TECHNISCHE HOCHSCHULE MITTELHESSEN

Fachbereiche MND und IEM

BACHELORARBEIT

Entwicklung und Evaluation neuer Interaktionsmöglichkeiten mit First-Person 3D-Simulationen und Games auf Touchscreen-Tablets unter Einbeziehung von Methoden der Usability

Vorgelegt von: Jan Salge
geboren am: 02.03.1984 in: Gießen

Referent der Arbeit: Prof. Dr.-Ing. Cornelius Malerczyk
Korreferentin der Arbeit: Dipl.-Math. (FH) Sabine Langkamm

Friedberg, 2011

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorgelegte Bachelorarbeit zum Thema

Entwicklung und Evaluation neuer Interaktionsmöglichkeiten mit First-Person 3D-Simulationen und Games auf Touchscreen-Tablets unter Einbeziehung von Methoden der Usability

eigenständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen, Darstellungen und Hilfsmittel benutzt habe.

Alle Textstellen, die wortwörtlich oder sinngemäß anderen Werken oder sonstigen Quellen entnommen sind, habe ich in jedem einzelnen Fall unter genauer Angabe der jeweiligen Quelle als Zitat gekennzeichnet.

Unterschrift

Danksagung

Ich möchte diese Gelegenheit nutzen, mich bei allen Beteiligten und Helfern dieser Arbeit zu bedanken.

Vor Ihnen liegt eine Arbeit, die nicht ohne die Beteiligung von freiwilligen Helfern möglich gewesen wäre. Die Durchführung von Usability-Tests kann nicht ohne Probanden gelingen. In diesem Zusammenhang fanden sich zehn mutige Studenten ein, bei denen ich mich besonders an dieser Stelle bedanken möchte. Ohne euch wäre es nicht möglich gewesen diese Arbeit fertig zu stellen und eure kreativen Impulse sind es, die zum Gelingen dieser Arbeit beigetragen und den Befragungen einen Sinn gegeben haben. In diesem Zusammenhang danke ich auch den Dozenten, die so freundlich waren, mich Studenten aus ihren Veranstaltungen entführen zu lassen.

Zudem gilt mein besonderer Dank Sabine Langkamm vom Fachbereich MND, die neben ihrer Tätigkeit als Betreuerin für diese Arbeit mit unermüdlicher Ausdauer immer wieder organisatorische Probleme gelöst und Probanden für die Tests in Friedberg gefunden hat. Ich hoffe, dass wir auch in Zukunft so gut zusammen arbeiten werden, und erhoffe mir in baldiger Zukunft auch etwas für diesen investierten Aufwand zurück geben zu können.

Ich danke außerdem allen, die für mich am Korrekturlesen dieser Arbeit beteiligt waren, allen voran meiner Mutter Petra Salge sowie meiner ganzen Familie, ohne deren Unterstützung mein Studium nicht möglich gewesen wäre.

Schließlich danke ich meinen engsten Freunden, die mich in der anstrengenden Zeit unterstützt haben: Carolina Klatt, Christoph Weber, Lars Händel, Rafael Bienia, Tom Heiden - Ohne den Rückhalt, den echte Freunde bedingungslos und unermüdlich immer wieder gewähren, wäre man nicht befähigt etwas auf dieser Welt zu erreichen.

In diesem Sinne,

Vielen Dank!

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Seit Jahren kann eine ständig wachsende Entwicklung in der Gesellschaft wahr genommen werden: Die Menschen umgeben sich mit mobilen technischen Helferlein, im populistischen Sprachgebrauch oftmals „Gadgets“ genannt. Ganze Webportale widmen sich dem technischen Spielzeug, alle Medien berichten darüber. So wurde aus dem klassischen Telefon zunächst das Mobiltelefon und letztlich das Smartphone. Im Lager der PCs entwickelt sich der Trend ebenfalls analog zunehmend in Richtung mobile Computer - was mit Netbooks begann mündet derzeit in einem anhaltenden Trend zum Tablet. Dabei werden die Geräte von Jahr zu Jahr komplexer und vor allem leistungsfähiger. So ist es mittlerweile möglich, ganze 3D Welten auf ihnen darzustellen - immer mehr Entwickler beschäftigen sich neben klassischer Anwendungssoftware daher mit der Entwicklung von Spielen. Zunehmend ist auch eine Entwicklung weg von 2D Casual Games hin zu den klassischen Video- und PC-Spielen ebenbürtigen komplexen 3D Anwendungen erkennbar. Aber auch andere Anwendungsbreiche von 3D kommen nicht zu kurz. Denkbar und auch sinnvoll sind ganze begehbare Visualisierungen, beispielsweise von Architektur oder komplexen Zusammenhängen. Auch hier spielt das Tablet einen enormen Vorteil gegenüber klassischen tragbaren Computern aus - es ist wesentlich mobiler, wesentlich weniger „sperrig“ (siehe Abb. 1.1).

Die Mobilität hingegen geht allerdings mit einer deutlichen Veränderung der Bedienung einher. Typischerweise ersetzt ein Touchscreen sämtliche anderen Eingabegeräte. So wird genauso die Maus zum „Zeigen und Klicken“ wie auch eine Bildschirmtastatur emuliert. Hier stehen die Entwickler allerdings vor neuen Herausforderungen: Die Bedienung durch den Nutzer ist völlig anderen Regeln unterworfen. So werden bisherige Konzepte oftmals als träge oder unpräzise wenn nicht sogar verwirrend wahrgenommen. Viele potentielle Nutzer machen einen Bogen um derartige Applikationen der Schwierigkeiten wegen, die sie mit der Steuerung haben. Dabei erscheinen die Grundvorraussetzungen eigentlich geradezu ideal - Menschen benutzen ihre Hände und Finger um mit ihrer gesamten Umwelt zu interagieren - warum wird aber dennoch nach wie vor eine Maus fast immer dem Touchscreen vorgezogen? So gelten offenbar zu einem beträchtlichen Anteil andere Regeln im Bereich der Usability als beispielsweise bei der typischen klassischen PC-Anwendung. Insbesondere natürlich, wenn es um die Navigation und Interaktion mit 3D-Welten geht,

Abb. 1.1: iPad - Quelle: Apple

bei denen Übersicht, Orientierung und Bedienelemente (sowie natürlich deren Design und Funktion) entscheidend über Nutzen bzw. bei Games Spielspaß entscheiden. Eine Visualisierung, in der man sich nicht zurecht findet ist genauso wenig nützlich, wie ein Spiel, dessen Bedienung so schwer ist, dass sich der Nutzer mit der Steuerung und nicht dem eigentlich Game beschäftigt.

Teile der Branche indes reagieren bereits auf die genannten Schwierigkeiten. Die Firma Ten One Design hat unlängst einen Joystick (Fling, siehe 1.2) für das iPad herausgebracht, der die Probleme der Nutzer mit den Kontrollelementen in Spielen verringern soll. Allerdings liegt es recht nahe, dass hiermit lediglich die Symptome eines Problems bekämpft werden, dessen eigentliche Wurzel in der Softwareentwicklung zu suchen ist. Somit ist das Gerät sicherlich bei der aktuellen Situation als nützlich zu bezeichnen, allerdings sicherlich eher als Workaround, denn als tatsächliche Lösung anzusehen.

1.2 Problemstellung und Zielsetzung

Diese Arbeit beschäftigt sich mit der Interaktion mit 3D-Welten auf modernen Multitouch-Tablets. Insbesondere soll dabei der Fokus auf die Art von Anwendung gelegt werden, bei der der Nutzer wie aus eigenen Augen die Szenerie betrachtet, der sogenannten First Person Perspektive. Dank der einzigartigen Kombination aus Sensoren und Eingabegeräten in besagten Tablets können und müssen alte vom PC bekannte Steuerungskonzepte

Abb. 1.2: Fling Joystick - Quelle: Ten One Design

neu überdacht werden. So haben die allermeisten dieser Geräte neben dem offensichtlich Touchscreen eine Reihe von Lage- und Beschleunigungssensoren, die gerade im Bereich 3D-Applikation eine wichtige Rolle spielen könnten. Diese bieten einen weiteren Eingabekanal, der gleichzeitig mit dem Touchscreen-System quasi ständig zur Verfügung steht. Hierbei stellt sich die Frage, wie die Lagesensorik als zusätzliche Steuerungsinstanz die Bedienung vereinfachen oder vielseitiger gestalten kann. Gerade durch den Verzicht auf hardwareseitige Buttons, die zur Eingabe genutzt werden können, ist dieser zusätzliche Steuerungskanal außerordentlich wertvoll - will aber aufgrund der Ungewohntheit für den Nutzer und der speziellen Charakteristiken solcher Sensoren richtig in das Gesamtkonzept integriert sein.

Aus der PC-Anwendungsentwicklung sind ausgereifte Methoden und Kriterien zur Untersuchung der Usability - also der Nutzbarkeit und Zugäglichkeit einer Anwendung für Benutzer bekannt. Diese Kriterien sind nicht nur für typische Büroanwendungen, sondern auch für 3D Spiele und Visualisierungen über die Jahre erarbeitet worden. Die Frage drängt sich auf, inwiefern diese Regeln und Methoden mit den veränderten Grundvorraussetzungen bei Tablets als übertragbar angesehen werden können und welchen Einfluss die andersartigen Eingabegeräte als auch die Haltung des Geräts sowie das typischerweise durch die eigenen Hände eingeschränkte Sichtfeld des Nutzers haben. Am Beispiel aktueller iPad Games werden diese Methoden und Evaluierungstechniken, sowie möglicherweise notwendige Änderungen an das veränderte Umfeld der Tablets untersucht. Bekannte Techniken und generelle Grundregeln werden zusammengefasst und anschließend sofern nötig an das veränderte Umfeld angepasst. Mit den herausgearbeiteten und abgeleiteten Kriterien wird anschließend an konkreten Beispielen untersucht, wann ein Bedienkonzept

Abb. 1.3: UDK Logo - © 2010 Epic Games, Inc.

Abb. 1.4: Epic Citadel, Interaktive Tech-Demo für das UDK auf iOS - © 2010 Epic Games, Inc.

auf Tablets als intuitiv und angenehm empfunden wird und wann nicht. Diese Untersuchungen und Methoden erzeugen ein konstruktives und differenziertes Feedback für den Entwickler, mit dessen Hilfe neue Konzepte und Optimierungspotentiale ermittelt werden können. Dem Entwickler wird ein Regel- und Methodensatz an die Hand gelegt um so die eigene Applikation besser an die Hardware und das Umfeld anpassen zu können.

Schließlich ist auch die technische Umsetzung einer derartigen Applikation, gerade unter Berücksichtigung derartiger Kriterien, sehr aufwändig. Fast alle modernen 3D Applikationen werden - sowohl auf dem PC als auch auf Tablets - auf Basis eines technischen Grundgerüsts, einer sogenannten Engine, aufbauend entwickelt. Die Engine übernimmt neben dem Rendering - also der Ausgabe der 3D Umgebung auf dem Bildschirm auch im Hintergrund viele weitere Aufgaben. Dem Entwickler wird ein Satz aus Werkzeugen an die Hand gegeben die ihm erleichtern sollen die Anwendung umzusetzen. Allerdings sind auch diese Werkzeuge außerordentlich komplex.

Ein Beispiel einer solchen Engine ist die Unreal Engine - eine bereits seit Jahren aus dem PC-Bereich bekannte und sehr erfolgreiche 3D-Engine. Für independent Entwickler ist diese frei verfügbar im Internet unter dem Namen UDK. Mit Hilfe des UDK (Unreal Development Kits) wird ein Spielprototyp unter Einbeziehung der Erkenntnisse aus den vorherigen Kapiteln entwickelt. Am Beispiel eines professionellen und anerkannten Basissystems werden somit die Folgen aus den gewonnenen Erkenntnissen direkt geprüft und nachvollzogen. Im Rahmen der Thesis wird ein Prototyp zum Testen und Evaluieren von neuen Steuerungskonzepten sowie innovativen Interaktionsmöglichkeiten mit der 3D-Welt implementiert. Anhand des praktischen Beispiels können diese neuen Methoden getestet und altbekannte Methoden verbessert werden. Ebenfalls wird das im UDK integrierte visuelle Scriptingsystem (genannt Kismet) vorgestellt, dass es ermöglicht schnell und effizient Änderungen am Prototypen vorzunehmen, um so den Entwicklungs- und Testprozess zu beschleunigen. Ziel ist es diesen Interface-Prototyp sowie seine Evaluation mit Usability-Methoden soweit zu optimieren, dass der Gesamtprozess als fundierte und effiziente Basis für Entwicklungsprojekte komplexerer Applikationen dienen kann. Parallel dazu werden neue innovative Ideen und Herangehensweisen im Bereich Interfacedesign entwickelt und ebenfalls direkt mit den erarbeiteten Methoden evaluiert. Mit Blick auf andere bereits auf dem Markt befindliche Beispiele wird schließlich, mit Hilfe von speziell für 3D Games auf Tablets im Zuge dieser Thesis entwickelten Methoden, an Probanden die Effizienz der Maßnahmen abschließend untersucht und bewertet.

1.3 Zusammenfassung der Ergebnisse

Diese Arbeit beschäftigt sich mit dem komplexen Problem der Steuerung von 3D-Anwendungen auf Touchscreen Devices. Dabei wird insbesondere auf direkte Steuerungskonzepte eingegangen, in der eine menschenähnlich agierende Spielfigur mittels Eingaben kontrolliert wird. Diese meistens in der First-Person-Perspektive nutzbaren Programme sind seit vielen Jahren ein wichtiges und erfolgreiches Konzept im Bereich Gaming und auch im produktiven Einsatz, beispielsweise für Architekturvisualisierungen oder Lernprogramme.

Im zweiten Kapitel werden zunächst in Fallstudien auf dem Markt befindliche bekannte und erfolgreiche Applikationen darauf untersucht, wie sie dem Problem der Steuerung begegnen. Es wird festgestellt, dass ein Großteil der untersuchten Fallbeispiele ihre Steuerung mit Emulationen von hardwarebasierten Eingabegeräten realisiert. Es wird versucht Sticks und Buttons in Haptik und Optik denen von Gamepads nachzuempfinden und wenig mit den dynamischen Möglichkeiten eines Touchscreens gearbeitet.

Im dritten Kapitel wird somit die Basis geschaffen, sich auf wissenschaftliche Weise mit besagtem Themenkomplex beschäftigen zu können. Es wird geklärt, was eigentlich Usability ist und wie man sie messen kann. Schließlich wird untersucht wie sie auf die

neuen Systeme übertragbar sein könnte. Es wird eine auf Nutzerbefragungen basierte Evaluationsmethode erstellt, die auf dem Verfahren ErgoNorm der Bundesanstalt für Arbeitsschutz basiert. Ein sehr verbreitetes und bekanntes Entwicklungskonzept - das des "iterative Designs" wird vorgestellt und soll schließlich auf einen selbst erstellten Prototypen angewendet werden.

Im vierten und fünften Kapitel wird mittels einer bekannten Engine, der bereits erwähnten Unreal Engine, schließlich die interaktive Grafikdemo "Epic Citadel" so modifiziert, dass sie die Basis für den Test einiger experimenteller Steuerungsideen darstellen kann. Das vierte und fünfte Kapitel behandelt zwei Durchgänge des iterative Design-Prinzips, bei dem Prototypen immer wieder mit Usability-Tests evaluiert und anschließend an aufgefallenen kritischen Punkten verbessert werden. Dieser Prozess wird im Idealfall bis zur Marktreife wiederholt. Diese Arbeit widmet sich beispielhaft zwei dieser Durchläufe inklusive Implementation eines praktischen Beispiels in Form des Prototyps.

Es wird schließlich im sechsten Kapitel festgestellt, dass es schwer ist sich gänzlich von der Idee der Emulation von Gamepads zu lösen aber noch viel Raum für Weiterentwicklung vorhanden ist. Insbesondere im Bereich der dynamischen Nutzung der Fähigkeiten von Touchscreens können Interfaceelemente dann angezeigt werden, wenn sie wirklich gebraucht werden. So kann die Oberfläche kontextsensitiv gestaltet werden. Es wird außerdem erkannt, dass die Methoden der Usability aus der PC-Welt mit leichten Modifikationen definitiv auf die neuen Geräte übertragbar sind, und nicht ignoriert werden sollten.

Abschließend wird in Kapitel 7 ein Ausblick auf mögliche zukünftige Entwicklungen gegeben und die Ergebnisse der Arbeit in ausführlicherer Form zusammengefasst.

2 Stand der Technik

Um zu bewerten was die derzeitigen Probleme bei 3D-Applikationen auf iPad und vergleichbaren Geräten sind, ist es zunächst hilfreich sich eine Übersicht über die derzeit am Markt befindlichen Applikationen und deren jeweilige Besonderheiten in Bezug auf Bedienung und Steuerung zu verschaffen. Es gibt für das entsprechende Segment einige plakative Beispiele, deren grundlegende Bedienphilosophie immer wieder aufgegriffen wird.

2.1 Fallstudien

Beispiele für bekannte Applikationen, die für die Betrachtungen dieser Arbeit relevant sind lassen sich recht schnell mittels Zeitschriften und Websuchen finden. Der einzigartige Vertriebsweg der AppStores, sowohl auf dem Betriebssystem Android von Google als auch von Apple, ermöglicht es den Besitzern der Geräte solche Applikationen sehr schnell und kostengünstig zu erwerben. Im Folgenden werden einige Beispiele für 3D-Spiele und Applikationen gezeigt, die auf das Schema der Problemstellung passen. Die Beispiele basieren auf Suchen in einschlägigen Zeitschriften und Bestenlisten und stellen somit eine repräsentative Auswahl dar.

2.1.1 Epic Citadel (Ausliefungszustand)

Epic Citadel	2.1
Entwickler	Epic Games, Inc.
Erscheinungsjahr	2010
Preis	gratis

Epic Citadel ist im Prinzip eher eine Technikdemo als ein wirkliches Spiel, kann aber als beispielhaft auch für viele 3D Visualisierungen abseits von Games herangezogen werden. In der Applikation bewegt man sich durch eine grafisch aufwändige Burg und kann diese sozusagen besichtigen. Darüber hinaus gibt es an sich in Epic Citadel nicht viel zu tun, aber dennoch ist die Demo wegen der verwendeten Steuerung und der Tatsache interessant, dass sie die Basis für den in dieser Arbeit entwickelten Prototypen darstellt, da sie frei verfügbar und modifizierbar ist.

Abb. 2.1: Epic Citadel - Logo und Startbildschirm - © 2010 Epic Games, Inc.

Abb. 2.2: Epic Citadel - Teil des Tutorials nach dem Start der Applikation - © 2010 Epic Games, Inc.

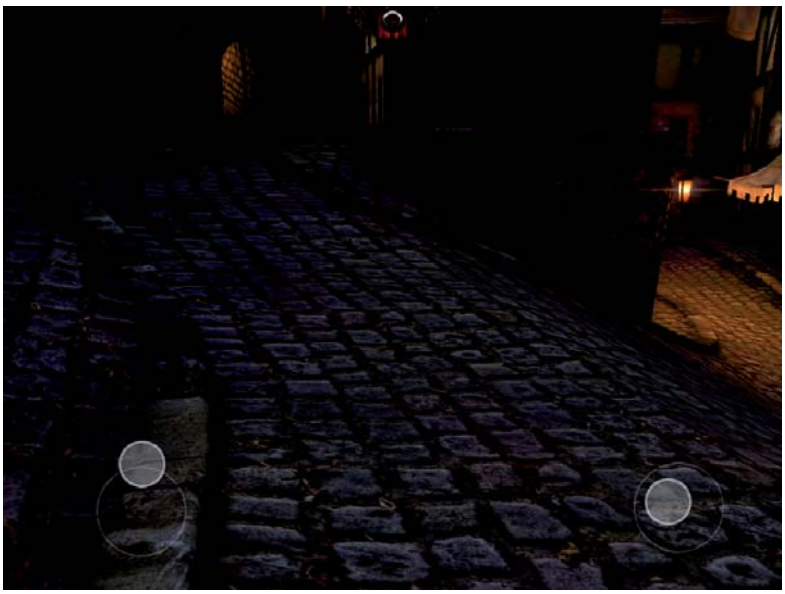

Abb. 2.3: Epic Citadel - Sichtbar sind die beiden Kontrollsticks (der Linke wird gerade benutzt) und der runde Button zum Erreichen des Menüs oben am Bildschirmrand - © 2010 Epic Games, Inc.

Im Auslieferungszustand wird Epic Citadel grundsätzlich mittels zweier emulierter Gamepad-artiger Joysticks gesteuert (siehe Abb. 2.3), die jeweils am linken bzw rechten Bildschirmrand auftauchen. Diese sind nicht in fester Position, sondern bewegen sich automatisch unter den Finger innerhalb bestimmter Zonen des Touchscreens. So können sie genutzt werden, ohne dass man sie exakt treffen müsste. Desweiteren kann man im mittleren Bereich des Bildschirms mittels wischen über den Bildschirm den Blickwinkel verändern. Als dritte Steuerungsvariante und damit Außergewöhnlichkeit kann man auch in die Spielwelt hinein tippen und das Programm bewegt die Spielfigur automatisch zum angetippten Bereich, sofern der Weg nicht allzu komplex ist. Es gibt keinerlei weitere Optionen zum Konfigurieren der Steuerung und auch keine Aktionen, die ausgeführt werden können. Am oberen Bildschirmrand ist schließlich noch ein Button zum Öffnen des Ingame-Menüs untergebracht (siehe Abb. 2.3).

Die Applikation gibt beim Start ein kurzes Tutorial wie die einzelnen Steuerungsarten genutzt werden können, dass den Einstieg erleichtern soll (siehe Abb. 2.2).

Hinweise auf Epic Citadel entnommen aus: [Wie10]

Abb. 2.4: N.O.V.A. 2 - Logo und Startbildschirm - © 2010 Gameloft

Fazit

Als Beispiel für eine Basissteuerung macht Epic Citadel einen guten Eindruck, sofern man mit dem Paradigma der emulierten Gamepad-Sticks vertraut ist - dennoch ist es im Vergleich zu von PCs genutzten Eingabegeräten oder Gamepads wegen des fehlenden physischen Feedbacks grundsätzlich weniger angenehm. Die Steuerung wirkt ebenfalls etwas unpräzise, was aber in einer hauptsächlich auf Begutachten der Landschaft ausgelegten Applikation nicht so sehr ins Gewicht fällt. Möglichkeiten der Interaktion sind nicht vorhanden. Die Idee die Sticks nicht fest an einen Punkt zu fixieren ist gut, da man, wenn man das Gerät "irgendwie" greift bereits meist ohne Probleme zumindest die Sticks benutzen kann.

2.1.2 N.O.V.A. 2 - Near Orbit Vanguard Alliance

N.O.V.A. 2 - Near Orbit Vanguard Alliance	2.4
Entwickler	Gameloft
Erscheinungsjahr	2010
Preis	5,99 €

Nova und Nova 2 sind beide sehr erfolgreich auf Apple's iOS-Geräten vertrieben worden, da sie einerseits zu ihrer jeweiligen Entwicklunsgzeit herausragendes, eher von Konsolen

Abb. 2.5: Das User-Interface von N.O.V.A. 2 während dem Spiel. Links die Touch-Zone zum Kontrollieren der Bewegungen, rechts die Aktionsbuttons, oben Menüs und Inventarsteuerung - © 2010 Gameloft

bekanntes Spieldesign mit ebenso herausragender Grafik kombinierten. Nova 2 wurde jüngst nochmals für das iPad 2 grafisch optimiert, um an den Erfolg des ersten Teils anzuschließen. Bei Nova 2 handelt es sich um einen Vertreter des Genres “First-Person-Shooter”, in dem eine Spielfigur aus der ersten Person-Perspektive gesteuert wird - also so, als würde man durch die Augen der Spielfigur schauen.

In der Standardeinstellung (siehe Abb. 2.5) verwendet Nova 2 als Steuerungsprinzip die Emulation von Gamepad-Sticks. Hierzu allerdings mit einer Reihe an Modifikationen, die es etwas aus der Masse hervorheben. Der Linke, zur Bewegung der Spielfigur gedachte Stick springt bei Berührung des Bildschirms unter den Finger des Nutzers. Solange der Finger auf dem Touchscreen aufgesetzt ist, bewegt sich die Spielfigur in die von der Startposition des Fingers abgeleitete Richtung. Es muss also vom Anwender nicht exakt ein vorgegebenes UI-Element mit dem Finger getroffen werden, sondern die grobe “Zone” auf dem Bildschirm ist relevant. Ähnlich verhält es sich mit dem anderen Stick, dessen visuelle Darstellung nie auftaucht und dadurch quasi der ganze restliche Bildschirm die Eingabezone ist. Innerhalb dieses weitgefassten Bereiches kann mit der rechten Hand der Blickwinkel kontrolliert werden.

Am rechten, unteren Bildschirmrand finden sich die direkten Kontrollen als Buttons, die zum Abfeuern der Waffe, zum Springen und zum Einsatz von Spezialfähigkeiten der

Abb. 2.6: Das Options-Menü von N.O.V.A. 2. Unter Steuerungsart kann zwischen verschiedenen Kontrollvarianten gewechselt werden, Interface anpassen ermöglicht UI Elemente entsprechend eigener Präferenzen zu verschieben. - © 2010 Gameloft

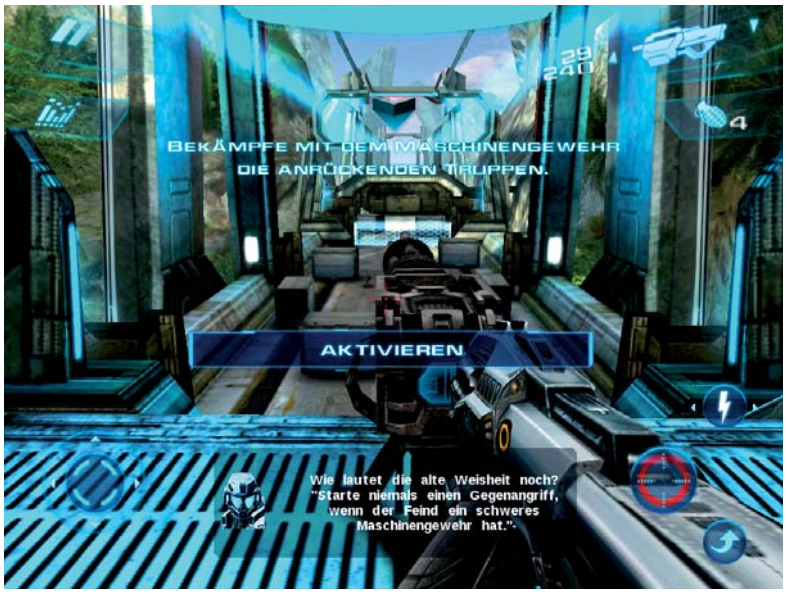

Abb. 2.7: In besonderen Situationen wenn mit Objekten interagiert werden soll blendet N.O.V.A. 2 einen besonders sichtbaren Zusatzbutton ein. - © 2010 Gameloft

Spielfigur genutzt werden können. Letztere können mittels Wischen mit dem Finger über den Button ausgewählt werden. In speziellen Situationen, wenn mit Gegenständen in der Umgebung interagiert werden soll blendet das Spiel einen sehr prominenten Zusatzbutton ein.

Im oberen Bereich des Bildschirms sind alle weiteren spielrelevanten Informationen untergebracht. Sehr präsent steht im Zentrum ein gut sichtbarer Balken, der die Lebensenergie der Spielfigur visualisiert und bei kritischem Stand auch zusätzlich visuelles Feedback gibt - rechts daneben befindet sich ein Bedienelement bei dem durch Wischen zwischen Gegenständen gewechselt werden kann bzw. diese durch Fingerdruck eingesetzt werden können. Auf der linken Seite befindet sich ein Button um das Spiel zu pausieren und ins Pausenmenü zu gelangen sowie ein spezieller Button um die in-Game Musik zu kontrollieren. In der Mitte befindet sich außerdem ein Pfeil, der die Navigation in der Spielwelt erleichtern soll, da er stets in die Richtung des nächsten Zieles zeigt. Beim Zielen auf Gegner rastet das Fadenkreuz im Zentrum in einem großzügigen Umfeld um den Gegner herum stets automatisch auf den Gegner ein, was weniger Präzision beim Zielen nötig macht.

Die Steuerung kann im Optionsmenü (siehe Abb. 2.6) auf eine Linkshändervariante umgestellt werden. Desweiteren können zwei alternative Steuerungsvarianten im Optionsmenü ausgewählt werden. Bei der Einen werden zwei von der Position her fest auf dem Bildschirm fixierte Sticks dargestellt (mit den üblichen Funktionen für Bewegung und Blickwinkel), bei der Anderen Variante der linke (Bewegungs-) Stick in seiner Position fixiert und der gesamten Bildschirm mittels wischen als Blickwinkelkontrolle genutzt. In einem Ingame-Menü lassen sich außerdem Buttons und UI Elemente frei verschieben und sortieren, sowie ein Linkshändermodus aktivieren. Die Empfindlickeit der Sticks kann genauso definiert werden, wie ob die Y-Achse invertiert werden soll, was manchen Nutzern beim Zielen logischer erscheint.

Fazit

Nova bietet ein breites Spektrum an Anpassungsmöglichkeiten, ohne dabei den Nutzer damit zu überfordern. Die Steuerung entspricht dem gängigen Standard von emulierten Gamepad-Sticks und erlaubt so aus anderen Applikationen erlerntes Verhalten weiter nutzen zu können. Die Buttons sind gut erreichbar, aber überwiegend statisch. Gesten werden rudimentär unterstützt, was die Vorteile eines Touchscreens etwas herausstellt.

Hinweis auf Nova entnommen aus: [iW11]

Abb. 2.8: Duke Nukem 3D - Logo und Startbildschirm - Entwickler: Machineworks Northwest LLC © 2010 3D Realms, Inc.

2.1.3 Duke Nukem 3D

Duke Nukem 3D	2.8
Entwickler	Machineworks Northwest LLC / 3D Realms, Inc.
Erscheinungsjahr	2010
Preis	0,79€

Duke Nukem 3D ist im Gaming-Bereich sozusagen ein alter Bekannter. Auf dem PC im Jahre 1996 erschienen, feierte ein Remake kürzlich auf dem iPad seinen erneuten Einstand. Neben Doom war Duke Nukem 3D sicherlich einer der Titel die das First Person Genre auf dem PC als Plattform zu dem gemacht haben, was es heute ist - inklusive diverser Jungendschutz-Kontroversen. Die PC-Version gilt bis heute als indiziert, die iOS Version wurde auch relativ schnell wieder aus dem Store entfernt, steht aber jetzt zwei Jahre später wieder mit einem “ab 12” Jugendschutz-Label in Apples AppStore. Damals wurde das Spiel überwiegend mit Tastatur gesteuert, die heutzutage typische Maus- Tastaturkombi war noch nicht so verbreitet. Das heißt man bewegte sich mit Pfeiltasten und schaute z.B. mit Bild-Auf und Bild-Ab nach oben und unten. Entsprechend schwerfällig war es nach oben und unten zu zielen. Auf dem iPad wirbt 3D Realms schon in der Beschreibung im App Store mit revolutionären Steuerungskonzepten, die um einiges effizienter und beliebter als der Standard sein sollen. Dennoch erinnert vieles an der Steuerung - zumindest die “digitale” Steuerungsvariante - an die Vergangenheit.

Abb. 2.9: UI im Spiel mit der Steuerungsvariante “analog” - Entwickler: Machineworks Northwest LLC © 2010 3D Realms, Inc.

Abb. 2.10: UI im Spiel mit der Steuerungsvariante “digital” - Entwickler: Machineworks Northwest LLC © 2010 3D Realms, Inc.

Abb. 2.11: In-Game Optionsmenü zum Verschieben sämtlicher Steuerungselemente- Entwickler: Machineworks Northwest LLC © 2010 3D Realms, Inc.

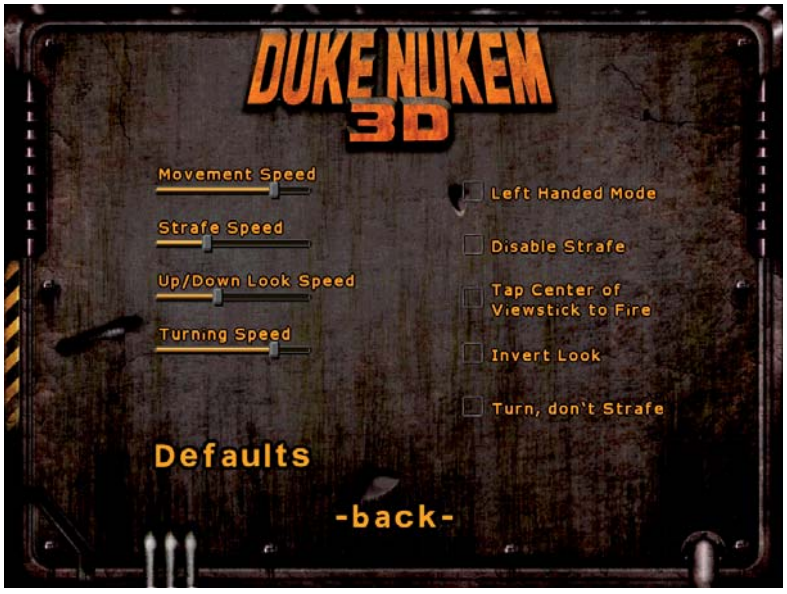

Abb. 2.12: Optionsmenü zum Konfigurieren der analogen Steuerung- Entwickler: Machineworks Northwest LLC © 2010 3D Realms, Inc.

Nach dem Start wird man in den Optionen geradezu von Einstellungsmöglichkeiten zur Steuerung überrannt. Zwei Obergruppen für Bewegung (digital und analog) kombinieren sich mit zwei Varianten zum Zielen und Schießen. Unter der Variante Analog (siehe Abb.2.9) verbirgt sich die von vielen Applikationen bekannte Steuerung mit zwei emulierten Gamepad-“Sticks” am unteren Bildschirmrand. Darum angeordnet sind einige Aktionsbuttons und ein nur eingangs im Ladebildschirm erwähnter Bereich, der keine grafische Repräsentation hat und den man drücken soll um Waffen zu wechseln. Die Sticks sind statisch und ändern ihre Position nicht selbständig im Spiel, können aber wie alle Elemente der Steuerung im Optionsmenü an andere Stellen bewegt werden (siehe Abb.2.11). In jedem Fall müssen die Finger immer sehr exakt auf den Sticks liegen, bewegt man sie im Eifer des Gefechts auch nur ein wenig aus der “Stick-Zone” heraus, bleibt die Spielfigur sofort stehen.

Die zweite Bewegungsvariante ist die “digitale”(siehe Abb. 2.10), die eine Masse an zusätzlichen Buttons auf den Bildschirm bringt und mehr an die klassische PC-Steuerung erinnert. Die Buttons sind alle recht klein gehalten und reagieren ebenfalls nur so lange wie der Finger exakt auf ihnen ruht. Jeder einzelne der Buttons ist frei verschiebbar durch bereits erwähntes Optionsmenü.

Eine wahre Masse an Unteroptionen existiert für beide Steuerungsmethoden im Optionsmenü - Angefangen bei Sensitivitäten für die einzelnen Funktionen (Movement Speed, Strafe Speed, ...) (siehe Abb.2.12) bis hin zu einem Umschalter für Linkshänder und viele weitere Optionen, von denen je nach gewählter Steuerungsart unterschiedliche zur Verfügung stehen. Beispielsweise gibt es für die digitale Steuerungsvariante eine Option für größere Buttons, die als Standardeinstellung aktiviert ist und im deaktivierten Zustand die digitalen Steuerungsbuttons auf die Größe einer Kinder-Fingerkuppe verkleinert.

Beim Zielen hat sich 3D Realms etwas besonderes einfallen lassen, da es eine “Tap-to-shoot” Funktion gibt, die dem Spieler erlaubt Gegner anzuvisieren, sofern sie irgendwo in seinem Sichtfeld sind. Der Spieler zielt, in dem er sie einfach mit dem Finger berührt. Auch diese Funktion kann umkonfiguriert werden, um mit einem Zielkreuz und den Bewegungskontrollen wieder genau und manuell zielen zu müssen.

Aktionen werden mit einem einzelnen Button ausgelöst, der in den meisten Fällen ohne merkliches Feedback gar nichts tut, oder wenn man vor etwas Unbenutzbaren steht und darauf zielt ein plakatives “keine Funktion”-Geräusch erzeugt. Man muss sehr genau auf zu manipulierende Gegenstände zeigen um sie benutzen zu können, die “Tap-to-shoot” Funktion ist hier nicht implementiert.

Das Spiel verzichtet auf Hilfen wie Waypoints oder Minimaps, die Lebensenergie der Spielfigur wird durch einen kleinen Balken in der linken oberen Ecke dargestellt und durch Geräusche des Charakters abhängig von seinem Status unterstrichen. Die restliche Munitionsmenge ist nur im schwer zu erreichenden Waffenmenü einsehbar, dass sich - wie bereits erwähnt - hinter einer unsichtbaren Schaltfläche über der Waffe befindet.

Fazit

Die Steuerung von Duke Nukem ist extrem sperrig und vermittelt kein Gefühl von Präzision. Es ist annähernd unmöglich mit den statisch angebrachten Sticks genau auf Spielelemente zu zielen oder bestimmte Orte anzusteuern. Dabei wird der Anwender mit Optionen zur Steuerung überfordert, bei denen ihm zunächst völlig unklar ist was sie bewirken.

Hinweis auf Duke Nukem entnommen aus: Zufallsfund iOS Appstore Suche

2.1.4 Rage

Rage	2.13
Entwickler	id Software
Erscheinungsjahr	2010
Preis	1,59€

Rage nimmt in dieser Liste eine Sonderstellung ein, da es sich dabei um einen sogenannten "Rail-Shooter" handelt. Das heißt die Spielfigur wird nicht direkt durch den Nutzer kontrolliert, sondern der Weg ist vorgegeben, der Nutzer kann lediglich Blickwinkel und alle weiteren Funktionen der Spielfigur kontrollieren. Dabei handelt es sich um ein System, dass bereits vor vielen Jahren in Arcade-Spielen, also Spielhallen-Automaten sehr populär war. Das Konzept verzichtet also bewusst auf einen Freiheitsgrad des Spielers um sich so mehr auf die anderen Bereiche konzentrieren zu können.

Mit dem Finger kann über den Bildschirm gestrichen werden um so Blickwinkel und Zielrichtung zu ändern, ähnlich wie mit einem emulierten Stick. Alle weiteren Funktionen werden über große Buttons an der rechten Seite des Bildschirms ausgeführt. So wird in diesem Spiel abweichend die linke Hand zum korrigieren des Blickwinkels genutzt, und die rechte Hand ausschließlich für Aktionen und Interaktionen (Interface von Rage siehe Abb.2.14). Die einzige Art von Bewegungskontrolle bleibt dabei eine Ausweichfunktion, die dem Spieler temporär erlaubt einen Seitwärtsschritt mit der Spielfigur auszuüben um Treffern zu entgehen. Die Steuerung kann im Optionsmenü (siehe Abb. 2.16) weiter individualisiert werden. So lassen sich Achsen umkehren und die Empfindlichkeit der Blickpunktänderung einstellen. Auch ein Linkshänder-Modus, der die Buttons auf die andere Seite verschiebt ist vorhanden. Eine Besonderheit stellt die Option dar die Neigungssensorik als Blickrichtungssteuerung zu nutzen. Dabei wird zusätzlich zum Zielen mit dem Finger auf dem Touchscreen die Blickrichtung abhängig von der Kipprichtung des Geräts geändert, was in der Reihe der hier gezeigten Beispielapplikationen einzigartig ist. In diesem Modus werden die Aktionsbuttons auch auf die linke und rechte Bildschirmseite verteilt, da ja theoretisch die zweite Hand "frei" ist (siehe Abb. 2.15).

Abb. 2.13: Rage - Logo und Startbildschirm - © 2010 id Software / Bethesda Softworks

Abb. 2.14: UI von Rage mit aktivierter Touch-Steuerung. Alle Aktionsbuttons befinden sich auf der rechten Seite, die linke Hand wird zum Zielen verwendet © 2010 id Software / Bethesda Softworks

Das Spiel bietet außerdem einen Museumsmodus, der einem erlaubt die Levels ohne Gegner zu erkunden. Hierbei kann die Bewegung durch die Levels allerdings auch nicht selbst kontrolliert werden, aber man selber bestimmt wann die Spielfigur auf dem vordefinierten Pfad weiter läuft.

Fazit

Rage bietet eine einfache und eingängige Standard-Steuerung ohne Verwendung von Sticks oder komplizierten Interface-Elementen, verzichtet aber gänzlich darauf den Spieler selbst die Bewegung kontrollieren zu lassen. Die alternative Steuerung über die Bewegungssensoren ist innovativ und eingängig aber relativ unpräzise.

Hinweis auf Rage entnommen aus [Woo11]

Abb. 2.15: UI von Rage mit aktivierter Neigungssteuerung. Die Aktionsbuttons sind rechts und links verteilt. Oben links befindet sich ein Button zum Wechseln ins Pausenmenü © 2010 id Software / Bethesda Softworks

2.1.5 Shadow Guardian

Shadow Guardian	2.17
Entwickler	Gameloft S.A.
Erscheinungsjahr	2010
Preis	5,49€

Shadow Guardian ist ein Adventure-Spiel mit Rätseln und Jump-and-Run-Einlagen. Desweiteren wird es aus der dritten Person gespielt, das heißt man sieht die Spielfigur vor sich und schaut nicht durch ihre Augen. Dennoch kann auch dieses Beispiel herangezogen werden, da auch für diesen Spieltyp ähnlich Anforderungen an die Steuerung gestellt werden, wie an diese die in der ersten Person gespielt werden. Auch Shadow Guardian nutzt das bereits erläuterte Gamepad-Stick-System (mit fest positioniertem Stick), ergänzt dies aber durch eine Reihe interessanter Details (siehe Abb. 2.18). So werden vom Spiel oft Buttons nur eingeblendet wenn sie für das aktuelle Spielgeschehen Relevanz besitzen, oder hebt einzelne Funktionen kontextabhängig mit grafischen Veränderungen hervor, wenn sie gerade eingesetzt werden können (z.B. in Deckung gehen hinter einem Steinhaufen, siehe Abb. 2.19). Wiederum kann die Bewegung mit einem der Position des Daumens folgenden Stick an der linken Bildschirmseite und der Blickwinkel auf der ganzen restlichen Bildschirmfläche mittels Wischen kontrolliert werden. Im oberen Bereich

Abb. 2.16: Optionsmenü von Rage. Oben kann zwischen Neigungs- und Touch-Steuerung umgestellt werden, darunter findet sich einige Detailkonfigurationen wie Empfindlichkeit und Linkshändermodus. © 2010 id Software / Bethesda Softworks

Abb. 2.17: Shadow Guardian - Logo und Startbildschirm - © 2010 Gameloft

Abb. 2.18: UI von Shadow Guardian nach dem Tutorial. Rechts befinden sich die Aktionsbuttons, links ein Stick zum Bewegen der Spielfigur. Der Blickwinkel kann auf dem gesamten Bildschirm mittels Wischen kontrolliert werden. Oben links befindet sich der Button für das Pausenmenü - © 2010 Gameloft

Abb. 2.19: Shadow Guardian passt seine Buttons kontextsensitiv an die Situation an und gibt auch Empfehlungen für den Spieler. Hier wird der Button für die Funktion Ducken hervor gehoben. - © 2010 Gameloft

befindet sich ein Auswahlfeld für den gerade aktiven Gegenstand und ein Button für das Pausenmenü. Interessant ist auch, dass Shadow Guardian gänzlich auf eine Lebenskraft-Anzeige verzichtet und versucht vieles auf anderen Kanälen zu kommunizieren. So wird beispielsweise der Bildschirm rot und unscharf wenn es dem Protagonisten schlechter geht.

