wäre der Fehler behoben. Da das motionEAP eine gewisse Größe erreicht hat, ist es nicht möglich gewesen eine Zusammenführung der Projekte im Rahmen dieser Bachelorarbeit durchzuführen.

Das Startverhalten der Unity-Anwendung wurde darauf festgelegt, die Anwendung parallel mit der motionEAP-Anwendung starten zu lassen. Die Unity-Anwendung läuft somit unbemerkt im Hintergrund mit, bis ein Befehl zur Aktivierung erhalten wird. Da die Parallelität der Ausführung keinerlei negative Konsequenzen mit sich zieht, ist dies, bis zur Zusammenführung, eine annehmbare Lösung. Um den Konzeptansatz dennoch zu realisieren sind nur noch sehr wenige Änderungen nötig, da alle anderen Komponenten auf beide Lösungsansätze optimiert wurden.

```
1 class BackendControl {
2    public void startUpApplication() {
3       startInfo.FileName = @"\run\run.exe";
4       Process.Start(startInfo);
5    ...}}
6 NetworkManager.Instance.SendDataOverUDP(ipAdressGamification, portGamification, "x");
```

Code 6.11: Der Unity-Startaufruf in motionEAP

Die Unity-Anwendung wird derzeit in der Methode *startUpApplication()* der Klasse *BackendControl.cs*, über den Befehl *Process.Start(startinfo)*, gestartet (Code 6.11, Z: 4). Der Befehl startet eine Anwendung in dem angegebenen Pfad. Um, die im Konzept vorgestellte Lösung, zu realisieren ist es nötig den Aufruf in beide Konstruktoren der *GamificationKo.cs*-Klasse zu schreiben. Bevor der Startaufruf getätigt wird ist der Schließbefehl aus Tabelle 6.1 bzw. (Code 6.11, Z: 6) der ersten Zeile nötig. Dadurch

wird sichergestellt, dass bei jedem Aufruf nicht immer weitere Instanzen dazu kommen, sondern, die zuvor geöffneten Unity-Anwendungen werden geschlossen, bevor eine weitere hinzukommt.

6.2.15. Startparameter der Unity-Anwendung

Eine weitere Besonderheit, die von der üblichen Vorgehensweise, nämlich das Unity-Programm vollständig über motionEAP steuern zu können, abweicht, bezieht sich auf das Startverhalten der Unity-Anwendung. Damit die Anwendung von Anfang an richtig positioniert wird, müsste ein Befehl genau zu dem Zeitpunkt kommen, nachdem die Anwendung vollständig geladen wurde. Es ist unmöglich diesen Zeitpunkt fest zu machen, da er je Typ und Auslastung des Computers variiert. Die Unity-Anwendung muss also nachdem sie vollständig geladen wurde, selbstständig auf die Positionierungsbefehle zugreifen können.

Ein gängiger Lösungsansatz für diese Problemstellung ist die Erstellung einer zusätzlichen Text-Datei, die die erforderlichen Befehle enthält. Die Text-Datei wird nach dem Start eingelesen. In diesem Fall wird eine Text-Datei mit dem Namen *settings.txt* in demselben Verzeichnis angelegt, in welchem auch die ausführbare Unity-Datei liegt. Diese Datei enthält die Befehle, ScreenChecker:0|1, ic:0|1 und po:x;y (Tabelle 6.1, Z: 3, 7, 9). Das Einlesen der Datei wird in der Unity-Anwendung in der Methode *getInitSettings()* ausgeführt. Darin wird die Datei Zeile für Zeile durchgegangen und die einzelnen Zeilen werden, analog zu den empfangenen Daten, mit regulären Ausdrücken verglichen. Die weitere Verarbeitung der Befehle aus der Text-Datei entspricht vollständig der vorgestellten Vorgehensweise das Abschnitts 6.2.3.

