technische universität
dortmund

Endbericht

PG 579: CyberReha - Ein immersives Videospiel

WS 2013/2014 – SS 2014

27. November 2014

Betreuer:
Prof. Dr. Heiko Krumm
Dipl.-Inf. Oliver Dohndorf

Lehrstuhl Informatik IV
Technische Universität Dortmund

fakultät für
informatik

Inhaltsverzeichnis

1 Vorwort

Dieser Bericht dokumentiert die Arbeit und die Ergebnisse der Projektgrupppe 579 „CyberReha“, welche im folgenden mit PG579 oder „CyberReha“ bezeichnet wird.

Projektgruppen werden seit 1972 an der Universität Dortmund (mittlerweile TU Dortmund) durchgeführt und sind in den Diplom- und Masterstudiengängen Informatik fest als Wahlpflichtveranstaltung vorgesehen. Projektgruppen (oder kurz „PGs“) sind üblicherweise Gruppen von 12 Studierenden und zwei bis drei Betreuern. Unabhängig vom konkreten Thema sollen PGs den Studierenden einen Einblick in übliche Prozesse und Strukturen geben, die sie im späteren Berufsleben erwarten können. Zudem sollen sie lernen, selbst organisiert über einen längeren Zeitraum an einem größeren Softwareprojekt zu arbeiten.

Die Verfasser dieses Endberichts glauben, dass diese Ziele im Rahmen der PG CyberReha voll und ganz erreicht wurden und bedanken sich für die angenehme und lockere Betreuung bei den beiden Betreuern Oliver Dohndorf und Prof. Heiko Krumm.

2 Einleitung

2.1 Übergeordnete Ziele und Aufbau der PG

Ziele der Projektgruppenarbeit sind die Vertiefung fachlicher Kenntnisse und das Einüben nichtfachlicher Kompetenzen wie Teamarbeit, Präsentationstechniken und Präsentation von Ergebnissen vor Publikum, Projektmanagement, das Einarbeiten in neue Problemkreise und die selbstständige Erarbeitung und Bewältigung umfangreicher Aufgaben. Außerdem soll in Vorbereitung auf Diplom- oder Masterarbeit das längere eigenständige Arbeiten an einem aktuellen wissenschaftlichen Thema trainiert werden.

Die PG CyberReha besteht aus 12 Studierenden und den beiden Betreuern Oliver Dohndorf und Prof. Heiko Krumm. Bei der Auswahl der Teilnehmer wurde berücksichtigt, dass die Studierenden die formalen und inhaltlichen Voraussetzungen erfüllten, um produktiv an der Projektgruppe teilnehmen zu können.

Noch vor Beginn der eigentlichen PG im Wintersemester 2013/2014 trafen sich die potenziellen Teilnehmer mit den Betreuern und Jan Dirk Hoffmann, einem Mitar-

beiter des Kooperationspartners, der Schüchtermann-Klinik, auf die im Folgenden noch näher eingegangen wird. Bei diesem Treffen wurde das Projektvorhaben grob skizziert und der Kooperationspartner vorgestellt. Nachdem die Zusammensetzung der PG abgeschlossen war, wurden unter den Teilnehmern 12 Themen verteilt, die fachlich und inhaltlich zu den Themen der PG passten. Jeder Teilnehmer bereitete im Rahmen dieser sogenannten „Seminarphase“ einen ca. 30-minütigen Vortrag, sowie eine Ausarbeitung im Umfang von ca. 15 Seiten vor. Diese wurden bei den ersten Treffen der PG der gesamten Gruppe vorgestellt, woraufhin diese die Möglichkeit hatte, Kritik und Feedback zu geben. Anschließend gab es parallel zum normalen Betrieb der PG eine sogenannte „Praktikumsphase“, in der zu den wichtigsten Themen von den jeweiligen Experten kleine praktische Aufgaben gestellt wurden, welche von den anderen Gruppenmitgliedern bearbeitet wurden.

In den anschließenden Treffen folgte eine Gesamtplanung des Projekts. Die wesentlichen Komponenten und Schnittstellen wurden identifiziert und analysiert. Daraufhin wurden Arbeitspakete erstellt, welche auf kleinere Gruppen verteilt wurden, die diese weitgehend eigenständig mit regelmäßiger Berichterstattung an die gesamte Gruppe bearbeiten sollten. Während der Planungsphase gab es zwei wöchentliche Treffen mit allen Teilnehmern. In der Entwicklungsphase gab es in Absprache mit den Betreuern ein teilgruppenübergreifendes Treffen, bei dem die Fortschritte und Probleme vorgestellt und diskutiert wurden. Die Betreuer sprachen Ziele und Probleme gezielt an und achteten auf eine faire Aufgabenverteilung. Es wurde auf eine konsequente Verteilung der Aufgaben und Verantwortlichkeiten innerhalb der Gruppe geachtet und beachtet, dass alle Teilnehmer nach Möglichkeit in allen auftretenden Tätigkeitsprofilen (z.B. Programmierung, Dokumentation, Berichtserstellung, Arbeitsorganisation) herangezogen wurden. Insbesondere wurden Teilnehmer mit nützlichen Vorkenntnissen ausgemacht und entsprechend eingebunden.

Parallel zur Planungsphase gab es Mitte November 2013 ein Treffen beim Kooperationspartner in Bad Rothenfelde, bei dem kurz das Projektvorhaben skizziert wurde und die Vorgänger-PG ihre abschließenden Ergebnisse vorstellte.

Gegen Ende des ersten Semesters wurde ein Pflichtenheft erstellt, welches als formale Spezifikation des Projekts betrachtet werden kann und als Zwischenbericht der PG dient.

Die Ergebnisse der PG, die Darstellung des bearbeiteten Problems, der zeitliche

Aufwand und die eingeschlagenen Lösungswege sind in diesem Abschlussbericht zusammengefasst.

2.2 Das Projekt „CyberReha“

2.2.1 Motivation

Herz-Kreislauf-Erkrankungen sind nach wie vor die häufigste natürliche Todesursache in Deutschland und treten gerade bei älteren Menschen häufig auf. Die Schüchtermann-Schiller'sche Kliniken (SSK) in Bad Rothenfelde haben sich auf die Behandlung und insbesondere die Rehabilitation von kardiologischen Erkrankungen spezialisiert und im Bereich der kardiologischen Rehabilitation in der Vergangenheit bereits mit Projektgruppen der Universität Dortmund bzw. TU Dortmund mit großem Erfolg kooperiert: die Informatik spielt im Bereich der Rehabilitation eine immer größere Rolle, da mit aktueller Hardware, maßgeschneiderter Software und guter Absprache mit fachlichen Ansprechpartnern aus der Medizin deutliche Fortschritte im Bereich der medizinischen Rehabilitation erreicht werden können.

Ein konkretes, häufig auftretendes Problem bei Rehabilitationsmaßnahmen (beispielsweise Training nach einem Herzinfarkt) ist mangelnde Motivation, über einen längeren Zeitraum sportliches Training auszuüben. Gerade dieses Problem sollte im Rahmen der PG CyberReha mit verschiedenen neuartigen Ansätzen gelöst werden. Der vielleicht wichtigste Ansatz ist die sogenannte „Gameification“, also der Einzug digitaler Spiele in alle Lebensbereiche abseits des klassischen Zeitvertreibs. Spätestens die Einführung der Spielekonsole Nintendo Wii hat gezeigt, dass die multimediale Kombination von Interaktivität, Virtualität und Realität nicht nur von jüngeren Menschen angenommen wird. Insbesondere führt ein hohes Maß von Immersivität bei den Spielern (in diesem Zusammenhang Patienten auch gleichzeitig Spieler) zu gesteigerter Motivation und infolgedessen zu besserem Trainingseffekt.

2.2.2 Zielsetzung der PG

Aus medizinischer Sicht ist das Ergometer ein wichtiges Gerät für Rehabilitationsmaßnahmen, da es unter ärztlicher Beaufsichtigung eine kontrollierte Umgebung für sportliches Training erlaubt, dass die allermeisten Herz-Patienten absolvieren können. Aus Sicht der Informatik handelt es sich um eine wohldefinierte sensorbasierte Spieleumgebung, da der Patient zusätzlich mit Sensoren ausgestattet werden kann, die beispielsweise Herzfrequenz oder Atemfrequenz messen. Unter Hinzunahme einer

3D-Brille entsteht ein sogenanntes *Cyber-Physical-System* (CPS), also einer Vernetzung diverser verteilter, eingebetteter Komponenten, die miteinander kommunizieren. Typischerweise erlaubt die Auswertung der physischen Komponenten (beispielsweise der Sensoren) eine Vermischung mit der virtuellen Welt, da Veränderungen in der physischen Welt durch entsprechende Regelungssysteme in der virtuellen Welt abgebildet werden können.

Konkret sollte daher ein CPS entwickelt werden, welches das Training im Sinne einer kardiologischen Rehabilitationsmaßnahme mit Methoden der Gameification unterstützt. Es sollten Ideen und Konzepte entwickelt werden, wie sich die diversen Komponenten des CPS miteinander integrieren lassen und wie insbesondere ein immersives Videospiel mit entsprechenden Regelungsmechanismen aussehen könnte. Weiterhin spielt gerade im Hinblick auf die Gameification die soziale Dimension eine große Rolle: gemeinsames Training fällt vielen Patienten leichter als das Einzeltraining.

Die Minimalziele der PG lauten gemäß PG-Absichtserklärung und PG-Zwischenbericht bzw. Pflichtenheft wie folgt:

1. Konzeptionierung und Umsetzung einer Spiele-Idee, die ein sinnvolles, medizinisches Training realisiert.
2. Entwurf und Entwicklung eines sensorbasierten CPS. Dazu müssen die Sensoren des Ergometers, der Sensor zur Messung der Kopfbewegung in der 3D-Brille (Oculus Rift) und mindestens ein medizinischer Sensor integriert werden.
3. Entwicklung einer virtuellen Realität als Videospiel. Austausch und Verarbeitung aus dieser virtuellen Welt mit der realen physischen Umgebung.
4. Das gesamte System wird in die bestehende soziale Community „RehaWeb“ der gleichnamigen Vorgänger-PG integriert, d.h. das Anlegen von Trainings kann durch den Arzt oder den Patienten in dieser Plattform erfolgen und die eigentliche Spielkomponente tauscht mit dieser Plattform Daten zum Training aus.
5. Das eigentliche Training im Spiel wird durch sogenannte „Minigames“, also kleinere Spieleinheiten, die frei wählbar sind, realisiert.
6. Die Steuerung erfolgt durch den Bewegungssensor der 3D-Brille Oculus Rift und die Bedienelemente am modifizierten Ergometer.

Darüberhinaus gibt es gemäß Pflichtenheft die folgenden Wunschziele:

1. Mehrspieler-Trainings sollen per Verabredung und mit Kommunikation möglich sein.

2. Ein Erfahrungspunktesystem, wie man es aus vielen modernen Videospielen kennt, soll zusätzliche Trainingsanreize bieten.

3. Verschiedene Bodenbeläge in der virtuellen Welt wirken sich zusätzlich auf die Belastung am Ergometer in der physischen Welt aus.

2.3 Organisation

2.3.1 Regelmäßige Abläufe

Die PG CyberReha traf sich während der Planungsphase zweimal wöchentlich und während der Implementierungsphase einmal wöchentlich mit den Betreuern. Wie bereits erwähnt wurde regelmäßig der Fortschritt präsentiert und zudem auf auftretende Probleme eingegangen. Oft wurden bei diesen Problemen auch gemeinsam Entscheidungen getroffen.
Im Kern gab es fünf Arbeitsgruppen, in die sich die Teilnehmer recht früh unterteilt haben:

- Minigame „Flugspiel“: Ein Minigame, bei dem der Spieler ein Flugzeug in den Wolken steuert und Powerups sammelt.

- Minigame „Parcours“: Ein Minigame, bei dem der Spieler eine Art Surfer steuert, der auf verschiedenen Oberflächen fährt, Hindernissen ausweichen und weitere Geschicklichkeitshürden überwinden muss.

- Ergometer: Modifikation und Ansteuerung des umgebauten Ergometers.

- RehaWeb: Integration des gesamten Projekts mit der sozialen Community, die aus einer früheren PG entstanden ist.

- Menü/Lobby/Oculus: Integration und Ansteuerung der 3D-Brille, sowie Gestaltung eines Menüs und Koordination der Mehrspieler-Aspekte.

Zusätzlich zu den verpflichtenden regelmäßigen Treffen, gab es in den Einzelgruppen gesonderte Treffen, sofern Bedarf bestand. Diese Treffen wurden selbstständig organisiert und geleitet.

Gegen Ende des ersten Semesters wurde ein Pflichtenheft erstellt, welches die formale Spezifikation des Projekts darstellt und die geplanten Abläufe dokumentiert.
Die Kommunikation innerhalb der Gruppen erfolgte per E-Mail, Telefon und Instant-Messaging. Wesentliches Organisations-Tool war das Projektverwaltungs- und Ticketing-System RedMine.

2.3.2 Erstes Semester (Planungsphase)

Bereits vor Beginn des ersten Semesters wurden an die Teilnehmer Themen verteilt, zu denen diese Vorträge und Ausarbeitungen verfassten. In den Vorträgen von ca. 30 Minuten wurden die anderen Teilnehmer über diese Themen unterrichtet. Im Anschluss an die Vorträge gab es zudem Zeit für Fragen und Feedback. Die vorgestellten Themen wurden in den Ausarbeitungen von ca. 15 Seiten noch einmal vertieft. Die Themen der Seminarphase lauteten wie folgt:

1. AAL: eHealth-Systeme
2. Training bei Herzpatienten
3. Sensorbasierte-Spieleumgebungen in der Medizin
4. Gamification
5. Game-Engines
6. Webservices und DPWS
7. RESTful services
8. Wireless Personal Area Networks
9. Die .NET-Plattform
10. Cyber-Physical-Systems
11. Grundzüge der Mess- und Regelungstechnik
12. Projektverwaltung – Vorgehen und Hilfsmittel

Eine Zusammenfassung der behandelten Themen der *Seminarphase* in Form von Abstracts findet sich im Folgenden:

- AAL: eHealth-Systeme:
  In dieser Ausarbeitung werden Bestandteile von Systemen zur Gesundheitsüberwachung näher betrachtet. Diese Bestandteile betreffen Hardwaresensoren sowie Softwareplattformen. Als Einstieg in dieses Themengebiet werden unterschiedliche Arten von Gesundheitssystemen kurz erläutert. Für die Realisierung solcher Gesundheitssysteme bedient man sich spezieller Softwareplattformen wie z.B. das OSGi oder die Microsoft Windows Mobile Plattform. Im dritten Abschnitt werden Sensoren zur Aufnahme sowie Uberwachung von biometrischen Daten eines Patienten näher betrachtet. Dazu gehören EKG, Beschleunigungsmesser aber auch Blutdruckmesser. Am Ende werden notwendige Eigenschaften der Datensicherheit von erfassten Patienten-Daten anhand eines Vergleiches von Frameworks dargestellt.

- Training bei Herzpatienten:
  Diese Arbeit beschreibt im ersten Teil den grundlegenden Aufbau des Herzkreislaufsystems, sowie Herzerkrankungen. Dabei wurden die Risikofaktoren von Herzerkrankungen wie Diabetes, Bluthochdruck, Fehl -und Überernährung, Rauchen, Bewegungsmangel und Stress näher betrachtet und festgestellt, dass körperliche Inaktivität der wichtigste Risikofaktor ist. Im zweiten Teil wird auf verschiedene Trainingsmöglichkeiten, insbesondere auf das aerobe Ausdauertraining, eingegangen.

- Sensorbasierte-Spieleumgebungen in der Medizin:
  In dieser Arbeit werden Ansätze vorgestellt, wie sensorbasierte Spieleumgebungen zu medizinischen Zwecken eingesetzt werden können. Anwendungen können beispielsweise erfolgen, um Reaktionszeiten zu verringern, den Bewegungsapparat zu verbessern, und insgesamt die körperliche Sicherheit und Stabilität zu erhöhen. Die Anfertigung dieser Arbeit erfolgte in der einleitenden Seminarphase der PG 578 (CyberReha) an der TU Dortmund.

- Gamification:
  Gamification ist ein Prozess aus spielbasiertem Denken und Spielmechaniken ist, welcher Benutzer anspricht und Probleme löst. Dieser Prozess soll dabei in eine Anwendung eingebettet sein, so dass der Benutzer angesprochen wird, diese Anwendung zu verwenden. Um die verschiedenen Motivationsgründe für Benutzer besser zu Analysieren werden in dieser Ausarbeitung zuerst die Benutzer charakterisiert und in spezielle Gruppen eingeteilt. Mit Hilfe dieser Charakterisierung wird im Folgenden darauf eingehen, worauf die einzelnen Gruppen von

Spielern besonderen Wert legen und wie man sie mit einer Anwendung ansprechen kann. Des Weiteren wird darauf eingegangen, welche Faktoren besonders motivierend für die einzelnen Gruppen von Benutzern sind und beschreiben, wie man durch gezielten Einsatz dieser Faktoren einen Benutzer dazu anregt, die Anwendung auch über einen längeren Zeitraum zu benutzen.

- Game-Engines:
  Sowohl Rehabilitationsmaßnahmen (Training nach Herzinfarkt) als auch rein sportliche Motivationen (radspezifisches Training im Winter) werden häufig auf einem häuslichen Ergometer durchgeführt. Die Erschaffung einer virtuellen Realität zur spielebasierten Förderung des Fitnesslevels ist in diesem Umfeld also naheliegend. In dieser Ausarbeitung soll ein Analysewerkzeug geschaffen werden, das einen Anhaltspunkt bei der Auswahl diverser Game Engines bietet. Es wird auf mehrere Faktoren eingegangen, die sowohl Nutzererfahrung als auch Entwicklerbeteiligung und Zeitaufwand bewerten. Mithilfe dieses Werkzeugs kann die Entscheidung auf zwei Kandidaten heruntergebrochen werden, die verschiedene Merkmale besonders gut beherrschen. Eine endgültige Auswahl kann im Hinblick auf die weitere Ausrichtung des Projekts und die Erfahrung der Entwickler getroffen werden.

- Webservices und DPWS:
  In der letzten Zeit ist die Entwicklung von automatisierten, autonomen und selbst-konfigurierten verteilten Systemen, die früher durch den Benutzer gesteuert wurden, zu beobachten. Web Services sind Bestandteil solcher Entwicklung, aber es fehlt die sichere Integration von Web Services in die ressourcenbeschränkten Geräte (Devices). Im Folgenden werden die Technologien rund um die Web Services beschrieben (SOA, SOAP, WSDL, UDDI, Sicherheit) und es wird auf die Technologien rund um die DPWS eingegangen.

- RESTful services:
  Die vorliegende Ausarbeitung soll eine Einführung in den Bereich der RESTful Web Services darstellen. Auch wenn die Prinzipien von REST jedem, der den Umgang mit dem World Wide Web gewohnt ist, bereits begegnet sind, fällt es in der Praxis doch immer wieder auf, dass das Entwickeln von RESTful Web Services eine große Herausforderung darstellt, da viele Prinzipien unzureichend oder gar nicht verstanden sind. Insbesondere Entwickler, die Erfahrungen in der objektorientieren Programmierung haben, tun sich oftmals schwer damit, die ressourcenorientierte Sicht, die REST erfordert, einzunehmen und empfin-

den SOAP daher oftmals als intuitiver.
Diese Ausarbeitung beschäftigt sich daher mit den vier grundlegenden Prinzipien, die den Architekturstil REST ausmachen: die Ressourcenorientierung, die Wahl einer festen Menge von Operationen, die Zustandslosigkeit des Web Service und das Prinzip der Ressourcenrepräsentation. Im Hinblick auf die Praxis werden außerdem ausgewählte Teile des HTTP Standards vorgestellt und ihr Bezug zu REST erläutert.
Abschließend wird ein Blick auf Richardsons Maturity Model geworfen, ein Modell zur Klassifizierung von Web Services, im Hinblick darauf, wie genau sie dem REST Standard folgen. Dies soll weitere Einsichten in das Konzept von REST liefern.