Das Spiel hilft bei der Navigation durch die Levels indem es leuchtende Pfeile auf dem Boden platziert, die immer genau so weit weg sind, dass sie sich noch in Sichtweite befinden (siehe Abb. 2.20). Auch nennenswert ist eine ausgedehnte Tutorialphase, in der erst nach und nach die Funktionen der Oberfläche in die Handlung eingebettet für den Spieler aktiviert werden, um ihn so langsam an die Steuerung heran zu führen. Die anderen Beispiel-Applikationen dieses Kapitels beschränken sich hierbei, so nicht speziell anders erwähnt, auf einfache Textmeldungen um dem Spieler die Steuerung zu erklären oder verzichten gänzlich darauf irgendwelche Hilfestellungen zu bieten.

Auch Shdow Guardian bietet in einem Optionsmenü die Möglichkeit die Position der Bedienungselemente an die Präferenzen des Nutzers anzupassen.

Abb. 2.20: Shadow Guardian hat ein Ausführliches Tutorial in dem der Spieler langsam an die Steuerung heran geführt wird. Pfeile in der Landschaft markieren während des Spiels Wegpunkte die erreicht werden sollen. - © 2010 Gameloft

Fazit

Shadow Guardian bietet mit kontextsensitiven Kontrollen und einer recht präzisen Steuerung ein solides, teilweise innovatives Gesamtpaket. Die Steuerung lässt sich bei Bedarf kleinteilig den eigenen Bedürfnissen anpassen und versucht so viele verschiedene Neigungen von Nutzern zufriedenzustellen.

Hinweis auf Shadow Guardian entnommen aus: [Ruh11]

2.2 Zusammenfassung

Zusammenfassend kann festgestellt werden, dass sich offensichtlich gewisse Steuerungs-Trends am Markt durchgesetzt haben. Die virtuellen Sticks scheinen allgegenwärtige Steuerungsinstrumente zu sein, die jeweils mit eigenen kleinen Modifikationen fast immer den Haupteingabevektor im Bereich 3D Applikation auf Tablets zu sein scheinen. Ebenfalls sehr verbreitet sind großflächige “Wischzonen”, die bevorzugt zum justieren des Blickwinkels genutzt werden. Generell ist anzumerken, dass die meisten aktuellen Applikationen Möglichkeiten anbieten Buttons und Eingabeelemente frei von der Position her anzupassen, manche Applikationen bieten auch die Möglickeit zwischen mehreren

Eingabeprofilen zu wechseln. Erstaunlich selten hingegen werden die Neigungssensoren der Geräte eingebracht, die in vielen anderen Genres wie beispielsweise Rennspielen die Steuerungskonzepte absolut zu Recht dominieren. In den nächsten Abschnitten wird kurz auf die gezeigten weit verbreiteten und üblichen Steuerungsvarianten eingegangen, um die Basis und die Begrifflichkeiten für die folgenden Untersuchungen zu bilden.

2.2.1 Emulation von Gamepads

Da offenbar viele iPad-Entwickler und damit Games besonders auf den typischen Konsolenspieler abzielen (Das System verhält sich, was Geschlossenheit und Einfachheit angeht, ähnlich), lag es offenbar nahe gewisse Eigenheiten der Bedienung von Konsolen zu übernehmen. Prinzipiell wird ein iPad oder auch ein Smartphone auch vergleichbar wie ein Gamepad gehalten, also die Daumen auf dem Display und damit der Oberseite liegend. Folgerichtig werden in vielen Games Eingabegeräte auf dem Touchscreen emuliert, die einem Gamepad (siehe Abb. 2.21) sehr ähnlich sind. Die hauptsächlichen Eingabemöglichkeiten an einem Gamepad sind Kontroll-Sticks, Steuerkreuze und Buttons, sowie im Spezialfall der Buttons auch mit dem Zeigefinger auslösbare Trigger [1]. Letztere fallen automatisch für die derzeit auf dem Markt befindlichen Tablets weg, da die Hardware solche Knöpfe nicht vorsieht. Bleiben Buttons, Steuerkreuze und Sticks. Gerade die analogen Sticks sind natürlich recht interessant, da dadurch sehr feinfühlige Eingaben für Bewegungen auf Gamepads möglich werden. Diese Analogsticks werden in vielen Applikationen auf Tablets wie in den letzten Abschnitten dargelegt auf dem Display emuliert (siehe Abb. 2.22) und ihr Handling gleicht, vom physischen Feedback am Finger abgesehen, meist dem der Gamepad-Sticks. Üblich sind am Gamepad ebenfalls zwei Sticks, einer für jeden Daumen, wobei der Linke meist die Spielfigur bewegt und der Rechte den Blickwinkel kontrolliert. Der rechte Daumen wechselt meist hin und wieder zwischen Sticks und Buttons. Auch dieses Verhalten wird auf dem Tablet nachgestellt, da sich dort auch die Buttons für einzelne Aktionen meist am rechten Bildschirmrand befinden (oder eben in einer möglicherweise vorhandenen Linkshänder-Option auf der linken Seite). Das fehlende physische Feedback beim Berühren der virtuellen Sticks stellt aber scheinbar ein Problem dar, wie man selbst beim Testen der Applikationen immer wieder bemerkt.

2.2.2 Accelerometer-Basiert

Viele der aktuellen Smartphone und Tablet-Modelle haben neben einem Touchscreen noch sogenannte Accelerometer oder Gyroskope integriert, die Bewegung und Orientierung des Geräts im Raum messen können[2]. Diese Sensorik ist gerade deshalb so interessant

[1] [FPM09]
[2] [Sta10]

Abb. 2.21: XBox 360 Gamepad, Funk-Variante - © 2005 Microsoft

Abb. 2.22: Typische Darstellung eines emulierten Gamepad-Sticks, der gerade nach oben gedrückt wird - aus Epic Citadel - © 2010 Epic Games, Inc.

für Spiele, da sie immer zur Verfügung steht und keine spezielle Aufmerksamkeit des Nutzers braucht um genutzt zu werden. So kann der Nutzer seine Hände auf dem Touchscreen behalten, aber trotzdem durch Neuorientierung des Geräts Variablen in der Spielwelt beeinflussen. Übliche Szenarien sind beispielsweise Rennspiele, bei denen das ganze Tablet so zum Lenkrad mutiert und beim in die Kurven fahren geneigt wird. Auch "Gestiken" wie schütteln oder schubsen sind möglich und werden teilweise auch verwendet. In den Fallbeispielen aus diesem Kapitel wird die Bewegungssensorik - außer beim Beispielprogramm Rage - als rein optionales Element nicht genutzt.

2.2.3 Touchscreen

Der Touchscreen, das zentrale Element sämtlicher Tablets, ermöglicht eine dritte häufig genutzt Variante mit der Spielwelt zu interagiern. Mit Fingern und Gesten können direkt auf dem Bildschirm Elemente der Spielwelt manipuliert werden. Dabei sind zum Einen bekannte Gesten wie "Pinch-to-Zoom" oder einfaches Wischen in eine bestimmte Richtung zu nennen, aber auch komplexere Gesten, die auf dem zu manipulierenden Objekt basieren sind prinzipiell denkbar. Somit kann der Anwender über den Touchscreen quasi direkt mit seinen Fingern in die Welt hineingreifen. Besagte Gesten wurden in den genannten Fallbeispielen auch eher selten eingesetzt, obwohl sie zum Alltag der Nutzung der Tablets gehören. Die häufigste beobachtete Geste war das Wischen über Kontrollelemente oder den Bildschirm um Eingaben auszuführen. Sehr häufig werden Buttons eingesetzt, die in ihrer meist runden Form auch an Gamepads erinnern. Die Buttons sind in seltenen Fällen kontextsensitiv, also bieten je nach aktueller Situation passende Funktionen an und sind sehr häufig in ihrer Position vom Nutzer frei konfigurierbar. Direkte Manipulation von Gegenständen oder Elementen in der Spielwelt kam ebenfalls erstaunlich selten vor, man muss sich fragen warum bei einem Touchscreen die zusätzliche Abstraktion eines "Benutzen"-Buttons einem direkten Anfassen des Objekts (z.B. einen Schalter) in der virtuellen Welt vorgezogen wird.

2.2.4 Schlussfolgerung

Zusammenfassend lässt sich ableiten, dass gerade im Bereich der direkten Manipulation von Objekten in der virtuellen Welt der Applikation, als auch der Nutzung sämtlicher im Gerät verfügbarer Sensorik noch einiger Raum für Innovation zu finden ist. Auch erscheint es fraglich, warum die Eingabeelemente fast immer den gleichen statischen Regeln unterworfen werden wie ihre physischen Äquivalente bei einem Gamepad. Kontextsensitive Bedienelemente sind durchaus mit der aktuellen Technologie vorstellbar und auch umsetzbar. Bis auf ein paar wenige, sehr zaghafte Ausnahmen (Kontextsensitive Buttons bei Shadow Guardian) folgen die virtuellen Eingabeelemente alle den gleichen

Regeln wie die statischen physikalischen Elemente eines Gamepads, obwohl wesentlich mehr Dynamik in Darstellung und Funktion denkbar wäre.

Bevor man sich allerdings Überlegungen für neue Elemente dieser Art widmen kann, müssen zunächst einige Regeln und Methoden definiert werden, die zum einen dabei unterstützen derartige Gedanken in die richtige Richtung zu lenken und zum anderen ermöglichen diese neuen Steuerungsmethoden auch zu bewerten. Das Stichwort hierzu lautet Usability.

3 Usability

Im vorigen Kapitel ist eindeutig ersichtlich, dass es sehr unterschiedliche und auch offenbar unterschiedlich gut funktionierende Herangehensweisen gibt, dem Problem der Steuerung von 3D-Applikationen auf Tablets gegenüber zu treten. Es stellt sich die Frage, ob es möglich ist, diese Herangehensweisen und ihre Ergebnisse auf wissenschaftliche Weise erfassen und bewerten zu können. Die Antwort auf diese Frage ist das Fachgebiet Usability.

In diesem Kapitel muss zunächst definiert werden was Usability bedeutet. Befragt man den Übersetzer Leo.org zum Begriff Usability3.1, so erhält man als Antwort Worte wie Bedienbarkeit, Benutzbarkeit, Benutzerfreundlichkeit und Brauchbarkeit. Das ist schon sehr nah an dem, was der Begriff bedeutet. Die International Organization for Standardization - besser bekannt als ISO definiert den Begriff folgendermaßen:

> "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use." ISO 9241-11 (1998)

Mit anderen Worten - es geht darum ein Produkt im Hinblick auf die maximale Effizienz und Effektivität sowie auch maximale Befriedigung des Nutzers hin zu optimieren, und zwar auf den typischen Nutzungskontext hin abgezielt. Im Allgemeinen wird der Begriff Usability üblicherweise folgendermaßen in Unterkategorien aufgefächert [1].

1. Learnability (Erlernbarkeit) = Wie einfach ist es für den Nutzer übliche Aufgaben gleich beim ersten Kontakt mit dem Produkt zu bewältigen?
2. Efficiency (Effizienz) = Wie schnell können Nutzer Aufgaben nach der Lernphase erledigen?
3. Memorability (Einprägsamkeit) = Wie schnell können Nutzer, die eine gewisse Zeit lang nicht mit dem Produkt gearbeitet haben, wieder vollständig damit umgehen?
4. Errors (Fehler) = Wie hoch ist die Häufigkeit von Fehleingaben durch den Nutzer, wie dramatisch sind diese Fehler und können sie die Fehler eigenständig korrigieren?
5. Satisfaction (Befriedigung) = Wie befriedigend ist es für den Nutzer das Produkt zu nutzen?

[1] (nach Jakob Nielsen[Nie94])

Substantive (8 of 8)		
usability *also:* useability	f	die Bedienbarkeit
usability		die Benutzbarkeit
usability *also:* useability		die Benutzerfreundlichkeit
usability *also:* useability		die Brauchbarkeit
usability		die Gebrauchstauglichkeit
usability *also:* useability		die Nutzbarkeit
usability *also:* useability		die Verwendbarkeit
usability		die Verwertbarkeit

Abb. 3.1: Screenshot von dict.leo.org zur Anfrage “Usability”

Dass der Begriff Usability für jede Art von Produkt anwendbar ist, ist mit diesen Definitionen automatisch gegeben. Im Speziellen gilt es im Themenbereich dieser Arbeit den Fokus allerdings auf Benutzerinterfaces von Software zu legen. Der Begriff Usability ist seit Aufkommen komplexer grafischer Nutzeroberflächen immer mehr in den Vordergrund geraten und vom eigentlichen Funktionsumfang einer Software entkoppelt. Usability ist kein Teil des Funktionsumfangs, sondern das Bindeglied zwischen Software und Nutzer. Demnach bildet sie das Kriterium, inwiefern der Nutzer den Funktionsumfang einer Software an sich effektiv nutzen kann.

Usability kann mit verschiedenen Methoden, die natürlich wieder abhängig vom jeweiligen Produkt sind, gemessen und verglichen werden. In sogenannten Usability Tests wird mit diesen Methoden versucht, Problemstellen zu identifizieren und so konstruktives Feedback an den Autor der Software zurückzugeben.

Natürlich definieren in Bezug auf Software verschiedenste Experten unterschiedliche Richtlinien und Methoden zur Evaluation von Software, um die diffuse Maßeinheit Usability möglichst zu optimieren.

Zum Thema Software-Ergonomie (= Usability, speziell auf Software bezogen) kann nochmals die ISO 9241 zitiert werden, die in ihren “Grundsätzen der Dialoggestaltung” Qualitätskriterien für Software definiert.

1. Aufgabenangemessenheit = angepasste, geeignete Funktionalität, Minimierung für die Aufgabe unnötiger Interaktion
2. Selbstbeschreibungsfähigkeit = Vorhandensein von Hilfen und Rückmeldungen
3. Lernförderlichkeit = leichte Erlernbarkeit des Umgangs, Verwendung von Metaphern
4. Steuerbarkeit = Steuerung des Dialogs durch den Benutzer
5. Erwartungskonformität = An die erwartete Nutzergruppe und ihre Erwartungen an ein solches System angepasster Dialog
6. Individualisierbarkeit = Der Nutzer kann die Software an seinen Arbeitsstil / Arbeitsumgebung anpassen
7. Fehlertoleranz = Auch bei Fehlern ist die Funktion sicher gestellt.

Es folgen einige grundlegende Prinzipien und Fragestellungen für den Entwickler, bevor genauer auf die eigentliche Problemstellung der Usability-Bewertung eingegangen wird.

3.1 Usability Grundregeln

In Literatur und Lehre haben über die Jahre eine große Anzahl Autoren und Wissenschaftler viele unterschiedliche Grundregeln entwickelt, die es bei der Softwareentwicklung zu beachten gilt. Je nach Autor wird hierbei der Fokus mal eher auf die eine oder die andere der oben genannten Sparten gelegt. Im Endeffekt kann aber festgestellt werden, dass diese Grundregeln nur der Beginn einer Usability-Betrachtung sein können, da die Benutzbarkeit eines Produkts immer vom jeweiligen Nutzungskontext abhängt. So muss für jedes Projekt zwangsweise die ganze Betrachtung entsprechend dem Nutzungskontext angepasst werden. Insbesondere sind die Grundregeln im Optimierungsprozess tatsächlich eher als Basisrichtlinien zu verstehen, die von vornherein eine gewisse Grundbedienbarkeit sicherstellen sollen. Ein entscheidender Teil von Usabilty-Optimierung sind eindeutig die Evaluation von Prototypen *während* des Entwicklungsprozesses und *danach*, und nicht das Aufstellen von Richtlinien *vor* dem Entwicklungsprozess - gerade auch, da sich Software und sogar deren Nutzungskontext oft im Laufe der Entwicklung oder Lebenszeit durch Anforderungen des Kunden oder Änderung der Einsatzbedingungen nochmals verändern kann.

Im Ergonomics in Design Journal veröffentlichte Arnold Lund 1997 eine Reihe von frei formulierten Grundregeln, die auch heute noch viel Relevanz haben und als verständlicher Einstieg in einen verhältnismäßig allgemein verwendbaren Regelsatz verstanden werden können[2].

1. Know thy user, and YOU are not thy user.
2. Things that look the same should act the same.
3. Everyone makes mistakes, so every mistake should be fixable.
4. The information for the decision needs to be there when the decision is needed.
5. Error messages should actually mean something to the user, and tell the user how to fix the problem.
6. Every action should have a reaction.
7. Don't overload the user's buffers.
8. Consistency, consistency, consistency.
9. Minimize the need for a mighty memory.

[2] [Lun97]

10. Keep it simple.
11. The more you do something, the easier it should be to do.
12. The user should always know what is happening.
13. The user should control the system. The system shouldn't control the user. The user is the boss, and the system should show it.
14. The idea is to empower the user, not speed up the system.
15. Eliminate unnecessary decisions, and illuminate the rest.
16. If I made an error, let me know about it before I get into REAL trouble.
17. The best journey is the one with the fewest steps.
18. Shorten the distance between the user and their goal.
19. The user should be able to do what the user wants to do.
20. Things that look different should act different.
21. You should always know how to find out what to do next.
22. Don't let people accidentally shoot themselves.
23. Even experts are novices at some point.
24. Provide help.
25. Design for regular people and the real world.
26. Keep it neat.
27. Keep it organized.
28. Provide a way to bail out and start over.
29. The fault is not in thyself, but in thy system.
30. If it is not needed, it's not needed.
31. Color is information.
32. Everything in its place, and a place for everything.
33. The user should be in a good mood when done.
34. If I made an error, at least let me finish my thought before I have to fix it.
35. Cute is not a good adjective for systems.
36. Let people shape the system to themselves, and paint it with their own personality.
37. To know the system is to love it.

In den folgenden Kapiteln werden einige dieser Regeln wieder auftauchen, die auch für das speziell in dieser Arbeit diskutierte Problem gesteigerte Relevanz haben - manche auch nicht. In jedem Fall handelt es sich um einen Satz leichtverständlicher Richtlinien, die jeder Softwareentwickler so mit seiner Software in Zusammenhang gebracht haben sollte.

3.2 Usability von Touchscreen-basierter Software

Im letzten Abschnitt wurden generelle Richtlinien und Regeln für Usability in Software besprochen. Diese sind zwar allgemein gültig, aber nicht speziell auf die Anwendbarkeit auf Touchscreen-Geräte hin optimiert. Da hier aufgrund der Art und Bedienung dieser Geräte andere Maßstäbe angelegt werden müssen, gelten auch zusätzliche Regeln oder sinnvolle Hinweise für die Entwicklung von Software auf diesen Geräten.

Die Firma Apple selbst definiert in zwei Dokumenten Richtlinien und Hinweise für Entwickler, die iOS Human Interface Guidelines(hier gekürzt und zusammen gefasst) [3]:

Es werden sechs Basiskriterien für Anwendungen auf Touch Devices präsentiert:

>>>Fußnote Link kollabiert

Aesthetic Integrity

An die Funktion angepasste Ästhetik, das heißt Produktivanwendungen sollen sich mit übermäßigen grafischen Verzierungen zurück halten. Im Gegensatz dazu sollen Applikationen wie Games Medien und Grafik gezielt einsetzen um den “Immersionsfaktor” des Nutzers zu erhöhen und ihn ermutigen die Welt zu erforschen.

Consistency

Das Halten an bekannte Standards was Buttons und Kontrollen betrifft. Die Applikation soll ihre Funktionen stets so präsentieren, dass sie intuitiv als solche erkannt werden. D.h. konsistente Verwendung von Symbolen, Metaphern und Wiederverwendung von dem Nutzer bereits aus anderen Applikationen bekannten Elementen. Buttons und Kontrollen sollen Wiederkennungswert haben, und auch an unterschiedlichen Orten im Programm die gleiche Funktion tragen, wenn sie gleich aussehen.

[3] http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/UEBestPractices/UEBestPractices
CH20-SW1

Direct Manipulation

Die Anwendung soll die Stärken des Touchscreens nutzen direkte Manipulation von Bildschirmelementen mit den Fingern wie in der Realwelt zuzulassen. Bekannte Gesten sollen erwartungsgemäß unterstützt werden (Zoom-Geste, wischen um zu scrollen etc.) und wenn möglich Buttons vorgezogen werden. Aktionen sollen sofortige und sichtbare Reaktionen erzeugen.

Feedback

Die Anwendung soll bei allen Aktionen des Nutzers Feedback geben und auch Warte- und Ladezeiten eindeutig visualisieren und abbrechbar machen. Hierzu wird auch ermutigt zurückhaltende Animationen zu nutzen. Sound kann auch als Feedback benutzt werden, sollte aber nicht alleinige Feedbackvariante einer Funktion sein, da viele Nutzer ihr Gerät stumm schalten könnten.

Metaphors

Die Anwendung sollte Gegenstände aus der realen Welt nachbilden, damit der Nutzer bereits erlernte Vorgänge aus der realen Welt auch virtuell nutzen kann (also z.B. Schalter umlegen mit dem Finger, Seiten umblättern mit Wischen über den Bildschirm, herumschieben und schubsen von Elementen auf dem Bildschirm etc.) Die Metaphern sollten aber Grenzen haben, wenn es die Nutzung erschweren würde. Beispielsweise ist ein Ordner eine gute Metapher für ein Objekt, in dem andere Objekte abgelegt werden können, aber es wäre kontraproduktiv Dinge darin wie in einem echten Akteordner sortieren zu müssen indem man sie umherschiebt um Übersicht zu erhalten und ansonsten einen unsortierten Haufen zu sehen bekommt, auch wenn es theoretisch in der realen Welt so wäre.

User Control

Die Nutzer sollen die Applikation kontrollieren und nicht umgekehrt, das heißt Aktionen sollen vom Nutzer und nicht dem Programm initiiert werden. Es können Vorschläge gemacht und vor Problemen gewarnt werden, aber es muss stets klar sein, dass der Nutzer in der Kontrollposition ist. Desweiteren soll der Nutzer länger andauernde Aktionen immer abbrechen können.

Darüber hinaus gibt es noch eine große Menge speziellerer Regeln in den Guidelines, die nicht alle zum Thema Gaming / 3D passen oder Redundant zu vorherigen genannten

sind und deswegen hier etwas gekürzt aufgezählt und erklärt werden[4]:

- Focus on the Primary Task = Das Programm soll möglichst nur das auf dem Bildschirm zeigen was gerade gebraucht wird, um besser mit dem begrenzten Display-Platz umzugehen und Verwirrung des Nutzers zu vermeiden.
- Elevate the Content People Care About = Das Programm soll das, weswegen die Leute es nutzen in den Vordergrund stellen. z.B. für Games die Spielwelt ansprechend und ausführlich inszenieren in z.B. Grafik und Story.
- Think Top Down = Abhängig davon, wie Leute ein Tablet halten, haben sie den Fokus ihrer Aufmerksamkeit auf der oberen Hälfte des Bildschirms. Demnach sollten häufig genutzte oder besonders wichtige Informationen dort zu finden sein.
- Give People a Logical Path to Follow = Die Interaktion mit dem Programm soll immer den gleichen Richtlinien folgen, also z.B. soll der Weg in ein bestimmtes Menü immer der Gleiche sein, bzw. immer gleich kenntlich gemacht sein. (oder im Gaming-Kontext eine Aktion mit dem gleichen Button versehen etc.)
- Use User-Centric Terminology = Technische und hochgestochene Formulierungen vermeiden und somit die Sprache des Nutzers sprechen.
- Minimize the Effort Required for User Input = Die Schwierigkeit eine Aktion durchzuführen muss im Verhältnis zum Effekt stehen.
- Make Targets Fingertip-Size = Buttons und Kontrollen müssen mindestens so groß wie die Fingerkuppe des Nutzers sein

Es ist auffällig, dass scheinbar keiner der Mitbewerber außer Apple einfach zu findende Usabilitykriterien für ihre mobilen Touch-Devices online zur Verfügung stellt oder diese offenbar so tief in ihren Dokumentationen verbirgt, dass sie mittels Websuchen nicht mehr zu finden sind. Weder Google mit Android noch Microsoft mit Windows Phone versuchen scheinbar einen derart tiefen Einblick in Usability-Überlegungen zu bieten und ihre Dokumentationen und Interface-Guidelines drehen sich eher darum Standards für Anwendungen auf diesen Plattformen zu definieren, damit eine gewisse Einheitlichkeit und Konsistenz unter den Applikationen entsteht. Die Guidelines sind dabei hoch technisch angelegt und beschreiben eher den Verwendunskontext bestimmter Elemente und Kontrollen und weniger allgemeine Paradigmen, die die Usability einer Anwendung erhöhen - ein Umstand der durchaus als bedenklich betrachtet werden kann.

In dem Buch “Brave NUI World” [5] beschreiben die Autoren Wigdor und Wixon - beides Microsoft Ingeneure, die mitverantwortlich für die Entwicklung von Microsofts Multi-Touch Tisch “Surface” waren einige Ideen und Konzepte zu sogegannenten “Natural User

[4] http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Principles/Principles.html#//app
CH5-SW1
[5] [Wix11]

Interfaces". In sehr umfangreichen Kapiteln wird die Idee eines für den User natürlich und instinktiv begreifbaren Bedienungskonzepts ausgearbeitet und mit praktischen Hinweisen für Entwickler kombiniert. "Natural" ist hier eher als "to be a natural" zu verstehen und beschreibt Nutzerinterfaces, bei denen der Nutzer automatisch Experte ist ohne große Lernphasen durchmachen zu müssen. Auszugsweise und stark zusammengefasst sind diese Anhaltspunkte und Ideen für natürliche Anwendungsinterfaces:

- Less is more = Der Versuch möglichst simple Eingabemethoden zu nutzen um komplexe Aufgaben zu bewältigen
- Contextual Environmets = Vom Kontext abhängige, der Aufgabe angemessene Bedienphilosophie. Die Bedienung soll für die jeweiligen zu transportierenden Inhalte arbeiten und nicht umgekehrt
- The Spatial NUI = Nutzen von 3D-Elementen und dem Nutzer so räumliche Interaktion mit der Applikation ermöglichen
- Seamlessness = Den Nutzer in eine Erfahrung integrieren, nicht ihn ein Programm benutzen lassen. D.h. direktes Feedback durch die Anwendung bei jeder Interaktion, flüssige Übergänge zwischen Zuständen im Programm mittels Animationen etc.
- Super-Real = Objekte der Anwendung verhalten sich so, wie sie es in der echten Welt tun würden. Z.B. Schubsen mit dem Finger erzeugt realistische Beschleunigungen, ziehen, drücken erzeugt erwartungsgemäße Reaktionen etc.
- User Differentiation = Auf den User abgestimmte Fähigkeiten und Möglichkeiten in der Anwendung - "Know your User" - die Fähigkeiten, Bedürfnisse und das Umfeld des Nutzers berücksichtigen.
- No Touch Left behind = Jeder Touch sollte ein Feedback geben. Es darf niemals nichts passieren, da der User sonst Fehler in seiner eigenen Bedienung oder dem System vermutet.
- Self-Revealing Gestures = Die Oberfläche sollte so gestaltet sein, dass sie dem Nutzer Gesten aktiv beibringt und diese nicht als ätherische Abkürzungen für Profis erscheinen.
- Know your platform = Das Design einer Anwendung ist inherent von ihren Eingabegeräten abhängig. Beispiel: Paradigmen aus der PC-Welt funktionieren nicht auf Touch-Devices, das diese Paradigmen alle für den Einsatz einer Maus optimiert wurden.
- Vertical, Horizontal and Mobile = Das Design der Anwendung hängt davon ab, wie das Gerät gehalten wird (an der Wand montiert, in der Hand mit Daumen gesteuert, ...)

- User-Derived Interface = Nutzerbefragungen durchführen um herauszufinden, auf welche Art und Weise Nutzer gerne die Anwendung bedienen würden, bzw. Beobachtungen durchführen, was sie versuchen.
- Rapid Iterative Testing and Evaluation = Die technische Basis schaffen schnell Änderungen an einem Prototyp vornehmen zu können, wiederholtes Testen von mehreren Alternativen und so Schritt für Schritt die Anwendung verbessern.

Gerade Letzteres ist eine aussichtsreiche Methode eine Anwendung zu optimieren. Im Rahmen von Usability Tests während der Entwicklung können so Verbesserungen schnell in ein System eingepflegt und erneut evaluiert werden. Man nennt dieses Vorgehen auch "iterative design" [6] Hierbei werden basierend auf den Erkenntnissen aus Studien alter Prototypen Änderungen in den aktuellen Prototyp einer Software eingepflegt. Dieser Prozess wird bis zur Marktreife iterativ wiederholt. Auch "Brave NUI World"[7] nennt dieses Verfahren wie beschrieben als "Rapid Iterative Testing and Evaluation" und misst ihm große Bedeutung zu. Dennoch ist das entscheidende Element dieses auch in dieser Arbeit angewendeten Verfahrens ein konsistenter Usability-Test.

3.3 Usability Tests

Ein grundlegendes Problem der Usability ist ihre Messbarkeit. Es handelt sich nicht um eine physikalische Größe an die man einfach ein Messgerät anlegen kann. Es gibt verschiedene Varianten solcher Usability Metriken (=griechisch für Zählung, allgemein: das Messbarmachen einer Größe), die sich diesem Problem annehmen. Stellvertretend soll hier ein auf ErgoNorm der Bundesanstalt für Arbeitsschutz und Arbeitsmedizin[8] angelehntes Verfahren eingegangen werden, dass versucht die bereits am Anfang des Kapitels zitierten DIN EN ISO 9241 Teile 10 und 11 in einen Methodensatz zur Evaluation von Softwareergonomie zu überführen. ErgoNorm, dass nicht für die Bewertung von Games erstellt wurde, wird dabei entsprechend den Zielen dieser Arbeit diskutiert und modifiziert. Es erscheint zunächst erstaunlich, warum gerade diese Bundesanstalt und ihre Arbeit für einen Fragebogen herangezogen werden soll, der Games bewertet, allerdings gibt es keine öffentlich verfügbaren Fragebögen für Spiele zum Thema Interface-Design. Es gibt Untersuchungen zum abstrakten Prinzip "Spielspaß" bei denen Interfacefragen sicherlich eine Rolle spielen, die aber diesen Teilbereich nicht ausreichend genau erfassen - erst recht nicht beim noch sehr jungen Thema Touch-Devices. Der Fragebogen von ErgoNorm wurde deshalb gewählt, da das Nicht-vorhandensein passender Methoden erfordert bei der Basis der Untersuchungen anzusetzen, und das sind die in der DIN EN ISO 9241 definierten Kriterien. ErgoNorm ist gerade deshalb interessant für die

[6] [Nie94]
[7] [Wix11]
[8] [fAuA00]

Untersuchungen dieser Arbeit, da es versucht sich sehr eng an dieser Basis zu orientieren. Es ist somit ein idealer Startpunkt das Modell ausgehend von internationalen Standards weiter zu entwickeln.

Kurz zusammengefasst besteht das ErgoNorm-Verfahren aus zwei Komponenten. Zum Einen wird ein Benutzerfragebogen zum Einsatz gebracht um so direkt Nutzungsprobleme am "Arbeitsplatz" feststellen zu können. Der zweite Teil besteht darin, dass Programm im Hinblick auf die besagten ISO-Normen hin auf wissenschaftliche Art zu prüfen. Als Grundlage für sämtliche Bewertungen erwartet die ErgoNorm die Notwendigkeit einer gemeinsamen Basis für Experten- und Nutzerbewertung einer Software. Diese Basis ist der Nutzungskontext, der eindeutig definiert sein muss, um Anforderungen an die Software daraus abzuleiten (Etwas, was auch in der ISO-Norm schon gefordert wird). Dies ist auch die Basis für sämtliche Nutzerbefragungen, da diese speziell auf besagten Kontext und typische Aufgaben hin ausgerichtet werden müssen. Vor jedem Test auf Nutzerebene muss also dieses Szenario eindeutig definiert sein und auch konstant gehalten werden, um Vergleichbarkeit einzelner sukzessive nacheinander durchgeführter Tests beizubehalten, was eine Grundforderung an eine Metrik ist.

Für das Thema dieser Arbeit ist die Normenkonformität (Normenkonformität, nicht aber alle der Norm zugrunde liegenden Prinzipien!) nicht so vital wie für ein produktiv eingesetztes Softwareprodukt an einem typischen Bildschirmarbeitsplatz (genau der Nutzungskontext, der von ErgoNorm abgedeckt wird), da bei Games und auch 3D Visualisierungen andere Anforderungen gelten müssen. So kann ein durch die ISO Norm als ineffizient ermitteltes Softwareelement durchaus erwünscht sein, wenn es Dramaturgie, Design damit Spielspaß und damit wiederum Nutzerzufriedenheit fördert. Als plakatives Beispiel seien hier unzählige Rätsel in Adventure-Games genannt, die den Nutzer absichtlich vor Probleme stellen, die er lösen möchte (und das teils absichtlich auf durchaus sehr unergonomische Art und Weise).

> "Video games have different design considerations and usability issues than other types of software. The ISO 9241-11 definition of usability includes three independent measures including efficiency, effectiveness, and satisfaction. In the case of video game usability, effectiveness and efficiency are secondary considerations in relation to satisfaction. A consumer may need to purchase or use other software to perform necessary tasks, but a game is bought on a voluntary basis purely for entertainment value. If a game is not fun to play, it will not sell in the marketplace. To ensure the satisfaction of game players, considerable care is required in the game design process and could be better guaranteed with the use of formal usability evaluation procedures by game developers." [Fed02]

Ein weiteres Beispiel dafür, wie wichtig die Betrachtung des Nutzungskontexts einer Software ist *bevor* irgendwelche Forderungen an die Usability postuliert werden. ErgoNorm

erfasst diesen Spezialfall prinzipiell durch eine an jeden Normbruch anschließende "Signifikanzbewertung" - dennoch muss hier klar zwischen designgeschuldeten absichtlichen Brüchen der Normen und Fehlern in der Entwicklung unterschieden werden, weswegen die Benutzerbewertung an Wichtigkeit gegenüber der Normprüfung durch Experten gewinnt. Bei besagten Anwendungen steht das "Erlebnis" des Nutzers im Vordergrund und nicht die effektive und effiziente Ausführung eines bestimmten Arbeitsablaufs. (oder sogar alternativ: Der zu erfüllende Arbeitsablauf eines Games ist es Zufriedenheit mit allen verfügbaren Mitteln zu erzeugen!). Die Interaktion mit der Software ist im Bereich Games oftmals nicht länger das Mittel zum Zweck, sondern sogar das Ziel selbst.

Daraus kann gefolgert werden, dass für den hier vorhanden Spezialfall diese subjektive nutzerseitige Prüfung einen wesentlich höheren Stellenwert einnehmen muss als die Expertenbewertung und damit die Prüfung von Normkonformität. Grundsätzlich kann allerdings eine rein subjektive Analyse eines Problems nicht die Prüfung der Ursachen und Probleme durch einen Experten gänzlich ersetzen, aber liefert in jedem Fall Ansatzpunkte für eine fortgeschrittene Ursachensuche, da so die Stellen an denen Probleme im speziellen auftreten direkt ganz besonders hervorgehoben werden. Dennoch ist laut ErgoNorm eine Expertenprüfung unabdingbar, da Nutzer dazu neigen sich an Probleme zu gewöhnen und sie somit nicht mehr als solche wahrnehmen, was aber nicht heißt, dass sie keinerlei Relevanz haben.

Es existieren viele Methoden zur Bewertung von Usability, deren Anwendung und Behandlung den Rahmen dieser Arbeit teilweise sprengen würde. Aus diesem Grund muss auf Methoden zurück gegriffen werden, die durch ihre Verbreitung und großen Erfolg in der Vergangenheit aufgefallen und etabliert sind. Diese sind unter anderem aber nicht ausschließlich:

Fragebogen [9]

Der Fragebogen ist der Dreh- und Angelpunkt der subjektiven Analyse auf Nutzerseite. Er ermöglicht das genaue Fragen nach bestimmten Problemstellen in einer validen, nachvollziehbaren und vergleichbaren Art und Weise. Auf die Fragebogenentwicklung wird im nächsten Abschnitt ausführlich eingegangen.

Think aloud protocol [10][11]

Das Think aloud protocol ist eine von Clayton Lewis 1982 entwickelte Methode der Evaluation. Der Nutzer wird angewiesen "laut zu denken" und so Informationen darüber

[9] [Sto02]
[10] [Lew82]
[11] [LR94]

zu liefern, was in ihm während der Nutzung der Software vorgeht. Er soll laut aussprechen was er sieht, denkt, tut und fühlt während er sich mit einer vordefinierten Aufgabe befasst. Beobachter werden angewiesen Notizen zu allem was die Nutzer aussagen anzufertigen ohne zu versuchen selbige zu interpretieren. Aus diesem Grund werden die Testsitzungen oftmals auch auf Video oder als Tonaufnahme aufgezeichnet. Es kann auch zusätzlich mit Hilfsmitteln wie Eyetracking gearbeitet werden bei denen der Blickpunkt und damit der Aufmerksamkeitsfokus des Nutzers überwacht wird, die eine weitere Dimension an Informationen zu der "Tonspur" liefern.

3.3.1 Fragebogenentwicklung nach ErgoNorm

Im Rahmen der hier verwendeten Bewertungstechniken soll vor allem der Fragebogen und das aktive Zuhören genutzt werden. Es existieren einige ISO-konforme Fragebogen-Formen auf dem Markt, jedoch schlägt ErgoNorm berechtigt vor, bei besonderen vom Standard abweichendesn Nutzungsszenarien einen eigenen Fragebogen zu entwickeln oder zumindest einen existenten anzupassen. Da der Nutzungskontext der in dieser Arbeit behandelten Thematik als speziell einzustufen ist, ist dieses Unterfangen unabdinglich. Wichtig ist hierbei, dass der Fragebogen stets mit Blick auf einen bestimmten Arbeitsablauf aufgebaut wird - es geht nicht darum ein generelles Meinungsbild einzuholen, sondern die Applikation in ihrem direkten Nutzungskontext bei wohldefinierten Aufgaben zu bewerten. Es geht auch nicht darum Vergleiche zu anderen Applikationen zu ziehen. Die allermeisten auffindbaren Fragebögen sind laut ErgoNorm auf diesem Vergleichsprinzip aufbauend, aber erzeugen kein stimmiges Gesamtbild für die Frage der Usability. Es werden folgende grundlegende Kriterien definiert, die ein Fragebogen zur Usability-Überprüfung erfüllen sollte: (Kriterien dieses Abschnitts entnommen aus [fAuA00], Kapitel 5)

1. Unterstützung der Phasen der Systementwicklung = Universelle Einsetzbarkeit des Fragebogens zu jeder Phase der Softwareentwicklung (Prototyp, fertiges Produktivsystem, ...)
2. Anonymität der Befragung = Anonymität stellt sicher, dass die Probanden ohne Druck durch Umgebungsvariablen (z.B. Betriebsklima etc.) die Fragen beantworten können.
3. Bearbeitung ohne Expertenbeteiligung = Der Fragebogen sollte möglichst ohne Usability-Experten auswertbar sein, und es soll auch kein Experte anwesend sein, da bereits die Anwesenheit eines Experten die Nutzer derart emotional manipulieren kann, dass sie Fehler unterschlagen oder anders darstellen um in einem positiven Licht zu stehen (o.Ä.)
4. Akzeptanz = Der Aufwand, den es erzeugt den Fragebogen einzusetzen, bzw. die Kosten müssen in akzeptablem Verhältnis zum Nutzen der Maßnahme stehen.

Desweiteren werden an die Qualität der Ergebnisse folgende Anforderungen gestellt:

1. DIN EN ISO 9241 – 10 als Gegenstand korrektiver Dialoggestaltung = Bezug der verwendeten Fragen auf die ISO-Norm, um internationale Vergleichbarkeit der Ergebnisse sicher zu stellen.

2. Verständlichkeit der Dialoggrundsätze = Dem Nutzer muss vorher klar gemacht werden, was gute Software ausmacht, damit er Fehler auch als solche erkennt, da Menschen dazu neigen Probleme im Umgang mit Technologie oftmals als nicht änderbar wahrzunehmen und sie deshalb nicht als Probleme identifizieren. Hier muss ein Bewusstsein für die Problematik geschaffen werden bevor evaluiert werden kann.

3. Eine gemeinsame Beurteilungsgrundlage bei Experten und Benutzern = Experte und Benutzer müssen sich beide auf einen vordefinierten Sachverhalt beziehen, d.h. es ist entscheidend, dass eindeutige Aufgaben definiert werden die im Zuge der Befragung ausgeführt werden.

4. Verständlichkeit der Dialoganforderungen für Benutzer = Die Fragen des Fragebogens müssen so definiert sein, dass der Nutzer sie eindeutig den gefragten Kriterien zuordnen kann und inhaltlich auf die evaluierte Aufgabe abzielen.

5. Psychometrische Qualität = Verlässlichkeit des Fragebogens (Erzeugt unter gleichen Bedingungen vergleichbare Ergebnisse), möglichst gute Erfolgsquote beim Verifizieren von durch einen Expertentest erkannter Probleme, sicherstellen, dass positive und negative Aussagen sich nicht durch eine falsche Auswertungstechnik gegenseitig aufheben (jede Problemmeldung muss einzeln bewertet werden)

Unter Zurhilfenahme von Experten und Nutzertests hat ErgoNorm einen Basisfragebogen entwickelt, der auch hier als Basis dienen soll und an den Nutzungskontext dieser Arbeit angepasst werden soll. (vgl. Anhang-Tabellen zur Fragebogen-Entwicklung). Dem Fragebogen vorausgestellt ist eine kurze Anleitung und die Beschreibung der jeweiligen Aufgabe. Jede Kategorie der Dialoggrundsätze und ihre zugehörigen Fragen haben eine kurze Einführung, die dem Nutzer klar macht was die Erwartungen an die Software sind die er mit seinen Erlebnissen vergleichen soll. Natürlich müssen einige der Fragen an den speziellen Nutzungskontext dieser Arbeit angepasst werden (wie auch in dem Verfahren grundsätzlich vorgesehen). Dazu müssen aber zunächst die Aufgaben ermittelt werden, die dem Nutzer im Zuge des Tests gestellt werden sollen, eben weil die Usability-Bewertung (und das kann nicht oft genug gesagt werden) immer vom speziellen Nutzungskontext abhängt.

3.4 Anpassung des ErgoNorm-Systems

Im vorigen Abschnitt wurde die Basis für die Anpassung des Fragebogens gelegt. Nachdem behandelt wurde, welchen Punkten und Verfahren besondere Wichtigkeit zugestanden werden muss, ist es nun nötig darauf aufzubauen die Anpassungen an den Nutzungskontext dieser Arbeit durchzuführen. Dafür müssen - bevor der Fragebogen selbst modifiziert und ergänzt wird zunächst die zu erfüllenden Aufgaben definiert sein, auf die sich die Fragen des Fragebogens beziehen.

