4 Dos and Don'ts – Game-Design für Virtual Reality

Warum fühlt sich Virtual Reality immersiver an als ein regulärer Monitor? Ein wichtiger Punkt ist sicherlich das große Blickfeld und die dreidimensionale Darstellung. Noch wichtiger ist die natürliche Art und Weise der Eingabe. Dadurch, dass du dich im Spiel umsiehst, indem du dich einfach in der Realität umschaust, ist kein abstraktes Eingabegerät wie eine Maus oder eine Tastatur notwendig. Diese Art der Eingabe gibt dir das Gefühl, wirklich mittendrin und von der Welt umgeben zu sein. Diese natürliche Art der Eingabe gehört zu den vielen Dingen, die man hauptsächlich *unbewusst* wahrnimmt, wenn man in virtuelle Welten abtaucht. Um die Interaktion mit der virtuellen Welt noch natürlicher zu gestalten, bieten unter anderem HTC und Oculus Motion-Controller an, mit denen zum Beispiel virtuelle Gegenstände aufgehoben werden können. Je natürlicher die Interaktion, desto immersiver kann die Erfahrung werden.

Das Streben nach natürlichen Eingaben hat allerdings auch einen Nachteil: Je realistischer man mit der virtuellen Welt interagieren kann, desto mehr erwartet man unbewusst auch, dass sich alles wie in der echten Welt verhält. Bei einem Spiel mit Motion-Controllern erwartet der Spieler zum Beispiel, dass er jeden herumliegenden Gegenstand aufheben und herumwerfen kann. Als Entwickler eines VR-Spiels solltest du das beachten, da ein Nichterfüllen von Erwartungen die Immersion des Spielers schnell zerstören und zu Frustration führen kann. Es gibt zum Beispiel nichts Schlimmeres, als wenn du mit Motion-Controllern nach einer Türklinke greifst und nichts passiert. In diesem Moment wird der Spieler nämlich daran erinnert, wie statisch und unecht diese virtuelle Welt eigentlich ist. Die Lösung für dieses Problem ist allerdings nicht zwingend, die echte Welt in all ihren Details nachzubilden. Dieses Vorhaben würde wohl ein wenig zu umfangreich und groß sein. Ein guter Lösungsansatz ist, die Spielwelt so zu gestalten, dass der Spieler gar nicht auf die Idee kommt, die beispielhafte Tür öffnen zu wollen.

Dieses Beispiel zeigt einen der zahlreichen Gamedesign-Punkte, die du beim Entwickeln eines Virtual-Reality-Spiels beachten solltest, und genau darum geht es in diesem Kapitel. Auf den nächsten Seiten werden wir uns diverse Dinge anschauen, die in Virtual Reality besonders gut oder schlecht funktionieren. Einige der Beispiele wirst du auch selber ausprobieren können. Das Ziel dieses Kapitels ist, dass du gewisses Gespür dafür entwickelst, was in Virtual Reality funktioniert und was nicht.

4.1 Beispielprojekt

Auf der begleitenden Webseite habe ich dir ein Beispielprojekt hochgeladen, welches ein paar kleinere Beispiele enthält, auf die ich in diesem Kapitel verweisen werde. Das Projekt wurde mit Unity 2017.10f3 erstellt. Wenn du eine neuere Version verwendest, dauert der Importvorgang eventuell einen Moment länger.

Unity-Projekt mit Beispielen

http://www.vrspieleentwickeln.de/zusatz/

Auf der begleitenden Webseite findest du unter **Kapitel 4** ein Beispielprojekt in einem Zip-Archiv zum Download, auf das ich im Laufe dieses Kapitels hin und wieder eingehen werde.

Hinweise SteamVR, Oculus Rift

Damit du das Projekt mit *SteamVR* oder *Oculus Rift* verwenden kannst, musst du lediglich in den PROJECT SETTINGS das jeweilige *Virtual Reality SDK* aktivieren, wie du es zuvor bereits für das Quickstart-Projekt getan hast.

Hinweise GearVR, GoogleVR

Damit du das Projekt auf deiner *GearVR*- oder *GoogleVR*-Brille verwenden kannst, musst du zunächst, wie du es in den vergangenen Kapiteln bereits getan hast, die Plattform zu ANDROID wechseln. Anschließend musst du in den PROJECT SETTINGS das korrekte *Virtual Reality SDK* aktivieren wie zuvor schon beim Quickstart-Projekt. Um die Scenes dann ausprobieren zu können, musst du sie zu deinem aktuellen *Build* hinzufügen und nach ganz oben schieben oder alle anderen Scenes deaktivieren. Die Scene, die du testen willst, muss den Index *0* haben, damit sie automatisch startet.

Falls du eine GearVR und Unity 2017.10f3 verwendest, musst du außerdem wie im Quickstart-Kapitel beschrieben noch das *GearVR Android Manifest Bugfix*-Package und deine Osig-Datei importieren.

4.2 Immersion

Du hast bestimmt schon bemerkt, dass man in Virtual-Reality-Spielen ein sehr gutes Gespür für den Raum bekommt. Man kann Distanzen und Größen sehr gut einschätzen, wodurch sich kleine Räume auch tatsächlich klein anfühlen. Alles wirkt deutlich plastischer und auch realer auf dich, wenn man Spiele mit einer VR-Brille spielt. Für diesen Effekt ist es völlig unwichtig, dass du ja eigentlich ganz genau weißt, dass das alles gar nicht real ist. Auf einer unterbewussten Ebene kann unser Gehirn nicht anders, als die Reize so zu verarbei-

ten, als wäre es echt. Um unser Gehirn auszutricksen, braucht es nicht einmal fotorealistische Grafik, selbst mit unserer simplen Test-Scene in Low-Poly-Optik funktioniert es. Nicht umsonst ist *Minecraft* schon seit dem Erscheinen der ersten inoffiziellen VR-Version für die Java-Variante des Spiels eines der beliebtesten VR-Spiele gewesen. Diese inoffizielle Version war so beliebt, dass der Entwickler *Mojang* in Zusammenarbeit mit Oculus nun eine offizielle VR-Version für GearVR und Oculus Rift veröffentlicht hat. Die Rift-Version bietet sogar *Oculus Touch*-Unterstützung.

Bild 4.1 In der Unterwasser-Experience *the Blu* (SteamVR) begegnest du unter anderem einem sehr imposanten Wal. (Quelle: Wevr, Inc.)

Vor allem im Angesicht von überraschenden und Angst einflößenden Situationen denkt man nicht mehr darüber nach, dass es nicht real ist. Man duckt sich instinktiv, wenn die Flosse des riesigen Wals in *the Blu* auf das Deck schlägt (Bild 4.1). Oder man traut sich nur mit großer Überwindung, in *Richie's Plank Experience* einen einfachen Schritt nach vorne zu machen, obwohl wir eigentlich wissen, dass sich dort eine tiefe Häuserschlucht befindet (Bild 4.2). Der Zustand, in dem du das Gefühl hast, dass die virtuelle Welt für dich real ist, nennt sich *Presence* und ist das ultimative Ziel in Virtual Reality. Die aktuelle Hardware ist jedoch noch nicht in der Lage, diesen Zustand dauerhaft zu erreichen. Vielmehr sind es derzeit kurze Moment, in denen wir die Welt einfach akzeptieren. Diese kurzen Momente werden als *Micro-Presence* bezeichnet.

Bild 4.2 Eine einfache Aufgabe, die aber nicht wenige in die Knie zwingt: Hole ein Stück von dem Kuchen. (Quelle: Toast.gg)

Bedrohung und *Angst* sind übrigens die einfachsten Mittel, Immersion und Micro-Presence zu erzeugen, da unser Gehirn in diesen Fällen schneller auf den Trick anspringt, um sich zu schützen. Aus diesem Grund wird Virtual Reality auch bereits für verschiedene Arten von Ängsten in Form einer Konfrontationstherapie eingesetzt. Über die Zeit sind auch immer mehr Anwendungen zur Selbsttherapie erschienen, die teilweise einen spielerischen Ansatz bieten. Neben Höhenangst gibt es zum Beispiel auch Erfahrungen, die dich mit Spinnen und anderen Insekten konfrontieren. Eine davon ist *Don’t Let Go* (Bild 4.3). In dieser kurzen, aber eindrucksvollen Erfahrung musst du einfach nur zwei Tasten auf deiner Tastatur gedrückt halten, während um dich herum allerhand Dinge passieren. Das klingt natürlich einfacher, als es in der Praxis ist, denn das Spiel konfrontiert dich und deine Hände mit allerhand Situationen, in denen die meisten in der Realität schnell weg von dem Tisch wären.

Bild 4.3 In *Don't Let Go* (SteamVR) darfst du deine Tastatur nicht loslassen, egal was passiert! (Quelle: Skydome Studios)

Im Vergleich zu Erfahrungen über *Angst* und *Bedrohung* sind *emotionale* Erfahrungen in der Regel aufwendiger und schwieriger zu erstellen. Trotzdem stellen sie ein gutes Mittel dar, um den Spieler die reale Welt vergessen zu lassen und ein hohes Maß an Immersion zu erzeugen, wie man an dem Kurzfilm *Henry* von Oculus erleben kann. Obwohl der in Bild 4.4 zu sehende Igel nur rein virtuell ist, löst der Film echtes Mitleid in uns aus und wir wollen den kleinen Igel am liebsten in den Arm nehmen.

Bild 4.4 In dem preisgekrönten VR-Kurzfilm „Henry" (Oculus Rift) begleitest du den putzigen namensgebenden Igel an seinem Geburtstag. (Quelle: Oculus)

Die Königsdisziplin ist das Erzeugen von hoher Immersion ohne diese „Türöffner“, nur durch eine lebendige und attraktive Spielwelt.

Kleines Beispiel

In dem Beispielprojekt findest du unter *VRSpieleMitUnity/DosAndDonts/Scenes* eine kleine Test-Scene mit dem Namen *Immersion*, die auf dem Jugendzimmer basiert. Die Scene zeigt dir, dass du trotz der einfachen Optik und wenig plausiblen Situation irgendwo das Gefühl verspürst auszuweichen, wenn dir etwas zu nahekommt.

4.2.1 Immersion brechen

Immersion wird in der Regel gebrochen, wenn das Erlebte nicht deiner Erwartung entspricht. Die nicht greifbaren Türklinken aus der Einleitung sind ein Beispiel, ein weiteres sind auch große Höhen und Stürze: So hast du vielleicht Angst, solange du oben auf der Planke über einer Häuserschlucht stehst, sobald du aber tatsächlich stürzt, fühlt es sich komisch und vollkommen anders als erwartet an. Solange du oben stehst, stellst du dir unbewusst vor, wie sich der Sturz jetzt anfühlen würde, und dein Körper reagiert entsprechend darauf. Stürzt du nun in den virtuellen Abgrund, bemerkt dein Körper schnell, dass du eben nicht wirklich in die Tiefe stürzt. Sobald du unbewusst zu viele widersprüchliche Signale wahrnimmst, bist du dann schnell wieder in der Realität. Dieser Moment tritt jedoch bei einigen Menschen früher und bei anderen später ein. Deshalb kann man manchmal Spieler beobachten, die versuchen, sich noch im letzten Moment in vermeintliche Sicherheit zu bringen.

Beim Entwickeln deines eigenen Spiels solltest du das Brechen der Immersion so gut wie möglich vermeiden. Es ist zwar reizvoll, den Spieler vor einen tiefen Abgrund zu stellen, um ihn einzuschüchtern, den eigentlichen Sturz solltest du aber so möglichst vermeiden. Fällt der „Trick“ auf, ist der Spaß daran nämlich häufig vorbei.

Immersion ist natürlich immer auch davon abhängig, ob sich der Spieler darauf einlässt. Wenn man solche Erfahrungen von Anfang an mit der Einstellung angeht, dass es sich nicht echt anfühlen kann und man auch keinen Spaß haben wird, ist die Chance, dass man ein immersives Erlebnis hat, auch deutlich geringer.