6.2.16. Endergebnis der Implementierung

Die letzte Version der Unity-Anwendung bei Beendigung der Bachelorarbeit (Abb. 6.7) (U7). Im Gegensatz zum Zwischenstand wurde die Verlaufsanzeige hinzugefügt. Die Spielfigur wurde zu einer blauen Kugel geändert. Im vorhandenen Pyramiden-Konzept besteht die Spielfigur tatsächlich aus eine vereinfachten Figur. In einer zweidimensionalen Darstellung kann eine Spielfigur mit etwas Aufwand zu einem akzeptablen Ergebnis führen. Anhand der Erstellung des Pokals ist die Erfahrung gemacht worden, dass die Erstellung von eigenen Objekten im dreidimensionalen Raum eine äußerst Zeitintensive Angelegenheit werden kann. Eine Spielfigur ist im Vergleich zu einem Pokal eine deutlich umfangreichere Aufgabe. Die Figur sollte vom Stil her in die Spielwelt passen und bestenfalls noch animierte Bewegungen unterstützen. Dies war aufgrund meiner fehlenden Erfahrung in der dreidimensionalen Modellierung in einem angemessenen zeitlichen Rahmen nicht zu realisieren. Eine Kugel passt optisch zum implementierten Verhalten der Spielfigur (siehe Abschnitt 6.2.12).

festgestellt worden, dass eine Spielfigur die vom Stil her in die Spielwelt passen würde, erheblich

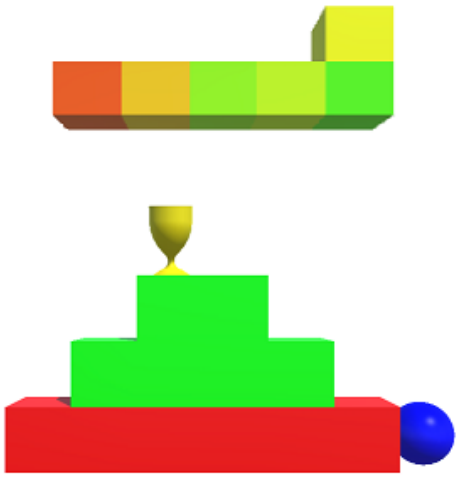


Abbildung 6.7.: Abschließende Version der Unity Anwendung

Die Anwendung wird vollständig über motionEAP gesteuert, nimmt insgesamt acht Befehle entgegen, kann dynamisch an jegliche Anordnung zweier Bildschirme angepasst werden, ist transparent und befindet sich immer im Vordergrund. Welche, in der Studie ermittelten Punkte nicht in die Implementierung aufgenommen wurden, aber dennoch interessante Anknüpfungspunkte bieten, wird im Kapitel Zusammenfassung und Ausblick weiter behandelt.

7. Zusammenfassung und Ausblick

Das letzte Kapitel dieser Bachelorarbeit ist in zwei Teile gegliedert, der erste Teil beinhaltet einen Rückblick und eine Zusammenfassung der geleisteten Arbeit. Der zweite Teil enthält einen Ausblick zur Zukunft der entstandenen Implementierung und die Einschätzung des Autors zum Thema Gamification.

7.1. Zusammenfassung und Fazit der Arbeit

Die Erkenntnis, dass Gamificationmechanismen Menschen erstaunlich effektiv motivieren können ist während der Erarbeitung des zweiten Kapitel gemacht worden. Darin sind auch psychologische Erklärungsmodelle zur menschlichen Psyche bezüglich Motivationsfaktoren enthalten. Nach der Einführung in das Thema Gamification erfolgt ein Überblick über den aktuellen Stand von Assistenzsystemen und Gamification in der Produktion. Man kann erkennen, dass neue digitale Technologien in große Produktionsunternehmen zunehmend Einzug erhalten. Welche Technologien sich durchsetzen werden und somit den Vorsprung vor der Konkurrenz gewährleisten, wird die Zukunft zeigen.

Nach den Erkenntnissen im vorangegangenen Abschnitt die sich durch Literaturrecherchen gewinnen ließen, erfolgen die ersten selbst erarbeiteten Ergebnisse. Anhand der Benutzerstudie bei der AUDI AG wurde die vorhandene Gestaltungslösung aus der Benutzerperspektive evaluiert. Die Gestaltungslösung beinhaltet drei Gamificationkonzepte im Projekte motionEAP. Der erste, der beiden Studientagen, wurde ist in Zusammenarbeit mit Herrn Dr. Oliver Korn durchgeführt worden. Die Auswertung der Evaluation ergab, dass die Mitarbeiter der AUDI AG das Gamificationkonzept der Pyramide am positivsten bewerten und mit den visuellen Aspekten der vorhandenen Implementierungen unzufrieden waren.