3.4.1 Definition der Aufgaben für den Usabilty-Test

Zur Auswahl aussagekräftiger Aufgaben für den Usability-Test muss geklärt werden was typische Aufgaben für eine 3D-Applikation sein könnten. Generell ist im Zuge dieser Arbeit eine möglichst breit anwendbare Betrachtung der Probleme vorzuziehen, da hier nicht Kriterien für eine einzelne spezielle Applikation sondern solche für ein breites Spektrum an Applikationen aus dem Bereich 3D auf Touch Devices definiert werden sollen. Demnach muss zunächst einmal nach von der Spielmechanik relativ unabhängigen Kriterien oder im Gegensatz omnipräsenten Kriterien die für die allermeisten Spielmechaniken vital sind gesucht werden, die auch jeweils für alle besagte Applikationen eine Problemstellung sind. Schaut man sich die Applikationen aus Kapitel 2 an, so zeichnen sich tatsächlich recht schnell zwei Aktivitäten der Nutzer ab, die immer wieder auftauchen - und zwar Navigation und Interaktion.

a) Navigation

Mit Navigation ist alles gemeint, was mit der Bewegung der Spielerfigur (oder im Falle z.B. einer Architekturvisualisierung der Kamera) durch die Spielwelt zu tun hat. Das heißt zum Einen die Genauigkeit und Benutzbarkeit von Buttons, Sticks und Kontrollen und zum Anderen der Vorgang des “sich orientierens” in der Welt. In den allermeisten Games und Visualisierungen kommt über kurz oder lang der Moment wo ein Nutzer einen bestimmten Ort erreichen will, aus welchem spielmechanischen Grund auch immer das sein mag. Typische Fragen, die sich der Nutzer stellen könnte wären z.B.:

- “Wie bewege ich mich in eine bestimmte Richtung?”
- “Wie ändere ich meinen Blickwinkel?”
- “Wie finde ich Ort x in der 3D Welt?”
- “In welche Richtung muss ich gehen um ’weiter zu kommen’?”

Diese und vergleichbare Fragen in der selben Richtung beschreiben die Problemstellung Navigation. Da keine interaktive 3D-Welt ohne Bewegung der Spielfigur durch selbige

vorstellbar ist und damit zu einem statischen Bild verkommen würde, ist die Nutzerfreundlichkeit dieser Funktionen ein entscheidendes Kriterium für die subjektiv wahrgenommene Qualität der Software. Dementsprechend sollte genau dieser Bereich, unabhängig von jeder "Spielmechanik" geprüft und optimiert werden. Es muss sicher gestellt werden, dass die Steuerung der Bewegung an sich nicht so viel Aufmerksamkeit erfordert, dass sie von der eigentlichen Erfahrung der Simulation ablenkt und damit den Immersionsfaktor reduziert.

Um also in einer zu testenden Aufgabe das gesamte Spektrum des Oberbegriffs "Navigation" zu erfassen kann ein Vergleich zur echten Welt gezogen werden. Seit Jahrhunderten üben Menschen in Hindernissparcours sowohl sich motorischen Herausforderungen zu stellen als auch einen Weg durch den Parcours zu einem Ziel zu finden, der möglichst optimal ist. Vergleichbar wird im Zuge des Tests folgende Aufgabe definiert:

Der Proband soll von einem Startpunkt aus einen bestimmten Ort (z.B. "gehen sie zur Kirche") in der Spielwelt aufsuchen. Die Probanden dürfen sich nicht gegenseitig dabei beobachten und vorher keinerlei Hilfen außerhalb der Applikation gezeigt bekommen. Während der Durchführung der Aufgabe wird der Nutzer angewiesen laut auszusprechen was er denkt. Es wird die Zeit gemessen die der Proband brauchte um die Abgabe abzuschließen. Mit Hilfe des Fragebogens wird anschließend die Wirkung der Aufgabe auf den Nutzer geprüft.

Hierbei ist zu beachten, dass die Zeit die der Nutzer brauchte im Gegensatz zum Hindernissparcours lediglich als Indiz für "Optimiertheit" des eingeschlagenen Lösungswegs dienen kann, nicht aber für den bei Games wesentlich wichtigeren Spielspaß. Es kann natürlich passieren, dass einige Nutzer weil es ihnen gerade zusagt einige Umwege durch die Spielwelt gehen wollen. Das ist nicht als schlecht zu bewerten, sondern kann auch als sehr positiv wahrgenommen werden, da der Immersionsfaktor des Nutzers offenbar recht groß ist und er sich für die Welt interessiert. In jedem Fall muss der Nutzer aber über kurz oder lang des Zielort erreichen um "voran zu kommen" weswegen ein Nicht-erreichen des Ortes einem Nicht-erfüllen der Aufgabe und damit dem Aufgeben des Nutzers und absolutem Versagen der Software gleich käme.

b) Interaktion

Interaktion ist als Überbegriff für sämtliche Aktionen des Nutzers in der Welt zu verstehen. Wenn Navigation ein spielmechanikunabhängiges Betrachtungsmerkmal ist, so ist Interaktion das entrscheidende Merkmal in jeder Spielmechanik. Dabei kann Interaktion als so trivial verstanden werden wie das Drücken eines Lichtschalters in einer Architekturvisualisierung bis hin zu komplexen Rätseln in einem Adventure Game, bei dem beispielsweise Gegenstände sortiert, bewegt, orientiert, usw. werden. All diese Aktionen sollten entsprechend gut für den Nutzer verständlich und erreichbar sein, ohne viel über

das “wie” nachdenken zu müssen. Der Fokus liegt also auf den Komponenten der Aufgabe selber. Diese müssen konsistent (nicht zwangsweise effizient, wie beim bereits genannten Beispiel des ineffizient zu nutzenden Rätsels in einem Adventure Game) an den Nutzer weitergegeben werden. Es ist nach dieser Überlegung von vornherein klar, dass man keine allumfassende Aufgabe definieren kann, die diesen ganzen Teilbereich abbilden kann, da die Interaktionsmöglichkeiten in den Applikationen wesentlich zu vielseitig aussehen können. Also muss man sich einen Spezialfall heraussuchen, an dessen Beispiel der Prüfprozess eines solchen Elements der 3D-Welt verdeutlicht werden kann.

Im Falle des in dieser Arbeit durchgeführten Tests wird die Aufgabe definiert einen Gegenstand, das sogenannte “Artefakt” zu bergen. Das Artefakt ist ein in der Spielwelt versteckter Gegenstand, der mittels der Navigationsfuktionen des vorigen Abschnitts aufgefunden werden soll. Der Nutzer soll dort mit dem Artefakt interagieren, es mitnehmen und zu einem anderen Ort in der virtuellen Welt bringen. Um den Test etwas vielsitiger zu gestalten werden einige aufgestellte Objekte den Weg zum Artefakt blockieren. Der Nutzer muss einen Weg finden dieses Hinderniss zu umgehen, indem er z.B. mit den Fässern interagiert um sie aus dem Weg zu schubsen. Die Aufgabe an sich ist relativ simpel, jedoch ermöglicht sie das Steuerungsparadigma für die Interaktion mit Objekten eindeutig zu prüfen, da der Nutzer gezwungen ist mit Gegenständen zu interagieren um die Aufgabe zu beenden. Auch hier gilt wiederum, ein Aufgeben des Nutzers würde einem völligen Versagen der Software gleich kommen.

3.4.2 Anpassung des ErgoNorm Fragebogens

Der Fragebogen ist bereits sehr allgemein formuliert, und somit können doch erstaunlicherweise bereits viele der Fragen direkt übernommen werden. Im Anhang werden die einzelnen Fragen des ErgoNorm Fragebogens durchgegangen und falls nötig angepasst oder ergänzt. Auffällig dabei ist, dass abgesehen von einigen Begrifflichkeiten der Fragebogen doch eine erstaunliche Übertragbarkeit für Gaming-Anwendungen auf Tablets aufweist. (siehe Tabellen im Anhang - A.1 bis A.7). Es ist überraschend, dass viele der Fragen trotz der Tatsache, dass der Bogen nicht für Gaming vorgesehen war schon gut passen und übernommen werden können. Dies widerspricht in der Tat den deutlichen Hinweisen in ErgoNorm, dass Usability-Tests bis ins letzte immer dem Nutzungskontext angepasst werden müssen und nie direkt wieder verwendet werden können. In der Tat gilt dies ohne Frage nur für die Definition der zugrunde liegenden Aufgaben, da keine einzelne Aufgabe für alle Softwareprodukte abgeleitet werden kann - das ergibt sich schon von selbst. Die deutlichsten Anpassungen im Fragebogen sind Anpassungen an Gaming / 3D-Termini, da der Wortschatz des Fragebogens eher auf Bürosoftware abzielt. Desweiteren müssen an einigen Stellen Fragen hinzugefügt werden, die speziell auf die Verwendung von Tablets abzielen. So muss z.B. gefragt werden, ob wichtige Eingabeelemente von Händen verdeckt werden und ob das Gerät wie erwartet auf Bewegungen reagiert (was

auch bedeuten kann, dass der Nutzer erwartet, dass es nicht reagiert). Im Anhang finden sich die einzelnen Fragen des Fragebogens inklusive einiger Kommentare zur Änderung sowie der endgültige an die Probanden ausgeteilte Fragebogen aus der Usability-Studie.

3.4.3 Durchführung des Fragebogentests

> "The basic findings are that (a) 80% of the usability problems are detected with four or five subjects, (b) additional subjects are less and less likely to reveal new information, and (c) the most severe usability problems are likely to have been detected in the first few subjects." [Vir92]

Die Methodik des Fragebogentests soll folgendermaßen aussehen: Es werden zufällig fünf Probanden, in diesem Fall Studenten ausgewählt. Fünf Probanden gelten nach einhelliger Expertenmeinung bereits als ausreichend die allermeisten Probleme zu identifizieren[12], halten aber den Aufwand für die Befragungen in Grenzen, so dass der Prozess schnell abgeschlossen und sinnvolle Erkenntnisse in die Anwendung eingepflegt werden können. Dem Probanden wird im Fragebogen eine einfache Einführung in wichtige Kriterien der Usability gegeben, damit ihm klar wird auf was besonders geachtet werden soll. Der Fragebogen wird dem Probanden komplett präsentiert und auch die zu erfüllende Aufgabe vorgelegt. Dem Probanden wird dann das iPad mit der laufenden Applikation gegeben und er wird gebeten der Aufgabe nachzugehen aber während dessen laut seine Gedanken zu artikulieren (Think aloud protocol). Die Testphase wird mittels einer Kamera aufgezeichnet und anschließends zusammen mit dem Fragebogen ausgewertet. Im Optimalfall lassen sich durch die Auswertung Folgerungen ableiten, die wieder dazu führen das getestete Produkt an entscheidenden Punkten in der nächsten Phase des Iterative Designs zu verbessern.

3.5 Zusammenfassung

Zusammenfassend lässt sich feststellen, dass viele der auf dem Markt verfügbaren Publikationen zum Thema Usability schon sehr gut mit dem Thema Games und auch der neuen Geräteform der Tablets harmonieren, auch wenn sie nicht explizit für diesen Spezialfall erdacht wurden. Viele Publikationen sind so allgemein gehalten, dass sie sich auf viele Bereiche von Nutzer-Produkt-Interaktion auch außerhalb von Software anwenden lassen. Dennoch müssen diese Evaluationsmethoden, Kriterien und Prozesse, gerade weil Usability immer vom Nutzungskontext und den zu bewältigenden Aufgaben abhängt entsprechend angepasst werden. Die gute Nachricht daran ist, dass das offensichtlich möglich ist. Mit dem angesammelten Grundwissen und den abgeleiteten Kriterien kann

[12] [Vir92]

nun begonnen werden neue Ideen zur Bedienung besagter Applikationen zu entwickeln. Außerdem bietet der Fragebogentest ein brauchbares Werkzeug diese Ideen direkt mit wenigen Probanden schon außerordentlich gut bewerten zu können. Doch zunächst müssen Prototypen entwickelt werden, die überhaupt getestet werden können.

4 Umsetzung des “iterative Design”-Konzepts, Iteration 0

Wie bereits am Ende von Kapitel 2 erwähnt, ist es relativ unverständlich, warum man sich bei Konzept und Fähigkeiten von UI-Elementen standardmäßig den gleichen Limitierungen aussetzt, die auch bei einem physischen Gamepad gegeben sind, wo doch an und für sich die Möglichkeiten des Tablets ganz andere Optionen eröffnen könnten. Dementsprechend scheint in diesem Bereich der dynamischen UI-Elemente auch der größte Raum Innovation auffindbar zu sein. Dies soll auch der Fokus dieses Kapitels sein. Dabei soll getreu dem Motto des zuvor gezeigten Modells des “iterative Designs” 4.1in zwei Iterationen die Oberfläche verbessert und angepasst werden. Iteration 0 stellt die Basis dar, die allein auf Ideen des Entwicklers basiert. Diese werden aufgebaut und in der Engine implementiert. In einem ersten Usability-Test werden diese Ideen dann der Prüfung durch Probanden unterzogen. Die Ergebnisse des Tests werden ausgewertet und anschließend in der Iteration 1 der Benutzeroberfläche umgesetzt. Diese wird erneut mit anderen Probanden evaluiert und es werden Schlüsse für theoretische weitere Iterationen geschlossen.

4.1 Designphase

Für die angestrebte Optik der Oberfläche bietet sich ein Screenshot an, in den mittels Bildbearbeitungsprogramm die Interfaceelemente hinein montiert werden. So kann im vorhinein ein Eindruck gewonnen werden, in welche Richtung anschließend programmiert und gestaltet werden soll, und auch mit stilistischen Elementen ohne großen Programmieraufwand experimentiert werden (siehe Abb. 4.2). Gemäß dem Prinzip des iterative Design werden Konzepte erdacht, diese in der Engine umgesetzt, mittels Tests bewertet und anschließend verfeinert. Somit handelt es sich in diesem Kapitel um Phase 0, da hier noch nicht durch Nutzertests bewertete Konzepte erdacht und umgesetzt werden. In Phase 1 nach dem ersten Usability-Test werden dann basierend auf den Erkenntnissen weitere Verfeinerungen vorgenommen, und bei Bedarf weitere neue Elemente ergänzt.

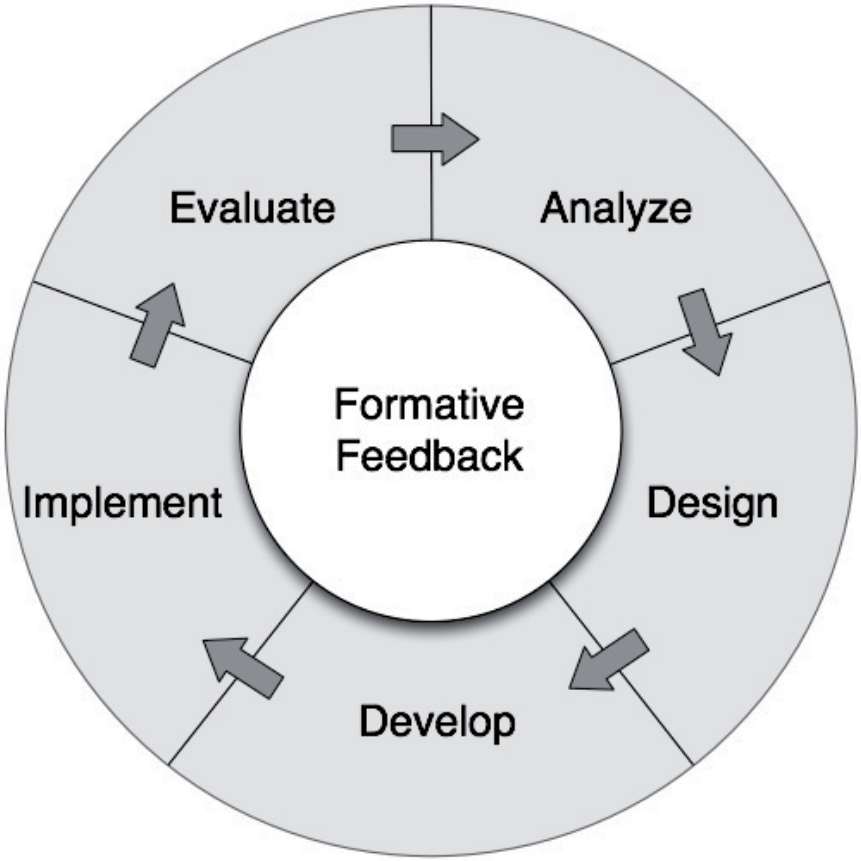

Abb. 4.1: Schematische Darstellung des Iterative Design Process. Die Grafik beschreibt einen andauernden Kreislauf zwischen Evaluation und Entwicklung bis zum Release des fertigen Produkts. Quelle: http://ux-fr.com/2010/08/30/iteration/

4.1.1 Aktiver Stick

Die Grundidee des “aktiven” Sticks ist es, den emulierten Gamepad-Stick mit zusätzlichen Eigenschaften auszustatten, die ihm einen Mehrwert geben. Emulierte Sticks sind prinzipiell nicht schlecht, und so sollte man eher versuchen dieses bereits sehr etablierte Element behutsam zu verbessern und zu erweitern, als ein gänzlich neues Eingabegerät zu erfinden. Allein die Tatsache, dass sich die emulierten Gamepad-Sticks als Standard durchgesetzt zu haben scheinen, spricht dafür sie nicht gänzlich abzuschaffen, da bereits ein breiter Nutzerkreis den Umgang mit dieser Art der Steuerung gewohnt ist. Dennoch: Der Vorteil eines Touchscreens ist eben, dass es sich im Gegensatz zu einem statischen Plastikteil an einem Gamepad, um einen Bildschirm handelt. Das heißt wiederum, dass sich das UI-Element begrenzt mit zusätzlichen Informationen und Funktionen bestücken lässt, zumindest in den Bereichen, die nicht durch den Daumen oder eingestetzten Finger verdeckt werden. Es bietet sich an, da es sich um Bewegungskontrollen handelt auch zu versuchen genau diesen Aufgabenbereich um Funktionalität zu bereichern. In Applikationen derart wie sie in dieser Arbeit betrachtet werden, also sowohl in Games als auch Visualisierungen geht es fast immer darum an bestimmten Orten in der virtuellen Welt mit Dingen zu interagieren - sei es um den Nutzer im Falle z.B. einer Architekturvisualisierung auf besondere Highlights hinzuweisen oder im Falle von Games Missionsziele abzuarbeiten. Diese “Wegpunkte” sind mitunter in verschachtelten Welten schwer zu finden.

Die simpelste und vermutlich (im Usability Test zu prüfen) eingängigste Variante ist ein Pfeil, der in Richtung von besagten Wegpunkten weist. Das heißt für den Steuerungsprototyp soll der rechte Stick (der für die Wahl des Kamerawinkels zuständige) um einen Pfeil erweitert werden, der dynamisch mit dem Stick jederzeit in die Zielrichtung des nächsten Zieles weist. Der Pfeil muss dabei am Rand des Sticks angebracht sein, da er im Zentrum vom Finger des Nutzers überdeckt werden würde und lang und auffällig genug sein um deutlich in der Wahrnehmung des Nutzers zu stehen. Desweiteren soll eine Einblendung die Entfernung zum Ziel verraten, damit der Nutzer abschätzen kann, wie nah er schon an den Wegpunkt heran gekommen ist - etwas was ihn auch im Zweifel motivieren kann nicht die Suche vorzeitig abzubrechen. Auf eine vollständige Minimap wird zunächst verzichtet, da diese nicht den Stick ersetzen könnte (sie wäre die meiste Zeit von einem Finger verdeckt) und würde ansonsten, sofern sie ausreichend groß wäre um wirklich sinnvoll informieren zu können ,einen sehr großen Bereich des Bildschirms bedecken und damit wiederum eventuell die Benutzeroberfläche überladen. Unter Abbildung 4.2 wird ein erster Entwurf der beschriebenen Elemente dargestellt. Im Zentrum befindet sich der Name des nächsten Wegpunktes, ein kleiner werdendes Rechteck im Hintergrund beschreibt den Abstand zu selbigem - eventuell reicht hierfür auch eine Zahlenangabe. Der Pfeil am Stick auf der rechten Seite ist noch recht klein im Entwurf, dies kann aber, da es sich bei dem Bild lediglich um eine Designstudie handelt, im Laufe der Programmierung, sollte es sich als sinnvoll erweisen, angepasst werden.

4.1.2 Kontextsensitive Interaktion

In Kapitel 2 gab es erstaunlich viele Beispielapplikationen mit statischen Buttons, die Aktionen und Interaktionen mit der Spielwelt auslösen. Auch hier soll versucht werden diese Elemente dynamischer zu gestalten. Kontextsensitive Buttons erscheinen hier als Mittel der Wahl. Das heißt abhängig von den im Blick befindlichen Elementen der Spielwelt sollen dem Spieler passende Interaktionsmöglichkeiten und damit auch Bedienelemente angezeigt werden. Desweiteren soll der Nutzer explizit auf Gegenstände hingewiesen werden, mit denen er interagieren kann, und diese - sofern möglich - direkt mittels Touchscreen manipulieren können. So soll ein Overlay im Sichtfeld des Nutzers benutzbare Gegenstände markieren, sofern sie in Interaktionsreichweite sind, und durch einen Druck mit dem Finger auf den Gegenstand soll direkt an dieser Position ein Menü erscheinen, was alle relevanten Interaktionen anbietet. Auf Abbildung 4.2 wird ein solches System mittels Bildbearbeitungsprogramm demonstriert. Alle interagierbaren Gegenstände werden mit kleinen Markern versehen, die den Nutzer auf Möglichkeiten der Interaktion hinweisen. Sobald dieser den betreffenden Gegenstand bzw. den Marker mit dem Finger berührt, erscheint eine Reihe vom ausgewählten Objekt abhängiger Buttons (hier 1,2,3) die alle möglichen Aktionen anbieten. Ein Button zum Abbrechen erscheint ebenfalls am rechten Bildschirmrand (bzw. am Linken Bildschirmrand für Linkshänder) um den Interaktionsmodus zu beenden. Der Button soll an der rechten Seite auftauchen, da dies die Primärhandseite des Nutzers ist. Ein Wegbewegen vom ausgewählten Gegenstand muss dennoch möglich sein und soll bei entsprechender Distanz ebenfalls automatisch ein Abbrechen der Aktion auslösen. Die kontextsensitiven Buttons müssen die ganze Zeit auch bei Änderung des Blickwinkels eine relative Position zum zugehörigen Objekt bewahren, damit sie eindeutig zugeordnet werden können. Außerdem sollten sie oberhalb des Touchpunkts erscheinen, damit sie nicht von der Hand verdeckt und direkt genutzt werden können.

4.1.3 Accelerometer

Überlegenswert ist schließlich noch die Integration der Bewegungssensoren in diese Applikation - etwas was für diese Art Programm bisher auf dem Markt recht unüblich ist. Es kann darüber diskutiert werden, ob es sinnvoll ist ein System einzubauen, dass vergleichbar zur Kamerasteuerung bei Rage ist (Kippen der Kamera durch Kippen des Tablets, siehe Kapitel 2). Absehbare Probleme wären, dass Buttons, die sich mit der Kamera bewegen, so wie sie für diesen Prototyp geplant werden, extrem schwerer bedienbar wären wenn Bewegungen des Tablets die Kamera weg drehen, zumal der Anwender das Tablet kaum stillhalten können wird, wenn er eine Hand zum antippen eines Gegenstands oder dessen Interaktionsoverlay bewegt. Dies erfordert das Tablet in einer Hand zu halten während man den Druck durchführt, und jedes Zittern hätte eine

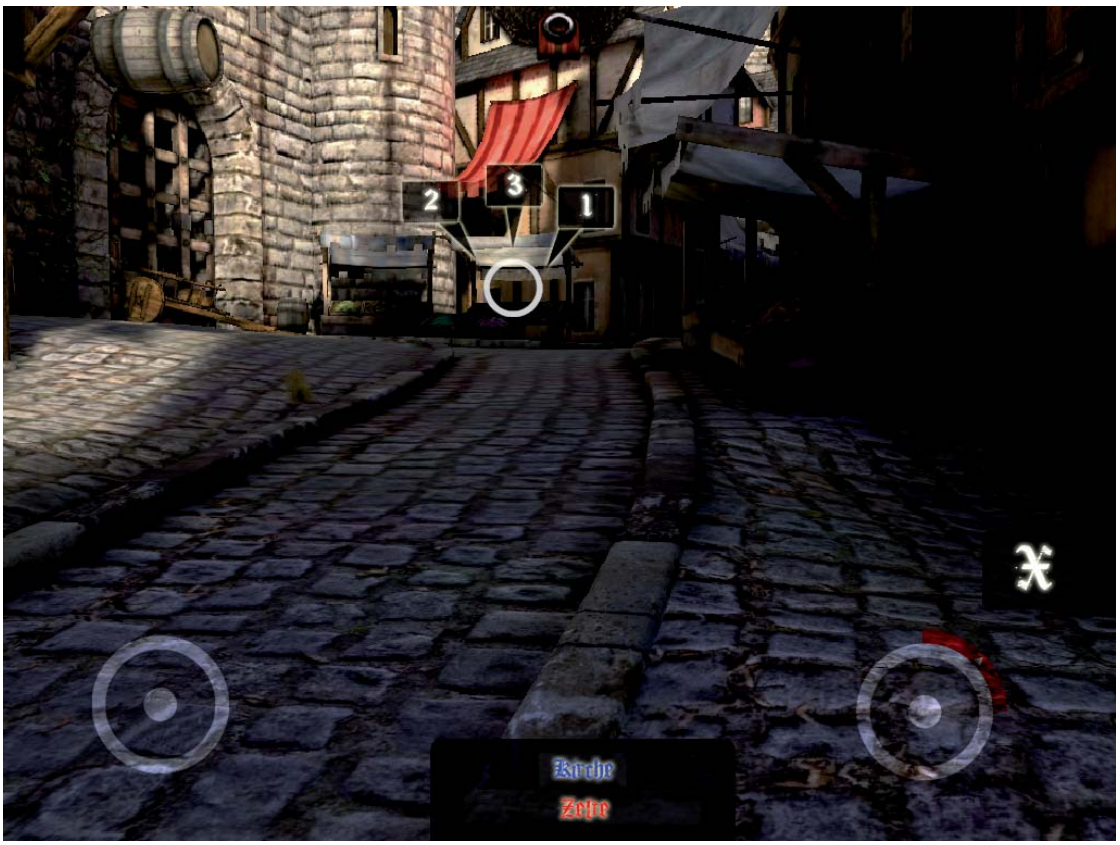

Abb. 4.2: Erster Entwurf des der Steuerungsmodifikationen - Bildmontage mit Hilfe von Photoshop (Adobe Software) in der Epic Citadel Applikation (Epic Games)

Bewegung der Buttons zur Folge - etwas was sicher zusätzlichen Stress beim Bedienen auslöst, auch wenn das "Umsehen" sicherlich angenehm wäre, solange es nichts gibt mit dem interagiert werden soll. Die Frage die sich stellt ist, ob eine solche Funktion im Kontext dieser Applikation eher verwirrend für den Nutzer ist oder nicht. In Iteration 0 soll deshalb hierauf zunächst verzichtet werden, aber dennoch im Fragebogen danach gefragt werden ob das Programm erwartungsgemäß auf Bewegungen des Tablets reagiert hat. Wird diese Frage trotz Nichtimplementation der Funktion mit "ja" beantwortet, kann davon ausgegangen werden, dass die Erwartungshaltung genau der Tatsache entspricht, dass eben das Accelerometer keinen Einfluss auf die Kamera hat. Falls die Frage mit "nein" beantwortet wird, soll solch eine Funktion in Iteration 1 integriert werden. Davon abgesehen kann aus den Videoaufzeichnungen des Usability-Tests eventuell abgeleitet werden, ob eine Bewegungssteuerung überhaupt gesucht oder ausprobiert wurde. Nach der ersten Usability Studie muss die Frage der Accelerometer-Integration definitiv neu bearbeitet werden.

4.2 Implementierungsphase

Mit Hilfe der Unreal Engine - eine bereits seit Jahren aus dem PC-Bereich bekannte und sehr erfolgreiche 3D-Engine sollen nun die neuen Funktionen implementiert werden. Die vom Hersteller Epic hergestellte Engine wurde bereits bei vielen großen PC-Games und einigen Architekturvisualisierungen eingesetzt und ist über die Jahre stetig weiter entwickelt worden. Seit Neuestem ist besagte Engine schließlich auch fähig für iOS Geräte (wie z.B. das iPad) Anwendungen zu kompilieren und steht im Internet in einer kostenlosen Variante mit dem Namen UDK zur Verfügung.

4.2.1 Entwicklung mit UDK - Einführung

Das UDK besteht aus zwei großen Bereichen, die für diese Arbeit relevant sind. Zum Einen ist der sehr mächtige und komplexe Unreal Editor (siehe Abb. 4.3) zu nennen, der grundsätzlich Levelleditor ist, aber auch Funktionen für Animationen, Texturen, Models und visuelles Scripting mitbringt. Vor allem letzteres - Kismet genannt, ist für dieses Kapitel relevant. Der andere große wichtige Komplex des UDK ist das Unreal Script[1]. Hierbei handelt es sich um eine an Java angelehnte Scriptsprache, mit der sämtliche im Hintergrund auszuführenden Aktionen definiert werden können, also auch eigene Klassen für sämtliche denkbaren Funktionen in die Engine eingebracht werden können (Beispiel siehe Listing4.1). Somit gibt es zwei Arten Abläufe in der Engine zu programmieren - besagtes Unreal Script und das bereits erwähnte Kismet. Kismet ist ein Scriptsystem, dass auf dem Platzieren von Logikknoten und deren Verbindungen auf einem grafischen

[1] [Fly06, Bus10]

Zeichenbrett basiert und somit optisch etwas ergibt, was Flussdiagrammen oder abstrakt dargestellten Stromkreisen ähnelt[2]. Events, also Ereignisse, die von Objekten in der Spielwelt ausgelöst werden können geben einen Impuls an weitere dieser Logikknoten (nachfolgend Nodes genannt) ab und können auf diese Weise Aktionen, Rechnungen etc. auslösen. Besagte Logikknoten können Ein- und Ausgänge für Variablen, Objekte oder weitere Logikknoten aufweisen (siehe Beispiel in Abb. 4.4). Auf den ersten Blick scheint es so, dass dies redundant zum Unreal Script ist, indem sich derartige Abläufe ebenfalls mittels echtem Programmcode implementieren ließen. Das ist überwiegend korrekt - die Stärke des Systems liegt allerdings in der Kombination beider Engine-Elemente. So können mittels Unreal Script eigene Kismet-Nodes geschrieben werden, die Spezialfälle abdecken können, die im mitgelieferten Standardsatz an Nodes nicht verfügbar wären. Der besondere Vorteil des Kismet-Systems ist auch, dass es im Gegensatz zum Unreal Script nicht nach jeder Änderung neu kompiliert werden muss und somit ein wesentlich schnelleres und intuitiveres Erstellen von Mechaniken im Programm ermöglicht. Zudem ist es mit Kismet wesentlich einfacher auf dynamisch zur Laufzeit des Programms erzeugte Objekte einzuwirken und zuzugreifen, was ebenfalls Vorteile im Workflow bietet. So werden mittels Unreal Script breit einsetzbare Funktionen in neue für das eigene Programm notwendige Nodes geschrieben und diese dann dynamisch in Kismet verwendet, um die gewünschte Funktionalität umzusetzen.

Jede Kismet Node beherbergt neben Ein- und Ausgängen zusätzlich eine Reihe von Properties, die es ermöglichen statische Optionen für die Node zu setzen. So kann z.B. ein festes Delay - eine Verzögerung eingestellt werden, die aussagt wie lange die Node wartet bis sie einen Impuls an ihren Nachfolger weiterreicht. Auch sind oft Farben für zu rendernde Objekte oder generelle Optionen in diesen “Properties” einstellbar. Grundsätzlich wird ein solches Property immer überschrieben, wenn die Node einen sichtbaren Eingang für die in den Properties ebenfalls vorhandenen Variablen besitzt, und an diesem etwas angekoppelt ist. Die logischen Verbindungen haben somit immer Priorität über den im Properties eingestellten Werten. Nodes an sich nehmen immer Einfluss auf Objekte in der Spielwelt (sogenannte Actors) oder führen mathematische und logische Operationen durch.

Für die Entwicklung von Benutzeroberflächen auf mobilen Geräten sind eine Reihe Klassen und Konzepte besonders relevant. Zunächst sei die Canvas-Klasse genannt. Das Rendering aller Interface-Elemente wird mit der Canvas-Klasse bewerkstelligt. Besagte im UDK mitgelieferte und per Unreal Script greifbare Klasse ermöglicht es simple Formen und 2D-Texturen in die Oberfläche der Anwendung zu rendern. Kismet bietet dabei eine kleine Auswahl Nodes an, die die Canvas-Klasse implementieren. Besagte Kismet-Nodes bieten theoretisch die Möglichkeit mittels Vektorvariablen Positionen von Interface-Elementen

[2] [Bus09]

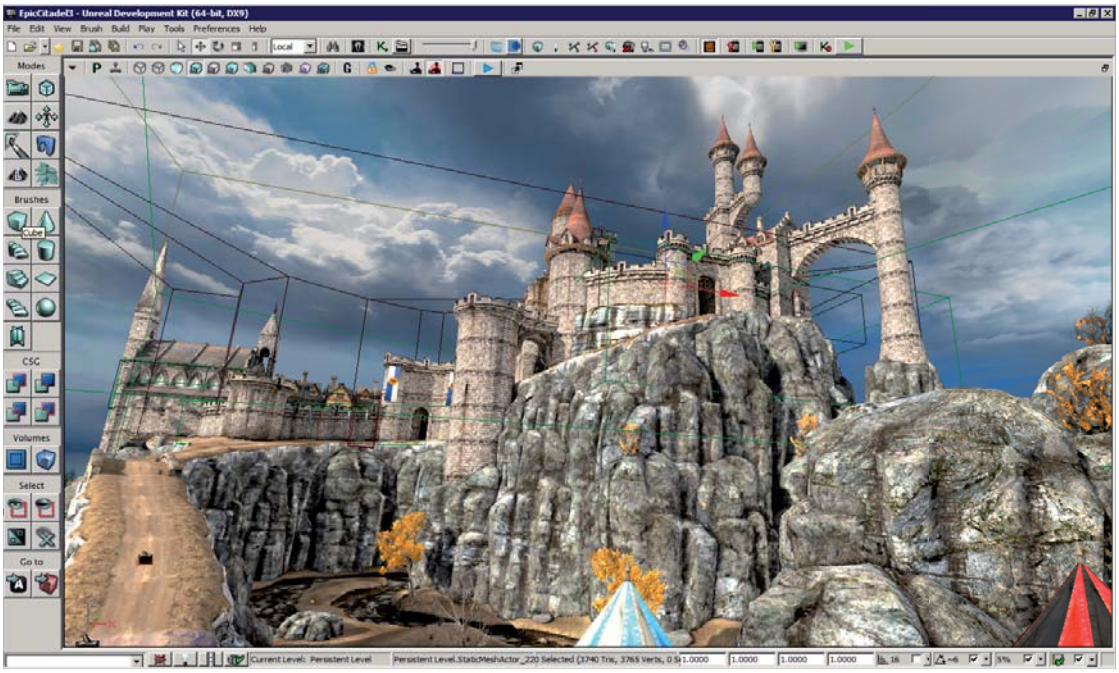

Abb. 4.3: Screenshot des Unreal Editors, das zentrale Werkzeug der Entwicklungsumgebung

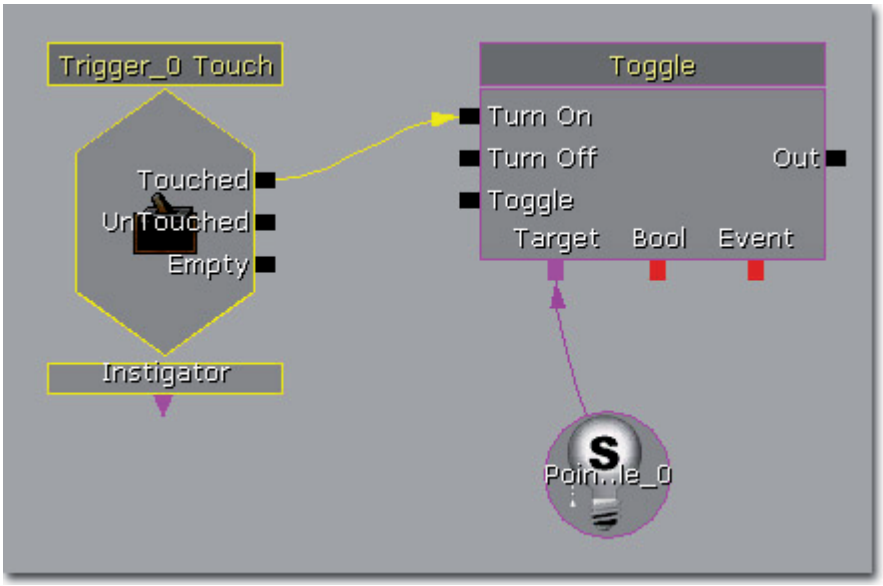

Abb. 4.4: Ein simples Kismet-Programm: Der Event “Trigger_0 Touch” löst bei Berührung der Spielfigur mit einem sogenannten Trigger in der Spielwelt einen Toggle-Node aus, der wiederum ein Licht-Objekt aktiviert. In der Applikation geht also das Licht an, wenn der Nutzer einen Raum betritt.

Listing 4.1: Ein kurzes Beispiel für Unreal Script. Die Verwandtschaft zu Java, bzw. C ist für den Experten sofort ersichtlich. Es handelt sich um einen Auszug eines Unreal Scripts zur Implementierung eines mittels Scriptaufrufen ein- und ausschaltbaren Lichts. (Quelle: Epic Games)

```
1  //=============================================================================
2  // TriggerLight. // A lightsource which can be triggered on or off.
3  //=============================================================================
4  class TriggerLight extends Light;
5  //-----------------------------------------------------------------------------
6  // Variables. var() float ChangeTime;
7  // Time light takes to change from on to off. var() bool bInitiallyOn;
8  // Whether it's initially on. var() bool bDelayFullOn;
9  // Delay then go full-on. var ELightType InitialType;
10 // Initial type of light. var float InitialBrightness;
11 // Initial brightness. var float Alpha, Direction; var actor Trigger;
12 //-----------------------------------------------------------------------------
13 // Engine functions.
14 // Called at start of gameplay.
15
16 function BeginPlay() {
17
18     // Remember initial light type and set new one. Disable( 'Tick' );
19     InitialType = LightType; InitialBrightness = LightBrightness;
20     if( bInitiallyOn )
21     {
22         Alpha = 1.0; Direction = 1.0;
23     } else {
24         LightType = LT_None; Alpha = 0.0; Direction = -1.0;
25     }
26 }
27 ...
```

zur Laufzeit in Echtzeit zu ändern, haben aber in der für diese Arbeit genutzten Version vom UDK (June 2011) einen Bug, der es erforderlich macht die Nodes als Workaround zu modifizieren, da die in Kismet angekoppelten Positions-Variablen sonst ignoriert werden - bzw. genauer gesagt, zuvor erwähnte Priorität zwischen angekoppelten Variablen und Properties in diesem speziellen Fall versagt und somit alle angekoppelten Nodes ignoriert werden. Zu diesem Zweck muss ein Codefragment im Script der Node implementiert werden, dass abhängig vom Namen des Eingangs des Nodes und der Vorhandenheit einer angekoppelten Variable manuell die Variable einliest und in die Properties schreibt. Die in allen anderen Nodes funktionierende automatische Funktionalität der Engine hierfür versagt aus ungeklärten Gründen in den auf die Canvas-Klasse zugreifenden Nodes - ein Umstand, der wenn er nicht bekannt ist sehr viel Zeit und Aufwand verschlingen kann.

Ein weiteres sehr wichtige Konzept für Anwendungs-Interfaces auf Mobilgeräten in UDK sind sogenannte Input Zones. Dabei handelt es sich um in einer Konfigurationsdatei oder per Kismet definierte Zonen auf dem Touchscreen, die eine bestimmte Funktionalität erhalten. Eine Input Zone kann alles von einem Button bis zu einem komplexen Bewegungsstick sein. Die Input Zones und ihre Ausgaben können aus Kismet heraus erzeugt, deaktiviert und ausgelesen werden. Somit ist es möglich zur Laufzeit neue Buttons einzublenden oder Steuerungselemente zu deaktivieren (zum Beispiel während Zwischensequenzen oder Videos). Eine Input Zone definiert sich unter anderem durch Art (Button, Slider, Stick, ...), Position, Größe und ausgeführte Aktion - es können so zum Beispiel auch klassische Keyboard-Eingaben über Input Zones emuliert werden.

Abschließend muss noch die Begrifflichkeit des Triggers erwähnt werden. Trigger sind unsichtbare, meist zylindrische, fest definierte Bereiche innerhalb eines Levels, die eine Aktion auslösen, sobald ein Spieler sie ansieht, sie berührt oder ähnliches. Sie werden für alle Arten von Interaktionen verwendet, die an bestimmten Orten eines Levels verfügbar sein müssen. Trigger sind durch ein charakteristisches Schalter-Symbol und ein grün dargestelltes zylindrisches Volumen im Leveleditor repräsentiert. Ein Beispiel eines Triggers findet sich in Abb.4.23.

4.2.2 Wegpunkt-System und Richtungspfeil

Das Wegpunkt-System soll dem Nutzer erleichtern sich in der Welt zurecht zu finden, indem ein Pfeil stets in die Richtung des nächsten Zieles auf seiner Aufgabenliste weist. Der Pfeil wird dabei an dem rechten Stick orientiert, der der Wahl des Kamerawinkels dient und rotiert um selbigen herum. Er zeigt immer in Luftlinie in Richtung des Zieles und ignoriert Höhenunterschiede. Die Höhenunterschiede sind in diesem speziellen Fall nicht relevant, da es sich bei der Epic Citadel um eine Spielwelt handelt, die nicht mehrere Stockwerke wie Keller und dergleichen enthält. Hierbei würde das System mitunter versagen, wenn man über oder unter dem Zielpunkt stehen würde. Der Pfeil soll sich in

etwa verhalten wie ein Kompass, und sich in Echtzeit mit den Bewegungen des Nutzers mit drehen.