4.3 Positional Tracking

Der englische Ausdruck *Positional Tracking* beschreibt das elektronische Erkennen bzw. Verfolgen eines Gegenstandes im Raum. Im Virtual-Reality-Kontext bezieht sich dieser Begriff in der Regel auf die Positionserkennung des Headsets. Unterstützt ein Headset Positional Tracking, kann es also erkennen, wenn du dich im echten Leben bewegst. Das kann

ein Lehnen oder können auch ganze Schritte sein. Deshalb gilt: Egal ob dein Spiel darauf ausgelegt ist, im Sitzen oder Stehen gespielt zu werden, *Positional Tracking* ist ein wichtiges Feature. Du solltest es bei einer gewöhnlichen Echtzeit-3D-Anwendung niemals deaktivieren. *Positional Tracking* ist ein wichtiges Element, um die Immersion zu steigern, und das sowohl auf einer bewussten als auch einer unbewussten Ebene.

Bild 4.5 Auch in Cockpit-Spielen wie Elite Dangerous ist Positional Tracking wichtig (SteamVR, Oculus). (Quelle: Frontier Developments plc.)

Wenn du zum Beispiel in einem Cockpit sitzt und dich nach vorne lehnen kannst, um alle Bedienelemente im Detail anzusehen, ist es ganz klar immersiver, als wenn die Position fest ist. Ohne *Positional Tracking* würdest du dich im echten Leben nach vorne lehnen und die virtuelle Perspektive würde sich nicht verändern. Aber auch bei Spielen, die nicht in einem Cockpit sind, macht es Sinn: Egal ob im Stehen oder im Sitzen, wenn du dich ein wenig zur Seite lehnen kannst, um heimlich um eine Ecke zu schauen, erhöht es klar die Immersion des Spiels.

Positional Tracking trägt aber auch *unbewusst* zur Immersion bei und verringert die Wahrscheinlichkeit von *Simulator Sickness*. Denn selbst wenn du dich nicht bewusst bewegst und deinen Kopf vermeintlich stillhältst, macht dein Kopf noch viele kleine Bewegungen, die du nur unbewusst wahrnimmst. Deaktivierst du das *Positional Tracking*, fehlen diese minimalen Bewegungen in der virtuellen Welt. Deinem Gehirn fällt das Fehlen dieser minimalen Bewegungen auf und je nach Empfindlichkeit kann dies bereits zu *Simulator Sickness* führen.

Hier kann man gut den Vergleich zu 3D-Filmen ziehen: Hier verändert sich die Perspektive auf die 3D-Szene ebenfalls nicht, wenn du deinen Kopf bewegst. Diese fehlende Änderung in der Perspektive ist einer der Gründe, warum einige Menschen bei 3D-Filmen Kopfschmerzen und Übelkeit erleiden können.

4.3.1 Mobile VR-Brillen

Bei mobilen VR-Brillen gibt es jedoch ein Problem: Hier gibt es meist keine Kamera und keinen Sensor, welcher die Position im Raum erkennen kann, da dieser die Portabilität einschränken würde. Einige Firmen arbeiten deshalb an einer *Inside-Out*-Lösung, wobei kein externer Sensor, sondern die Smartphone-Kamera verwendet wird, um die Position im Raum zu ermitteln. Im Smartphone-Bereich gibt es jedoch noch keine ausgereifte Lösung, die bereits einsatzbereit ist.[1]

Um trotzdem ein bestmögliches Erlebnis bieten zu können, helfen die Entwickler der VR-Brillen künstlich nach und simulieren über ein sogenanntes *Head-Neck-Model* zumindest eine natürliche Kopfbewegung, während du dich in VR umsiehst. Im Gegensatz zum *Eye-Ball-Model*, welches bei den meisten Nicht-VR-Videospielen verwendet wird, dreht sich die Kamera dann nicht um die eigene Achse, sondern sie bewegt sich in einer Kreisbahn um einen simulierten Nacken (siehe Bild 4.6). Natürlich ist diese Methode keine wirkliche Alternative zu echtem *Positional Tracking*, aber es ist trotzdem besser, als vollkommen auf Kopfbewegungen zu verzichten.

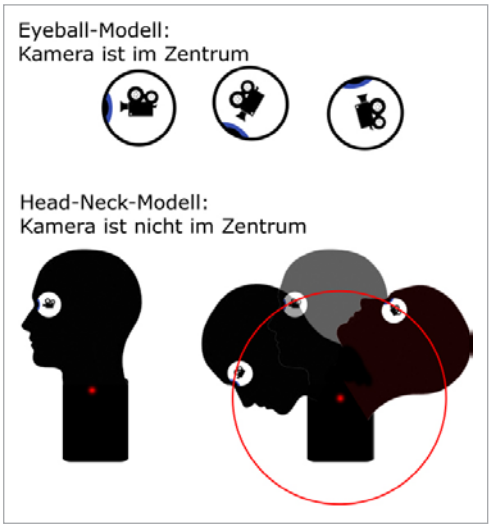

Bild 4.6 Die Position der Kamera in den beiden Modellen im Vergleich

[1] Stand 2017

4.4 Locomotion

Locomotion beschreibt die Art und Weise, mit der du dich in der virtuellen Welt fortbewegst. Am bekanntesten ist die klassische Tastatur bzw. Gamepad Locomotion, bei der man eine Taste oder einen Thumbstick bedient und die Spielfigur losläuft. In Virtual Reality ist diese *klassische Locomotion* jedoch häufig nicht wie in Nicht-VR-Spielen umsetzbar. Findige VR-Entwickler haben sich deshalb alternative Locomotion-Methoden ausgedacht, welche einige Probleme lösen, dafür jedoch häufig mit anderen Problemen und Einschränkungen behaftet sind. Es gibt jedoch noch keine Standard-Fortbewegungsart, die für jedes VR-Spiel optimal verwendet werden kann.

In diesem Abschnitt werden wir uns die gängigsten Locomotion-Methoden mit ihren Vor- und Nachteilen ansehen. Welche du für deine Projekte idealerweise verwendest, hängt von dir, deiner Zielgruppe und natürlich deinem Spielprinzip ab.

4.4.1 Klassische Locomotion

Wie schon erwähnt, ist das die typische Art und Weise, mit der du dich in den meisten Videospielen fortbewegst. Du drückst eine Taste oder bewegst einen Thumbstick und die Spielfigur läuft los. In vielen Spielen wird heutzutage auch *Momentum* berücksichtigt: Läufst du aus dem Stand los, muss die Spielfigur erst einmal beschleunigen, und lässt du die Taste los, braucht die Spielfigur noch einen Moment, um abzubremsen und stehen zu bleiben.

In den Anfängen der aktuellen Virtual-Reality-Revolution wurde ausschließlich diese klassische Locomotion verwendet. Über die Maus oder einen Thumbstick konntest du dich zusätzlich zum Umsehen mit dem Headset drehen. Lediglich das Schauen nach oben oder unten wurde auf das Headset beschränkt. Mit einem VR-Headset auf dem Kopf würde sich das nämlich so anfühlen, als würde die Welt um dich herum umkippen. In einem Flug- oder Raumschiffspiel macht dies wohl Sinn, wenn du aber zu Fuß unterwegs bist, sorgt das eher für Verwirrung, Unwohlsein und Übelkeit. Du solltest deshalb vermeiden, den Horizont des Spielers zu verändern, und ihn höchstens horizontal drehen.

Leider reichte diese Einschränkung nicht aus, um *Simulator Sickness* bei jedem Spieler zu verhindern. Nicht gerade wenige Leute beklagten nach dem ersten Ausprobieren der *Tuscany Demo* (siehe Bild 4.7), welche dem Oculus Rift Development Kit 1 beilag, Übelkeit oder Kopfschmerzen. Neben dem reinen Laufen wurden das Drehen und Treppenlaufen als besonders „schlimm“ empfunden.

Bild 4.7 Die Standarddemo für das Oculus Rift DK1 war dieses Haus in der Toskana.

Viele der *Early-Adopter* gewöhnten sich schnell an die virtuellen Bewegungen und hatten keine Probleme mehr, andere jedoch nicht. Diese Situation erzeugte zwei Lager, welche noch bis heute bestehen: Die einen sagen „*So wie man auf einem Schiff erst See-Beine bekommen muss, muss man sich auch in Virtual Reality seine VR-Beine erarbeiten*". Grundsätzlich sei also *jeder* in der Lage, *Simulator Sickness* zu überwinden und danach völlig ohne Übelkeit und Kopfschmerzen zu spielen. Das andere Lager ist jedoch der Ansicht, dass Leute, die unter Simulator Sickness leiden, *sich nicht durch Training an die virtuellen Bewegungen gewöhnen können*.

- Meiner Ansicht nach ist es eine Mischung aus beidem: Als Erstes gibt es Leute, die gar keine Probleme mit solchen Bewegungen haben. Andere haben Probleme, können sich aber mittels Training und Ausdauer daran gewöhnen. Als Drittes gibt es jedoch auch Spieler, die sich wohl nie an diese Art der *Locomotion* gewöhnen können.

Die Übelkeit wurde zum Teil natürlich auch durch die niedrige Auflösung und die verwaschene Darstellung des ersten *Developer Kits* von Oculus erzeugt, aber selbst bei aktuellen Brillen lässt sich das Phänomen noch beobachten. Schon zu *DK1*-Zeiten haben diverse VR-Entwickler versucht, durch verschiedene Maßnahmen die Probleme in den Griff zu bekommen, ohne auf das freie Herumlaufen zu verzichten.

Aus diesen frühen Tests wurden verschiedene Komfortfunktionen entwickelt, die man mittlerweile in den meisten Spielen finden kann.

4.4.1.1 Umsetzung der klassischen Locomotion

Nach dieser Einführung in die klassische Locomotion möchte ich dir jetzt näherbringen, was du beim Entwickeln beachten solltest, wenn du sie für dein Spiel verwendest. Grundsätzlich gilt: Je komfortabler deine Anwendung ist, desto mehr Leute können dein Spiel

problemlos spielen. Manche der Komfortfunktionen schränken jedoch die Freiheit des Spielers und dadurch eventuell den Spielspaß ein. Deshalb habe ich meine Tipps.

Ich habe die Punkte jeweils in zwei Kategorien unterteilt: Dinge, die du *immer beachten* solltest, und Dinge, die du als *optionale* Funktion für empfindliche Spieler einbauen solltest.

Die meisten der folgenden Komfortfunktionen hast du wahrscheinlich bereits in dem ein oder anderen VR-Spiel bemerkt.

Drehen der Camera/Umsehen

IMMER BEACHTEN: Das Umsehen ist in jedem Spiel wichtig. In einem Virtual-Reality-Spiel solltest du aber unbedingt vermeiden, den Horizont des Spielers künstlich zu verändern, solange er virtuell zu Fuß unterwegs ist. Das bedeutet, du solltest es dem Spieler nicht ermöglichen, dass er per Maus oder Thumbstick nach oben oder unten schauen kann. Diese Bewegungen dürfen ausschließlich mit dem Headset ausgeführt werden. Ansonsten bekommt der Spieler das Gefühl, dass die Welt um ihn herum kippt. Auch das Rollen der Kamera solltest du nicht ermöglichen, wenn der Spieler zu Fuß unterwegs ist.

OPTIONAL: Um zu verhindern, dass empfindlichen Spielern schlecht wird, kannst du bei VR-Spielen das Drehen über Maus oder Gamepad auch komplett deaktivieren. Deine Spieler können sich dann also nur noch mit dem Kopf umsehen. „Echtes“ Umsehen stellt kein Problem dar, weil die Bewegungen deines Kopfes mit dem übereinstimmen, was deine Augen sehen.

Alternativ gibt es auch die Möglichkeit, den Spieler ruckartig, also ohne Übergang, in 45-Grad-Schritten zu drehen. Der Spieler sieht die Drehbewegung nicht, sondern schaut nach Tastendruck direkt in die neue Blickrichtung. Diese Möglichkeit wird allerdings von dem Spieler häufig als unnatürlich empfunden und verringert eventuell die Immersion. Eine verringerte Immersion trägt aber auch dazu bei, dass empfindlicheren Spielern nicht so schnell übel wird. Diese Option wird häufig *Snapping Rotation* oder *Comfort Rotation* genannt.