- Wireless Personal Area Networks:
  Diese Arbeit bietet eine kleine Übersicht über Drahtlos-Netzwerke für den Nahbereich - Wireless Personal Area Networks (WPAN's). Paar ausgewählte standardisierte Verfahren werden genauer präsentiert, anhand des ISO/OSI-Referenzmodells beschrieben und kurz miteinander verglichen. Zu diesen Verfahren zählen Bluetooth, ZigBee sowie Wi-Fi. Proprietäre Verfahren wurden nicht mit betrachtet.

- Die .NET-Plattform:
  .NET (ausgesprochen „dot net") ist ein Sammelbegriff für einen von Microsoft entwickelten Technologiestack zur Erstellung und Ausführung von Desktop- und Webanwendungen. Da .NET viele Ansätze, Technologien, Plattformen und Programmiersprachen vereint, wird es häufig selbst als Plattform oder Framework bezeichnet. Wir betrachten die Kerntechnologien und die wichtigsten Komponenten von .NET und identifizieren die grundlegenden Ziele, die Microsoft mit der Entwicklung des Frameworks verfolgt hat. Wegen der massiven Größe der Plattform beschränken wir uns dabei auf die essentiellen Aspekte und wollen abschließend noch kurze Programmierbeispiele geben und diskutieren, inwieweit .NET für die Projektgruppe CyberReha relevant ist.

- Cyber-Physical-Systems:
  Unsere heutige Technologie erlaubt uns Interaktionen miteinander und mit der Welt um uns herum, die vor fünfzig Jahren kaum denkbar gewesen wären. Genauso lässt sich vermuten, dass wir uns heutzutage kaum vorstellen können, was die Technologie uns in den nächsten fünfzig Jahren erlauben wird. Heute befinden wir uns in einer Zeit des schnellen technologischen Wandels und erleben in kurzen Abständen computerwissenschaftliche Revolutionen. Cyber-

Physical Systems sind eine davon: Unter diesem Begriff versteht man Computersysteme, die zugleich mit der physischen Umwelt als auch mit anderen (möglicherweise weltweit verstreuten) Computersystemen interagieren. Was zunächst nicht besonders revolutionär oder interessant klingen mag, eröffnet bei näherer Betrachtung eine immense Bandbreite an Anwendungsmöglichkeiten, die uns erlauben, Ziele zu erreichen, die wir anders zu erfüllen vermutlich nicht in der Lage wären. Praktisch jede industrielle oder wissenschaftliche Branche findet in dieser zukunftsweisenden Technologie ein Mittel zur Steigerung der Effizienz, Zuverlässigkeit oder Sicherheit ihrer Systeme.
Die nachfolgende Arbeit soll zunächst dieses computerwissenschaftliche Gebiet in seinen Grundzügen und Anwendungsmöglichkeiten vorstellen. Sowohl ein grober Überblick über die Anwendungsmöglichkeiten als auch die Präsentation einiger konkreter Anwendungsfälle wird erfolgen. Da es sich bei Cyber-Physical Systems um eine recht junge wissenschaftliche Disziplin handelt, gibt es noch einen großen Bedarf an Formalismen, Standardisierungen und Verfahren, die einen souveränen und routinierten Umgang damit ermöglichen würden, wie in anderen etablierten Bereichen der Informatik auch. Welche Probleme momentan im Umgang mit Cyber-Physical Systems bestehen, soll somit ein weiteres Thema dieser Arbeit sein. Doch nicht nur die negativen Aspekte in der Gegenwart der Cyber-Physical Systems sollen auf den folgenden Seiten beleuchtet werden: Den Abschluss dieser Arbeit bildet ein Ausblick über die große Bedeutung und die zentrale Stellung dieser Systemgattung in unserer technologischen, wirtschaftlichen und gesellschaftlichen Zukunft.

- Grundzüge der Mess- und Regelungstechnik:
  In allen möglichen Zweigen der Industrie und Technik sind verschiedenste physikalische Größen zu messen, um mithilfe dieser ein technisches System auf eine bestimmte Weise zu regeln. Während die Zielsetzung der Messtechnik die Gewinnung von Informationen über den Zustand eines Systems oder Problems ist, beschäftigt sich die Regelungstechnik mit der Steuerung und Regelung von technischen Größen und Stoffen, um den Zustand eines Systems in gewünschter Weise zu beeinflussen.
  Ziel dieser Arbeit ist es eine kurze Einführung in beide Themengebiete zu geben und die wichtigsten Grundaspekte zu erläutern. Im ersten Teil werden die Grundbegriffe des Messens, die Fundamentalvoraussetzungen, die Darstellung von Messsystemen und mögliche Fehlerquellen, die graphische Darstellung von Messbeobachtungen und die Charakterisierung von Messreihen definiert und

erläutert.
Der zweite Teil führt in die Themen der Regelungstechnik ein und beschreibt den Systembegriff, die wichtigsten Begriffe für geregelte Systeme und die einzelnen Komponenten am Beispiel eines Regelkreises.
Im letzten Teil wird kurz die Modellierungs- und Simulations-Software MATLAB/Simulink im Bezug auf dynamische Systeme vorgestellt.

- Projektverwaltung – Vorgehen und Hilfsmittel: *Eingehende Überlegungen über das Management jedes geplanten Projekts sind die Basis für eine erfolgreiche Durchführung. Allgemein stellen Projekte zeitlich befristete, komplexe und interdisziplinäre Aufgaben dar. Zur erfolgreichen Durchführung von Projekten müssen die einzelnen Teilaufgaben und der Personen- bzw. Ressourceneinsatz organisiert, geplant, gesteuert und kontrolliert werden. Die vorliegende Arbeit gibt einen Überblick über diese Funktionen und ihre Herausforderungen und stellt zudem einen Bezug zu aktuellen Technologien und Hilfsmitteln her. Somit wird die Basis geschaffen, um richtige strategische Entscheidungen zu treffen und das Projekt erfolgreich an das gewünschte Ziel zu bringen.*

Zu ausgewählten wichtigen Themen der Seminarphase wurden zudem in der sogenannten *Praktikumsphase* Aufgabenblätter erstellt, die dann alle Teilnehmer bearbeitet haben. Die Themen der Praktikumsphase waren

- .NET und C#,
- Ansteuerung des Ergometers und
- Webservices.

Gegen Ende des ersten Semesters wurde zudem ein erster, lauffähiger Prototyp erstellt und getestet. Dabei wurden insbesondere zwei schwerwiegende Probleme festgestellt:

1. Über den Unity Webplayer konnten die Treiber für die 3D-Brille Oculus Rift nicht geladen werden, da es sich um nativen Code handelte und
2. bei der Übermittlung der Sensordaten vom Ergometer zum 3D-Spiel gab es eine zu lange Latenz von mehreren Sekunden, welche das Spielerlebnis deutlich beeinträchtigte.

2.3.3 Zweites Semester (Implementierungsphase)

Nachdem in der ersten Iteration der Entwicklungs-Spirale im ersten Semester ein lauffähiger Prototyp erstellt wurde, galt es diesen im zweiten Semester zu einem vollständigen Produkt auszubauen. Das wichtigste zusätzliche Feature war dabei die Entwicklung eines Mehrspielermodus.
Zunächst wurden die wichtigsten Probleme behoben, die beim gemeinsamen Test gegen Ende des ersten Semesters festgestellt wurden. Dazu wurde erstens die Integration mit RehaWeb nicht mehr über den Unity Webplayer vorgenommen. Stattdessen wurde eine Java WebStart Lösung entwickelt, über die es möglich ist, nativen Code auszuführen, was wiederum das Laden der Oculus-Treiber wieder ermöglicht. Weitere Details zur Umsetzung der Problemlösung finden sich in den folgenden Abschnitten dieses Dokuments.
Zweitens wurden die (Latenz-)Probleme bei der Ansteuerung der Hardware mit einem Kalibrierungsmechanismus und einer grundsätzlich überarbeiteten Implementierung der Schnittstelle behoben.

Die Arbeitsgruppen in der zweiten Phase waren ähnlich aufgebaut wie schon im ersten Semester, zusätzlich gab es nach ungefähr der Hälfte der Zeit zudem eine Gruppe, die ein einfach zu bedienendes Hauptmenü entwickelt hat.
Parallel zur Arbeit am Kernprodukt wurde zudem in der Gruppe Struktur und Inhalt dieses Endberichts festgelegt. Anschließend haben alle Gruppen neben der Implementierung ihre Beiträge zu diesem Endbericht verfasst. Außerdem wurde beschlossen, ein wissenschaftliches Papier für die Einreichung bei einer Konferenz zu verfassen. Dazu musste insbesondere eine Evaluation des Systems mit Probanden durchgeführt werden. Diese erfolgte gegen Ende des Semesters an mehreren Terminen. Wesentliches Werkzeug zur Ermittlung der Ergebnisse war ein Fragebogen mit Pre- und Post-Anamnese, der von einer neu gebildeten Arbeitsgruppe in Kooperation mit der SSK entworfen wurde.
Den voraussichtlichen Abschluss der Implementierungsphase bilden ein Vortrag auf dem „European Congress on e-Cardiology & e-Health“ am 31. Oktober 2014 in Bern, sowie die Ergebnispräsentation bei einem Abschlusstreffen in der SSK. Das Abstract des Beitrags lautet wie folgt:
Gamification in Endurance Sports - Possible Use in Coronary Secondary Prevention to Increase Motivation and Compliance
Gamification describes the introduction of game elements into hitherto non-game environments like physical exercises with the intention of raising user engagement

and motivation. The cyber-physical system (CPS) „CyberReha“ offers a generic approach to a more entertaining cardio-training environment that may be implemented in hospitals and medical centres or at home. This architecture is based on the previously developed „RehaWeb“ platform and the Unity game engine. It is composed of a main application installed on a client's computer and an ergometer with a RJ45 output that is hooked up to the computer. The communication is bidirectional. The ergometer sends various inputs such as steering angle, speed and heart rate. The resistance of the pedals is set by the application. An optional but recommended component is an application server located in the cardiologist's office that receives any vital information of the training session and can be used to fine-tune the exercise parameters of each patient using the aggregated and graphically edited data to avoid over- and underexertions.
The system is evaluated in cooperation with the Schüchtermann-Schillersche' Kliniken in Bad Rothenfelde (Germany).
Test persons who underwent physical exercise with our system have shown high user acceptance, increased motivation and better overall compliance based on a questionnaire in comparison to test persons who exercised with only the traditional ergometer.
Our work strongly indicates positive effects of gamification on endurance training in heart patients.

3 Softwareumgebung

3.1 Unity Game-Engine

3.2 Einleitung

Unity stellt als Game-Engine zahlreiche Funktionen wie Terrain-Erschaffung, Physik- und Partikel-Simulation, Import aller gängigen 3D, 2D und Audio-Dateien und Input-Verwaltung zur Verfügung. Anfangs nur von Privatanwendern oder kleinen Entwicklerstudios genutzt, hat sich Unity zur *State of the Art* Game-Engine entwickelt. So unterstützt die Grafik-Pipeline DirectX 11 und OpenGL, viele Dateiformate anderer Entwicklungssoftware können problemlos importiert werden. Für Projekte mit geringem Umfang ist die Entwicklung kostenlos.
Unity ist dabei mehr als eine Code-Bibliothek für diverse Programmiersprachen. Mit Hilfe des Editors kann durch das Level navigiert und 3D Modelle, Sounds und Lichter platziert werden. Im Gegensatz zum *Unreal Development Kit* auf Basis der

Unreal Engine 3 wird der Editor aber nur für die Verwaltung der Spielumgebung eingesetzt. Ein Zusammenstellen der Spiellogik mittels Mausklick ist nicht möglich. Unity glänzt dafür bei der Unterstützung verschiedener Plattformen und der Benutzerfreundlichkeit. Mittels Script kann das Verhalten einen Objektes sogar während der Laufzeit verändert werden.
Im Vergleich mit anderen populären Entwicklungsumgebungen wie der *CryEngine* oder der *Unreal Engine* punktet Unity vor allem mit einer umfangreichen Dokumentation und einem großen Forum, in dem Entwickler Hilfestellungen geben und sich austauschen können.

3.3 Lizensierung

Für diese Ausarbeitung wurde Unity in der Basis-Version 4.5.2f1 verwendet. Die Einschränkungen zur kostenpflichtigen Variante sind:

- Kein LOD-Support.
  3D Modelle in unterschiedlichen Detailstufen (engl. ***l****evel* ***o****f* ***d****etail*) werden nicht unterstützt. Bei Spielen mit geringem Umfang ist dies allerdings nicht relevant.

- Render-to-Texture (z.B. Spiegelungen) ist nicht möglich.

- Realtime-Shadows sind nicht vorgesehen.
  Mit Sicherheit die stilistisch größte Einschränkung der kostenlosen Unity-Variante.

- Projekte können nur als PC, Mac oder Linux Standalone oder im WebPlayer veröffentlicht werden. Exporte auf iOS, Android, Flash, PlayStation3, XBox 360 und Wii U sind nicht möglich.

Weitere Punkte sind zum großen Teil kosmetischer Natur und wie die bereits genannten Einschränkungen für die Entwicklung des in dieser Ausarbeitung beschriebenen Spiels verzichtbar.

3.4 Das Benutzer-Interface

Der Unity-Editor aus Abbildung 1 ähnelt auf den ersten Blick den gängigen Programmen der 3D-Modellierung (Autodesk 3ds Max, Maya). So sticht zunächst das Scene-Fenster heraus. Dort findet die Zusammenstellung der einzelnen Objekte zu einer virtuellen Szene statt. Die Optik entspricht in weiten Teilen dem eigentlichen

Spiel. Darüber hinaus werden für den Designer wichtige Objekte angezeigt. Dazu gehören Gitternetzlinien des ausgewählten 3D Modells oder Symbole für Kameras oder Lichter, die für den Spieler später nicht sichtbar sein sollen. So ist eine weiße Lichtquelle, als Sonnensymbol hervorgehoben, deutlich erkennbar.
Um das Scene-Fenster sind mehrere Panels angeordnet, die Informationen über das Projekt, die Hierarchie und Eigenschaften der Objekte liefern. Die Funktion dieser wird im folgenden Abschnitt weiter deutlich. Mitten über dem Scene-Fenster befindet sich eine Besonderheit. Mit den Schaltflächen lässt sich die aktuelle Szene bzw. das Spiel starten, pausieren und wieder stoppen. Einzelne Levels lassen sich so bereits im Editor testen. Der Compilerdurchlauf ist so schnell, dass er weitgehend unbemerkt bleibt.
Das spätere Spiel kann über ein Build-Menü als .exe-Datei gespeichert werden. Ein separates *Runtime Environment* wird nicht benötigt.

3.5 Das Unity Vokabular

In den folgenden Unterabschnitten werden die wichtigsten Begriffe des Unity Sprachgebrauchs geklärt. Da diese Ausarbeitung sich an professionellen Spieleentwicklungen orientiert, werden Vokabeln nicht „eingedeutscht“, sondern im englischen Original verwendet.

3.5.1 Projects

Ein Project bezeichnet die höchste Ebene im Unity-Editor und spiegelt im Allgemeinen genau ein Spiel-Projekt wieder.
Es stellt eine Ordnerstruktur bereit, die alle benötigten Dateien enthält. Dabei gleicht das Project-Panel aus Abbildung 1 dem *Asset*-Ordner auf Betriebssystem-Ebene. Aktualisierte Dateien werden erkannt und ersetzen automatisch im Unity-Editor ihren Vorgänger.

3.5.2 Scenes

Eine Scene kann zumindest ansatzweise als Level interpretiert werden. Die meisten Spiele sind in Level aufgeteilt und so ist auch jedes Unity-Projekt in mehrere Scenes aufgeteilt.
Der Begriff Scene wird deshalb verwendet, da er etwas robuster als Level ist. Denn auch eine Eröffnungssequenz oder das Hauptmenü können separate Scenes in Unity

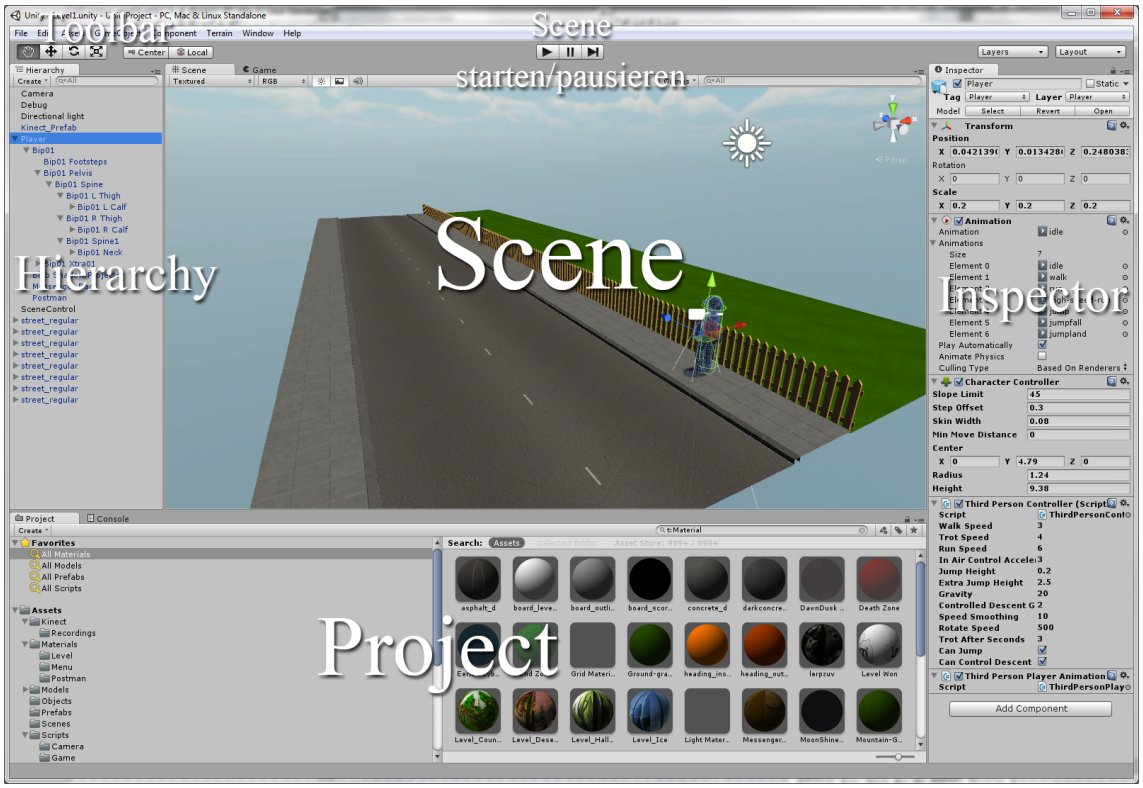

Abbildung 1: Der Unity-Editor in der Entwicklungsphase des Spiels. Hervorgehoben sind die Bezeichnungen der einzelnen Fenster. Das Scene-Fenster (mittig) zeigt eine Vorschau des Levels. In den umliegenden Fenstern werden die Einstellungen des Projekts und der Spielobjekte vorgenommen. Quelle: eigene Abbildung.

sein. Genauso können Scenes ineinander verschachtelt werden und zusammen ein Level formen.
Das Hierarchy-Panel in Abbildung 1 zeigt bei einer geöffneten Scene alle sich in dieser befindlichen Game Objects.