Zunächst muss man sich über die notwenige Theorie, die hinter einem solchen auf ein Ziel weisenden Pfeil steht auseinandersetzen. Die 3D-Welt und ihre Objekte definieren sich über Vektoren. Da der Pfeil ein zweidimensionales Interface-Element werden soll, soll auf die Anzeige des Höhenunterschieds zum Ziel ähnlich wie bei einem PKW-Navigationsgerät, verzichtet werden. Jedem Objekt in der Spielwelt (UDK Terminus “Actor”) wird ein Positionsvektor und eventuell - sofern die Rotation relevant ist - ein Blickrichtungsvektor zugeordnet - der vom Nutzer kontrollierten Spielfigur durch dessen Augen er sieht somit in jedem Fall, da er diese und damit den Kamerawinkel rotieren kann. Der Navigationspfeil muss diese Rotation natürlich berücksichtigen und in Echtzeit umsetzen. Somit ist es das rechnerische Ziel den Winkel zwischen Blickrichtung und Zielrichtung zu erhalten, und diesen dann auf den Pfeil im HUD des Nutzers anzuwenden, und zwar genau so, dass der Pfeil nach oben (für den Nutzer gleichbedeutend mit geradeaus, da analog zu der Stickbewegung die er zum geradeaus-gehen nutzt) zeigt, wenn er in Richtung des Zieles blickt. Dafür sind mehrere aus der Vektorrechnung bekannte Operationen nötig[3]. Abbildung 4.5 zeigt die Situation inklusive aller bekannten aus der Engine abrufbaren Werte. Zunächst muss aus der Vektor zwischen der Position des Zieles und der Spielfigur ermittelt werden, damit dieser anschließend zur Blickrichtung in Relation gesetzt werden kann. Mittels Vektorsubtraktion ist es möglich den besagten Vektor zu berechnen.

$$\overrightarrow{WegpunktPosition} - \overrightarrow{Spielerposition} = \overrightarrow{RichtungsvektorZumZiel}$$

Der Ergebnisvektor zeigt von der Spielfigurposition in Richtung des Ziels - seine Länge gibt die Distanz zum Ziel an. Der nächste Schritt ist die Berechnung des Drehwinkels zwischen Blickrichtungsvektor und Zielvektor. Hierfür kann das Skalarprodukt herangezogen werden. Die Vektoren werden zunächst normiert, um die Werte in eine leicht verwendbare Skala zu transformieren.

$$cos(\alpha) = \overrightarrow{RichtungsvektorZumZiel} \bullet \overrightarrow{Blickrichtungsvektor}$$

Allerdings erzeugt die einfache Verwendung des Skalaprodukts ein Problem. Selbiges ist grundsätzlich nur für 0 bis 180° definiert, also für den eingeschlossenen Winkel (bzw von -1 bis 1, wenn der cosinus nicht aufgelöst wird, was die Verwendung im Programmcode einfacher macht). Das ist im Falle der Pfeilrotation ungünstig, da für jeden Wert, den die Rechnung ergibt zwei mögliche Drehwinkel für den Pfeil in Frage kommen würden (siehe auch Abb. 4.6). Das heißt vorgeschaltet muss untersucht werden, ob der Zielvektor links oder rechts des Blickwinkelvektors liegt. Auf diese Weise kann der komplette Drehwinkel von 0-360° errechnet werden. Zur Prüfung ob der Vektor rechts oder links liegt wird folgende Bedingung herangezogen:

[3] alle Rechnungen des Kapitels nach[Pap01]

$$x = \overrightarrow{RichtungsvektorZumZiel}.Y \circ \overrightarrow{Blickrichtungsvektor}.X - \overrightarrow{Blickrichtungsvektor}.Y \circ \overrightarrow{RichtungsvektorZumZiel}.X$$

Falls x größer als 0 ist befindet man sich rechts vom Vektor (damit von 0-180°) und falls x kleiner als 0 ist links davon (damit 180-360°). Somit kann durch diese vorgelagerte Unterscheidung das jeweilige Intervall von -1 bis 1 auf die entsprechenden Intervalle umgerechnet werden (0-180 bzw. 180-360).

Im UDK musste zunächst gerade vorgestellte Vektorrechnung in eine neue Kismet-Node geschrieben werden, da ein verwendbares Konstrukt nicht im Standardsatz existierte. Zu diesem Zweck wurde die Node in Unreal Script geschrieben und in die Engine kompiliert, um sie dann zur Laufzeit im Level für verschiedene Probleme einsetzen zu können. In der Engine mitgeliefert ist eine Node zum Berechnen des Abstands zweier Actors, die für das Wegpunkt-System übernommen werden kann.

Im Skript für die neue GetHeading Node (siehe Listing 4.2) wird die ganze zuvor erklärte Rechnung in Unreal Script durchgeführt. Das Ergebnis muss zuletzt in sogenannte Unreal Units konvertiert werden, die Ausgabe des Skalarprodukts geht standardmäßig von -1 bis 1 für jeden 180° Abschnitt. Die Skala für Unreal-Rotationen geht hingegen von 0 bis 65536. Entsprechend werden die beiden Intervalle, je nachdem auf welcher Seite sich der Vektor befindet, mittels Operationen auf 0-32768 bzw 32768-65536 konvertiert und dann ausgegeben. Die Node wird anschließend in den Kismet-Scripten zur Bestimmung aller für das Wegfindungssystem nötiger Variablen verwendet. Die gesamte Berechnung wird immer dann durchgeführt wenn der Spieler den Bildschirm berührt, was als Auslöse-Event sehr gut geeignet ist, da er das zwangsläufig tun muss, um die Kontrollen zu bedienen (siehe Abb. 4.7). Für spätere Verwendung wird dann zunächst der Abstand zum Ziel bestimmt (siehe Abb.4.8). Der neue Knoten wird anschließend für die Winkelberechnung (siehe Abb. 4.9) herangezogen.

Zum Rendern des Pfeiles an den Kamerastick muss zunächst die Position des selbigen von Kismet gelesen werden. Zu beachten ist hierbei, dass der auf die Input Zone zugreifende Event “Mobile Input Access” das Zentrum des Sticks ausgibt, die Klasse Canvas aber immer von der linken oberen Ecke als Basis für die Positionierung von UI-Elementen ausgeht. So muss die Größe der Stick-Grafik bedacht werden und die obere linke Ecke des Sticks als Position für die Pfeilgrafik errechnet werden (siehe Abb. 4.10). Schlussendlich müssen alle nötigen Informationen zusammen getragen und mit einem Draw Image-Event gerendert werden. Selbiger ist in der Grundausstattung allerdings nicht fähig rotierte Texturen zu rendern, und muss deshalb erneut in Unreal Script etwas angepasst werden, damit er Rotationswerte aus Kismet akzeptiert (siehe Listings 4.3und4.4). Wie bereits eingangs erwähnt befindet sich im Juni 2011 UDK ein Bug, der das Auslesen von an die Node angekoppelten Variablen verhindert. Dies erfordert eine Reihe zusätzlicher

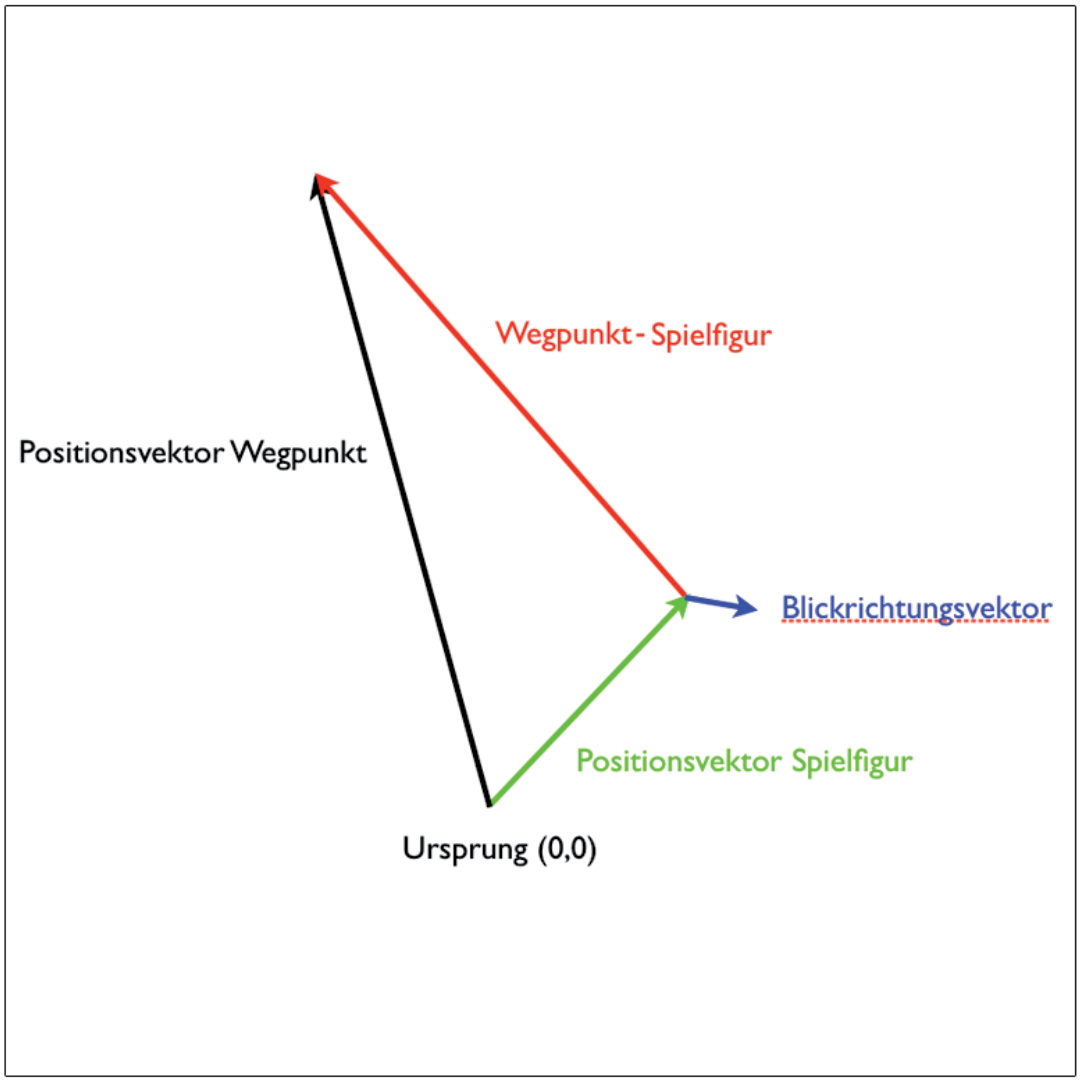

Abb. 4.5: Vektorsubtraktion zur Bestimmung des Richtungsvektors von der Spielerposition zum Ziel

Listing 4.2: Die neue Kismet-Node “Get Heading”, die Basis für die Darstellung des Navigationspfeils

```
1  class SeqAct_Heading extends SequenceAction;
2  // Erweiterung der Klasse SequenceAction, die Kismet Nodes implementiert
3
4  //Definition von Variablen
5  var vector PlayerPos;
6  var vector ObjPos;
7  var vector PlayerRot;
8  var float angle;
9
10 function Activated()
11 {
12     local vector temp;
13     // Vektorsubtraktion zum errechnen des Zielvektors
14     temp = Normal(ObjPos - PlayerPos);
15
16     //Bestimmung ob Zielvektor rechts oder links vom Blickwinkelvektor
17     if ((temp.Y*PlayerRot.X - PlayerRot.Y *temp.X) < 0 ) {
18     //linker Quadrannt - Skalarprodukt und Konvertieren in Unreal Units
19     angle = (PlayerRot dot Normal(ObjPos - PlayerPos)+3)*16384;
20     } else {
21     //rechter Quadrant - Skalarprodukt und Konvertieren in Unreal Units
22     angle = ((PlayerRot dot Normal(ObjPos - PlayerPos)*(-1))+1)*16384;
23     }
24
25 }
26
27 // Definition aller Ein- und Ausgänge der neuen Kismet Node
28 defaultproperties {
29     //Name und Kategorie der Node
30     ObjName="Get Heading" ObjCategory="Actor"
31
32     //Standard Ein- und Ausgänge zum aktivieren nachfolgender Nodes in
           der Kette
33     InputLinks(0)=(LinkDesc="In")
34     OutputLinks(0)=(LinkDesc="Out")
35
36     //Skriptvariablen werden exposed und damit in Kismet greifbar
37     VariableLinks(0)=(ExpectedType=class'SeqVar_Vector',LinkDesc="
           PlayerPos",PropertyName=PlayerPos)
38     VariableLinks(1)=(ExpectedType=class'SeqVar_Vector',LinkDesc="
           PlayerRot",PropertyName=PlayerRot)
39     VariableLinks(2)=(ExpectedType=class'SeqVar_Vector',LinkDesc="ObjPos"
           ,PropertyName=ObjPos)
40     VariableLinks(3)=(ExpectedType=class'SeqVar_Float',LinkDesc="
           Rotation_Float",bWriteable=true,PropertyName=angle)
41     //bWriteable Ausgänge sind schreibend, geben Variablen nur aus und
           lesen nicht.
42 }
```

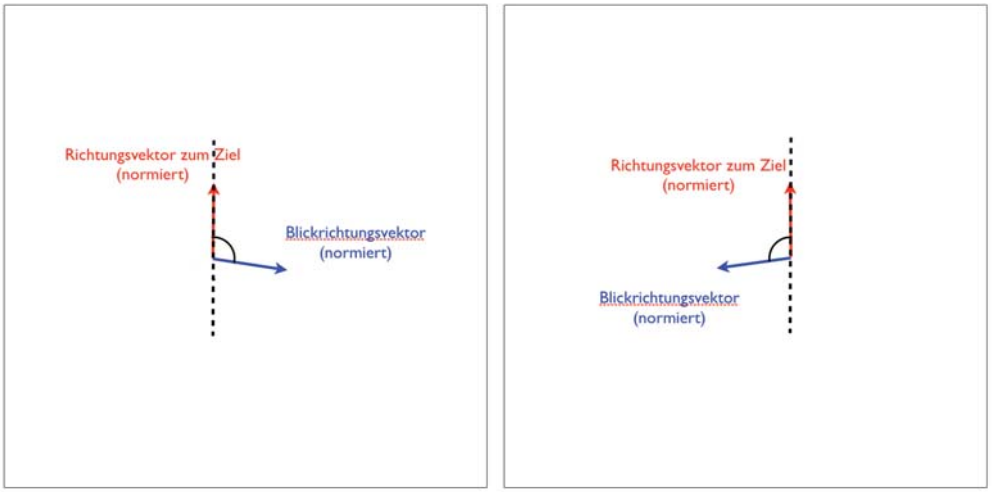

Abb. 4.6: Das Skalarprodukt zur Berechnung des Zwischenwinkels gibt immer den spitzen Winkel aus. Daher muss zusätzlich berechnet werden, ob der Zielvektor rechts oder links vom Blickwinkelvektor liegt, da sonst unklar ist in welche Richtung der Pfeil im Interface rotiert werden muss.

Abfragen innerhalb des Unreal Scripts, die natürlich auch für die neuen Variablen nötig sind. Es wird ein neuer Eingang für die Node definiert, der in Form einer Vektorvariable erlaubt Drehwinkel und Drehpunkt während der Laufzeit in die Node einzugeben. Der ursprüngliche Aufruf der Canvas Klasse in besagter Node muss durch eine andere Funktion besagter Klasse ersetzt werden, der die Funktionalität zur Rotation bietet. Dieser wird mit den neuen zusätzlichen Variablen befüllt. Schlussendlich wird der modifizierte Draw Image-Event dann in Kismet eingebracht und mit allen errechneten Werten befüllt (siehe Abb.4.11). Als Folge daraus wird der Pfeil erfolgreich in die Benutzeroberfläche gerendert - seine Position passt sich immer der Position des zugehörigen Sticks an und er prüft seine Orientierung immer dann, wenn der Touchscreen benutzt wird. Zusätzlich wurde noch mittels Canvas-Node zur Ausgabe von Text die Ausgabe des Distanz-Parameters implementiert. Die Anzeige der Distanz ist in Version 0 ein simpler Text im Zentrum des unteren Bildschirmdrittels. Die Integration der Funktion ins Skript erfolgt analog zu den anderen Canvas-Nodes und erfordert keine speziellen Anpassungen. Damit sind diese Steuerungselement erfolgreich integriert (siehe Abb. 4.12).

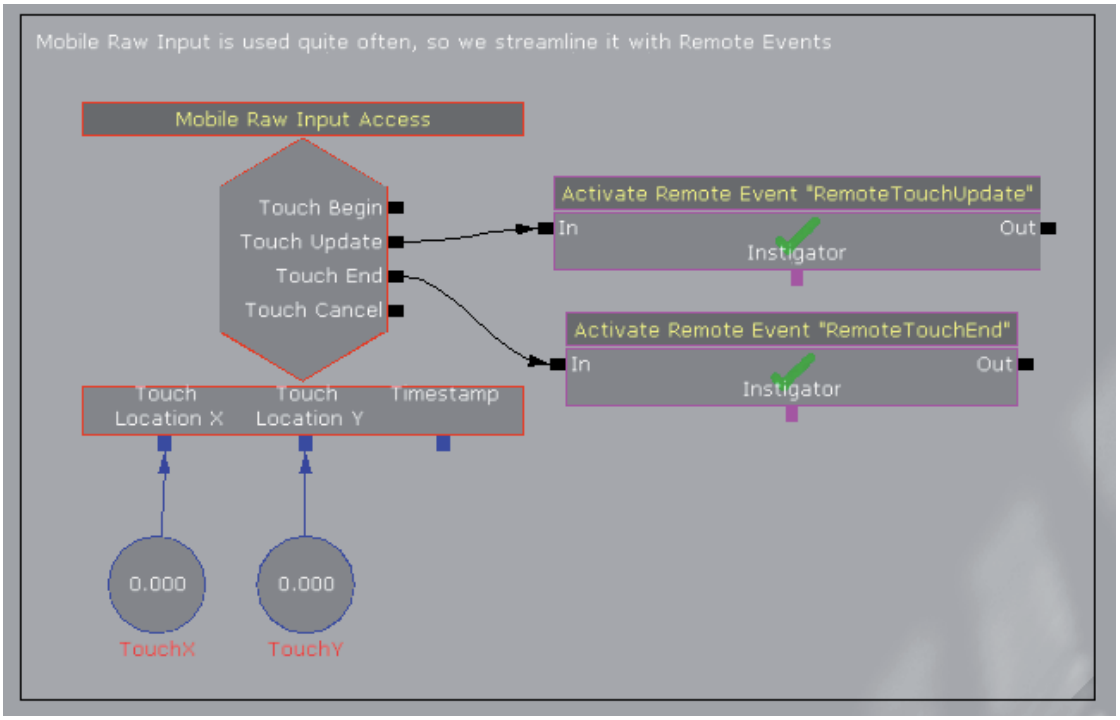

Abb. 4.7: Mobile Raw Input ermöglicht grundsätzliche Zugriffe auf Touchscreen-Ereignisse. Touch Update aktiviert immer wenn sich ein Finger über den Touchscreen bewegt, Touch End wenn der Finger herunter genommen wird. Gleichzeitig werden die X und Y-Korrdinaten der Berührung in Variablen geschrieben. Die Remote Events können den Ausgabeimpuls eines Events auf viele Orte innerhalb des Kismet-Konstrukts verteilen und ermöglichen so mehr Ordnung in den Code zu bringen, da ansonsten lange Linien quer durch die Arbeitsfläche gespannt werden müssten.

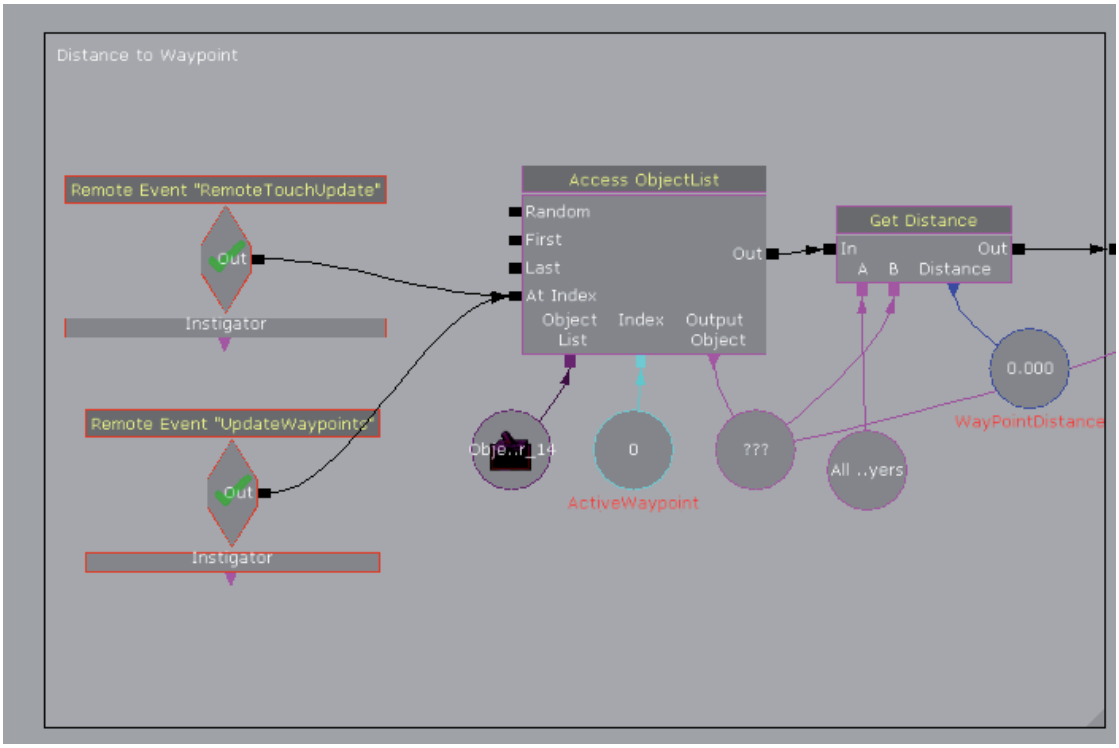

Abb. 4.8: Mittels Remote Events wird die Berechnung aktiviert. Der RemoteTouchUpdate Event feuert, sobald der Spieler den Bildschirm berührt, der UpdateWaypoints-Event ist nötig um beim Erreichen eines Wegpunktes die Anzeige für den nächsten zu aktualisieren und wird in anderen Funktionen angesprochen. Aus einer Objektliste mit allen wegpunktdefinierenden Objekten wird mit Hilfe einer ID der gerade aktive Wegpunkt in die pinke Objektvariable geladen und dann die Distanz zum Player (“All Players”) berechnet. Der Abstand wird in eine Fließkomma-Variable übertragen. Das Skript wird in Abb. 4.9fortgesetzt.

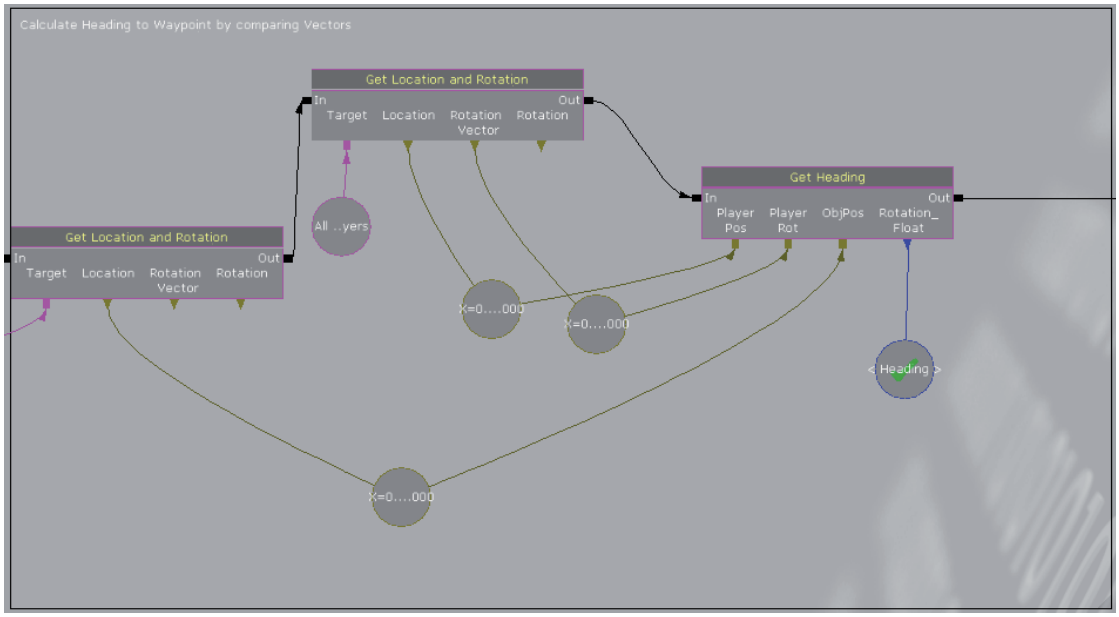

Abb. 4.9: Zunächst wird die Position des Wegpunkts bestimmt (die Objektvariable wurde dafür links aus Abb.4.8 weitergegeben). Dann werden Position und Rotation der Spielfigur (Objektvariable All Players) berechnet und alle Werte in den neuen selbstgeschriebenen Knoten “Get Heading” übergeben, der den Differenzwinkel in Unreal Units in die Variable Heading schreibt. Das Skript gibt seinen Impuls weiter nach rechts zum rendern des Pfeils (In Abb. 4.11).

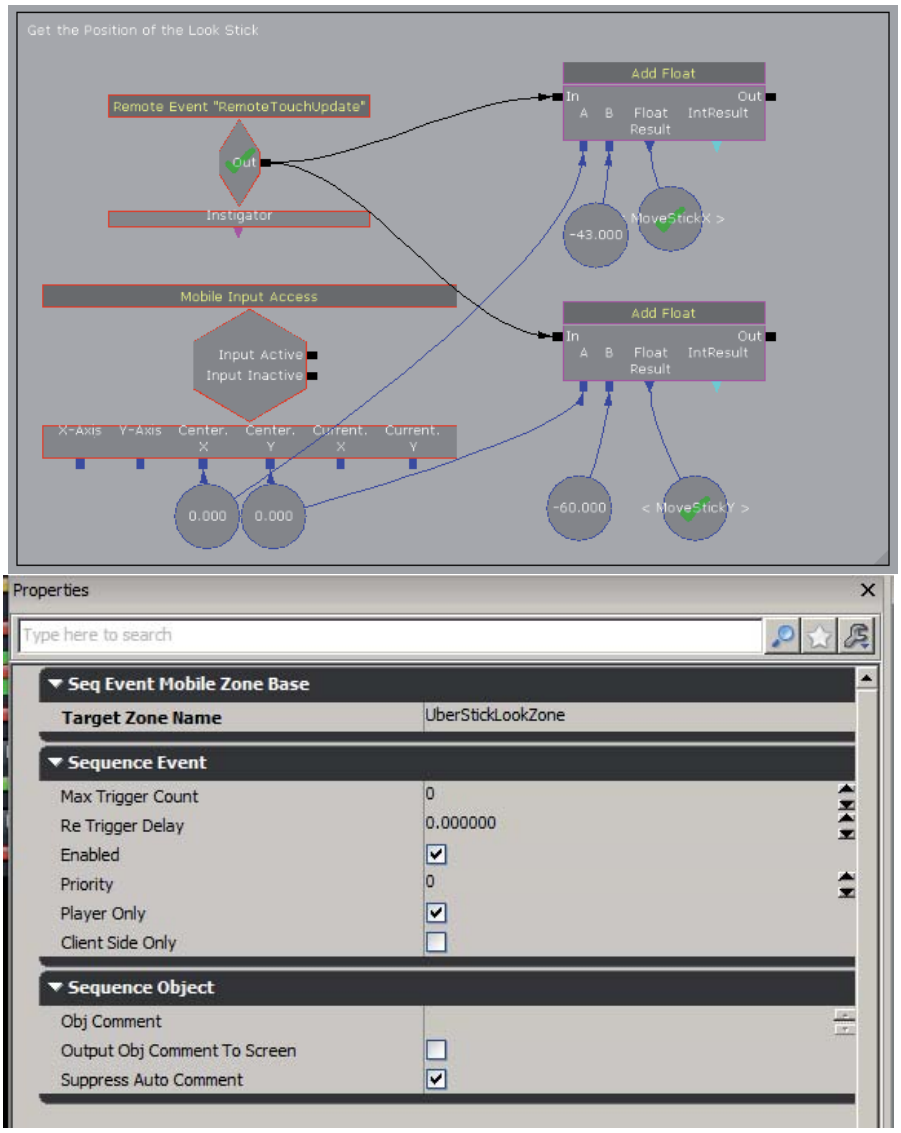

Abb. 4.10: Mittels “Mobile Input Access Event” wird auf die bereits vorhandene Input Zone “UberStickLookZone” (In den Properties zur Node definierbar, siehe rechtes Bild) zugegriffen und deren aktuelle Position gelesen. Die ausgegebenen Werte “CenterX” und “CenterY” geben das Zentrum der Input-Zone an und müssen umgerechnet werden, da die Linke obere Ecke des Sticks als Position gebraucht wird. Mittels “Add Float” wird die entsprechend halbe Bildgröße der Stickgrafik subtrahiert und die Koordinaten anschließend in die “Variable MoveStickX” bzw “Variable MoveStickY” geschrieben. Der Remote Event “RemoteTouchUpdate” aktiviert hierbei die kontinuierliche Berechnung der Werte. Der “Mobile Input Access” Event ist hier reiner Wertelieferant und aktiviert selbst keine weiteren Nodes.

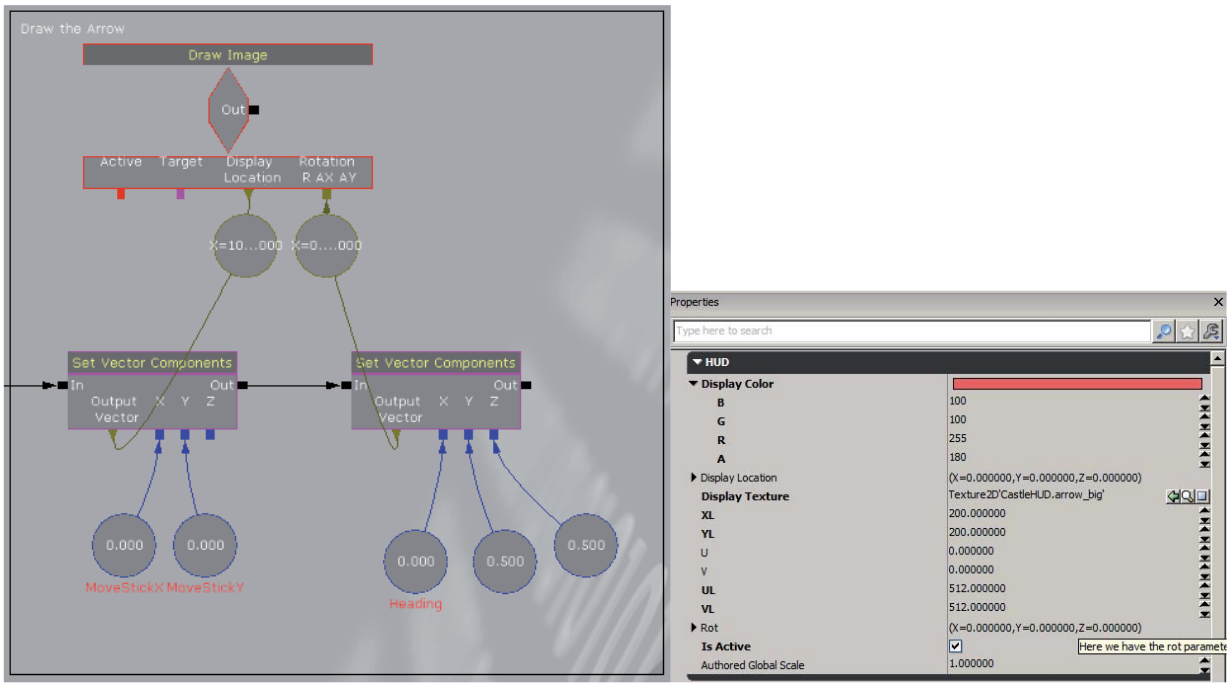

Abb. 4.11: Aus allen durch die vorigen Programmteilen errechneten Werte werden Vektoren gebildet, die für das Rendering des Pfeiles notwendig sind. Die Draw Image Node wurde modifiziert, damit sie einen externen Rotationsparameter akzeptiert. Der Vektor am Eingang “Rotation R AX AY” beinhaltet in seiner X-Koordinate den Drehwinkel in Unreal Units und in seiner Y und Z-Koordinate einen prozentualen Wert für den Drehpunkt um den die zu rendernde Textur gedreht werden soll - im Falle 0.5 um den Mittelpunkt der Grafik. “Heading” stammt hierbei aus dem direkt vorgeschalteten Programmteil aus Abb. 4.9. “DisplayLocation” gibt die Position in der Benutzeroberfläche für die zu rendernde Grafik an. MoveStickX und MoveStickY wurden in einem Extra-Programmelement (siehe Abb. 4.10) vorberechnet. In den Properties zur DrawImage Node ist zudem eingestellt welche Textur aus der Engine-Bibliothek gerendert werden soll (“Display Texture”). Die weiße Pfeiltextur wird mit dem Parameter “Display Color” auf rot umgefärbt und leicht durchsichtig gestellt.

Listing 4.3: Part 1/2 der modifizierten Node “Render Image” mit Erweiterungen zum Rendern von rotierten Texturen auf die Nutzeroberfläche. Dem Script wurde die neue Variable “vector rot” hinzugefügt, die die neuen Parameter für die Bildrotation halten soll. Des weiteren wird eine lokale Drehmatrix-Variable angelegt, da dies von der Canvas-Klasse erwartet wird. Die if Abfragen dienen dem Umgehen des in Kapitel 4 eingangs erklärten Bugs im UDK June 2011 - Part2: Listing 4.4

```
1  //  * Copyright 1998-2011 Epic Games, Inc. All Rights Reserved.
2  class SeqEvent_HudRenderImage extends SeqEvent_HudRender;
3
4  var(HUD) Color DisplayColor; /** The color to modulate the text by */
5  var(HUD) vector DisplayLocation; /** The Location to display the text */
6  var(HUD) Texture2D DisplayTexture; /** The texture to display */
7  var(HUD) float XL, YL; /** The Size of the image to display */
8  var(HUD) float U,V,UL,VL; /** The UVs */
9  var(HUD) vector rot; // Der neue Vektorparameter für die Rotation
10
11 /**   * Perform the actual rendering  */
12 function Render(Canvas TargetCanvas, Hud TargetHud)
13     {
14     local int VarLinkIdx;
15     local float UsedX, UsedY, UsedXL, UsedYL;
16     local float GlobalScaleX, GlobalScaleY;
17     local Rotator R; //Die hinzugefügte Drehmatrix
18     if (bIsActive)  {   // @todo: Not sure why the PropertyName stuff
           doesn't just hookup and overwrite the properties, so do it
           manually for now
19         for (VarLinkIdx = 0; VarLinkIdx < VariableLinks.length ; ++
               VarLinkIdx)
20         {
21             if (VariableLinks[VarLinkIdx].LinkDesc == "Display Location")
22                 {
23                 if (VariableLinks[VarLinkIdx].LinkedVariables.length > 0
                       && VariableLinks[VarLinkIdx].LinkedVariables[0] !=
                       none)
24                     {
25                     DisplayLocation = SeqVar_Vector(VariableLinks[
                           VarLinkIdx].LinkedVariables[0]).VectValue;
26                     rot = SeqVar_Vector(VariableLinks[VarLinkIdx].
                           LinkedVariables[1]).VectValue;
27                     }
28                 }
29             if (VariableLinks[VarLinkIdx].LinkDesc == "Rotation R AX AY")
                   //Neue Abfrage zum Lesen des neuen Vektorinputs
30                 {
31                 if (VariableLinks[VarLinkIdx].LinkedVariables.length > 0
                       && VariableLinks[VarLinkIdx].LinkedVariables[0] !=
                       none)
32                     {
33                     rot = SeqVar_Vector(VariableLinks[VarLinkIdx].
                           LinkedVariables[0]).VectValue;
34                     }
35                 }
36         }
```

Listing 4.4: Part 2/2 der modifizierten Node “Render Image” mit Erweiterungen zum Rendern von rotierten Texturen auf die Nutzeroberfläche. Die Drehmatrix wird mit einem Yaw Drehwinkel in Unreal Units befüllt. Der Drehpunkt wird gelesen und der Orginalaufruf der Canvas Klasse durch einen anderen ersetzt, der Rotationen zulässt. In die defaultproperties wurde der neue Anschluss an die Node angefügt.

```
1
2       TargetCanvas.SetDrawColor(DisplayColor.R, DisplayColor.G,
            DisplayColor.B, DisplayColor.A);
3       // cache the global scales
4       GlobalScaleX = class'MobileMenuScene'.static.GetGlobalScaleX() /
            AuthoredGlobalScale;
5       GlobalScaleY = class'MobileMenuScene'.static.GetGlobalScaleY() /
            AuthoredGlobalScale;
6       // for floating point values, just multiply it by the canvas size
7       // otherwise, apply GlobalScaleFactor, while undoing the scale factor
             they author at
8       UsedX = (DisplayLocation.X < 1.0f) ? DisplayLocation.X * TargetCanvas
            .SizeX : (DisplayLocation.X * GlobalScaleX);
9       UsedY = (DisplayLocation.Y < 1.0f) ? DisplayLocation.Y * TargetCanvas
            .SizeY : (DisplayLocation.Y * GlobalScaleY);
10      UsedXL = (XL <= 1.0f) ? XL * TargetCanvas.SizeX : (XL * GlobalScaleX)
            ;
11      UsedYL = (YL <= 1.0f) ? YL * TargetCanvas.SizeX : (YL * GlobalScaleY)
            ;
12      TargetCanvas.SetPos(UsedX, UsedY);
13
14      //Die neue Drehmatrix wird befüllt
15      R.Pitch = 0;
16      R.Yaw = rot.X;
17      R.Roll = 0;
18      //Der Orginalaufruf von Canvas wird auskommentiert
19      //TargetCanvas.DrawTile(DisplayTexture, UsedXL, UsedYL, U, V, UL, VL,
             DisplayColor);
20      TargetCanvas.DrawRotatedTile(DisplayTexture, R, UsedXL, UsedYL, U, V,
             UL, VL, rot.Y, rot.Z); //Der neue Canvas Aufruf mit Rotation rot.
            Y und rot.Z sind der Drehpunkt, R die Drehmatrix
21  }
22  }
23  defaultproperties {
24      ObjName="Draw Image"
25      ObjCategory="HUD"
26      VariableLinks(2)=(ExpectedType=class'SeqVar_Vector',LinkDesc="Display
             Location",bWriteable=true,PropertyName=DisplayLocation,MaxVars=1)
                 VariableLinks(3)=(ExpectedType=class'SeqVar_Vector',LinkDesc=
            "Rotation R AX AY",PropertyName=rot) //Der neue Anschlusspunkt für
             den Variablenvektor
27  }
```

Abb. 4.12: Screenshot des Navigationssystems Version 0 - An der rechten Seite der Stick mit dem Zielpfeil, in der Mitte der Text mit dem Abstand zum Wegpunkt und dessen Name

4.2.3 Interaktionsmarker

Als Interaktionsmarker wird die Hervorhebung einzelner Gegenstände und Bereiche in der Spielwelt bezeichnet, mit denen der Nutzer interagieren kann. Dieses wird mit Hilfe kleiner Markierungskreise verwirklicht. Hierzu sind einige Regeln zu beachten, denen dieses System unterworfen werden muss. Die Markierungen sollen erst dann auftauchen, wenn der Nutzer auch interagieren kann. Die Kreise müssen sich mit dem Bewegen der Spielfigur auf der Benutzeroberfläche projiziert mitbewegen und so die Position optisch auf dem Gegenstand halten, damit zu jeder Zeit klar ist um welchen Gegenstand es sich handelt. Desweiteren müssen die Kreise sich automatisch beim Verlassen der Interaktionsreichweite oder dem Wegdrehen der Kamera selber ausblenden.

Auch für diese Funktion wäre eine weitere neue Kismet-Node erforderlich - die 3D-Koordinaten eines Objekts müssen auf den Bildschirm rückprojeziert - d.h. wieder in 2D-Bildschirmkoordinaten umgerechnet werden, die abhängig von Kameraposition sowie relativer Position zwischen Spielfigur und Objekt sind. Das UDK bietet hierzu in der Canvas-Klasse die Funktionen Project und Deproject an. Canvas.Deproject ist dafür nötig um die Position in der Welt von einem Klick oder auch Touch auf die 2D-Bildschirmebene abzuleiten, Project für das genaue Gegenteil. Die Project-Klasse müsste also in eine Kismet-Node implementiert werden um sie nutzen zu können. Es bietet sich allerdings an, die bereits aus dem vorigen Abschnitt bekannte “Draw-Image”-Node so weiter zu modifizieren, dass sie diese Funktionaliät zusätzlich besitzt. Das Script ist bis auf einige zusätzliche Aufrufe identisch zu den Listings 4.3 und 4.4. Die zusätzlichen Modifikationen sind im Listing4.5 zu finden. In Prinzip wurde das Script um zusätzliche Eingänge für einen 3D-Vektor erweitert, der die Position des Gegenstands angibt, über den das Overlay gerendert werden soll sowie einen Offset um das Rendering nochmals in X und Y Richtung davon abweichend schieben zu können (basierend auf der Tatsache, dass Canvas immer die linke obere Ecke als Koordinate einer Textur annimmt, und das Overlay aber zentriert

dargestellt werden soll. Demnach ist der Offset die halbe Bildgröße in X und Y). Zuletzt wurde ein bool(Ja / Nein)-Parameter integriert mit dem das Rendering aus Kismet heraus aktiviert und deaktiviert werden kann.

Weiterhin ist es nötig ein System zu implementieren das prüft, ob sich ein Gegenstand im Blickfeld des Nutzers befindet oder nicht, denn nur dann soll ein Marker gerendert werden. In der Unreal Engine existiert ein Actor, der genau diese fehlende Funktionalität bietet. Ein spezieller Trigger - der “Line of Sight-Trigger” - aktiviert einen zugehörigen Kismet-Event,wenn es sich im Sichtfeld des Nutzers und in einer definierbaren Reichweite befindet. Eines dieser Trigger-Volumen wird mit jedem Objekt kombiniert, dass “interaktiv” ist. Sobald der Marker dargestellt wird, werden die Positionen des Triggers ebenfalls an die Position des Gegenstands gekoppelt, damit das Triggervolumen im Falle einer Interaktion des Nutzers mit dem Gegenstand, die dessen Position ändern könnte ebenfalls seine Position mit verändert. Dies verhindert, dass der Marker die vorherige Position hält, während z.B. ein weggestoßenes Fass längst an einem anderen Ort zum Liegen gekommen ist. (siehe Abb. 4.13) Im Rahmen eines fertigen Programms wäre es mitunter vorteilhaft eine eigene Actor-Klasse zu definieren, die automatisch die Funktionalität des Trigger-Volumens integriert hat. Das würde bei großen Mengen an interagierbaren Gegenständen den Aufwand stark reduzieren, würde aber den Rahmen des hier vorgestellten Prototypen bei weitem sprengen, da man tief in den Code der Engine eingreifen müsste um eine komplett eigene Actor-Klasse zu definieren. Ein Screenshot des Interaktionsmarkers in der Testanwendung findet sich in Abb. 4.14.

4.2.4 Interaktionsoverlay mit Aktionsbuttons

Die Interaktionsbuttons sind beim Berühren des Objekts mittels Touchscreen auftauchende Buttons über dem Finger des Nutzers, die dazu dienen eine Aktion auszuwählen die mit dem Gegenstand durchgeführt werden soll. Der Nutzer begibt sich beim Berühren eines interaktionsfähigen Gegenstands sozusagen in einen Interaktionsmodus. Die Buttons sollen ihre Position - ähnlich dem Interaktionsmarker - ständig in Echtzeit an die Position des zugehörigen Gegenstands anpassen, solange sie sichtbar sind. Bewegt sich der Nutzer außerhalb der Interaktionsreichweite oder dreht sich weg, soll der “Interaktionsmodus” automatisch beendet werden. Desweiteren wird am rechten Rand des Bildschirms ein Abbrechen-Button integriert, der den Interaktionsmodus auch manuell abbrechbar macht. Wählt der Nutzer eine Aktion aus, so wird diese ausgeführt und der Interaktionsmodus beendet.

Touchscreen-fähige Buttons - eine Unterart der sogenannten Input-Zones (zu denen auch z.B. die emulierten Gamepad-Sticks gehören) können im UDK mittels eigens für diesen Einsatz vorhandener Kismet-Nodes oder statisch in einer Konfigurationsdatei erzeugt

Listing 4.5: Das Unreal Script der “Draw-Image”-Node wird erneut modifiziert um eine zusätzliche Variante zur Verfügung stellen, die in Abhängigkeit einer 3D-Position im Raum passend projeziert auf die Bildschirmebene eine Markierung genau dort rendert, wo besagte Position dargestellt wird. Bereits in den vorigen Listings 4.3 und 4.4 vorhandener Code wird durch kommentierte Auslassungen (...) ersetzt, um mehr Übersicht für die modifizierten Bereiche zu schaffen.

```
1  class SeqEvent_HudRenderImageActor extends SeqEvent_HudRender;
2
3  ... //Variablen-Definitionen
4
5  var(HUD) vector ActorLocation; //zusätzliche Variable für die Position
       des Zielgegenstands in der 3D Welt
6  var(HUD) vector DisplayOffset; //Der Offset um den das Rendering
       geschoben wird
7  var() int DrawActive; //Der zusätzliche Parameter der angibt ob gerendert
        werden soll, int und bool sind für UDK prinzipiell das Gleiche
8
9  function Render(Canvas TargetCanvas, Hud TargetHud)
10     {
11     ... //Aufrufe zum Abfangen des Variable-Auslesen-Bugs und lokale
           Variablendefinitionen
12
13     if (DrawActive > 0) { //Nur rechnen und rendern wenn der Parameter
           nicht 0 ist
14
15         ... //Canvas-Aufrufe für Farbe und Scaling
16
17         DisplayLocation = TargetCanvas.Project(ActorLocation); //2D
               Projektion des 3D Vektors auf die Bildschirmebene
18         DisplayLocation.X = DisplayLocation.X + DisplayOffset.X;//
               Addition des Offsets
19         DisplayLocation.Y = DisplayLocation.Y + DisplayOffset.Y;
20
21         ... //Canvas-Aufrufe für Rotation, Scaling und Rendering
22         }
23     }
24 defaultproperties {
25     ObjName="Draw Image Projected from 3D"
26     ObjCategory="HUD"
27     VariableLinks(0)=(ExpectedType=class'SeqVar_Bool',LinkDesc="BActive",
           PropertyName=DrawActive) //neu, bool Schalter zum aktivieren oder
           deaktivieren des Renderings
28     VariableLinks(1)=(ExpectedType=class'SeqVar_Vector',LinkDesc="Actor
           Location",bWriteable=true,PropertyName=ActorLocation,MaxVars=1) //
           neu, 3D Position die projeziert werden soll
29     VariableLinks(2)=(ExpectedType=class'SeqVar_Vector',LinkDesc="
           Rotation R AX AY",PropertyName=rot)
30     VariableLinks(3)=(ExpectedType=class'SeqVar_Vector',LinkDesc="Screen
           Coord",bWriteable=true,PropertyName=DisplayLocation) //Ausgabe der
            Koordinaten (zusätzlich zum Rendern)
31 74  VariableLinks(4)=(ExpectedType=class'SeqVar_Vector',LinkDesc="Offset"
           ,bWriteable=true,PropertyName=DisplayOffset) //neu, Offset für das
            Rendering
32 }
```

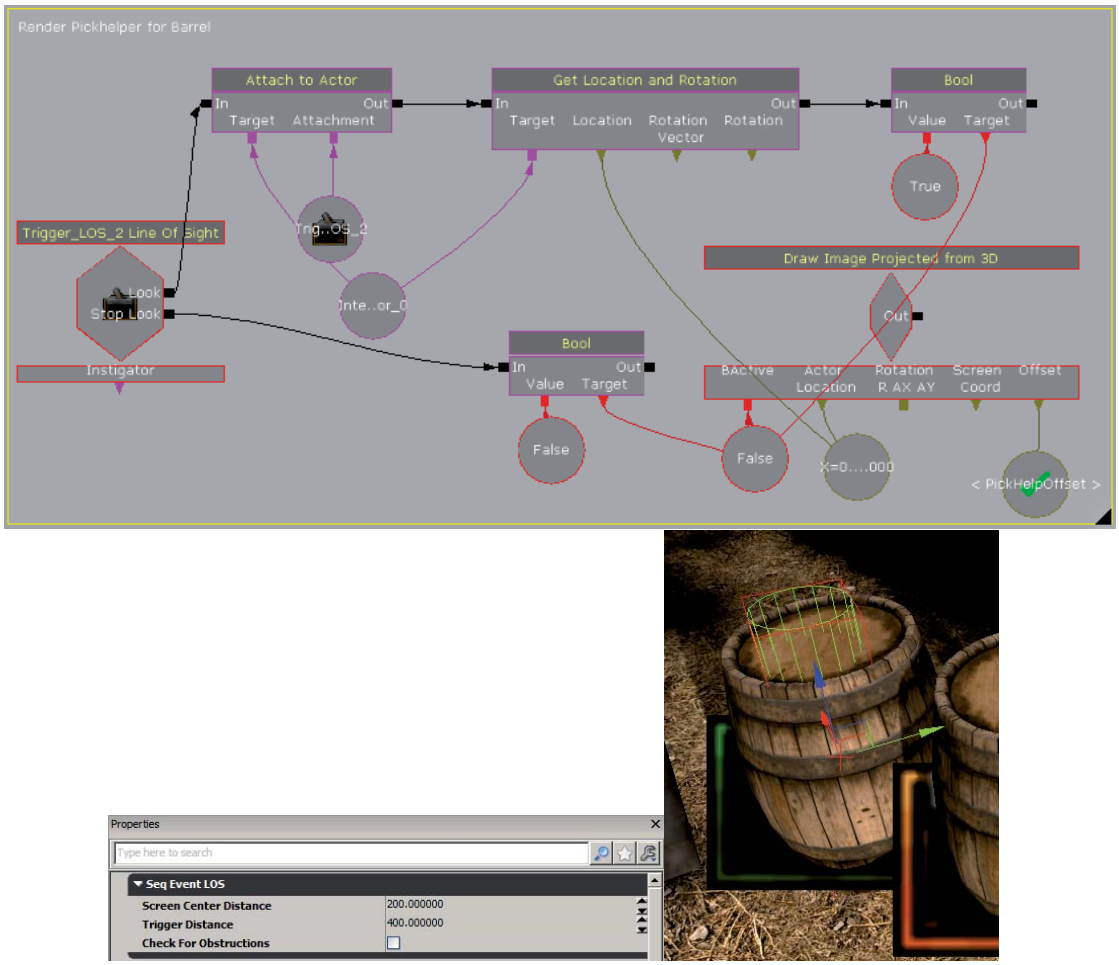

Abb. 4.13: Das Kismet System zum Rendern des Interaktionsmarkers. Jedem interaktionsfähigen Gegenstand wird ein Trigger zugeordnet (unten rechts, grüner Zylinder). Der zugehörige Event “Trigger_LOS_2 Line of Sight” wird so konfiguriert, dass er ab einem Abstand von 400 Units am Look-Ausgang feuert (Trigger Distance). Screen Center Distance ist quasi der Abstand vom Zentrum des Spielerblickfelds, indem der Event ausgelöst wird und limitiert damit, wie genau der Spieler auf den Gegenstand zielen muss um den Event auszulösen. Wird der “Look” Ausgang des Events ausgelöst wird die Position des Triggers an den zugehörigen Gegenstand gekoppelt, damit er sich im Falle einer Bewegung des Gegenstands mitbewegt. Die Position des Gegenstands wird gelesen und das Rendering des Overlays durch die modifizierte Draw Image Node (siehe Listing 4.5) mittels einer Boolvariable aktiviert. Dreht sich der Spieler weg, feuert der “Stop Look” Ausgang am Event und das Rendering wird mittels Wechsel der Boolvariable auf “False” beendet.

Abb. 4.14: Ein Screenshot des Interaktionsmarkers in der Spielwelt.

werden. Dennoch sind auch in Kismet generierte Buttons und Input-Zones prinzipiell in der Art statisch, dass ihre Position, Größe und Gestalt fest definiert sein muss und sich nicht mehr nach dem Erzeugen ändern lässt. Sie können lediglich zur Laufzeit beliebig aktiviert und deaktiviert werden. Das macht die mitgelieferte Funktionalität natürlich für die Buttons dieses Projekts, die prinzipiell ständig bewegbar bleiben müssen, absolut ungeeignet. Erschwerend kommt hinzu, dass es sich bei der Erzeugung dieser Input-Zones um sogenannte Native-Classes handelt, die nicht über Unreal Script modifiziert werden können, weswegen es auch aussichtslos ist zu versuchen eine alternative Kismet-Node zu schreiben, die die entsprechende Funktionalität hinzufügt. Ein derart tiefer Zugriff auf die Native-Klassen ist nur einem begrenzten registrierten Nutzerkreis des UDK möglich und in der frei verfügbaren Fassung, die im Internet bezogen werden kann, unmöglich. Allerdings ist auch dieses Problem umgehbar, da es wiederum einen Event gibt, der rohe Touch-Positionsdaten ausgeben kann.

Dieser “Raw Input Access”-Event (Bereits vorgestellt, siehe Abb. 4.7), der auch schon bereits für die anderen neu implementierten Funktionen der letzten Abschnitte eine Rolle spielte kann dazu benutzt werden X und Y-Koordinaten von Touch-Ereignissen zu erhalten, die wiederum in einer eigenen Logik mit entsprechenden Bedingungen (siehe Abb. bis 4.16) genutzt werden können um die Buttons zu emulieren. Zunächst wird der angeklickte Gegenstand isoliert (siehe Abb. 4.15). Mittels einer Reihe von Bedingungen wird anschließends der Touch interpretiert. So kann mit der Texturgröße des Button-Overlays, der Position an der die Textur über das Objekt gerendert wird und den Koordinaten des Touchs errechnet werden ob und welchen Teil der Grafik (= welchen Button) der Nutzer gedrückt hat. Da das Canvas-Texturrendering durchaus aktiv zur Laufzeit beeinflusst und animiert werden kann, ist es so möglich die nicht brauchbaren Input-Zones für diesen Fall zu ersetzen. Wurde ein entsprechender Klick auf einen der Buttons ausgewertet kann unterschieden werden, um welche Aktion es sich handelte(siehe Abb. 4.16). Im Falle des rechten Buttons werden dem Nutzer textbasierte Informationen auf dem Bildschirm zum Gegenstand ausgegeben (siehe Abb.4.17), was als Platzhalter für Informationen beispielsweise in Architekturvisualisierungen dienen kann. Alternativ wird der Gegenstand benutzt, d.h. im Falle des Fasses ein kinetischer Stoß ausgelöst um dieses weg zu schubsen (Abb, 4.18) oder im Falle des Artefakts um es einzusammeln (Abb.4.19) - dies wird anhand einer bereits am Beginn des Scripts gesetzten Variable entschieden, die davon abhängt um welchen Typ von Gegenstand es sich handelt. Das ganze Skript muss nach der Aktivierung ständig in einer Schleife laufen, bis die Interaktion beendet oder abgebrochen wurde, um die Position auf dem Display zu aktualisieren, wenn sich der Spieler bewegt. So kann er z.B. ein Fass aktivieren, dann herum laufen bis er meint in der richtigen Richtung zu stehen und dann erst den Stoß auslösen. Die Buttons halten dabei immer die richtige Position auf dem Bildschirm über dem Objekt. Schlussendlich wird das Overlay mit der bereits mofifizierten Draw Image-Node gerendert (siehe Abb. 4.20).Ein Screenshot des Overlays in der Testanwendung findet sich in Abb. 4.14.

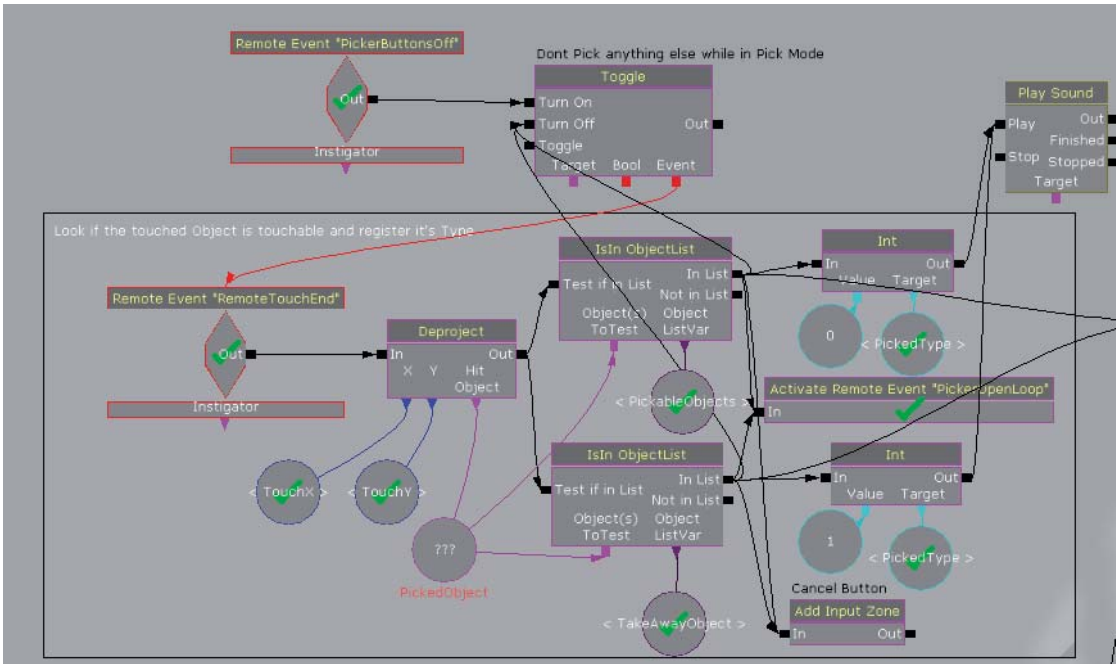

Abb. 4.15: Durch den “Raw-Input Event” (Siehe Abb. 4.7) wurde ein Touch erkannt und per Remote Event wird dieses Kismet-Script aktiviert. Zunächst wird verfolgt ob und welchen Actor in der Spielwelt der Nutzer mittels Touch getroffen hat. Die “Deproject” Node projeziert einen Strahl abhängig von Blickwinkel und Koordinaten des Touches in die Spielwelt und gibt ein Objekt aus, dass getroffen wurde. Danach wird von der Logik geprüft, ob das getroffene Objekt in einer der Listen für interagierbare Gegenstände vorhanden ist. Je nachdem um welchen Typ es sich handelt, wird eine Variable “Picked Type” gesetzt. Gleichzeitig wird der “Abbrechen”-Button mittels “Add Input Zone” hinzugefügt und ein Sound abgespielt, der dem Nutzer verdeutlicht, dass sein Touch erfolgreich war. Mittels “Activate Remote Event PickerOpenLoop” wird eine Bedingung gesetzt, die später ermöglicht das folgende Script im Loop laufen zu lassen. Dies ist notwendig um ständig die Position des Overlays an den Gegenstand anzupassen. Schließlich wird noch der ursprünglich auslösende Remote Event mittels “Toggle” Node deaktiviert, damit keine anderen Gegenstände berührt werden können, so lange sich das Programm im Interaktionsmodus befindet. Diese Toggle-Node kann durch Fernzugriff angeregt werden, das Skriptelement wieder zu aktivieren, wenn die Interaktion mit dem Gegenstand abgebrochen oder beendet wurde. So kann dann ein neuer Gegenstand gewählt werden.

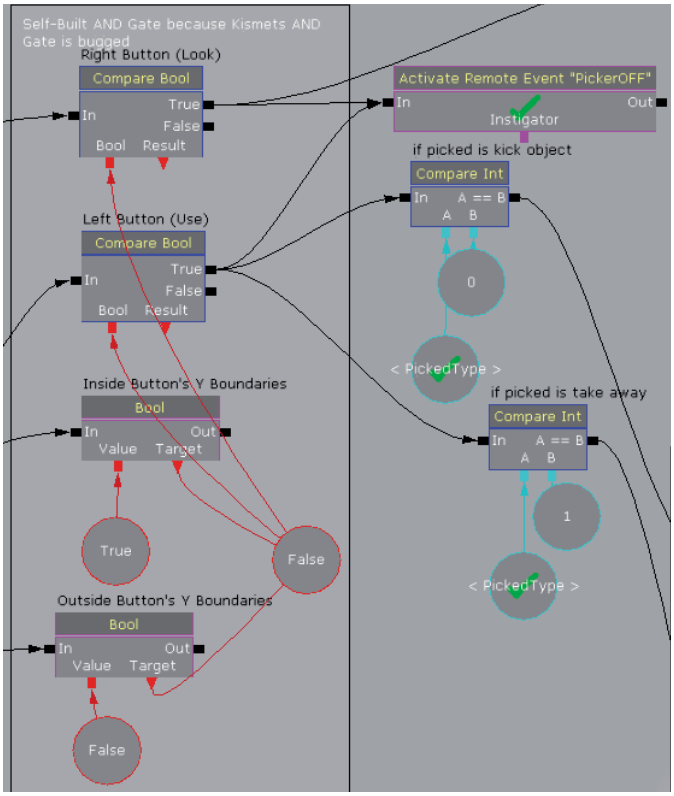

Abb. 4.16: Kismet bietet von Haus aus ein AND Gate an, dass den Impuls dann weiterleitet, wenn zwei ankommende Impulse gleichzeitig aktiv sind. Leider ist selbiges fehlerhaft, da es nur ein einziges mal während der gesamten Laufzeit des Programms feuert und danach sämtliche Impulse ignoriert. Deshalb wird hier mittes einer Bool-Variable eine Verschaltung der erfüllten (oder eben nicht erfüllten) Bedingungen realisiert. Sind diese erfüllt wird entweder (falls der rechte Button ausgewählt wurde) mittels Remote Event das Button Overlay deaktiviert und die Anzeige der Informationen zum Gegenstand eingeleitet (siehe Abb. 4.17) oder falls der linke Button ausgelöst wurde, erneut anhand der am Anfang des Skripts gesetzten Typ-Variable unterschieden um welchen Typ von Gegenstand es sich handelt und dann entweder die Aktion für das Bewegen des Gegenstands (Abb. 4.18) oder das Einsammeln des Artefakts (Abb. 4.19) ausgelöst. Auch in diesem Fall wird mittels Remote Event das Button Overlay deaktiviert.

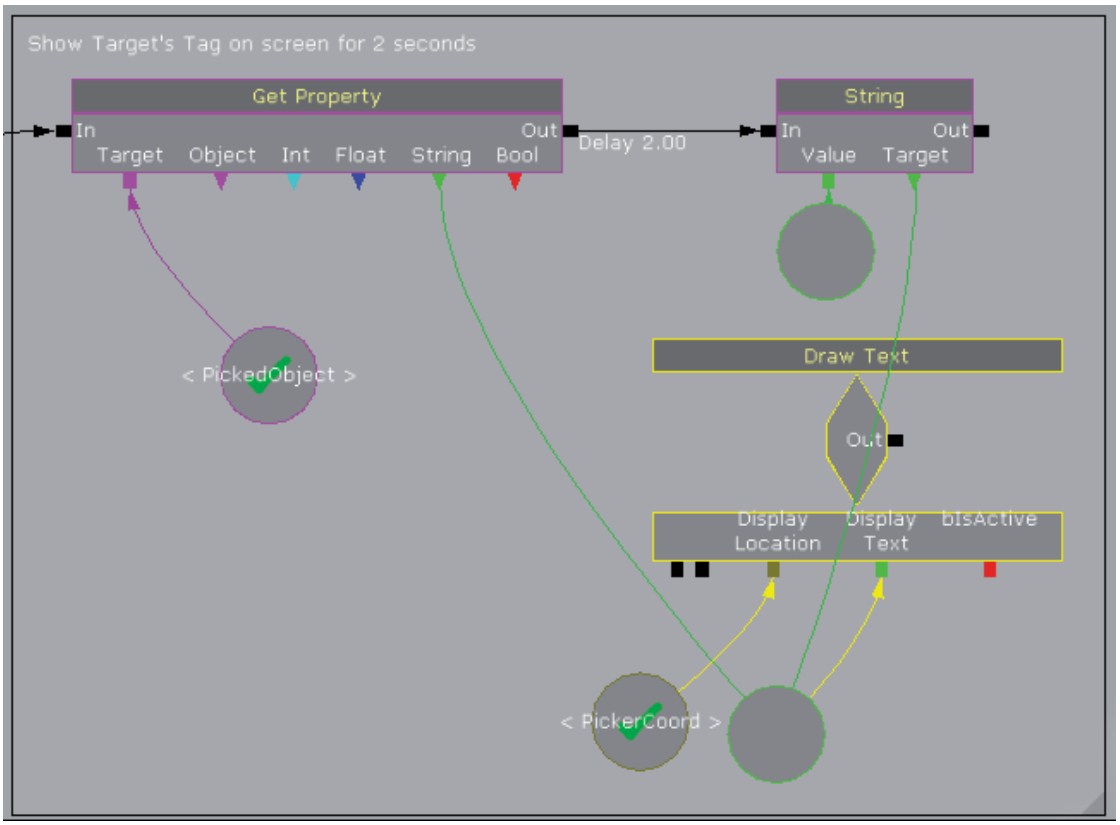

Abb. 4.17: Die Funktion des rechten (Look) Buttons im Interaktionsoverlay. Jeder Actor in UDK hat ein “Tag”, dass vom Leveldesigner vergeben werden kann, um Gegenstände zu sortieren. Das Tag wird hier genutzt um Informationen zum benutzbaren Gegenstand zu hinterlegen. Diese werden mittels “Get Property” gelesen und auf dem Bildschirm ausgegeben. Mit einem Delay von zwei Sekunden wird der Text wieder ausgeblendet, indem die zugehörige String-Variable geleert wird. Der Text wird an Position des mittlerweile ausgeblendeten Button Overlays angezeigt, also genau über dem Gegenstand, der ausgewählt wurde.

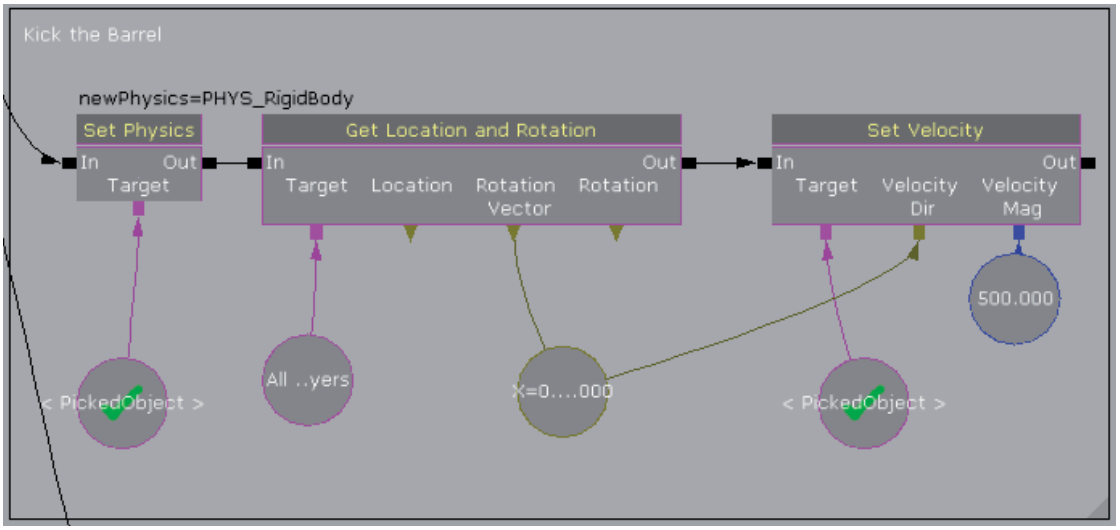

Abb. 4.18: Die Funktion des linken (Use) Buttons im Interaktionsoverlay im Falle eines bewegbaren Gegenstands. Mittels “Set Physics” wird die Physikberechnung für den ausgewählten Gegenstand aktiviert. So kann er mit anderen Gegenständen, Wänden und Levelelementen kollidieren und fällt nicht einfach durch den Boden. Es wird mittels “Get Location and Rotation” der Blickrichtungsvektor des Spielers ermittelt und dann mittels “Set Velocity” eine Beschleunigung auf den Gegenstand angewendet. “Velocity Dir” gibt die Richtung der Kraft an, die hier dem Blickrichtungsvektor des Spielers entspricht, und damit den Gegenstand immer vom Spieler weg stößt.

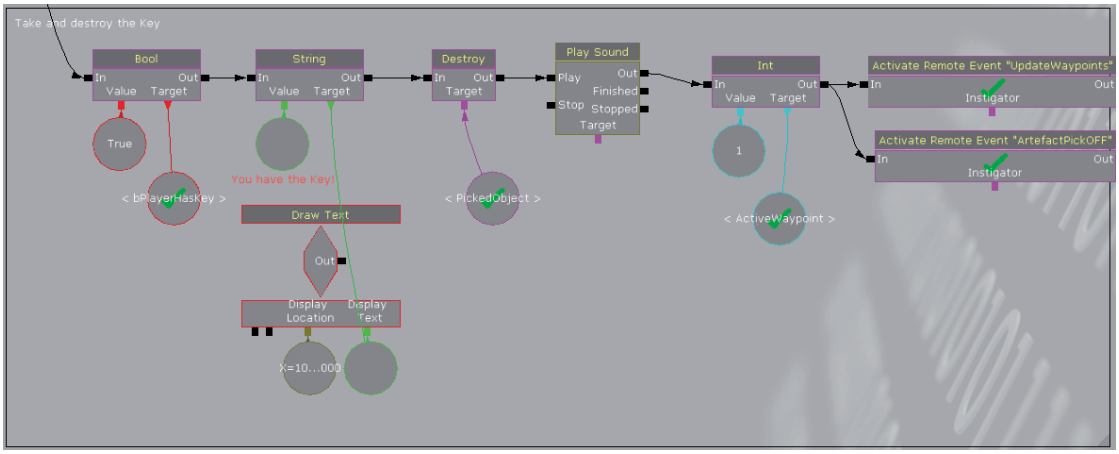

Abb. 4.19: Die Funktion des linken (Use) Buttons im Interaktionsoverlay im Falle des Artefakts. Eine Bool-Variable wird gesetzt, die aussagt, dass der Spieler das Artefakt erfolgreich eingesammelt hat (“bPlayerHasKey”). Anschließend wird ein Text auf dem Bildschirm ausgegeben und mittels “Destroy” der Actor aus der Spielwelt entfernt, schließlich wurde er “eingesammelt”. Ein bestätigender Sound wird mittels “Play Sound” abgespielt und der Aktive Waypoint für den Pfeil auf den nächsten Wegpunkt, die Kirche umgestellt. Schließlich wird mittels Remote Events noch eine manuelle Aktualisierung des Pfeils eingeleitet. Der RemoteEvent “ArtefactPickOFF” deaktiviert das Rendering für den Marker, der das Artefakt markiert hat, da dieser ja nicht auf dem Objekt selber sondern dem zusätzlich hinein gesteckten Trigger-Volume basiert, was nach wie vor existent ist. So wird verhindert, dass ein Marker da gerendert wird, wo das Artefakt einmal war, und aber nichts mehr zum interagieren vorhanden ist.

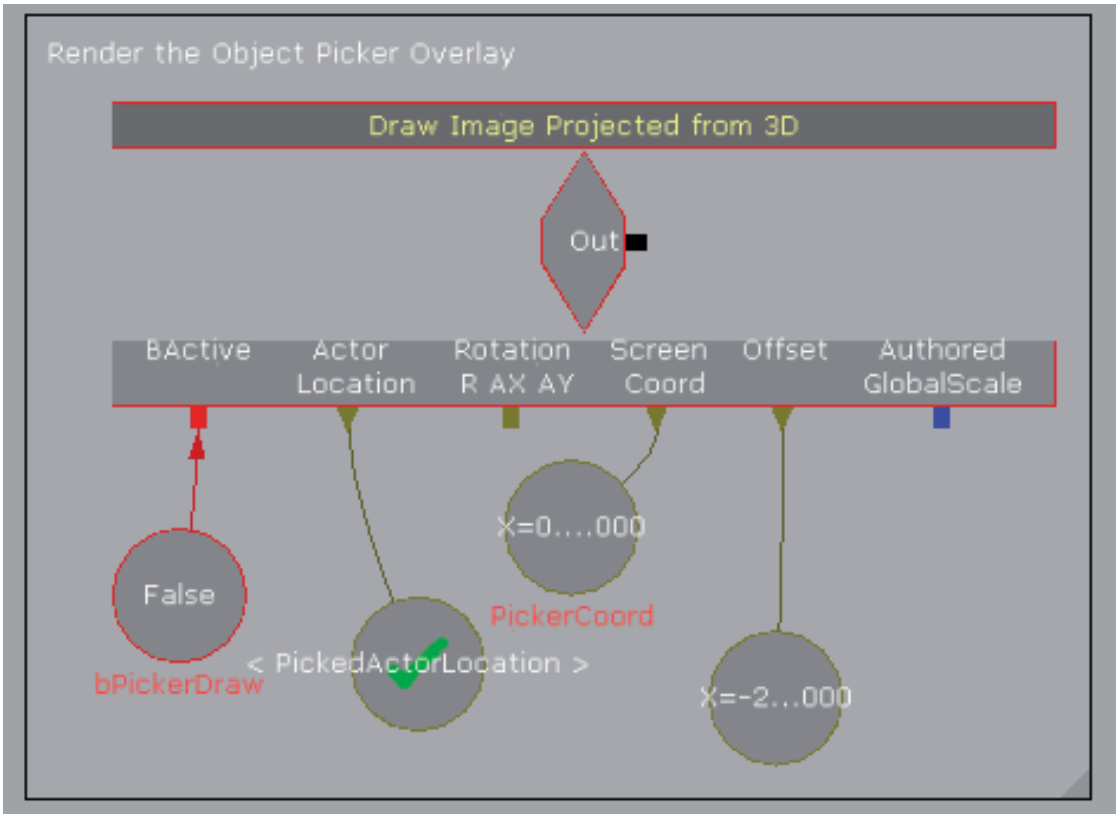

Abb. 4.20: Die Render-Node für die Interaktionsbuttons. Das eigentliche Rendering der Buttons wird hier durchgeführt. Die bereits bekannte modifizierte DrawImage Node aus dem vorigen Abschnitt über den Marker wird auch hier verwendet. Ein Bool-Parameter kontrolliert, ob das Overlay gerendert wird, die Koordinaten werden von Außen ständig mit jedem Schleifendurchgang aktualisiert.

Abb. 4.21: Ein Screenshot des Button Overlays, wie man es in der Testanwendung antreffen kann.

Die Information ob das Artefakt eingesammelt wurde ist indes entscheidend die Aufgabe des Tests abzuschließen. Sie wird in der Levelmechanik vor der Statue gegengeprüft. Dieses letzte fehlende Element muss noch im folgenden implementiert werden.

4.2.5 Vorbereitung des Levels für den Usability Test

Die Aufgabe des Usability-Tests sieht vor, dass die Probanden zunächst von ihrem Startpunkt aus zum Artefakt gehen, es einsammeln und zu einem anderen Ort in der Spielwelt zurück bringen. Zunächst werden für das Wegpunkt-System entsprechende Trigger-Volumes gesetzt, die als Platzhalter für den Zielpunkt dienen sollen. Der erste Trigger, der keinerlei Funktion hat außer die Position zu halten, wurde in einem der Zelte in der Außenanlage platziert. Das Artefakt selbst wurde in das Zelt gesetzt und an einer Wand angebracht. Das Artefakt-Objekt wurde entsprechend den Erfordernissen der Interaktionslogik mit seinem eigenen Trigger-Volume und der zugehörigen Kismet-Schaltung für den Interaktionsmarker versehen und in die Liste der anklickbaren Gegenstände aufgenommen.

Vor den Eingang des Zeltes wurden mehrere Kopien des in den vorigen Abschnitten näher gezeigten umstoßbaren Fasses platziert, die den Nutzer hindern das Artefakt zu erreichen, ohne vorher mit den Fässern zu interagieren und sie damit aus dem Weg zu schubsen. Dies soll sicherstellen, dass der Nutzer ein konkretes Problem mit dem neuen User-Interface lösen muss, um die gesamte Aufgabe beenden zu können. Weitere dieser Fässer wurden zufällig auf dem wahrscheinlichsten Weg des Probanden aufgebaut, um bereits vorher seine Aufmerksamkeit auf die auftauchenden Interaktionsmarker zu lenken

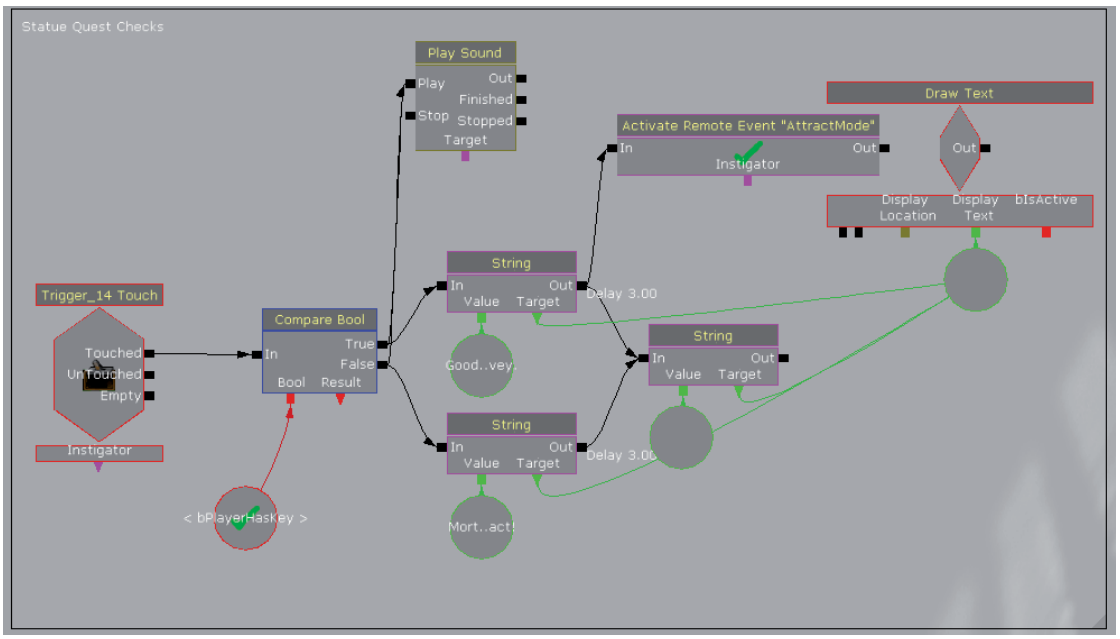

Abb. 4.22: Die Kismet-Logik für den Trigger in der Kirche. Es wird zunächst geprüft ob der Spieler das Artefakt eingesammelt hat, indem die Bool-Variable “bPlayerHasKey” betrachtet wird. Hat er das Artefakt wird ein entsprechender Text eingeblendet, ein Sound abgespielt und die Videosequenz (Remote Event “AttractMode”) aktiviert. Ansonsten wird ein Text eingeblendet der den Spieler darauf hinweist, dass noch etwas fehlt.

und ihn eventuell dazu zu bewegen diese vorher schon zu testen bevor er bei den Zelten dazu gezwungen ist.

In der Kirche, dem Zielort für das Artefakt, den der Proband betreten soll nachdem er das Artefakt eingesammelt hat, wird ein weiteres Trigger-Volume plaziert, dass als Wegpunkt für den Navigationspfeil dient. Dieses wird aber zusätzlich mit einem eigenen Event versehen: Sofern das Artefakt gefunden wurde (die entsprechende Variable intern gesetzt wurde) wird der Test mit einer Botschaft beendet. Betritt der Proband das Volumen ohne dass die Variable gesetzt wurde, bekommt er den Hinweis, dass ihm noch etwas fehlt (siehe Abb. 4.22). Das Triggervolumen wurde entsprechend großzügig um die Statue im Hauptraum der Kirche herum dimensioniert, damit es auch sicher bei Betreten auslöst (siehe Abb. 4.23). Es stellt selber den zweiten Wegpunkt dar, der aktiv geschaltet wird, sobald das Artefakt bei den Zelten gefunden wurde. Betritt der Spieler schließlich mit dem Artefakt den Bereich vor der Statue, endet der Usability-Test - das Programm schaltet in eine Videosequenz.

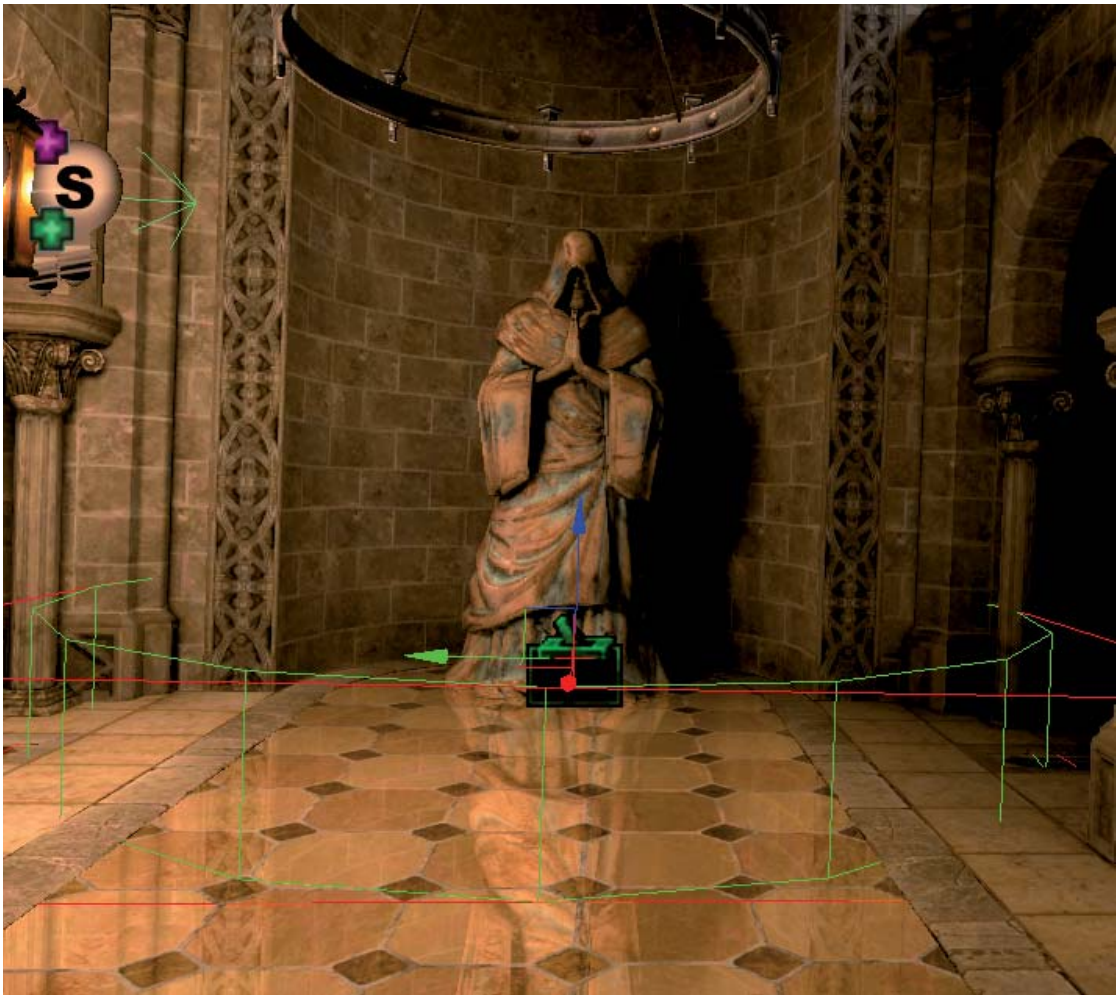

Abb. 4.23: Der Endabgabebereich im Unreal Editor. Gut zu erkennen ist das grüne Trigger-Volumen, das auslöst sobald der Spieler selbiges betritt und damit das Kismet Script in Abb. 4.22startet.

4.3 Evaluationsphase

Der Usability-Test wurde mit der erstellten Applikation auf einem iPad mit fünf zufällig gewählten Probanden aus dem Bereich der Medieninformatik durchgeführt. Drei dieser fünf Probanden gaben im Test an, noch keine Erfahrungen mit Tablets sowie Touchscreen-basierten Spielen gemacht zu haben. Alle fünf haben, nach mehr oder weniger langer Zeit, den Test erfolgreich beenden können, die einzelnen Tests wurden gefilmt und jeder Proband hat den Fragebogen bearbeitet. Im folgenden werden die Fragebögen und Videos ausgewertet und Schlüsse für die nächste Entwicklungsstufe, Iteration 1 daraus gezogen.

4.3.1 Auswertung des Fragebogens

Zunächst die Auswetung des Fragebogens nach den einzelnen Fragenkategorien. Eine Tabelle mit Auswertungsdaten findet sich im Angang A.3.

Bereich: Aufgabenangemessenheit

Im Bereich der Aufgabenangemessenheit ist auffällig, dass alle Probanden der Meinung waren, der für die Erledigung der Aufgabe nötige Aufwand sei der Aufgabe angemessen gewesen, ein erfreuliches Ergebnis. Durchwachsen war die Resonanz auf die vorhandenen Funktionen: Ein Teil der Probanden wünschte sich zusätzliche Funktionen, allen voran eine Funktion zum Springen, um der generell etwas umständlich empfundenen Interaktion mit den Fässern aus dem Weg zu gehen. Auch wünschten sich Probanden die Einführung eines Knopfes, mit dem sie die Spielfigur schneller laufen lassen können, um die Laufwege zu verkürzen. Positiv wurde die Platzierung der Eingabeelemente aufgenommen, offenbar hat der Versuch diese so zu platzieren, dass sie immer oberhalb des Fingers auftauchen, Früchte getragen. Lediglich eine Person kritisierte, dass der Zielpfeil vorübergehend verdeckt war. Auffällig ist, dass die unerfahrenen Probanden generell Probleme mit den Bewegungssticks hatten - einem war nicht klar, dass der Stick sich zu seinem Finger hin schiebt, und er ihn nicht präzise treffen muss. So wurde auch eine Funktion gewünscht, die die Bewegung der Spielfigur automatisiert wenn ein Zielort angeklickt wird. Der betreffende Proband gab an die Demo, die als Grundlage für diesen Prototyp zu kennen (Epic Citadel) und diese in der Urversion vorhandene Funktion zu vermissen. Durchwachsen ist die Resonanz auf die Frage nach dem Vergleich zu anderen Spielen - die Probanden mit Vorerfahrung erkannten Elemente wieder, die Gruppe der nicht-erfahrenen Probanden empfand die Steuerung als ungewohnt.

Bereich: Selbstbeschreibungsfähigkeit

Im Bereich Selbstbeschreibungsfähigkeit wurde immer wieder die Unklarheit der Icons bemängelt. Es war vielen Nutzern (durch alle Professionalitätsschichten hindurch) nicht klar, welche Interaktion durch welches Piktogramm symbolisiert wurde, ohne es auszuprobieren - hier ist definitiv Verbesserungspotential. Explizit wurden Texte gewünscht, die die Buttons genauer beschreiben. Überwiegend positiv ist die Resonanz auf die Frage danach, ob jederzeit klar ist was als nächstes zu tun ist. Dies ist sicherlich dem Navigationspfeil geschuldet, der recht eindeutig aussagt wo etwas zu tun ist. Überhaupt liegen hauptsächliche Beschwerden im Bereich des Interaktions-Overlays, also der Buttons zum interagieren mit den Fässern und deren Handhabe. Sehr positiv ist, dass alle durchweg der Meinung waren, dass das Programm im richtigen Moment richtige Funktionen anbietet. Dies ist ein klarer Sieg für kontextsensitive Funktionen und Overlays - lediglich ein Proband gab trotz positiver Beantwortung der Frage ein relativierendes “manchmal” mit auf den Weg.

Bereich: Steuerbarkeit

Wiederum wird auch im Bereich Steuerbarkeit der Umgang mit den Fässern bemängelt - die Funktion mache nicht unbedingt das, was von den Probanden erwartet wurde. Generell positiv ist, dass die Probanden sich nicht durch die Funktionen des Programms in ihrer “Arbeitsweise” beeinflusst oder unterbrochen fühlten.

Bereich: Erwartungskonformität

Im Bereich Erwartungskonformität wird wieder die Unklarheit des “Benutzen” Buttons deklariert. Ein nicht-erfahrener Proband empfand es als sehr störend, dass er nicht ohne Ausprobieren wußte, was welche Funktion der Steuerung tat - er würde sich Hilfetexte oder ein explizites Tutorial wünschen. Wiederum wurde nach einer Springen-Funktion gefragt. In den Kommentaren finden sich, davon abgesehen, viele Rechtfertigungen der nicht-erfahrenen Probanden, dass sie noch nie mit so etwas gearbeitet hätten und sie deswegen nicht sofort wußten wie es funktionieren würde. Interessant ist die Frage nach der Bewegungssteuerung. Es wurde ausrcüklich gefragt, ob die Probanden die Reaktionen des Programms auf Bewegungen als erwartungsgemäß empfanden. Als Vorgriff auf die Videoaufnahmen ist zu betonen, dass nicht eine Person direkt versucht hat mit einer Bewegung des Tablets eine Aktion auszulösen. Zwei der fünf Probanden haben nicht einmal versucht das Tablet in die Hand zu nehmen. Dennoch ist nur ein Proband der Meinung, er hätte etwas anderes erwartet. Es handelte sich um einen nicht-erfahrenen Probanden und er kommentierte die Frage mit “keine Bewegung fest gestellt”. Zwei Probanden sagen, es sei erwartungsgemäß nichts passiert, zwei hielten die Frage für nicht-zutreffend. Ein deutlicher

Wunsch oder eine Bemängelung die Integration der Bewegungssensoren entsprechend wurde in keinem der Fälle ausgesprochen. Ebenfalls geben einige der Probanden an, keinen Versuch unternommen zu haben bekannte Touchgesten wie Zoomen mit zwei Fingern getestet zu haben.

Bereich: Fehlertoleranz

Bis auf eine Person, die ein Ruckeln beanstandet hat, wurde der Fehlertoleranz-Bereich weitgehend positiv oder mit "nicht zutreffend" beantwortet. Eine Person bemängelte, dass die Funktion zum Bewegen der Fässer unzuverlässig arbeitete.

Bereich: Individualisierbarkeit

Die Frage nach der Anpassungsfähigkeit der Steuerung wurde beachtlich unterschiedlich bewertet, wenn man davon ausgeht, dass der aktuelle Prototyp keiner derartige Funktion enthielt. Eine Person beantwortete die Frage sogar mit Ja, die Steuerung sei anpassbar. Ein nicht-erfahrener Proband, der auch Probleme damit hatte das Konzept der sich automatisch zum Finger bewegenden Sticks zu verstehen, kritisierte, dass ihm nicht klar war wie er selbige bewegen konnte.

Bereich: Lernförderlichkeit

Die Frage nach dem gefahrlosen Ausprobieren von Funktionen wurde ebenfalls unterschiedlich beantwortet. Eine Person hat die Frage auf der lezten Seite übersehen, zwei sind der Meinung gefahrloses Ausprobieren sei möglich, eine nicht. Es ist unklar in welcher Situation die Person das so empfand, sie hat keinen weiteren Kommentar gegeben.

Bereich: Kommentare

Im Bereich der freien Kommentare wurde erneut nach einer textlichen Beschreibung der Icons gefragt - ebenso wie nochmal die fehlende Rennen-Funktion bemängelt. Ein erfahrener Proband wünschte sich zusätzlich zum Navigationspfeil eine vollständige Minimap um Abkürzungen herausfinden zu können. Eine Person hatte den Endpunkt - die Kirche nicht als solche erkannt.

4.3.2 Auswertung der Videoaufnahmen

Während des gesamten Testprozesses wurden alle Probanden gefilmt (siehe Abb. 4.24). Die Probanden wurden explizit nach dem Verfahren des Think-Aloud Protocols darauf

hingewiesen ihre Gedanken zu artikulieren, während sie den Test durchführten. Nur wenige kamen dieser Anfrage nach, die Meisten waren überwiegend still während sie der Aufgabe nachgingen. Es muss in Betracht gezogen werden, ob nicht etwas deutlicher auf diesen Teil des Tests hätte hingewiesen werden sollen. Fraglich ist auch, ob die Probanden das Gerät eher aus Scheu nicht in die Hand nahmen und es auf dem Tisch liegen ließen. Selbst der erfahrene Proband hat nicht versucht es in die Hand zu nehmen. Bei den ersten Probanden wurde explizit darauf hingewiesen die Geräte aufzunehmen, was der Tester aber schnell als zu starke Manipulation empfand. Für zukünftige Tests sollte von Anfang an klar gestellt werden, dass die Probanden das Gerät benutzen sollen als wäre es ihres, inklusive hochnehmen und in den Händen halten.

Ein nicht-erfahrener Proband war besonders aufschlussreich, da er viele Gesten versuchte um bestimmte Aktionen auszulösen. Es erschien ihm offenbar instinktiv passend, dass ein Streichen nach oben auf dem Display ein Springen auslösen müsste. Es handelte sich dabei auch um einen der wenigen Probanden, die konsequent versuchten ihre Gedanken auszusprechen. Er wies außerdem darauf hin, dass ihm zunächst nicht klar war, ob der Navigationspfeil auf ein Ziel oder nach Norden weist, es wurde ihm aber mit der Zeit klar. Ein Proband fragte sich, ob es sich bei der ebenfalls eingeblendeten Distanz um Meter handelt. Ebenfalls immer wieder angefragt wurde eine Sprinten-Funktion, ein Proband tippte dabei während er die Funktion andeutete spielerisch in die rechte obere Ecke des Bildschirms. Öfters wurde versucht statt einen Button zu drücken die im Weg stehenden Fässer mit dem Finger zu greifen und wegzuschieben. Es war den Probanden offenbar überwiegend klar, dass sie das Fass ausgewählt hatten, aber die Funktion der Buttons war unklar oder wurde sogar ignoriert. Erschwerend kam hinzu, dass aus technischer Sicht die Kismet-Auswertung der Touchposition offenbar extrem ungenau wurde, wenn sich das Button-Overlay im unteren Viertel des Bildschirms befand. Die Engine rechnete zudem teilweise dynamische Schatten auf Fässer nachdem diese sich ein paar Millimeter bewegt hatten, was von vielen Probanden als negatives Feedback auf ihre Aktion verstanden wurde, da die Fässer “schwarz wurden”. Für Verwirrung sorgte auch der Abbrechen-Knopf an der Seite des Bildschirms, der nicht deutlich genug als solcher markiert war. Er wurde von einigen Probanden völlig anders wahr genommen, nämlich als eigentlicher Aktionsbutton anstatt des Overlays. Die Marker auf den Fässern wurden generell sehr gut aufgenommen - jedem Probanden war sehr schnell klar, dass er irgendetwas mit den Fässern tun kann und muss. Ebenfalls irritierend empfanden die meisten Probanden das Konzept keinen anderen Gegenstand anwählen zu können, solange das Interaktionsoverlay für einen Gegenstand aktiv war. Sehr auffällig war auch, dass die Probanden, die das Tablet in der Hand hielten es offenbar umständlich ablegen oder in einer Hand halten mussten um mit Gegenständen interagieren zu können.

Abb. 4.24: Ein Proband beim Testen des Prototypen in der Videoaufnahme.

4.3.3 Fazit

Die größten Probleme sind eindeutig im Bereich des Interaktions-Button-Overlays zu suchen. Hier ist ein breites Spektrum an Verbesserungen nachzuholen. Weniger stark ausgeprägt waren Probleme mit der Markierung von Gegeständen und dem Wegpunktsystem. Somit ist der nächste Schritt nun, die Erkenntnisse aus dem Test zusammen zu fassen und nach möglichen Lösungen zu suchen.

5 Umsetzung des “iterative Design”-Konzepts, Iteration 1

Gemäß des Usability-Prinzips des iterative Designs muss nun auf die Erkenntnisse des ersten Durchlaufs (hier Iteration 0 genannt) aufgebaut werden. Hierzu müssen für eine erneute Designphase zunächst die erkannten Probleme nochmals zusammengefasst werden und anschließend mögliche Gegenmaßnahmen benannt werden um diesen Problemen entgegen zu wirken.

5.1 Designphase

Aus den Tests in Phase 0 ließen sich einige Teilbereiche der Applikation ableiten, die noch als problematisch angesehen werden. Diese sind im Speziellen:

Bereich Wegpunktsystem:

- Es ist unklar, ob es sich um einen Kompass oder einen Pfeil auf ein Ziel handelt.
  - Mögliche Lösung: An der Spitze des Richtungspfeils wird der Wegpunktname eingeblendet, evtl. inklusive der Distanz.

Bereich Interaktionsmarker:

- Die Markierungen auf den Gegenständen werden generell sehr gut wahr genommen.
- Die Markierungen werden als Buttons augefasst, die eher gedrückt werden als der Gegenstand selbst.
  - Mögliche Lösung: Die Marker als Buttons verfügbar machen

Bereich Interaktionsoverlay:

- Die Icons werden nicht mit der zugehörigen Funktion assoziiert.

  - Mögliche Lösung: Icons durch Texte ersetzen oder ergänzen.
- Der Wechsel zu anderen Gegenständen ist nicht möglich ohne die Aktion zunächst abzubrechen oder zu beenden
  - Mögliche Lösung: Die Auswahl von Gegenständen weniger restriktiv und dynamischer gestalten
- Der Abbrechen-Button am Rand wird an Wichtigkeit den Einblendungen über dem Gegenstand vorgezogen
- Der Nutzer muss das Tablet umständlich absetzen um die Interaktionsbuttons zu nutzen
  - Mögliche Lösung: Die Buttons nicht mehr über dem Gegenstand sondern seitlich platzieren, so dass sie mit dem Daumen erreicht werden können
  - Die Auswahl eines Gegenstands umgestalten. Alternativ entweder den Marker klicken oder aber mit der Kamera darauf zielen. Der Marker sollte seine Farbe ändern, wenn er gerade aktiv ist um Feedback zu geben

Bereich Bewegung:

- Sprinten-Funktion erwünscht
  - Mögliche Lösung: Die Gehgeschwindigkeit langsam bei längeren Bewegungen erhöhen.
  - Die maximale Gehgeschwindigkeit erhöhen
  - Einen Button auf der rechten Seite integrieren
- Springen-Funktion erwünscht
  - Durch einen Accelerometer-Stoß (Ruckartige Bewegung des Pads) die Figur zum springen bringen

Bereich Generell:

- Es fehlen Hilfetexte zur Bedienung
  - Hilfetexte und Grafiken beim ersten Benutzen einblenden

Somit ergibt sich ein recht eindeutiges Bild. Wie bereits zuvor festgestellt sind die hauptsächlichen Probleme im Bereich der eigentlichen Interaktion mit Gegenständen zu suchen. Davon abgesehen gibt es einige Verbesserungsmöglichkeiten im Bereich der Steuerung der Spielfigur. Die Designbereiche aus Kapitel 4 werden im Folgenden gewissen Revisionen unterworfen.

5.1.1 Aktiver Stick

Der aktive Stick, also das Wegpunktsystem wurde prinzipiell gut angenommen. Die größten Probleme waren, dass den Probanden nicht durchgehend klar war, dass der Pfeil auf ihr Ziel zeigt, sondern eventuell wie ein Kompass stets in die gleiche Richtung. Dem soll entgegen gewirkt werden, indem die Bezeichnung des Wegpunktes sowie dessen Distanz direkt am Stick über dem Pfeil eingeblendet wird. Die Position des Textes soll sich dabei, genau wie der Pfeil, automatisch immer der Position des Sticks anpassen, damit immer klar ist, dass diese Information diesem Interfaceelement zugehörig ist.

5.1.2 Kontextsensitive Interaktion

Im Falle des Interaktionsoverlays soll in dieser Version selbiges Overlay über dem Gegenstand, dass die Buttons zum Interagieren anzeigt, stark umgebaut werden. Es ist definitiv nachteilig, dass die Probanden das iPad aus der Hand legen mussten, um die Buttons zu klicken, deren Funktionalität sogar recht eingeschränkt war, da die implementierte Koordinatenabfrage sehr ungenau zu sein schien, was unterschiedliche Fingerkuppen etc. anbelangt. Es soll daher auf Buttons am rechten Bildschirmrand, also in Reichweite des Daumens, zurück gegriffen werden, die auch mit einheitlichen eindeutigen Bezeichnungen versehen werden und immer am gleichen Ort auftauchen. Auch die Verwendung der Piktogramme wurde stark kritisiert, da sie nicht so verständlich waren wie erwartet. Diese Buttons sollen nach wir vor kontextsensitiv sein und nur dann auftauchen, wenn sie auch benötigt werden. Das Konzept der Kontextsensitivität von Bedienelementen bleibt nach wie vor entscheidend, da es die Stärken des Touchscreens Bedienelemente dynamisch zu erzeugen voll ausnutzt. Der rechte Bildschirmrand ist deswegen als Ort angepeilt, da der in Iteration 0 dort vorhandene Abbrechen-Button den eigentlichen Buttons im Overlay mehrfach vorgezogen und bei weitem größere Aufmerksamkeit beigemessen wurde.

Die Auswahl eines Gegenstands wird insofern revisioniert, dass nicht mehr ein Touch auf den Gegenstand nötig ist, sondern ein Ausrichten des Blickwinkels zur Auswahl führt. Die prinzipiell gut angenommene Markierung von interagierbaren Gegenständen soll ihre Farbe ändern bzw. ergänzt werden, wenn ein Gegenstand ausgewählt wurde. Um das Zielen zu erleichtern wird ein Zielkreis eingeblendet, den der Nutzer über den Marker schieben kann, sobald er sich in der Nähe eines Gegenstands befindet. Dieser Zielkreis wird nur dann eingeblendet, wenn er auch eine Funktion hat und einen Sinn erfüllt, um nicht beim Erleben der virtuellen Welt zu stören. Der Nutzer soll jederzeit, ohne die Interaktion zu beenden, den Gegenstand mit dem er gerade interagiert einfach mittels Zielen abwählen oder ändern können. Ein “Abbrechen” Button wird somit unnötig. Die Info-Tags der “Betrachten” Funktion werden angepasst, um Hinweise zur Lösung des mit diesem Gegenstand verbundenen Problems oder seiner Funktionen zu geben.

5.1.3 Bewegung der Spielfigur

Die Steuerung mittels emulierter Sticks ist prinzipiell von den Probanden angenommen worden. Die Frage nach einer Funktion zum Sprinten ist zwiespältig zu sehen. Die Frage die sich stellt ist, ob die Nutzer diese Funktion tatsächlich vermissen, oder ob sie eine Folge des Leveldesigns ist - das heißt, dass die Laufwege für die Aufgabe so lang waren, dass sie sich diese Komfortfunktion schlichtweg wünschen würden, um die Wartezeit zu verringern. Die Frage nach der Funktion zum Sprinten tauchte in der Tat immer auf dem Weg zu den Zelten auf, eine Phase des Tests in dem nicht viel passiert und auch nicht viel zu sehen ist. Das heißt es handelt sich hierbei eher um ein Problem des Leveldesigns. Natürlich greift die Steuerung mit dem Leveldesign Hand in Hand - im Falle eines großen Projekts muss deswegen definitiv geprüft werden, ob solche langen Laufwege vorhanden und auch nötig sind, und falls ja sollte eventuell eine derartige Funktion integriert werden. Im Falle der Epic Citadel Demo wird die Spielfigur mit der Zeit schneller wenn man läuft - ein Umstand der keinem der Probanden aufgefallen ist, und auch nur bei vollem Ausschlag des Sticks auftritt. Für Iteration 1 soll noch keine erweiterte Funktion zum Rennen integriert werden.

Eine weitere Nachfrage ist eine Funktion zum Springen. Diesem Wunsch soll folge geleistet werden, und er soll als alternative Lösungsmöglichkeit für die Fässer-Barrikade vor dem Zelt mit dem Artefakt propagiert werden. Zu diesem Zweck soll in Iteration 1 dann tatsächlich, wie bei den Betrachtungen für den vorigen Durchlauf dargelegt, das Accelerometer genutzt werden. Ganz einfach aus dem Grund, weil Springen nur dann Sinn macht, wenn es mit einer Bewegung der Spielfigur einher geht - ansonsten würde der Charakter nur auf der Stelle springen. In der Bewegung hat der Nutzer typischerweise beide Hände am Gerät, die Finger ruhen auf den Sticks - er kann aber das Gerät aus dieser Position heraus recht einfach schütteln. Das heißt die Funktion “Springen” soll mittels eines Stoßes ausgelöst werden. Der Nutzer schüttelt das Gerät einmal und die Figur springt. Dabei hat er nach wie vor Gelegenheit die Sticks weiter zu nutzen.

5.1.4 Hilfetexte

Der Ruf nach Hilfetexten und Einblendungen war bei vielen Probanden sehr ausgeprägt. Eine Reihe von Tutorialbildschirmen vor dem Start der Anwendung soll dem Nutzer einen Einblick in die Steuerung gewähren. Der Nutzer soll diese mittels Touches auch überspringen können, sofern er es wünscht. Auf diesen Grafiken sollen die wichtigsten Elemente der Steuerung erklärt werden. Das betrifft natürlich vor allem Bewegungssteuerung, Wegpunktsystem und der Themenkomplex Interaktion. Mittels kommentierter Screenshots in denen das entsprechende Interface-Element hervor gehoben wird soll der Nutzer so auf die Anwendung vorbereitet werden.

5.2 Implementierungsphase

5.2.1 Wegpunktsystem

Die eher geringen Anpassungen am Wegpunktsystem sind schnell vorgenommen und bedürfen nur geringen Erklärungen. Aus dem Kismet Skript zur Platzierung des Pfeils am blickpunkt-kontrollierenden Stick kann eine Positionsvariable abgegriffen werden, die sich immer mit der Position des Sticks ändert. Dies war nötig, damit der Pfeil immer die richtige Position hält. Mittels eines Offsets in Y-Richtung zu dieser eh bereits errechneten Position kann einfach eine weitere Position errechnet werden, an der der Name des Wegpunkts und die Distanz dahin mittels “Render Text”-Node gesetzt werden kann. Die Implementation ist analog zu der des Pfeiles aus dem letzten Kapitel, lediglich die Offsets müssen geändert werden und die Rotation ist für den Text unnötig. Die neue Platzierung des Textes ist in Hilfebildschirm 2 (Abb. 5.6) zu sehen.

5.2.2 Revision des Interaktionssystems

Das Interaktionssystem erfuhr nach der Evaluation der ersten Version der Anwendung die größten Änderungen. Sämtlicher Code für das Button-Overlay wurde entfernt. Statt dessen wurden die Funktionen zum Render der Marker, also der Markierungskreise über benutzbaren Gegenstände stark erweitert, da diese die Basis für das neue System stellen. Ein weiterer “Line Of Sight”-Event wird eingerichtet, der so eingestellt ist, dass er nur reagiert, wenn der Spieler recht genau mit dem Zentrum seines Sichtfensters auf den Gegenstand zeigt. Dieser neue Event, der in jedem Gegenstand eingebracht werden muss erzeugt die eigentlich statischen Input Zones, die jetzt tatsächlich doch verwendet werden können, da ihre Position nicht mehr dynamisch geändert werden muss und sie immer am gleichen Ort am rechten Bildschirmrand auftauchen (siehe Abb. 5.1). Mit dem gleichen Event wird zudem ein kleines Kreuz in die Markierung des Gegenstands hinein gerendert, dass dem Nutzer verdeutlicht, dass er diesen und keinen anderen Gegenstand ausgewählt hat. Der Typus des ausgewählten Gegenstands wird mittels Variable gesetzt und weiter gegeben. Die eigentliche Aktion wird über einen auf die generierten Buttons hörenden “Mobile Input Access”-Event gestartet. Dieser löst die bereits aus dem vorigen Kapitel bekannten Funktionen zum Einsammeln des Artefakts oder Stoßen der Fässer aus, bzw. die Darstellung des zu dem Gegenstand zugehörigen Hilfetextes (siehe Abb. 5.2). Die Hilfetexte zu den Gegenständen wurden mit sinnvollen Informationen ausgestattet - z.B. enthalten die Fässer einen Hinweis darauf, dass man sie schubsen kann.

Schlussendlich wird noch ein Zielkreis in die Mitte des Blickfelds der Spielfigur gerendert wann immer sie in der Nähe eines interagierbaren Gegenstands ist und somit es etwas gibt, worauf gezielt werden muss. Zusammen mit dem Rendering eines Interaktionsmarkers wird auch immer eine Bool-Variable auf “True” gesetzt, die ständig geprüft wird. Ist diese

“True” wird der Zielkreis in das Interface gerendert, ist sie “False” wird der Zielkreis wieder entfernt (siehe Abb. 5.3). Das heißt immer dann, wenn irgendein Marker gerendert wird, wird auch der Zielkreis gerendert. So ist dieser kontextabhängig immer nur dann zu sehen, wenn er auch einen Nutzen hat. Das neue Interaktionssystem ist auf dem Screenshot für den Hilfebildschirm 3 (Abb. 5.7) in Aktion zu sehen.

5.2.3 Springen mittels Accelerometer

Die Implementierung der Springen-Funktion ist erstaunlich simpel. Das UDK bietet einen speziellen Event zum Auslesen der Bewegungssensorik an, den sogenannten “Mobile Motion Access”-Event. Dieser Event ist im Stande sowohl Rohdaten und damit aktuelle Orientierung des Gerätes auszugeben als auch einen Delta-Wert für alle Drehachsen, der die Differenz der Messwerte zwischen zwei Messintervallen ausgibt. Diese Delta-Werte sind damit ein Maß für die Stärke einer ruckartigen Ausrichtungsänderung des Geräts und damit genau der Wert, der eine bestimmte Schwelle überschreiten muss um das Springen auszulösen. Mittels Testen wurde ein Wert gefunden, der die Funktion gut auslösbar macht ohne das Gerät zu sehr schütteln zu müssen, aber hoch genug ist um nicht bei jeder kleinen Bewegung ein Springen auszulösen. Der Wert Delta-Pitch wird folglich ständig mit diesem Wert (800 bzw -800) verglichen. Der Vergleich muss sowohl in positiver wie in negativer Richtung gemacht werden, da die Sprungfunktion nur von der Stärke des Schüttelns abhängig sein soll, und nicht von der Richtung. Wird nun die Schwelle überschritten gibt eine “Console Command”-Node einen Befehl an die Engine, die Spielfigur springen zu lassen. Das nötige Command “jump” ist bereits fest integriert und musste nicht extra selbst aufgebaut werden. Das ganze sehr übersichtliche Kismet Script findet sich in Abb. 5.4.

5.2.4 Hilfetexte

Drei Hilfebildschirme sollen beim Start der Anwendung eingeblendet werden, die die grundlegenden Bedienkonzepte erläutern. Zunächst wurde versucht ein Video einzubinden, dass nach dem Intro abgespielt werden soll. Dies funktionierte auch zunächst scheinbar indem entsprechende Parameter in der Konfigurationsdatei der Engine angepasst wurden und das Video mit in das Gesamtpaket aufgenommen wurde, allerdings kam es nach dem Ende des Videos immer zu unerklärbaren aber reproduzierbaren Abstürzen der Anwendung. Deshalb wurde letztendlich eine andere Methode verwendet - die Grafiken werden nach dem Ende des Ladevorgangs wie alle anderen Interface-Elemente auch mittels Canvas-Klasse angezeigt. So wird mit den bereits bekannten “Draw Image”-Nodes nacheinander mit einem Delay von acht Sekunden jeder der drei Bildschirme angezeigt, und danach die Steuerung, die ja hinter den Grafiken liegt, frei gegeben. Der Nutzer kann die Bilder vorzeitig mittels Touches “weg drücken” wenn er die Information nicht

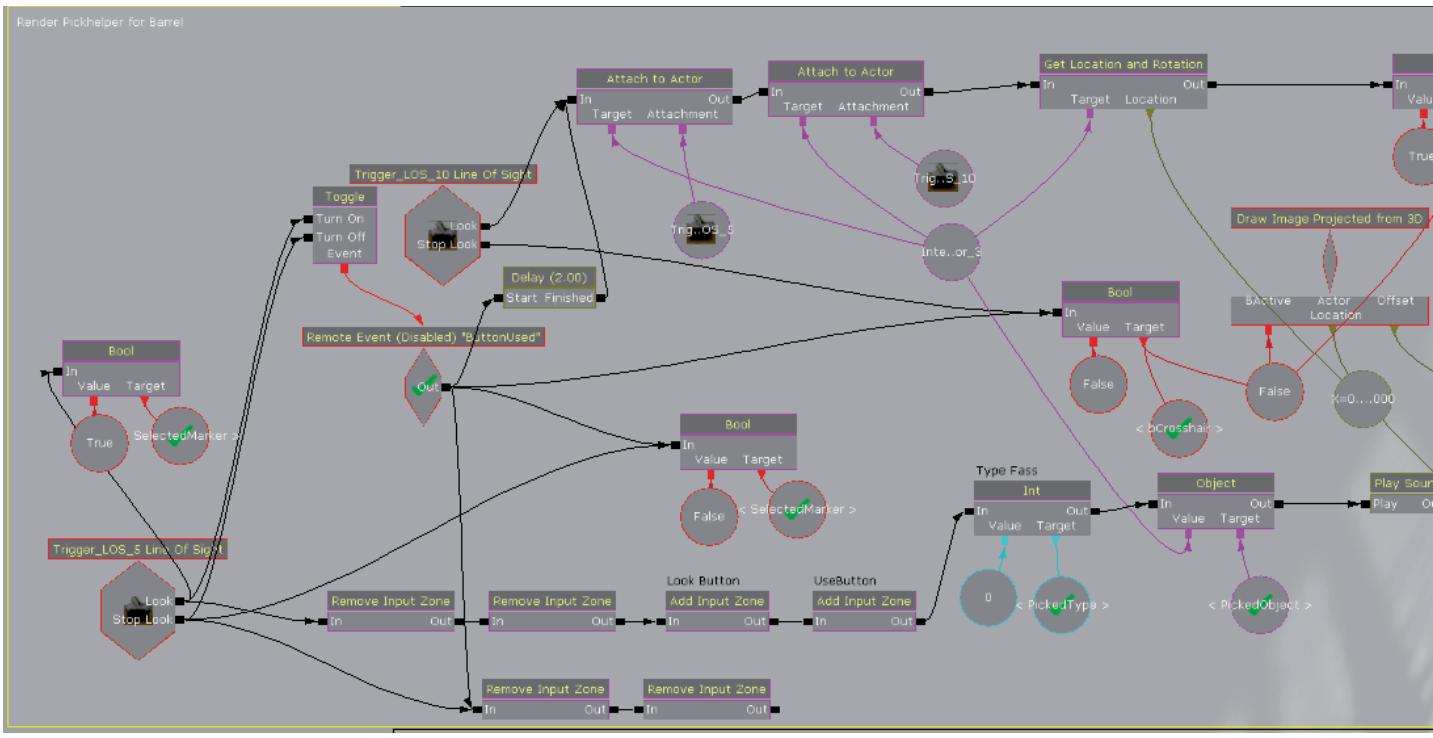

Abb. 5.1: Erweitertes Kismet Script eines interagierbaren Fasses. Das Script aus Abb. 4.13 wurde um einen zweiten “Line Of Sight”-Event erweitert. Dieser aktiviert sich nur, wenn der Gegenstand im Zentrum des Sichtfelds der Spielfigur ist. Sobald dies der Fall ist werden mittels “Remove Input Zone”-Nodes eventuell noch vorhandene Buttons zunächst gelöscht und anschließend mit “Add Input Zone”-Nodes neue Buttons für das Benutzen und Ansehen des Gegenstands angelegt. Das zusätzliche Löschen der Buttons vor dem Neu-generieren ist nötig, da manchmal das Löschen der Buttons beim Wegsehen vom Gegenstand (ausgelöst am Ausgang “Stop Looking” des “Line of Sight”-Events) nicht richtig funktioniert, wenn zwei Gegenstände sehr dicht nebeneinander stehen. Nach dem Anlegen der Buttons wird eine Variable mit dem Typ des ausgewählten Gegenstands gesetzt und die Variable “Picked Object” mit der ausgewählten Objektinstanz besetzt, damit Funktionen, wie das Aufsammeln des Artefakts oder das Bewegen der Fässer, einen Verweis auf das gerade aktive Objekt bekommen. Schließlich wird die Variable “ActivePickObjLocation” noch mit der Position des ausgewählten Gegenstands beschrieben. Diese wird den Marker, der ja eh auf dem Gegenstand zu sehen ist mit einem roten Kreuz zu versehen, damit klar ist, dass dieser ausgewählt wurde. Das Rendering dieser zusätzlichen Markierungen geschieht analog mit den bereits erklärten “Draw Image”-Events. Wurde ein Button vom Nutzer gedrückt, löst der Remote-Event “Button Used” aus, der das Skript wieder zurück setzt und so weitere Interaktionen mit dem Gegenstand ermöglicht.

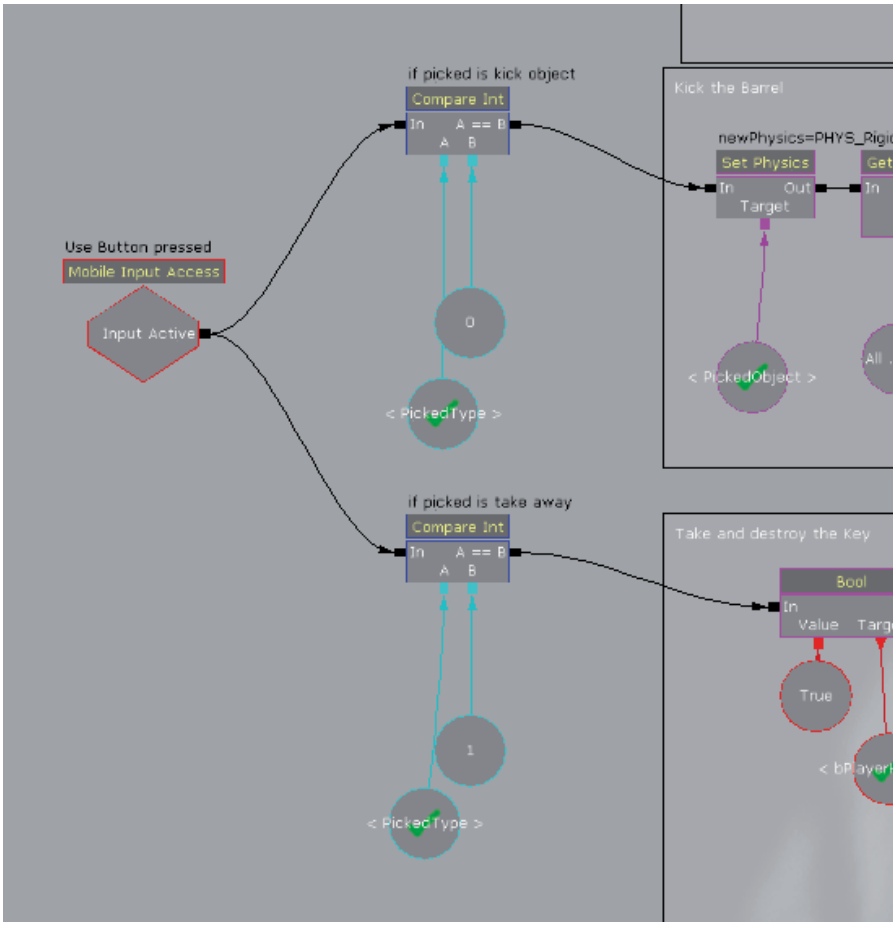

Abb. 5.2: Kismet Script zur Aktion mit einem Gegenstand. Die komplexe Verschaltung aus Abb.4.16 wird durch einen simplen “MobileInputAccess”-Event ersetzt, der dann feuert wenn der Benutzen-Button gedrückt wird. Danach wird, wie bekannt, anhand einer Variable entschieden, ob es sich um einen aufnehmbaren Gegenstand (das Artefakt) oder einen bewegbaren Gegenstand handelt. Die Funktion zum Betrachten des Gegenstands und Ausgabe des Hilfetextes wird analog mit einem weiteren “Mobile Input Access”-Event gezündet.

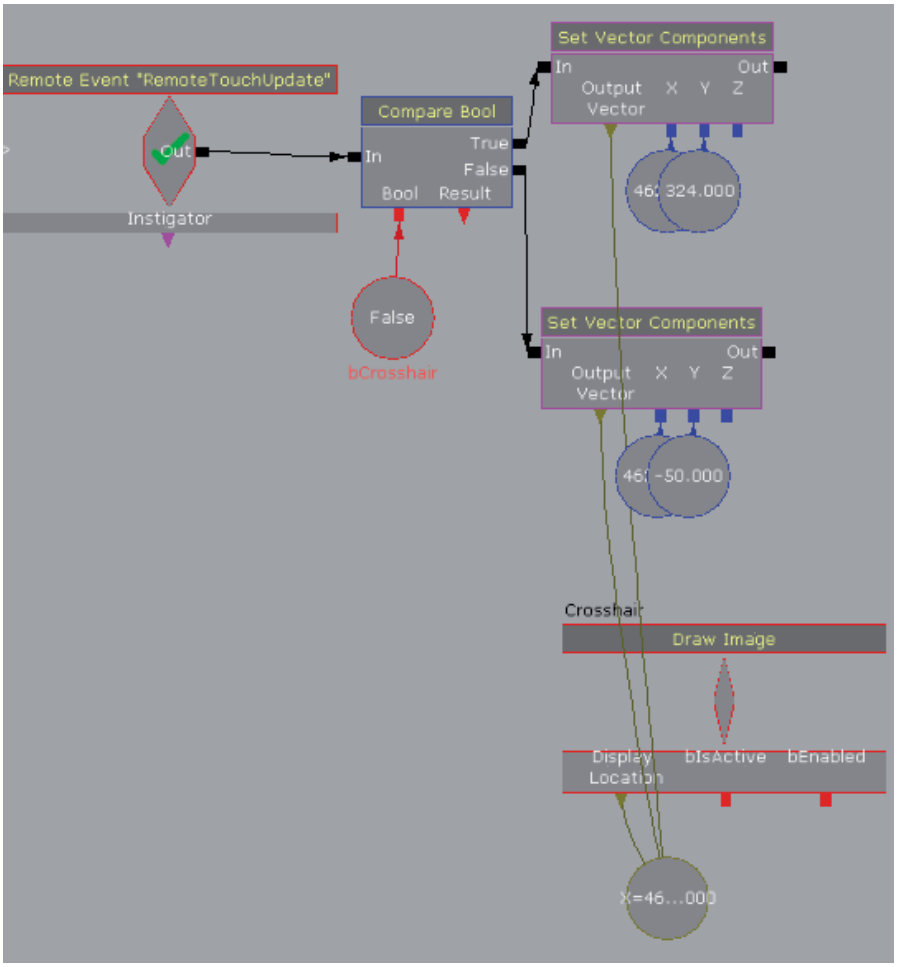

Abb. 5.3: Das Kismet Script zum Zielkreis. Der Remote Event “RemoteTouchUpdate”, der immer ausgeführt wird wenn der Nutzer den Bildschirm berührt, prüft die in Abb. 5.1 gesetzte bool-Variable. Ist die Spielfigur in der Nähe eines benutzbaren Gegenstands wird diese Variable “true” und die Position des Zielkreises so umgeschrieben, dass er im Zentrum des Bildschirms auftaucht. Ist die Variable “false” und die Spielfigur damit außer Reichweite, verschiebt sich der Zielkreis außerhalb des Bildschirms und wird unsichtbar.

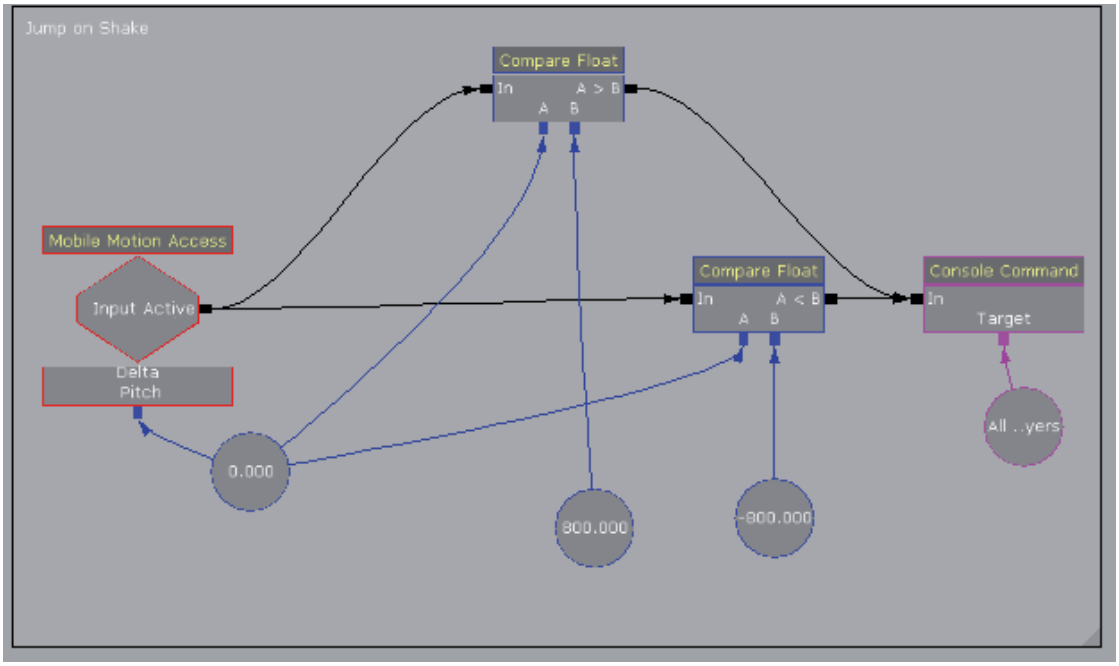

Abb. 5.4: Das Kismet Skript zum Springen mittels Schütteln des Geräts. Der “Mobile Motion” Event liest permanent die Bewegungsdaten des integrierten Accelerometers aus und gibt bei jedem Lesevorgang einen Impuls an die folgenden Nodes. “Delta Pitch” ist hierbei die Differenz der Messwerte zwischen zwei Messzyklen für den Rotationssensor. Wird das Gerät geschüttelt geht dieser Wert entsprechend der Stärke hoch - je nach Richtung der Bewegung entweder in die positive oder negative Richtung. Da hier die Richtung des Schüttelns nicht relevant sein soll wird geprüft ob der Wert entweder über 800 oder unter -800 geht und falls ja mittels “Console Command”-Node ein Befehl an die Engine geschickt. In den Properties der Node ist der Befehl hinterlegt, der schlicht und ergreifend “jump” lautet und die bereits in der Engine integrierte Funktion zum Springen ausführt.

Abb. 5.5: Der Erste von drei beim Start der Anwendung eingeblendeten Hilfebildschirme. Dieser Hilfebildschirm erklärt die Verwendung der Sticks. Zusätzlich wird erwähnt, dass das Schütteln des Geräts die Spielfigur springen lässt.

sehen will. Dies wurde auch analog zu vorigen implementierten Funktionen mittels des "Remote-Touch-Update"-Events realisiert, der immer dann zündet, wenn der Nutzer auf den Bildschirm tippt. Da es sich bei all dem um bereits gezeigte Konzepte handelt, soll hier auf eine genaue Wiedergabe des Kismet Scripts in Bildform verzichtet werden. Die drei Hilfebildschirme sind unter Abb.5.5 , Abb.5.6 und Abb.5.7 einzusehen.

5.3 Evaluationsphase

Der Usability-Test wurde mit der erstellten Applikation auf einem iPad erneut mit fünf zufällig gewählten Probanden durchgeführt. Vier von fünf Personen gaben an keine Erfahrung mit Touch-Tablets zu haben - drei von fünf Personen waren sogar

Abb. 5.6: Der Zweite von drei beim Start der Anwendung eingeblendeten Hilfebildschirme. Dieser Hilfebildschirm erklärt den Pfeil zum nächsten Wegpunkt und die Einblendung der Entfernung zum Ziel.