Daraufhin ist ein Konzept erarbeitet worden, dass die vorhandenen Gestaltungslösungen auf Grundlage der Evaluationsergebnissen verbessern soll. Im Konzept sind nicht alle Eventualitäten bedacht worden, sodass während der Implementierung einige, jedoch wenige, Herausforderungen zu meistern waren. Die Problematik der Verarbeitung der über die UDP-Verbindung empfangenen Befehle, die durch eine zu rasche Abfolge entstanden war, ebenso das Abfangen aller Sonderfälle der Bildschirmanordnungen wurden schließlich mit optimalen Lösungen beantwortet. Die rasche Abfolge konnte durch Vereinfachung der Befehle in motionEAP gelöst werde. Die Sonderfälle der Bildschirmanordnung konnten über einige zusätzliche Abfragen gelöst werden.

Die Erstellung der Spielwelt und gesamte visuelle Darstellung hat, bis auf einige Anpassungsschwierigkeiten, keinerlei Probleme bereitet. Daran wurde deutlich, dass die Unity-Engine eine sehr gute Wahl war. Das ist besonders erwähnenswert, da es keine Selbstverständlichkeit ist, dass es sich mit Entwicklungsumgebungen so zufriedenstellend arbeiten lässt. Die einzige Herausforderung, die nicht

so gelöst werden konnte, wie im Konzept vorgesehen, ist das Startverhalten der Unity-Anwendung. Es war vorgesehen die Anwendung nur bei Bedarf auszuführen, das konnte durch gegebene Voraussetzungen nicht realisiert werden. Da es zu fehlerhaften Speicherungen von Arbeitsabläufen in der verwendeten Version von motionEAP kommt, wurde eine parallele Ausführung von motionEAP und dem Unity-Programm umgesetzt. Nachdem die motionEAP-Version auf den neusten Stand gebracht wird, kann die im Konzept erarbeitete Lösung durch wenige Anpassungen erreicht werden.

Während der Arbeit ist deutlich geworden, dass es sinnvoll ist die Entwicklung von interaktiven nutzerzentrierten Anwendungen anhand der verwendeten ISO 9241-210:210-Norm zu realisieren. Ein einzelner (oder auch mehrere) Entwickler kann nicht alle Probleme und Herausforderungen in Erwägung ziehen, die durch eine Benutzerstudie deutlich werden. Eine standardisierte Vorgehensweise wie die verwendete ISO-Norm gibt dabei wichtige und hilfreiche Orientierungspunkte bei der Entwicklung. Mit der vorliegenden Arbeit wurde somit ein weiterer Entwicklungszyklus der erwähnten ISO-Norm abgeschlossen. Mögliche Anknüpfungspunkte und ein allgemeiner Ausblick in die Zukunft der gemachten Gamification-Implementierung werden im nächsten Abschnitt erläutert.

7.2. Ausblick

Der Abschnitt Ausblick ist in zwei Teile gegliedert. Als erstes erfolgt ein Ausblick zur erfolgten Implementierung. Anschließend wird die Meinung des Autors zur Zukunft von Gamification ausgedrückt.

7.2.1. Implementierung

Durch die vorhandene Implementierung, sind auf technischer Seite geeignete Voraussetzungen für mögliche Nachfolgeprojekte entstanden. Die durch Teilnehmer der Benutzerstudie oft bemängelte visuelle Darstellung hat durch die Verwendung der Unity-Engine eine deutlich Aufwertung, sowohl auf der Benutzerseite, als auch auf der Entwicklerseite erhalten. Die Implementierung von weiteren Gamification Mechanismen wurde, aufgrund der umfangreichen und zeitlich intensiven Portierung der gesamten vorhandenen Gamification-Konzepte in das neue System, nicht vorangetrieben. Durch die gemachte Implementierung wurde eine Datenübertragung zwischen motionEAP und der neuen Unity-Anwendung erstellt. Das somit entstandene System bietet eine geeignete Ausgangssituation für die Implementierung weiterer Gamification-Mechanismen.

In der Studie wurde der Wunsch nach erweiterten Bewegungserkennungen, die sich nicht ausschließlich auf die Arbeitsfläche beziehen, geäußert. Der Wunsch ist für eine weitere Bachelorarbeit womöglich zu umfangreich, allerdings könnte eine erweiterte Bewegungserkennung in ein Nachfolgeprojekt von motionEAP einfließen. Somit könnten, neben der Arbeitsgeschwindigkeit und -qualität, neue Faktoren wie die Haltung und die Benutzung von vorgesehenen Hilfsmitteln in die Gamification-Implementierungen einfließen.