3.5.3 Game Objects

Game Objects sind das Fundament für jeden Gegenstand in einer Scene. Sie bezeichnen einen Transform-Knoten, also einen Punkt im Raum, der je nach Bedarf angepasst werden kann.
Jedes Game Object kann

- bewegt, rotiert und skaliert werden
- benannt werden
- einem *Tag* (Kennzeichnung) und einem *Layer* (Schicht) zugewiesen werden

- einen Hierarchie-Knoten bzw. Pivot-Punkt für andere Game Objects bilden
- durch Components definiert werden

3.5.4 Components

Ein Component ist ein modulares Stück von Funktionalität und Parametern, das definiert, was ein Game Object ausmacht.
Hierzu ein Beispiel: Eine Kugel (engl. Sphere), die man im Unity-Editor als ein Basis-Objekt erstellen kann, ist lediglich ein leeres Game Object mit zusätzlichen Components.
Der ursprüngliche Transform-Knoten wird in diesem Fall durch einen *Mesh-Filter* erweitert, der die einzelnen Positionen der Vertices der Kugel enthält. Ein *Mesh-Renderer* Component ist in der Lage, die Verticies zu einem dreidimensionalen Objekt zusammenzufügen und auf dem Bildschirm anzuzeigen. Desweiteren verfügt die Kugel über einen (optionalen) *Sphere Collider*, der durch den gegebenen Radius der Kugel einen Teil der Physik-Engine implementiert und auf Kollisionen überwachen kann.
Alle Components des ausgewählten Game Objects werden im Inspector angezeigt. In Abbildung 1 befinden sich diese am rechten Bildrand. Unter den Transform-Einstellungen des Game Objects sind Animation-, Character Controller- und zwei Script-Controller zu sehen.

3.5.5 Assets

Ein Asset ist ein Aspekt des Spiels, das von einem Component oder einem anderen Asset referenziert wird. Einige Beispiele sind in Tabelle 1 zu sehen. Es kann weiterhin zwischen externen und internen Assets unterschieden werden. Intern ist ein Asset, falls es in Unity erstellt und bearbeitet werden kann. In den meisten Fällen lässt sich ein internes Asset nicht für andere Entwicklungsoftware exportieren.
Im Gegenzug ist ein externes Asset außerhalb von Unity erstellt worden, es kann aber mittels Drag & Drop zu jedem Projekt hinzugefügt werden.
Wie bereits im Abschnitt über Projects erwähnt, befindet sich im zugehörigen Panel aus Abbildung 1 die Sammlung aller Assets im aktuell geöffneten Projekt.

3.5.6 Scripts

Unity-Scripts können wahlweise in Java-Script, C# oder Boo geschrieben werden. Unity stellt dazu einen Editor namens Mono-Develop bereit. Scripts, die das Inter-

Interne Assets	Externe Assets
Materials	3D Modelle
Shader	Texturen
Physic Materials	Soundeffekte
Prefabs	Scripts

Tabelle 1: Die Aufteilung der Assets in die Kategorien *Intern* und *Extern*.

face MonoBehavior implementieren, können mit einem Script Component an bestehende Game Objects angefügt werden. Durch diesen Schritt werden zur Laufzeit diverse Methoden (falls vorhanden) aufgerufen. Einige Beispiele finden sich in Tabelle 2.
In diesen Methoden kann nun auf die Spiellogik Einfluss genommen werden. Der Zugriff auf die Funktionalität der anderen Components oder auch anderen Game Objects und deren Manipulation kann ebenfalls erfolgen. In Abbildung 1 nimmt das Script *Third Person Player Animation*, mit einem Aufruf des *animation* Objektes, Änderungen an der Bewegung des Charakters vor.

Methode	Aufgerufen zum Zeitpunkt
Update()	einmal pro Frame
Start()	einmalig bevor der Aufruf von Update() zum ersten Mal erfolgt
OnCollisionEnter()	falls der Collider mit einem anderen Collider zusammengestoßen ist
OnBecameVisible()	sobald der Renderer für eine Kamera sichtbar wird
...	...

Tabelle 2: Eine Auswahl der Methoden des Interface *MonoBehavior* und ihr Aufruf.

3.5.7 Prefabs

Prefabs stellen Container dar, die Assets und deren Einstellungen bzw. Parameter zusammenfassen. Sie bilden ein Template aus dem dann mehrere Kopien dieser Gruppierung von Assets instanziiert werden können. Dies spart Zeit bei der häufigen Verwendung von ein und den selben Assets.
Man unterscheidet zwei Arten der Instanziierung.

3.5.7.1 Prefabs im Scene-Editor Verschiedene Game Objects und angehängte Scripts werden als Prefab zusammengefasst und mehrfach im Level wiederverwendet. Dies ist gängige Praxis für Level-Designer.
Zur Verdeutlichung stelle man sich als Beispiel eine Straßenlaterne vor. In Unity könnte diese aus mehreren Assets bestehen.

- 3d Modell(e) (möglicherweise sind Stange und Lampenkörper getrennt)
- Unity-Lichtquelle
- Sound-Effekt (Summen der Lampe)
- Partikel-Effekte, z.B. Motten die um das Licht kreisen

Das Level stellt eine urbane Gegend dar und es müssten möglicherweise hunderte Straßenlaternen auf die Umgebung verteilt werden. Ein Container dafür ist ein weitaus angenehmerer Weg, als alle Objekte einzeln zu anzuordnen.
Ein weiterer Vorteil ist, dass Änderungen am Prefab sofort auf die verschiedenen Kopien propagiert werden. So könnte beim Testen des Beispiel-Szenarios aufgefallen sein, dass die Straßenlaternen nicht hell genug sind oder das Licht einen größeren Gelbanteil haben sollte. Mit einem Prefab ist dies in wenigen Mausklicks für alle instanziierten Objekte im Level geändert.

3.5.7.2 Prefabs zur Laufzeit Als Programmierer wird man Prefabs zu schätzen wissen, denn sie können dynamisch zur Laufzeit instanziiert werden. Hierfür ein weiteres Beispiel:
In einem First-Person-Shooter soll ein Pistolenschuss implementiert werden. Dazu können wieder mehrere GameObjects in einem Prefab zusammengefasst werden:

- 3d Modelle der Patrone und Hülse
- ein angehängtes Script der Patrone, das die Positionsänderung vornimmt und Kollisionen überwacht
- Sound-Effekte, die beim Abfeuern und Auftreffen auf ein Hindernis abgespielt werden
- ein grelles Licht, das beim Abfeuern die Explosion im Pistolenlauf darstellt

All dies muss beim Starten des Levels nicht vorhanden sein, sondern kann mit Hilfe eines Prefabs zur Laufzeit instanziiert werden.

3.5.8 Packages

Ein Package ist ein Archiv-Dateiformat speziell für Unity. Es wird genutzt um Assets inklusive ihrer Ordnerstruktur und Metadaten zusammenzufassen und für andere Projekte zu exportieren. Bei dem Export-Vorgang werden bestehende Verbindungen und Abhängigkeiten zu anderen Assets analysiert und, falls benötigt, ebenfalls

eingefügt.
Es kann also davon ausgegangen werden, dass ein 3d Modell immer mit dem zugehörigen Material und Textur(en) exportiert wird.

3.6 Unity für Designer und Programmierer

Als Spiele-Entwickler kann man sich der Unity Entwicklungsumgebung auf zwei Arten nähern.
Design: Ohne jegliche Programmierkenntnisse können im Unity-Editor Spiel-Welten gestaltet werden. Berge, Bäume, Häuser und vieles mehr können auf dem Terrain platziert werden.
Dies macht zwar noch kein fertiges Spiel, da z.B. die Haustür noch nicht geöffnet werden kann, ist aber grundlegend und benötigt mindestens so viel Aufwand wie der Programmierarbeit.
Programmierung: Die nötige Logik, um das Spiel ins Leben zu rufen, muss programmiert werden. Nur so ist es möglich Gegenstände einzusammeln oder Highscores zu verwalten.

3.7 Unity Asset Store

Es ist sicherlich nützlich beide Aspekte - Design und Programmierung - zu beherrschen. Zwingend notwendig, um mit Unity Profit zu machen oder Spiele fertigzustellen ist dies allerdings nicht.
Im Asset Store können sowohl gestalterische Elemente als auch Scripts eingekauft oder verkauft werden. Unity behält 30% des Verkaufspreises als Provision.
Abbildung 2 zeigt die Startseite des Asset-Stores im Magazin-Stil. Aktuelle Angebote oder neue Assets werden mittig hervorgehoben. Rechts befindet sich die Navigation durch die verschiedenen Kategorien mit einer Chart-Liste der aktuell meistverkauften Assets.

3.8 CyberRehaWeb

3.8.1 Einleitung

RehaWeb ist eine soziale Community für (Herz-)Patienten. Den Benutzern stehen die typischen Bedienelemente sozialer Netzwerke zur Verfügung: Vernetzung, das Versenden von Nachrichten, Austausch in Foren und auf Pinnwänden, Freundschaftsbeziehungen, Statusupdates und diverse Weitere. RehaWeb verfügt zudem über ein

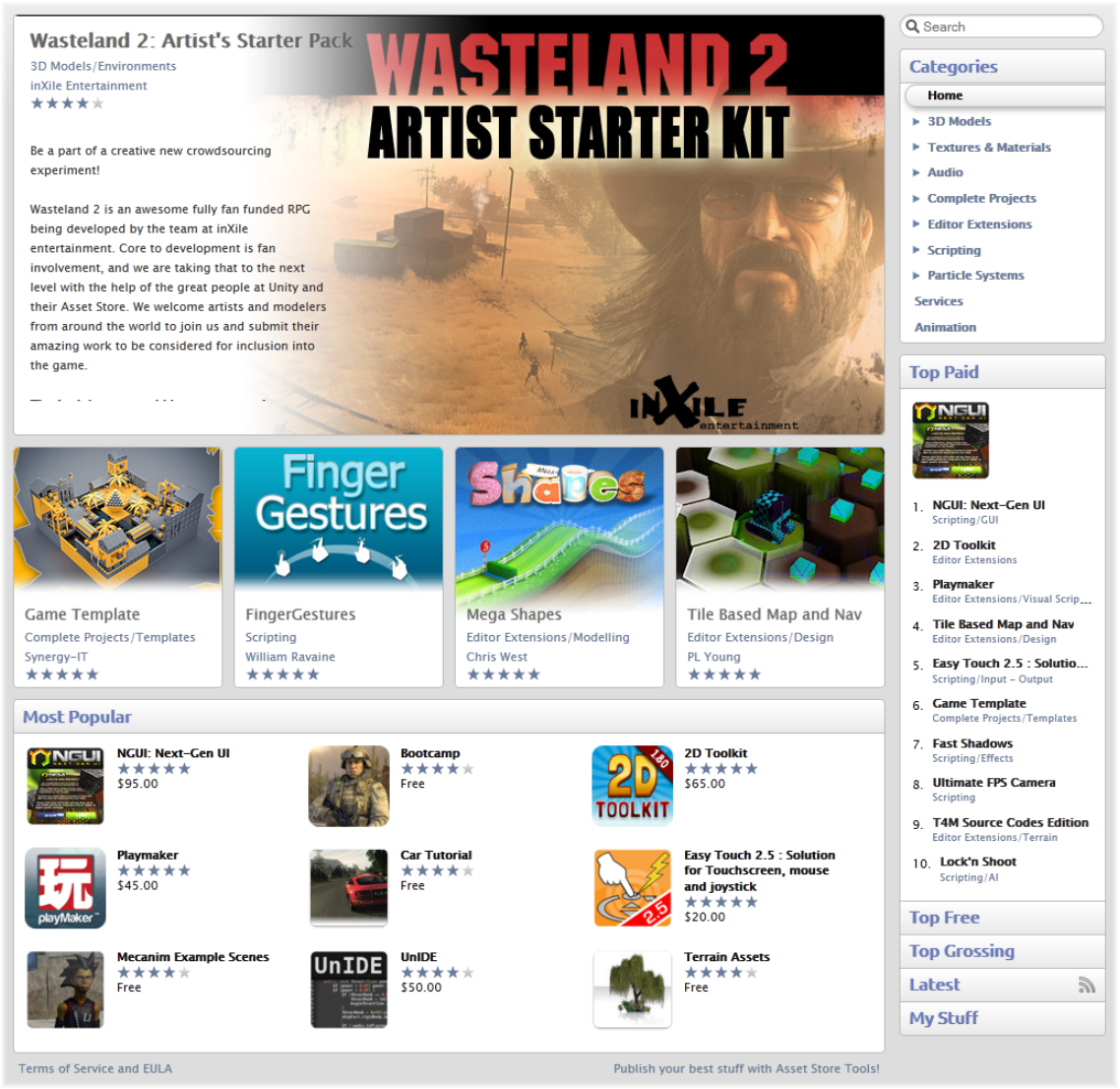

Abbildung 2: Der Asset Store mit aktuellen Angeboten und Verkaufs-Charts. Quelle: Unity Asset Store [**?**].

Rollensystem, das zwischen normalen Benutzern (Patienten), betreuendem Personal (Ärzte, Therapeuten) und Administratoren unterscheidet. So können Ärzte beispielsweise Vitaldaten und statistische Analysen über die Trainingseinheiten ihrer Patienten einsehen, während Administratoren Foren verwalten und neue Trainingskategorien verwalten können.

Der Funktionsumfang von RehaWeb ist nicht unbedeutend, daher wollen wir an dieser Stelle auf den Endbericht der RehaWeb-PG verweisen und uns im weiteren darauf beschränken, die Neuerungen im Zuge der PG CyberReha zu nennen.

Die um CyberReha erweiterte RehaWeb-Plattform haben wir zur Unterscheidung „CyberRehaWeb“ genannt. Diese sieht beim initialen Aufruf folgendermaßen aus:

Abbildung 3: Die CyberRehaWeb Startseite.

Für den Patienten ist die wichtigste Änderung das CyberReha-Menü:

Abbildung 4: Das CyberReha-Menü von CyberRehaWeb.

Dieses ist als Untermenü von „Mein RehaWeb“ angesiedelt. Hier kann der Patient neue Trainingseinheiten in Einzel- und Mehrspielermodus anlegen, Freunde zu diesen einladen und vergangene Trainingseinheiten betrachten.
Für die Rollen Therapeut und Arzt existieren unter dem Menüpunkt ähnliche Ansichten, um Trainingseinheiten für den Patienten beispielsweise anhand eines Trainingsplans anzulegen. Außerdem können diese auf erweiterte Statistiken zugreifen. Administratoren können zudem die auswählbaren Minigames konfigurieren.

Der Aufruf der Unity-Instanz für eine konkrete Trainingseinheit erfolgt durch Klicken des Links „Spielen“ direkt aus dem Browser. Die im Folgenden erläuterten Mechanismen bleiben vor dem Benutzer verborgen.

3.8.2 Installationsanleitung und Betriebshandbuch

3.8.2.1 Einleitung In diesem Dokument wird schrittweise beschrieben, welche Tätigkeiten nötig sind, um eine lauffähige Instanz von „CyberRehaWeb“, also der um die Features von CyberReha ergänzten RehaWeb-Plattform, zu erzeugen.
Grundsätzlich gibt es zwei Wege, eine solche Instanz zu erzeugen. Zum einen aus der klassischen Entwicklersicht über das Einholen der Sourcen, Bauen des Projekts und anschließendes Deployment. Alternativ kann auch ein vorkompiliertes .war-Archiv bei den CyberReha-Autoren bezogen werden.
Für die erste Variante sind alle Informationen in diesem Dokument ab Abschnitt 3.8.2.2 relevant.
Für den zweiten Weg genügt es, sich ein vorkompiliertes .war-Archiv von den CyberReha-Autoren zur Verfügung stellen zu lassen. Die Ansprechpartner dafür und für CyberRehaWeb überhaupt sind Michael Freimuth (michael.freimuth@tu-dortmund) und Lukas Pradel (lukas.pradel@tu-dortmund).
Zudem können Hasan Simsek (hasan.simsek@uni-dortmund.de) und Michael Pantförder(michael.pantfoerder@tu-dortmund) weiterhelfen, da diese Projektleiter bzw. stellvertretender Projektleiter sind. In diesem Fall ist Kapitel 3.8.2.3 nicht relevant.

3.8.2.2 Voraussetzungen CyberRehaWeb ist eine Java-basierte Webapplikation. Die Ausführungsumgebung muss mit einer Java 7 Runtime Environment (JRE) ausgestattet sein.
Hinweis: In diesem Punkt unterscheidet sich CyberRehaWeb von RehaWeb, welches unter Java 6 betrieben wurde!
Idealerweise sollte ein Subversion-Client installiert sein, ein bekannter Client mit GUI unter Windows ist beispielsweise TortoiseSVN: `http://tortoisesvn.net/`.
Die Zugangsdaten für öffentlichen Lesezugriff des CyberReha-Repositories lauten:

> Benutzer: publicuser
> Passwort: CyberReha

Der Servlet-Container für CyberRehaWeb ist ein Apache Tomcat 6 (`http://tomcat.apache.org/download-60.cgi`).

Hinweis: Es ist zu beachten, dass frühere (5 und früher) und insbesondere neuere Versionen des Tomcat (8 und später) nicht von CyberRehaWeb unterstützt werden! Erfolgreiche getestete Versionen sind 6.0.26 und 6.0.37. Ein vollkommen vorkonfigurierter Tomcat 6.0.37 kann über das CyberReha-SVN unter folgender URL bezogen werden:

> `https://cyberreha.cs.uni-dortmund.de:60003/svn/CyberReha/impl/CyberRehaWeb/apache-tomcat-6.0.37`.

3.8.2.3 Sourcen Das Wurzelverzeichnis des CyberReha-SVN-Repositories lautet `https://cyberreha.cs.uni-dortmund.de:60003/svn/CyberReha/`. Im Ordner `docs` findet sich die Dokumentation des Projekts (darunter auch dieses Dokument), während Sourcen im Verzeichnis `impl` zu finden sind.
Alle CyberRehaWeb-relevanten Sourcen sind unter folgender URL verfügbar:

> `https://cyberreha.cs.uni-dortmund.de:60003/svn/CyberReha/impl/CyberRehaWeb`

Das CyberRehaWeb-Projekt ist im Unterverzeichnis `CyberRehaWeb/trunk` (bzw. `tags`) zu finden.
Es handelt sich um ein mit Maven (`http://maven.apache.org/`) gebautes Projekt. Empfohlen wird Maven in der Version 3.
Hinweis: Erfahrungsgemäß kann es zu Schwierigkeiten kommen, wenn auf dem Entwicklerrechner Antiviren-Software, insbesondere AntiVir, installiert ist. In diesem Fall ist sicherzustellen, dass URLs wie `mvnrepository.com` und `maven.org` nicht blockiert werden. Anderenfalls kann es sein, dass das Laden der Dependencies über Maven fehlschlägt.
CyberRehaWeb besteht aus den folgenden Modulen:

> - `CyberRehaWeb` - Unter diesem Projekt sind alle Teilprojekte angesiedelt
> - `CyberRehaWeb-ControlledBlend` - Das Parent-Projekt für ControlledBlend
> - `CyberRehaWeb-ControlledBlend-Decider`
> - `CyberRehaWeb-ControlledBlend-RSBAC`
> - `CyberRehaWeb-Server` - Das Parent-Projekt für die Webapplikation
> - `CyberRehaWeb-Server-API` - APIs der Webapplikation

- `CyberRehaWeb-Server-Access` - Zugriffskontrolle der Webapplikation
- `CyberRehaWeb-Web` - Die eigentliche Webapplikation, welche ein .war generiert

Die offiziell unterstütze IDE für CyberRehaWeb ist Eclipse (`http://www.eclipse.org/`). Hier bietet sich insbesondere das m2e-Plugin an. Vermutlich kann das Projekt auch mit äquivalent ausgestattetem IntelliJ gebaut werden, dies wurde jedoch nicht getestet.
Das Bauen des Projekts erfolgt jetzt wie folgt: Rechtsklick auf das Super-Projekt CyberRehaWeb → Run as → Maven build . . . - Hier trägt man unter goals folgendes ein:

```
clean install
```

Verwendet man den vorkonfigurierten Tomcat 6 kann zusätzlich das Goal

```
tomcat:deploy
```

konfiguriert werden. Dann wird die Anwendungen direkt aus Eclipse heraus auf den Tomcat deployed, sofern dieser gestartet ist.

3.8.2.4 Deployment Wie bereits in Kapitel 3.8.2.3 beschrieben, kann CyberRehaWeb direkt aus Eclipse heraus über das Maven Goal

```
tomcat:deploy
```

deployed werden, sofern der vorkonfigurierte Tomcat eingesetzt wird.
Anderenfalls findet sich im `target`-Verzeichnis des `CyberRehaWeb-Server-Web`-Projekts die Datei „CyberRehaWeb-Server-Web.war“. Diese wird anschließend in das Verzeichnis %`TOMCAT_HOME`%`/webapps` kopiert.
Nun kann der Tomcat beispielsweise unter Windows über das Skript „startup.bat“ gestartet werden, CyberRehaWeb wird dann automatisch deployed.