Bewegen der Camera/Laufen

IMMER BEACHTEN: Beim Laufen müssen ein paar etablierte Elemente des virtuellen Laufens deaktiviert werden: Der Spieler darf zum Beispiel nicht beschleunigen (*Momentum* aufbauen), sondern sollte direkt in einem festen Lauftempo loslaufen. Auch auf das *Head Bobbing* genannte Schütteln der Kamera während des Laufens muss verzichtet werden. Außerdem sollte das Lauftempo einem realistischen Tempo angepasst werden. In vielen Spielen ist das Gehen immer noch ein Joggen, was in VR zu Unstimmigkeiten bei der Wahrnehmung führen kann.

OPTIONAL: Als Komfortfunktion für empfindliche Personen kann das Lauftempo noch stärker reduziert werden. Um Übelkeit zu verhindern, sollte die Laufgeschwindigkeit auf das Tempo eines langsamen Ganges reduziert werden. Auf die Möglichkeit zu sprinten sollte gegebenenfalls vollständig verzichtet werden.

4.4.2 Teleport Locomotion

Einige Spiele, darunter besonders viele Spiele für *SteamVR*, nutzen die sogenannte *Teleport Locomotion*, bei der man sich vollkommen ohne künstliche Bewegung in einem Level bewegen kann. Wie der Name es schon sagt, geschieht dies über Teleportation: Du hältst in der Regel eine Taste gedrückt und aktivierst damit eine Zielhilfe. Wie in Bild 4.8 wird die Zielposition meist durch eine gebogene Linie markiert, deren Reichweite begrenzt ist. Gezielt wird entweder über einen der Motion-Controller, wenn sie vorhanden sind, oder direkt mit dem Kopf. Lässt du jetzt die Taste los, teleportierst du dich direkt an die gewählte Zielposition. Bei manchen Varianten der *Teleport Locomotion* ist es auch möglich, die gewünschte Zielrotation zu bestimmen.

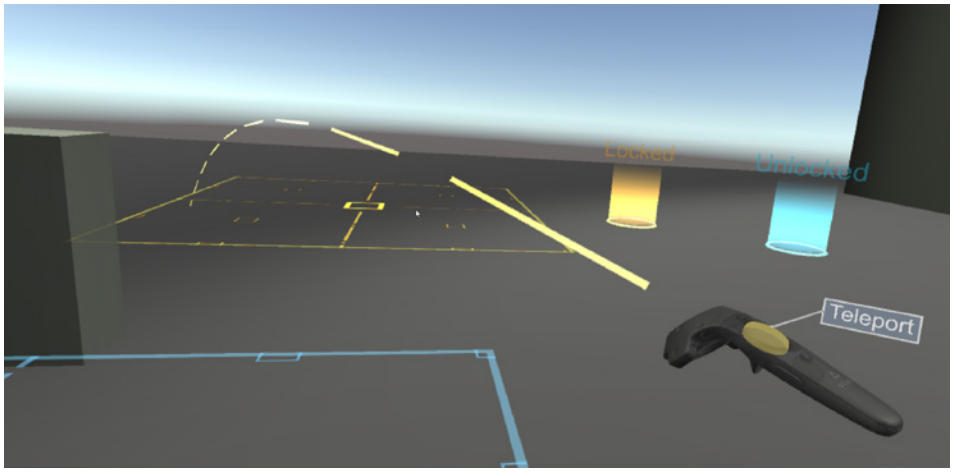

Bild 4.8 So sieht die Teleportmechanik in Valves *the Lab* (SteamVR) aus.

Ein Vorteil dieser Technik ist, dass du vollkommen auf künstliche Bewegungen verzichtest und dich trotzdem völlig frei in dem Level bewegen kannst. Für Spieler, die schnell unter *Simulator Sickness* leiden, ist es also eine sehr gute Lösung.

Ein großer Nachteil ist allerdings, dass diese Art der Fortbewegung in den meisten Fällen nicht zu der Spielwelt passt und deshalb die Immersion stört. In einem Sci-Fi-Szenario macht es vielleicht Sinn, dass man ein Teleportations-Device besitzt, im Mittelalter-Szenario eher nicht. Viele Kritiker der Technik bemängeln, dass diese den Spielfluss sehr stark stört, weil man mit jedem Teleport für einen Moment aus der Welt gerissen wird.

Manche Spiele schaffen es jedoch, das Teleportieren so in die Spielwelt zu integrieren, dass es nicht aufgesetzt wirkt, sondern so, als ob das Spiel so oder so mit Teleportation realisiert worden wäre. Ein sehr gutes Beispiel hierfür ist das Spiel *Budget Cuts* von dem Entwickler Studio *Neat Corporation*. In dem Schleichspiel schießt du zunächst mit einer speziellen Waffe einen Teleporter an den Ort, wohin du dich teleportieren möchtest. Dann kannst du mit einem Portal, welches du wie in Bild 4.9 in der Hand hältst, die Lage am Zielort auskundschaften. Ist die Luft rein, steigst du durch das Portal und befindest dich am neuen Standort. Einen Link zur kostenlosen Demo im *Steam Store*, wo du auch weitere Screenshots und einen Trailer ansehen kannst, findest du im Kasten.

Bild 4.9 Teleportation ist als wichtiges Spielelement perfekt in die Welt von *Budget Cuts* (SteamVR) integriert. (Quelle: Neat Corporation)

Teleportation done right: Budget Cuts

http://store.steampowered.com/app/459860/

4.4.3 Ohne künstliche Locomotion

Es gibt auch Spiele, die ihr Spielkonzept so entwerfen, dass sie vollkommen ohne künstliche *Locomotion* auskommen. Vor allem im mobilen VR-Bereich findet man zahlreiche Spiele, bei denen es schlicht keinen Bedarf gibt, frei herumzulaufen. Um dies zu bewirken, werden die Entwickler kreativ: Manche Spiele versetzen den Spieler in die Perspektive eines alles übersehenden Spielleiters. Wie in einem Puppenhaus schaut der Spieler auf seine Spielfigur, die Gegner und das Level. Die Spielfigur wird aus dieser Top-down-Perspektive gesteuert, ohne dass die Camera bewegt werden muss. Die Levels wären dann von ihrer Größe her so aufgebaut, dass sie stets vollständig überblickt werden können, ohne dass sich die Kamera bewegen muss.

Bild 4.10 In dem Spiel *Herobound* (GearVR) befindet sich die Kamera an festen Punkten und wechselt zur nächsten Perspektive, wenn die Spielfigur den sichtbaren Raum verlässt. (Quelle: Gunfire Games LLC)

Andere Spielkonzepte kommen auch trotz Ich-Perspektive vollkommen ohne künstliche *Locomotion* aus. Manche Puzzlespiele wie *ESPER* (GearVR, Oculus Rift) oder *I Expect You to Die* (Oculus Rift, SteamVR, PlayStationVR) setzen dich an einen Schreibtisch in einem Büro und erlauben es dir, mittels einer Art Telekinese auch mit entfernten Objekten zu interagieren. Du musst also nicht durch den virtuellen Raum laufen, sondern kannst alles bequem von deinem Schreibtisch erreichen. Hast du alle Rätsel in einem Raum gelöst, gelangst du ins nächste Level.

Bild 4.11 In dem Agenten-Spiel *I Expect You to Die* (Oculus Rift, SteamVR, PlaystationVR) löst du alle Rätsel vollkommen ohne Locomotion. (Quelle: Schell Games)

Ein weiter interessanter Ansatz, der vollkommen auf *Locomotion* verzichtet, ist der des Room-Scale-Thrillers *A Chair in a Room: Greenwater* von *Wolf & Wood Interactive* (SteamVR). Das SteamVR-Spiel erwartet vom Spieler, dass er mindestens eine Spielfläche von 2,5 x 2,5 Metern besitzt. Alle Orte des Spiels, zum Beispiel eine Blockhütte, eine Abstellkammer oder auch ein Hotelzimmer, sind auf diesen Bereich angepasst. Durch geschicktes Level-Design kommen einem die Räume beim Spielen aber nicht unnatürlich klein, sondern vollkommen plausibel vor. Manche Orte im Spiel bestehen aus mehreren Räumen, die über geschlossene Türen miteinander verknüpft sind. Öffnet man eine Tür, wird die Sicht kurz abgedunkelt und man befindet sich danach direkt in dem neuen Raum und kann weiterlaufen. Das gesamte Spiel kann also ausschließlich über das Herumlaufen in dem eigenen Spielbereich gespielt werden.

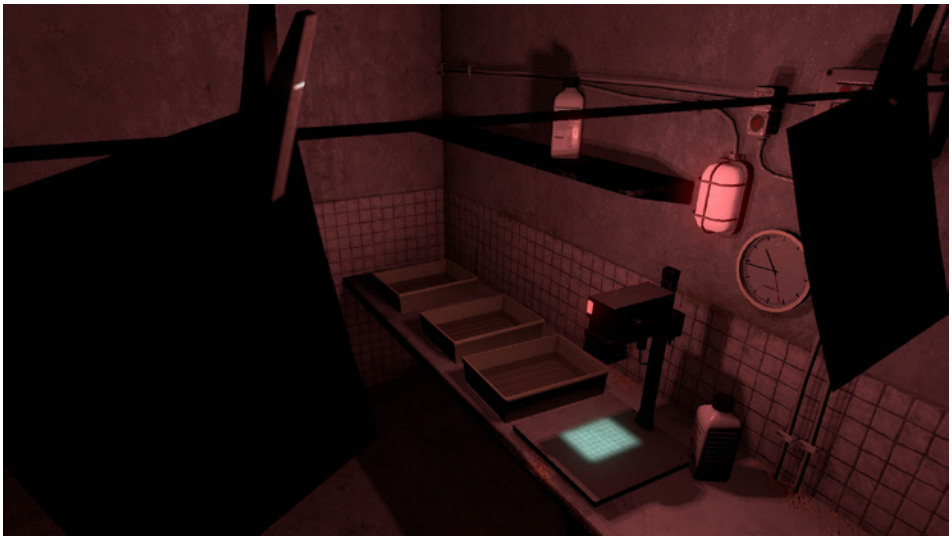

Bild 4.12 Das Fotolabor aus dem Spiel *A Chair in a Room* (SteamVR). Alle Levels des Spiels sind auf den Spielbereich des Spielers angepasst.
(Quelle: Wolf & Wood Interactive Ltd)

Ein ebenfalls interessantes *Room-Scale*-Konzept ist das sogenannte *Redirected Walking*. Bei diesem System lässt das Spiel den Spieler immer wieder durch seine vorhandene Spielfläche laufen, während der Spieler allerdings denkt, er läuft durch stetig neue Räume und Flure. Ziel ist es, dass der Spieler gar nicht bemerkt, dass er die ganze Zeit nur in seinem Spielbereich hin und her läuft. Um dieses Ziel zu erreichen, gibt es verschiedene Ansätze. Ein gängiger Ansatz ist zum Beispiel, dass die virtuelle Rotation nicht mit der tatsächlichen Rotation übereinstimmt.

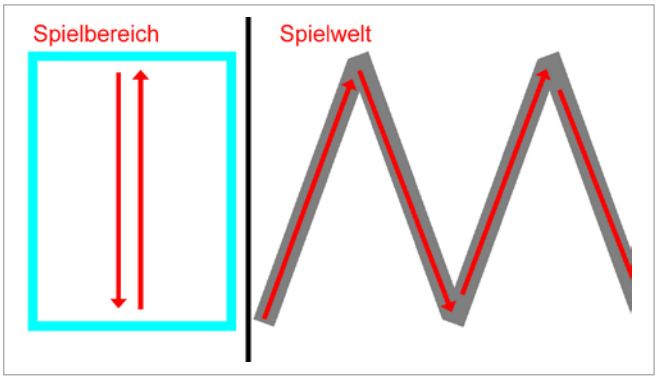

Bild 4.13 Links: Laufweg des Spielers in der Realität. Rechts: Der virtuelle Laufweg des Spielers.

Bild 4.13 zeigt das Konzept vereinfacht: Der Spieler dreht sich in der Realität immer um 180°, wenn er das Ende seines Spielbereiches erreicht, und läuft dieselbe Strecke wieder zurück. Virtuell dreht er sich jedoch jedes Mal nur um 160°, wodurch er virtuell nicht wieder zurückläuft, sondern in eine neue Richtung. Wechselt man jetzt noch links herum und rechts herum, kann der Spieler theoretisch unendlich weit laufen. Das Besondere: Ist der Unterschied zwischen echter und virtueller Rotation perfekt gewählt, ist es möglich zu erreichen, dass es dem Spieler gar nicht auffällt.

In dem Kasten mit den weiterführenden Links findest du ein Beispiel-Video, welches das Konzept in einer Unity-Demo zeigt.

Weiterführende Links

Herobound: Spirit Champion:
https://www.oculus.com/experiences/gear-vr/1088313964515376/

Esper: *https://www.oculus.com/experiences/gear-vr/797549690323260/*

I Expect You to Die: *https://iexpectyoutodie.schellgames.com/*

A Chair in a Room: *http://store.steampowered.com/app/427760/*

Redirected Walking (Konzept-Video):
https://www.youtube.com/watch?v=ZliqiTOr1BY

4.4.4 Andere Eingabemethoden für das Laufen

Die bisher vorgestellten Locomotion-Arten sind die derzeit am meisten genutzten Arten. Die VR-Scene entwickelt sich aber stetig weiter und so haben sich über die Zeit verschiedene weitere Möglichkeiten der Fortbewegung in VR entwickelt. Die meisten davon kombinieren klassische Locomotion, also künstliche Bewegung, mit einem innovativen Eingabeprinzip, das *Simulator Sickness* verhindern soll.

Gängige Varianten sind zum einen *Walk-in-Place*-Locomotion und *Arm-Swing*-Locomotion. Bei Ersterem läuft der Spieler auf der Stelle, um sich in der virtuellen Welt fortzubewegen. Die einzelnen Schritte werden entweder über einen zusätzlichen Sensor an den Beinen oder über die Auf- und Abbewegung des Headsets erkannt und die Geschwindigkeit des virtuellen Spielers wird entsprechend angepasst. Diese Technik ist entstanden, nachdem aufgefallen war, dass Auf-der-Stelle-Laufen *Simulator Sickness* vorbeugen kann. Darauf basierend wurde *Arm-Swing*-Locomotion entwickelt. Mit dieser Technik kann die Laufgeschwindigkeit genauer erkannt werden als über die Auf- und Abbewegungen des Headsets. Zudem ist es auch nicht notwendig, einen Controller an deinen Beinen zu befestigen. Bei *Arm-Swing*-Locomotion schwingst du deine Arme so, als ob du laufen würdest, und auf diese Weise wird deine Bewegungsgeschwindigkeit erkannt und virtuell umgesetzt.

Freedom Locomotion VR

Eine Demo-Anwendung, die dir erlaubt, viele verschiedene Locomotion-Arten mit deiner SteamVR-kompatiblen Brille auszuprobieren.

http://store.steampowered.com/app/584170/Freedom_Locomotion_VR/

4.5 Lebendige Spielwelt erschaffen

Eine Welt kann noch so fotorealistisch aussehen, in Virtual Reality wirkt sie schnell leblos oder statisch, wenn sich nichts bewegt. Je realistischer die Optik, desto mehr vergleichen wir sie unterbewusst mit der echten Welt und desto mehr Unterschiede fallen auf. Eine statische Welt wirkt deshalb noch schneller als in einem Nicht-VR-Spiel leblos und statisch.

Über das Thema, wie man eine lebendige Spielwelt erschafft, könnte man wahrscheinlich ein komplett eigenes Buch schreiben, weshalb ich dir in diesem Abschnitt nur ein paar Einsteigertipps geben möchte. Grundsätzlich sind die Vorgehensweisen, um eine plausible Spielwelt zu erschaffen, nicht für Virtual Reality spezifisch, sodass es eher allgemeine Vorgehensweisen in Videospielen sind. In VR macht eine lebendige Spielwelt jedoch noch mehr von der Erfahrung aus.

Um eine Welt in einem Virtual-Reality-Spiel lebendiger wirken zu lassen, sollte sie zunächst nicht vollkommen statisch sein. Es sollten sich in der Scene Objekte bewegen, Geräusche machen und das Interesse des Spielers wecken. Auch Partikelsysteme, die den Raum mit Rauch, Staub oder Fliegen füllen, können benutzt werden, um ihn lebendiger wirken zu lassen.

Bild 4.14 In dieser Scene mit einer von Pilzen befallenen verlassenen Wohnung wurde ein Partikelsystem verwendet, um die Situation auf den Spieler weniger passiv wirken zu lassen. Die Partikel sollen Staub und Pilz bzw. Schimmelsporen darstellen. Viele Spieler halten in VR instinktiv die Luft für einen Moment an.

Der nächste, noch schwierigere Punkt ist, die Spielwelt plausibel auf den Spieler reagieren zu lassen. Hier sollte der Spieler möglichst nicht bemerken, dass es sich um ein „gescriptetes“ Event handelt. Auf den Spieler sollte es möglichst natürlich und plausibel wirken. Auch hier gilt, je realistischer dein Szenario, desto detaillierter muss deine Spielwelt reagieren. Nimmst du zum Beispiel Blickkontakt mit einem Menschen auf, erwarten wir, dass er darauf reagiert. Diese Reaktion muss bei einer sehr detaillierten Welt ebenfalls viel realistischer ausfallen als bei der simplen Welt von *Job Simulator* (Bild 4.15).

Ein weiteres besonderes Merkmal von *Job Simulator* ist, dass du sehr viele Freiheiten hast. Alles, was du siehst, kannst du anfassen, verwenden oder bedienen. Damit der Spieler niemals enttäuscht wird, weil etwas nicht funktioniert, wurden die meisten Maschinen vereinfacht. Dafür kann aber jede Funktion, die man von dieser vereinfachten Maschine erwartet, auch tatsächlich verwendet werden. Der Spieler kommt deshalb nie in die Situation, dass eine Taste auf einer Maschine keine Funktion hat, und wird deshalb auch nie enttäuscht. Dadurch bekommt der Spieler das Gefühl, dass er „alles machen kann, was er will“. Dabei wurde die Spielwelt nur von Anfang an so entworfen, dass er nicht auf den Gedanken kommt, Dinge zu tun, die nicht vorgesehen sind. In einem realistischen Szenario wäre es komisch, eine Kasse mit nur vier Tasten zu haben, und enttäuschend, wenn die Kasse nicht vollständig bedient werden kann.

Bild 4.15 *Job Simulator* (SteamVR, Oculus Rift, PlaystationVR) bietet eine in sich schlüssige Spielwelt. (Quelle: Owlchemy Labs)

Ein weiteres Beispiel, warum die *Job Simulator*-Welt sehr schlüssig auf den Spieler wirkt, ist, dass die meisten deiner Taten auch Auswirkungen auf die Spielwelt haben: Bedienst du einen Kunden nicht, weil du eifrig damit beschäftigt bist, das Feuer zu löschen, das du versehentlich entfacht hast, weil du eine Banane in den Ofen gelegt hast, verlässt dieser wütend das Geschäft. Die Reaktionen und Auswirkungen sind in der Regel nicht realistisch, sondern eher comichaft. Weil dies aber mit der Optik und Atmosphäre des Spiels gut übereinstimmt, wirkt die Welt in sich schlüssig und plausibel auf den Spieler.

Natürlich kann und soll nicht jedes Spiel aussehen wie *Job Simulator*. Gerade wenn du aber in einem kleinen Team oder alleine arbeitest, kannst du mit dem einfachen Look leichter eine sehr gute, stimmige Welt erzeugen. In Spielen mit anderer Optik, wo du Maschinen zum Beispiel nicht einfach so stark vereinfachen kannst, musst du es dem Spieler *innerhalb der Spielwelt* plausibel erklären, warum bestimmte Dinge eingeschränkt sind, auch wenn der tatsächliche Grund einen anderen Hintergrund hat. *Lone Echo* von dem Entwicklerstudio *Ready At Dawn* spielt zum Beispiel weit in der Zukunft und der Spieler ist ein Roboter. Neben dem Punkt, dass es auch ein sehr interessanter Wechsel der Perspektive ist, hat ein Szenario, mit dem der Spieler nicht vertraut ist, auch den Vorteil für die Entwickler, dass sie sich mehr Freiheiten erlauben können, ohne dass die Spielwelt künstlich auf den Spieler wirkt. Durch das Zukunftsszenario sehen die meisten Geräte anders aus, als wir es kennen, und der Spieler möchte mit ihnen nicht so interagieren, wie er es aus dem echten Leben kennt. Die Tatsache, dass der Spieler ein Roboter ist, erklärt außerdem, warum bestimmte Situationen anders ablaufen, als man es aus dem echten Leben kennt.

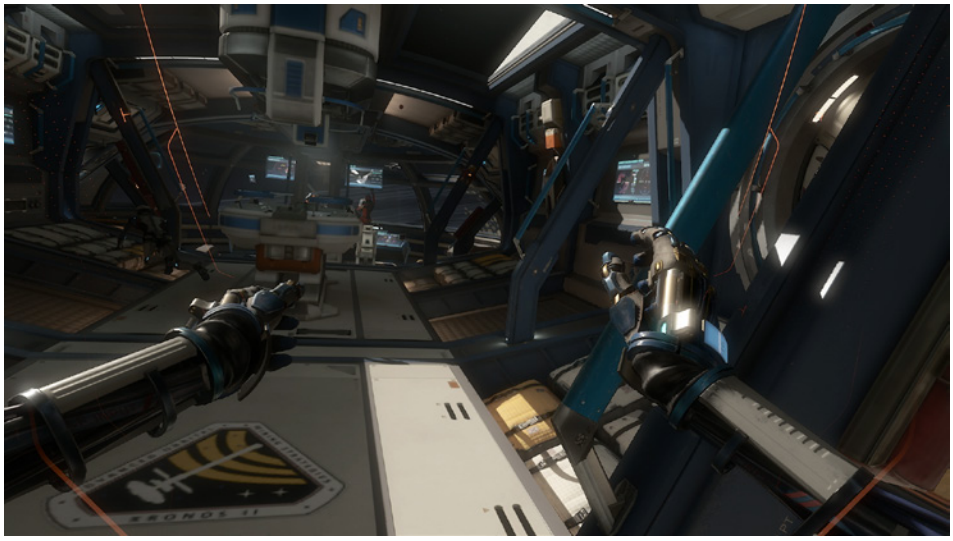

Bild 4.16 *Lone Echo* (Oculus Rift) entführt den Spieler in eine ihm unbekannte Welt. (Quelle: Ready At Dawn Studios)

Je plausibler und interaktiver die Welt ist, desto mehr zieht sie den Spieler in sich hinein und er fühlt sich nicht nur wie der Betrachter eines 360°-Stilllebens. Die hier verwendeten Beispiele sind wahrscheinlich viel komplexer als deine ersten eigenen Projekte und wahrscheinlich hast du auch kein riesiges Budget, um deine Welt mit derart vielen Details zu füllen. Deswegen ist der folgende Punkt besonders am Anfang für dein Spiel sehr wichtig: *Konsistenz*. Kann der Spieler in manchen Bereichen jeden einzelnen Gegenstand hochheben und im nächsten Bereich findet er nur statische Deko-Elemente, führt dies zu Frustration. Wenn der Spieler bereits am Anfang lernt, dass er mit bestimmten Dingen nicht interagieren kann, wird er es später nicht immer wieder versuchen und so auch nicht enttäuscht werden.

Kleines Beispiel

In dem Beispielprojekt findest du unter *VRSpieleMitUnity/DosAndDonts/Scenes* eine kleine Test-Scene mit dem Namen *LivingWorld*, die auf dem Jugendzimmer basiert. Die Scene zeigt dir zwei einfache Beispiele, um die, dir bekannte Welt ein wenig lebendiger wirken zu lassen: Zum einen ein stetig animiertes Flugzeug, zum anderen reagiert der kleine Roboter auf dem Tisch auf deine Blicke.