Weitere Projekte die Gamificationkomponenten beinhalten, sollten sich mit dem Problem der Langzeitmotivation der Gamification beschäftigen. Dies wurde als eine der Hauptsorgen der Mitarbeiter in der Auswertung der Studie identifiziert. Wie reagieren Mitarbeiter auf ein Programm, das über

Wochen und Monate hinweg nebenher am Arbeitsplatz läuft, ist eine dringende Frage die geklärt werden sollte. Eine Serienproduktion ist ohne die Beantwortung dieser Frage, meiner Meinung nach undenkbar.

Durch die Studie konnte außerdem gezeigt werden, dass die Akzeptanz von Gamification von regulären Montagemitarbeitern, ebenso vorhanden ist, wie von leistungsgeminderten Mitarbeitern. Da Gamification in anderen Anwendungen weitläufig ebenso von nicht-leistungsgeminderten Menschen genutzt wird, kann somit davon ausgegangen werden, dass Gamification in der Produktion mit nicht-leistungsgeminderten Mitarbeitern in Zukunft gut möglich ist.

Nachdem mit dieser Bachelorarbeit ein Zyklus der ISO-Norm 9241-210 abgeschlossen wurde, bleibt abzuwarten, wie die hier realisierte Implementierung in einem neuen Zyklus abschneidet und was den Benutzern verbesserungswürdig erscheint.

7.2.2. Gamification

Aufgrund der im Kapitel 2 – Hintergrund gezeigten Entwicklungen bezüglich der Verbreitung von Gamification, kann man davon ausgehen, dass die Mechanik auch in den kommenden Jahren an Bedeutung dazugewinnt. Die im selben Kapitel erörterten Erkenntnisse, dass die verschiedenen Gamificationmechanismen sehr geeignet sind die verschiedenen Motivationsquellen von Menschen aktivieren, stützen meine Behauptung, dass Gamification zukünftig nicht an Bedeutung verliert. Es ist bemerkenswert, wie mit vergleichsweise wenig Aufwand Menschen zu mehr Leistung motiviert werden können. Da durch Gamification keine höheren Löhne, Bonuszahlungen oder größere Geschäftswägen an Mitarbeiter fällig werden, aber es dennoch zu Leistungssteigerungen kommen kann, ist das eine Mechanik die für viele Unternehmensbereiche interessant wäre.

Nach eigenem Empfinden wurden in den letzten Jahren viele digitale Anwendungen massiv mit Gamification erweitert. In Computerspielen ist man dazu übergegangen die Spieler für jeden gemachten Schritt zu bewerten und zu belohnen. Beim Erlernen von herkömmlichen Sprachen und Programmiersprachen setzen die meisten Webseiten mittlerweile auf Fortschrittsanzeigen, Highscorelisten und Levelstufen. Eine Webseite ohne die genannten Elemente ist schon fast undenkbar. Das Erreichen eines Flow-Erlebens ist, durch die Perfektionierung der Anpassung der Schwierigkeitsgrade an alle Geschicklichkeitsstufen in den verschiedenen digitalen Anwendungen, zudem einfacher als in der analogen Welt. Das sind alles Gründe, weshalb Gamification in vielen Anwendungen einen festen Platz erhalten hat.

Da Gamification oft keine großen Kosten mit sich bringt und in einigen Anwendungen erfolgreich ist, wird die Zukunft zeigen wie gut sich Gamification in der Produktion etablieren kann. Durch die kommende Industrie 4.0 werden zudem geeignete Voraussetzungen für Gamification in der Industrie und Produktion geschaffen.

Durch die mit der Bachelorarbeit verbundenen Recherchearbeiten, ist mir aufgefallen, dass es wenige kritische Stimmen gegenüber Gamification gibt. Kritische Meinungen, die sich nicht auf kurze Blogeinträge beschränken, sondern gut fundierte wissenschaftliche Abhandlungen sind, wie es auf der Seite der Fürsprecher für Gamification der Fall ist, sind noch seltener der Fall. Da Gamification meist so implementiert wird, dass es nicht störend oder behindernd wirkt, scheint es eine Methodik

zu sein, die im schlimmsten Fall auf Desinteresse stößt. Es ist schwierig zu einer Sache eine wirkliche Ablehnung zu entwickeln, wenn man sie so einfach ignorieren kann. Hingegen werden Menschen die auf Gamification reagieren in vielen Fällen mehr Leistung zeigen, das kann ich nicht zuletzt aus vielfacher persönlicher Erfahrung bestätigen. Somit erzeugt Gamification im schlimmsten Falle wenige und unbedeutende negative Erfahrungen und im besten Falle ein deutliches Plus an Motivation. Durch die aufgezählten Punkte, sehe ich keinen Grund, dass die Popularität von Gamification abnehmen könnte und bin gespannt auf die weitere Entwicklung dieser vielversprechenden Methodik.