3.8.2.5 Betrieb Bei Verwendung des vorkonfigurierten Tomcat gibt es für den Betrieb von CyberRehaWeb nichts zu beachten.
Verwendet man einen eigenen Tomcat, muss im Verzeichnis %`TOMCAT_HOME`%`/rehaweb/conf` eine Datei „datasources.properties“ mit folgendem Inhalt angelegt werden:

```
resource.ds1.className=org.apache.derby.jdbc.EmbeddedXADataSource
resource.ds1.uniqueName=jdbc/rehaweb/server
resource.ds1.minPoolSize=2
resource.ds1.maxPoolSize=10
resource.ds1.allowLocalTransactions=true
resource.ds1.driverProperties.databaseName=../rehaweb/data/db/rehaweb-server
resource.ds1.driverProperties.createDatabase=create
```

Zusätzlich müssen in das Verzeichnis %`TOMCAT_HOME`%`/libs` folgende Dateien kopiert werden:

- AuthenticRoast-API-0.3.3.jar
- AuthenticRoast-Impl-0.3.3.jar
- AuthenticRoast-Extras-0.3.3.jar

Die Dateien können unter folgender URL heruntergeladen werden: `http://code.google.com/p/authenticroast/`.

Die eingesetzte Datenbank ist Apache Derby 10.7.1.1. Diese wird Dateisystembasiert betrieben und befindet sich im Verzeichnis %`TOMCAT_HOME`%`/rehaweb/data/db`. Sie wird über die JPA 1.0 mit EclipseLink 2.1 Implementierung aus der Anwendung angesteuert. Als Frontend-Framework dient Java Server Faces in der Version 2.1.

3.8.3 Webschnittstelle CyberRehaWeb-Unity

3.8.3.1 Einleitung Dieses Dokument beschreibt die Webschnittstelle der Webapplikation CyberRehaWeb, über die Unity-Instanzen Spiele bzw. Trainings Informationen aus der Anwendung zu diesen Trainings beziehen können.
Es wird über das CyberReha-SVN unter folgender URL gepflegt:
`https://cyberreha.cs.uni-dortmund.de:60003/svn/CyberReha/docs/CyberRehaWeb/Webschnittstelle`.
Die Ansprechpartner für die Schnittstelle und für CyberRehaWeb überhaupt sind Michael Freimuth (michael.freimuth@tu-dortmund.de) und Lukas Pradel (lukas.pradel@tu-dortmund.de).

Zudem können Hasan Simsek (hasan.simsek@uni-dortmund.de) und Michael Pantförder(michael.pantfoerder@tu-dortmund.de) weiterhelfen, da diese Projektleiter bzw. stellvertretender Projektleiter sind.

3.8.3.2 Web-Schnittstelle Da die Unity-Instanz über RehaWeb aus dem Browser heraus gestartet werden soll, wird die Anwendung in zwei Schritten deployed. Klickt ein Benutzer in RehaWeb auf den entsprechenden „Spielen“-Button wird auf eine Java WebStart Datei (.JNLP) umgeleitet und dann aufgefordert, den Zugriff zu berechtigen. Stimmt er diesem zu, wird die Java-Applikation auf den Client-Rechner geladen und ausgeführt. Diese verfügt im Gegensatz zum Unity WebPlayer über vollen Lese- und Schreibzugriff, was für CyberReha essentiell ist, wie wir später noch beschreiben. Die Webstart-Anwendung erhält von RehaWeb einige benötigte Informationen, wie z.B. ein JSON-Objekt mit Informationen zum aktuellen Spiel, aber auch den Hash-Wert des aktuellen Unity-Spiels.
Die Java-Applikation lädt vom RehaWeb-Server das Unity-Spiel unter der URL

```
/CyberRehaWeb-Server-Web/resources/cyberreha/cyberreha-unity.zip
```

herunter, falls lokal keine Kopie existiert, oder die lokale Kopie einen falschen (also alten) Hash-Wert hat. Anschließend entpackt die Applikation das Spiel und führt die .exe-Datei aus. Das gezippte Unity-Spiel ist dabei das klassische „Desktop-Deployment“ aus dem Unity-Editor heraus. Nur so kann gewährleistet werden, dass die Treiber für die Oculus Rift geladen werden, da diese nativen Code enthalten und nicht aus der geschützten Umgebung des Webplayers genutzt werden können.
Die Java-Applikation liefert der Unity-Instanz beim Start per Kommandozeilen-Argument die Informationen zum Spiel als JSON-Objekt. Dieses kann aus Unity heraus per C#-Skript folgendermaßen abgerufen werden:

```
string[] arguments = Environment.GetCommandLineArgs();
string[] gameInfo = arguments[1];
```

Das JSON-Objekt ist also immer der nullte übergebene Parameter.
Das von CyberRehaWeb übertragene JSON-Objekt enthält die folgenden Elemente:

- `username` (String): Der (technische) Benutzername des CyberRehaWeb-Users, der die Unity-Instanz startet

- `profile` (Long): Die ProfileID des CyberRehaWeb-Users, der die Unity-Instanz startet
- `session` (String): Die SessionID des gerade eingelogten CyberRehaWeb-Users (der die Unity-Instanz startet)
- `workout` (Long): Die WorkoutID zur gestarteten Trainingseinheit
- `self` (String): Die URL zum REST-Webservice mit Informationen zur gestarteten Trainingseinheit
- `type` (String/Enum): Die Art des Trainings, das gestartet wird. Mögliche Werte sind: `SINGLE_PLAYER_FREE`, `SINGLE_PLAYER_SCHEDULED`, `MULTI_PLAYER_FREE`, `MULTI_PLAYER_SCHEDULED`. Die -FREE-Varianten sind dabei vom Patienten selbst angelegte und gestartete Trainingseinheiten, während die -SCHEDULED-Varianten von einem Administrator (Arzt) angelegt wurden
- `host_ip` (String): Die IP-Adresse des Users, der dieses Spiel hostet
- `participants` (Liste von String): Alle Teilnehmer des Trainings, angefangen mit dem Host
- `mini_games` (Liste von String): Die Namen der Minigames, die für die Trainingseinheit ausgewählt wurden in der ausgewählten Reihenfolge (werden in der RehaWeb-DB gepflegt; weitere Informationen können sobald implementiert anhand des Namens über die REST-Schnittstelle erfragt werden)

Für das Parsen von JSON-Objekten in Unity gibt es zahlreiche Bibliotheken. „SimpleJSON“ wurde erfolgreich getestet. Es kann unter folgender Adresse heruntergeladen werden: `http://wiki.unity3d.com/index.php/SimpleJSON`. Auf der Seite gibt es auch eine gute Dokumentation. Ein Beispiel für die API-Verwendung anhand des CyberRehaWeb-JSON-Objekts sieht folgendermaßen aus:

```
void parseGameInfo() {
    var N = JSON.Parse(info);
    String username = N["username"];
    String workoutType = N["type"];
}
```

3.8.3.3 Beispiele Ein vollständiges JSON-Objekt, wie es CyberRehaWeb an Unity übertragen würde, sähe beispielsweise folgendermaßen aus:

```
{
    "username":"test",
    "profile":2,
    "session":"EEDB49B4B724ED9DD6A8A8E75C2BEE94",
    "workout":601,
    "self":"http://localhost:8081/CyberRehaWeb-Server-Web/cyberreha/
    playSingleplayer.xhtml/workouts/351",
    "type":"MULTI_PLAYER_FREE",
    "host_ip":"127.0.0.1",
    "participants":["test, test2"],
    "mini_games":["Flugspiel","Parcours-Spiel"]
}
```

In diesem Fall handelt es sich um ein Freestyle-Mehrspieler-Spiel, das vom (technischen) User „test“ selbst angelegt wurde. Seine profileID in RehaWeb ist 2. Dieser startet (innerhalb der Session mit ID EEDB49B4B724ED9DD6A8A8E75C2BEE94) das Training (mit der ID 601 innerhalb von RehaWeb) und ist neben dem Nutzer „test2“ der erste Mitspieler (der Host ist immer der erste Teilnehmer in der Liste). Da CyberRehaWeb in diesem Beispiel lokal betrieben wurde, ist die IP-Adresse des Users der localhost. Die konfigurierten Minigames sind das Flugspiel gefolgt vom Parcoursspiel.
Hinweis: Das JSON-Objekt wurde hier aus Gründen der Leserlichkeit formatiert gesetzt, CyberRehaWeb liefert, wie allgemein üblich, einen unformatierten String ohne Leerzeichen und Zeilenumbrüche, um den entstehenden Traffic zu minimieren. Technisch gesehen ist die Formatierung für JSON-Parser natürlich ebenfalls unerheblich.
Weiterhin existiert ein Minimalbeispiel für eine Schnittstellen-gerechte Unity-Instanz im CyberReha-SVN unter folgender Adresse:
`https://cyberreha.cs.uni-dortmund.de:60003/svn/CyberReha/impl/Beispielprojekte/CyberRehaWeb-Webschnittstelle`.

4 Anforderungen

4.1 Verwendete Hardware

4.1.1 Ergometer cardio pro

Das Ergometer *cardio pro* (2003) verfügt über eine RS232-Schnittstelle zur Kommunikation mit Datenpaketen. Auf Anfrage sendet das Ergometer aktuelle Fahrdaten als Byte-String. Das Datenpaket enthält u.a.:

- eingestellte Wattleistung
- Geschwindigkeit
- Drehzahl
- gefahrene Zeit

Außerdem enthält das Kommunikationsprotokoll einen Befehl zum Einstellen der Wattzahl. Das *cardio pro* kann in 5W-Schritten auf Leistungsstufen zwischen 25W und 400W eingestellt werden.

4.1.2 Ergometer medical 8i

Das Ergometer *medical 8i* (2010) verfügt über eine Ethernet-Schnittstelle zur Kommunkation mit Datenpaketen. Das Kommunikationsprotokoll enthält Befehle zum Abfragen einzelner Fahrdaten, sowie für ein Gesamtpaket mit allen aktuellen Daten. Das Gesamtpaket enthält u.a.:

- eingestellte Wattleistung
- Geschwindigkeit
- Drehzahl
- gefahrene Zeit

Außerdem existiert ein Befehl zum Einstellen der Wattzahl. Das *medical 8i* kann in 1W-Schritten auf Leistungsstufen zwischen 20W und 1000W eingestellt werden.

4.1.3 Brustgurt Zephyr BioHarness 3

Während des Trainings trägt der Spieler einen kabellosen Brustgurt *Zephyr BioHarness 3*, mit dem per Bluetooth kommuniziert werden kann. Ist eine Verbindung zum Brustgurt hergestellt, so sendet dieser selbstständig in regelmäßigen Abständen verschiedene Daten wie Herzfrequenz, Atemfrequenz oder Hauttemperatur.

4.2 Mindestanforderungen Betriebs-Hardware

4.2.1 CyberRehaWeb-Server

Der CyberRehaWeb-Server sollte auf einem Rechner mit mindestens 4 GB Arbeitsspeicher (empfohlen: 8 GB) und einem Mehrkern-Prozessor betrieben werden. Außerdem sollte er über eine Breitband-Internetverbindung verfügen. Darüber hinaus wird ein 64 Bit Befehlssatz des Prozessors und ein entsprechendes Betriebssystem zum Betrieb empfohlen.

4.2.2 CyberReha-Client (Unity)

Die Client-Rechner der Patienten sollten ebenfalls über mindestens 4 GB Arbeitsspeicher (empfohlen: 8 GB) und einen Mehrkern-Prozessor verfügen. Sie benötigen ebenfalls eine Breitband-Internetverbindung. Weiterhin ist hier zur ruckelfreien Darstellung der Spielinhalte eine moderne Grafikkarte mit mindestens 1 GB VRAM nötig.

5 Anwendungsentwurf

5.1 Erfassung von Vitaldaten

Während des Trainings werden mit verschiedenen Sensoren die Vitaldaten des Spielers erfasst. Diese dienen zum Einen der Überwachung und Darstellung im Spiel, zum Anderen aber auch zur Erstellung und Auswertung von Statistiken. Dabei kommen ein Brustgurt sowie ein Pulsoxymeter zum Einsatz, deren Aufgaben und Funktionsweise in den folgenden Abschnitten näher erläutert werden.

5.1.1 Brustgurt

Wesentliches Merkmal zum Feststellen einer körperlichen Belastung ist die Herzfrequenz. Die ideale Herzfrequenz zum Erzielen eines optimalen Trainingseffekts ist für jeden Spieler unterschiedlich, abhängig von seiner normalen Herzfrequenz ohne körperliche Belastung, auch Ruhepuls genannt. Damit ein möglichst guter Trainingseffekt erzielt werden kann, muss sich die Herzfrequenz während des Spielens also in einem bestimmten Bereich um die ideale Herzfrequenz bewegen.
Auf der anderen Seite muss aber bei bestimmten Spielergruppen auch darauf geachtet werden, dass diese nicht überlastet werden. Dies ist zum Beispiel bei Spielern mit Herzerkrankungen und daraus resultierendem erhöhtem Herzinfarktrisiko der Fall, oder auch, wenn ein Spieler in der Vergangenheit bereits einen Herzinfarkt hatte. In diesem Fall darf eine bestimmte maximale Herzfrequenz nicht überschritten werden. Das bedeutet, dass die Herzfrequenz permanent überwacht wird und als wesentlicher Faktor in die Lastregelung einfließen muss. Weiterhin werden die Daten jeder Trainingseinheit zur statistischen Auswertung aufgezeichnet.

5.1.2 Pulsoxymeter (SpO2)

Zur Messung der Sauerstoffsättigung des Bluts wird ein Pulsoxymetrie-Sensor verwendet, den der Spieler während des Trainings am Finger trägt. Dieser Sensor durchleuchtet den Finger, misst dabei das absorbierte Licht und berechnet daraus die Konzentration der roten Blutkörperchen, die vollständig mit Sauerstoff gesättigt sind. Die Aufzeichnung findet ausschließlich zur statistischen Auswertung statt.

5.2 Lastregelung

Abhängig von verschiedenen Einflüssen wird während des Trainings die am Ergometer anliegende Last variiert. Dadurch verändert sich der erforderliche Kraftaufwand

für das Treten der Pedale. Zum Einen werden Gegebenheiten aus dem Spiel, wie z.B. der Bodenbelag oder die aktuelle Steigung berücksichtigt. Wenn sich während des Spiels die zu überwindende Steigung oder der Bodenbelag ändert, macht sich dies durch eine veränderte Last für den Spieler bemerkbar. Allerdings hält dieser Effekt nur einen vergleichsweise kurzen Zeitraum an. Wesentlich wichtiger ist allerdings der Einfluss der Vitaldaten, um einen tatsächlichen Trainingseffekt zu erzielen. Um die gewünschten Trainingseffekte zu erreichen, müssen die Vitaldaten des Spielers berücksichtigt werden (Regelung). Während der Hauptphase des Trainings soll die Last mit fortschreitender Trainingsdauer immer weiter zunehmen.
Für die Regelung werden die Vitaldaten des Spielers berücksichtigt. Maßgeblich ist dabei der Abstand der aktuellen Herzfrequenz zur vorgegebenen Herzfrequenz, die im Spielerprofil in der Rehaweb-Community hinterlegt ist. Dabei werden verschiedene Bereiche für die Herzfrequenz unterschieden, nach denen die Last geregelt wird:

- ideal: Vorgabe ±10%
- ok: Vorgabe ±20%
- zu niedrig: mehr als 20% unter Vorgabe
- zu hoch: mehr als 20% über Vorgabe
- viel zu hoch: länger als 30s in *zu hoch*

5.3 Hauptmenü

Das Hauptmenü ist, neben dem neu geschriebenen RehaWeb-Modul, die erste Anlaufstelle des Nutzers. Im Gegensatz zu RehaWeb wird der Nutzer hier jedoch erstmals mit dem entwickelten Spiel und der zu verwendenden Hardware konfrontiert. Insbesondere soll das Menü hierzu zwei Dinge leisten:

1. Der Nutzer soll einen ersten (optischen) Eindruck des Spiels erhalten.
2. Der Nutzer soll an die Verwendung der *Oculus Rift* gewöhnt werden.

Um optimal einen ersten Eindruck zu vermitteln, soll das Menü aus Elementen der implementierten Minigames gestalten werden.
Da eine Verwendung gewöhnlicher Eingabegeräte durch die Oculus Rift, insbesondere bei Nichtspielen, stark eingeschränkt oder sogar unmöglich ist, wurde bereits früh in der Entwurfsphase entschieden, die Menünavigation vollständig durch die Oculus Rift zu ermöglichen.

Um die Usability zu fördern soll das Menü nur wenige Schaltflächen besitzen. Komplexe Einstellungen der Applikation wurden zu Gunsten des Konzepts der Einfachheit geopfert. Lediglich Schaltflächen zur Auswahl eines Minigames sollen vorhanden sein; all diese Schaltflächen müssen über die Oculus Rift bedienbar sein.
Während das Menü im Einzelspielermodus die Auswahl von Minigames ermöglicht, stellt sich der Sachverhalt im Mehrspielermodus etwas anders da. Hier wird vorab in RehaWeb eine Abfolge von Minigames für die Trainingsgruppe festgelegt. In diesem Szenario soll das Menü die Funktion eines Warteraums übernehmen. Insbesondere soll es dem Spieler Rückmeldung über den Fortschritt der Vorbereitung der Session geben. Hierzu soll visuelles Feedback implementiert werden. Als Kriterium für den Fortschritt wurde die Anzahl der verbundenen Spieler gewählt.

5.4 Spielprinzip des Minigames "Parcours"

Im Parcours-Spiel hat der Anwender die Möglichkeit, eine Surfer-Figur durch eine lange Rennstrecke zu navigieren. Hierbei werden die Pedale des Ergometers intuitiv zum Antrieb des Surfers und das Lenkrad zu seiner Steuerung verwendet. Das Ziel des Spiels ist es, den Parcours möglichst schnell zu durchlaufen - als Motivation wird hierfür am Ende des Spiels ein Punktestand angezeigt, der auf einem Quotienten zwischen den passierten *Checkpoints* und der benötigten Zeit basiert. Neben dem Einzelspielermodus existiert hier die Möglichkeit, die Strecke mit bis zu vier Mitspielern zu teilen. Im Mehrspielermodus wird der schnellste Fahrer am Spielende auf dem Bildschirm ausgegeben.

Erschwerend kommen hierbei teils bewegliche Hindernisse hinzu, die der Spieler nicht berühren darf, da er sonst eines seiner vier Leben verliert. Die Lenkung am Ergometer dient daher hauptsächlich dem Ausweichen nach links und rechts. Verliert der Spieler alle seine vier Leben, muss er die Fahrt von der Position des letzten durchlaufenen Checkpoints aus fortsetzen. Um einen möglichst hohen finalen Punktestand zu erreichen, sollte der Spieler sich demnach bemühen, möglichst selten mit Hindernissen zu kollidieren. Des Weiteren sollte es der Spieler vermeiden, durch *sandiges Gelände* zu fahren, da dies sowohl die Fahrt verlangsamt als auch den Widerstand der Ergometerpedale erhöht. *Eisiger Boden* bewirkt zwar eine Senkung des Pedalwiderstandes, bringt allerdings auch einen Kontrollverlust bei der Navigation des Surfers mit sich. Auf der Strecke befinden sich vereinzelt beschleunigende Flächen, sogenannte *Turbo-Boosts*, welche die Fahrgeschwindigkeit für einen kurzen Zeitraum deutlich erhöhen.