Weiterführende Links

Job Simulator:

http://store.steampowered.com/app/448280/Job_Simulator/

Lone Echo:

https://www.oculus.com/experiences/rift/1368187813209608/

4.6 User Interface

Als *User Interface* (kurz „UI“, teilweise auch „GUI“ für „Graphical User Interface“) bezeichnet man alle möglichen Arten von Einblendungen und Menüs, die der Spieler sieht und mit denen er interagieren kann. Dazu zählen zum Beispiel Texteinblendungen für Tutorials, Munitionsanzeigen und eben auch jede Art von Menü: Sei es das Inventar, eine Level-Auswahl oder auch Highscore-Listen. In den meisten Spielen ohne Virtual-Reality-Unterstützung wird das gesamte *User Interface*, wie in Bild 4.17 oben, einfach zweidimensional auf deinem Monitor angezeigt. Diese Darstellungsform nennt sich *Screen Space*, weil das User Interface nicht Teil der virtuellen Welt ist, sondern nur auf dem Bildschirm existiert.

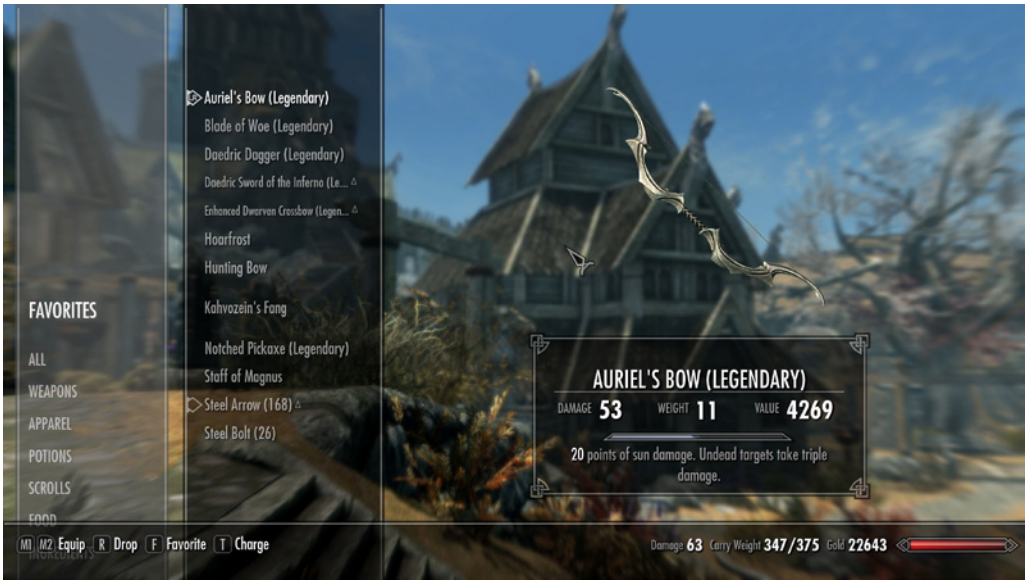

Bild 4.17 Beispiel für ein 2D-Menü auf der Bildschirmebene (Screen Space) in Skyrim (nicht VR-Version)

In Virtual Reality ist diese Darstellungsform nicht gut geeignet, weil man das Gefühl hat, das User Interface würde einem vor dem Kopf kleben. Je nach Größe des Menüs kann dies auch sehr unangenehm werden und im schlimmsten Fall sogar Simulator Sickness hervorrufen. Deshalb musst du dir immer überlegen, wie du die User-Interface-Elemente, die du für dein Spiel benötigst, gut in Virtual Reality umsetzen kannst.

Als Beispiel möchte ich dir hier das User Interface von Half-Life 2 zeigen, wie es im Original-Spiel aussah und wie es in der inoffiziellen Virtual-Reality-Modifikation[2] umgesetzt wurde. Bild 4.18 zeigt die beiden Varianten.

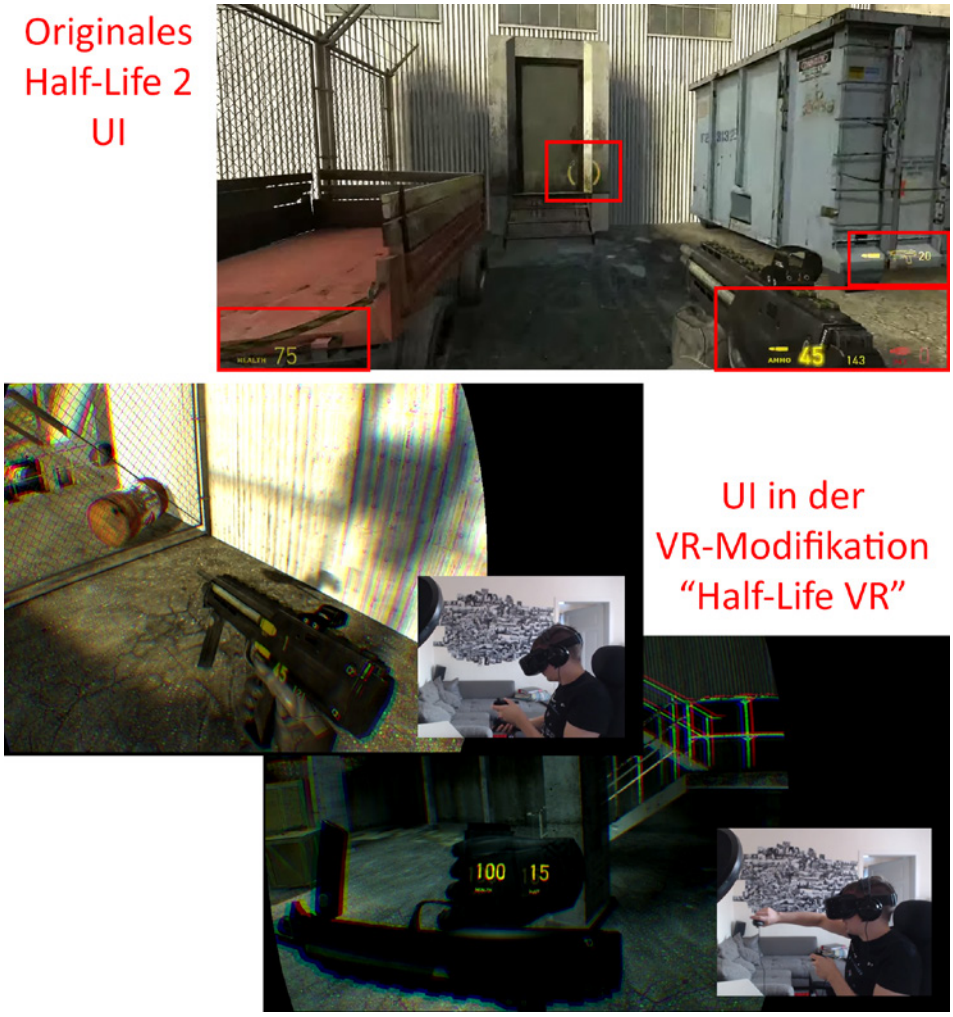

Bild 4.18 Das originale und das Virtual Reality User Interface von Half-Life 2 im Vergleich

Die Ersteller der Modifikation haben das User Interface so angepasst, dass es nicht mehr an der Sicht des Spielers klebt, sondern an der rechten Hand angezeigt wird. Die Munitionsanzeige befindet sich dabei stets an der Seite der ausgewählten Waffe und die Lebensanzeige findet man auf dem Handrücken des Spielers. Dieser Ansatz fühlt sich sehr intuitiv an und

[2] Die Mod erlaubte es, das Spiel mit dem Oculus Rift Development Kit 2 und Razer Hydra-Controllern in Virtual Reality mit Handtracking zu spielen. Es gab auch einen offiziellen Half-Life 2 VR-Modus, dieser bot jedoch kein Hand-Tracking. Beide Möglichkeiten, Half-Life 2 in VR zu spielen, funktionieren nicht mehr. Der Ersteller der inoffiziellen Mod arbeitet jedoch an einer neuen Version. Neuigkeiten dazu findest du im Subreddit der Modifikation: https://www.reddit.com/r/hlvr/

erlaubt es jederzeit, schnell die entsprechenden Werte abzulesen, ohne dass sie die Immersion stören. Auf das Fadenkreuz wurde vollständig verzichtet, da man über die Zielhilfe der Waffe oder, wenn keine vorhanden ist, mittels Kimme und Korn zielen kann.

Es gibt aber auch den Ansatz, vollkommen auf künstliche User Interfaces zu verzichten und ausschließlich auf sogenannte „natürliche“ Interfaces zu vertrauen. Bei diesen natürlichen Interfaces gibt es keine traditionellen Menüs oder Einblendungen, sondern alles geschieht auf eine Art und Weise, wie es im echten Leben auch passieren würde. Einstellungen nimmst du zum Beispiel vor, indem du, wie in Bild 4.19, Regler verschiebst und Schalter umlegst.

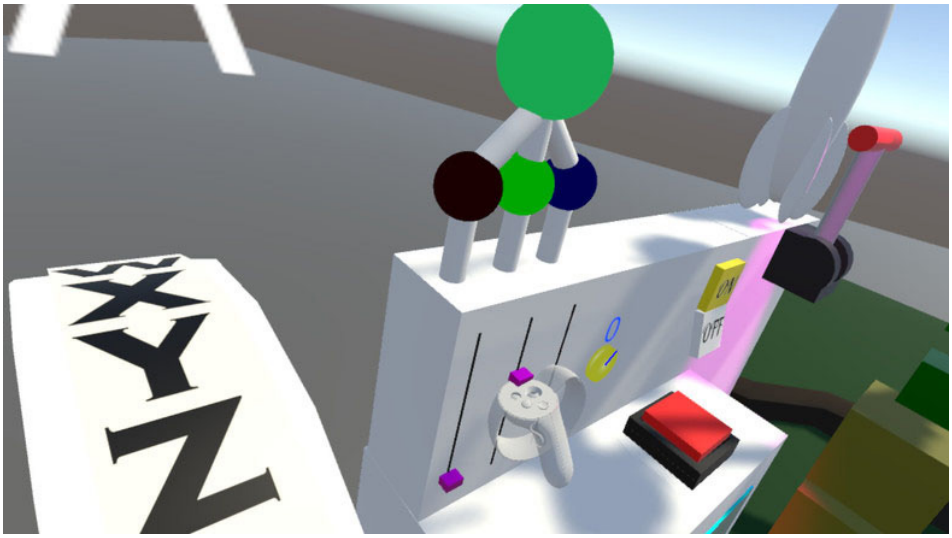

Bild 4.19 Beispiele für natürliche User Interfaces
(Quelle: NewtonVR)

Bei einem *Shooter* hätte die Waffe entweder ein Display mit einer Munitionsanzeige oder man muss wie im echten Leben die Magazingröße kennen und die Schüsse zählen. Eine Level-Auswahl könnte man über einen Stapel Disketten realisieren, aus denen man das gewünschte Level heraussucht und die jeweilige Diskette dann in einen virtuellen Computer einlegt, um das Level zu laden. Hinweise und Anleitungen würde man ausschließlich über Zettel und Plakate in der 3D-Welt erhalten und nicht als Einblendung. Ein Inventar könnte ein virtueller Rucksack sein, in dem man seine Sachen sammeln kann. Vor allem in Spielen, die *Motion Controller* unterstützen, ist dies eine sehr intuitive und in den meisten Fällen auch einfache Lösung.

Besonders bei Spielen ohne Motion-Controller ist es allerdings nicht immer möglich, komplett auf „künstliche“ User Interfaces zu verzichten. Es gibt jedoch eine einfache Möglichkeit, ein beliebiges 2D-User Interface VR-tauglich zu machen: Du verschiebst es vom *Screen Space* in den *World Space*, das bedeutet, du integrierst das Menü in die virtuelle Welt.

Kleines Beispiel

In dem Beispielprojekt findest du unter *VRSpieleMitUnity/DosAndDonts/Scenes* zwei kleine Test-Scenes mit den Namen *Menu_Bad* und *Menu_Good*. *Menu_Bad* zeigt dir als negatives Beispiel ein Menü, das an der Kamera klebt. *Menu_Good* zeigt dir ein einfaches Menü, das gut in VR zu bedienen ist. Schaue eine Schaltfläche an, um sie zu markieren.