Danksagung

Am Schluss dieser Arbeit möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Abschlussarbeit und während des Bachelorstudiums begleitet und unterstützt haben: Dominik Bilgery, Jonathan Kausch, Michael Blatz, Jennifer Them, Martin Zraly, Thomas Kosch, Dr. Oliver Korn und Hana Muschick.

Besonders möchte ich mich bei Herrn Michael Blatz von der Korion GmbH bedanken, der mir bei der Implementierung hervorragende Hilfe geleistet hat und mir jederzeit ein exzellenter Ansprechpartner war.

Zusätzlich bedanke ich mich bei Herrn Jonathan Kausch, der über einen langen Zeitraum auf unvergesslich zuvorkommende und freundliche Art und Weise alle Fragen zu vielen Teilgebieten des Studiums beantwortet hat.

Des Weiteren bedanke ich mich für das erstklassige Korrekturlesen bei Jennifer Them, die mir darüber hinaus mit zahlreichen wertvollen Ratschlägen beim Schreiben der Arbeit geholfen hat.

Abschließend möchte ich mich besonders bei den zahlreichen Kommilitonen und Freunden bedanken, die mir mit viel Geduld, Interesse und Hilfsbereitschaft während des Studiums zur Seite standen und die erlebte Zeit unvergesslich gemacht haben.

A. Anhang

	Vor dem Experiment: Befinden	sehr gut	gut	normal	nicht gut	Schlecht
01A	Wie geht es Ihnen heute?					
	Vor dem Experiment: Erfahrung	gar nicht	<2 Stunden	2-5 Stunden	5-10 Stunden	über 10 Stunden
02A	Wieviele Stunden pro Woche arbeiten Sie mit dem Computer?					
03A	Wieviele Stunden pro Woche spielen Sie Computer- oder Konsolenspiele?					
	Vor dem Experiment: Haltung	stimmt überhaupt nicht	stimmt nicht	neutral	stimmt	stimmt genau
04A	Ich mag Computerspiele.					
05A	Spielerische Elemente in Arbeitsprozesse einzubinden ist eine gute Idee.					
06A	Spielerische Elemente in die Montage bei Audi einzubinden ist eine gute Idee.					
	Nach dem Experiment: Visuelle Gestaltung	stimmt überhaupt nicht	stimmt nicht	neutral	stimmt	stimmt genau
07T	Die visuelle Darstellung beim Tetris-Design hat mir gefallen.					
08K	Die visuelle Darstellung beim Kreis&Balken-Design hat mir gefallen.					
09P	Die visuelle Darstellung beim Pyramiden-Design hat mir gefallen.					
I1	**Wie müsste eine für Ihren Arbeitsbereich geeignete Visualisierung aussehen? (Farbigkeit, animiert oder statisch…)**					
10A	Spielerische Elemente sollten direkt in den Arbeitsbereich projiziert werden.					
11A	Spielerische Elemente sollten besser neben dem Arbeitsbereich angezeigt werden.					