Die Länge der Rennstrecke ist nicht fest vorgegeben. Je nach vorgegebener Spieldauer verlängert sich die Rennstrecke automatisch bis die Spieldauer fast erreicht ist: So soll eine Einhaltung der vorgegebenen Spielzeit mit einer kleinen Toleranz angestrebt werden. Im Mehrspielermodus wird eine übermäßige Überschreitung der vorgegebenen Spielzeit insofern vermieden, dass die verbliebenen Spieler ab dem Moment, in dem der Sieger die Ziellinie überquert, noch eine Minute Zeit haben, um ebenfalls das Ziel zu erreichen. Andernfalls wird das Spielende erzwungen.

5.5 Spielprinzip des Minigames "Flugspiel"

In diesem Minispiel kann ein Flugzeug durch eine frei erkundbare 3D-Welt navigiert werden. Die Pedale des Ergometers bzw. die Trittfrequenz der Fahrers regelt hierbei die Flughöhe, dessen aktueller Stand grafisch auf dem Bildschirm angezeigt wird. Das Lenkrad des Ergometers wird für die Rechts-/Linkssteuerung verwendet.

Das Ziel des Spiels ist es, die Welt zu erkunden und möglichst viele Münzen aufzusammeln, die an unterschiedlichen Orten verteilt sind. Dabei besteht die Aufgabe darin, eine möglichst effiziente Strategie zu entwickeln, um die Münzen so schnell wie möglich zu finden und aufzusammeln. In der Voreinstellung sind es 50 Münzen, die gesammelt werden müssen, was aber individuell eingestellt werden kann. Wie viele Münzen sich aktuell auf dem Konto befinden und wie viel Zeit man für das einsammeln aller Münzen benötigt hat, wird ebenfalls auf dem Bildschirm visualisiert.

Neben dem Einzelspielermodus existiert auch in diesem Spiel die Möglichkeit, die Strecke mit bis zu vier Mitspielern zu teilen. Im Mehrspielermodus ist das Ziel, eine vorher festgelegte Anzahl an Münzen vor allen anderen Mitspielern zu finden und einzusammeln. Der schnellste Fahrer wird am Spielende auf dem Bildschirm ausgegeben.

Die Welt besteht aus unterschiedlich großen Inseln, auf denen sich der Größe entsprechend viele Objekte befinden. Die Objekte, die sich je nach Inselart (Schneeinsel, Wüsteninsel, usw.) unterscheiden, sollen beim Einsammeln der Münzen hinderlich wirken. Um die Inseln zu erreichen, müssen mit dem Flugzeug unterschiedliche Entfernungen zurückgelegt werden. Die kleineren Inseln sind vom Startpunkt aus schneller zu erreichen, als die größeren Inseln. Der Nachteil des längeren Weges zu einer großen Insel gleicht sich aber aus, indem dort mehr Münzen gesammelt werden

können, als auf den kleineren Inseln. Es gibt also mehrere Flugrouten zur Auswahl und der Spieler muss für sich, in Abhängigkeit von seinen individuellen Fähigkeiten (Steuerung des Ergometers) und der aktuellen Spielsituation, eine geeignete Strategie herausfinden.

Auch in diesem Spiel sind in der Welt *Turbo-Boosts* verteilt, welche die Fluggeschwindigkeit für einen kurzen Zeitraum deutlich erhöhen. Diese sind im Grunde so angeordnet, dass sie sich auf dem Weg zu den großen Inseln befinden, um diese schneller zu erreichen. Gekennzeichnet sind sie durch einen bunten Würfel, der unter einem Heißluftballon hängt.

6 Softwareentwurf

6.1 Gesamtarchitektur

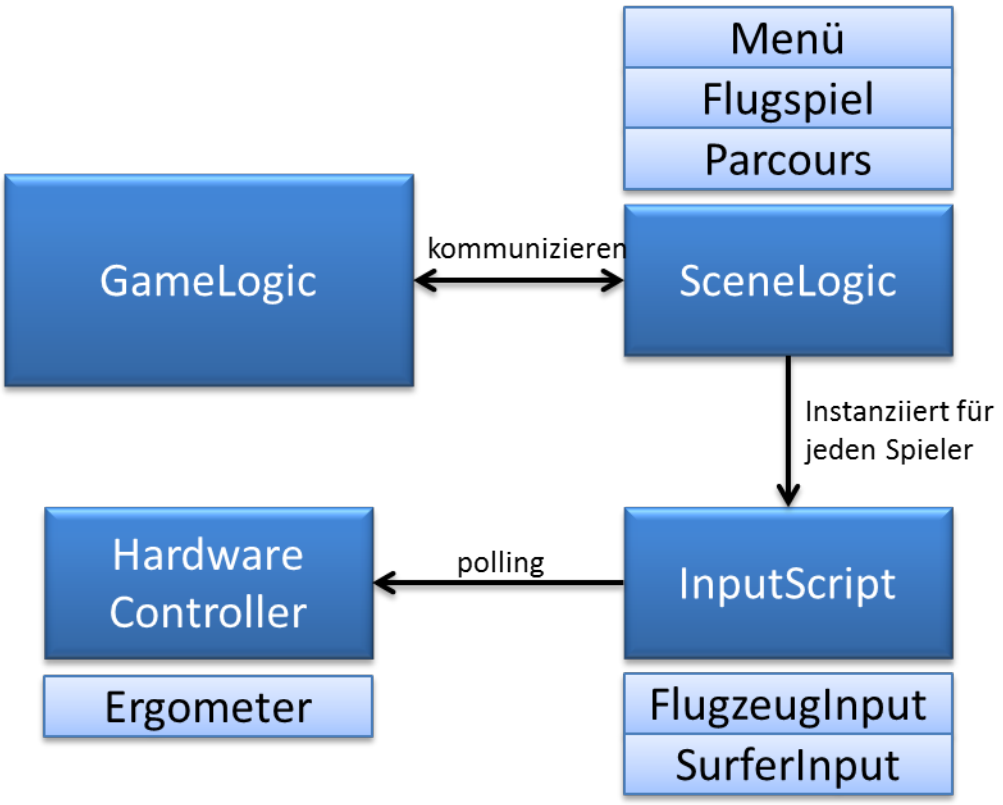

Abbildung 5: Software-Architektur des Unity-Spiels

Ein Schaubild der Software-Architektur ist Abbildung 5 zu sehen. Das oberste Skript zur Verwaltung der Spiellogik ist das GameLogic-Skript. Es bleibt beim Szenenwechsel (z.B. von Menü zu Minispiel) erhalten und verwaltet alle Daten, die für das Gesamtspiel wichtig sind. Dazu gehören die übergebenen Parameter aus RehaWeb, wie Spielreihenfolge und Workout-ID, sowie die bereits verstrichene Zeit seit dem Spielstart und weitere für den Multiplayer-Modus wichtige Daten und Funktionen. Das GameLogic-Skript instanziiert die SceneLogic-Skripte der Szenen und kommuniziert anschließend mit ihnen. Nach einem Szenenwechsel werden allerdings alte SzeneLogic-Skripte zusammen mit der Szene aus dem Speicher entfernt. Vorher werden daher Traningsdaten und der finale Punktestand der Spieler ins GameLogic-Skript geschrieben. Für eine neue Szene wird immer ein neues SceneLogic-Skript erstellt, das eine entsprechende Start-Konfiguration durchführt.
Zu Beginn einer Szene wird für jeden verbundenen ein Spielerobjekt erstellt. In

diesen Objekten ist die Input-Verwaltung verzahnt. Je nach Szene können diese unterschiedlich ausfallen. So wird im Flugspiel das Flugzeug mit der Steuerung für Höhen- und Querruder erstellt und im Parcours-Spiel findet man pro Spieler einen Surfer mit einer Links/rechts- und Geschwindigkeits-Steuerung. Im Menü wird lediglich eine Kamera erstellt, die zur Auswahl der Minispiele dient (siehe Abbildung 6.
Soll in den Spielen die Eingabe mittels Ergometer erfolgen, werden die Klassen des Hardware-Controllers geladen und für jeden Update-Zyklus erneut abgefragt.
Kann keine Verbindung zu einem Ergometer hergestellt werden, wird automatisch auf Tastatur/Joystick-Eingabe umgestellt, um das Debuggen so einfach wie möglich zu gestalten.

Abbildung 6: Auswahl-Menü für die Minispiele. Rechts Parcours", links das Flugspiel Äviation".

6.2 Integration der Hardware

Das Konzept der Hardwareintegration ist in Abbildung 7 und 8 visualisiert. Der Zugriff aus den Spielen heraus auf die Hardware-Komponenten erfolgt über eine API, die alle benötigten Funktionen gekapselt zur Verfügung stellt. Die API erhält die in Rehaweb hinterlegten Informationen zum gestarteten Spiel.
Eine weitere Komponente kapselt die Ansteuerung sämtlicher Hardware-Komponenten. Diese erhält ebenfalls die relevanten Informationen zum Spiel und liefert aktuelle Daten der Hardwarekomponenten an die API zurück. Weiterhin werden die Daten

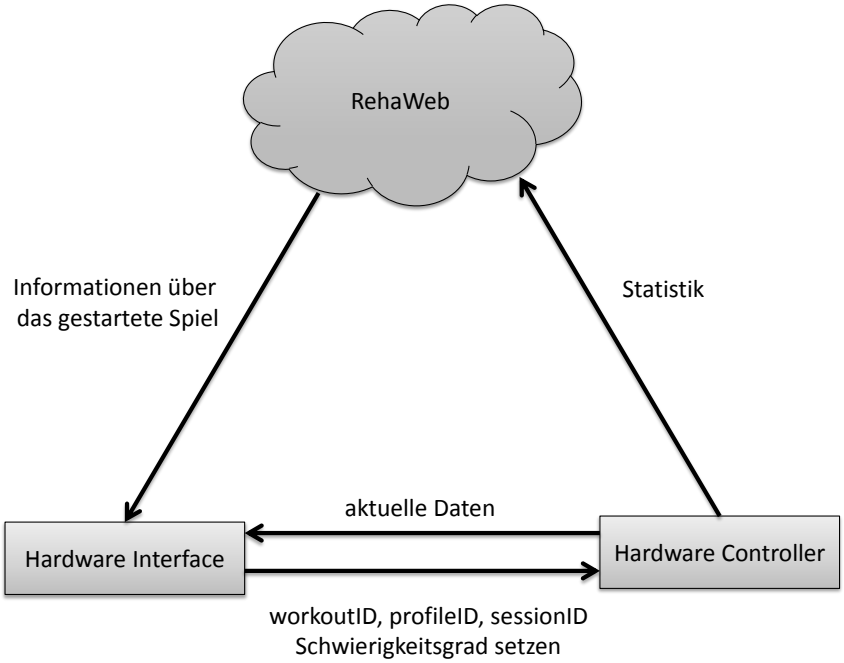

Abbildung 7: Interaktion mit den Hardwarekomponenten

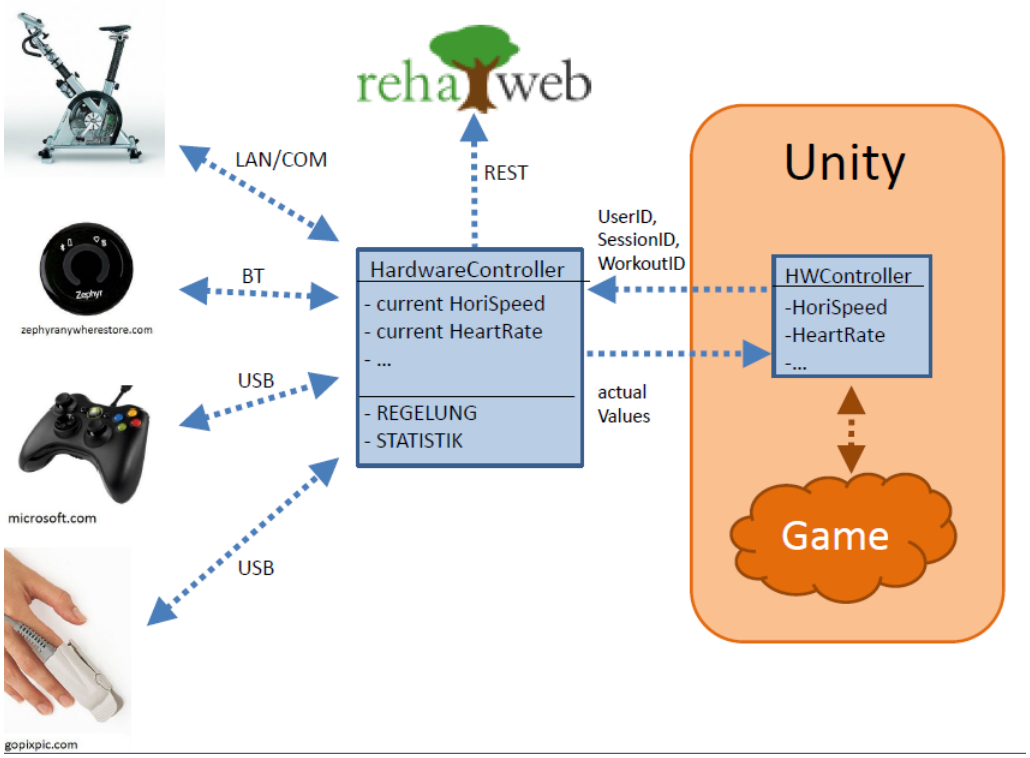

Abbildung 8: Interaktion mit den Hardwarekomponenten

direkt an die an RehaWeb angeschlossene Datenbank übermittelt. Zum Hardware-Controller gehört eine grafische Oberfläche, über die Verbindungen mit angeschlossenen Geräten hergestellt werden können. Diese muss gestartet sein, wenn ein Spiel durchgeführt wird. Zur Verwendung ist es erforderlich, dass die *SlimDX Runtime* installiert ist. Die Oberfläche ist in Abbildung 9 dargestellt.

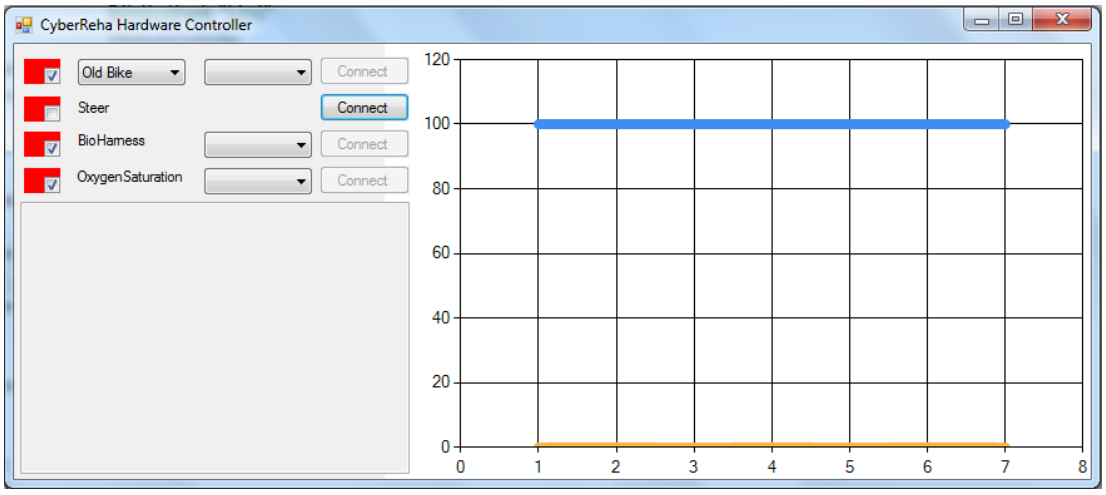

Abbildung 9: Grafische Benutzeroberfläche für den Hardware-Controller. Links können angeschlossene Geräte ausgewählt werden und über *Connect* eine Verbindung hergestellt werden. Rechts kontinuierliche Ausgabe von Statistiken (jeweils die letzten 60 Sekunden)

Während des Spiels werden die aktuellen Werte im HardwareController zyklish an RehaWeb geschickt. Dazu wurde die REST-Schnittstelle ausgenutzt, wo zusätzliche Endpunkte dafür eingerichtet wurden. Der HardwareController erhält vom HWController die benötigten Informationen (wie die ProfileID, WorkoutID und SessionID), um die Statistik dem richtigen Training und Patienten zuzuordnen.

Loggt man sich als Administrator in RehaWeb ein, kann die Statistik für alle abgeschlossenen Trainings eingesehen werden (siehe Abbildung 10). Gesichert werden die Werte der:

- Trittfrequenz,
- Herzfrequenz,
- Sauerstoffsättigung, und
- der Widerstand.

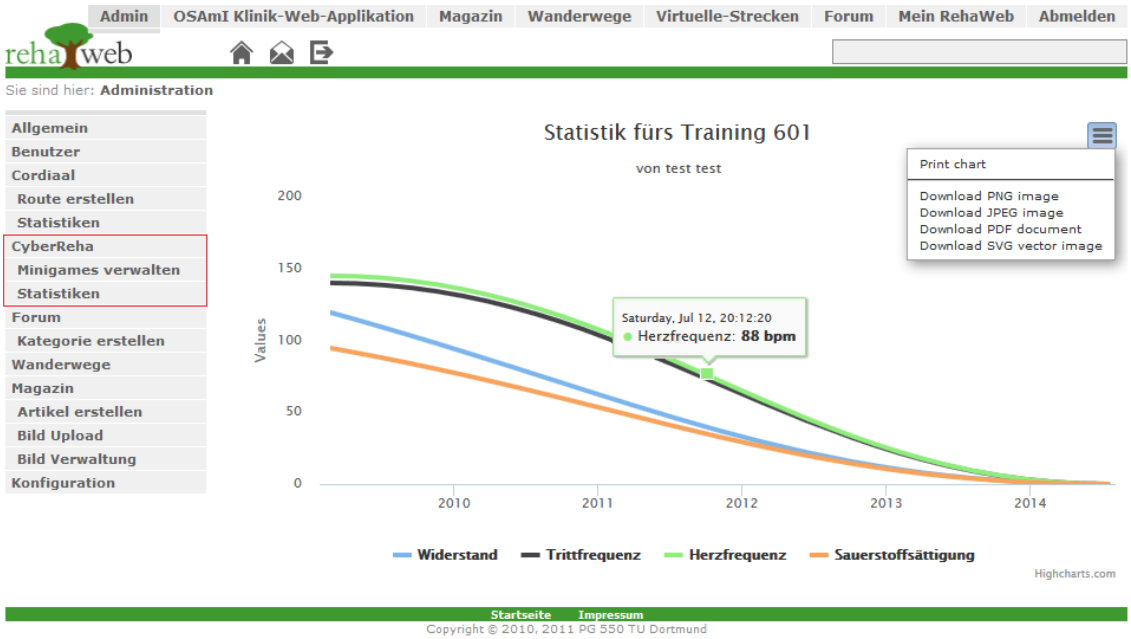

Abbildung 10: Beispiel einer Statistik für ein Training in RehaWeb

7 Implementierung

7.1 Anpassungen der Hardware

7.1.1 Erweiterung der Ergometer um eine Lenkung

Damit ein Spieler sich in einer immersiven 3D-Welt realistisch bewegen kann, muss er natürlich auch die Möglichkeit haben, Lenkbewegungen durchzuführen, die dann in die virtuelle Realität übertragen werden. In der Standardausführung verfügen die Ergometer über keine Lenkmöglichkeit. Das *cardio pro* wurde jedoch von einer Vorgänger-PG um einen beweglichen Lenker erweitert, dessen Bewegung mithilfe eines Potentiometers und eines Joystick-Controllers verarbeitet wurde. Passend dazu musste auch das *medical 8i* mit einem beweglichen Lenker ausgestattet werden. Das Drehen des Lenkers entspricht dabei der Bewegung des Joysticks entlang einer beliebigen Achse.
Für sämtliche Achsen liefert der Joystick ganzzahlige Rohwerte im Intervall (-128, 128) zurück, der Lenker kann also logisch 256 verschiedene Ausrichtungen einnehmen. Da in der Unity-Engine jedoch nur positive Rohdaten verarbeiten werden können, und diese direkt vom Windows-Gamecontroller abgegriffen werden, ist der Joystick ohne eine aufwändige Treiber-Implementierung zur Anpassung der Rohdaten nicht verwendbar. Neuere Gamecontroller liefern jedoch direkt Unity-kompatible Rohdaten, deshalb werden statt des Joysticks XBOX360-Controller für die Lenkung beider Ergometer verwendet. Der XBOX360-Controller liefert ganzzahlige Werte im Intervall (0, 65536), die Lenkrichtung kann zudem also mit wesentlich höherer Genauigkeit bestimmt werden als durch den Joystick-Controller. Außerdem wurden beide Lenker um zwei Knöpfe erweitert, die ebenfalls in die Controller integriert sind, und somit abhängig vom Spiel für verschiedene Funktionalitäten zur Verfügung stehen.

7.1.2 Umbau des alten Ergometers

Beide Ergometer berechnen intern die aktuell gefahrene Geschwindigkeit und versenden diese auf Anforderung zusammen mit weiteren Trainingsdaten. Die Ermittlung der Geschwindigkeit unterscheidet sich jedoch grundlegend. Im Inneren der Ergometer befindet sich jeweils eine Schwungmasse, die über das Treten der Pedale in Bewegung gebracht wird und, sobald der Fahrer aufhört zu treten, langsamer wird. Das *cardio pro* in der Standardausführung misst zur Ermittlung der Geschwindigkeit die Umdrehungen der Schwungmasse mit einer Lichtschranke. Das *medical 8i*

hingegen misst direkt die Drehzahl der Pedale.
Das hat zur Folge, dass beim *cardio pro* die Geschwindigkeit nach dem Anhalten der Pedale langsam abnimmt, während die Geschwindkeit des *medical 8i* direkt 0 wird. Auch wenn beide Verfahren für sich genommen je nach Anwendung Vorteile haben können, so ist der parallele Einsatz in Cyberreha absolut unpraktiabel, da das Verhalten der Spielfigur dann vom verwendeten Ergometer abhängen würde. Aus diesem Grund wurde das *cardio pro* so umgebaut, dass die Lichtschranke zur Bestimmung der Geschwindigkeit nicht länger die Umdrehungen der Schwungmasse misst, sondern, wie beim *medical 8i* die Drehzahl der Pedale. Dazu wurde die Position der Lichtschranke verändert und auf der Scheibe, die direkt mit den Pedalen verbunden ist, zwölf Pappstreifen angebracht, um die Lichtschranke zu unterbrechen.
Die interne Berechnung der Geschwindigkeit hängt von der Anzahl der Impulse der Lichtschranke in einem bestimmten Zeitintervall ab. Es ist zu erwarten, dass durch den Umbau stark verfälschte Werte für die Geschwindigkeit ausgegeben werden. Da auf die interne Berechnung keinerlei Zugriffsmöglichkeit besteht, müssen die vom Ergometer übertragenen Werte nachträglich korrigiert werden. Experimentelle Versuche haben ergeben, dass die Geschwindigkeit des *cardio pro* um den Faktor 1.2 korrigiert werden muss, damit Cyberreha bei gleicher Trittfrequenz beider Ergometer auch den gleichen Wert für die Geschwindigkeit erhält.

7.1.3 Kalibrierung

Nachdem die Lenk-Controller an beiden Ergometern angebracht wurden, musste eine Kalibrierung vorgenommen werden. Aufgrund der mechanischen Beschaffenheit der prototypisch modifizierten Lenker stimmten die Rohwerte nicht exakt mit den tatsächlichen Lenkerstellungen überein. So lagen die Mittelwerte nicht exakt in der mittleren Lenkerposition, und auch die Extrempositionen links und rechts stimmten nicht miteinander überein. Daraufhin mussten die Mittelposition der Lenkerstellung einmalig kalibriert werden, so dass die verfälschten Rohwerte automatisch umgerechnet werden können.

7.2 Integration der Hardware

Die Ansteuerung sämtlicher Hardwarekomponenten wird in der Klasse *MainForm.cs* gekapselt, die auch die GUI zur Hardware-Steuerung (s. Abschnitt 6.2) erzeugt. In *MainForm.cs* werden Instanzen der folgenden Klassen verwaltet:

- *BikeController.cs* (zum Ansteuern des Ergometers)
- *SteerController.cs* (XBOX-Controller zur Lenkung des Ergometers)
- *BioController.cs* (zur Ansteuerung des Brustgurts)
- *OxyController.cs* (Messung der Sauerstoffkonzentration im Blut)

Eine Gesamtübersicht über die Implementierung der Hardware-Einbindung ist in Form eines UML-Klassendiagramms in Abbildung 21 am Ende des Berichts gegeben. Für die Entwicklung der Spiele wird die Klasse *HardwareController.cs* als API bereitgestellt. Die Interaktion zwischen Spiel und Hardware erfolgt über die folgenden Funktionen:

- *getVertSpeed()* liefert die normierte Trittgeschwindigkeit im Bereich [0;1]
- *getHoriSpeed()* liefert die normierte Lenkerstellung im Bereich [-1;1]
- *getHeartRate()* liefert den absoluten Wert der Herzfrequenz
- *getSpO2()* liefert den absoluten Wert der Sauerstoffkonzentration im Blut
- mit *setDifficult()* können Trittwiderstandsanpassungen im Kontext des Spiels als Faktor im Bereich [0;2] eingestellt werden

In der Implementierung der Hardware-Ansteuerung werden auf mehreren Ebenen parallel laufende Threads eingesetzt. In der Klasse *HardwareController.cs* existieren zwei Threads: Ein Thread dient dazu, in regelmäßigen Abständen aktuelle Daten der unterschiedlichen Hardware-Komponenten von der entsprechenden Controller-Klasse (z.B. *BioController.cs* für den Brustgurt) abzufragen und die zurückgegebenen Werte in lokale Variablen zu schreiben. Der andere Thread wartet darauf, dass von Seiten der Spiele die bereitgestellten Funktionen (z.B. *getVertSpeed()*) aufgerufen werden. Erfolgt ein solcher Aufruf, wird der Wert der entsprechenden lokalen Variable zurückgegeben, dies ist also immer der Wert der letzten regelmäßigen Aktualisierung.
Auch in den Controller-Klassen für die einzelnen Hardware-Komponenten existieren mehrere Threads. Auf dieser Ebene wird ebenso ein Thread dazu verwendet, die Werte direkt von der Hardware entsprechend dem hardwareseitig verwendeten Kommunikationsprotokoll abzufragen. Ein weiterer Thread dient dazu, die abgefragten Werte für die Abfrage durch die Klasse *HardwareController.cs* bereitzustellen.

7.3 Lastregelung

Die Lastregelung ist in der Klasse *Regelung.cs* implementiert. Die Regelung wird beim Initialisieren der Hardware-Controller gestartet, dabei wird mit der *ProfileID* des Spielers die ideale Herzfrequenz aus dem RehaWeb-Profil des Spielers übergeben. Ist kein Wert verfügbar, wird als Standardwert eine Herzfrequenz von 130 BPM angenommen. Die herzfrequenzabhängige Regelung wird dabei automatisch in der Klasse *Regelung.cs* durchgeführt; die spielspezifischen Einflüsse werden über die Hardware-Controller-API realisiert.

7.3.1 Herzfrequenz-Regelung

Anhand von vier HF-Bereichen wird die Last des Ergometers folgendermaßen gesteuert:

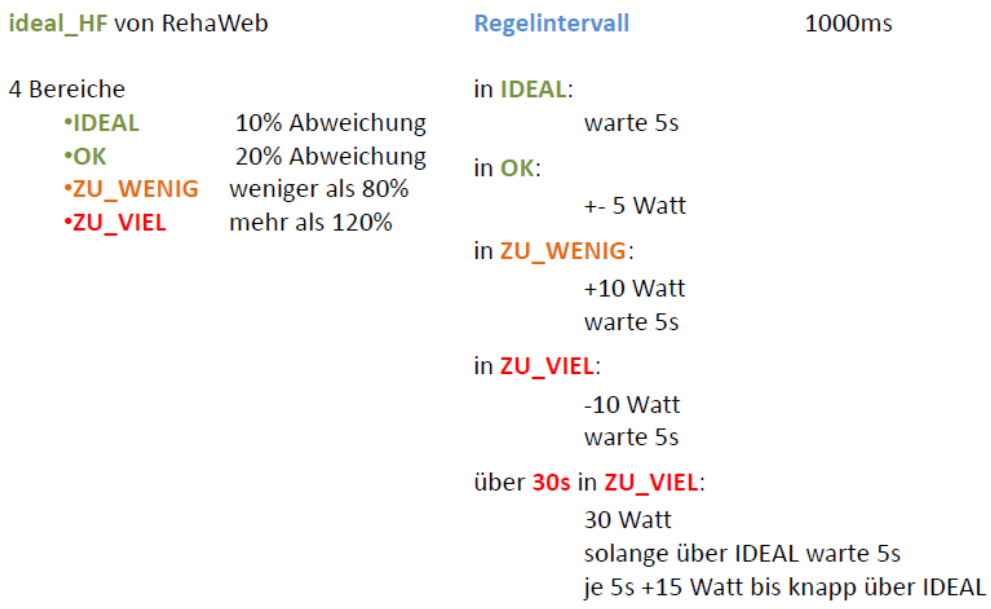

Abbildung 11: Zusammenfassung der Regelung

Der aktuelle Herzfrequenzbereich wird in Intervallen von 400 ms abgefragt. Befindet der Spieler sich im Idealbereich, greift die Regelung nicht ein und pausiert 10 Sekunden.
Im Bereich *ok* wird die Last um 5 Watt korrigiert - wenn sich die Herzfrequenz des Spielers über der Vorgabe befindet wird die Last verringert, ist die Herzfrequenz zu niedrig, wird die Last erhöht. Danach wird eine längere Pause (15 Sekunden) eingelegt, um den Effekt abzuwarten und ein mögliches Schwingen zu vermeiden.
Im Bereich *zu niedrig* (bzw. *zu hoch*) wird die Last um 10 Watt erhöht (bzw. ge-

senkt), gefolgt von wiederum 15 Sekunden Pause. In dieesen Bereichen läuft zudem ein Timer, um die Zeit zu messen, die der Spieler sich schon in diesem Bereich befindet. Nach 90s am Stück wird der Bereich *viel zu hoch* erreicht.
Im Bereich *viel zu hoch* wechselt die Regelung in einen Leerlaufmodus. Dabei wird die Last sofort auf den minimal möglichen Wert (25 Watt) reduziert, bis der Spieler wieder den idealen Bereich erreicht hat. Danach wechselt die Regelung wieder zurück in den normalen Modus.

7.3.2 Spielspezifische Einflüsse

Ein spielspezifischer Einfluss überlagert die Regelung, indem der eingestellte Trittwiderstand für eine kurze Zeit mit einem Faktor aus dem Bereich [0;2] multipliziert wird. Damit ist es möglich, den Widerstand einerseits ganz auszuschalten (Faktor 0), oder ihn maximal zu verdoppeln (Faktor 2). Dies kann direkt aus den Spielen heraus über die Hardware-Controller-API realisiert werden, indem die Funktion `setDifficult(value)` aufgerufen wird.

7.4 Hauptmenü

Die Anforderungen an das Menü wurden bereits in Abschnitt 5.3 ausführlich beschrieben. Zusammengefasst lassen sich diese wie folgt definieren:

- Das Menü soll einfach sein.
  - Das Menü soll selbsterklärend sein.
  - Das Menü soll wenige Elemente enthalten.
- Das Menü soll den Nutzer an die Verwendung der *Oculus Rift* gewöhnen.
  - Insbesondere sollen sämtliche Schaltflächen ausschließlich mit der Oculus Rift bedienbar sein.
- Das Menü soll dem Nutzer einen ersten Eindruck von den Minigames vermitteln.

Im Folgenden soll die Implementierung mit dem Fokus auf diesen Designentscheidungen schrittweise durchgegangne werden.

7.4.1 Einfachheit

Ein zentrales Ergebnis des entstandenen Menüs ist seine Einfachheit. Das Menü für den Einzelspielermodus enthält nur zwei Schaltflächen: eine zum Start von *Parcours* und eine zum Start von *Flugspiel*. Zur Bedienung des Spiels hat sich jedoch herausgestellt, dass mehr Schaltflächen nicht notwendig sind. Diese Schaltflächen werden durch zwei große, sofort auffallende Fernseher repräsentiert. Abbildung 12 zeigt das entstandene Menü mit beiden Fernsehern. Betrachtet der Nutzer einen der Fernseher erhält er sofort visuelles Feedback hierüber. Durch das Betrachten des Fernsehers wird das Spiel implizit ausgewählt und kann nun durch den Nutzer gestartet werden.

Abbildung 12: Das Hauptmenü von CyberReha

7.4.2 Verwendung der Oculus Rift

Die Verwendung der Oculus Rift hat sich als besondere Herausforderung herausgestellt. Die Auflösung der Brille hat es notwendig gemacht, ein möglichst textfreies Menü zu designen. Auch die Verwendung eines klassischen *Graphical User Interface* war mit der verwendeten Version der API nicht möglich. Dies hat maßgeblich zur Entscheidung für ein Menü mit wenigen, großen Schaltflächen beigetragen.
Da das Spiel sowohl mit als auch ohne Brille spielbar sein sollte, musste ein spezielles Asset entwickelt werden, welches diese Umschaltung möglich macht. Dies ist in der aktuellen Version der Applikation möglich.

Da ein Fernseher – und damit ein Spiel – nur über die Oculus Rift ausgewählt werden konnte, musste ein Skript geschrieben werden, welches den Blickkontakt mit einem Objekt identifiziert und entsprechend reagiert. In diesem Kontext ist ein vielseitig einsetzbares Skript entstanden: der `CameraControllerForMenue`.
Dieses Skript stellt mit Hilfe der intern zur Kollionsbestimmung verwendeten *Raycasts* fest, ob ein Objekt vom Spieler betrachtet wird. In dem Fall, dass ein Objekt betrachtet wird, werden auf ihm alle Skripte vom Typ `ActivationHandler` gesucht. Auch hierbei handelt es sich um selbstgeschriebene Skripte, welche die abstrakte C#-Klasse `ActivationHandler` implementieren.
Zur Implementierung des Handlers ist die Implementierung von drei Klassen notwendig:

- `onSelect(GameObject focusedObject)`
- `onFocus(GameObject focusedObject)`
- `onDeselect(GameObject focusedObject)`

Die Methode `onSelect` wird aufgerufen, wenn ein Objekt vom Spieler betrachtet wird, nachdem es bei der vorherigen Abfrage nicht betrachtet wurde. Es handelt sich also um einen neuen Blickkontakt mit dem Element.
Die `onFocus` Methode kann beliebig oft aufgerufen. Dieses Event tritt immer dann ein, wenn ein Objekt vom Spieler betrachtet wird, dies aber bereits über längere Zeit der Fall ist. Beim ersten Blickkontakt wird also das `onSelect` Event aktiv, bleibt der Blickkontakt bestehen aber bei jeder weiteren Abfrage das `onFocus` Event.
Das `onDeselect` Event schließlich ist das Gegenstück zum `onSelect` Event. Diese Methode wird genau dann aufgerufen, wenn bei einer Abfrage der Blick des Nutzers erstmals nicht mehr auf dem zuvor fokussierten Objekt liegt.

7.4.3 Optische Ähnlichkeit zu den Minigames

Auch unter visuellem Aspekt ist das Menü der Einstiegspunkt des Nutzers und insbesondere bei neuen Nutzern ist es der Punkt des ersten Kontakts. Der Nutzer sollte möglichst schnell auch optisch mit dem Spiel vertraut werden. Hierzu werden besonders markante Modelle der Minigames aufgegriffen. So wurde etwa das Modell des *Surfers* aus *Parcours* verwendet oder das Modell des Flugzeugs aus *Flugspiel*.

7.4.4 Warteraumfunktionalität

Bisher wurde das Menü unter den Aspekten der Hardwareeinbindung besprochen, ebenso wie seine Funktionalität im Einzelspielermodus. Doch auch im Mehrspielermodus hat das Menü spezielle Aufgaben. Hier dient das Menü als ein sogenannter Warteraum, einem Teil der Applikation, in welchem ein oder mehrere Clients auf weitere Mitspieler warten können. Für eine technische Aufarbeitung dieser Funktionalität sei auf den Abschnitt zum Thema Multiplayerintegration verwiesen.

7.5 Implementierung des Minigames "Parcours"

7.5.1 Übersicht der Skripte

Die Spiellogik verteilt sich zwar im Wesentlichen auf wenige Spielobjekte, dafür umfassen diese Spielobjekte eine Vielzahl von Skripten, die miteinander interagieren. Es folgt eine Übersicht dieser Skripte.

Das Surferobjekt umfasst diese Skripte:

- `CollisionHander`: Leitet Maßnahmen bei Kollisionen zwischen dem Spieleravatar und bestimmten Objekten aus der Spieleumgebung ein. Neben dem Großteil der Aktionen auf der audiovisuellen Ebene (Soundeffekte, „Flackern" des Surfers) werden hier wesentliche Aktionen der Spielmechanik durchgeführt. Zum Aufgabenfeld dieses Skriptes gehört vordergründig der Umgang mit dem *Zusammenstoß zwischen dem Spieler und einem Hindernis* - dass der Spieler in diesem Fall ein Leben verliert, ist hier kodiert. Im Fall eines Verlustes aller Leben wird hier insbesondere eine kurze Pause und die Verlagerung der Spielerposition initialisiert. Ebenso wird die *Durchquerung von Checkpoints* in diesem Skript verarbeitet, indem die Position und Ausrichtung an diesem Punkt gespeichert wird, falls der Spieler zu einem späteren Zeitpunkt die Fahrt an der Stelle wiederaufnehmen muss. Außerdem wird bei der Durchquerung eines Checkpoints der Weiterbau der Rennstrecke im Skript `LevelBuilder` angestoßen. Erreicht der Spieler die Ziellinie, signalisiert der `CollisionHandler` dies anderen Skripten, die diese Information benötigen.

- `PlayerMovement`: `PlayerMovement` ist ein Skript, welches sämtliche Belange steuert, die mit der Bewegung des Avatars zu tun haben. Die wichtigste Tätigkeit ist hierbei, den Input des Ergometers auf die Bewegung des Spieleravatares zu übertragen. Auch sind *Einschränkungen oder Modifikationen der Bewegung*

in diesem Skript von Bedeutung: Wenn der Surfer etwa mit den Streckenbanden, einer Sand- oder Eisfläche oder einem Turbo-Boost kollidiert, eine Kurve betritt oder die befahrene Strecke ihre Höhe ändert, wird hier die Bewegungsrichtung bzw. -geschwindigkeit entsprechend manipuliert.

- `SoundController`: Obwohl der Großteil der Soundeffekte bereits im `CollisionHandler` angestoßen wird, ist dieses separate Skript für die Steuerung der Fahrgeräusche zuständig. Diese Soundeffekte werden separat von denen im `CollisionHandler` behandelt, da sie zum einen nicht durch Kollisionen ausgelöst werden, und zum anderen, da sie eine eigene *AudioSource* verwenden, um die restlichen Soundeffekte nicht abzuschneiden.

Das Spielobjekt `SessionManager` bündelt die folgenden Skripte:

- `LevelBuilder`: Die zufällige Streckengenerierung wird vom `LevelBuilder`-Skript realisiert. Sämtliche Teilstrecken, deren Checkpoints und die unterschiedlichen Objekte zur Darstellung des Himmels werden diesem Skript als Variablen zugewiesen, um in der Methode für die Generierung der nächsten Teilstrecke genutzt zu werden. Hierbei ist der Streckenabschnitt zwischen zwei Checkpoints als Teilstrecke zu verstehen. Die komplexeste Aufgabe, die dieses Skript übernimmt, ist die Bewahrung der Konsistenz der Strecke unter den Anwendungen sämtlicher Mitspieler.

- *GUIHandler*: Neben dem eigentlichen Spielgeschehen innerhalb der 3D-Welt werden im Laufe des Spiels auf dem Bildschirm weitere Informationen wie die Anzahl der Leben, die Geschwindigkeitsleiste, die verstrichene Zeit, eventuell die verbleibende Zeit, der Punktestand und der aktuelle Spielstatus ausgegeben. Das `GUIHandler`-Skript kontrolliert diese Ausgaben. Da dieses Skript ohnehin schon die verstrichene Spielzeit misst, signalisiert es bei fortgeschrittener Zeit dem `LevelBuilder`, dass die Ziellinie generiert werden soll, um die vorgegebene Spielzeit einzuhalten.