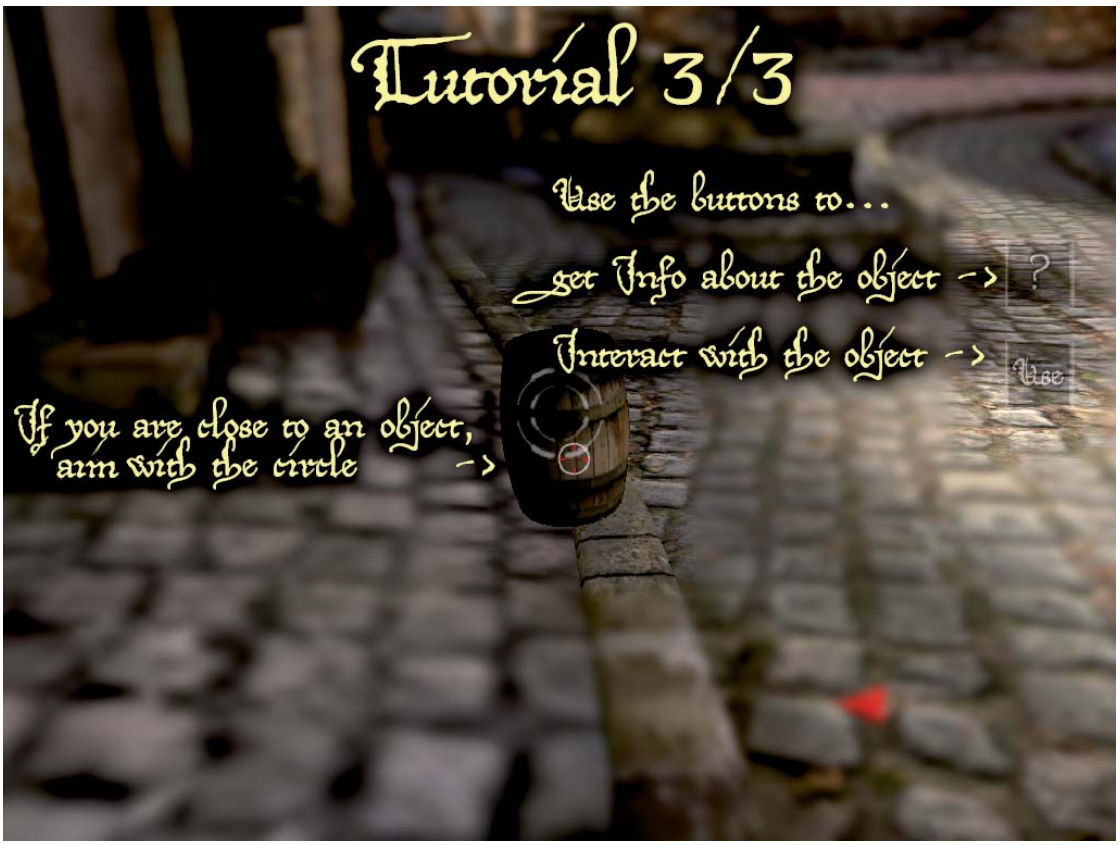

Abb. 5.7: Der Dritte von drei beim Start der Anwendung eingeblendeten Hilfebildschirme. Dieser Hilfebildschirm erklärt die Verwendung des Interaktionssystems.

in Studiengängen beheimatet, die nichts mit Informatik zu tun hatten. Entsprechend sind die Ergebnisse des Fragebogens etwas anders als beim ersten Durchgang. Es ist erstaunlich, welchen Unterschied grundsätzliche Technik- und PC-Erfahrung für den Umgang mit Tablet-Applikationen machen, auch wenn die Probaden keine Erfahrung mit der Geräteklasse an sich angaben. Dabei geht es nicht einmal unbedingt um die Bedienung, sondern sogar schlicht um Wahrnehmung und Identifizierung von Bedienelementen.

5.3.1 Auswertung des Fragebogens

Zunächst, wie zuvor, die Auswertung des Fragebogens nach den einzelnen Fragenkategorien. Eine Tabelle mit Auswertungsdaten findet sich im Angang A.4.

Bereich: Aufgabenangemessenheit

Deutliche Unterschiede in der Beantwortung der Fragen im Gegensatz zum ersten Test sind vor allem bei der Frage nach der Notwendgigkeit von “Tricks” zum Lösen der Aufgabe erkennbar. Vier von vier Probanden, die die Frage beantwortet haben hatten nicht den Eindruck Unzulänglichkeiten des Programms mit Tricks kompensieren zu müssen. Das ist eine deutliche Verbesserung gegenüber dem ersten Test, wo die Hälfte der Probanden das Gefühl hatte umständlich tricksen zu müssen - insbesondere bei den Fässern. Lediglich eine Kritik ist noch zu der Interaktion mit Gegenständen vorhanden, nämlich die, dass man zu genau zielen müsse. Offenbar wurde hier also die neue Variante der Interaktion deutlich besser angenommen. Der Fokus richtete sich interessanterweise dann eher auf die Navigation. Mehrere Probanden fragten nach einer vollwertigen Mini-Karte um sich besser zurechtfinden zu können und eine Stimme empfand den Richtungspfeil als verwirrend. Auch die Platzierung des Richtungspfeil wurde teilweise kritisiert, man wünschte sich eine fixe Platzierung am oberen oder unteren Bildschirmrand. Sehr erstaunlich sind auch die Angaben eines Probanden, der den Test vorzeitig abbrach - der Proband bemängelte, dass die Aufgabenstellung unklar war und das das iPad unverständlicherweise bei der Statue vibriert hätte, was natürlich tatsächlich sehr unverständlich ist, da das iPad kein Vibrationssystem besitzt. Generell wurde die Bewegungs- und Blickwinkel-Steuerung wesentlich stärker als verwirrend kritisiert als noch beim ersten Test, obwohl sich nicht viel geändert hatte.

Bereich: Selbstbeschreibungsfähigkeit

Auch im Bereich Selbstbeschreibungsfähigkeit wird erneut nach einer Minimap gefragt, außerdem wurde mehrfach gewünscht, dass die Aufgabe, die zu erledigen ist auch im Programm stärker kommuniziert werden soll. So hatten zwei Probanden offenbar Probleme

den Fokus für die Aufgabe zu halten, sie hätten sich deutlichere Hinweise darauf gewünscht was zu tun ist. Ein Proband wünschte sich eine Art “Questlog”, dass ihn zu jeder Zeit darüber auf dem Laufenden hält welche Schritte noch vor ihm liegen. Außerdem empfanden zwei Probanden die Schriftarten in der Applikation als schwer lesbar - etwas was ein Experiment war, da diese bewusst etwas altertümlicher gehalten waren, um sie an die Stimmung der Applikation anzupassen. Offenbar wurde dies eher negativ empfunden. Die Fragen an sich wurden recht ähnlich zum ersten Test beantwortet, der deutlichste Unterschied liegt in den Kommentaren. Die Buttons und die Zuordnung zu ihrer Funktion wurden nicht mehr kritisiert.

Bereich: Steuerbarkeit

Sehr erfreulich ist im Bereich Steuerbarkeit die Frage danach ausgefallen, ob die Spielfigur manchmal etwas tut, was der Nutzer nicht wollte. Hier wurde im ersten Test noch die Interaktion mit den Fässern als unvorhersehbar gewertet. Im zweiten Test hingegen wird diese Frage von allen Probanden mit “Nein” beantwortet. Zunehmend häufig wurden Performanceprobleme bemängelt - das Programm würde zu sehr ruckeln. Ein Proband hatte durch Zufall herausgefunden, dass das Programm durchaus eine Speicherfunktion besitzt, sofern man es beendet und wieder einsteigen möchte.

Bereich: Erwartungskonformität

Für die getestete Gruppe gab es im Bereich Erwartungskonformität vor allem Probleme im Bereich der Platzierung und Handhabung von Bewegungskontrollen. Ein Proband bemängelt, dass er immer nur entweder Laufen oder den Blickwinkel ändern kann - offenbar war diesem unbekannt, dass es sich um einen Multitouch-Bildschirm handelte, der mehr als einen Finger gleichzeitig verarbeiten kann. Die Springen-Funktion wurde in ihrer Handhabung von einem Probanden als “tricky” empfunden, generell wurde die nach wie vor sehr interessante Frage nach dem erwartungsgemäßen Verhalten des Programms auf Bewegungen des Tablets aber außerordentlich positiv beantwortet. Im Gegensatz zur ersten Version sind vier von fünf Probanden nun der Meinung, dass die Applikation so auf Bewegungen reagiert wie sie sollte. Ebenfalls leichte Verbesserungen sind in der Verständlichkeit der Bedienelemente nachweisbar, ein sicherlich positives Ergebnis, wenn man von der generell weniger starken Vertrautheit der Probanden mit dem Medium ausgeht. Auch sind alle bis auf einen Probanden (der die Frage als nicht zutreffend betitelte) nicht vom Verhalten des Programms auf bestimmte Eingaben überrascht, was eine deutliche Verbesserung gegenüber dem ersten Test darstellt. Erstaunlich: Ein Proband wünschte sich, dass er die Applikation mit seinen Daumen steuern könnte - besagter Proband hatte das Tablet nicht einmal vom Tisch aufgehoben, obwohl genau

das, was er sich wünschte, die vermutlich effizienteste Art gewesen wäre die Applikation zu bedienen und auch definitiv in der aktuellen Fassung möglich gewesen.

Bereich: Fehlertoleranz

Im Bereich Fehlertoleranz wurde ebenfalls die erste Version wesentlich schlechter bewertet, allen voran wegen der Schwierigkeiten mit der Gegenstandsinteraktion. Im aktuellen Test wurde in den Kommentaren lediglich auf Ruckeln hingewisen - generell wird das Programm als robust empfunden.

Bereich: Individualisierbarkeit

Wiederum überraschte im Bereich Individualisierbarkeit ein Proband mit der Antwort auf die Frage ob er die Steuerung seinen Wünschen entsprechend anpassen könne, die er mit ja beantwortete, obwohl keine derartige Funktion vorhanden war. Kritisiert wurde, dass man zu leicht von den Sticks abrutschen würde.

Bereich: Lernförderlichkeit

Die Frage danach, ob man gefahrlos Dinge ausprobieren konnte, wurde exakt wie im ersten Test beantwortet. Ein Proband (der den Test vorzeitig abbrach) fand, dass es zu wenig zu probieren und erkunden gäbe. Ein anderer Proband hingegen kritisierte, dass unsichtbare Wände einen Absturz der Spielfigur von der Klippe verhindern würden, er hätte sich offenbar mehr Gefahr gewünscht.

Bereich: Kommentare

Im Bereich der freien Kommentare wurde wiederum hauptsächlich die Steuerung der Spielfigur kritisiert. Der Stick hätte sich verschoben - offenbar wurde das Feature, dass sich der Stick ständig der Daumenposition anpasst, als Fehler empfunden. Generell wurde nach zusätzlichen Hinweisen zur Steuerung gefragt - die Steuerung wurde von einem Probanden sogar als unbequem empfunden. Besagter Proband wünschte sich, dass Programm mit einem Stift bedienen zu können. Ein Proband empfand außerdem die Sicht aus der First-Person-Perspektive als ungünstig, er würde Spiele mit Sicht von oben vorziehen. Zu guter letzt wurde erneut nach einer Minimap gefragt.

5.3.2 Auswertung der Videoaufnahmen

Die Videoaufnahmen unterstützen grundsätzlich die Aussagen des Fragebogens. Viele Probanden hatten Probleme mit den Steuersticks. Recht häufig ist das Problem aufgetreten, dass der Finger aus dem Touchbereich des iPads herausrutscht und dann auf den nicht sensitiven Rand kommt, was natürlich die Bewegung abbricht. Dies ist eher ein Usability-Problem des iPads, hat aber dennoch enorm viele Probanden betroffen. Generell wird die Steuerung als schwierig empfunden, ein Proband meinte er würde sich daran erinnert fühlen, wie es war, als er das erste mal Auto gefahren ist. Ein Proband zog es vor nur einen Stick (den zum Bewegen) zu nutzen und den Blickwinkel mittels Wischen über den ganzen Bildschirm zu ändern, offenbar ist diese Funktion nach wie vor sinnvoll.

Auch wenn es kaum Kritik zum Interaktionssystem in den Fragebögen gab, ist doch versucht worden Gegenstände mittels Touch auszuwählen. Hier kann überlegt werden, ob doch noch eine Funktion eingebaut wird, die die Kamera zum Gegenstand dreht, sobald dieser mittels Touch ausgewählt wird. Die Barriere vor den Fässern war allerdings ein weniger großes Hindernis wie zuvor - sie wurde leicht entschärft. Da man auch über sie springen konnte, konnte man sich mit Glück so sehr in sie hinein drücken, dass man das Artefakt einsammeln konnte ohne die Fässer zu bewegen. Vier von fünf Probanden haben allerdings die Funktion zum Bewegen der Fässer genutzt.

Erschreckend ist auch in der Auswertung der Videosequenzen, dass es fachfremde Probanden gab, die einzelne Interface-Elemente gar nicht wahrgnommen haben. So wurden ganze Sticks übersehen - inklusive Wegpfeil - und sich widerholende Meldungen in Textform auf dem Bildschirm überhaupt nicht gesehen wurden. Auch hier könnte ein kontextsensitives intelligentes Hilfesystem eventuell solche Elemente zusätzlich hervorheben.

5.3.3 Fazit

Man kann es schlichtweg erstaunlich nennen, inwiefern sich doch die Wahrnehmung von Probanden aus anderen Studiengängen, die weniger häufig mit Computersystemen umgehen, von denen unterscheidet, die dies öfters tun. Ein Informatik-affiner Nutzer, der noch nie ein Tablet benutzt hat, nimmt dennoch die Anwendung völlig anders wahr als einer, der wenig mit Computern zu tun hat. So rückten dann auch andere Bereiche der Applikation sehr viel mehr in den Fokus, allen voran die Bewegungssteuerung an sich. Das Stick-Konzept - wenn auch grundsätzlich oft als problematisch empfunden - wurde von der zweiten Gruppe dramatisch schlechter akzeptiert als von der ersten Gruppe. Teilweise wurden Elemente des Interfaces wie der Wegpfeil oder sogar der ganze rechte Stick vollkommen übersehen. Es wird umso ersichtlicher, dass man sich als Entwickler sehr genau darüber Gedanken machen muss, welchen Nutzerkreis die eigene Applikation erreichen soll und dann insbesondere diese Nutzer für die Tests heranziehen

sollte. Dennoch gab die Befragung völlig fachfremder Probanden durchaus interessante Erkenntnisse. Man kann davon ausgehen, dass solche Probanden nicht so sehr in ihrer Arbeits- und Herangehensweise durch andere Programme “vorbelastet” sind. Auf der Suche nach wirklich intuitiven Interfaceelementen ist dort das Ausmaß an möglichen Ergebnissen natürlich prinzipiell höher.

Im Vergleich zur ersten Iteration ist dennoch eine deutliche Verbesserung erkennbar. Der Hauptkritipunkt der ersten Version - die Interaktion mit Gegenständen wurde annähernd komplett behoben. Ausgenommen eine Stimme, die der Meinung war man müsse zu genau zielen, gab es nicht eine negative Meldung zum neuen System, und selbst diese Stimme hatte offenbar erkannt, dass sie auf Gegenstände zielen muss, um mit ihnen zu interagieren, was ja quasi der Schlüssel zur gesamten Funktion ist.

Interessant ist auch, dass nicht erneut nach einer Funktion zum Sprinten gefragt wurde, ein Kritikpunkt der im ersten Test noch sehr präsent war aber die These unterstützt, dass es sich eher um ein Leveldesign-Problem handelt. Dass die eher unerfahrenen Probanden des zweiten Tests dieses Problem nicht als solches wahr genommen haben mag daran liegen, dass ihnen Vergleichsmöglichkeiten fehlten - somit ist es tatsächlich weniger ein Problem der Steuerung an sich und muss eher im Leveldesignkontext betrachtet werden.

Die Integration der Hilfetexte vor dem Start der Applikation ist zwiespältig zu betrachten, denn viele der Nutzer übersprangen sie teils absichtlich, teils unabsichtlich - selbst die Probanden, die sie aufmerksam gelesen haben, haben teilweise danach die Informationen vergessen. Auch hier kann überlegt werden das Konzept der Kontextsensitivität konsequent weiter zu führen und zu erkennen, wann ein Nutzer offenbar Probleme mit einer Funktion hat und dann entsprechende Hilfe anzubieten.

Sehr positiv ist auch die Verwendung des Accelerometers verlaufen. Die geringe Präzision des Sensors spielte für die Verwendung als Buttonersatz zum Springen keine Rolle, zwei der fünf Probanden nutzten die Springen-Funktion explizit um über die Barrikade zu kommen, was als voller Erfolg gewertet werden kann. Zudem empfand der Proband, der die Funktion als “tricky” titulierte offenbar sogar etwas Spaß daranm, das etwas unkonventionelle System des Schüttelns zu verwenden um die Spielwelt zu manipulieren. Hier ist sicherlich Raum für weitere mögliche Funktionen, sofern sie nicht präzises agieren erfordern.

Abschließend kann für eine dritte Iteration einiges weiteres abgeleitet werden. Da im Zuge dieser Arbeit keine dritte Iteration durchgeführt wird, soll dies nun theoretisch bleiben. Es ergeben sich analog zur ersten Evaluation folgende Kernbereiche und Ideen zur Beseitigung der Probleme:

Bereich Wegpunktsystem:

- Die Platzierung des Wegpfeils wird als ungünstig empfunden

  - Mögliche Lösung: Den Radius um den Stick vergrößern, damit der Pfeil weniger häufig verdeckt ist.
  - Alternativ: Den Pfeil gemäß Gestaltungsrichtlinien aus Kapitel 3 zentriert oben anbringen und auf die Erweiterung des Sticks mit dem Pfeil verzichten.

- Die Entfernungsanzeige und der Pfeil an sich reagieren manchmal zu langsam oder zu spät
  - Mögliche Lösung: Es handelt sich offenbar um ein "Script-Lag" - das iPad kommt nicht hinterher die Kismet Scripte auszuführen. Hier sind umfangreiche Performancetests nötig, um festzustellen wo Optimierungspotentiale liegen.
- Die Probanden wünschen sich eine Minimap
  - Mögliche Lösung: Eine Minimap integrieren, hier allerdings mit der Einschränkung, dass das Interface so sauber und minimalistisch wie möglich bleiben sollte. So ist denkbar die Minimap mit einem Button bei Bedarf einzublenden.

Bereich Interaktion:

- Die Markierungen auf den Gegenständen werden generell sehr gut wahr genommen.
- Alle Probanden haben die Funktion zum Interagieren mit Gegenständen erfolgreich ohne viele Fehlversuche oder Frustrationserlebnisse verwendet.
- Manche Probanden versuchten dennoch Gegenstände durch Touch auszuwählen
  - Mögliche Lösung: Es sollte zusätzlich eine Funktion integriert werden, die die Kamera bei einem Touch auf den ausgewählten Gegenstand zentriert (und ihn damit auch automatisch auswählt)

Bereich Bewegung:

- Die Springen-Funktion wurde überwiegend gut angenommen
  - Verbesserung: Die Funktion etwas sensibler einstellen, um sie etwas leichter bedienbar zu machen.
- Die automatisch zum Finger springenden Sticks werden nicht verstanden, die Bewegung als Fehler aufgenommen
  - Mögliche Lösung: Die Sticks optional in ihrer Position fixieren. Eine enstprechende Funktion gehört in ein Optionsmenü.

  - Alternativ aber experimentell: Ein kleiner Button unterhalb des Sticks um ihn zu fixieren (beispielsweise ein kleines Schloss) - diese Funktion müsste allerdings sehr ausgiebig evaluiert werden.

Bereich Generell:

- Die Informationen der Hilfetexte sind nur mäßig aufgenommen worden
  - Mögliche Lösung: Das Konzept der Kontextsensitivität auch auf die Hilfe anwenden. Mithilfe von Scripts erkennen, wann der Nutzer vermutlich Hilfe braucht (häufige Kollisionen mit Wänden, langes Stehen an einem Platz, im Kreis gehen etc.) und dann unaufdringliche Tips einblenden.
- Die Texte sind schwer zu lesen
  - Mögliche Lösung: anderen Font benutzen.
- Die Aufgabe wird vom Programm nicht genau genug verfolgt, Missionsziele sind unklar.
  - Mögliche Lösung: Eine Taskliste, die durch ein Menü erreichbar ist, oder besser entsprechend des Prinzips der kontextsensitiven Hilfe auftaucht, wenn Probleme erkannt werden.
- Die Applikation neigt zum Ruckeln.
  - Mögliche Lösung: Wie bereits erwähnt extensive Performancetests um fest zu stellen wo Optimierungspotentiale liegen.
- Die Steuerung ist nicht an spezielle Bedürfnisse anpassbar.
  - Mögliche Lösung: Ein Menü integrieren, dass gewisse Optionen (beispielweise das Fixieren von Sticks) optional anbietet.

Damit endet die Evaluationsphase des zweiten Durchlaufs im iterative Design Prozess. Nun ist es an der Zeit Ergebnisse und Schlüsse aus den Abläufen zu ziehen und finale Erkenntnisse festzuhalten.

6 Ergebnisse

In diesem Kapitel werden die Ergebnisse der Arbeit vorgestellt. Dabei wird auf die gewählten Methoden, Lösungswege und Probleme eingangen. Das Kapitel ist hierfür in die einzelnen komplexen Themenbereiche unterteilt und endet in einem Fazit.

6.1 Usability

Retrospektiv kann festgestellt werden, dass die teilweise gut erprobten Methoden und Erkenntnise aus der Usability auch im speziellen Gebiet dieser Thesis Anwendung finden können. Es ist allerdings ohne Frage so, dass gerade im Bereich der Grundregeln noch verhältnismäßig wenig an Material existiert, dass speziell für diesen Bereich entwickelt wurde. Einige Hersteller versuchen ihren Entwicklern Konzepte, Ideen und Richtlinien an die Hand zu legen um Anwendungen für Touchscreen-Geräte zu entwickeln, was sehr löblich ist, da dies nur dazu führen kann, dass die Qualität der Anwendungen steigt. Andere Hersteller hingegen sparen sich derartige Kapitel in ihren Dokumentationen. Dabei ist gerade die grundsätzliche Übertragbarkeit von Usability-Prinzipien über die Hardwaregrenze hinweg absolut gegeben. In den vergangenen Kapiteln wurde gezeigt, dass ein System wie das der Bundestanstalt für Arbeitsschutz, dass vom Anschein her absolut unanwendbar auf Thematiken wie Gaming oder auch Hardware wie Tablets scheint doch anwendbar ist, wenn man es nur leichten Modifikationen unterzieht. Hier ist ohne Frage in tiefergehenden Publikationen noch viel Raum für neue Forschungsarbeiten, die auf alten Erkenntnissen aufbauen können. Es lässt sich festhalten, dass Usability, wie auch immer wieder definiert wurde, kontextabhängig ist. Es ist deshalb erforderlich, Fragen und Bewertungsmethoden an die eigenen Bedüfnisse anzupassen. Man kann sich nur wünschen, dass die Hersteller von Hard- und Software neuerer Generationen sich dieser Tatsache bewußt sind und bleiben, auch wenn der Post-PC, wie Steve Jobs selbst in einer seiner Präsentationen die modernen Tablets bezeichnete, eines Tages von einer anderen, wiederum völlig neuen Geräteklasse abgelöst werden sollte.

6.2 Iterative Design-Prozess

Der “Iterative Design-Prozess” ist die Basis, auf der sich die gesamte Entwicklung der Prototyp-Anwendung in dieser Arbeit stützt. In zwei Iterationen wurde die Anwendung

entwickelt, evaluiert, verfeinert und wiederum evaluiert. Auch hier hat sich ohne Frage die Übertragbarkeit bekannter Ideen und Herangehensweisen auf die Welt der Tablets eindeutig gezeigt. Es ist überraschend, wie sehr sich die Wahrnehmung des Entwicklers auf eine Applikation, von der eines Nutzers unterscheidet, obwohl sich dies bereits im Kapitel über Usability angedeutet hatte - der Effekt ist um einiges dramatischer als Texte und Meinungen von Autoren vermuten lassen. Bedienelemente, die als überlegen oder besonders intuitiv empfunden werden sind es nach Überprüfung mit Probanden mitunter überhaupt nicht. Hier ist besonders das Interaktionsoverlay aus Iteration 0 zu nennen, dessen Handhabung offenbar für die meisten Tester völlig unintuitiv war. Hätte man auf ein Korrektiv wie das iterative Design verzichtet und wäre das Produkt auf dieser Basis weiter entwickelt worden hätte dieses Bedienelement seinen Weg in ein fertiges Produkt gefunden und damit eventuell sehr viele Menschen sehr frustriert.

Es ist auch gerade in diesem Kontext nochmals sinnvoll darauf hinzuweisen, dass gerade auch unfertige Software und Prototypen häufigen Tests unterzogen werden sollten, da ein derartiges Problem in frühen Stadien noch viel leichter behoben werden kann - schon allein aus programmiertechnischer Sicht. Die Frage danach, wie häufig solche Tests wiederholt werden sollten ist dabei nicht so einfach zu beantworten - im Zuge dieser Arbeit waren in der zweiten Befragung Probleme aufgetaucht, die in der ersten noch gar keine Erwähnung fanden, obwohl sie in gleicher Art offenbar vorhanden waren. Hierbei deutet sich ein Effekt an, dass zunächst nur die dringendsten Probleme von Probanden wahrgenommen werden, und so die Probleme, deren Dringlichkeit nicht so hoch ist gar keine Erwähnung finden. Den in dieser Arbeit entwickelten Prototy betreffend wäre auf jeden Fall eine dritte Iteration sinnvoll und hilfreich gewesen, da bei der zweiten Untersuchung noch sehr viel Neues ans Tageslicht kam, dass bis dato unbekannt war. Das heißt die Häufigkeit der Tests hängt sicherlich vom jeweiligen Softwareprojekt ab, es ist aber davon auszugehen, dass mit zunehmenden Durchgängen - sofern die Probleme immer gleich behoben werden - der nutzbare Informationsgehalt mit der Zeit geringer wird.

Die Tests müssen dabei prinzipiell gar nicht so aufwändig gestaltet sein: die Menge von fünf Probanden war absolut ausreichend um die schlimmsten Probleme identifizieren zu können. Es ist sogar fraglich, ob der Fragebogen in diesem Ausmaß nötig gewesen wäre, denn die interessantesten und aufschlussreichsten Informationen resultierten nicht aus der Beantwortung der Fragen an sich mittels ankreuzen, sondern aus den jeweiligen Kommentaren, die die Probanden zu den Fragen schriftlich geben konnten. Die angekreuzten Fragen selbst gaben eher ein Stimmungsbild wieder, dass den Eindruck erweckte aufgrund der geringen Probandenanzahl nicht wirklich repräsentativ zu sein. Dennoch sind die Fragen notwendig, um die Denkweise des Probanden in Richtung der Fragen anzuregen. Das heißt, will man den Fokus eher auf ein generelles, gut belegbares Stimmungsbild über bestimmte Bereiche einer Applikation legen, müsste man die Probandenzahl erhöhen, allerdings erhält man dadurch nicht unbedingt mehr konkrete Verbesserungshinweise. Diese begannen sich sogar schon bei fünf Probanden enorm schnell zu wiederholen. Auch

sind die Videoaufnahmen ein interessantes Mittel, da Details enorm entscheidend sind. Wenn mehrere Probanden bestimmte Gesten oder Ideen haben eine bestimmte Aktion auszuführen muss in Betracht gezogen werden, dass diese Variante eventuell intuitiver wäre - die Probanden selbst wissen aber teilweise gar nicht, dass sie gerade eine sehr wertvolle Information gegeben haben. Dieser Versuch an einer bestimmten Stelle der Software auf auf eine bestimmte Art und Weise etwas auszulösen wird auch nicht in den Fragebögen von den Probanden erwähnt, kann aber eindeutig im Video nachvollzogen und beobachtet werden.

Zuletzt soll nochmals auf die Auswahl der Tester hingewiesen werden. Gerade wenn geplant ist viele kleine Tests mit etwa um die fünf Probanden durchzuführen, ist es offenbar umso wichtiger, dass man Probanden nicht auf gänzlich zufälliger Basis auswählt, sondern optimalerweise solche aus der Gruppe, die typische Benutzer der eigenen Anwendung darstellen. Es hat sich zwar gezeigt, dass gerade die Probanden interessant sind, die noch nicht zu viel Erfahrung mit ähnlichen Produkten hatten, da sie dann eher unbeeinflusst an den Test heran gehen, jedoch ist es sinnvoll wenn sie grundsätzlich zur Zielgruppe für die Applikation gehören. Im ersten Test wurden ausschließlich Medieninformatiker befragt, im zweiten Test handelte es sich um eine sehr gemischte Gruppe. Das Feedback der Medieninformatiker war eindeutig differenzierter, da sie grundsätzliche Fähigkeiten im Umgang mit technischen Geräten und eine entsprechend geschulte Wahrnehmung mitbrachten. Im zweiten Test kam es in zumindest einem Beispiel zu einem Probanden, der derart technikfremd im Denken war, dass schon schlichtweg die Verwendung des Gerätes an sich eine Barriere darstellte. Die aus dem Fragebogen und Video entnehmbaren Informationen werden dann immer weniger auf die eigenen Anwendung beziehbar, da grundsätzliche Probleme zum Gerät oder technischen Systemen an sich an die Oberfläche treten. Diese Informationen sind zwar auch interessant, aber zunehmend irrelevant für die eigene Applikation - oder zumindest in Bereichen angesiedelt die nicht durch die eigene Applikation geändert werden können. Die Frage “wer ist mein User” ist also für die Auswahl der Probanden sehr relevant.

6.3 Tablet-Game-Entwicklung mit UDK

In diesem Abschnitt soll auf die eigentliche Entwicklungsarbeit mit der Unreal Engine eingegangen werden. Die Unreal Engine ist ohne Frage eine extrem fortschrittliche Engine, die auch einen verhältnismäßig einfachen Einstieg in die Entwicklung ermöglicht. Die Barrieren sind für ein derart komplexes Softwarepaket erstaunlich niedrig gesteckt, was ohne Frage ein positiver Punkt ist. Sehr visuelle und logisch begreifbare Werkzeuge wie Kismet erlauben dem Entwickler schnell durchaus komplexe Scripts zusammen zu setzen, ohne dafür tief in die Programmiersprache einsteigen zu müssen. Vorteilhaft ist auch, dass diese Kismet-Scripts sich sehr schnell abändern lassen, ohne dass es große

Zwischenschritte wie langwieriges Kompilieren erfordern würde - das heißt gerade für die Vorgänge im iterative Design ist es sehr geeignet, da es darum geht schnell Änderungen einzupflegen und erneut evaluieren zu können. Die Probleme tauchen dann aber recht schnell im Detail auf. So trifft man häufig auf nicht dokumentierte Fehler und Probleme in der Engine, die gerade im Entwicklungsbereich für mobile Geräte noch sehr vertreten sind. Zwar konnten alle auftretenden Fehler im Rahmen dieser Arbeit auf die eine oder andere Weise geschickt umgangen werden, aber dennoch ist der Zeitaufwand, der allein in die Identifikation des Problems investiert werden muss, enorm. Es kann mitunter Stunden dauern bis klar wird, dass Fehler die auftauchen nicht im eigenen Programmcode zu suchen sind, sondern in Problemen in der Engine. Natürlich handelt es sich beim UDK gerade im Bereich Anwendungen für mobile Geräte noch um eine Beta, aber dennoch ist es sehr ärgerlich, dass bestimmte Fehler nicht ausreichend dokumentiert vorliegen. So ist auch die generelle Performance gerade bei starkem Einsatz von Kismet ein großes Problem. Scripte, die im Simulator auf dem Entwicklungsrechner absolut sauber liefen, können auf dem Gerät selber plötzlich enorme Probleme machen - ohne dass es einen sichtbaren Ansatzpunkt gäbe, warum. Die Performance-Probleme äußern sich in Ruckeln oder sogar dem vorübergehenden Einfrieren aller Skriptaktivtäten, sind aber nicht immer reproduzierbar. So konnte so manches Performance-Problem durch Neustart der Applikation oder des ganzen Test-Geräts temporär gelöst werden, manch andere aber auch nicht. Diese ganzen Details lassen eine Unsicherheit beim Entwickler zurückbleiben, die zwar verkraftbar ist, solange man nur Prototypenentwicklung betreibt, aber enorm fatal sein könnte, wenn ein fertiges Produkt auf dem freien Markt ähnliche Verhaltensweisen zeigt. Das grundlegende Vertrauen in die eigene Software, zumal es Bereiche in der Engine gibt die dem Entwickler verschlossen bleiben und auf die er keinen Einfluss nehmen kann, ist dadurch eingeschränkt.

Zusammenfassend lässt sich festhalten, dass das UDK eine fortschrittliche Engine ist, in die ein Einstieg sehr gut auch für Neuanfänger möglich ist. Verschiedene große und kleine Probleme im Bereich der Entwicklung für mobile Geräte machen den Einsatz der Engine im produktiven, professionellen Umfeld zu einem gewissen Risko, es ist aber davon auszugehen, dass diese Probleme in den nächsten Monaten behoben werden. Davon abgesehen ist das Gesamtpaket UDK, dass derzeit kostenlos zu beziehen ist, für die Leistungen die es anbietet als absolut konkurrenzlos anzusehen.

6.4 Umsetzbarkeit neuer Steuerungsmethoden auf Touchscreen Tablets

In diesem finalen und letzten Abschnitt soll schließlich ein Fazit gezogen werden. Im Zuge dieser Arbeit wurden Probleme mit der Steuerung im behandelten Segment - den 3D-Anwendungen auf Touchscreen Tablets - beleuchtet und anschließend nach

Lösungen geforscht. In Fallbeispielen wurde gezeigt, dass es eindeutige Strömungen im Steuerungsbereich gibt, Ideen - die trotz der Tatsache, dass sie nicht optimal sind - immer wieder für die Steuerung von Applikationen Anwendung finden. Man kann zunächst mit Unverständnis auf diese Strömungen reagieren, sie beinahe unkreativ titulieren. Es ist nach wie vor fraglich, warum immer und überall von Konsolen bekannte Gamepads auf die eine oder andere Weise emuliert werden, und warum die speziellen Fähigkeiten der Geräte kaum oder nur auf seltsame Weise genutzt werden. Dennoch, und das ist das erstaunliche Ergebnis dieser Arbeit, kommt man nicht gänzlich um die Gamepad-Emulation herum.

Es handelt sich bei dieser Steuerungsweise um ein Konzept, dass so sehr in den Nutzern verankert zu sein scheint, dass es mittlerweile eine Erwartungshaltung darstellt. Der Versuch dieses Paradigma zumindest bei den Aktionsbutton zu brechen (das Interaktionsoverlay in Iteration 0) ist klar ersichtlich gescheiert. Dennoch heißt das nicht, dass es keinen Raum für Innovation gibt: Einige Ideen, allem voran die Kontextsensitivität von Interfaceelementen konnten sich auch in die zweite Version des Prototyps retten und sollten in einem theoretischen dritten Prototyp sogar noch stärker eingebracht werden. Es handelt sich hierbei aber eher um Evolution und weniger um Revolution. Die Frage die sich stellt und die auch in dieser Arbeit nicht ausreichend beantwortet werden kann, ist schließlich, ob die Gamepad-Emulation als Basis eines Steuerungskonzepts für die mobilen Geräte nun tatsächlich die Beste herangehensweise ist, oder nur eine, die einfach von einer sehr breiten Benutzerschicht erlernt wurde und deshalb jetzt erwartet wird. Es ist auch nicht zu unterschätzen, dass das Fehlen echter hardwareseitiger Kontrollelemente am Gerät definitiv grundsätzliche Probleme im Bereich Gaming schafft - die optimale Lösung für diese spezielle Niesche an Applikationen wäre wahrscheinlich tatsächlich hardwareseitig Gamepad-ähnliche Instrumente in die Geräte zu verbauen. Da dies aber nur ein Teilbereich der Anwendungen dieser Geräte ist - einer von vielen Anwendungsbereichen für ein mobiles Gerät wie ein Tablet, muss man wohl damit leben, dass gewisse Kompromisse notwendig sind um die Verwendbarkeit für ein breites Spektrum an Aufgaben zu ermöglichen. Dennoch - und das konnte eindeutig durch die Befragungen weniger technikaffiner Benutzer gezeigt werden - sind besagte Kompromisse alles andere als intuitiv und ohne Einarbeitung erlernbar.

Abschließend soll gerade hier die zuvor als für die eigentliche Programmentwicklung weniger hilfreich bezeichnete Stimme eines speziellen Probanden genannt sein, der sehr technikfremd war und nach eigener Aussage keine Computerspiele spielt. Besagter Proband wünschte sich, unterstützt durch eine kleine Zeichnung auf seinem Fragebogen, das Gerät wie ein Gamepad benutzen zu können und mit zwei Daumen steuern zu können. Es ist natürlich geradezu Ironie, dass dem Probanden nicht klar war, dass er genau das hätte mit dem Prototypen tun können, aber dennoch ist es eine interessante Tatsache, dass selbst ein Proband, der völlig themenfremd ist eine, derartige Steuerung als bevorzugt angeben würde. Der Proband zeichnete an den typischen Positionen in seiner Zeichnung - im unteren Bereich rechts und links - mittels zweier Kreuze zwei Sticks ein, die er als

Bewegungs- und Blickpunktkontrolle nutzen wollte.

7 Zusammenfassung und Ausblick

Diese Arbeit beschäftigt sich mit dem komplexen Problem der Steuerung von 3D-Anwendungen auf Touchscreen Devices. Dabei wird insbesondere auf direkte Steuerungskonzepte eingegangen, in der eine menschenähnlich agierende Spielfigur direkt mittels Eingaben kontrolliert wird. Diese meistens in der First-Person-Perspektive genutzten Programme sind seit vielen Jahren ein wichtiges und erfolgreiches Konzept im Bereich Gaming und auch dem produktiven Einsatz, beispielsweise für Architekturvisualisierungen oder Lernprogramme.

Touchscreenbasierte Geräte erobern mehr und mehr Bereiche des Computer-Alltags, und so werden zunehmend auch derartige Applikationen für besagte Geräte angeboten. Hier stellt sich allerdings die besondere Herausforderung eine Steuerungsform für diese Geräte zu finden, die sowohl inutitiv als auch effizient ist. Durch das Fehlen bekannten hardwareseitiger Schnittstellen mit dem Nutzer, allen voran Maus und Tastatur muss nach neuen Möglichkeiten gesucht werden mittels der vorhandenen Schnittstellen wie Touchscreen oder Bewegungssensoren ähnlich gute Konzepte zu realisieren, wie auf klassischen Computersystemen und Konsolen.

In einer Marktübersicht werden bekannte und erfolgreiche Applikationen untersucht und ihre Lösungen für besagte Problemstellungen bewertet. Dabei lässt sich feststellen, dass es einige sehr stark präsente Strömungen gibt eine Steuerung für derartige Systeme zu realisieren. Allen voran ist dies vor allem die Emulation von aus der Konsolenwelt bekannten Gamepad-Sticks und Buttons. Auf den Touchscreens werden Bedienelemente dargestellt, die versuchen die Eigenschaften dieser eigentlich aus der Hardware stammenden Lösungen nachzuahmen. Es ist überraschend, dass dabei nicht oder nur wenig versucht wird, die eigentlich sehr dynamischen Fähigkeiten des Touchscreens zum ständigen kontextbezogenen Anpassen von Bedienelementen zu nutzen und überwiegend rein statische Bedienelemente implementiert werden.

Um das Problem an der Basis greifen zu können, werden im dritten Kapitel deshalb Grundprinzipien der Usability beleuchtet. Es wird definiert, was Usability eigentlich ist und bedeutet und inwiefern sie auf Software im Allgemeinen, und auf Tablets im Speziellen anwendbar ist. Es wird festgestellt, dass viele der Tablet-Hersteller kaum Usability-Richtlinien für externe Entwickler zur Verfügung stellen, was einen beträchtlichen Missstand darstellt. Schließlich wird ein von der Bundesagentur für Arbeitsschutz entwickeltes ErgoNorm genanntes Verfahren modifiziert, um es im Gaming-Kontext

und für Tablets nutzen zu können. Im Zuge dieser Untersuchung wird ein Fragebogen erstellt, der zur Evaluation derartiger Systeme geeignet ist. Es werden typische Aufgaben abgeleitet, die in vielen Applikationen der behandelten Art eine Rolle spielen und in entsprechende Aufgabenstellungen umgesetzt, die Probanden im Zuge einer Usability-Studie ausführen sollen. Dabei handelt es sich im Wesentlichen um die Themenkomplexe Navigation in einer virtuellen Welt und Interaktion mit der Gegenständen und Orten in der Welt.

An einem praktischen Beispiel wird in Kapitel vier und fünf das Verfahren des iterative Designs auf einen Prototyp einer Tablet-Anwendung angewendet. Für die Methode des iterative Design wird zwischen jeden Entwicklungsschritt eine Evaluation mit Probanden zwischengeschaltet und die Ergebnisse daraus direkt in einen Prototyp implementiert. Hierbei markiert Kapitel vier den ersten Durchgang und Kapitel fünf den zweiten Durchgang. Im Zuge der Entwicklung des Prototyps wird mit innovativen neuen Steuerungsmethoden experimentiert, bei denen versucht wird, sich von den beobachteten statischen Interface-Elementen zu entfernen und neue Herangehensweisen zu testen.

Mit Hilfe der bekannten Unreal Engine wird besagter Prototyp auf Basis der interaktiven Grafikdemo “Epic Citadel” aufgebaut. Zum Einsatz kommt neben grundlegender Techniken zum Leveldesign das visuelle Skriptsystem Kismet, dass es ermöglicht schnell und einfach Prototypen für eine Evaluation zu erstellen. Im Zuge der ersten Iteration des Prozesses wird ein kontextbasiertes System zum Interagieren mit Gegenständen eingeführt. Ein Overlay über Objekten in der Spielwelt soll das Konzept der statischen Buttons ablösen. Der Nutzer kann mittels Touchscreen Objekte auswählen und dann mit ihnen interagieren. Parallel wird ein Navigationssystem implementiert, dass mit Hilfe eines an einen Steuerungsstick angepassten Zielpfeiles die Richtung zum nächsten Teil der für den Usability-Test zu erfüllenden Aufgabe weist. Das gesamte Steuerungskonzept wird anschließend mit Hilfe der zuvor erarbeiteten Evaluationsmethode untersucht. Es zeigt sich, dass das neue System zum Interagieren mit Gegenständen sehr schlecht angenommen wird.

Im Rahmen der zweiten Implementierungsphase wird das Interaktionssystem komplett überarbeitet. Der Nutzer kann in der neuen Version Gegenstände auswählen mit denen er interagieren will indem er mit der Kamera auf sie zielt. Buttons und Bedienelemente werden dann eingeblendet, wenn sie auch eine Funktion haben, ansonsten wird versucht nur so viel wie nötig in die Erfahrung der Spielwelt einzugreifen. Es werden erstmals Hilfebildschirme implementiert mit deren Hilfe die Steuerung erklärt wird. Ein System zur Nutzung der Bewegunssensoren des Tablets ermöglicht es dem Nutzer die Spielfigur mit einem Schütteln des Geräts zum Springen zu bringen. Auch diese Entwicklungsphase wird mit Hilfe der erarbeiteten Methoden überprüft. Es zeigt sich, dass die Änderungen am Interaktionssystem außerordentlich gut angenommen, aber dafür andere Probleme in den Vordergrund gedrängt werden. Die Zweite Evaluation, die hauptsächlich mit fachfremden und wenig technikerfahrenen Probanden durchgeführt wurde belegt, dass

es durchaus eine Rolle spielt welche Probanden gewählt werden und in Zukunft solche aus der typischen Zielgruppe für die Applikation bevorzugt herangezogen werden sollten. Dennoch können auch weniger erfahrene Probanden, die auch weniger mit Erkenntnissen aus anderen Produkten vorbelastet sind, interessante Strömungen und Erkenntnisse liefern. Es zeigt sich, dass die Idee Interfaces kontextsensitiv aufzubauen durchaus funktioniert. In den der zweiten Evaluation abgeleiteten Erkenntnissen wird für eine theoretische dritte Entwicklungsstufe beschlossen, diese Grundidee weiter auszubauen.

Abschließend wird in einem letzten Kapitel der gesamte Ablauf der Entwicklung zusammengefasst und bewertet. Es wird geschlossen, dass Methoden aus der Usability, auch von eher fremden Systemen wie Bürosoftware durchaus mit gewissen Anpassungen an den Gamingbereich und sogar auch für Tablets anpassbar sind. Es können viele Regeln und Erkenntnisse übertragen werden und auf diese Weise die Benutzerfreundlichkeit verbessert werden. Die Unreal Engine wird dabei als sehr fortschrittliche Engine wahrgenommen, die aber noch einige Monate an Entwicklungsarbeit braucht um sie wirklich produktiv für große kommerzielle Projekte einsetzbar zu machen, da sie noch viele Fehler beinhaltet - gerade im Bereich der Entwicklung für mobile Geräte. Dennoch handelt es sich dabei um ein sehr gutes und sogar kostenlos verfügbares Software-Paket, dass einen relativ einfachen Einstieg in die Welt der Spieleentwicklung bietet.

Es kommt letztendlich zu der erstaunlichen Erkenntnis, dass selbst fachfremde wenig technikaffine Probanden, die nicht durch das Wissen von anderen Applikationen beeinflusst wurden dazu neigen eine Steuerung zu verlangen oder zu bevorzugen, die sich an der Benutzung eines Gamepads orientiert. Es zeigt sich, dass es sehr schwer ist mit den Kompromissen und Einschränkungen der Tablets dieses Konzept grundlegend zu revolutionieren, aber durchaus noch viel Raum für Evolution vorhanden ist.

7.1 Ausblick

Im Laufe dieser Arbeit wurden Methoden der Usability mit einem noch sehr jungen und neuen Feld kombiniert, dem des Gamings und der 3D-Anwendungen auf Tablets. Es zeigt sich, dass hier noch sehr viel Raum für Innovationen vorhanden ist und dass es sinnvoll ist dabei auch nicht den Blick in die Vergangenheit und ihre Erkenntnisse zu scheuen. Auch wenn eventuell die Emulation von Gamepad-Hardware der einzige Weg sein könnte, ein konsistentes für die meisten Menschen verständliches Steuerungssystem auf diesem Systemen für 3D-Welten umzusetzen, so hat die Entwicklung des Prototyps doch gezeigt, dass noch viel Raum für Evolution vorhanden ist. Bereits etablierte Prinzipien können vielleicht nicht gänzlich revolutioniert, aber mit Sicherheit noch weiter verbessert werden. Ein Touchscreen als solches ist ein unglaublich vielseitig einsetzbares Instrument, dass es sowohl ermöglicht Abläufe der echten Welt metaphorisch in Interfaces für Programme

umzusetzen, Gesten zu erfassen als auch dynamisch auf Gegebenheiten in einem Softwareprodukt zu reagieren - und das in einem Ausmaß wie es kein hardwarebasierter Knopf oder Schalter jemals könnte. Hier ist noch viel Raum erkennbar für zukünftige Verbesserungen und Innovationen ,die lediglich durch die Kreativität von Designern und Programmierern limitiert sein werden. Auch die integrierten Bewegungssensoren sind noch lange nicht ausgereizt. Neuere Geräte versprechen mittels Gyroskopen, die um ein vielfaches genauer sind als die Beschleunigungssensoren der hier behandelten Geräte wieder ganze neue Möglichkeiten zu bieten. Zusammen mit integrierten Kameras lassen sich zudem noch weitere Steuerungskanäle erschließen, die in das Feld der augmented Reality münden. Die Zukunft wird voll von derartigen Geräten sein, die unseren Alltag und den Umgang mit Elektronik nach und nach revolutionieren werden - die Tage von Maus und Tastatur sind vielleicht auf lange Sicht gezählt. Man darf gespannt sein auf neue Entwicklungen auf diesem rapide wachsenden Feld und - sowohl von Industrie als auch Wissenschaft - noch viele Innovationen und neue Denkansätze erwarten.

A Anhang

A.1 Modifikation des ErgoNorm-Fragebogens mit Erklärung

Item	Kommentar, Bewertung	neues Item
Enthält das Programm alle für Ihre Aufgabe benötigten Funktionen?	OK	
Müssen Sie Eingaben oder Dialogschritte machen, die eigentlich überflüssig wären?	OK	
Ist es Ihnen möglich, das wiederholte Eingeben von Daten oder Texten zu vereinfachen?	OK	
Finden Sie, dass der erforderliche Aufwand für Ihr Arbeitsergebnis jeweils angemessen ist?	leichte Umformulierung	Finden Sie, dass der erforderliche Aufwand für das Erfüllen der Aufgabe angemessen ist?
Haben Sie das Gefühl, dass Sie Arbeiten machen müssen, die besser das Programm erledigen sollte?	OK	
Müssen Sie Werte und Texte eingeben, die der Computer eigentlich wissen könnte?	Frage passt nicht zu Games, Spielmechanik entschiedet, was dem Nutzer gezeigt wird.	Gestrichen
Müssen Sie sich mit Umwegen oder Tricks behelfen, um Ihre Arbeitsergebnisse so zu erzielen, wie Sie diese haben möchten?	OK	
Finden Sie in dem Programm Hilfetexte, die Ihnen auch tatsächlich weiterhelfen?	OK, aber spezifizieren, da es nicht nur um Texte geht.	Finden Sie in dem Programm Hilfsfunktionen, die Ihnen auch tatsächlich weiterhelfen?
Passt das Programm zu Ihren Formularen und bisherigen Formaten?	Frage passt nicht zu Games - Anpassung an Games nötig	Passt der Umgang mit dem Programm zu anderen von Ihnen getesteten Spielen ?
	Zusatzfrage Tablets	Werden wichtige Funktionen des Programms beim Benutzen durch Ihre Hände verdeckt?

Tab. A.1: Fragensatz Aufgabenangemessenheit

Item	Kommentar, Bewertung	neues Item
Sind die Informationen, die zur Erledigung der Aufgabe notwendig sind, auf dem Bildschirm übersichtlich verfügbar?	OK	
Können Sie bei der Arbeit mit dem Programm erkennen, welche Eingabe als nächstes von Ihnen erwartet wird?	OK	
Sind die Meldungen des Systems für Sie immer verständlich?	OK	
Werden Sie vor Aktionen, die nicht rückgängig gemacht werden können, von der Software gewarnt?	OK	
Hilft Ihnen die Hilfefunktion wirklich weiter, wenn einmal ein Dialogschritt oder Menüpunkt nicht ganz klar ist?	leichte Modifikation für Games, da Dialogschritte und Menüs oft nicht so präsent	Helfen Ihnen die Hilfefunktionen wirklich weiter, wenn einmal etwas im Programm nicht ganz klar ist?
Müssen Sie oft Kollegen oder ein Handbuch konsultieren, um weiterarbeiten zu können?	OK	
	Zusatzfrage	Ist zu jeder Zeit klar, was sie als nächstes in der Spielwelt erledigen sollen?
	Zusatzfrage	Bietet Ihnen das Programm im richtigen Moment die passenden Funktionen an?

Tab. A.2: Fragensatz Selbstbeschreibungsfähigkeit

Item	Kommentar, Bewertung	neues Item
Können Sie Ihre Arbeitsschritte in der Reihenfolge erledigen, die Ihnen am sinnvollsten erscheint?	OK	
Macht das Programm manchmal etwas, ohne dass Sie es zu dem Zeitpunkt wollen?	Ist zwiespältig zu betrachten, da die Dramaturgie Überraschungen erfordern kann, also Frage spezifizieren.	Macht ihr Avatar / Spielfigur / Kamera manchmal etwas, ohne dass Sie es zu dem Zeitpunkt wollen?
Können Sie bei Bedarf eine Aufgabe unterbrechen und später wieder fortsetzen, ohne alles neu eingeben zu müssen?	leichte Anpassung an den Gaming-Kontext	Können Sie bei Bedarf Ihren Fortschritt speichern und später wieder fortsetzen, ohne wieder neu anfangen zu müssen?
Können Sie einen Arbeitsschritt wieder zurücknehmen, wenn es für Ihre Aufgabenerledigung zweckmäßig ist?	in den meisten Games irrelevant, bzw. durch die Frage nach der Save-Funktion abgedeckt.	gestrichen
Fühlen Sie sich in Ihrem Arbeitstempo durch das Programm manchmal gebremst, z.B. durch zu lange Wartezeiten?	leichte Anpassung an den Gaming Kontext	Fühlen Sie sich in Ihrem Spielerlebnis durch das Programm manchmal unterbrochen, z.B. durch zu lange Ladezeiten?

Tab. A.3: Fragensatz Steuerbarkeit

Item	Kommentar, Bewertung	neues Item
Finden Sie Menüpunkte oder Funktionen dort, wo sie Ihrer Meinung nach auch sein sollten?	leichte Anpassung an Gaming Termini	Finden Sie Kontrollen, Buttons oder Funktionen dort, wo sie Ihrer Meinung nach auch sein sollten?
Sind Sie sich bei Wartezeiten immer noch sicher, ob das Programm weiter- arbeitet?	OK	
Sind Sie manchmal überrascht, wie das Programm auf Ihre Eingabe reagiert?	OK	
	Zusatzfrage	Erkennen und Verstehen Sie die Elemente der Steuerung ohne Ausprobieren oder Hilfsfunktionen?
	Zusatzfrage speziell Tablets	Werden bekannte Touch-Gesten (Pinch to Zoom etc.) erwartungsgemäß in dem Programm eingesetzt?
	Zusatzfrage speziell Tablets	Reagiert das Programm für Sie erwartungsgemäß auf das Bewegen des Tablets?

Tab. A.4: Fragensatz Erwartungskonformität

Item	Kommentar, Bewertung	neues Item
Bekommen Sie bei fehlerhaften Eingaben Korrekturhinweise?	OK	
Können Sie die Folgen einer fehlerhaften Eingabe mit geringem Aufwand beheben?	Anpassung an Gaming-Termini	Können Sie die Folgen eines Fehlers im Spielablauf mit geringem Aufwand beheben?
Arbeitet das Programm während der Ausführung Ihrer Aufgabe immer stabil und zuverlässig?	OK	

Tab. A.5: Fragensatz Fehlertoleranz

Item	Kommentar, Bewertung	neues Item
Können Sie am Computer alles so einstellen, dass Ihnen das Lesen und Arbeiten leichter fällt?	Anpassung an Aufgabenstellung	Können Sie die Steuerung so anpassen, dass Ihnen der Umgang damit leichter fällt?

Tab. A.6: Fragensatz Individualisierbarkeit

Item	Kommentar, Bewertung	neues Item
Ermöglicht Ihnen das Programm, auch einmal etwas gefahrlos auszuprobieren?	OK	

Tab. A.7: Fragensatz Lernförderlichkeit

A.2 Ausgeteilter Fragebogen für die Usability-Untersuchungen

Usability-Fragebogen für 3D-iPad Anwendungen

(basierend auf ErgoNorm Fragebogen der Bundesanstalt für Arbeitsschutz und Arbeitsmedizin, Dortmund 2000)

Hinweise zur Auswertung:

Befragung-Nr. und Proband-Nr.:

Ausführungszeit:

Zusätzliche Notizen:

Ihre Aufgabe:

Auf dem iPad befindet sich eine modifizierte Version der bekannten Grafikdemo Epic Citadel. Die Steuerung und Benutzeroberfläche der Demo wurden angepasst. Ihre Aufgabe ist es ein Artefakt (eine dunkle schwarze Scheibe) in der Spielwelt in der Nähe einiger Zelte zu finden, es aufzusammeln und bei der Statue in der Kirche wieder abzugeben. Sie sind nicht unter Zeitdruck. Die Aufgabe ist beendet, sobald der Gegenstand abgegeben wurde.

Bitte kreisen sie die für sie passenden Antworten (kursiv gedruckt) ein oder markieren diese mit einem Kreuz. Falls sie den Eindruck haben, sie können eine Frage nicht beantworten, da sie nicht relevant oder passend für die gestellte Aufgabe war markieren sie bitte „Frage trifft nicht zu“. Im Falle bestimmter Antworten werden sie gebeten das Problem genauer zu spezifizieren. Wenn sie ein auftretendes Problem besonders dramatisch empfunden haben, können sie zusätzlich die Aussage „ich empfinde dies als sehr störend“ markieren.

Aufgabenangemessenheit

Ein Computerprogramm ist aufgabenangemessen, wenn es zur Erledigung Ihrer konkreten Tätigkeit brauchbar ist. "Brauchbar" bedeutet, dass alle Tätigkeiten, die Sie erledigen müssen, vom Programm unterstützt werden. Es ist Ihnen dabei wirklich eine Hilfe und kein nötiges Übel, das Ihre Arbeit in manchen Situationen eher erschwert oder umständlicher macht.

Enthält das Programm alle für Ihre Aufgabe benötigten Funktionen?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Bitte benennen Sie den Arbeitsschritt, bei dem Sie sich wünschen würden, dass das Programm "mehr kann" als gerade möglich ist.

_____

_____

_____

ich empfinde dies als sehr störend

Müssen Sie Eingaben oder Dialogschritte machen, die eigentlich überflüssig wären?

Ja

Nein

Frage trifft nicht zu

wenn ja:
Bitte benennen Sie die in ihren Augen überflüssigen Eingaben und Dialogschritte.

_____

_____

_____

ich empfinde dies als sehr störend

Ist es Ihnen möglich, das wiederholte Eingeben von Daten oder Texten zu vereinfachen?

Ja

Nein

Frage trifft nicht zu

wenn nein:
In welcher Situation würden Sie sich wünschen, dass Sie nicht so oft dasselbe eingeben müssten?

_____

_____

_____

ich empfinde dies als sehr störend

Finden Sie, dass der erforderliche Aufwand für das Erfüllen der Aufgabe angemessen ist?

Ja

Nein

Frage trifft nicht zu

wenn nein:
In welcher Situation haben Sie schon mal gedacht "Das könnte man auch mit weniger Aufwand bewerkstelligen."

_____

_____

_____

ich empfinde dies als sehr störend

Haben Sie das Gefühl, dass Sie Arbeiten machen müssen, die besser das Programm erledigen sollte?

Ja

Nein

Frage trifft nicht zu

wenn ja:
Bitte benennen sie diese Arbeiten

_____

_____

_____

ich empfinde dies als sehr störend

Müssen Sie sich mit Umwegen oder Tricks behelfen, um Ihre Arbeitsergebnisse so zu erzielen, wie Sie diese haben möchten?

Ja

Nein

Frage trifft nicht zu

wenn ja:
Beschreiben Sie bitte die Situationen, in denen Sie das Gefühl haben, umständlich "tricksen" zu müssen, um Ihr Arbeitsergebnis zu erreichen.

_____

_____

_____

ich empfinde dies als sehr störend

Finden Sie in dem Programm Hilfsfunktionen, die Ihnen auch tatsächlich weiterhelfen?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Benennen Sie die Situationen, in denen Sie diese Funktionen nicht weitergebracht haben.

_____

_____

_____

ich empfinde dies als sehr störend

Passt der Umgang mit dem Programm zu anderen von Ihnen getesteten Spielen ?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Benennen Sie die Tätigkeit, bei der das Programm Erfahrungen und Erwartungen von anderen vergleichbaren Programmen widersprochen hat.

_____

_____

_____

ich empfinde dies als sehr störend

Werden wichtige Funktionen des Programms beim Benutzen durch Ihre Hände verdeckt?

Ja

Nein

Frage trifft nicht zu

wenn ja:
Benennen Sie die Situationen, in denen das Problem aufgetreten ist.

_____

_____

_____

ich empfinde dies als sehr störend

Selbstbeschreibungsfähigkeit

Eine Computerprogramm ist selbstbeschreibungsfähig, wenn Sie jederzeit informiert sind, was der Computer gerade macht und was er als nächstes von Ihnen als Eingabe oder Reaktion erwartet. Dies bedeutet unter anderem, dass Sie alle Rückmeldungen verstehen können, immer wissen, wo Sie als nächstes etwas eingeben müssen und sich jederzeit klar über die Folgen sind, die eine Eingabe von Ihnen haben wird.

Sind die Informationen, die zur Erledigung der Aufgabe notwendig sind, auf dem Bildschirm übersichtlich verfügbar?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Nennen Sie bitte die Informationen, die Sie benötigen, aber nicht "auf einen Blick" zur Verfügung stehen.

_____

_____

_____

ich empfinde dies als sehr störend

Können Sie bei der Arbeit mit dem Programm erkennen, was als nächstes von Ihnen erwartet wird?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Schildern Sie bitte kurz die Situationen, in der Sie unsicher waren, was als nächstes zu tun ist.

_____

_____

_____

ich empfinde dies als sehr störend

Sind die Meldungen des Systems für Sie immer verständlich?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Nennen Sie die Situationen, in denen Ihnen unverständliche Meldungen aufgefallen sind.

_____

_____

_____

ich empfinde dies als sehr störend

Werden Sie vor Aktionen, die nicht rückgängig gemacht werden können, von der Software gewarnt?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Bitte benennen Sie Situationen, in denen Sie keine Warnung erhalten haben.

_____

_____

_____

ich empfinde dies als sehr störend

Helfen Ihnen die Hilfefunktionen wirklich weiter, wenn einmal etwas im Programm nicht ganz klar ist?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Beschreiben Sie die Situationen, in denen die Hilfe nicht verständlich oder hilfreich waren.

_____

_____

_____

ich empfinde dies als sehr störend

Müssen Sie oft Kollegen oder ein Handbuch konsultieren, um weiterarbeiten zu können?

Ja

Nein

Frage trifft nicht zu

wenn ja:
Nennen Sie bitte Situationen, in denen Sie auf die Hilfe von Kollegen oder eines Handbuchs angewiesen waren.

_____

_____

_____

ich empfinde dies als sehr störend

Ist zu jeder Zeit klar, was sie als nächstes in der Spielwelt erledigen sollen?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Nennen sie die Situation in der das unklar war.

_____

_____

_____

ich empfinde dies als sehr störend

Bietet Ihnen das Programm im richtigen Moment die passenden Funktionen an?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Nennen sie die Situation in der das Programm falsche oder keine Funktionen angeboten hat.

_____

_____

_____

ich empfinde dies als sehr störend

Steuerbarkeit

Eine Computerprogramm ist steuerbar, wenn Sie als Benutzer die Abfolge der Arbeitsschritte weitgehend selbst bestimmen können. Wenn es die Arbeitssituation erfordert, können Sie die Arbeit am Computer unterbrechen und diese dann ohne Verlust der bis dahin erreichten Arbeitsergebnisse wieder aufnehmen.

Können Sie Ihre Arbeitsschritte in der Reihenfolge erledigen, die Ihnen am sinnvollsten erscheint?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Nennen Sie bitte Arbeitsschritte, bei denen Ihnen eine andere Reihenfolge sinnvoller erscheinen würde.

_____

_____

_____

ich empfinde dies als sehr störend

Macht ihr Avatar / Spielfigur / Kamera manchmal etwas, ohne dass Sie es zu dem Zeitpunkt wollen?

Ja

Nein

Frage trifft nicht zu

wenn ja:
Nennen Sie die Situation in der das Fehlverhalten auftrat.

_____

_____

_____

ich empfinde dies als sehr störend

Können Sie bei Bedarf Ihren Fortschritt speichern und später wieder fortsetzen, ohne wieder neu anfangen zu müssen?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Nennen Sie die Situation in der das Problem auftrat.

_____

_____

_____

ich empfinde dies als sehr störend

Fühlen Sie sich in Ihrem Spielerlebnis durch das Programm manchmal unterbrochen, z.B. durch zu lange Ladezeiten?

Ja

Nein

Frage trifft nicht zu

wenn ja:
Nennen Sie die Situation in der das Fehlverhalten auftrat.

_____

_____

_____

ich empfinde dies als sehr störend

Erwartungskonformität

Ein Computerprogramm ist erwartungskonform, wenn Sie bei der Arbeit mit dem Computer keine "Überraschungsmomente" erleben. Solche Momente können zum Beispiel sein, dass sich eine Funktion an einer ganz anderen Stelle im Menü befindet, als Sie gedacht hätten oder dass Aufgaben nicht, wie Sie es gewohnt sind, ausgeführt werden können.

Finden Sie Kontrollen, Buttons oder Funktionen dort, wo sie Ihrer Meinung nach auch sein sollten?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Nennen Sie bitte das fehlplatzierte Kontrollelement, und in welcher Situation sie es als solches empfanden.

_____

_____

_____

ich empfinde dies als sehr störend

Sind Sie sich bei Wartezeiten immer noch sicher, ob das Programm weiter arbeitet?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Nennen Sie bitte die Situationen, in denen Sie sich nicht sicher sind, ob das Programm noch arbeitet, z.B., wenn das Programm sehr lange benötigt, um Daten zu speichern.

_____

_____

_____

ich empfinde dies als sehr störend

Sind Sie manchmal überrascht, wie das Programm auf Ihre Eingabe reagiert?

Ja

Nein

Frage trifft nicht zu

wenn ja:
Beschreiben Sie die Situationen, in denen sie über die Reaktionen des Systems erstaunt sind.

_____

_____

_____

ich empfinde dies als sehr störend

Erkennen und Verstehen Sie die Elemente der Steuerung ohne Ausprobieren oder Hilfsfunktionen?

Ja

Nein

Frage trifft nicht zu

wenn ja:
Beschreiben Sie die Steuerungselemente, die sie nicht auf Anhieb verstanden haben.

_____

_____

_____

ich empfinde dies als sehr störend

Werden bekannte Touch-Gesten (Pinch to Zoom etc.) erwartungsgemäß in dem Programm eingesetzt?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Beschreiben sie Situationen in denen sie gerne auf bekannte Gesten zurückgegriffen hätten.

_____

_____

_____

ich empfinde dies als sehr störend

Reagiert das Programm für Sie erwartungsgemäß auf das Bewegen des Tablets?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Beschreiben sie Situationen in denen sie erwartet hätten, dass das Gerät anders auf Bewegungen reagiert.

_____

_____

_____

ich empfinde dies als sehr störend

Fehlertoleranz

Ein Computerprogramm ist fehlertolerant, wenn Sie ihr Arbeitsergebnis trotz fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand erreichen können. Dies bedeutet, dass es durchaus erlaubt sein muss, sich zu vertippen oder einen falschen Arbeitsschritt zu machen, ohne dass das Programm gleich abstürzt, oder Sie den Fehler nur mit Mühe wieder gut machen können. Außerdem sollte das Programm Sie darauf aufmerksam machen, wenn es einen Fehler bemerkt und Ihnen mögliche Korrekturhinweise liefern.

Bekommen Sie bei fehlerhaften Eingaben Korrekturhinweise?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Nennen Sie bitte Situationen, in denen Sie sich vielleicht wünschen würden, dass das Programm Ihnen einen Vorschlag für eine richtige Eingabe macht.

_____

_____

_____

ich empfinde dies als sehr störend

Können Sie die Folgen eines Fehlers im Spielablauf mit geringem Aufwand beheben?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Nennen Sie bitte Situation, in der das Problem auftrat.

_____

_____

_____

ich empfinde dies als sehr störend

Arbeitet das Programm während der Ausführung Ihrer Aufgabe immer stabil und zuverlässig?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Nennen Sie die Situationen, in denen Sie der Software nicht trauen oder Sie einen "Absturz" befürchten.

_____

_____

_____

ich empfinde dies als sehr störend

Individualisierbarkeit

Ein Computerprogramm ist individualisierbar, wenn Sie Einstellungen des Programms an Ihre individuellen Bedürfnisse anpassen können.

Können Sie die Steuerung so anpassen, dass Ihnen der Umgang damit leichter fällt?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Nennen Sie bitte Anpassungsmöglichkeiten, die sie gesucht aber nicht gefunden haben.

_____

_____

_____

ich empfinde dies als sehr störend

Lernförderlichkeit

Ein Computerprogramm ist lernförderlich, wenn es Ihnen unter anderem ermöglicht, selbständig einfach mal "rumzuprobieren", ohne dass Sie Angst haben müssen etwas "kaputt" zu machen. Zusätzlich sollten Sie durch das Programm die für Sie relevanten Informationen erhalten, die Sie Ihrer Meinung nach benötigen, um das Programm besser zu verstehen.

Ermöglicht Ihnen das Programm, auch einmal etwas gefahrlos auszuprobieren?

Ja

Nein

Frage trifft nicht zu

wenn nein:
Beschreiben Sie bitte die "Strafen", die Sie von dem Programm durch "Rumprobieren" bekommen haben.

_____

_____

_____

ich empfinde dies als sehr störend

Der letzte Teil des Fragebogens ist für Ihre individuellen Anmerkungen reserviert. Hier ist Platz für weitere Kritik an dem Computerprogramm oder für die Probleme, die Sie bei Beantwortung der Fragen nicht losgeworden sind.

A.3 Ergebnisse der ersten Befragung (Iteration 0)

Aufgabenangemessenheit

Frage	J	N	NZ	BS	Kommentare
Enthält das Programm alle für Ihre Aufgabe benötigten Funktionen?	3	2			„Wußte nicht wie ich die Fässer bewege“, "Hilfetexte bei den Fässern", "Vielleicht schneller laufen"
Müssen Sie Eingaben oder Dialogschritte machen, die eigentlich überflüssig wären?	1	2	2		„Es wäre einfacher über die Fässer zu springen“
Ist es Ihnen möglich, das wiederholte Eingeben von Daten oder Texten zu vereinfachen?		1	4		
Finden Sie, dass der erforderliche Aufwand für das Erfüllen der Aufgabe angemessen ist?	5				
Haben Sie das Gefühl, dass Sie Arbeiten machen müssen, die besser das Programm erledigen sollte?	1	4			„Man sollte auf einen Zielort klicken können und die Figur automatisch dort hin laufen.“
Müssen Sie sich mit Umwegen oder Tricks behelfen, um Ihre Arbeitsergebnisse so zu erzielen, wie Sie diese haben möchten?	2	2			„Fässer müssen aus dem Weg geräumt werden“, „Ich habe mich versucht an den Fässern vorbei zu mogeln“
Finden Sie in dem Programm Hilfsfunktionen, die Ihnen auch tatsächlich weiterhelfen?	3	1	1		„Die Verwendung der Lauf- bzw. Sichtpunkte“, „Die Fässer bewegen“
Passt der Umgang mit dem Programm zu anderen von Ihnen getesteten Spielen ?	2	2	1		„Steuerung per Hand ungewohnt“
Werden wichtige Funktionen des Programms beim Benutzen durch Ihre Hände verdeckt?	1	4			„Ich habe die Bewegungssteuerung versehentlich verschoben“, "Das Quest Radar wird häufig verdeckt"

Legende: J = Ja, N = Nein, NZ = Nicht zutreffend, BS = besonders schlimm

Fragebogenauswertung - Iteration 0

Selbstbeschreibungsfähigkeit

Frage	J	N	NZ	BS	Kommentare
Sind die Informationen, die zur Erledigung der Aufgabe notwendig sind, auf dem Bildschirm übersichtlich verfügbar?	2	3			„unklares Use-Icon“, „Die Benutzung der Kreise auf den Fässern“,“Das Laufen am Anfang, Die Funktionen mit dem Fass“
Können Sie bei der Arbeit mit dem Programm erkennen, was als nächstes von Ihnen erwartet wird?	4	1			„Die Fässer“, „Wie ich mit dem Artefakt interagiere“
Sind die Meldungen des Systems für Sie immer verständlich?	2		3		
Werden Sie vor Aktionen, die nicht rückgängig gemacht werden können, von der Software gewarnt?	1	1	3		„Bei den Fässern“
Helfen Ihnen die Hilfefunktionen wirklich weiter, wenn einmal etwas im Programm nicht ganz klar ist?	1	2	2		„Es gibt zu wenige“, „Es gab eine Hilfe?“
Müssen Sie oft Kollegen oder ein Handbuch konsultieren, um weiterarbeiten zu können?	2	3			„Nach ein wenig testen ging es auch alleine“
Ist zu jeder Zeit klar, was sie als nächstes in der Spielwelt erledigen sollen?	4	1			„Ich hatte meinen Quest vergessen weil ich lang brauchte die Fässer zu bewegen“
Bietet Ihnen das Programm im richtigen Moment die passenden Funktionen an?	5				„Manchmal“

Legende: J = Ja, N = Nein, NZ = Nicht zutreffend, BS = besonders schlimm

Fragebogenauswertung - Iteration 0

Steuerbarkeit

Frage	J	N	NZ	BS	Kommentare
Können Sie Ihre Arbeitsschritte in der Reihenfolge erledigen, die Ihnen am sinnvollsten erscheint?	4		1		
Macht ihr Avatar / Spielfigur / Kamera manchmal etwas, ohne dass Sie es zu dem Zeitpunkt wollen?	2	3			„Die Fässer bewegen“
Können Sie bei Bedarf Ihren Fortschritt speichern und später wieder fortsetzen, ohne wieder neu anfangen zu müssen?		1	4		„Es gab keine Speicherfunktion“
Fühlen Sie sich in Ihrem Spielerlebnis durch das Programm manchmal unterbrochen, z.B. durch zu lange Ladezeiten?		3	2		

Legende: J = Ja, N = Nein, NZ = Nicht zutreffend, BS = besonders schlimm

Fragebogenauswertung - Iteration 0

Erwartungskonformität

Frage	J	N	NZ	BS	Kommentare
Finden Sie Kontrollen, Buttons oder Funktionen dort, wo sie Ihrer Meinung nach auch sein sollten?	2	3			„Das Benutzen-Icon hat mich irritiert“, „Ich kenne solche Spiele wenig, deswegen weiß ich nicht wo die Kontrollen sein sollten“, „Das erste mal mit sowas gearbeitet, deswegen keine Ahnung gehabt wie man sich bewegt“
Sind Sie sich bei Wartezeiten immer noch sicher, ob das Programm weiter arbeitet?			5		
Sind Sie manchmal überrascht, wie das Programm auf Ihre Eingabe reagiert?	3	2			„Beim Fässer verschieben“, „Bei den Fässern“, „Springen klappt nicht bei den Fässerm, Blickrichtung wird geändert“
Erkennen und Verstehen Sie die Elemente der Steuerung ohne Ausprobieren oder Hilfsfunktionen?	3	2		x	„Ohne Ausprobieren für Neueinsteiger nicht ersichtlich“, „Zuerst habe ich den Wegweiser übersehen“, „die Steuerung, der Pfeil, die Kreise bei der Interaktion“
Werden bekannte Touch-Gesten (Pinch to Zoom etc.) erwartungsgemäß in dem Programm eingesetzt?	2	1	2		„Nur Laufen und Benutzen getestet“, „nicht getestet“, „Ich habe keine Ahnung von Touch-Gesten“
Reagiert das Programm für Sie erwartungsgemäß auf das Bewegen des Tablets?	2	1	2		„keine Bewegung fest festellt“

Fehlertoleranz

Frage	J	N	NZ	BS	Kommentare
Bekommen Sie bei fehlerhaften Eingaben Korrekturhinweise?		2	3		„Wieso funktionieren die Funktionen bei den Fässern nicht immer?“
Können Sie die Folgen eines Fehlers im Spielablauf mit geringem Aufwand beheben?	1		4		„Ich kam nicht in eine solche Situation“
Arbeitet das Programm während der Ausführung Ihrer Aufgabe immer stabil und zuverlässig?	4	1			„Ruckeln“

Legende: J = Ja, N = Nein, NZ = Nicht zutreffend, BS = besonders schlimm

Individualisierbarkeit

Frage	J	N	NZ	BS	Kommentare
Können Sie die Steuerung so anpassen, dass Ihnen der Umgang damit leichter fällt?	1	2	2		„Ich wusste nicht genau wie ich die Buttons bewegen kann“

Lernförderlichkeit

Frage	J	N	NZ	BS	Kommentare
Ermöglicht Ihnen das Programm, auch einmal etwas gefahrlos auszuprobieren?	2	1	1		„Ich habe nicht mehr als gefordert versucht“

Zusätzliche Kommentare:

„Textliche Beschreibung der Icons fände ich hilfreich.“
„Ich hatte mir einen Rennen-Knopf gewünscht“
„Eine Minimap wäre hilfreich um Abkürzungen und weitere Dinge entdecken zu können.“
„Ich habe die Kirche nicht sofort als solche erkannt.“

Legende: J = Ja, N = Nein, NZ = Nicht zutreffend, BS = besonders schlimm

A.4 Ergebnisse der ersten Befragung (Iteration 1)

Aufgabenangemessenheit

Frage	J	N	NZ	BS	Kommentare
Enthält das Programm alle für Ihre Aufgabe benötigten Funktionen?	3	1			„Eine durchsichtige Karte oder einen genaueren Richtungspfeil“, "Entfernungsangabe verändert sich manchmal nicht"
Müssen Sie Eingaben oder Dialogschritte machen, die eigentlich überflüssig wären?	1	3			„Einige Probleme mit der Ausrichtung, fühlte sich an wie ein Flugsimulator (kann aber auch an der ungewohnten Steuerung liegen)“
Ist es Ihnen möglich, das wiederholte Eingeben von Daten oder Texten zu vereinfachen?	1	1	2		„Die Fässer müssen zu genau anvisiert werden“
Finden Sie, dass der erforderliche Aufwand für das Erfüllen der Aufgabe angemessen ist?	5				
Haben Sie das Gefühl, dass Sie Arbeiten machen müssen, die besser das Programm erledigen sollte?		4			
Müssen Sie sich mit Umwegen oder Tricks behelfen, um Ihre Arbeitsergebnisse so zu erzielen, wie Sie diese haben möchten?		4			
Finden Sie in dem Programm Hilfsfunktionen, die Ihnen auch tatsächlich weiterhelfen?	2	2	1		„Richtungspfeil war etwas verwirrend“, "Wenn das Artefakt gefunden wurde, wurde neben der Statue Vibration eingeschaltet", "Ich hatte die Aufgabe nicht verstanden, Aufgabenstellung wurde im Spiel nicht mehr gezeigt"
Passt der Umgang mit dem Programm zu anderen von Ihnen getesteten Spielen ?	3	1	1		"Ich spiele keine PC-Spiele"
Werden wichtige Funktionen des Programms beim Benutzen durch Ihre Hände verdeckt?	3	2			„Die Zielführung könnte mittig unten/oben platziert sein“, „Da ich Rechtshänder bin wurde der Bildschirm von meinem rechten Arm verdeckt, wenn ich die Sicht links unten ändern wollte“, „Es gibt ja nur zwei Funktionen, Sicht und Bewegung“

Legende: J = Ja, N = Nein, NZ = Nicht zutreffend, BS = besonders schlimm

Selbstbeschreibungsfähigkeit

Frage	J	N	NZ	BS	Kommentare
Sind die Informationen, die zur Erledigung der Aufgabe notwendig sind, auf dem Bildschirm übersichtlich verfügbar?	2	3			„Minimap“, „Evtl. eine Art Questlog sollte man das Ziel seiner Quest vergessen“,“Habe nicht genug Erfahrung mit Spielen, es wurde nicht gesagt wo das Artefakt zu finden ist“
Können Sie bei der Arbeit mit dem Programm erkennen, was als nächstes von Ihnen erwartet wird?	3	1			„Eventuell nach Abschluss einer Teilaufgabe den nächsten Schritt zentriert ausgeben“
Sind die Meldungen des Systems für Sie immer verständlich?	2	1	1		„Die Schrift ist schwer lesbar“, „Schriftart schwer lesbar“
Werden Sie vor Aktionen, die nicht rückgängig gemacht werden können, von der Software gewarnt?			4		„Situation nicht erlebt“
Helfen Ihnen die Hilfefunktionen wirklich weiter, wenn einmal etwas im Programm nicht ganz klar ist?	2	2	1		„Sehr spät die Hilfsfunktion als solche erkannt“, "Habe Aufgabenstellung falsch verstanden"
Müssen Sie oft Kollegen oder ein Handbuch konsultieren, um weiterarbeiten zu können?	1	3	1		
Ist zu jeder Zeit klar, was sie als nächstes in der Spielwelt erledigen sollen?	4	1			
Bietet Ihnen das Programm im richtigen Moment die passenden Funktionen an?	4	1			„Nachlesen der Aufgabe“, „Es wird ja nicht viel verlangt, Steuerung war klar“

Legende: J = Ja, N = Nein, NZ = Nicht zutreffend, BS = besonders schlimm

Steuerbarkeit

Frage	J	N	NZ	BS	Kommentare
Können Sie Ihre Arbeitsschritte in der Reihenfolge erledigen, die Ihnen am sinnvollsten erscheint?	5				„Es soll ja nur gelaufen werden“
Macht ihr Avatar / Spielfigur / Kamera manchmal etwas, ohne dass Sie es zu dem Zeitpunkt wollen?		5			
Können Sie bei Bedarf Ihren Fortschritt speichern und später wieder fortsetzen, ohne wieder neu anfangen zu müssen?	1		4		„War nicht erforderlich“, „Bin aus versehen auf den falschen Knopf gekommen, das Spiel ließ sich ohne Probleme fortsetzen“
Fühlen Sie sich in Ihrem Spielerlebnis durch das Programm manchmal unterbrochen, z.B. durch zu lange Ladezeiten?	2	2	1		„ruckelte etwas“, "es ruckelte", „vielleicht am Anfang“

Legende: J = Ja, N = Nein, NZ = Nicht zutreffend, BS = besonders schlimm

Erwartungskonformität

Frage	J	N	NZ	BS	Kommentare
Finden Sie Kontrollen, Buttons oder Funktionen dort, wo sie Ihrer Meinung nach auch sein sollten?	1	4			„Zielführung ist am falschen Platz“, „Ich erwartete das der Richtungspfeil oben rechts auf dem Bildschirm ist“, „Sicht- und Laufjoystick zusammenfügen, so konnte man nur entweder oder.“, „Mehr wie ein Controller mit Daumen oder mit einem Stift statt mit den Fingern“
Sind Sie sich bei Wartezeiten immer noch sicher, ob das Programm weiter arbeitet?	1	1	3		„Habe nur das Laden des Programms erlebt“, „keine Ladebalken oder ähnliches“
Sind Sie manchmal überrascht, wie das Programm auf Ihre Eingabe reagiert?		4	1		„keine Eingaben nötig“
Erkennen und Verstehen Sie die Elemente der Steuerung ohne Ausprobieren oder Hilfsfunktionen?	4	1			
Werden bekannte Touch-Gesten (Pinch to Zoom etc.) erwartungsgemäß in dem Programm eingesetzt?	2	2	1		„Finger auseinander ziehen hat keinen Effekt gehabt“
Reagiert das Programm für Sie erwartungsgemäß auf das Bewegen des Tablets?	4	1			„Jump-to-move war etwas tricky, vor allem wenn man gleichzeitig laufen wollte“

Fehlertoleranz

Frage	J	N	NZ	BS	Kommentare
Bekommen Sie bei fehlerhaften Eingaben Korrekturhinweise?	2	1	2		
Können Sie die Folgen eines Fehlers im Spielablauf mit geringem Aufwand beheben?	3		2		
Arbeitet das Programm während der Ausführung Ihrer Aufgabe immer stabil und zuverlässig?	3	1			„ruckelte“

Legende: J = Ja, N = Nein, NZ = Nicht zutreffend, BS = besonders schlimm

Individualisierbarkeit

Frage	J	N	NZ	BS	Kommentare
Können Sie die Steuerung so anpassen, dass Ihnen der Umgang damit leichter fällt?	1	1	2		„man rutscht öfters zu weit vom Kreis(Stick)“

Lernförderlichkeit

Frage	J	N	NZ	BS	Kommentare
Ermöglicht Ihnen das Programm, auch einmal etwas gefahrlos auszuprobieren?	2	1	1		„Unsichtbare Wände verhindern einen Suizid“, „Obwohl es wenig zu Probieren / erkunden gibt“

Zusätzliche Kommentare:

„Ich hatte einige Probleme beim Ausrichten der Kamera, vielleicht würde ein Hinweis auf die Steuerung helfen“
„Fehler: Der Steuerpunkt für Bewegung hat sich verschoben! War auf einmal links am Rand“
„Nach einiger Zeit war die Steuerung unbequem, vielleicht kann man es mit einem Stift oder durch einfaches Drücken steuern“
„Finde die Sicht kompliziert und Blickwinkel zu klein. Mir gefällt es besser wenn man den Spielcharakter von oben sieht und mehr von der Umgebung sieht“
„Kleine Landkarte würde helfen sich zurecht zu finden“

Legende: J = Ja, N = Nein, NZ = Nicht zutreffend, BS = besonders schlimm

A.5 Literaturverzeichnis

[Bus09] Jason Busby. *Mastering Unreal Technology: A Beginner's Guide to Level Design in Unreal Engine 3*. Sams, 2009.

[Bus10] Jason Busby. *Mastering Unreal Technology, Volume III: Introduction to Unrealscript with Unreal Engine 3*. Sams, 2010.

[fAuA00] Bundesanstalt für Arbeitsschutz und Arbeitsmedizin. *Gebrauchstauglichkeit von Software ErgoNorm: Ein Verfahren zur Konformitätspruefung von Software auf der Grundlage von DIN EN ISO 9241 Teile 10 und 11*. 2000.

[Fed02] Melissa A. Federoff. Heuristics and usability guidelines for the creation and evaluation of fun in video games. Master's thesis, Indiana University, 2002.

[Fly06] John Flynt. *UnrealScript Game Programming All in One*. Thomson Learning, 2006.

[FPM09] Agnes F. Vandome Frederic P. Miller. *Gamepad*. Alphascript Publishing, 2009.

[iW11] iPhone Welt. Nova 2 - test. *iPhone Welt*, 02, 2011.

[Lew82] C. H. Lewis. Using the "thinking aloud"method in cognitive interface design. *Technical Report IBM RC-9265*, 1982.

[LR94] Clayton Lewis and John Rieman. *Task-Centered User Interface Design: A Practical Introduction*. 1993-1994.

[Lun97] Arnold Lund. Expert ratings of usability maxims. *Ergonomics in Design*, 1997.

[Nie94] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann, 1994.

[Pap01] Lothar Papula. *Mathematik für Ingenieure und Naturwissenschaftler Band 2, 10. Auflage*. vieweg, 2001.

[Ruh11] Thomas Ruhk. Shadow guardian - test. *Gamepro*, 01, 2011.

[Sta10] *Programmieren für iPhone und iPad: Der Einstieg in die App-Entwicklung für das iOS 4*. Dpunkt Verlag, 2010.

[Sto02] Sabine Stoessel. *Methoden des Testings im Usability Engineering*. X.media.press, 2002.

[Vir92] Robert A. Virzi. Refining the test phase of usability evaluation: How many subjects is enough? *Human Factors: The Journal of the Human Factors and Ergonomics Society*, Volume 34:457–468, 1992.

[Wie10] Georg Wieselsberger. Gratis-demo epic citadel - unreal engine 3 auf iphone & co. *Gamestar*, 10, 2010.

[Wix11] Daniel Wigdor Dennis Wixon. *Brave NUI World*. Morgan Kaufmann, 2011.

[Woo11] Patrick Woods. Rage: Mutant bash tv im test. *iPhone Welt*, 01, 2011.