Weiterführende Links

NewtonVR ist eine Library für physikalische und natürliche Interaktion:
http://www.newtonvr.com/

4.6.1 Distanz und Größe

Erstellst du ein User Interface, das nicht zweidimensional am Bildschirm klebt, sondern dreidimensional in der Welt platziert ist, musst du dir auch Gedanken über die Distanz zur *Camera* machen. Denn die tatsächliche Anzeigegröße wird bei einem User Interface im *World Space* zum einen durch die Größe des User Interface und zum anderen durch Distanz zur Kamera bestimmt.

Ein großes User Interface, das weit weg ist, kann also dieselbe Anzeigegröße haben wie eines, das nah an der Kamera und eigentlich recht klein ist. Doch obwohl die Größe im Endeffekt gleich ist, macht es in VR einen Unterschied: Die Kombination aus Distanz und eigentlicher Größe bestimmt, wie gut und vor allem wie angenehm die Texte zu lesen sind. Ist der Text zu nah an der Kamera, kann es auf Dauer sehr anstrengend werden, den Text zu lesen, und so Müdigkeit, Kopfschmerzen oder auch Simulator Sickness hervorrufen. Zu weit entfernt sollte das User Interface für eine gute Lesbarkeit allerdings auch nicht liegen.

Eine bestimmte minimale und maximale Distanz lässt sich nicht so einfach definieren, da diese abhängig von Größe und Inhalt wären. Als Richtlinie lässt sich jedoch festhalten, dass gewöhnliche User Interfaces mindestens eine Distanz von zwei Metern zum Spieler haben sollten, damit sie angenehm lesbar sind. Wenn du ein User Interface erstellst, experimentiere immer ein wenig mit unterschiedlichen Distanzen und Skalierungen, bis es gut in die Welt passt und angenehm zu lesen ist.

4.7 Aufmerksamkeit lenken

Wenn du ein Spiel entwickelst, stehst du meistens früher oder später vor dem Problem, dass du den Spieler in eine bestimmte Richtung lenken oder ihn zumindest auf ein Ereignis aufmerksam machen möchtest, damit er es nicht verpasst. In Nicht-VR-Spielen geschieht dies meist mittels Zwischensequenzen oder geskripteten Kamerafahrten: Man betritt einen Raum und eine epische Kamerafahrt weckt, wie in Bild 4.20, unser Interesse an einem bestimmten Ort oder Gegenstand, den wir uns nächstes Mal genauer ansehen sollten.

Bild 4.20 In Nicht-VR-Spielen werden, wie in hier in *Tomb Raider*, Cutscenes und Kamerafahrten verwendet, um die Aufmerksamkeit des Spielers auf bestimmte Dinge zu lenken.
(Quelle: Raiding the Globe, Eidos Montreal, Square Enix, Crystal Dynamics)

In diesen meist cineastischen Zwischensequenzen verliert der Spieler vollständig die Kontrolle über die Kamera. In vielen Fällen verändert sich auch die Kameraperspektive, die eigene Spielfigur steht dann nicht mehr zwingend im Mittelpunkt, sondern die Kamera bewegt sich frei durch den Raum. Auch ein Wechsel von der Ego-Ansicht in eine Third-Person-Ansicht ist nicht selten.

In Virtual Reality können die meisten dieser Mittel nicht so einfach angewandt werden. Das größte Tabu bei einem VR-Spiel ist es nämlich, das freie Umsehen des Spielers zu unterbinden. Deaktivierst du die Rotationserkennung des Headsets, ist Simulator Sickness quasi vorprogrammiert. Außerdem solltest du auch auf geskriptete Rotationen und Kamerafahrten verzichten, die nicht durch den Spieler direkt gesteuert werden. Denn selbst Spielern, denen klassische Locomotion nichts ausmacht, wird bei unvorhergesehenen Bewegungen schnell mal komisch. Dazu kommt natürlich noch der Fakt, dass eine solche fremde Steuerung der Perspektive die Immersion des Spielers stark beeinflussen würde.

Aufgrund dieser Einschränkungen ist es in Virtual Reality deutlich schwieriger, die Aufmerksamkeit des Spielers zu steuern. Im schlimmsten Fall schaut der Spieler in entschei-

denden Schlüsselsituationen gerade in eine andere Richtung und verpasst somit eventuell wichtige Elemente des Spiels. Wenn dies passiert, könnte man dem Spieler die Schuld geben: „Warum schaut er auch in eine andere Richtung? Selber schuld!"

Dieses Problem auf den Spieler zu übertragen, ist allerdings keine gute Idee. Denn in der Regel macht der Spieler es nicht mit Absicht und wusste schlicht nicht, wo er genau hinschauen sollte. Es ist also deine Aufgabe, mit verschiedenen Hilfsmitteln und ohne Zwischensequenzen dafür zu sorgen, dass der Spieler im richtigen Moment auch in die richtige Richtung schaut.

Grundsätzlich spricht natürlich nichts dagegen, die Möglichkeit, Informationen zu verpassen, als ein Spielelement zu integrieren. Diese verpassten Informationen sollten für das Fortschreiten im Spiel aber nicht relevant sein, da dies sonst frustrierend für den Spieler werden kann.

Aber wie steuert man besten die Aufmerksamkeit des Spielers, wenn man ihn nicht, wie bei vielen aktuellen Spielen, dazu zwingen kann, eine bestimmte Sequenz nicht zu verpassen? Die Antwort ist denkbar einfach: Man nimmt ihn bei der Hand und macht ihn so lange auf etwas aufmerksam, bis er verstanden hat, was er tun soll. Wie du das machst, sollte möglichst kreativ sein und in deine Spielwelt passen. Um ein gutes Konzept zu entwickeln, solltest du dir überlegen, was im echten Leben verwendet wird, um die Aufmerksamkeit von Personen auf sich zu ziehen.

In den nächsten Abschnitten gebe ich dir ein paar grundlegende Tipps, die du unabhängig von deinem Spielkonzept verwenden kannst.

4.7.1 Objekte untersuchen

Soll der Spieler ein bestimmtes Objekt ein wenig genauer ansehen, kannst du natürlich darauf warten, dass der Spieler irgendwann zufällig das Objekt von alleine untersucht, dies kann allerdings frustrierend für ihn sein, da er keine klare Aufgabe hat. In Virtual Reality tritt Frust häufig deutlich schneller auf als in vergleichbaren Situationen in normalen Spielen. Du solltest deshalb das Interesse des Spielers an dem jeweiligen Objekt wecken, damit er von alleine auf die Idee kommt, das Objekt genauer zu untersuchen.

Hier ein paar Beispiele, wie du das erreichen kannst:

- **Blinken/Leuchten:** Es ist das klassische Mittel, damit ein Spieler sich etwas genauer ansieht: Lass das Objekt blinken. In Spielen wie *Resident Evil* leuchten alle Items, die der Spieler aufheben kann, mit kleinen bunten Lichtern. Das ist grundsätzlich ein gutes Mittel, um dem Spieler zu signalisieren, dass er hier etwas machen kann. Diese Lichter gleichen allerdings eher einem Wink mit dem Zaunpfahl. Im Idealfall sollte der Spieler das Gefühl haben, er sei selber auf die Idee gekommen, das Objekt zu untersuchen, und nicht, dass es vollkommen offensichtlich ist. Blinkende Lichter sind an elektronischen Geräten, die der Spieler untersuchen soll, vollkommen in Ordnung. Für alle anderen Objekte ist es in jedem Fall eine Möglichkeit.

Bild 4.21 Auf der Kommode links leuchten zwei Munitionsschachteln, damit der Spieler sie auch im Dunkeln findet.

- **Anstrahlen:** Dieses Mittel ist auch nicht wirklich neu, aber effektiv. Wenn es keinen Sinn ergibt, dass ein bestimmtes Objekt in der Spielwelt blinkt, oder du eine etwas subtilere Lösung suchst, dann strahle das Objekt an. Dieser Effekt funktioniert besonders gut, wenn der Raum oder die Spielwelt ein wenig dunkler ist. Interessante Objekte können dann zum Beispiel in dem Schein einer Lampe liegen oder durch die Sonne angestrahlt werden. Gegenstände, die im Schatten liegen, sind für den Spieler schlechter zu sehen und deswegen auch uninteressanter. Wie offensichtlich du die Aufmerksamkeit des Spielers auf einen bestimmten Bereich lenken solltest, hängt von dem Kontext der Situation ab.

Bild 4.22 Dieser vermeintlich bereits besiegte Gegner wird subtil vom Licht angestrahlt, damit der Spieler ihn sofort sieht und untersuchen möchte. Wenn der Spieler sich ihm nähert, erwacht der Gegner zum Leben.

- **Geräusche:** Nicht nur optische Reize ziehen die Aufmerksamkeit des Spielers auf sich, auch Geräusche wecken Interesse. Ein Geräusch ist allerdings schwieriger einer exakten Position zuzuordnen als ein optischer Effekt. Der Spieler kennt nämlich zunächst nur die Richtung, aus der das Geräusch kommt. Dafür kannst du Geräusche aber auch verwenden, wenn das Objekt für den Spieler gar nicht sichtbar ist. Liegt zum Beispiel ein Funkgerät in einem Schrank und der Spieler soll es finden, macht es Sinn, das Funkgerät regelmäßig ein wenig rauschen und sprechen zu lassen. Dasselbe Prinzip funktioniert auch, wenn der Gegenstand in einem vollkommen anderen Zimmer ist und der Spieler ein blinkendes Licht gar nicht sehen würde.

Bild 4.23 Dieses Radio steht im Dunkeln, spielt aber ein stetiges Rauschen ab. Wenn der Spieler mit seiner Taschenlampe in die Richtung strahlt, sieht er die zwei Munitionsschachteln, die davorliegen.

- **Anweisungen:** Soll der Spieler ohne Hilfsmittel ein bestimmtes Objekt suchen, damit er weiterkommt, solltest du ihm das mitteilen und ihn nicht ohne offensichtliche Aufgabe suchen lassen. Solche Anweisungen können von anderen Charakteren ausgesprochen werden oder zum Beispiel auch auf einem Zettel im Spiel stehen. Alternativ kann es auch etwas subtiler eingebaut werden, wichtig ist jedoch, dass der Spieler die Aufgabe versteht: Findet er eine verschlossene Kiste, weiß er nicht, ob er diese Kiste überhaupt öffnen kann oder nicht. Demnach gibt er eventuell schon nach kurzer Suche auf. Je schwieriger das Rätsel, desto klarer sollte dem Spieler seine Aufgabe sein, damit er nicht ungeduldig wird und aufgibt.

  Diese Liste enthält natürlich nur ein paar Beispiele, du kannst deiner Fantasie freien Lauf lassen. Es ergibt durchaus auch Sinn, unterschiedliche Effekte zu kombinieren. Ein optischer und ein akustischer Effekt sind für besonders wichtige Objekte zum Beispiel häufig sinnvoll.

4.7.2 Ereignisse nicht verpassen

Wenn du den Spieler auf ein bestimmtes Ereignis aufmerksam machen möchtest, kannst du grundsätzlich dieselben Dinge verwenden, wie um ihn auf ein Objekt hinzuweisen. Du musst jedoch ein paar Dinge berücksichtigen, damit der Spieler wichtige Ereignisse auf keinen Fall verpasst. Denn anders als bei Objekten, die darauf warten, von dem Spieler gefunden zu werden, gibt es bei Ereignissen häufig eine zeitliche Komponente, die es erlaubt, dass der Spieler sie verpassen kann. Selbst wenn das Ereignis durch eine bestimmte Aktion des Spielers ausgelöst wird, kann er sie verpassen, wenn sie nicht sofort und direkt vor ihm geschieht. Wenn der Spieler in eine andere Richtung schauen soll, musst du ihm das durch entsprechende Aktionen deutlich machen:

- **Spotlights/Anstrahlen:** Wenn sich alles um den Spieler herum verdunkelt und nur in eine Richtung ein Spotlight angeht, kann man davon ausgehen, dass er früher oder später auch in die korrekte Richtung schaut. Ohne zusätzliche Hinweise wird ihm die zeitliche Dringlichkeit allerdings eventuell nicht bewusst.
- **Geräusche:** Geräusche sind bei Ereignissen, die der Spieler eventuell verpassen könnte, sehr wichtige Signale. Springt plötzlich aus einer bestimmten Richtung ein Motor an oder beginnt jemand zu sprechen, wird der Spieler sich wahrscheinlich so schnell wie möglich in diese Richtung umdrehen.
- **Reaktionen der Umwelt:** Auch Reaktionen der Umwelt können benutzt werden, um den Spieler in eine bestimmte Richtung blicken zu lassen. Rennen zum Beispiel alle Tiere plötzlich ängstlich weg, wird der Spieler wahrscheinlich nachschauen wollen, vor was sie wegrennen. Oder sitzt der Spieler auf einer Tribüne, wird er wahrscheinlich nach vorne schauen, wenn plötzlich alle anderen Fans um ihn herum anfangen zu applaudieren.
- **Direkte Befehle:** Direkte Befehle funktionieren natürlich auch. Ruft eine Spielfigur zu dem Spieler „*Hinter dir!*“, wird er sich sehr wahrscheinlich schnell umdrehen, um zu prüfen, was dort passiert.

  Trotzdem gibt es grundsätzlich noch die Möglichkeit, dass der Spieler die Hinweise nicht versteht und deshalb ein wichtiges Ereignis verpasst. Deswegen ist der Zeitpunkt wichtig, wann du das Ereignis auslöst. Der Spieler sollte Zeit haben zu reagieren, wenn es inhaltlich Sinn ergibt, könntest du auch auf den Spieler warten:

- **Zeitliche Verzögerung:** Wichtig ist auch die zeitliche Abfolge zwischen dem ersten Aufmerksammachen und dem eigentlichen Ereignis, damit der Spieler nicht den Anfang verpasst. Der Spieler benötigt schließlich auch ein wenig Zeit, um das Signal zu interpretieren und dann in die korrekte Richtung zu blicken.
- **Blick- oder Positionsaktivierung:** Als Alternative zu einer zeitlichen Verzögerung kannst du das Ereignis auch erst auslösen, wenn der Spieler in die richtige Richtung schaut oder den entsprechenden Raum betreten hat. Es sollte dabei, je nach Situation, der Eindruck entstehen, dass der Spieler sich zufällig genau im richtigen Moment umgedreht hat. Je nach Situation sollte man die Blickaktivierung mit einer zeitlichen Verzögerung kombinieren: Schaut der Spieler nach einer gewissen Zeit immer noch nicht in die richtige Richtung, startet das Ereignis automatisch.

Diese Techniken sind selbst in scheinbar eindeutigen Situationen häufig sinnvoll. Sitzt der Spieler zum Beispiel in einem vollen Kino, könnte man als Entwickler davon ausgehen, dass

er stets auf die Leinwand schaut. In der Praxis sieht es allerding anders aus: Viele Spieler werden sich wahrscheinlich auch im Kinosaal umsehen und andere Gäste beobachten, vor allem solange der Film noch nicht begonnen hat. Wenn also etwas Wichtiges an der Leinwand des Kinos passiert, solltest du dem Spieler vorher ein Signal oder Zeichen geben, um ihm einen Anreiz zu geben, in die korrekte Richtung zu schauen. Je nach Situation solltest du passende Reize setzen, um das Interesse des Spielers zu wecken.

4.8 Uncanny Valley

Das Konzept des *Uncanny Valley* (dt. „unheimliches Tal") stammt bereits aus den 70ern und bezog sich damals in erster Linie auf Roboter. Damals beobachteten Forscher, dass ein Roboter sympathischer auf einen Menschen wirkt, je mehr menschliche Eigenschaften er aufweist. Unser Gehirn ist darauf trainiert, Formen zu analysieren und zu interpretieren, deswegen erkennen wir in den groben Umrissen von Wolken zum Beispiel Tiere und Gesichter. Wenn wir einen Roboter sehen, der ein paar menschliche Eigenschaften hat, rücken diese menschenähnlichen Eigenschaften in unserer Wahrnehmung in den Vordergrund. Sie wirken dann wie kleine tollpatschige Wesen, die uns an Kinder erinnern. Bild 4.24 demonstriert den Effekt an einem einfachen Beispiel. Ein etwas prominenteres Beispiel wäre der *Wall-E* aus dem gleichnamigen Pixar-Film.

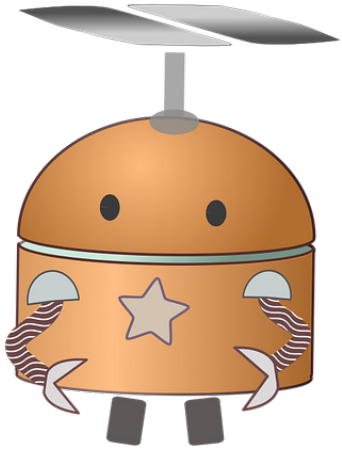

Bild 4.24 Obwohl dieser Roboter nicht sehr viele menschliche Eigenschaften hat, stechen die menschlichen Merkmale unweigerlich für uns heraus.

Die Forscher fanden heraus: Je menschlicher der Roboter, desto sympathischer wird er von den meisten Menschen aufgenommen. Doch die Sympathie hat eine Grenze: Übersteigt die optische Ähnlichkeit des Roboters ein gewisses Maß, hebt unser Gehirn nicht mehr die Ähnlichkeiten mit einem Menschen hervor, sondern konzentriert sich nur noch auf zahllose Unstimmigkeiten. Wir sehen jeden Tag Menschen und sind darauf trainiert, diese zu interpretieren. Ähnelt ein Roboter einem Menschen zu sehr, versuchen wir ihn unbewusst wie

einen anderen Menschen zu lesen. Solange der Roboter jedoch keine perfekte Kopie darstellt, funktioniert das nicht so richtig. Weil wir unfähig sind, Roboter wie den in Bild 4.25 richtig einzuschätzen, wirken sie unheimlich. Ihre bloße Anwesenheit erfüllt viele Menschen mit Unbehagen oder sogar Angst.

Bild 4.25 Werden die Roboter zu menschlich, erfüllen sie uns mit Unbehagen.

Erst wenn der Roboter von einem echten Menschen kaum zu unterscheiden wäre, würde dieses Gefühl verschwinden. Der Bereich, in dem ein Roboter so real wird, dass wir nur noch die fehlenden menschlichen Eigenschaften erkennen können und sie deshalb als unheimlich empfinden, ist das *Uncanny Valley* (dt. „unheimliches Tal"), welches in Bild 4.26 abgebildet wird.

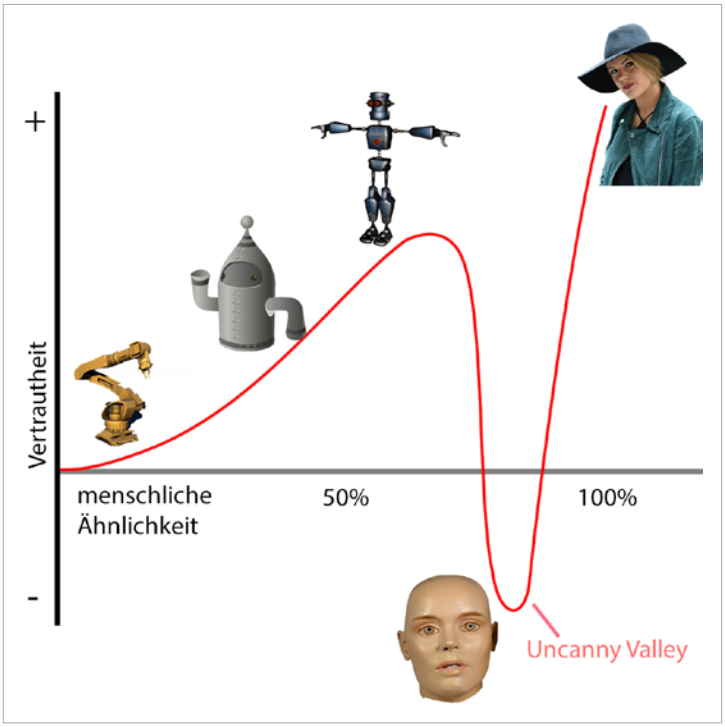

Bild 4.26 Der Weg zur perfekten menschlichen Darstellung führt durch das „unheimliche Tal“, welches man am besten vermeiden sollte.

Auch wenn die Ursprünge aus dem Bereich der Roboter stammen, ist das Phänomen überall zu finden, wo man versucht, Menschen zu imitieren; also auch in Animationsfilmen, Schaufensterpuppen, Videospielen und ganz besonders in Virtual Reality. Figuren, die wir in einem gewöhnlichen Videospiel akzeptieren würden, wirken in Virtual Reality plötzlich unheimlich, da die Welt realer wirkt und unser Gehirn deswegen höhere Ansprüche an die Darstellung der Menschen stellt.

Doch nicht nur die reine Optik lässt menschliche Darstellungen in Virtual Reality in das *Uncanny Valley* rutschen. Sehr wichtig für eine glaubhafte Darstellung sind auch natürliche Animationen, Mimik, Gestik und das Verhalten der virtuellen Personen. Selbst wenn man eine Figur fotorealistisch mit natürlicher Mimik und Gestik darstellen würde, könnte sie noch unheimlich auf den Spieler wirken; zum Beispiel, weil der Spieler versucht, mit ihr zu interagieren, und die virtuelle Person nicht plausibel darauf reagiert.

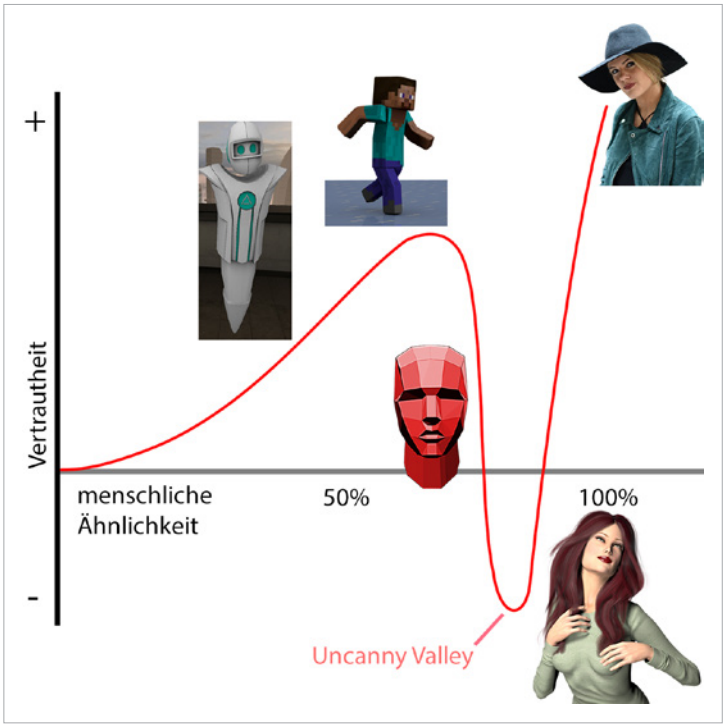

Bild 4.27 Beispielhaft das *Uncanny Valley* bei 3D-Modellen. Das Ziel ist hier eine fotorealistische Darstellung, die sich nicht von der Realität unterscheiden lässt.

4.8.1 Auswirkung für dich

Wenn du ein Virtual-Reality-Spiel entwickelst, musst du dich für einen der beiden Gipfel entscheiden. Dein Spiel sollte also idealerweise entweder einen abstrakteren oder comichaften Grafikstil haben oder sehr realistisch werden. Alles dazwischen wirkt unheimlich auf den Spieler. Als einzelner Entwickler ohne großes Budget empfiehlt es sich daher, einen einfacheren Grafikstil zu wählen, wenn du menschliche Figuren einbauen möchtest. Außerdem ist das Erstellen von stilisierten Assets für Anfänger erheblich einfacher, als direkt Modelle in AAA-Qualität modellieren zu wollen. Im Endeffekt liegt die Entscheidung natürlich bei dir. Bevor du ein großes Projekt startest, solltest du zunächst mit ein paar passenden Stilen herumprobieren und dich dann festlegen.