- *Initializer*: Bei `Initializer` handelt es sich um ein kleines Skript, welches eine kurze Einführungssequenz zu Beginn des Spiels einleitet, bei der die Spielsteuerung noch abgeschaltet ist. Hiermit soll sich der Spieler in die Umgebung einfinden, bevor das Spiel startet.

7.5.2 Streckenabschnitte konsistent halten

Um zu gewährleisten, dass die zufällig generierte Strecke bei allen Teilnehmer eines Multiplayer-Rennens identisch ist, verwaltet das Skript `LevelBuilder` eine Vielzahl von Variablen und Methoden.

Übersicht über die Variablen:

- *GameObject[]* `alleStrecken`: Hier sind die Spielobjekte sämtlicher Teilstrecken in einer festen Reihenfolge gespeichert.
- *GameObject[]* `alleCheckpoints`: Hier sind die Spielobjekte der zugehörigen Checkpoints in derselben Reihenfolge gespeichert. Checkpoints werden separat verwaltet, weil eine Teilstrecke exakt an der Position des Checkpoints ihres Vorgängers angesetzt wird.
- *ArrayList* `strecken`: Die Auswahl der Teilstrecken soll fair erfolgen. Diese *ArrayList* enthält zunächst alle Teilstrecken. Die Auswahl der als nächstes zu generierenden Teilstrecke wird in dieser *ArrayList* getroffen und das ausgewähle Element wird anschließend daraus entfernt. Wenn die *ArrayList* leer ist, wird sie wieder mit sämtlichen Teilstrecken aus dem Array `alleStrecken` aufgefüllt. So wiederholt sich das Vorkommen von Strecken nur, wenn sämtliche Teilstrecken für den Streckenbau verwendet wurden.
- *ArrayList* `checkpoints`: Analog zu strecken. Wird eine Strecke ausgewählt bzw. entfernt, wird ihr zugehöriger Checkpoint ebenfalls ausgewählt bzw. entfernt.
- *ArrayList* `collected`: Eine temporäre Sammlung von Teilstrecken, von denen ein Kandidat als nächste Teilstrecke ausgewählt wird.
- *ArrayList* `trackHistory`: Eine komplette Auflistung der Teilstreckenfolge, aus der die gesamte Strecke besteht.
- *int* `historyBuildPosition`: Ein Index, der angibt, bis zu welcher Teilstrecke `trackHistory` bereits tatsächlich generiert wurde.
- *int* `checkpointCnt`: Gibt an, wie viele Checkpoints bisher durchlaufen wurden.

Basierend auf diesen Daten wird folgender Algorithmus angestoßen, sobald der Spieler einen Checkpoint passiert:

- Zunächst wird überprüft, ob die nächste fällige Teilstrecke bisher definiert wurde oder zumindest Kandidaten hierfür existieren. Ist dies nicht der Fall, wird als Kandidat ein zufälliges Element aus `strecken` ausgewählt. Mittels eines RPC-Calls werden alle weiteren Mitspieler angewiesen, diesen Kandidaten in ihrer `collected`-Datenstruktur abzuspeichern. Anschließend folgt eine Wartezeit von sechs Sekunden, da es nicht auszuschließen ist, dass ein anderer Mitspieler zeitgleich ebenfalls einen Checkpoint passiert hat und einen eigenen Kandidaten propagiert. Nach dieser Wartezeit wird erneut bei allen Mitspielern das folgende Verhalten per RPC-Call angestoßen:
  Die Kandidatenmenge `collected` wird sortiert, so dass diese bei jedem Mitspieler in der gleichen Reihenfolge geordnet ist. Das erste (nullte) Element dieser Liste ist nun der Kandidat, welcher für die nächste Teilstrecke ausgewählt wird. Dieser Kandidat soll nun der Datenstruktur `trackHistory` hinzugefügt und die ensprechenden Einträge aus den Datenstrukturen `strecken` und `checkpoints` entfernt sowie die Datenstruktur `collected` gänzlich geleert werden. Falls die Datenstrukturen `strecken` und `checkpoints` an dieser Stelle leer sind, werden sie an dieser Stelle wieder (mit allen Strecken und Checkpoints, die zu diesem Zeitpunkt nicht mit der Markierung *"in Use"* gekennzeichnet sind) aufgefüllt. Anschließend wird der Bau der nächsten Teilstrecke initialisiert.

- Ist die nächste fällige Teilstrecke beim Erreichen eines Checkpoints bereits bekannt, so wird der Bau der nächsten Teilstrecke sofort initialisiert.

- Der Bau der nächsten Teilstrecke selbst läuft nach folgendem Schema ab: Es werden nach dem Methodenaufruf nur dann Aktionen ausgeführt, wenn einerseits die Bedingung `(historyBuildPosition == checkpointCnt)` und andererseits die Bedingung `(trackHistory.Count >= checkpointCnt + 1)` erfüllt ist. Diese Bedingungen sollen sicherstellen, dass einerseits immer nur eine Teilstrecke im Voraus gebaut werden soll und andererseits die zu bauende Teilstrecke überhaupt schon fest definiert ist. Werden diese Bedingungen nicht erfüllt, erfolgt keine weitere Aktion und die Methode wird in Zukunft rechtzeitig erneut aufgerufen (entweder durch Passieren eines neuen Checkpoints oder durch RPC-Call eines anderen Mitspielers).
  Werden diese Bedingungen schließlich erfüllt, werden Position und Rotation der nächsten Teilstrecke so gesetzt, dass sie nahtlos an den letzten bisher gebauten Checkpoint anknüpfen. Darüber hinaus wird die neue Teilstrecke mit

der Markierung *"in Use"* versehen.
Es kann in diesem Algorithmus vorkommen, dass sich die nun gebaute Teilstrecke immer noch in der Datenstruktur `strecken` bzw. `checkpoints` befindet (ein Ausnahmefall beim Wiederauffüllen der leeren Liste) - was unerwünscht ist, da die es möglich macht, dass die selbe Teilstrecke zwei mal hintereinander gesetzt wird. Um dies zu verhindern, wird an dieser Stelle des Codes überprüft, ob das der Fall ist, und die Teilstrecke in einem solchen Fall aus den Datenstrukturen `strecken` und `checkpoints` entfernt.

7.6 Implementierung des Minigames "Flugspiel"

7.6.1 Assets

Für das Flugspiel wurden diverse 3D-Objekte und Animationen erstellt. Die Modellierung wurde in Autodesk 3ds Max 2014 und Autodesk Mudbox 2014 durchgeführt. Die Texturen wurden von Hand in Adobe Photoshop CC 2014 angefertigt. Ein kurzer Überblick folgt in den nachfolgenden Paragraphen.

7.6.1.1 Flugzeug Das 3D-Modell des Flugzeugs wurde diversen Comic-Flugzeugen nachempfunden. Es besteht aus mehreren Unterobjekten, die nach dem Box-Modelling Verfahren entstanden sind. Höhen-, Seiten- und Querruder sowie Propeller sind mit Keyframe Animationen versehen und können ebenfalls mit dem FBX-Dateiformat in Unity importiert werden. Die Texturen wurden in Photoshop gezeichnet und anschließend in Mudbox mit einem Umgebungsschatten versehen. Umgebungsschatten (*engl. Ambient Occlusion*) ist eine Shading-Methode, die in der 3D-Computergrafik verwendet wird, um mit relativ kurzer Renderzeit eine realistische Verschattung von Szenen zu erreichen. Das Ergebnis ist zwar nicht physikalisch korrekt, reicht jedoch in seinem Realismus oft aus, um auf rechenintensive globale Beleuchtung verzichten zu können.

7.6.1.2 Level Die Sammlung der Level-Assets (siehe Abbildung **??**) entstammt der Toon-Asset Sammlung aus dem Unity Asset-Store. Sie wurden einzeln zu einem weitläufigen Level arrangiert und mit einsammelbaren Items dekoriert. Alle Objekte weisen einen Collider auf, können also von dem Flugzeug nicht durchflogen werden. Mitunter muss der Spieler dadurch besondere Vorsicht walten lassen, um Items einzusammeln.

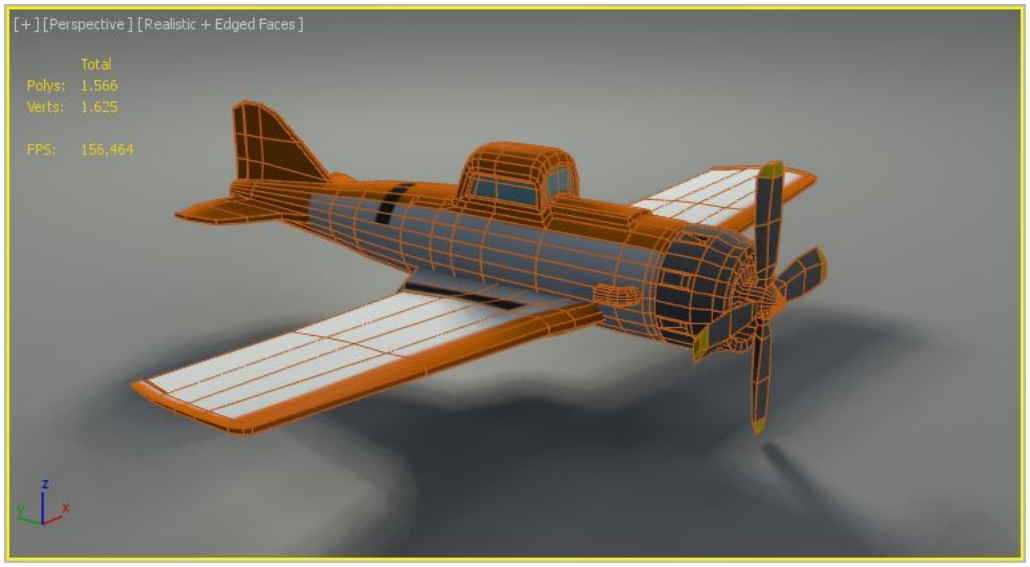

Abbildung 13: Ansicht des Flugzeug 3D-Modells in Autodesk 3ds Max. Das sichtbare Gitternetz beschreibt die 1566 Polygone. Eine möglichst realistische Ansicht inklusive Beleuchtung und Schatten hilft bereits in der Modellierungsphase um mögliche Fehler früh erkennen und beseitigen zu können.

Abbildung 14: Die Ansicht des Levels im Spiel. Das Flugzeug im Vordergrund überfliegt eine Graslandschaft mit Felsen und Bäumen. Im Hintergrund sind eine Ritterburg und mehrere Türme zu sehen.

7.6.2 Übersicht der Skripte

Die für das Flugspiel geschriebenen Skripte können in zwei Kategorien eingeteilt werden. Spielerbezogene Skripte, die auf dem Flugzeug-GameObject liegen und mit lediglich mit diesem interagieren, und Levelbezogene-Skipte, die an Ablauf des Spiels steuern.

Zuerst folgt die Erläuterung der Flugzeug-Skripte:

- `FlugzeugInput`: Das FlugzeugInput-Skript liest die benötigten Parameter zur Steuerung des Flugzeugs aus. Im Falle eines vorhandenen Hardware-Controllers (und damit eines Ergometers) werden horizontale und vertikale Geschwindigkeit ausgelesen. Falls kein Controller gefunden wird, verwendet das Skript die standardmäßige Input-Komponente der Unity-Engine, sodass das Spiel auch mit einem Gamepad oder der Tastatur gespielt werden kann.
  Aus diesen Parametern wird ein Richtungsvektor erzeugt. Aufgrund des leicht unterschiedlichen Spielprinzips, indem die Trittgeschwindigkeit nicht über die Geschwindigkeit des Flugzeugs, sondern über Steig- oder Sinkflug entscheidet, muss der Wertebereich der vertikalen Steuerung (von 0.0 bis 1.0) auf -1.0 bis 1.0 angepasst werden. Von dem ausgelesenen Wert wird dazu 0.5 subtrahiert und das ergebnis mit 2 multipliziert.
  Der finale Richtungsvektor wird dem AeroplaneController übergeben.

- `AeroplaneController`: Der AeroplaneController ist abgeleitet von dem Skript, das in den Unity-Sample-Assets bereits mitgeliefert wird, was auch den englischen Namen erklärt. in diesem Skript wird die Transformation, also Richtungs- und Positionsänderung des Flugzeugs, durchgeführt. Der im FlugzeugInput ausgelesene Richtungsvektor wird auf die bisherige Flugrichtung angewendet. Mittels linearer Interpolation lässt sich ein konstanter Übergang herstellen.
  Für einen Kurvenflug steht nur der Lenkausschlag des Ergometers zur Verfügung. Dieser könnte zwar auf das Seitenruder des Flugzeugs angewendet werden, allerdings ist eine Änderung der Hochachse des Flugzeugs für den normalen Anwender nicht grafisch genug, um an einen Kurvenflug zu erinnern. Zudem wird auch ein echtes Flugzeug, um ein SSchlingern"(*Gieren* in der Fligersprache) zu vermeiden, zuerst mit dem Querruder schräg gestellt (Rollen) um dann mit dem Höhenruder die Kurve einzuleiten. Um diese zwei Achsen mit nur einem Eingabeparameter zu simulieren, muss bei der Rotation des GameObjects zumidest teilweise vorgetäuscht werden.
  Ist die richtige Rotation gefunden, wird das GameObject um den Faktor von (konstanter) Geschwindigkeit und verstrichener Zeit seit dem letzten Update in Richtung der Längsachse verschoben.

- `FlugzeugAnimation`: Auch wenn die Animation des Flugzeugs in einer der vorherigen Skripte implementiert werden könnte, ist es gängige Praxis diese in ein separates Skript auszulagern. Die benötigte Logik ist sehr klein und leitet

hauptsächlich den Rotationsvektor an den Animation-Controller des Flugzeugs weiter. Dieser ist wiederum in einem graphischen Tool von Unity modelliert und wählt die abzuspielende Animation des 3D-Modells entsprechend aus. Ebenfalls können so Animationen überlagert werden (*Animation-Blending*; z.B. Querruder und Höhenruder gleichzeitig bewegen), obwohl diese im 3D-Modell als separate Bewegungsabfolgen abgespeichert sind.

- `AeroplaneAudio` und `BoostEffect`: Um die Immersion des Spielers weiter zu steigern werden Motorengeräusche, Propellergeschwindigkeit und Auspuffqualm abhängig von der Fluggeschwindigkeit und aufgesammelten Items beeinflusst. So lässt ein aufgesammeltes *Boost*-Item für 10 Sekunden eine graue Partikelwolke aus den zwei Auspuffen des 3D-Modells austreten. Ebenfalls steigt die Propellergeschwindigkeit und der Motorenlärm.

Zur Spiellogik gehören die folgenden Skripte:

- `SceneLogicFlugspiel`: Das SceneLogic Skript übernimmt die Initialisierung der Szene. Dazu gehören die Instanziierung des lokalen Spielers, die Zuweisung der Verfolger-Kamera zu diesem und die Erstellung der GUI. Im laufenden Spielbetrieb werden die eingesammelten Punkte des lokalen Spielers aufaddiert.

- `FlugspielGameOver`: Ein kurzes Skript zum Abfangen einer vorher festgelegten maximalen Spielzeit. Nach Ablauf dieser Spielzeit wird synchron mit den anderen Clients in einem Multiplayer-Spiel ein Game-Over Logo eingeblendet und zum Hauptmenü gewechselt.

- `ReturnToMenu`: Falls ein Spieler das Spiel frühzeitig abbrechen möchte, kann er durch dreisekündiges Gedrückthalten des Fire2Buttons am Ergometer den Abbruch einleiten und kehrt zum Hauptmenü zurück.

7.7 Anpassungen für Multiplayer-Spiele

Das Erstellen eines Netzwerk-Spiels erfordert viel Aufmerksamkeit auf einige sehr spezifische Details. Selbst in Unity, wo die meisten Aktionen leicht zu entwerfen und zu erstellen sind, bleibt Networking äußerst komplex. Auf den folgenden Seiten werden die Grundlagen von Netzwerk-Konzepten und die Unity-spezifische Ausführungen dieser Konzepte erklärt.

7.7.1 Grundlagen

Es gibt zwei gängige und bewährte Ansätze zur Strukturierung eines Netzwerk-Spiels. Diese Ansätze werden authoritative Server und non-authoritative Server genannt. Beide Ansätze beruhen auf einem Server, der die Clients verbindet und Informationen zwischen Ihnen versendet. Sie bieten auch mehr Privatsphäre für den Endbenutzer, da die Clients nie direkt miteinander verbunden sind.
Der authoritative Server muss die komplette Simulation der Spielwelt durchführen, die Spielregeln anwenden und die Client-Player-Eingaben verarbeiten. Jeder Client sendet seine Eingaben (in Form von gedrückten Tasten oder um angeforderte Funktionen zu bearbeiten) an den Server und erhält fortlaufend den aktuellen Stand des Spieles vom Server. Der Client führt nie eine Spiel-Logik aus. Mit authoritativen Servern findet eine lokale Bewegung erst statt, wenn der Server reagiert hat. Dies kann zu unrealistisch aussehenden Bewegungen führen.
Ein non-authoritative Server kontrolliert die Ergebnisse der einzelnen Benutzereingaben nicht. Jeder Client verarbeitet die lokalen Benutzereingaben und die davon abhängige Spiellogik selbst und sendet die Ergebnisse zum Server. Die Clients sind die Eigentümer ihrer Spielerobjekte und damit die einzigen, welche lokale Änderungen dieser Objekte zum Server senden können. Der Server synchronisiert alle Aktionen. Dies ist einfacher aus der Entwickler-Perspektive zu implementieren, da der Server hauptsächlich Nachrichten zwischen den Clients versendet und wenig Berechnungen durchführen muss.
Der für Unity benötigte Weg der Kommunikation zwischen Client und Server sind sogenannte *Remote Procedure Calls* (RPCs). Sie werden verwendet, um Methoden auf verschiedenen Rechnern des Netzwerk-Spiels aufzurufen. Clients können RPCs zum Server schicken und der Server kann RPCs zu einem oder mehreren Clients schicken. RPCs werden am häufigsten bei selten genutzten Aktionen verwendet. Zum Beispiel wenn ein Client eine Tür öffnet. So sendet der Client per RPC, dass die Tür geöffnet wurde und der Server sendet per RPCs an alle Clients das die Tür offen ist.

7.7.2 Verbindungsaufbau

Bevor ein Netzwerk-Spiel gestartet werden kann, müssen die Rollen der beteiligten Computer festgelegt werden. Auf einem Computer muss ein Server initialisiert werden. Dieser kann eine ebenfalls Teilnehmer am Spiel sein oder sich speziell nur um die Verwaltung des Spiels kümmern. Um den Server zu erstellen, wird die *Network.InitializeServer()* Methode aus einem Skript aufgerufen. Wenn zu einem vor-

handenen Server als Client verbunden werden soll, kann *Network.Connect()* verwendet werden.
Bereits jetzt synchronisiert Unity alle GameObjects, die ein NetworkView-Component besitzen. So können Transforms, Rigidbodies oder Animationen sehr leicht unter den Clients kommuniziert werden. Für komplexere Vorgänge werden weitere Skripts benötigt.

7.7.3 Übersicht der Skripte

- `GameLogic`: Die GameLogic ist die zentrale Komponente, die den Zustand des gesamten Spiels überwacht. Beim Start wird der Befehl DontDestroyOnLoad() ausgeführt, was bewirkt, dass das GameObject inklusive Skripts bei einem Levelwechsel in die nächste Szene übernommen wird. Für den lokalen Spieler werden RehaWeb-spezifische Daten, wie Workout- oder Profil-ID, sowie das eigene Spieler-Objekt gespeichert. Die Netzwerk-Funktionen sind Server() (zum Starten eines Servers), Client(string host) (zum Verbinden zu einer bekannten IP), StartGame() (zum Starten eines Minigames; nur für den Server aktiviert) und die per RPC aufrufbare Funktion LoadLevel() die vom Server aus auf allen Clients aufgerufen wird um ein bestimmtes Minigame zu laden oder zurück ins Menü zu wechseln.