Abbildung A.1.: Fragebogen Teil 1

	Nach dem Experiment: Spielmechanik	stimmt überhaupt nicht	stimmt nicht	neutral	stimmt	stimmt genau
12T	Der Bewertungsmechanismus beim Tetris-Design war leicht verständlich.					
13K	Der Bewertungsmechanismus beim Kreis&Balken -Design war leicht verständlich.					
14P	Der Bewertungsmechanismus beim Pyramiden-Design war leicht verständlich.					
I2	Welche Bewertungsmechanismen fänden Sie attraktiv und für Ihren Arbeitsbereich geeignet? (Zeit und Fehler, Leaderbords…)					
15T	Eine Gamification mit dem Tetris-Design würde mich beim Arbeiten motivieren.					
16K	Eine Gamification mit dem Kreis&Balken-Design würde mich beim Arbeiten motivieren.					
17P	Eine Gamification mit dem Pyramiden-Design würde mich beim Arbeiten motivieren.					
I3	**Welche Form von Gamification könnte Sie beim Arbeiten motivieren?**					
	Nach dem Experiment: Akzeptanz	stimmt überhaupt nicht	stimmt nicht	neutral	stimmt	stimmt genau
18T	Den Einsatz der Tetris Gamification würde ich Kollegen empfehlen.					
19K	Den Einsatz der Kreis&Balken-Gamification würde ich Kollegen empfehlen.					
20P	Den Einsatz der Pyramiden-Gamification würde ich Kollegen empfehlen.					
I4	**Wie müsste Gamification aussehen, damit Sie sie Kollegen zum Einsatz empfehlen würden?**					
21T	Die Gamification mit Tetris ist zu simpel für den Einsatz in Produktionsbereich.					
22K	Die Gamification mit Kreis&Balken ist zu simpel für den Einsatz in Produktionsbereich.					
23P	Die Gamification mit Pyramiden ist zu simpel für den Einsatz in Produktionsbereich.					
I5	**Welche Komplexität der Gamification ist der Arbeit in der Produktion angemessen?**					
24T	Die Gamification mit Tetris würde mich zu stark von der Arbeit ablenken.					
25K	Die Gamification mit Kreis&Balken würde mich zu stark von der Arbeit ablenken.					
26P	Die Gamification mit Pyramiden würde mich zu stark von der Arbeit ablenken.					
I6	**Bei welche Form der Gamification halten Sie den Grad der Ablenkung von der Arbeit für vertretbar?**					

Abbildung A.2.: Fragebogen Teil 2

	Nach dem Experiment: Haltung	stimmt überhaupt nicht	stimmt nicht	neutral	stimmt	stimmt genau
27A	Spielerische Elemente in Arbeitsprozesse einzubinden ist eine gute Idee.					
I7	**Hat sich Ihre Ansicht während der Diskussion über die Design-Vorschläge geändert? Warum?**					
28A	Spielerische Elemente in die Montage bei Audi einzubinden ist eine gute Idee.					
I8	**Wo sehen Sie Ansatzpunkte, wo Hemmnisse?**					
29A	Ich könnte mir vorstellen, Gamification zu nutzen, wenn es ein passendes Design geben würde.					

Abbildung A.3.: Fragebogen Teil 3

Quellen

[AAUJ13] A. Amriani, A. Aji, A. Utomo, K. Junus. An empirical study of gamification impact on e-Learning environment. In *Computer Science and Network Technology (ICCSNT), 2013 3rd International Conference on*. 2013.

[ABB12] D. A. Boone, H. N. J. Boone. Analyzing Likert Data. 2012.

[BHWA95] S. Byström, C. Hall, T. Welander, K. A. Clinical Disorders and Pressure-Pain Threshold of the Forearm and Hand among Automobile Assembly Line Workers. 1995.

[Bib16] Bibliographisches Institut GmbH. Wörterbuch: -fizieren. http://www.duden.de/node/819099/revisions/1617941/view, 2016.

[Bor05] J. Bortz. *Statistik: Für Human- und Sozialwissenschaftler*. Springer, 2005.

[CCD06] M. Claypool, K. Claypool, F. Damaa. The Effects of Frame Rate and Resolution on Users Playing First Person Shooter Games. 2006.

[Coo84] C. A. Coonradt. *The Game of Work: How to Enjoy Work As Much As Play*. Game of Work, 1984.

[Csi95] M. Csikszentmihalyi. *Flow Das Geheimnis des Gluecks*. Klett Cotta, 1995.

[DKND11] S. Deterding, R. Khaled, L. Nacke, D. Dixon. Gamification: Toward a Definition. 2011.

[DOS+11] S. Deterding, K. O'Hara, M. Sicart, D. Dixon, L. Nacke. Gamification: Using Game Design Elements in Non - Gaming Contexts. 2011.

[FFRS14a] M. Fuchs, S. Fizek, P. Ruffino, N. Schrape. *Exploring the endgame of gamification*. meson press by Hybrid Publishing Lab, 2014.

[FFRS14b] M. Fuchs, S. Fizek, P. Ruffino, N. Schrape. *Making points the point: towards a history of ideas of gamification*. meson press by Hybrid Publishing Lab, 2014.

[Ger09] H. E. Dr. phil. Gerr. *Einführung in die Pfadfinderpädagogik*. Grin, 2009.