Zur Inspiration lohnt es sich, immer wieder einen Blick in die jeweiligen Stores zu werfen: Welche Stile gibt es, welche kommen gut bei den Spielern an und so weiter.

4.9 Sound Design und Spatial Sound

Geräusche, Musik und alles andere, was der Spieler hört, sind ein weiterer sehr wichtiger Faktor für die Immersion, der häufig unterschätzt wird. Wir sind es aus dem echten Leben gewohnt, dass es niemals völlig still ist und alles, was wir machen, gewisse Geräusche erzeugt. Soundeffekte sollte man deswegen niemals als optionalen Bonus ansehen. Für die perfekte Immersion sollte, wie im echten Leben, jede Bewegung, jede Kollision ein Geräusch bewirken. Außerdem kann Musik unterstützend verwendet werden, um den Spieler in die richtige Stimmung zu bringen.

4.9.1 Spatial Sound

Aber Sound ist nicht gleich Sound. Je realistischer der Sound, desto besser ist es für die Immersion. Unity unterstützt von sich aus Stereo- und Surround-Sound. Beide Soundvarianten sind für den Anfang in Ordnung, aber eigentlich nicht perfekt für Virtual Reality geeignet. Dieses Kapitel wird sich mit dem Warum und möglichen Alternativen beschäftigen.

Der Sound in Videospielen wird hauptsächlich von zwei Parametern beeinflusst: dem Abstand und der Position relativ zum Spieler. Bei gewöhnlichem Stereo-Sound wird je nach Abstand der Soundquelle die Gesamtlautstärke des Sounds angepasst. Zusätzlich wird basierend auf der Position der Soundquelle die Stereo-Balance entsprechend korrigiert. Das bedeutet: Befindet sich die Soundquelle links oder rechts vom Spieler, wird der Sound nur auf der jeweiligen Seite des Headsets abgespielt. Befindet sich die Soundquelle direkt vor oder hinter dem Spieler, wird der Sound auf beiden Seiten gleich laut abgespielt. Wenn die Soundquelle zwischen den genannten Positionen liegt, wird die Balance interpoliert (zum Beispiel 75% Lautstärke links und 25% rechts). Es kann auf diese Weise gut wahrgenommen werden, ob sich die Soundquelle links oder rechts von dem Spieler befindet. Eine Unterscheidung zwischen vorne und hinten ist allerdings nur begrenzt möglich. In erster Linie klappt dies zusammen mit der Optik: Hörst du etwas, das du nicht siehst, ist es wahrscheinlich hinter dir. Bild 4.28 zeigt beispielhaft für verschieden positionierte Soundquellen, wie sie auf einem Stereo-Headset wiedergegeben werden würden. Die Grafik berücksichtigt nur die Position relativ zur Blickrichtung des Spielers, nicht die Distanz.

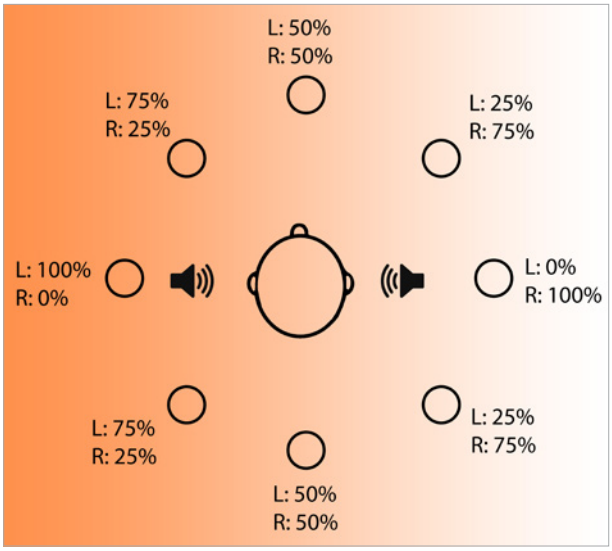

Bild 4.28 Vereinfachte Darstellung der Stereo-Balance je nach Position der Soundquelle im Spiel

Surround-Sound klappt auf eine ähnliche Art und Weise, nur dass es mehr Kanäle als nur „links“ und „rechts“ gibt. Dadurch ist es möglich, Geräusche tatsächlich vor oder hinter dir abzuspielen. Mit einer 5.1- oder 7.1-Surround-Anlage kannst du die Richtung eines Sounds schon relativ genau orten. Doch leider ist das Verwenden einer Surround-Anlage mit einem Virtual-Reality-Headset nicht so einfach möglich. Anders als bei klassischen Medien schaust du nämlich nicht immer in die gleiche Richtung (z. B. in Richtung des Fernsehers), sondern drehst dich ständig um, um dir die virtuelle Welt genau anzusehen. Und das ist ein Problem: Drehst du dich um 180°, wären alle Kanäle spiegelverkehrt, und Geräusche, die links von der Spielfigur abgespielt werden, würden aus dem rechten Kanal der Surround-Anlage kommen. Zwar könnte man dies über eine Software, welche das Head-Tracking überwacht, korrigieren, jedoch gibt es noch mehr Probleme. Ein besonders wichtiges ist, dass die Mehrheit der Spieler heutzutage keine Surround-Anlage am PC verwendet, sondern Kopfhörer. Im Mobile-VR-Bereich wäre die Lösung auch nicht wirklich praktisch und würde eine Surround-Anlage mit Bluetooth erfordern. Die Zielgruppe eines VR-Spiels, das auf Surround-Anlagen ausgelegt ist, weil die Ortung von Geräuschen im Spiel besonders wichtig ist, wäre also relativ klein.

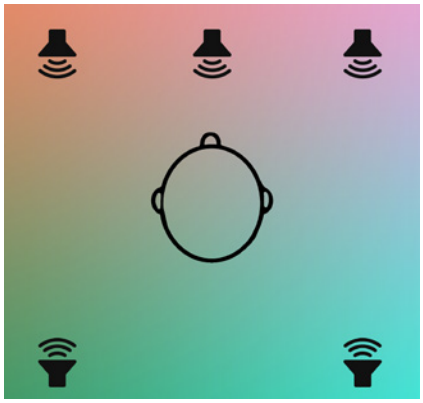

Bild 4.29 Lautsprecherverteilung in einem typischem 5-Kanal-Surround-System

Die Lösung für möglichst realistischen Sound in Virtual Reality nennt sich *Spatial Audio* und wird manchmal auch als „Binaural Audio“ bezeichnet. Diese Technik funktioniert mit jedem Stereo Headset, wodurch jeder Spieler es verwenden kann, ohne dass er sich neue Hardware oder Software kaufen muss. *Spatial Audio* muss allerdings von dir als Entwickler in dein Spiel eingebaut werden und funktioniert nicht automatisch.

Aber wie funktioniert *Spatial Audio*? Dafür müssen wir berücksichtigen, wie wir in der Realität hören. Im echten Leben orten wir Geräusche nicht nur über die Lautstärke („Ist es lauter links als rechts?“), sondern vollkommen unterbewusst auch darüber, wie das Geräusch klingt. Kommt ein Geräusch zum Beispiel von hinten rechts, klingt das Geräusch aufgrund der Form unserer Ohrmuschel minimal anders, als wenn es von vorne kommen würde. Zudem braucht ein Geräusch, wenn es von rechts kommt, ein klein wenig länger zum linken Ohr als zum rechten. Diese minimalen Unterschiede in Klang und Zeit verarbeitet unser Gehirn sofort und sagt uns, dass das Geräusch beispielsweise von hinten rechts kam.

Verwendet man einen Kopfhörer, gehen alle Geräusche sofort ins Ohr, wodurch diese Zeit- und Verzerrungsinformationen nicht vorhanden sind. Genau hier kommt dann *Spatial Audio* ins Spiel. Bild 4.30 veranschaulicht nochmals, welche Parameter für die Ortung eines Geräusches im echten Leben wichtig sind.

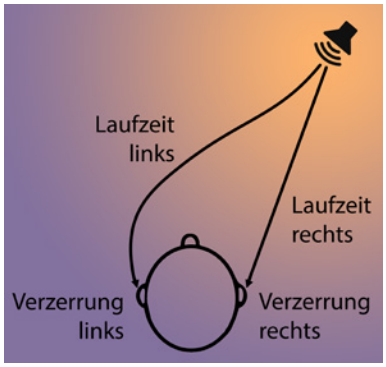

Bild 4.30 Vereinfachte Darstellung der Funktionsweise der akustischen Ortung

Spatial Audio nutzt eine sogenannte „Head Related Transfer Function“ (kurz *HRTF*), um diese minimalen Verzerrungen und Verzögerungen zu simulieren. Dadurch ist möglich, die Richtung einer Soundquelle im Spiel sehr genau zu bestimmen, da man bei der akustischen Simulation des Sounds nicht auf die tatsächlichen Kanäle angewiesen ist.[3]

Jeder Mensch hat eine andere Kopf- und Ohrform. Deswegen müsste eine ideale *HRTF* auch bei jedem Menschen andere Parameter aufweisen. Mangels einer Möglichkeit, alle benötigten Parameter für das eigene Ohr zuverlässig zu ermitteln, arbeiten alle aktuellen Spatial-Audio-Engines aber mit einer allgemeinen Funktion, welche quasi dem Durchschnittsohr entspricht. Dies reicht aber bereits vollkommen aus, um unser Gehirn auszutricksen.

Bei *Spatial Audio* werden zusätzlich zur Ortung auch noch andere Eigenschaften der virtuellen Welt mit einberechnet: Bewegt sich die Soundquelle auf die Spielfigur zu oder entfernt sie sich (Dopplereffekt)? Befindet sich der Spieler in einer kleinen Besenkammer, auf freiem Feld oder in einer riesigen, hallenden Kirche? All diese Parameter bestimmen, wie das Geräusch im Endeffekt klingt, und erzeugen einen sehr plausiblen Klang.

4.9.2 Spatial-Audio-Engines

Unity kann ohne zusätzliche Hilfsmittel kein Spatial-Audio ausgeben. Dafür bietet es Entwicklern aber eine Möglichkeit, die Unity-interne Audio-Engine zu erweitern, was von verschiedenen Entwicklern verwendet wird, um Add-ons für *Spatial Audio* zu entwickeln. Da *Spatial Audio* in Virtual Reality sehr wichtig ist, bietet mittlerweile jeder der drei großen Hersteller (*Google*, *SteamVR* und *Oculus*) zusätzlich zu den Entwicklerwerkzeugen für die VR-Brille auch ein eigenes *Audio SDK* an. Bei Google ist dies in dem normalen GoogleVR Kit integriert, bei SteamVR und Oculus sind es jeweils eigene Unity-Packages.

Doch du musst nicht für jede Plattform ein anderes SDK entwickeln, denn alle drei *Audio SDKs* sind plattformunabhängig. Das bedeutet, obwohl es z. B. von Valve für SteamVR entwickelt wurde, kannst du das *SteamVR Audio SDK* auch mit allen anderen VR-Brillen verwenden, inklusive mobiler Brillen wie GearVR und GoogleVR. Dasselbe gilt auch für das *Oculus Audio SDK* und die Lösung von *Google*.

Ich habe dir in dem Kasten weiterführende Links für mehr Informationen zu den Audio SDKs zusammengefasst. Grundsätzlich würde ich dir empfehlen, dich erst mit dem Unity-internen Audio-System vertraut zu machen, da die Audio-SDKs auf diesem aufbauen. In diesem Buch werden wir uns auf das Unity-interne Audio-System beschränken. Sobald du aber vertraut im Umgang von Unity und VR bist, solltest du dir die SDKs für dein Spiel einmal genauer ansehen. In dem Kasten habe ich dir zu jedem der SDKs einen Link zu detaillierten Informationen hinterlegt.

[3] Virtual-Surround-Headsets funktionieren übrigens auf die gleiche Art und Weise und nutzen eine HRTF, um das Surround-Sound-Signal für Stereo-Headsets umzurechnen.

GoogleVR Audio SDK

https://developers.google.com/vr/concepts/spatial-audio

SteamVR Audio SDK

https://valvesoftware.github.io/steam-audio/

Oculus Audio SDK

https://developer.oculus.com/documentation/audiosdk/latest/concepts/book-audiosdk/