- `SceneLogic`: Für jedes Minigame existiert ein zusätzliches Skript, das Minigame-spezifische Daten speichert und die Verwaltung des Levels übernimmt. Dazu gehören gesammelte Punkte im Level oder auch die Zuweisung verschiedener Farben zu den Spieler-Objekten zur besseren Unterscheidung.

- `NetPlayerLogic`: Das NetPlayerLogic-Skript ist am Spieler-Objekt angehängt und muss zu jedem Spielstart überprüfen, welches Spieler der Lokale ist. Alle nicht lokalen Spieler sollen von dieser Spielinstanz nicht beeinflusst werden und benötigen aus diesem Grund auch nur eine Positions- und Rotations-Komponente. Der Rest der Komponenten kann gelöscht werden. Das Löschen in der lokalen Spielinstanz beeinflusst die anderen Clients nicht.

- `NetworkInterpolatedTransform`: In der Entwicklung des Flugspiels ist es aufgefallen, dass die Übertragungsrate von 15 Ticks pro Sekunde nicht ausreicht, um die Flugzeuge flüssig darzustellen. Ein Tick steht für ein komplettes Abbild des Spiels, das der Server zur den Clients schickt. Ein lokaler Spieler sieht also nur 15 mal in der Sekunde eine Positionsänderung der anderen Spieler und

damit ein deutliches Ruckeln der Objekte.
Eine Erhöhung der *Sendrate* ist zwar möglich, aber erhöht auch die Belastung für den Server. Da es beim Flugspiel auch keine plötzlichen Ausreißer in der Position oder Flugrichtung gibt, ist es sinnvoller eine Interpolation auf den Clients vorzunehmen. Falls der Fall eintritt, dass ein Netzwerk-Paket verloren geht, muss das Flugverhalten extrapoliert werden. Es wird dazu angenommen, dass sich die Position des Flugzeugs um ein gleiches verändert wie zwischen den zwei vorausgegangenen Ticks. Diese Berechnungen übernimmt das Skript NetworkInterpolatedTransform.

8 Evaluierung

8.1 Versuchsaufbau

Durchgeführt wurde die Evaluation in dem Fitnessförderwerk Dortmund. Von Montag dem 6.10. bis Samstag den 11.10. waren täglich zwei PG-Mitglieder für mindestens zwei Stunden anwesend, um die Evaluation durchzuführen. Die Freiwilligen, die unser System ausprobieren wollten, konnten sich über das Buchungssystem des Fitnessförderwerks ab dem 2.10. hierfür anmelden. Um möglichst viele darauf aufmerksam zu machen, wurde seitens des Fitnessförderwerks ein Rundbrief verschickt. Auch nicht Angemeldete, aber im Fitnessförderwerk anwesende Freiwillige haben bei der Evaluation teilgenommen. Insgesamt haben 40 Probanden das System getestet.

Abbildung 15: Fitnessförderwerk

Zur Evaluation wurde das neuere Ergometer verwendet und in Mitte des Fitnessraumes platziert. Davor wurde ein Rollwagen positioniert, auf dem der Computer, samt Monitor und den restlichen Geräten aufgebaut wurde. Die Probanden konnten, falls sie nicht mit der 3D-Brille spielen wollten, auch auf dem Monitor die Spiele ausprobieren. Für die Soundeffekte wurde ein Lautsprecher angeschlossen.

Das Spiel wurde immer direkt (d.h. ohne RehaWeb) gestartet. Der Proband konnte

sich mit der 3D-Brille eines der beiden Spiele aussuchen, dabei wurde das Parcours-Spiel als leichterer Einstieg in die Systemumgebung empfohlen. Es wurde keine feste Zeit für das Spielen der Spiele festgelegt. Wollte ein Proband das Spiel wechseln bzw. beenden, hat er dies mitgeteilt.

Jeder Freiwillige wurde von uns über das Projekt und dessen Evaluation aufgeklärt. Nachdem ein Proband sein Einverständnis erklärt und den Pre-Fragebogen ausgefüllt hat, konnte er mit dem Training beginnen. Über mögliche Nebenwirkungen, wie Übelkeit oder Schwindel, wurden die Probanden informiert. Nach dem Spielen wurden die Probanden gebeten, einen Post-Fragebogen auszufüllen.

8.2 Auswertung

Die Evaluation des kompletten Systems fand im Fitnessförderwerk der Universität Dortmund statt. Insgesamt meldeten sich 40 Probanden - 34 männliche und 6 weibliche -, um das *CyberReha*-Projektes zu testen und Feedback zu geben. Die Altersreichweite lag zwischen 18-56 Jahren, wobei insgesamt ein Altersdurchschnitt von 25 Jahren herrschte, das heißt mehr jüngere Leute teilgenommen haben. Durch 2 Fragebögen, dem sogenannten Pre- und Postinterview wurden die für uns wichtigen Aspekte behandelt und anonym ausgewertet. Dabei wurden Fragen aus dem Bereich der Gamification, des System Usability Scale, dem eHealth, sowie wirtschaftliche Aspekte berücksichtigt.
Die Fragestellung die verfolgt wurde war, ob das System zur tatsächlichen langzeitigen Motivationsteigerung für Ergometertraining geeignet ist und ob es erfolgreich gelungen ist Standardtrainingsverfahren durch die Einbindung der Informationstechnologie interessanter für den Trainierenden zu gestalten.

8.2.1 Gamification

Die Fragestellungen im Bereich der Gamification bezogen sich im Wesentlichen auf die Auswertung und Analyse verschiedener Aspekte der beiden getesteten Spiele *Parcour* und *Aviation*.
In Abbildung 16 ist die Auswertung der beiden Spiele zu sehen. Die Skala hier liegt in einem Bereich von 1 bis 5, wobei 1 einen schlechten und 5 einen guten Wert repräsentiert.
In beiden Spielen liegen ähnlich gute Auswertungen im Bereich der Verständlichkeit, des Spielspaß sowie des 3D-Erlebnisses vor. Die Unterschiede sieht man im

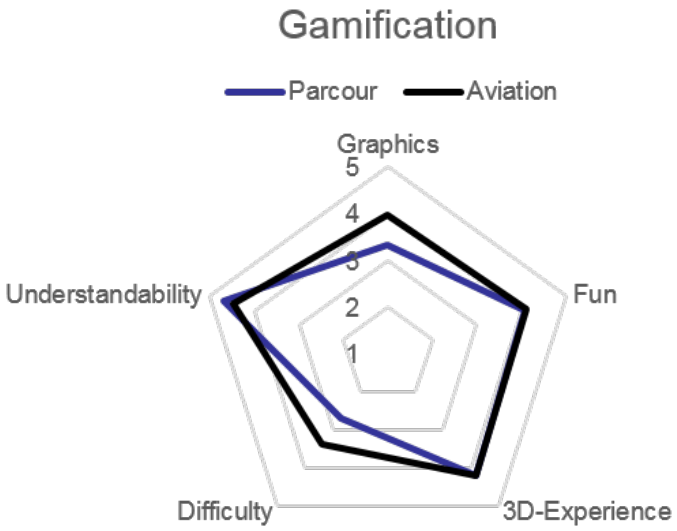

Abbildung 16: Gamification

Bereich der Graphik und der Schwierigkeit. Während *Parcour* eine immer noch zufriedenstellenden Graphik aufweist, gefällt den Probanden die Graphik von *Aviation* eindeutig besser. Dafür ist die Schwierigkeit des Flugspiels jedoch erheblich größer wahrgenommen worden.
Abbildung 17 zeigt die Einordnung in die Borg-Skala. Die Borg-Skala gibt das körperliche Belastungsempfinden bei verschieden Belastungsformen an. Das Belastungsempfinden kann dabei von 6 *sehr sehr leicht* bis 20 *zu stark, geht nicht mehr* eingestuft werden.
Die Probanden gaben ihr Belastungsempfinden in einer Reichweite von 11-13 Punten an, was einem *leicht* bis *etwas anstrengend* entspricht und durchaus im gewünschten Rahmen liegt.

8.2.2 System Usability Scale

Der *System Usability Scale (SUS)* Fragebogen umfasst 10 Fragen mit einer fünfgliedrigen Antwortskala im Format *stimme nicht zu* bis *stimme zu*. Die Fragestellungen sind stark an die Inhalte der *EN ISO 9241 Norm*-Definition von Usability angelehnt und das Ergebnis gibt einen groben Überblick über die Güte der Struktur und der subjektiven Usability des zu testenden Systems.
In Abbildung 18 ist die Darstellung der 10 Fragen mit ihren Durchschnittspunkten zu sehen. Die Skala hier liegt in einem Bereich von 1 bis 5, wobei 1 *stimme nicht zu* und 5 *stimme zu* entspricht.

Exertion	RPE scale
Very very light	6
	7
Very light	8
	9
Fairly light	10
light	11
	12
Somewhat hard	13
	14
hard	15
	16
Very hard	17
	18
Very very hard	19
Maximum of hardness	20

Abbildung 17: Borg-Skala

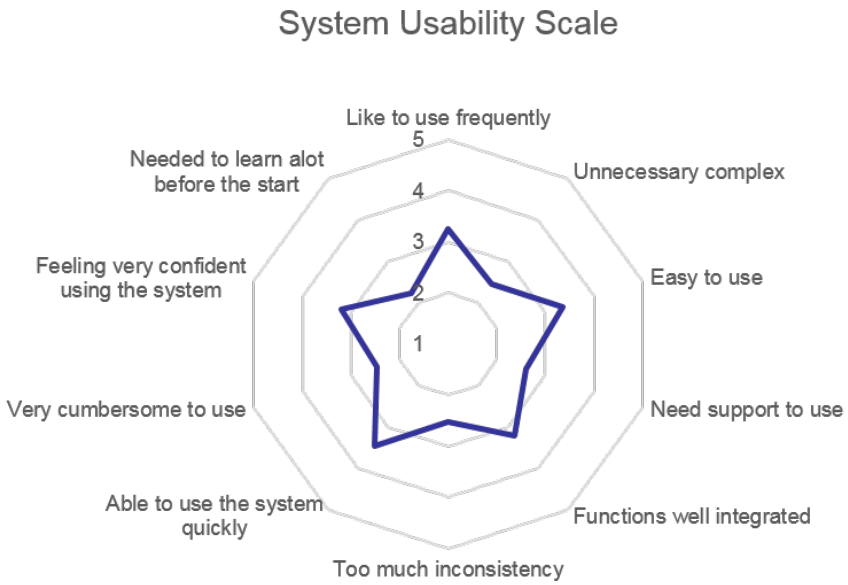

Abbildung 18: SUS-1

Die sternförmige Anordnung ist durch die Art der 10 Fragen zu erklären, denn es gibt zu jeder Frage eine negierte oder gegensätzliche Form der Aussage. Zum Bei-

spiel *unnötig komplex* und *einfach zu benutzen.*
Abbildung 19 zeigt die Auswertung des *System Usability Scale* Fragebogens für die einzelnen Testpersonen. Resultierend ergibt sich für jede Person eine subjektive *System Usability Score*, welche einen Wert von 0 bis 100 annehmen kann. 100 entspricht dabei der Interpretation von einem perfekten System mit sehr guter Usability, wobei 0 das Gegenteil darstellt.

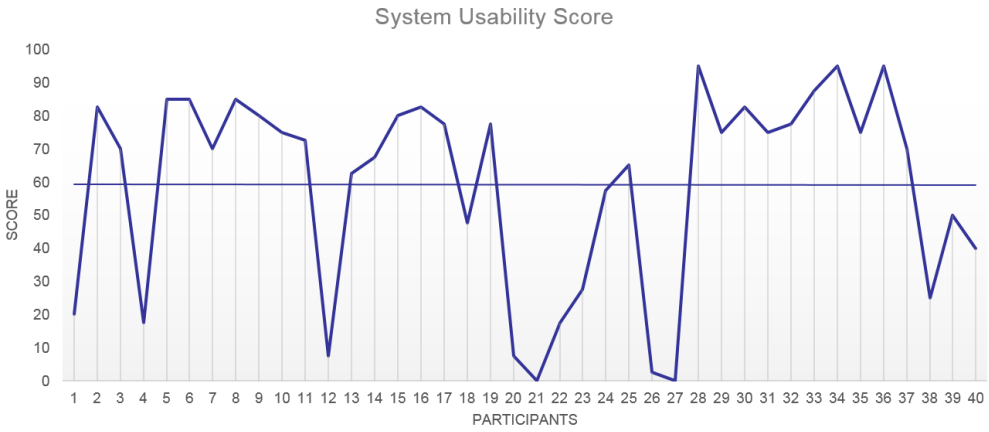

Abbildung 19: SUS-2

Der durchschnittliche Score liegt bei ungefähr 60,5 Punkten, was einem Usability-Grad von *OK/Good* entspricht (nach: Bangor,Kortum, Miller; 2008; P.118). Der höchste Wert ist mit 95 Punkten sehr nah an der Maximalpunktzahl, der niedrigste Wert 0 ist als Ausreißer anzusehen. Alle Ausreißer im unteren Bereich der Punkte sind auf die Oculus Rift zurückzuführen. Die Probanden die Probleme mit Oculus Rift hatten, haben generell das ganze System schlechter bewertet.

8.2.3 eHealth

Folgende Abbildung 20 zeigt die Auswertungen des Bereichs *eHealth* und stellt einige Ergebnisse zu Beantwortung unserer Ausgangsfragen dar.
Das CyberReha-System steigerte die Motivation von 90 % der Probanden im Vergleich zu normalen Ergometertraining. 92 % wären durch einen Multiplayermodus noch zusätzlich motiviert. Circa 80 % der Testpersonen konnten es sich vorstellen das System für ihr Training - auch von zu Hause - zu benutzen. Auch eine Rangliste wurde motivationssteigernd angegeben. Die Probanden waren im Schnitt bereit für das Home-Training 297,50 Euro für die Applikation auszugeben. Außerdem motivierte die Applikation 89 % der Leute zu weiteren Trainingseinheiten mit dem System. Generell ist die Idee Ergometertraining in spielerischer Form mit der IT zu verknüp-

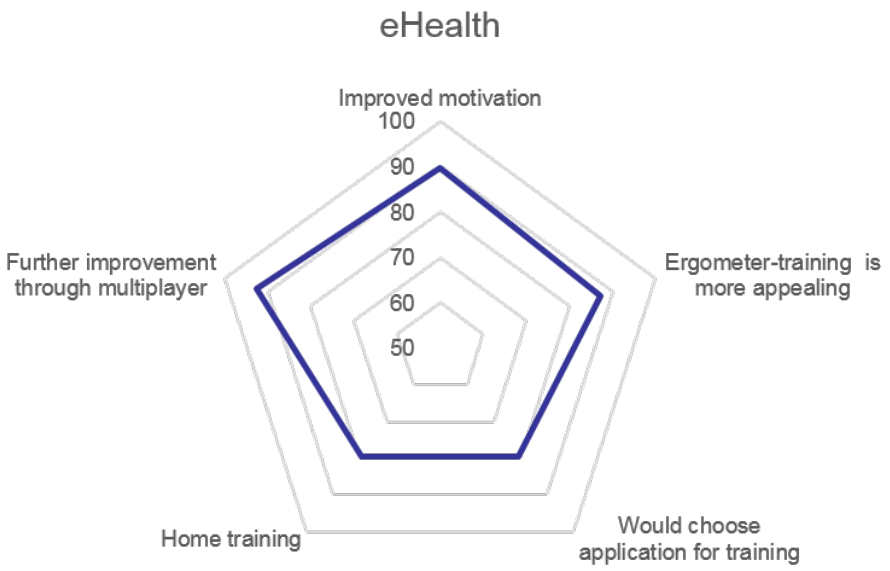

Abbildung 20: eHealth

fen einheitlich auf positive Resonanz gestoßen und die Probanden hatten während der Test-Läufe sehr viel Ehrgeiz gezeigt, die beiden Spiele erfolgreich abzuschließen.

8.2.4 Probleme

Auch wenn das System zu größten Teilen sehr positiv aufgenommen wurde, gibt es dennoch einige wenige Kritikpunkte. Im wesentlichen hat die 3D-Brille bei circa 33 % der Probanden ein Unwohlsein herbeigeführt, welches sich dann auf den kompletten restlichen Fragebogen ausgeweitet hat. Natürlich sind dementsprechend auch die Erfahrungen dieser Testpersonen als subjektiv schlechter anzusehen. Ein weiterer Kritikpunkt an die Applikation ist, dass die Grafik der Spiele zwar als ausreichend und von vielen sogar als gut wahrgenommen wurde, jedoch aufgrund der geringen Auflösung der 3D-Brille verfälscht wurde. Dies zeigt demnach aber, das die meisten Probanden von unserer Applikation überzeugt waren und die negativen Kommentare alle auf die proprietäre Brille zurückzuführen sind.

8.2.5 Zusammenfassung

Zusammenfassend lässt sich erkennen, das das Konzept von *CyberReha* sehr gut angenommen wurde und unsere Zielsetzungen der Motivationssteigerung sowie der Anreiz zu weiteren Trainingseinheiten durch das System erfüllt wurde. Auf die Problematik der 3D-Umgebung mittels der Brille müsste in der weiteren Entwicklung

ein besonderer Augenmerk gelegt werden und eventuell auf neue ausgereifte Technik in diesem Bereich gewartet werden. Somit könnten die jetzt noch negativen Punkte vollständig behoben werden.

9 Zusammenfassung und Ausblick

9.0.6 Zusammenfassung

An dieser Stelle soll der Ablauf der Projektgruppe zusammengefasst werden.
Die Zielsetzung war die Steigerung der Motivation von Patienten mit Herz-Kreislaufschwächen für ein Ergometertraining ,unter Verwendung einer 3D-Brille sowie Interaktionen zwischen den Patienten in einer virtuellen Welt. Die Idee bestand darin, das Training der Patienten mit einem 3D-Spiel zu verbinden und auf dieser Art dem Patienten mehr Spaß und Abwechslung am Training zu ermöglichen.
Als Vorgaben dienten die Verwendung eines Ergometers sowie einer 3D-Brille. Nicht zu vernachlässigen ist der Einfluss des Arztes auf das Training des Patienten.
Umgesetzt wurden diese Vorgaben durch zwei Spiele (Aviation und Parcour), welche für die 3D-Brille und das Ergometer implementiert wurden. Zusätzlich wurde das Lenkrad eines Ergometers umgebaut und mit einem Controller versehen, damit die Patienten größere Interaktionen in der virtuellen Welt durchführen können. Die Trainingsdaten jedes Patienten werden gesammelt und dem zuständigen Arzt zur Auswertung zur Verfügung gestellt. Des Weiteren kann der Arzt durch Vorgaben der Schwellenwerte für das Ergometer das Training des Patienten beeinflussen. Die implementierten Spiele stellen eine Laststeuerung bereit, welche auf Basis dieser Schwellenwerte die Lastregelung übernimmt.

Bei der Umsetzung der Ziele traten Problemstellung in der Reaktionszeit des Ergometers sowie die Ausführung der Spiele durch einen Client-Rechner ohne vorinstallierte Software auf. In der Projektgruppe wurden diese Problembereiche weitgehend gelöst. Die schlechte Auflösung sowie eine zulange Verwendung der Brille führte bei den Probanden oft zu Gleichgewichtsstörungen.

9.1 Ausblick

Bei der Weiterführung des Projektes können bessere Belohnungssysteme in den Spielen für die Patienten eingebaut werden, welche die Lust und den Spaß an der Verwendung der 3D-Brille bei den Patienten steigern.
Es sollte eine bessere 3D-Brille eingesetzt werden um Gleichgewichtsproblemen entgegen zu wirken.
Mit einem Ergometer dessen Reaktionszeit nicht zu groß ist, lassen sich bessere Spieleideen umsetzen.
Und natürlich sollten die Spiele selbst Spaß machen.

A Weitere Abbildungen

Abbildung 21: UML-Klassendiagramm der Hardware-Integration