[Goo16] Google Inc. Google Trends. https://www.google.de/trends/explore#q=gamification, 2016.

[Haa14] J. Haas. A History of the Unity Game Engine. 2014.

[HKS14] J. Hamari, J. Koivisto, H. Sarsa. Does Gamification Work? — A Literature Review of Empirical Studies on Gamification. 2014.

[HsB97] G. M. Hägg, J. Öster, S. Byström. Forearm muscular load and wrist angle among automobile assembly line workers in relation to symptoms. 1997.

[ISO10] ISO. ISO 9241-210:2010. 2010.

[JS07] V. Jusufi, M. Saitovic. How to motivate assembly line workers. 2007.

[jso16] json.org. JSON Documentation. http://www.json.org/json-de.html, 2016.

[Kar07] I. Karenovics. Fallende Ost-Blöcke. Tetris oder Wie die Sowjetunion den Game Boy zum Superstar machte. 2007.

[KFS14] O. Korn, M. Funk, A. Schmidt. Assisitive Augmentation at the Manual Assembly Workplace using In-Situ Projection. 2014.

[KFS15a] O. Korn, M. Funk, A. Schmidt. Design Approaches for the Gamification of Production Environments. A Study Focusing on Acceptance. 2015.

[KFS15b] O. Korn, M. Funk, A. Schmidt. Towards a Gamification of Industrial Production. A Comparative Study in Sheltered Work Environments. 2015.

[Klu11] S. J. Kluge. *Methodik zur fähigkeitsbasierten Planung modularer Montagesysteme.* Dissertation, Universität Stuttgart, 2011.

[KO12] M. Koch, F. Ott. Gamification - Steigerung der Nutzungsmotivation durch Spielkonzepte. 2012.

[Kor14] O. Korn. Contextaware assistive systems for augmented work. A framework using gamification and projection. 2014.

[KSHK12] O. Korn, A. Schmidt, T. Hörz, D. Kaupp. Assistive system experiment designer ASED: A toolkit for the quantitative evaluation of enhanced assistive systems for impaired persons in production. 2012.

[KZa08] W. Kirsch, D. Zache, andere. *Schichtwechsel von der Kohlekrise zum Strukturwandel.* LWL, 2008.

[McC84] D. C. McClelland. *Human motivation.* Cambridge University Press, 1984.

[Mic16a] Microsoft. Kinect for Xbox One. http://www.xbox.com/de-DE/xbox-one/accessories/kinect-for-xbox-one, 2016.

[Mic16b] Microsoft. .NET Framework Documentation. https://msdn.microsoft.com/de-de/library/system.windows.forms.screen.allscreens%28v=vs.110%29.aspx, 2016.

[mot16a] motionEAP. Industrielle Nachfrage übersteigt Erwartungen. http://www.motioneap.de/tag/gips-schuele-preis/, 2016.

[mot16b] motionEAP. motionEAP. http://www.motioneap.de/, 2016.

[MPSB87] D. C. McClelland, V. Patel, D. Stier, D. Brown. The relationship of affiliative arousal to dopamine release. 1987.

[Nel12] M. J. Nelson. Soviet and American Precursors to the Gamification of Work. 2012.

[Oli14] Oliver Korn. WerkstättenMesse: Nominierung für den Preis exzellent:Kooperation. http://www.motioneap.de/nominierung-preis-exzellent-kooperation/, 2014.

[Pos80] J. Postel. User Datagram Protocol. 1980.

[PVM]

[SKH12] A. Schmidt, O. Korn, T. Hörz. Assistive Systems in Production Environments: Exploring Motion Recognition and Gamification. 2012.

[Uni16a] Unity Technologies. Unity-Documentation. http://docs.unity3d.com/Manual/index.html, 2016.

[Uni16b] Unity Technologies. Unity-Documentation-Player.Prefs. http://docs.unity3d.com/ScriptReference/PlayerPrefs.html, 2016.

[WD15] S. Walz, S. Deterding. *Gamification and Post-Fordist Capitalism.* MIT Press, 2015.

[WG52] C. R. Walker, R. H. Guest. The man on the assembly line. 1952.

[ZC11] G. Zicherman, C. Cunningham. *Gamification by Design. Emplementing Game Mechanics in Web and Mobile Apps.* O'Reilly, 2011.

Alle URLs wurden zuletzt am 27. 05. 2016 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift