UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Effiziente Navigation in virtuellen Szenen

Diplomarbeit

zur Erlangung des Grades einer Diplom-Informatikerin
im Studiengang Computervisualistik

vorgelegt von
Adriane Niepel

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	Dipl. Inf. Rodja Trappe (Institut für Computervisualistik, AG Computergraphik)

Koblenz, im Dezember 2007

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....
(Ort, Datum) (Unterschrift)

Institut für Computervisualistik
AG Computergraphik
Prof. Dr. Stefan Müller
Postfach 20 16 02
56 016 Koblenz
Tel.: 0261-287-2727
Fax: 0261-287-2735
E-Mail: stefanm@uni-koblenz.de

UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

Aufgabenstellung für die Diplomarbeit
von Frau Adriane Niepel
(Matr.-Nr.: 202121016)

Thema: **Effiziente Navigation in virtuellen Szenen**

Während der rasanten Entwicklung der Computergrafik und für die Nutzung virtueller Welten wird das Navigieren zunehmend wichtiger, aber auch komplexer. Je nach Aufgabengebiet und Zielgruppe gestalten sich die Anforderungen für eine effiziente Navigation auf verschiedenste Art. Hierzu gibt es bereits einige Navigationsmetaphern, die sich in ihrem Umfeld bewährt haben.

Diese gilt es noch genauer zu erforschen und für die unterschiedlichsten Einsatzgebiete anwendbar zu machen.

Diese Arbeit macht sich zur Aufgabe, das effiziente Navigieren in virtuellen Szenen näher zu betrachten.

Hierzu wird eine Umgebung geschaffen in der sich der Benutzer auf einige unterschiedliche Arten bewegen kann. Dies wird ihm durch Standard-Desktop-Eingabegeräte ermöglicht. Die Navigationsmöglichkeiten orientieren sich dabei an bereits bewährten Metaphern.

Die virtuelle Szene beschränkt sich im Folgenden auf einen Gebäudekomplex, welches durch einfaches 3D-Rendering dargestellt wird.

In den abschließenden Benutzertests sollen sich die Testpersonen zu bestimmten Orten begeben oder Gegenstände auffinden. Auf diese Art und Weise lässt sich die Effizienz der betrachteten Navigationsmöglichkeiten ableiten.

Die inhaltlichen Schwerpunkte der Arbeit sind:

1. Untersuchung bekannter Navigationsmöglichkeiten
2. Entwicklung einer Testumgebung
3. Einbindung einiger ausgewählter Navigationsmöglichkeiten
4. Benutzertests im Hinblick auf die Effizienz
5. Dokumentation der Ergebnisse

Koblenz, den 15.05.2007

Betreuer: Dipl.-Inf. Rodja Trappe

S. Müller

– Prof. Dr. Stefan Müller –

Inhaltsverzeichnis

1. Einführung **1**
- 1.1. Problemstellung . . . 1
- 1.2. Zielsetzung und Motivation . . . 2

2. Grundlagen **5**
- 2.1. Virtuelle und augmentierte Realität . . . 5
- 2.2. Renderingverfahren . . . 8
- 2.3. Navigation und Interaktion . . . 8
- 2.4. Benutzerperspektiven . . . 11

3. Stand der Technik **17**
- 3.1. Navigationsmetaphern . . . 18
  - 3.1.1. Laufen-Metapher . . . 19
  - 3.1.2. Pfad-Metapher . . . 19
  - 3.1.3. Teleport-Metapher . . . 28
  - 3.1.4. weitere Navigationsmetaphern . . . 29
  - 3.1.5. Anwendungsgebiete . . . 34
- 3.2. Navigationsunterstützende Verfahren . . . 43
  - 3.2.1. Aktive Verfahren . . . 44
  - 3.2.2. Passive Verfahren . . . 48

4. Testszenario **65**
- 4.1. Analyse der Möglichkeiten . . . 66
- 4.2. Auswahl . . . 70
  - 4.2.1. Navigationsmetaphern: . . . 71
  - 4.2.2. Navigationsunterstützende Verfahren: . . . 73
- 4.3. Anforderungen . . . 75

5. Technische Umsetzung **79**
- 5.1. Auswahl der Werkzeuge . . . 79
  - 5.1.1. Modellierung . . . 80
  - 5.1.2. Rendering . . . 82
  - 5.1.3. Kollisionserkennung . . . 82

5.1.4. Eingabegeräte 83
5.2. Software-Architektur 84
5.3. Benutzerperspektiven 86
5.4. Navigationsmetaphern 87
5.4.1. Laufen-Metapher 87
5.4.2. Pfad-Metapher 87
5.4.3. Teleport-Metapher 92
5.5. Navigationsunterstützende Verfahren 93
5.5.1. Aktive Verfahren 93
5.5.2. Passive Verfahren 94
5.6. Integration aller Bausteine 98

6. Benutzertests **101**
6.1. Hypothesen 101
6.1.1. Navigationsmetaphern und Benutzerperspektiven . 101
6.1.2. Navigationsunterstützende Verfahren: 104
6.2. Aufbau der Testumgebung 105
6.3. Testablauf 107
6.4. Ergebnisse 110
6.4.1. Navigationsmetaphern und Benutzerperspektiven . 110
6.4.2. Navigationsunterstützende Verfahren 115

7. Fazit **121**
7.1. Bewertung 121
7.2. Zusammenfassung 122
7.3. Ausblick 124

A. Umfragebogen **129**

B. CD-ROM **135**

Abbildungsverzeichnis

3.1. Pfad-Metapher: Feste Blickrichtung 20
3.2. Pfad-Metapher: Blickrichtung entlang eines Pfades inkl. Rotation um die y-Achse 20
3.3. Pfad-Metapher: Blickrichtung entlang eines Pfades inkl. Rotation um alle Achsen 21
3.4. Pfad-Metapher: Vollautomatischer Pfad 22
3.5. Pfad-Metapher: Führungsperson 24
3.6. Pfad-Metapher: Inspektionspfad (Orbiting) 26
3.7. Pfad-Metapher: dynamische Pfaderstellung 27
3.8. Speed coupled flying-Metapher 31
3.9. Inverse Scaling-Metapher 32
3.10. Object Manipulation and Ghost Copy-Metapher 33
3.11. Verknüpfungen: Genres, Navigationsmetaphern, Benutzerperspektiven 35
3.12. Genre: Shoot'em-up 36
3.13. Genre: Beat'em-up 37
3.14. Genre: Jump'n Run 38
3.15. Genre: Rennspiele 39
3.16. Genre: Rollenspiele 39
3.17. Genre: Strategiespiele 40
3.18. Genre: Adventures 41
3.19. Kamerasteuerung: automatischen Kameraführung mit Hilfe der Snake-Metapher 45
3.20. Grafische Abstraktion 51
3.21. Grafische Abstraktion durch Interaktion des Benutzers . . . 52
3.22. Schattendarstellung 53
3.23. Anwendungsbeispiele: Gitterdarstellungen 54
3.24. Anwendungsbeispiele: Koordinatendarstellungen 55
3.25. Anwendungsbeispiele: Übersichtskarten 59
3.26. Besondere Hervorhebung: Pfeil und Spotlight 60
3.27. Anwendungsbeispiel: Besondere Hervorhebung 61
3.28. Besondere Hervorhebung: Snake 61

3.29. Anwendungsbeispiel: Transparenz 64

5.1. Modelle der Szene 81
5.2. Softwarearchitektur 84
5.3. Szenengraph 85
5.4. Umsetzung: Benutzerperspektiven 86
5.5. Umsetzung: Inspektionspfad (Orbit-Metapher) 88
5.6. Funktionsweise: Inspektionspfad (Orbit-Metapher) 89
5.7. Umsetzung: Vollautomatischer Pfad 90
5.8. Funktionsweise: Vollautomatischer Pfad 92
5.9. Umsetzung: Transparenz 95
5.10. Umsetzung: Multiple Fenster 96
5.11. Umsetzung: Übersichtskarte 97

6.1. Aufbau der Testumgebung 106
6.2. Durchschnittszeiten bei der Explore-Aufgabe 112
6.3. Durchschnittszeiten abhängig der Übersichtskarte 117

1. Einführung

Die effiziente Navigation in virtuellen Szenen stellt kein triviales Problem dar. Vor dieser Problematik stehen die Entwickler vor jeder Umsetzung einer Anwendung in einer virtuellen Umgebung. Die Betrachtung besteht dabei aus verschiedenen Aspekten, die das Produkt anwenderfreundlich machen. Hierzu gehört nicht nur die Wahl der Art der Navigation, sondern zum Beispiel auch die Wahl der Benutzerperspektive. Wie ergonomisch die Umsetzung der Entscheidung über die einzelnen Bausteine ist, kann nur durch Benutzertests entschieden werden.

Diese Punkte werden in der vorliegenden Arbeit betrachtet und erläutert. Angefangen von der Problemstellung und Zielsetzung (Abschnitt 1.1) über Grundlagen (Kapitel 2) und den heutigen Stand der Navigationsmöglichkeiten (Kapitel 3) bis hin zur praktischen Umsetzung. Die Umsetzung besteht dabei aus der theoretischen Konzeption des Szenarios (Kapitel 4), der praktischen Implementierung (Kapitel 5) und der Durchführung von Benutzertests inklusive Auswertung dieser (Kapitel 6). Zuletzt schließt Kapitel 7 mit einem Fazit der Arbeit und dem Ausblick für weitere Forschungs- und Entwicklungsideen.

1.1. Problemstellung

Aufgrund der kurzen Historie des Computers befindet sich die Entwicklung von Hard- und Software noch immer in einem Stadium mit rasantem Fortschritt. Durch die Einbindung des Computers in das tägliche Leben wird die Kommunikation zwischen Mensch und Maschine zunehmend wichtiger. Diese Kommunikation stellt in jeder Anwendung eine neue Herausforderung dar, da die optimale Anpassung zum Erfolg oder Misserfolg der Software maßgeblich beiträgt. Betrachtet werden kann dies sowohl aus technischer Sicht (Hardware), als auch aus Programmsicht (Software).

Mit Hilfe der Hardware werden Schnittstellen zwischen Mensch und Maschine hergestellt. Zur Umsetzung gibt es viele Möglichkeiten. Angefangen

bei Standard-Desktop-Eingabegeräte (z. B. Maus und Tastatur), über spezielle Eingabegeräte (z. B. *Spacemouse*, *Phantom*) bis hin zu Eingabegeräten, die *Augmented Reality* unterstützen und besondere Tracker für die Position benutzen (z. B. Datenhandschuhe). Die beste Schnittstelle für die jeweilige Anwendung zu finden ist, in Hinblick auf die Hardware, eine große Herausforderung.

Die Anforderung an die Software besteht daraus, so auf den Benutzer zu reagieren, wie er es erwarten würde und somit eine möglichst intuitive Kommunikation zu ermöglichen. Diese besteht je nach Anwendung aus Interaktion und/oder Navigation. Mit Hilfe der bereits existierenden Softwareprodukte und deren Evaluationen haben sich verschiedene Metaphern in den verschiedenen Anwendungsgebieten bereits als mehr oder weniger effizient erwiesen.

1.2. Zielsetzung und Motivation

Im Folgenden sollen Anwendungen betrachtet werden, deren Hauptaugenmerk auf der Benutzung virtueller Welten liegt. Für diese Art von Anwendungen ist die Navigation besonders ausschlaggebend. Anhand dieser entscheidet ein Benutzer sehr schnell, ob er sie weiterhin verwendet oder nicht. Vor allem 3D-Spiele sind abhängig von einer guten Navigation, um den Nutzer nicht zu „verlieren“. Virtuelle Welten finden inzwischen jedoch auch in Lernexperimenten und Montageanleitungen Anklang, in denen vor allem auch *Augmented Reality* eine große Rolle spielt. Durch Vergleiche verschiedener bestehender Navigationsmöglichkeiten mit verschiedenen Zielgruppen zu verschiedenen Aufgabenstellungen ist es möglich, eine gute Wahl für eine Navigationsmetapher zu treffen, die den Benutzer unterstützt. Dabei spielt die Komplexität der Anwendung eine zusätzliche Rolle. Diese orientiert sich dabei an der Zielgruppe und dem Aufgabengebiet.

Diese Arbeit macht sich zur Aufgabe, das effiziente Navigieren in virtuellen Szenen näher zu betrachten, um bereits bestehende Navigationsmetaphern genauer zu erforschen und für unterschiedliche Einsatzgebiete anwendbar zu machen. Betrachtet werden Softwareprodukte für durchschnittliche Desktop-Benutzer, die mit Standard-Desktop-Eingabegeräten arbeiten.

Hierzu werden zunächst bestehende Navigationsmetaphern analysiert sowie Möglichkeiten der Benutzerperspektive und anderen Hilfsmitteln mit Hinblick auf ihr aktuelles Anwendungsgebiet diskutiert.

Anschließend wird ein theoretisches Testszenario erstellt, in dem alle

analysierten Metaphern, betrachtet und verglichen werden können. Ein Teil dieses Gesamtszenarios wird praktisch umgesetzt, in eine Testumgebung integriert und anhand von Benutzertests evaluiert.

Zuletzt werden Möglichkeiten aufgezeigt, inwiefern die betrachteten Navigationsmetaphern für verschiedene Einsatzgebiete sinnvoll und effizient anwendbar gemacht werden können.

2. Grundlagen

Ohne ein Verständnis für die einzelnen Bausteine, die eine Anwendung in beschriebenem Rahmen benötigt, bleiben eventuell anfängliche Fragen bis zuletzt offen. Um dem vorzubeugen, werden zunächst einige Grundlagen erläutert. Hierunter fallen besonders auch Begriffserklärungen, um vor allem Missverständnissen vorzubeugen. Die Grundlagen schaffen ein Verständnis für benötigte Bestandteile, wie unter anderem was eine virtuelle Szene ausmacht und was Navigation in dieser bedeutet.

2.1. Virtuelle und augmentierte Realität

> „VIRTUAL REALITY WORKS BECAUSE REALITY IS VIRTUAL."
> Professor Lawrence Stark, University of California, Berkley [SSC02]

Der Begriff virtuelle Szene wird im Folgenden genauer definiert, um zu verstehen in welcher Umgebung und unter welchen Voraussetzungen die Navigation in dieser Arbeit betrachtet wird. Hierzu ist es wichtig zu verstehen, was die Bezeichnung *virtual reality* (VR) bedeutet.

Das Grundkonzept besteht daraus, simulierte dreidimensionale Welten zu erschaffen, in die der Benutzer einbezogen wird (*Immersion*). Das bedeutet, der Benutzer soll eine künstlich geschaffene Welt als möglichst real wahrnehmen [Her06]. Die Welt soll dabei nicht statisch sein, sondern dynamisch auf die Eingaben des Benutzers reagieren [BC03]. Diese Art des Eingriffes eines Benutzers in eine künstliche Welt wird auch *mixed reality* genannt. Unter diesen Begriff wird zusätzlich auch die *augmented reality* gefasst, welche einen Schritt weiter geht als die virtuelle Realität. Die Immersion wird um ein Vielfaches erhöht, indem „die gerade wahrgenommene Realität dynamisch um digitale Informationen ergänzt" [Her06] wird.

Der Erfolg solcher Anwendungen hängt nach Burdea und Coiffet [BC03] von folgenden drei Aspekten ab:

1. Interaktion 2. Immersion 3. Vorstellungsvermögen des Menschen

Das entspricht genau dem, was Stark aussagen will[1], indem er erklärt, dass jeder Mensch die Realität anders wahrnimmt. Darum ist die gesehene Realität nur eine Illusion des Einzelnen. Die virtuelle Realität stellt dabei **eine** mögliche Sicht in eine Welt dar und ist somit auch „nur" Illusion [SSC02]. Die virtuelle Welt ist zusätzlich eine Illusion, die auch mit mehreren Menschen geteilt werden kann [SC03].

Damit also eine virtuelle Anwendung erfolgreich ist, muss sich der Mensch fühlen wie in der realen Welt. Er muss mit ihr interagieren können, er muss empfinden, als wäre er ein Teil der Welt und er muss ein gewisses Vorstellungsvermögen besitzen, diese Illusion in seinem Gehirn aufzubauen. Das Level der Immersion beeinflusst den Benutzer somit in seiner Leistung [TRC01].

Sind diese Punkte nicht gut integriert, kann es unter anderem zu *Cybersickness*[2] kommen. Diese entsteht zum Beispiel durch zu schnelle Bewegungen im virtuellen Raum, die der Benutzer vor dem Bildschirm nicht schnell genug verarbeiten kann [BC03].

Eine virtuelle Szene, wie sie im Folgenden verstanden werden soll, beschreibt ausschließlich eine virtuelle und keine augmentierte Realitäts-Anwendung. Die Szene wird also künstlich erstellt und entspricht nicht der gerade wahrgenommenen Realität. Dennoch sollte augmentierte Realität der Vollständigkeit halber mitbetrachtet werden, da die Navigation in beiden Bereichen wichtig ist und sich ähnlich anwenden lässt.

Die Aufgabengebiete und Zielgruppen virtueller und augmentierter Realitäts-Anwendungen nehmen stetig zu. Hierunter finden sich die verschiedensten Gebiete der Wissenschaft und Forschung, aber auch Entertainment- und Informationsvermittlungsbereiche.

Einige Beispiele für aktuelle Anwendungsfelder nach [Her06]:

Virtuelle Realität:

- Architektur (z. B. virtuelle Begehungen, Modellierungen)
- Konstruktion (Betrachtung und Handhabung, vor allem Pkws und Flugzeuge)
- Installationen in Freizeitparks (Phantasiewelten)

[1] „Virtual reality works because reality is virtual." [SSC02]
[2] Weiterführende Literatur [BC03]

- Spiele
- Museumsinstallationen

Augmentierte Realität:

- medizinische Anwendungen
- militärische Informationssysteme

Werden nun die verschiedenen Anwendungsmöglichkeiten betrachtet, so fällt auf, dass je nach Bereich, verschiedenste Zielgruppen angesprochen werden. Dieses Verständnis ist wichtig, da die Navigation, vollkommen verschiedenen Anforderungen gerecht werden muss. Jedoch macht keine Anwendung mit einer 3D-Szene Sinn, wenn in ihr nicht mindestens navigiert werden kann. Je größer die Szenen werden, umso wichtiger wird auch eine gute Navigation [TRC01].

Diese Arbeit spezialisiert sich auf virtuelle Szenen, in denen ein Charakter (*Avatar*) den Benutzer in der künstlichen Welt darstellt und für ihn agiert. Die Szenerie kann für verschiedene Gebiete betrachtet werden. Es sind vor allem Spiele, aber auch Museumsführungen oder ähnliches, die auf diese Weise dargestellt werden können.

Die Aufgaben für eine solche Szenerie können ganz unterschiedlich sein. Vor allem sind Bereiche wie zum Beispiel Spiele nicht allgemein fassbar. Sie haben je nach Genre andere Ziele und benötigen andere Kommunikationsmöglichkeiten zwischen Mensch und Maschine.

Eine Auflistung gängiger Spielegenres, die in virtuellen Szenen dargestellt werden können, finden sich vielfach. Eine Auswahl derer findet sich in [Sta02] und sollen soweit als Einblick dienen:

- *Shoot-'em-ups*
- *Beat'em-ups*
- Rennspiele
- Strategiespiele
- Adventures
- Rollenspiele

Einen tieferen Einblick in die einzelnen Spielegenre wird in Abschnitt 3.1.5 gegeben, indem sie im Zusammenhang mit den Benutzerperspektiven und Navigationsmöglichkeiten betrachtet werden.

2.2. Renderingverfahren

Um eine virtuelle Szene darstellen zu können, muss sie gerendert werden. Dies kann auf verschiedene Weisen geschehen: Über direkte Befehle wie zum Beispiel *OpenGL* oder über *Grafik-Engines*, die für größere Anwendungen ausgelegt sind und bereits Funktionen mitbringen, die nicht aufwendig „per Hand“ nachgestellt werden müssen. Entsprechend ist eine höhere Leistung erreichbar, in dem die vorhandene Klassen und Funktionen verwendet werden können, die wesentlich effizienter programmiert sind, als es für einen einzelnen Entwickler in kurzer Zeit möglich wäre.

Sehr wichtig bei fremden *Engines* ist es, dass es entsprechende Dokumentation und Ansprechpartner gibt. Dazu dienen zum Beispiel Foren, in denen ein schneller Austausch mit anderen Entwicklern möglich ist, um gemeinsam effiziente Lösungen zu finden. Zusätzlich sollte es Einstiegstutorials oder/und Bücher geben, um auch alleine mit dem fremden *Framework*[3] klar zu kommen.

2.3. Navigation und Interaktion

Wird der Begriff Navigation betrachtet, so stellt sich die Frage:
Ist Navigation nicht auch nur eine Interaktion?

Nach Michael Herczeg definiert sich Interaktion wie folgt:
„Interaktionsformen sind stereotype Abläufe zur Ein- und Ausgabe von Informationen, die Eingabe- mit Ausgabetechniken in definierten Abfolgebedingungen verknüpfen.“[Her06]

Demnach ist jede Form des Eingreifens des Benutzers, also einer Mensch-Maschine-Kommunikation, eine Interaktion. Von dieser Definition ausgehend beschreibt Navigation eine bestimmte Form der Interaktion. Dennoch soll hier ganz deutlich zwischen Interaktion und Navigation unterschieden werden.

Im Folgenden soll Interaktion als Einwirken auf Objekte in der virtuellen Welt verstanden werden, beziehungsweise **„das wechselseitige aufeinander Einwirken von Akteuren“** [Int07] und Systemen.

Navigation hingegen wird von Chittaro und Scagnetto in [CS01] wie folgt definiert:

[3] System, Rahmen

„navigation can be informally defined as the process whereby people determine where they are, where everything else is, and how to get to particular objects or places“
oder wie auf Wikipedia zu finden auf den Punkt gebracht:
„das sich Zurechtfinden in einem geografischen Raum, um einen gewünschten Ort zu erreichen.“ [Nav07]

Das Navigieren in diesem Sinne besteht nach [Nav07] aus:

1. aktueller Positionsbestimmung
2. festlegen der Zielposition
3. Berechnung der Wegstrecke zum Ziel
4. Führen des Charakters und/oder der Kamera über die Wegstrecke zum Ziel

Die Navigation setzt sich demnach aus Position, Geschwindigkeit und Blickrichtung zusammen [dSGA+00].

Zusätzlich zu beachten ist, dass eine dreidimensionale Navigation wesentlich mehr Schwierigkeiten aufzeigt als eine Zweidimensionale. Durch die Tiefenwirkung wird der Benutzer in die Szene integriert und erlebt die Welt räumlich. Daher muss auch die Eingabemöglichkeit des Akteurs betrachtet werden. Diese können Standard-Desktop-Eingabegeräte wie Maus und Tastatur, oder aber besondere Geräte, wie *Spacemouse*, Datenhandschuhe und ähnliches sein. Verschiedene Navigationsmöglichkeiten lassen sich mit allen Eingabegeräten umsetzen, manche sind sehr speziell und benötigen bestimmte Eingabemöglichkeiten, vor allem gilt dies für Anwendungen die mit augmentierter Realität arbeiten. Hier werden oft Informationen benötigt, die eine Standard-Maus nicht mehr liefern kann, zum Beispiel spezielle Handbewegungen an die Anwendung zu übertragen.

Da in dieser Arbeit ausschließlich VR-Anwendungen betrachtet werden sollen, werden die Eingabegeräte auf die Standard-Desktop-Geräte, Maus und Tastatur, beschränkt. Die im Folgenden zu betrachteten Navigationsmetaphern können jedoch auch mit anderen Geräten, sowie durch anderes abfragen derselben Geräte, erfolgen.

Mit diesem Verständnis von Navigation und dem vorangegangenen Ab-

schnitt fällt unter Betrachtung zweier Beispiel-Anwendungen auf, dass Navigation nicht gleich Navigation ist.

Ein Beispiel für VR-Anwendungen aus dem Entertainment-Bereich sind 3D-Computerspiele. Hier besteht die Zielgruppe vor allem aus Jugendlichen und die Navigation unterliegt meist physischen Gesetzen. Dies ist sicherlich der Großteil der Anwendungen, die sich zurzeit im Umlauf befinden.

Im Gegensatz dazu spielt aber auch in der Forschung und Entwicklung die Möglichkeit eine virtuelle Umgebung zu nutzen eine immer größere Rolle. Hier entstehen Prototypen (z. B. eines PKWs oder einer Architektur) am Computer. Diese sollen in virtuellen Szenen betrachtet werden können. Bei der Architektur geht es vor allem auch um die Betrachtung der Innenräume. Da keine Bewegung der Umgebung stattfindet, diese also statisch angenommen wird, spielen physische Gesetze kaum eine Rolle. Die Zielgruppe besteht hierbei nicht aus normalen Nutzern, sondern aus Entwicklern und Designern.

Die Navigation in diesen Anwendungsgebieten obliegt vollkommen unterschiedlichen Voraussetzungen und Umsetzungsmöglichkeiten. Somit ist eine Unterscheidung nach Anwendungsgebiet, Zielgruppe und ähnlichen Aspekten wichtig, um sich für die „richtige“ Navigation in der entsprechenden Szene zu entscheiden.

Welche Aufgaben die Navigation erfüllen muss, hängt dabei von der entsprechenden Anwendung ab [dSGA+00]. Denkbar sind Aufgaben wie:

- Die Umgebung erkunden: *Explore*-Aufgaben:
  Der Benutzer soll sich in einer ihm fremden Umgebung orientieren und zurechtfinden.
- An einen bekannten Ort zurückkehren: *Go-back-to-known-target*-Aufgaben:
  Der Benutzer soll in einer ihm bekannten Umgebung an einen speziellen Ort, den er kennt zurückkehren.
- Nach Objekten suchen: *Search*-Aufgaben:
  Der Benutzer soll in einer ihm bekannten Umgebung nach Objekten an unbekannten Orten suchen.
- Einzelne Objekte betrachten: *Inspect*-Aufgaben:
  Der Benutzer soll sich ein Objekt genau anschauen.

Natürlich werden diese Aufgaben in Anwendungen vermischt und der Benutzer muss unter Umständen mehrere Aufgaben gleichzeitig erfüllen, dennoch können diese Aufgaben getrennt betrachtet werden. Beispielsweise in einem Egoshooter, in dem es darum geht seine Widersacher aufzuspüren und sie zu bekämpfen, muss sich der Benutzer in einer ihm fremden Umgebung zurechtfinden und nach den Gegnern suchen. Im der Regel gibt es spezielle Gegenstände, die sich der Benutzer anschauen kann, beispielsweise Waffen oder sogenannten *Health-Packages*. Sollte er diese in einem Moment nicht benötigen, so kann er jederzeit wieder zu diesem ihm bekannten Ort zurückkehren, um diese aufzusammeln und zu benutzen.

2.4. Benutzerperspektiven

Für die Navigation spielt es eine entscheidende Rolle aus welcher Perspektive der Benutzer die Welt wahrnimmt. Je nachdem ist die Anwendung mehr oder weniger immersiv. Dies ist für manche Aufgaben wichtiger als für andere. Die Wirkung der Perspektive und die Unterstützung für den Akteur sollten unter keinen Umständen unterschätzt werden.

Verschiedene Perspektiven bieten natürlich auch verschiedene Vorteile und sollten in entsprechenden Situationen eingesetzt werden. Dies lässt schon darauf schließen, dass es nicht **die** Perspektive für alle Anwendungen gibt. Im Folgenden sollen die gängigen Perspektiven und ihre Wirkung auf den Benutzer inklusive Anwendungsgebiete deutlich gemacht werden.

1. Die Egoperspektive (auch: *first-person view*)
In dieser Perspektive befindet sich die Kamera in dem Spielercharakter. Er sieht durch seine Augen und ermöglicht so einen hohen Grad der Immersion. Der Benutzer sieht die Spielfigur nicht, ausgenommen Arme, Beine oder Gerätschaften, die der Avatar in den Händen hält. Diese Perspektive wird vor allem in Shootern eingesetzt [Ego07]. Der Spieler erlebt mit dieser Möglichkeit die Umgebung selber und erforscht die Umgebung, sozusagen mit „seinen eigenen Augen“. Alles Erlebte wirkt subjektiv auf den Benutzer [VS99].

Er hat dabei die Möglichkeit sich selber zu spielen, er ist kein Superheld, sondern er selber. Der Avatar symbolisiert ihn nur, besitzt aber selber keinen Charakter. Das Personalisieren seiner Spielfigur bleibt dem Spieler und seiner Vorstellung überlassen. Natürlich kann er sich auch vorstellen er sei ein Superheld, aber es gibt keine vorgegebene Personalität, die durch eine Welt geführt wird [Rou99].

2. Die Drittpersonansicht (auch: *third-person view*)
In dieser Perspektive befindet sich die Kamera außerhalb des Spielercharakters und der Benutzer kann die virtuelle Figur, die ihn repräsentiert, sehen. Dieser Modus wird oft in Adventures, Rollenspielen oder Strategiespielen genutzt, da sie eine bessere Übersicht über das Terrain bietet. Sie vermittelt dem Spieler jedoch weniger Immersion, da der Benutzer immer außerhalb steht. Dafür sieht der Spieler jedoch Aktionen und Bewegungen seiner Spielfigur. Bei besonderen Bewegungen ist es sicherlich auch ein Ansporn, wenn diese beobachtet werden können [Dri07]. Unter Umständen es ist sogar ein Muss die Bewegung betrachten zu können, wenn es sich zum Beispiel um Sport-Lernprogramme handelt. Der Betrachter studiert also nicht sich selber, sondern die Bewegungen des Charakters. Es ist eine Art objektive Betrachtungsweise von außerhalb des Geschehens [VS99].

Dies bringt vor allem einen Vorteil für die Programmierer. Sie können einen Charakter formen, der ganz spezielle Eigenschaften hat. Der Avatar ist also nicht irgendeine Figur, sondern zum Beispiel *Lara Croft* in dem Spiel *Tomb Raider*. Sie hat ganz spezielle Fähigkeiten und ein ausgeprägtes Äußeres. Es wird also eine Person erschaffen.

Gefällt die Person einem Benutzer jedoch nicht, so wird er dieses Spiel womöglich meiden. Dies ist die Gefahr, wenn der Charakter, der modelliert wird, eine sehr eigene Art mitbringt. Wenn der Charakter jedoch gefällt, wird sich gerne an ihn und an das Spiel zurückerinnert. Die Spielfigur verschafft dem Spiel somit einen höheren Wiedererkennungswert [Rou99].

Zu unterscheiden sind die folgenden Möglichkeiten eine Sicht außerhalb der Spielfigur zu ermöglichen:

- **Verfolgerperspektive:**
  Die Kamera befindet sich in einem festen Abstand hinter der Spielfigur und wird synchron zu ihren Bewegungen mitgeführt [Dri07].

- **Feste Kameraeinstellung:**
  Die Kamera befindet sich statisch an einer Stelle und der Benutzer bewegt die Spielfigur, ohne dass sich die Kameraposition verändert. Somit sieht der Anwender seine Figur aus den verschiedensten Winkeln. Die Steuerung für den Benutzer kann hierdurch jedoch sehr erschwert werden [Dri07].

- **Vogelperspektive:**
  Die Kamera befindet sich in einem deutlichen Abstand schräg über

> dem Avatar. Die Kamera bewegt sich dabei synchron zu der Spielfigur oder durch eine separate Bewegungssteuerung. Der Überblick entspricht dabei der bestmöglichen Art und Weise. Dadurch können unter Umständen auch angrenzende Räume oder ähnliches eingesehen werden, indem die Decke dabei durchsichtig oder transparent gemacht wird. Diese Perspektive wird vor allem für die Übersicht und taktische Gegebenheiten angewendet. Zusätzlich können hier auch mehrere Gruppen statt nur einer Person ohne Schwierigkeiten im Blick behalten und durch die Szenerie gelenkt werden. Aus diesem Grund wird diese Perspektive auch vor allem für Strategie und Rollenspiele genutzt [Dri07].

Das Problem der Drittpersonansicht liegt offensichtlich in der Kameraführung. Die Kamera nimmt eine Position hinter der Spielfigur ein. Was passiert jedoch, wenn sich die Kamera dadurch außerhalb der Spielewelt befindet? Zum Beispiel, wenn hinter der Spielfigur eine Wand ist oder sie ein Raum betreten wird? [Rou99]

Verschiedene Lösungsansätze können in den verschiedensten Situationen helfen. Die Kamera kann zum Beispiel bei betreten eines Raumes eine Position direkt hinter dem Avatar einnehmen. Oder die Vogelperspektive wird aktiviert und das Dach des Hauses wird transparent. All dies kann jedoch auch zu einem Orientierungsverlust führen, wenn die Kameraführung und die neue Position für den Benutzer nicht verständlich werden [Rou99].

Werden nun die beiden vorgestellten Benutzerperspektiven verglichen, so wird klar, dass die Drittpersonansicht einen Überblick über die Umgebung erlaubt, die Egoperspektive jedoch Detailblick und mehr Immersion bietet [Rou99].

Je nach Anwendung muss die Priorität festgelegt und demnach entschieden werden, welche Perspektive sich besser eignet. Inzwischen werden sie jedoch auch gerne kombiniert.

Wie Varela sagte:
„don't leave home without it[4], but do not forget to bring along third-person accounts as well." [VS99]

Um beide Perspektiven in Kombination nutzen zu können, bekommt der

[4] er meint hier die Egoperspektive

Benutzer entweder die Möglichkeit die Perspektive nach seinen Wünschen umzuschalten oder dies geschieht ohne spezielle Benutzereinwirkung zum Beispiel durch bestimmte Schlüsselaktionen, wie etwa das Betreten eines Hauses oder in einen Kampf verwickelt zu werden [Rou99].

Neben den betrachteten Eigenschaften der beiden Perspektiven ist es wichtig vor der Entwicklung einer Anwendung zu entscheiden wie immersiv diese gestaltet sein soll. Soll sie möglichst immersiv gestaltet werden, so hilft es nicht unbedingt einfach die Egoperspektive zu benutzen. Die Immersion bezieht sich nicht ausschließlich auf die Blickrichtung und -höhe. Nicht zu vergessen ist die Realität das Immersivste, was sich der Mensch vorstellen kann. Demnach sollte die virtuelle Welt der realen Welt angepasst werden. Anhand einiger Beispiele fällt auf, dass dies jedoch nicht so einfach zu bewerten ist.

- **Reale Welt:** Wenn sich der Mensch in der Realität umschaut, bewegt er seinen Kopf und seine Augen.
  Virtuelle Welt: Schaut sich der Benutzer um, so muss er in der Egoperspektive keine reale Kopf- oder Augenbewegung, sondern den Avatar/die Kamera mit dem Eingabegerät drehen. In der Drittansichtperspektive muss er Kopf- und Augenbewegungen machen, da er den Bildschirm um den Avatar herum anschaut.
  => **Drittpersonansicht** ist realitätsnäher. [STV06]

- **Reale Welt:** Der eigenen Kopf wird nicht gesehen, eventuell Arme, Beine, wenn der Kopf geneigt wird.
  Virtuelle Welt: Egoperspektive: Den Avatarkopf sieht der Benutzer nicht, eventuell Arme und Beine bei Kopfneigung; Drittpersonansicht: der Benutzer sieht immer den ganzen Körper des Avatars.
  => **Egoperspektive** ist realitätsnäher. [STV06]

- **Reale Welt:** Der Mensch kann seine eigenen Körpermaße in Beziehung zur Umwelt sehr gut einschätzen.
  Virtuelle Welt: Egoperspektive: Der Benutzer kann die Größe seines „virtuellen-Ichs“ nicht gut einschätzen; Drittpersonansicht: der Benutzer kann die Größe seines Avatars sehr gut einschätzen.
  => **Drittpersonansicht** ist realitätsnäher. [STV06]

- **Reale Welt:** Eine Übersicht über die Umgebung wird durch alle Sinne beeinflusst. Zum Beispiel durch das Gehör wird der Radius erweitert, der wahrgenommen wird.

Virtuelle Welt: Egoperspektive: Sehr eingeschränktes Sichtfeld. Entspricht grob dem aus der realen Welt, jedoch können die weiteren Sinne nicht ersetzt werden[5]; Drittpersonansicht: Durch die Ansicht von weiter oben wird eine größere Sichtreichweite ermöglicht. Dies ersetzt unter Umständen die anderen Sinne.
=> **Egoperspektive** zu wenig Sicht, **Drittpersonansicht** zu viel Sicht, Realität liegt dazwischen. [STV06]

- **Reale Welt:** Benutzer redet über sich selbst: „ich“, „mein“.
  Virtuelle Welt: Egoperspektive: Benutzer redet über den Avatar: „ich“, „mein“; Drittpersonansicht: Benutzer redet über den Avatar: in der Regel „er“, „sein“. Je nach Aktion des Avatars aber auch „ich“, „mein“.
  => **Egoperspektive** ist realitätsnäher. [VF03]

- **Reale Welt:** Gehirnaktivität bei Aktion: linke Gehirnhälfte aktiv; Gehirnaktivität bei Bewegung: rechte Gehirnhälfte aktiv
  Virtuelle Welt: Egoperspektive: linke Gehirnhälfte aktiv; Drittpersonansicht: rechte Gehirnhälfte aktiv
  => Bei Handlungen ist die **Egoperspektive** realitätsnäher.
  => Bei Bewegungen ist die **Drittpersonansicht** realitätsnäher. [VF03]

- **Reale Welt:** Beanspruchung des Ganzen Gehirns.
  Virtuelle Welt: Egoperspektive: Nur „Point-of-interest“-Betrachtung, nur Teile des Gehirns aktiv; Drittpersonansicht: ganze Umgebung in Bezug zu dem Avatar betrachtet, Beanspruchung des ganzen Gehirns.
  => **Drittpersonansicht** ist realitätsnäher. [Vel91]

Auch nach diesen Betrachtungen ist es schwer zu entscheiden, welche Perspektive diejenige ist, die ein höheres Realitätsempfinden hervorruft. Daran ist festzustellen, dass es nicht reicht die Realität nachzubilden, um die beste Immersion zu erzeugen.

Denn fraglich ist, ob es immersiver ist, zu sehen wie in der realen Welt, also in der Egoperspektive, oder ob es immersiver ist sich selbst einschätzen zu können. Denn obwohl wir in der realen Welt unseren Körper nur bedingt sehen, können wir ihn ganz einschätzen, wie in der virtuellen Welt in der Drittpersonansicht.

[5] mit Hilfe von *Dolby Surround*-Systemen lässt sich der Sound teilweise noch zu Orientierung nutzen

Auch ist es ein Problem zu unterscheiden, ob es immersiver ist, nur den kleinen Sichtbereich aus der Egoperspektive zu sehen oder die ganze Umgebung, wie in der Drittpersonansicht. In der realen Welt wird die Umgebungswahrnehmung durch Geräusche und ähnliches, also die anderen Sinne, ausgeglichen und entspricht eher einer Umgebungseinschätzung in der Drittpersonansicht.

Aus diesem Grund ist es nötig beide Perspektiven zu betrachten und abhängig von dem Anwendungsgebiet und der Zielgruppe in eine Anwendung zu integrieren.

3. Stand der Technik

Um sich in einer Welt von einer Position zur Nächsten zu begeben, benötigt es ein Navigationsverfahren. Grundsätzlich ist dieses je nach Anwendung festgelegt. Unter Umständen hat der Benutzer die Möglichkeit die Art der Navigation zu beeinflussen. Dies ist jedoch meist nur in kleineren Teilen möglich, wie zum Beispiel das Ändern der Benutzerperspektive.

Aktuell gibt es für virtuelle Welten viele verschiedene Navigationsmöglichkeiten und navigationsunterstützende Verfahren. Einige Dieser werden im Folgenden vorgestellt. Da die verschiedenen Anwendungen aufgrund ihres Genres oder ihrer Komplexität unterschiedliche Gegebenheiten mitbringen, muss die Navigation unterschiedlichen Anforderungen gerecht werden, damit der Nutzer auch wirklich in seinen Aufgaben und Intentionen unterstützt wird. Aus diesem Grund sind unterschiedliche Verfahren entstanden, die als Basis dienen, aber zusätzlich für jede Anwendung individuell angepasst werden müssen.

Die Navigationsmöglichkeiten (Abschnitt 3.1) beeinflussen dabei die Steuerungsart. Hierzu wird die Position, Geschwindigkeit und Blickrichtung des Avatars, abhängig von den Signalen, die von dem Benutzer mit Hilfe der Eingabegeräte übermittelt werden, auf verschiedenste Weise berechnet [dSGA+00]. Da die Navigation jedoch auch von dem Anwendungsgebiet und der Benutzerperspektive abhängt, soll ein Einblick in diese für spätere Kapitel eine bessere Beurteilungsgrundlage bieten (Abschnitt 3.1.5).

Im zweiten Teil des Kapitels werden navigationsunterstützende Verfahren betrachtet. Die dort dargestellten Möglichkeiten sollen dem Benutzer die Erfüllung seiner Aufgaben lediglich erleichtern (z. B. Übersichtskarte), ändern aber nicht die gewählte Navigationsmetapher. Sie unterstützen den Anwender in den primären Aufgaben der Anwendung. Dies kann auch die Orientierung beinhalten, damit der Benutzer in der virtuellen Welt nicht verloren geht - beziehungsweise sich besser zurecht findet [DS93]. Falsch eingesetzt können sie jedoch für Verwirrung sorgen, anstatt unterstützend zu wirken. Dabei spielt vor allem die Komplexität der Anwendung eine wichtige Rolle, denn was bei einfachen Anwendungen hilft, muss nicht unbedingt bei Komplexen auch von Vorteil sein und umgekehrt.

3.1. Navigationsmetaphern

Eine Navigationsmetapher beschreibt die Art und Weise, wie sich Positionsbestimmung, Geschwindigkeit und Blickrichtung auf die Kamera und, wenn vorhanden, auch auf die Spielfigur ausüben. Dabei kann der Benutzer durch Eingabemöglichkeiten, wie Maus und Tastatur, die genannten Aspekte beeinflussen.

Ohne jegliche Einschränkungen der Freiheitsgrade hat der Benutzer die Möglichkeit sich absolut überall in der virtuellen Szene zu bewegen (z. B. fliegen, durch Wände laufen). Dies entspricht wahrscheinlich nicht dem Ziel der Anwendung und die Freiheitsgrade müssen von den Entwicklern vorher eingeschränkt werden, denn jede Freiheit, die der Anwender gegeben bekommt, wird er auch nutzen. Wenn eine Steuerungsmöglichkeit jedoch für die zu erfüllende Aufgabe vollkommen unwichtig, sogar verwirrend oder überfordernd für den Benutzer ist, wäre es eine Hilfe, wenn er die Wahl nicht selber treffen könnte. Die Beschränkung der Freiheitsgrade auf das Wesentliche um die Aufgaben zu erfüllen, bringt schneller die gewünschte Gewöhnung an die Steuerung und somit auch den besseren Umgang mit der Anwendung, als die Freiheit sämtliche Optionen verwenden zu können [dSGA+00].

Dabei kann Navigation zwei Ziele verfolgen. Zum einen die Fortbewegung der Spielfigur und somit die des Betrachters oder aber das Betrachten einzelner Objekte in der virtuellen Welt und ein damit verbundenes inspizierendes Navigieren.

Im Folgenden werden einige Ansätze von Navigationsmöglichkeiten vorgestellt. Dabei werden zunächst diejenigen hervorgehoben und genauer beschrieben, die grundlegend sind, wie beispielsweise das Laufen. Diese Möglichkeit kennt der Benutzer auch aus der realen Welt und kann dadurch intuitiver bedient werden, als unnatürliche[1] Metaphern. Eine Erläuterung weiterer interessanter und auch unnatürlicher Methoden findet sich in Abschnitt 3.1.4. In Abschnitt 3.1.5 werden die ausgewählten Navigationsmöglichkeiten in Hinsicht auf ihre Anwendungsgebiete und den Benutzerperspektiven verglichen und bieten somit einen Überblick über die Zusammenhänge dieser Aspekte.

[1] nicht mit der Realität vergleichbar, ungewöhnlich

3.1.1. Laufen-Metapher

Das Laufen beschreibt wohl die natürlichste Navigationsmöglichkeit, da jeder Benutzer sie aus dem realen Leben bereits kennt und sich an keine speziellen Fähigkeiten gewöhnen und anpassen muss. Diese Navigationsmöglichkeit dient vor allem zur Fortbewegung. Durch ein gezieltes Navigieren um ein Objekt, kann er diese Möglichkeit der Navigation zusätzlich nutzen, um Objekte von allen Seiten zu inspizieren.

Unter dem Begriff *Laufen* wird im Folgenden ein sich **Fortbewegen auf einem definierten Boden ohne Höhenveränderung der Figur** verstanden. Das bedeutet, dass der Avatar des Nutzers nicht fliegen, sondern sich lediglich vor, zurück und zu beiden Seiten bewegen kann. Zudem ist ein Drehen des Körpers möglich. Der Anwender kann dabei die Position in den Achsen (bis auf die Höhenachse) bestimmen und die Blickrichtung über das Drehen des Avatarkörpers verändern.

Die Geschwindigkeit kann entweder, je nach Ziel der Anwendung, als konstant angenommen, oder sie kann währenddessen verändert werden. Die Umstellung der ursprünglichen Laufgeschwindigkeit in eine Höhere, in einem so genannten *run-mode*, dient der schnelleren und kurzweiligeren Bewegung innerhalb der virtuellen Welt und entspricht einer *Gaspedal*-Metapher. Dabei kann diese Umstellung entweder automatisch erfolgen, sobald der Spieler zum Beispiel die Vorwärtsbewegung lange genug bestätigt oder aber der Benutzer kann manuell die beschleunigte Geschwindigkeit an- und abschalten, indem er eine spezielle Taste drückt, die die Geschwindigkeit auf einen höheren konstanten Wert festlegt.

3.1.2. Pfad-Metapher

Beschränkt der Entwickler dagegen nicht nur eine mögliche Bewegungsrichtung, wie bei der *Laufen*-Metapher die Vertikale, sondern legt einen Weg fest, so kann der Anwender nur noch Geschwindigkeit und Blickrichtung, aber nicht mehr die Position beeinflussen. Dies führt dazu, dass er von dem erstellten Weg abhängig ist.

Die Blickrichtung des Anwenders wird entweder fest auf einen Punkt gerichtet, durch den Pfad vorgegeben oder anhand der Oberflächenstruktur berechnet [HW97]. Ist die Blickrichtung fest auf einen Punkt fixiert, so ist es für diese nicht relevant, wohin sich der Benutzer bewegt und wie der Pfad verläuft. Der fixierte Punkt bleibt im Mittelpunkt des Blickfeldes des Betrachters (siehe Abb. 3.1).

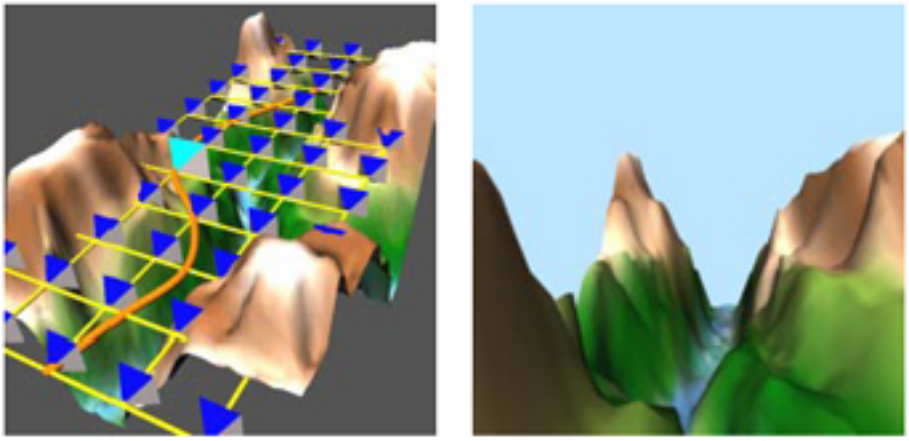


Abbildung 3.1.: *Pfad*-Metapher: Feste Blickrichtung; Abbildung aus: [HW97]

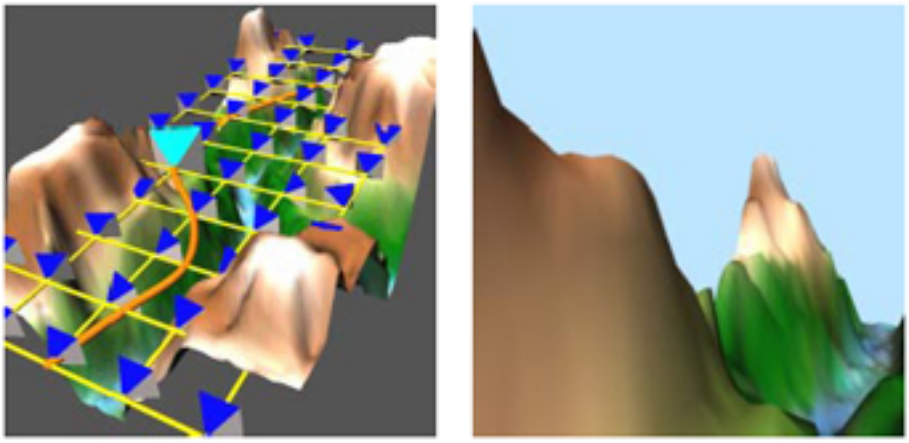


Abbildung 3.2.: *Pfad*-Metapher: Blickrichtung entlang eines Pfades inkl. Rotation um die y-Achse; Abbildung aus: [HW97]

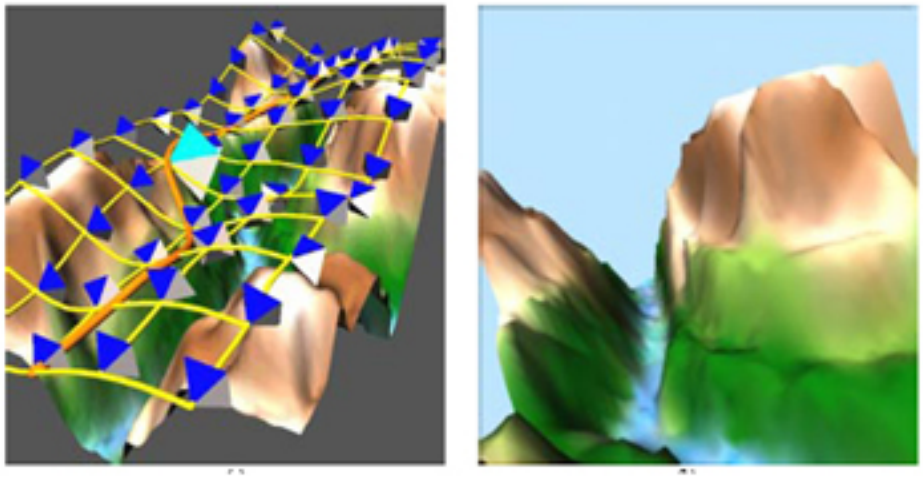


Abbildung 3.3.: *Pfad*-Metapher: Blickrichtung entlang eines Pfades inkl. Rotation um alle Achsen; Abbildung aus: [HW97]

Hängt die Blickrichtung von einem Pfad ab, der sich ausschließlich auf einer Ebene befindet, so schaut der Anwender den Pfad entlang, rotiert dabei aber nicht um die Ebenenachse[2] sondern dreht ausschließlich um die vertikale Achse[3], somit bleibt der Horizont ununterbrochen auf derselben Höhe (siehe Abb. 3.2).

Bei einer Beeinflussung der Blickrichtung durch die Richtung des Pfades und zusätzlich der Oberflächenstruktur, also inklusive Betrachtung von unterschiedlichen Höhen und Tiefen, rotiert der Blickwinkel in der virtuellen Welt. Befindet sich der Benutzer in der Drittpersonansicht rotiert zusätzlich der Avatar um alle Achsen und schaut nicht mehr unbedingt horizontal auf die Welt, sondern der Blickwinkel kann auch wie in Abbildung 3.3 zu sehen ist schräg stehen.

Für verschiedene Aufgabengebiete erscheint diese Art der Navigation sinnvoll, da der Benutzer davon ausgehen kann, dass alle für ihn wichtigen Eckpunkte der virtuellen Umgebung angelaufen werden, ähnlich einer Stadtführung. Führt ein Pfad jedoch nicht an die wichtigen Punkte, also die für die Aufgabenerfüllung relevanten Orte, so kann der Weg eher störend oder sogar hinderlich sein. Wichtig ist natürlich auch, dass der Pfad keine Kollision mit anderen Objekten herbeiführt. Auch zu scharfe

[2] in der Regel die x-und z-Achsen
[3] in der Regel die y-Achse

und plötzliche Drehungen können verwirrend sein und sich somit negativ auf den Benutzer auswirken und sogar bis zum Verlust der Orientierung führen [CRI03]. Dabei ermöglicht die *Pfad*-Metapher verschiedene Umsetzungsmöglichkeiten, um unterschiedlichen Aufgaben gerecht zu werden. Im Folgenden sollen vier Möglichkeiten unterschieden werden:

Vollautomatischer Pfad:
Der erstellte Pfad nimmt dem Benutzer die gesamte Navigation ab, indem dieser ausschließlich angibt, dass er sich bewegen möchte. Dabei kann der Benutzer entweder angeben, dass er den Pfad vorwärts oder rückwärts abgehen möchte, oder er hat die Möglichkeit einen Zielort auszuwählen. Das „wie“ wird dabei komplett von der Anwendung übernommen. Ein vollautomatischer Pfad, der dem Benutzer jegliche Möglichkeit nimmt die Richtung der Navigation zu beeinflussen, kann dem Anwender vorkommen wie ein Film, der abgespielt wird [HBL02]. Dabei hat der Benutzer nur die Option *Pause* oder *Play* zu wählen - was auf dem Weg passiert liegt nicht in seinem Wirkungsbereich.

Kann der Benutzer den Ort wählen, an den er gelangen möchte, so kann dies in einem Gebäude zum Beispiel bedeuten von einem in einen anderen Raum zu gelangen, wie in Abbildung 3.4 dargestellt.

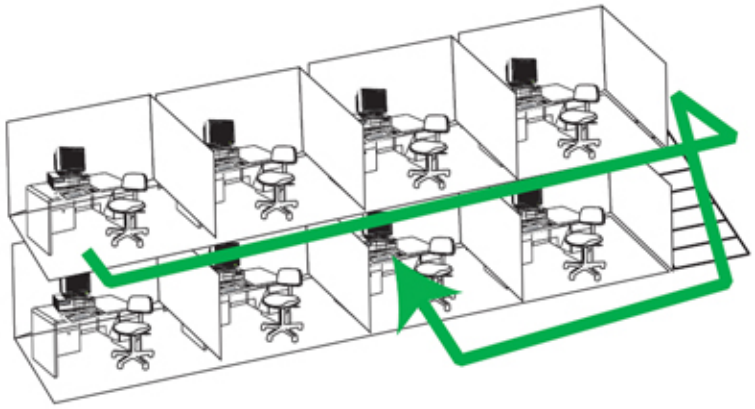


Abbildung 3.4.: *Pfad*-Metapher: Vollautomatischer Pfad; Zu beachten ist hierbei, dass der Pfad nicht einfach die kürzeste Strecke von Start zu Zielpunkt ist, sondern, dass eine Höhenveränderung stattfindet, die der Benutzer nur über die Treppe erreichen kann. Solche Details sind für eine computergenerierte Pfaderstellung schwer zu meistern. Abbildung aus: [dSGA+00]

Die Anforderung bei einem vollautomatischen Pfad bezieht sich jedoch nicht ausschließlich auf den Pfad an sich, sondern auch auf die Bewegungsabläufe des Spieleravatars. Beim automatischen Bewegen der Spielerfigur kann das Programm zusätzlich auch Details der Bewegung übernehmen, wie etwa die Figur klettern oder sich drehen lassen [dSGA+00]. Somit übernimmt die Programmierung nicht nur das Festlegen des Weges für den Benutzer, sondern die gesamte Navigation bis zum Erreichen der Zielposition.

Halbautomatischer Pfad:
Statt eines Pfades, der jede genaue Position beinhaltet, die die virtuelle Figur später abläuft, kann der Pfad auch aus einer Menge von nahe gelegenen Punkten bestehen, die somit eine Variation des Weges der letztendlich abgelaufen wird ermöglicht. Eine solche Ansammlung von Punkten, die in der Umsetzung einem breiten Pfad ähnelt, entspricht der Metapher einer Straße, deren Seiten begrenzt sind. Auf dem Weg hat der Anwender die Freiheit seine Bewegungen selbst zu steuern und kann sich beliebig seitlich bewegen. Den Pfad verlassen kann er allerdings nicht. Somit wird er zu einem speziellen Zielort geführt, wobei sein Weg nur passiv festgelegt ist.

Pfad mit Hilfe einer Führungsperson:
Eine weitere Möglichkeit den symbolisierten Anwender auf einem automatisierten Pfad durch die virtuelle Welt zu führen und ihm kaum bzw. keine Bewegungsfreiheiten zu lassen, ist die Verwendung eines virtuellen Führers. Dadurch wird der Pfad nicht auf den Anwender direkt bezogen, sondern in die Hand eines möglicherweise animierten Drittcharakters gelegt, der den festgelegten Weg entlang läuft. Dieser kann dann bestimmte Orte in der Szene ansteuern und den eigentlichen Akteur „mitnehmen“. Das bedeutet, der Benutzer kann ihm auf diesem Weg folgen. Die *Pfad*-Metapher bietet dem Anwender hier jedoch auch die meisten Freiheiten in der Navigation mit Hilfe eines Pfades, denn er kann den Pfad auf Wunsch verlassen und sich seinen eigenen Weg suchen. Das entspricht der Vorstellung einer Museums- oder Stadtführung. Ein Beispiel wie dies aussehen kann ist in Abbildung 3.5 zu sehen.

Dabei kann die Führungsperson entweder auf den Anwender warten und erst weitergehen, wenn dieser in einem bestimmten Radius um sie steht oder sie geht den Pfad in einer bestimmten Geschwindigkeit ab. Dann kann der Benutzer seine Führungsperson allerdings verlieren. Unter diesen Umständen sollte jedoch eine Möglichkeit gegeben sein diese wiederzufin-

Abbildung 3.5.: *Pfad*-Metapher: Führungsperson; Die Führungsperson befindet sich weiter hinten in dem dargestellten Raum, während der Benutzer sich jedoch weiter vorne umschaut. Abbildung aus: [CRI03]

den oder am eigenen Standpunkt erscheinen zu lassen, um sich von dort aus weiterzubegeben.

Hat der Führer der Route außerdem die Möglichkeit zu sprechen, kann er Zusatzinformationen auf eine intuitive Art und Weise vermitteln, wie der Benutzer sie auch aus dem realen Leben kennt. Die Metapher eines Führers, um sich in unbekanntem Gelände zurechtzufinden, ist dem Benutzer ebenfalls von Stadt- oder Museumsführungen bekannt. Die Navigationsart muss dazu nicht erst erlernt werden, da sich der Avatar ständig ausschließlich in der *Laufen*-Metapher bewegt. Aufgrund des Bezugs zu der Realität liegt es nahe, den Benutzer auch wie in der Realität „nur“ gehen zu lassen. Werden Spieler und Führer in der virtuelle Szene jedoch beispielsweise durch einen Vogel oder einen Geist repräsentiert, können auch spezielle Fähigkeiten wie fliegen oder durch Wände gehen durchaus im Kontext wirken. Ein weiterer Vorteil durch einen animierten „Freund“ ergibt sich daraus, dass die virtuelle Welt zudem lebendiger und attraktiver für den Spieler wird. Da der animierte Charakter auf seine Umgebung reagieren muss, ist es meistens notwendig den Pfad an dieser Stelle von Hand zu erstellen. Somit ist er nicht einfach in andere Welten übertragbar [CRI03]. Durch diese Metapher und die Wirkung des Führers auf den Benutzer wird eine ganz andere Art der Beeinflussung der Navigation des Spielers ermöglicht. Ihm werden dabei keine Freiheiten genommen, sondern Hilfestellungen geboten, die zudem seine Neugier wecken können.

Methoden zur automatischen Pfaderzeugung werden unter anderem von Chittaro et. al in [CRI03] vorgestellt. Darin wird der Boden in ein Gitter unterteilt, welches zur Berechnung der Position verwendet wird. Dies ist für den Anwender jedoch nicht notgedrungen sichtbar. Den einzelnen Sektoren wird jeweils ein Wert zugeteilt, je nachdem ob der Führer auf ihm stehen darf oder nicht. Während der Anwendung werden diese Werte abgefragt. Der Führer bewegt sich entsprechend den Werten der einzelnen Sektoren nur an die Orte, die er betreten darf. Problematisch wird die Berechnung unter anderem dann, wenn sich die Bewegung nicht nur auf einer planaren Ebene befindet, sondern Berge oder Stockwerke (in Gebäuden), also Höhenunterschiede vorhanden sind.

Eine Führung muss aber nicht unbedingt durch einen virtuellen Charakter dargestellt werden. Wird beispielsweise eine Linie zwischen zwei *points-of-interests*[4] auf dem Boden dargestellt, kann der Weg, dem der Benutzer folgen soll, hierdurch symbolisiert werden. Nachdem die Linie bis zum nächstgelegenen interessanten Punkt verfolgt und der Zielpunkt erreicht wurde, erscheint eine neue Linie, die nun von diesem Punkt zu dem nächsten *point-of-interest* führt [HBL02]. Auch hier entsteht eine Führung für den Anwender, ohne dass ihm die Freiheit in der Navigation entzogen wird.

Inspektionspfad (Orbit-Metapher):
Ein Pfad kann jedoch nicht nur durch eine bestimmte Szene führen, er kann auch zur Inspektion dienen. Hierbei besteht das Ziel des Pfades nicht primär aus der Fortbewegung und der Navigation in der Welt, sondern dem gezielten Betrachten eines einzelnen Objektes. Ohne um das Objekt herum laufen zu müssen bewegt sich der Avatar, beziehungsweise der Blickwinkel des Betrachters, auf einem Pfad um das Objekt. Der Pfad führt den Benutzer demnach in einem bestimmten Abstand um ein Objekt herum, wobei der Blickpunkt immer auf das Objekt gerichtet ist [TRC01]. Diese Metapher wird im Folgenden auch als *Orbiting*[5] bezeichnet. Hierbei fällt auf, dass der Benutzer weder die Möglichkeit hat die konkrete Position zu bestimmen, noch die Blickrichtung zu ändern. Er kann dabei die Position auf dem Pfad oder/und die Geschwindigkeit der Bewegung über den Pfad verändern, mit der sich die Kamera um das Objekt bewegt. Durch eine Art Zoomfunktion kann zusätzlich der Abstand zu dem Objekt zu verändert werden, statt diesen vorher festzulegen.

[4] interessante Orte
[5] aus dem Kontext: Umlaufbahn eines Objektes um einen Himmelskörper

Bei dieser Metapher muss der Weg um das Objekt nicht notwendigerweise auf dem Boden entlang führen. Bei kleineren Objekten bringt eine Führung auf dem Boden/ auf der Ebene sicherlich mehr, im Gegensatz zu größeren Objekten bei denen es sich anbietet die Rotation in einer gewissen Höhe/ einem gewissen Abstand zur Ebene durchzuführen, damit das Objekt in seiner Gänze erfasst werden kann (siehe Abb. 3.6).

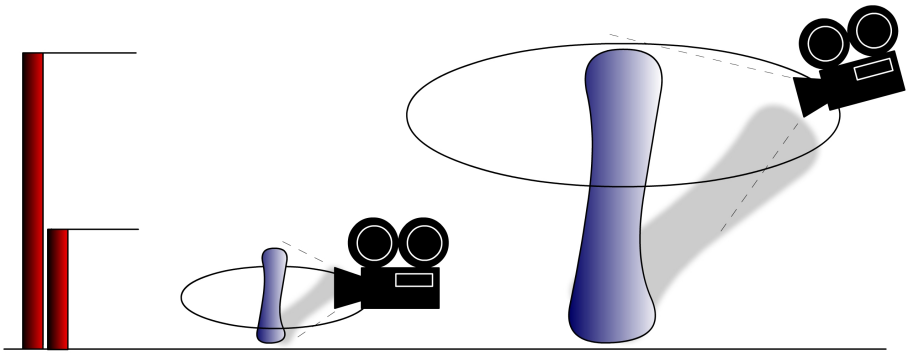


Abbildung 3.6.: *Inspektionspfad*: *Orbit*-Metapher; Abgebildet ist ein kleines und ein großes Objekt und ihre zugehörigen Inspektionspfade, auf denen sich die Kamera entlang bewegt. Dabei ist der Pfad bei einem größeren Objekt wesentlich höher über dem Boden angesetzt ist. Zudem ist die Neigung der Kamera anders als bei kleinen Objekten.

***Point-of-interest*-Orientierung:**
Vorstellbar nach Hughes [HBL02] ist es auch, den Anwender die Position bestimmen zu lassen und die Orientierung automatisch auf den nächstliegenden *point-of-interest* zu setzen. Seiner Überlegung nach werden den verschiedenen Orten bei der Entwicklung der virtuellen Szene Prioritäten zugewiesen. Somit kann je nach Standort die Blickrichtung berechnet werden. Auf diese Weise entsteht ein Pfad durch die Lenkung der Orientierung des Benutzers. Diese Art der Navigation scheint jedoch nicht intuitiv zu sein, weshalb sie im Folgenden nicht weiter aufgegriffen, dennoch aber genannt werden soll.

Für all diese Pfadnavigationsmöglichkeiten können die Pfade automatisch erstellt, von den Entwicklern vorher per Hand eingefügt oder vom Benutzer in Echtzeit erzeugt werden. Eine automatische Erstellung zur Laufzeit

bringt den Vorteil, dass er für jeden Benutzer speziell angepasst werden kann [CRI03]. Wenn vor Erstellung des Pfades Prioritäten des Benutzers festgestellt werden, kann der Pfad entsprechend seinen Interessen erstellt werden. Die Führung kann dann individuell auf den einzelnen Benutzer ausgerichtet werden und bringt zum Beispiel in einem Museum den Vorteil, dass der Betrachter bestimmte Orte die ihn nicht interessieren überspringen kann. Benötigt der Benutzer allerdings zum Erfüllen spezieller Aufgaben in der Anwendung die Kenntnis über alle Bereiche der virtuellen Szene, oder soll er sich einen Überblick über die gesamte Welt machen können, so ist ein statischer Pfad durchaus sinnvoller, da davon ausgegangen werden kann, dass der Benutzer jeden relevanten Ort besuchen muss. Im Umkehrschluss kann der Benutzer bei einem statischen Pfad davon ausgehen, dass er alles benötigte Wissen über die Szene erhalten hat. Der Nachteil eines in Echtzeit erstellten Pfades ist die erforderliche höhere Rechenleistung des eingesetzten Systems, dass unter Umständen zu Verzögerungen oder Unterbrechungen im Anwendungsfluss führen kann.

Igarashi stellt in [IKMT98] eine Methode vor, in der der Benutzer den Pfad selber zeichnet und die virtuelle Figur dann entsprechend zu diesem bewegt wird. Befindet sich der Spieler einmal auf dem von ihm definierten Weg, so ist die Bewegung losgelöst von der Orientierung und der Benutzer kann sich entlang seines Pfades auf die Umgebung konzentrieren. Igarashi demonstriert dies wie in Abbildung 3.7 zu sehen ist.

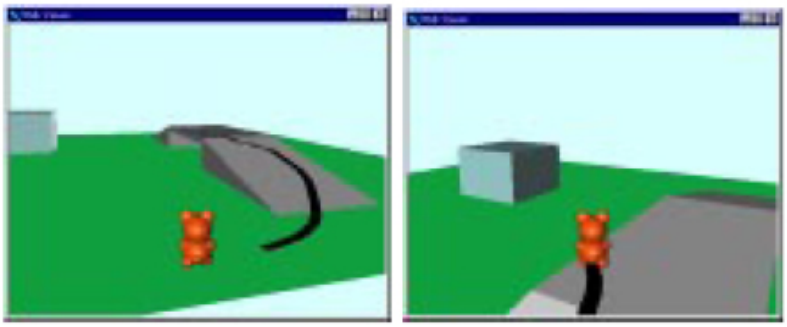


Abbildung 3.7.: Igarashi: dynamische Pfaderstellung; Zu sehen ist der Avatar des Anwenders, eine Umgebung und der in schwarz eingezeichnete Pfad. Im ersten Bild ist die Zeichnung des Pfades gerade beendet, in der zweiten Abbildung ist der Avatar den Pfad bereits abgelaufen und befindet sich am Endpunkt des Pfades, dem Zielort. Abbildung aus: [IKMT98]

Somit bietet die Navigation mit Hilfe von Pfaden eine gezielte, passive Steuerung eines Avatars, wodurch der Benutzer seine Aufmerksamkeit besonders gut auf die Umgebung richten kann.

3.1.3. Teleport-Metapher

Die *Teleport*-Metapher vereint Aspekte der beiden zuvor vorgestellten Methoden. Sie entspricht dem Laufen in der Hinsicht, dass sich der Benutzer frei in der Welt bewegen kann. Dabei bewegt er sich „springend“ durch die Welt, indem er durch Klicken in die virtuelle Szene seinen Zielort wählt. An diesen wechselt der Avatar, ohne dass der Benutzer den Weg zwischen den beiden Positionen kalkulieren oder kennen musste. Dies entspricht einem Aspekt aus der *Pfad-Metpaher*. Der Benutzer bestimmt zwar wohin er gelangen möchte, ihm wird aber die Problematik der Navigation des Weges zu dem Zielort abgenommen. Das Ablaufen des Weges ist hier im Gegensatz zu dem Pfad jedoch nicht sichtbar, da sich der Avatar nach dem wählen des Zielortes dorthin teleportiert. Was Teleportation bedeutet bringt Wikipedia auf den Punkt:

> *„Teleportation is the movement of objects or elementary particles from one place to another, more or less instantaneously, without traveling through space.“ [Tel07]*

Diese Metapher eignet sich besonders zur Fortbewegung und nicht zum Inspizieren von Objekten, da es ein Springen um ein Objekt mit einer sich ständig ändernden Blickrichtung bedeuten würde. Die Bedienung kann dabei unterschiedlich implementiert werden. Zur möglichen Auswahl des Zielortes gibt es zwei Varianten. Entweder der Benutzer kann an jeden Ort gelangen, an den er möchte, oder die Orte an die teleportiert werden kann, wurden bereits während der Entwicklung der Anwendung eingeschränkt.

Ohne Einschränkung bedeutet es, dass der Benutzer durch Klicken in die Welt oder durch weitere Möglichkeiten, wie das Auswählen über eine Übersichtskarte oder Icons, an jedem Ort ankommt, den er ausgewählt hat. Eine solche Möglichkeit bietet dem Benutzer viele Freiheiten, da er schnell an ein von ihm gewähltes Ziel gelangen kann.

Jedoch besteht ein Problem in der Eventualität der Desorientierung, da der Anwender an einer Stelle in der Welt verschwindet und an einer Anderen wieder auftaucht. Eine Orientierungslosigkeit wird wahrscheinlicher, je unbekannter die Welt ist. Während dem Teleportieren kennt er weder seine globale Position, noch die Relative zu den anderen Objekten,

welches zu unerwünschten Kollisionen an der Zielposition führen kann. Dies geschieht vor allem dann, wenn nicht nur statische, sondern auch bewegte Objekte in der virtuellen Welt vorhanden sind [dSGA+00]. Zudem wird die Benutzung dadurch erschwert, dass es für diese Navigation keine Analogie zu einem realen Fortbewegungsmittel gibt [CRI03].

Um die Orte einzuschränken, können verschiedene Mittel je nach Anwendung betrachtet werden. Die einfachste Möglichkeit besteht darin, das Teleportieren nur an speziellen Orten zuzulassen. Dazu müssen diese bei der Erstellung der Anwendung bereits festgelegt werden. In der virtuellen Welt sollte dann eine entsprechende Aktivierung mittels eines Gegenstandes oder eines kenntlich gemachten Bereiches[6] ermöglicht werden. Vergleichbar ist diese Sichtweise auch mit der Metapher des *Hyperlinks* im *WorldWideWeb* [dSGA+00], hier sind Start und Zielposition aneinander gebunden.

Es kann vorteilhaft sein dem Benutzer die Freiheit zu geben seine Start- und Zielorte der Teleportationen selbst zu bestimmen, statt den Entwickler vorher Orte festlegen zu lassen. Dies kann umgesetzt werden, indem der Anwender an verschiedenen Orten an denen er sich befindet, eine Markierung setzt und somit später wieder zu diesen zurückkehren kann.

3.1.4. weitere Navigationsmetaphern

Außer den erwähnten Navigationsmetaphern, gibt es natürlich zusätzliche Möglichkeiten, die sich vor allem auch anbieten, um weite Strecken zurückzulegen. Robinett beschreibt dazu in [TRC01], dass sich Skalierung sehr gut verwenden lässt, um dem Benutzer einen schnellen „Flug“ zu verschiedenen Orten zu erlauben. Somit bietet sich dies für die Benutzung in großen Welten und der Notwendigkeit weite Strecken zurücklegen zu müssen an. Die im Folgenden erläuterten Metaphern nutzen die Skalierung, um große Strecken zurückzulegen. Dabei ist es gleichermaßen möglich die Welt kleiner, als auch den Avatar größer zu skalieren, was letztendlich dasselbe Ergebnis liefert.

Ephemeral World Compression:
Diese Methode bedeutet übersetzt „kurzweilige Weltstauchung“. Mit Hilfe der Metapher lassen sich auch in großen virtuellen Welten weite Strecken schnell und leicht zurücklegen. Die Technik bietet dem Benutzer nicht nur die Option sich durch einfaches Laufen in der virtuellen Welt zu bewegen,

[6] wie beispielsweise die „Beammaschine“ aus der TV Serie *Enterprise*

was bei langen Strecken langwierig und daher auch schnell langweilig werden kann, sondern eine Kombination zwischen der *Laufen*-Metapher und einer Skalierung. Dazu wird die Szenerie um ihn herum kleiner skaliert, wobei die Spielfigur selbst ihre ursprüngliche Größe beibehält. Sie steht somit wie ein Riese in der Szene, wodurch der Benutzer einen sehr weitreichenden Überblick erlangt. Details kann er so allerdings nicht wahrnehmen.

Nachdem die Welt kleiner skaliert wurde, kann er sich in ihr bewegen, wie er es mit Hilfe der *Laufen*-Metapher auch machen würde. Der Unterschied besteht nun darin, dass er mit einem Schritt eine wesentlich größere Wegstrecke zurücklegt. Aus diesem Grund ist die Metapher besser für große Welten geeignet. Durch die großen Schritte entsteht jedoch zusätzlich eine Ungenauigkeit in der Navigation.

Ist der Benutzer an dem Punkt angelangt an den er sich bewegen wollte, so kann er die Welt in die ursprüngliche Größe zurückskalieren. Nun ist es ihm möglich wieder Details zu sehen und sich auch mit kleinen Schritten weiter zu bewegen. Damit kann er genau an den Punkt gelangen, den er als Zielpunkt gewählt hat, die er jedoch über die großen Schritte in der klein skalierten Welt nicht erreichen konnte. Zusammenfassend liegt der große Vorteil dieser Methode genau darin, mit wenigen „Schritten“ große Distanzen zurücklegen zu können [TRC01].

Speed coupled flying:
In dieser Metapher wird die Annahme verfolgt, dass ein Spieler, wenn er sich fortbewegt und seine Geschwindigkeit erhöht eine größere Strecke zurücklegen möchte. Weiterhin besagt sie, dass es dabei hilfreich wäre, wenn er währenddessen einen Gesamtüberblick über seine direkte Umgebung hat. Verlangsamt er seine Bewegung wieder, so möchte er nahe an Objekte heran und bevorzugt wahrscheinlich einen detaillierteren Blick in die Welt. Somit soll die Metapher dem Nutzer die Möglichkeit bieten zwischen lokalem Blick auf die Welt und einer Übersicht von oberhalb zu wechseln [TRC01].

Um dies zu automatisieren, wird die Geschwindigkeit der Spielfigur mit der Kamerahöhe und ihrer Neigung verknüpft. Erhöht der Spieler die Geschwindigkeit, so erhöht sich auch der Abstand zwischen Kamera und Boden. Umso schneller der Avatar läuft, desto mehr sieht der Benutzer von der Welt um den Avatar herum. Wird der Avatar wieder langsamer, senkt sich die Kamera zu dem Avatar hinunter bis zu dem Punkt der ursprünglichen Kameraposition. Somit hat er die Möglichkeit durch die

Abbildung 3.8.: *Speed coupled flying*-Metapher: normale Perspektive, fligende Perspektive; Abbildung aus: [TRC01]

Geschwindigkeit die Perspektive zu steuern und entsprechend sowohl in lokaler, als auch in globaler Sicht zu navigieren.

In der entwickelten Anwendung von Tan et al. [TRC01], wurde diese Möglichkeit wie in Abbildung 3.8 integriert. Veranschaulicht ist sowohl die normale ursprüngliche Perspektive, sowie die abgehobene Perspektive, die zu sehen ist, wenn der Benutzer die Geschwindigkeit erhöht hat und die Kamera sich nicht mehr hinter der Spielerfigur befindet.

Ein großes Problem dieser Metapher bezieht sich auf die Notwendigkeit aus der Höhe zurück auf den Boden zu gelangen. Wenn der Benutzer langsamer wird, senkt sich die Kamera wieder Richtung Boden. Dies muss langsam geschehen, da der Benutzer die Orientierung verlieren kann. Dabei landet die Kamera nach Tan et al. nicht einfach indem sie sich Richtung Boden absenkt, sondern sich dabei leicht vorwärts bewegt. Dies entspricht der Metapher eines Landeanfluges. Somit ergibt sich die Endposition aus dem Punkt der Kamera in der Luft abzüglich der Höhe und einer gewissen Vorwärtsbewegung, die es dem Benutzer erleichtert diesen Zielpunkt als den solchen einzuschätzen, weil er diesen Punkt während der Landung betrachtet und ihm langsam näher kommt. Bei dem Versuch den Landepunkt unterhalb der Kamera festzulegen, indem nur die Höhe verringert wurde, konnten die Testpersonen ihren Landepunkt nicht gut einschätzen und bevorzugten demnach die Variante mit dem Landen durch leichte Vorwärtsbewegung [TRC01].

Inverse Fog and Inverse Scaling:
Soll sich der Benutzer in einer Welt orientieren und zurechtfinden, so ist es wichtig, dass er nicht ausschließlich die Objekte, die nahe dem Avatar sind betrachtet, sondern auch entfernte Objekte wahrnehmen kann.

Diese beiden Metaphern beschreiben die Vorstellung es existiere eine Sphäre in einem bestimmten Radius um die virtuelle Spielfigur, die dabei ihr Zentrum bildet. Der Benutzer kann den Radius der Sphäre kontrollieren und seine Umgebung als innerhalb oder außerhalb definieren. Alles was sich in ihr befindet wird transparent dargestellt (in *Inverse Fog*) oder kleiner skaliert (in *Inverse Scaling*). Somit ergibt sich eine schnellere Übersicht über die Welt [TRC01].

Es ist dem Nutzer hierdurch möglich eine gewünschte Grenze der Tiefe zu wählen, die er anschauen möchte (siehe Abb. 3.9). So kann er ganz speziell die Objekte betrachten die sich in einer bestimmten Entfernung zu ihm befinden. Eine Fortbewegung wird durch diese Metapher zwar nicht beschleunigt, aber übersichtlicher. Zusätzlich wird die Auswahl des Zielpunktes erleichtert.

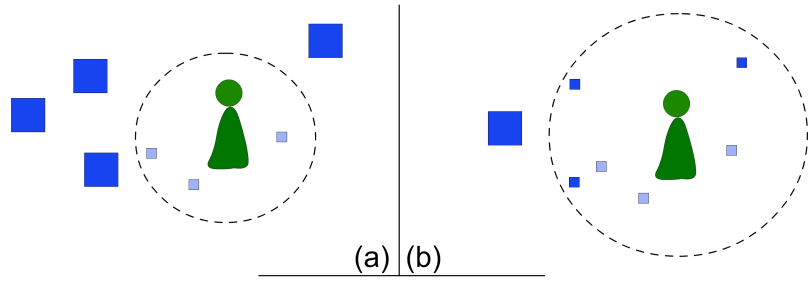


Abbildung 3.9.: Darstellung der Funktionsweise anhand der *Inverse Scaling*-Metapher: (a) kleiner Radius, (b) großer Radius; Entsprechend des Radius der Sphäre um die virtuelle Figur sind mehr oder weniger Objekte von der Skalierung betroffen.

Posession-Metapher:
In allen betrachteten Metaphern wurde (in der Drittpersonansicht) davon ausgegangen, dass die Blickrichtung der Kamera auf den Spieleravatar gerichtet ist. Denkbar wäre jedoch auch die Möglichkeit die Welt aus der Sicht eines anderen Objektes wahrnehmen zu können. Nach Auswahl eines Objektes wird die Kamera in dessen Position verschoben und der Anwender sieht die Welt aus der Sicht, die das Objekt innehat. Dies können zum Beispiel andere Spielfiguren, ein Fernrohr oder ähnliches sein.

Nach [TRC01] wird diese Metapher als *Posession*-Metapher bezeichnet, weil das gewählte Objekt von dem Spieler eingenommen wird. Betrachtet der Benutzer die Welt so aus verschiedenen Blickwinkeln, indem er verschiedenste Objekte aus der virtuellen Welt eingenommen hat, so kann er einen Überblick über die Welt erlangen und sich fortbewegen ohne,

wie bisher davon ausgegangen, seinen Avatar bewegen zu müssen. Zur eigentlichen Navigation in der Welt ist diese Methode darum jedoch nur beschränkt sinnvoll, da ein Spieler in den gängigen Anwendungen eine eigene Spielfigur besitzt, um sich mit dieser zu identifizieren. Ein Hinausspringen aus dem Körper seiner Figur reißt den Benutzer somit immer wieder aus dem Spiel heraus. Eine Verwendung lässt sich jedoch sicher als Zusatzfunktion oder für bestimmte Anwendungsfälle finden.

Object Manipulation and Ghost Copy:
Statt der Aufgabe der Fortbewegung zu dienen, gibt es auch für die Inspektion weitere Möglichkeiten der Umsetzung. Die in [TRC01] beschriebene Technik verbindet dazu Navigation mit Interaktion, indem sich der Benutzer durch das indirekte Manipulieren des Objektes um das Objekt selbst bewegt. Abbildung 3.10 zeigt den Prozess, wie der Zielpunkt durch Interaktion mit dem Objekt gewählt wird und letztendlich die Navigation folgt.

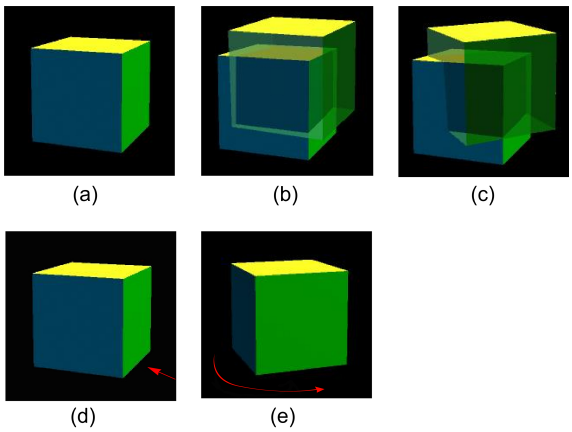


Abbildung 3.10.: *Object Manipulation and Ghost Copy*-Metapher: Durch das Auswählen eines Objektes, um das rotiert werden soll (a), wird eine semitransparente Kopie erzeugt, die über dem Objekt dargestellt wird (b). Durch das „Anfassen“ der Kopie kann der Benutzer diese nun aktiv rotieren. Er dreht sie durch Bewegung mit der Maus in die Position, aus der er das Original-Objekt betrachten möchte (c). Durch das „Loslassen“ der Kopie des Objektes verschwindet diese und die neue Blickrichtung für die Kamera wurde festgelegt (d). Die Kamera wird nun in die gewählte Position überführt (e), so dass er letztendlich nicht das Objekt sondern die Kamera, also sich selber gedreht hat.

Diese Art Navigation und Interaktion zu verbinden birgt die Gefahr, dass der Benutzer erst lernen muss mit dem Konzept umzugehen, da er zwar mit dem Objekt (genauer mit der Kopie des Objektes) interagiert, jedoch letztlich eigentlich nicht das Objekt bewegt, wie vielleicht erwartet, sondern die Kamera. Wenn der Anwender die Maus also zum Beispiel nach rechts bewegt, so wird die Rotation der Kamera anschließend nach links um das Objekt ausgeführt (vergleiche Rotationsrichtungen in (c) und (e)).

3.1.5. Anwendungsgebiete

Werden verschiedene Anwendungen betrachtet, so fällt auf, dass die verschiedenen Navigationsverfahren nicht immer ausschließlich einzeln vorhanden sind. Häufig wird nicht nur ein Überblick oder eine Objektinspizierung benötigt, sondern beides. In virtuellen Anwendungen ist meistens nicht der Weg das Ziel. Also nicht die Navigation an sich, sondern die Interaktion mit der Umgebung. Dazu muss der Benutzer allerdings navigieren, weshalb die Navigation ein wichtiger Teil einer jeden Anwendung in einem dreidimensionalen Raum ist. Die Verknüpfung der verschiedenen Navigationsmöglichkeiten ist demnach eine Notwendigkeit, um den Ansprüchen der Anwendungen gerecht zu werden.

Ein Beispiel für eine solche Verknüpfung findet sich in [TRC01] mit der erwähnten Technik *Speed coupled flying with orbiting*. Hier werden die Vorteile der beiden Möglichkeiten verknüpft und bringen durch den Überblick des *Speed couples flying* eine schnellere Navigation in einer großen Welt, sowie Orientierung. Die Option Objekte zu inspizieren ist zusätzlich durch die *Orbiting*-Metapher integriert.

In Hinblick auf die vielen verschiedenen Genres, die Navigation in dreidimensionalen Szenen benötigen, ist eine allgemeine Aussage über die beste Möglichkeit eine Navigation zu verwirklichen nicht möglich. Je nach Aufgabe und Ziel der Anwendung werden unterschiedliche Anforderungen gestellt. Ein Blick auf gängige Anwendungen und somit eine Verknüpfung der Genres mit der entsprechenden Auswahl der Benutzerperspektive gibt einen Einblick in die aktuellen Einsatzgebiete der einzelnen Navigationsmetaphern.

Es gibt Genres für die sich aufgrund ihres Aufgabenfeldes einige Navigationsarten nicht eignen. Dieselbe Situation gilt auch für die Benutzerperspektiven. In vielen Genres werden bestimmte Kombinationen heutzutage immer wieder umgesetzt. Somit zeigt sich, dass sich diese Zuordnungen

über die Jahre als intuitiv für den Benutzer erwiesen haben müssen. In Abbildung 3.11 ist eine Gegenüberstellung zu sehen, die die Genres mit den meist genutzten Perspektiven den bevorzugten Navigationsmetaphern zuordnet. Dabei wurde ausschließlich eine Auswahl der Navigationsmetaphern betrachtet und stellt die häufigsten Vorkommnisse dar. Die Tabelle erhebt dabei keinen Anspruch auf Vollständigkeit.

Navigations-metaphern / Genre	Fliegen ohne Einschränkung	Laufen	Teleport	Pfad: Voll-automatisch	Pfad: Halb-automatisch	Pfad: Führung	Pfad: Inspektion
Shoot'em-up		Egopersp. + Drittpersonans.					
Beat'em up		Drittpersonans.			Drittpersonans.		
Jump'n Run					Drittpersonans.		
Rennspiele					Egopersp. + Drittpersonans.		
Rollenspiele		Egopersp. + Drittpersonans.	Drittpersonans.				Drittpersonans.
Strategiespiele		Drittpersonans.	Drittpersonans.				
Adventures		Drittpersonans.			Drittpersonans.		
virtueller Rundgang	Egopersp.			Egopersp.	Egopersp.	Egopersp.	
virtuelle Montage	Egopersp.		Egopersp.	Egopersp.		Egopersp.	Egopersp.

Abbildung 3.11.: Verknüpfungen: Genres, Navigationsmetaphern, Benutzerperspektiven

Werden nun die Genres und ihre Aufgabengebiete genauer betrachtet, so fällt auf, dass diese sehr unterschiedlich sind und durch verschiedene Navigationsmöglichkeiten besser unterstützt werden können als durch andere. Definitionen dieser Genres werden vielzählig[7] angeboten, beschreiben aber alle denselben Zusammenhang und sollen im Folgenden kurz zusammenfassend umschrieben werden, um die Abbildung 3.11 und damit die Zusammenhänge verständlicher zu machen.

[7] siehe diverse Lexika, Computerspieleseiten

Shoot'em-up:
Das Ziel dieses Genres besteht aus dem Finden und Eliminieren seiner Gegner. Dazu stellt der Benutzer einen Charakter dar, der durch die virtuelle Welt läuft und sich orientieren und unbekannte Objekte (in dem Fall die Gegner) finden muss. In den meisten Fällen ist der Benutzer dabei ein Mensch oder ein sich auf dem Boden bewegendes Individuum, weshalb die *Laufen*-Metapher vorzugsweise genutzt wird. In einigen Spielen, in denen jedoch Flugzeuge als Avatare dienen, ist der Benutzer-Avatar in seiner Navigation entsprechend nicht auf den Boden beschränkt, sondern er kann fliegen. Jedoch existieren immer auch Einschränkungen wie weit und wohin er fliegen kann, was sich aber aus dem Spielekontext ergibt. *Shoot'em-up*s können sowohl als *Egoshooter* wie zum Beispiel *Halflife* oder *Unreal*, als auch als *Third-Person-Shooter* wie *Tomb Raider* oder *Grand Theft Auto* umgesetzt werden (siehe Abb. 3.12).

(a) (b)

Abbildung 3.12.: Genre: Shoot'em-up: (a) Egoshooter: *Halflife*[8], (b) Third-Person-Shooter: *Tomb Raider*[9]

Beat'em-up:
Dieses Genre bezeichnet sogenannte „Prügelspiele", in denen es darum geht Gegner durch besondere Tritte, Schläge und ähnliches k.o. zu schlagen. Der Benutzer braucht in dieser Welt keine Orientierung und muss keine Gegenstände finden, sein Avatar muss ausschließlich besondere Bewegungen ausführen. Um dieses Genre für Benutzer attraktiver zu gestalten, soll er sich diese anschauen können. Aus diesem Grund befindet sich diese

[8] Abbildung aus: http://www.panix.com/ğiblin/cv/halflife2.jpg, Stand: Nov. 2007
[9] Abbildung aus: http://img2.kult-mag.com/photos/00/00/74/92/ME0000749209_2.jpg, Stand: Nov. 2007

Art von Spiel immer in der Drittpersonansicht, weil es auf die besonderen Bewegungsausführungen ankommt, die die beiden Kämpfer in einer normalerweise recht kleinen Arena ausführen. Die Eingabemöglichkeiten sind dabei beschränkt auf ein paar Tastenkombinationen für die einzelnen Kampfbewegungen und ein Vor- und Zurückgehen. Dabei läuft die Spielfigur in verschiedenen Bewegungskombinationen wie springen, treten etc. den vorgegebenen Pfad vor und zurück (siehe Abb. 3.13 (a)). In neueren *Beat'em-up*-Spielen gibt es jedoch auch mehr Bewegungsfreiheiten (siehe Abb. 3.13 (b)). Ein Beispiel dieses Genres ist die *Mortal Combat* Reihe, wo in neueren Titeln der Reihe mehr Freiheiten angeboten werden und auch eine Rundumansicht möglich ist.

(a) (b)

Abbildung 3.13.: Genre: Beat'em-up: (a) *Mortal Kombat 2*[10], (b) *Mortal Combat 5*[11]

Jump'n Run:
Als *Jump'n Run* werden die Spiele bezeichnet, in denen es vor allem darauf ankommt den Avatar rennen und springen zu lassen. Daraus ergibt sich auch der Name des Genres. Die Welt besteht meist aus verschiedenen Plattformen, die über Leitern und ähnliches erreicht werden können. Zusätzlich müssen Fallen und anspruchsvolle Passagen überwunden werden. Von dem Benutzer erwartet dieses Genre Geschicklichkeit und Reaktionsvermögen. Eine Orientierung in der gesamten Welt ist meistens nicht nötig, da der Weg vorgegeben ist, sowie auch nicht immer möglich, da meist nur ein kleiner Teil der Szene zu sehen, und ein Zurückkehren in bereits durchlaufene Gebiete nur selten möglich ist. Als Perspektive ist immer die Drittpersonansicht gewählt, da der Benutzer genau einschätzen können

[10] Abbildung aus: http://gotps3.ru/files/images/mortal_kombat_ii_574523050.jpg, Stand: Nov. 2007
[11] Abbildung aus: http://img223.imageshack.us/img223/2764/51mm0.jpg, Stand: Nov. 2007

muss wie weit sein Avatar läuft, springt oder durch etwas durchpasst, um die verschiedenen Plattformen zu erreichen. Bekannte Beispiele hierfür sind *Super Mario*, *Rayman* und *Sonic* (siehe auch Abb. 3.14).

(a) (b)

Abbildung 3.14.: Genre: Jump'n Run: (a) *Super Mario*[12], (b) *Rayman 1*[13]

Rennspiele:
Dieses Genre beschreibt vor allem Motorsport-Rennspiele aber auch Skirennen und ähnliches. Diese Spiele erwarten von dem Benutzer Geschicklichkeit und Reaktionsvermögen. Es existiert ein vorgegebener Weg, der abgefahren wird, auf dem der Benutzer aber sonst seine Freiheit in seiner Navigation behält. Demnach ist eine Orientierung über die gesamte Welt nicht notwendig. Der Benutzer hat meist die Wahl der Benutzerperspektive (siehe Abb. 3.15 (a), (b)) und kann sie mittels Tastenkombination wechseln. Beispiele für dieses Genre sind *Need for speed* und *Ski-Challenge*.

Rollenspiele:
In Rollenspielen übernimmt der Spieler eine Rolle in dem Geschehen der virtuellen Welt. Dieses Geschehen ist verstrickt zu einer Geschichte, die während des Spiels weiterentwickelt wird. Darin spielt der Benutzer einen Charakter, mit dem er sich identifizieren kann, wenn er möchte. Dessen Fähigkeiten und Charaktereigenschaften kann er beeinflussen und im Laufe der Geschichte ausbauen. Das Ziel dieses Spielegenres besteht darin die virtuelle Welt zu erforschen, Aufgaben zu erfüllen oder den Avatar mit neuen Gegenständen auszurüsten. Meistens geht es dabei um die stufenweise Steigerung von Stärke und Geschicklichkeit des Charakters, um seine Gegner in einer Schlacht siegreich zu bekämpfen, während diese im

[12] Abbildung aus: http://pspdev1.com/wp-content/uploads/2006/08/snap0012.png, Stand: Nov. 2007
[13] Abbildung aus: http://www.planet-atari.de/img/articles/2496/Rayman-1.jpg, Stand: Nov. 2007

(a) (b) (c)

Abbildung 3.15.: Genre: Rennspiele: (a) Drittpersonansicht: *Need for Speed Underground 2*[14], (b) Egoperspektive: *Need for Speed Carbon*[15], (c) *Ski-Challenge*[16]

Spielverlauf an Stärke gewinnen. Die gängigsten Navigationsmetaphern sind dabei das Laufen und Teleportieren. Zusätzlich gibt es beispielsweise in *Gothic 2* die Möglichkeit einen Gegner anzuwählen und ihn immer im Blickfeld zu behalten. Durch Bewegungen erfolgen Rotationen um diesen Fixpunkt. Dies entspricht einer Inspektion, wie sie durch das Orbiting vorgestellt wurde. Am häufigsten sind Rollenspiele in Drittpersonansicht gehalten, wie *World of Warcraft* oder *Diablo*. Doch es gibt auch einige, die die Egoperspektive verwenden, wie *Dark Messiah*.

(a) (b) (c)

Abbildung 3.16.: Genre: Rollenspiele: (a) *World of Warcraft*[17], (b) *Dark Messiah*[18], (c) *Diablo*[19]

[14] Abbildung aus: http://www.hardwaremania24.de/ebay/mich/need2.jpg, Stand: Nov. 2007

[15] Abbildung aus: http://pcarena.pl/uploads/files/nfs_carbon/NFSC%202006-11-03%2017-12-54-20.jpg, Stand: Nov. 2007

[16] Abbildung aus: http://www.softaware.fr/screenshots/skichallenge.jpg, Stand: Nov. 2007

[17] Abbildung aus: http://images.blizzard-fr.com/image_1307.jpg, Stand: Nov. 2007

[18] Abbildung aus: http://www.winsoftware.de/newsbilder/darkmessiaomam.jpg, Stand: Nov. 2007

[19] Abbildung aus: http://www.battle.net/diablo2exp/images/other/worldevent.jpg, Stand: Nov. 2007

Strategiespiele:
Dieses Genre erfordert von dem Benutzer strategisches Denken und Handeln, da das Ziel dieses Genres vor allem der Aufbau von Städten oder eine erfolgreiche Kriegsführung ist. Die Orientierung über die gesamte Umgebung ist für ein strategisches Manöver unentbehrlich. Die Steuerung kann dabei rundenbasiert oder in Echtzeit eingesetzt werden. Dabei orientieren sich rundenbasierte Spiele an der Idee des Brettspiels. Jeder Akteur spielt nacheinander eine Runde, die beschränkt ist durch Zeit oder Anzahl an Aktionen, und kann auf die vorangegangenen Aktionen seiner Mitspieler reagieren. Am gängigsten sind inzwischen jedoch Echzeitspiele, da jeder Mitspieler kontinuierlich weiterspielen und direkt auf seinen Gegenspieler reagieren kann. Aus diesem Grund wird das Spiel sehr viel hektischer und schneller als ein rundenbasiertes Spiel. Die Benutzerperspektive erstreckt sich bei diesem Genre von der Vogelperspektive bis hin zu einer Art Drittpersonansicht. Meist ist es dem Benutzer möglich zwischen den Perspektiven zu Zoomen, um zwischen globaler Übersicht und lokalem Blick in zum Beispiel ein Kampfgeschehen zu wechseln. Die Navigationsmetapher besteht hierbei vor allem aus der *Laufen*-Metapher, wobei Möglichkeiten des Teleport gerne für lange Strecken eingebaut werden.
Aktuelle rundenbasierte Beispiele sind dabei: *Battle Isle*, *American Civil War - Gettysburg* (siehe Abb. 3.17);
echzeitbasierte Spiele: *Warcraft* (siehe Abb. 3.17), *Herr der Ringe: Schlacht um Mittelerde*, *Die Siedler*

(a) (b) (c)

Abbildung 3.17.: Genre: Strategiespiele: (a) Rundenbasiert während dem Zug: *American Civil War - Gettysburg*[20], (b) Rundenbasiert während der Bewegung: *American Civil War - Gettysburg*[21], (c) Echtzeit: *Warcraft 3*[22]

[20] Abbildung aus: http://static.twoday.net/spieledemos/images/american%20civil%20-war%20-%20gettysburg.jpg, Stand: Nov. 2007
[21] Abbildung aus: http://static.twoday.net/spieledemos/images/ausfuehrung.jpg, Stand: Nov. 2007
[22] Abbildung aus: http://www.secretdesign.de/content/images/Warcraft_3_Humans_-

Adventures:
Bei einem Adventure steht eine Geschichte im Mittelpunkt des Geschehens, die der Anwender durch eine Spielfigur erlebt. Der Benutzer wird durch diese geführt und trifft auf Rätsel, die er lösen muss, um weiterspielen zu können. Dabei ist ein Adventure meist friedfertig gestaltet im Gegensatz zu den Rollenspielen. Es kommt selten zu Kämpfen und ein Charakter kann in der Regel nicht weitergebildet werden. Das Ziel ist dabei das Erforschen der Welt, das Finden von Gegenständen und deren Anwendung an den richtigen Stellen. Dazu benötigt der Benutzer einen Überblick über die Welt und die unmittelbare Umgebung, weshalb die Drittpersonansicht als Standard gewählt ist. Der Spieler folgt einem Spielverlauf der vorher festgelegt ist, darum kann auch die halbautomatische *Pfad*-Metapher Anwendung finden. Adventures sind zum Beispiel *Maniac Manson* und *Monkey Island* (siehe Abb. 3.18).

(a) (b)

Abbildung 3.18.: Genre: Adventures: (a) *Maniac Manson 2*[23], (b) *Monkey Island 2*[24]

Virtueller Rundgang:
Ein virtueller Rundgang besteht aus einem begrenzten Umfeld, durch das der Benutzer geführt werden soll. Dabei können die verschiedenen Metaphern des Pfades eingesetzt werden um dem Benutzer entsprechend die Szene näher zu bringen. Das Ziel ist die Betrachtung der Umgebung sowie einzelner Objekte. Dafür benötigt der Benutzer keinen eigenen Avatar. Technisch ist eine Drittpersonansicht natürlich machbar, bietet dem Benutzer aber keine Vorteile gegenüber der Egosicht. Aus diesem Grund ist hier eine Egoperspektive zweckmäßiger. Vollautomatische virtuelle Rundgänge werden vor allem zu Präsentationszwecken gebraucht und können wie bei [JRH07] eine Fahrt durch ein virtuelles Hotelzimmer zeigen, um

fight_Orcs_screenshot%20%5B640x480%5D.jpg, Stand: Nov. 2007
[23] Abbildung aus: http://www.aikon.ch/c64/bilder/maniac2.gif, Stand: Nov. 2007
[24] Abbildung aus: http://www.elerik.de/monkey_2/files/monkeyisland2_2.jpg, Stand: Nov. 2007

sich dieses vorstellen zu können. Der Anspruch auf Realitätsnähe ist bei solchen Projekten besonders hoch. Auch für die Präsentation eines neuen Designs eines PKWs und ähnlichen Dingen kann ein virtueller Rundgang um den PKW und Kameraflug durch das Innere einen besseren Einblick geben. Statt nur zu Präsentieren kann der Benutzer auch interaktiv miteinbezogen werden, indem der Rundgang nicht vollautomatisch, sondern halbautomatisch oder über einen Führer verwirklicht wird. Der Benutzer kann dazu einem Führer folgen oder durch navigationsunterstützende Verfahren geleitet werden.

Virtuelle Montage:
Für eine virtuelle Montage wird meist keine ganze Welt benötigt, sondern lediglich eine kleine Szene, in der der Benutzer mit ihr interagieren kann. Um besser auf die virtuellen Gegebenheiten einzugehen gibt es meist spezielle Eingabegeräte. Die Navigation steht bei solchen Anwendungen an zweiter Stelle, da der Benutzer unter Umständen ausschließlich vor einem einzelnen Gerät steht, welches er bedienen soll und sich gar nicht durch die Szene bewegen muss. Durch die Notwendigkeit einer genauen Betrachtung der einzelnen Geräte wird ein Avatar hinfällig und stört die Sicht auf die sich im Fokus befindlichen Objekte. Da der Benutzer im wahrsten Sinne des Wortes selber Hand anlegen muss, ist die Egoperspektive zusätzlich intuitiver. Dabei kann der Benutzer auf verschiedene Weisen durch eine solche Szene navigieren. Ist die Montageanwendung eine Anleitung, so kann es einen genauen Pfad geben, damit der Benutzer weiß in welcher Reihenfolge er sich wann von welcher Stelle an eine andere zu bewegen hat. Die Notwendigkeit einzelne Objekte zu betrachten legt die Möglichkeit nahe einen *Inspektionspfad* zu nutzen, um ein schnelles Navigieren, zum Beispiel auf die andere Seite des Gerätes, zu ermöglichen. Ohne den *Inspektionspfad* müsste der Benutzer um das Objekt herum laufen, was die Navigation unnötig verlangsamen würde. Je nach Anwendung ist eine Kollision für die erfolgreiche Nutzung der Anwendung sinnvoll. Soll der Benutzer für eine Montage einen Widerstand sehen (oder auch spüren, je nach Eingabegerät) ist die Integration einer Kollisionserkennung notwendig. Für andere Anwendungen kann die Kollision vernachlässigt werden und die Kamera des Benutzers kann im Raum ohne Einschränkung fliegen. All diese Punkte müssen für individuelle Anwendungen entsprechend betrachtet und angepasst werden.

3.2. Navigationsunterstützende Verfahren

Navigationsunterstützende Verfahren werden zusätzlich zu den Navigationsmetaphern eingesetzt. Als navigationsunterstützend soll all das verstanden werden, was dem Benutzer dabei hilft in der virtuellen Welt effizienter zu navigieren. Da der Nutzer während einer Anwendung nicht zwingend dadurch am effizientesten unterstützt wird, indem er eine Vielzahl an Auswahlmöglichkeiten erhält, ist es denkbar, durch eine gezielte Einschränkung an Navigationsmöglichkeiten Verwirrung für den Anwender zu vermeiden. Hierzu kann die Einschränkung von den Entwicklern entweder vorab bei der Erstellung der Anwendung komplett integriert werden oder dem Benutzer wird während der Laufzeit der Anwendung eine vorsortierte Auswahl dargestellt, so dass er seine Navigationsmöglichkeit auswählen kann. Dazu können den Navigationsmetaphern während der Erstellung der Anwendung Prioritäten zugeordnet und dementsprechend für den Benutzer in einer Rangfolge veranschaulicht werden. Diese Darstellung der Auswahl für den Anwender stellt somit eine Vorsortierung dar, so dass er die zu bevorzugenden Navigationsmetaphern am einfachsten auswählen kann [HBL02].

Sie helfen dem Benutzer sich zu orientieren oder beeinflussen seine Navigationsmöglichkeiten direkt. Dabei fallen sie dem Benutzer während der Bedienung nicht notwendigerweise auf. Unauffällig sind dabei all die Möglichkeiten, die nicht erst durch einen Auslöser aktiviert werden, sondern den Benutzer die ganze Anwendung über begleiten, wie die Schattendarstellung.

Manche Verfahren wirken sich aktiv auf die Navigation des Benutzers aus und verändern die Bewegung der Spielfigur oder der Kamera. Andere bieten demgegenüber nur eine passive Unterstützung und überlassen dem Benutzer die Entscheidung, ob er die Hilfe annimmt oder nicht. Im Folgenden werden verschiedene aktive (Abschnitt 3.2.1) und passive Verfahren (Abschnitt 3.2.2) beschrieben. Die Erläuterungen der jeweiligen Optionen bieten ausschließlich einen Auszug an Möglichkeiten und stellen keine vollständige Liste aller denkbaren Metaphern dar. Dabei sollen die unterschiedlichen Ansätze veranschaulicht, die Auswirkung auf den Benutzer dargelegt und die Anwendungsgebiete für eine sinnvolle Realisierung eingegrenzt werden.

Die Verfahren können dabei so implementiert werden, dass sie erst durch eine Aktion des Benutzers ausgelöst werden oder bereits von Anfang an integriert sind. Die Aktion des Benutzers kann dazu sowohl eine bewusste als auch eine unbewusste Handlung sein. Durch ein bewusstes Auslösen

eines Verfahrens ist ihm klar was als nächstes in der virtuellen Welt passieren wird. Er verfolgt somit ein Ziel mit seiner Aktion. Durch unbewusstes Auslösen eines Verfahrens, kann eine Verwirrung eintreten, falls es nicht intuitiv ist.

Da in virtuellen Szenen verschiedene Gegebenheiten existieren, je nachdem ob die Szene sich innerhalb oder außerhalb von Gebäuden befindet, müssen sie verschiedenen Ansprüchen genügen. Entsprechend weisen die verschiedenen Verfahren jeweils unterschiedliche Vor- und Nachteile auf. Aus diesem Grund werden sie auch speziell in dieser Hinsicht betrachtet.

3.2.1. Aktive Verfahren

Unter aktiven Verfahren sollen die navigationsunterstützenden Möglichkeiten verstanden werden, welche die virtuelle Figur des Benutzers oder die Perspektive und somit den Blickwinkel des Benutzers in die Szene verändern. Als aktive Verfahren sollen die *Kamerasteuerung*, die *Bewegungssteuerung* und die *Kollisionserkennung* näher betrachtet werden.

Kamerasteuerung:
Die Kamera beschreibt, wie der Benutzer in die virtuelle Welt hineinblickt. Dabei wird die Steuerung der Kamera vor allem von der Benutzerperspektive beeinflusst, die grundsätzlich zu Beginn festgelegt wird.

Dabei bewegt sich die Kamera in der Egoperspektive über die Kopfdrehung des Avatars, also durch die direkte Steuerung des Benutzers. Bei der Drittpersonansicht befindet sich die Kamera nicht in dem Avatar, sondern hinter ihm und hat eine eigene Steuerung. Dazu gibt es zum einen die Möglichkeit, dass die Kamera sich jederzeit in einem bestimmten Abstand zu dem Avatar befindet. Sie bleibt somit steif, wie festmontiert, hinter dem Benutzer. Eine andere Option für diese Perspektive ist das Hinterherschwingen der Kamera bei einem Richtungswechsel des Avatars. Sie befindet sich dabei zwar auch in einem gewissen Abstand zum Avatar, aber sie ist nicht an den Avatar „montiert“. Die Kamera fährt sozusagen hinter dem Nutzer eine Bahn ab, um ihn immer im Blick zu behalten. Dabei kann sie selbst jedoch verschiedenen Kollisionsobjekten ausweichen und somit auf die Umwelt reagieren.

Statt nur von der Perspektive auszugehen ist es in manchen Situationen sinnvoll die Kamerasteuerung zu verändern und dem Benutzer somit zusätzliche Funktionen zu bieten. Hierzu gehört zum Beispiel das *Zoomen*. Diese Methode wird in der Egoperspektive kaum umgesetzt, da sich die Kamera in der virtuellen Figur befindet und der Benutzer sozusagen in

ihr feststeckt. In der Drittpersonansicht kann dies jedoch sinnvoll genutzt werden. Durch ein hinein- und hinauszoomen kann beispielsweise der Abstand zwischen Kamera und Spielfigur bestimmt und dem Benutzer somit ein Überblick über die Umgebung der Spielfigur verschafft werden. Vor allem bei besonderen Aufgaben, wie zum Beispiel dem Zielen durch ein Fernrohr oder der Benutzung der *Orbit*-Metapher, können so Abstände zu bestimmten Objekten verändert werden. Dies bietet eine Detailsicht auf die Objekte und verlässt trotz allem nicht den Spielekontext [TRC01].

Der Zoom kann vor allem auch dann hilfreich sein, wenn besonders kleine Dinge in der virtuellen Realität hervorgehoben werden sollen. Mit Hilfe einer kurzen automatischen Kamerafahrt kann eine Lokalisierung eines kleinen Objektes und der Einordnung dessen in der Gesamtszene erfolgen. Dazu betätigt der Benutzer einen entsprechenden Auslöser und die Kamera bewegt sich nun eine bestimmte Zeit automatisch und übernimmt die Navigation für ihn. Die Kamera bewegt sich hierzu von der aktuellen Position zu dem Startpunkt, beziehungsweise dem kleinen Objekt, und zoomt dazu unter Umständen heran. Dann zoomt sie soweit heraus, dass das ganze Objekt erfassbar, der kleine wichtige Punkt in dem Ganzen nun lokalisiert ist und das ganze Objekt global erfasst werden kann. Dann führt die Kamera eventuell zu einem in dem Zusammenhang weiteren wichtigen Punkt und letztendlich wieder zu der Ursprungsposition. Dies kann dem Benutzer zeitweise wie eine kurze Videosequenz vorkommen (siehe Abb. 3.19).

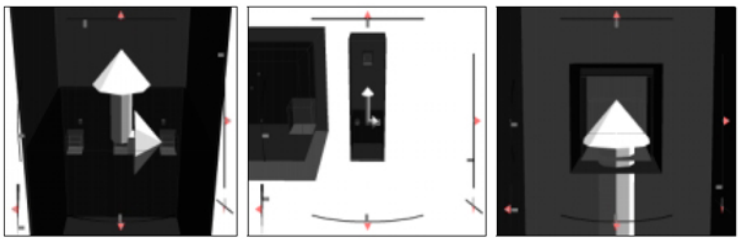


Abbildung 3.19.: *Kamerasteuerung*: Orientierungshilfe mit einer automatischen Kameraführung mit Hilfe der *Snake*-Metapher; Zunächst ist die Kamera nahe an dem Objekt, im Folgenden ist die Ansicht so weit hinausgezoomt, dass das ganze Objekt erfasst werden kann und zuletzt wird an den Endpunkt herangezoomt. Eine Hervorhebung der kleinen Punkte erfolgt zusätzlich mit einer sogenannten *Snake* (siehe *Besondere Hervorhebungen* (3.2.2)) [PFW98]. Abbildung nach: [PFW98]

Die Art der Anwendung des Zooms um Detailgriffe darzustellen, kann vor allem auch in Montageanwendungen helfen. Dazu können an den einzelnen Punkten während der Detailsicht zusätzlich entsprechende Aktionen aufgezeigt werden.

Auf den Anwender hat die Kamerafahrt insofern Auswirkung, dass sie, wenn sie sich zu ruckelig bewegt oder sich an eine Position bewegt, die für die Betrachtung der Szene unpraktisch ist, sehr verwirrend und auf Dauer störend sein kann. Da die Kamera den Blick in die virtuelle Welt für den Benutzers darstellt, wird er die Anwendung nicht weiter nutzen wollen, wenn die Führung zu verwirrend oder nicht intuitiv genug ist.

Die Auslöser können für eine *Kamerasteuerung* sowohl bewusst als auch unbewusst vom Anwender betätigt werden. Löst zum Beispiel das Eintreten in ein Gebäude eine neue Kamerapositionierung aus, indem sich die Kamera näher zum Boden und näher an den Avatar bewegt, so hatte der Benutzer dies nicht vorher beabsichtigt. Zoomt er jedoch die Ansicht, so war dies eine bewusste Entscheidung.

Bewegungssteuerung:
Die *Bewegungssteuerung* beschreibt die Art, wie die Bewegung des Avatars gesteuert wird. Ist kein Avatar vorhanden, so wirkt sich diese direkt auf die Kamera aus. In der Regel übernimmt das der Benutzer direkt durch bestimmte Eingabegeräte, die an das System angebunden sind. Das bedeutet, wenn er sich beispielsweise in der *Laufen*-Metapher befindet, kann er mit Hilfe der Richtungs- und Orientierungseingabe (z. B. mittels Maus und Tastatur) den Avatar bewegen.

Durch eine Teil-Automatisierung der Steuerung ist sie in sofern beeinflussbar. So z. B. durch die Einschränkung der Eingabemöglichkeiten unter Verwendung von Icons zur Navigation. Angenommen es existiert ein Pfad in der virtuellen Welt, auf dem sich eine virtuelle Figur bewegen kann, so erfolgt die Einschränkung der Bewegungssteuerung nun dadurch, dass nur noch über diese Icons navigiert wird, die sich in der Bedienoberfläche[25] der Anwendung befinden. Sprechen diese beispielsweise nur zwei Richtungen an, wie rechts und links, wurde die Bewegungsfreiheit eingeschränkt. Für eine Bewegung wurde dem Benuzter jedoch nicht die volle Kontrolle entzogen. Zudem muss die Bewegung durch eine Eingabe bewusst ausgelöst werden. Das kann vor allem für Anwendungen nützlich sein, bei denen beispielsweise eine Führung durch ein Museum simuliert werden soll (siehe auch [PC99]).

[25] Graphical User Interface(GUI)

Für den Benutzer ist jede Übernahme der *Bewegungssteuerung* eine Einschränkung in seiner Bewegungsfreiheit. Jedoch ist es je nach Anwendung sinnvoll den Nutzer von bestimmten Aufgaben zu befreien, damit dieser sich voll und ganz auf seine eigentliche Aktion konzentrieren kann. Ein Beispiel ist die bereits erwähnte Museumsführung, bei der die Bewegung lediglich „aktiviert“ werden muss.

Kollisionserkennung:
Die *Kollisionserkennung* ist ein Mittel, das für verschiedene Zwecke eingesetzt wird. Letztendlich ist es eine einfache Einschränkung des Benutzers in seiner Navigationsfreiheit. Der Benutzer hat durch eine aktive *Kollisionserkennung* nicht die Möglichkeit durch Wände oder Mauern zu laufen, was ihm ein realistischeres Gefühl gibt. Zusätzlich kann eine Kollision auch die Aufgabenbewältigung der Anwendung erschweren, da der Nutzer sich einen Weg überlegen - also über die Navigation nachdenken - muss, statt sich einfach durch alle vorhandenen Objekte an jeden beliebigen Ort bewegen zu können.

Für die Navigation kann eine Kollisionsabfrage deshalb als Hilfestellung betrachtet werden. Denn ist keine Kollision vorhanden und der Benutzer kann jeden Ort der Szene erreichen, so spielt für ihn die Navigation keine besondere Rolle mehr und der Sinn der Anwendung geht meist verloren. Mit Hilfe der Kollision kann der Anwender geführt werden, wie zum Beispiel in einem Labyrinth. Wäre die Kollision nicht vorhanden, würde er nicht auf den Weg achten, was aber genau das Ziel der Anwendung ist. Will sich der Benutzer hinter einer Wand verstecken, wie häufig in Egoshootern zu beobachten, ist es nicht sinnvoll, wenn die Wand keine Kollision auslösen würde. Ohne Kollision ist der Weg zu jeder Zeit in jede Richtung offen, auch ein Navigieren über Treppen oder Leitern kann nicht ohne *Kollisionserkennung* erfolgen.

Meist wird eine Kollision durch eine unbewusste Aktion ausgelöst, beispielsweise sobald zwei Objekte aufeinander treffen. Dies geschieht entweder durch den Anwender, der gegen ein Objekt stößt oder aber zwei andere Objekte, die er dabei beobachtet, wie sie kollidieren. So ist beispielsweise eine Kollision bei dem Zusammenstoß zwischen der virtuellen Figur und einer Wand meist nicht weiter erstaunlich für den Benutzer.

Kollision muss dabei nicht immer an jeder Stelle aktiv sein, so können auch Geheimgänge daraus resultieren, indem die Entwickler an einer Stelle, an der ein Anwender eine Kollision erwartet diese bewusst auslassen. So kann Kollision auch zum Spielverlauf und der Geschichte beitragen.

Die Anwendung der *Kollisionserkennung* innerhalb oder außerhalb eines Gebäudes unterscheidet sich nicht. Es liegt nahe, dass es ohne Kollision kein Innerhalb gibt, da der Benutzer die Wände durchlaufen kann. Ein reales „drinnen- und draußen-Gefühl“ entsteht erst durch Kollision mit den Wänden und der Notwendigkeit durch Türen Räume zu betreten und zu verlassen.

Die *Kollisionserkennung* ist somit wichtig für alle virtuellen Anwendungen, die einen Anspruch auf Realitätsnähe haben oder für deren Anwendungsziel die Kollision essentiell ist, wie bei einem Labyrinth. Ebenso auch bei Montageanwendungen, wo es wichtig ist, dass der Benutzer eine *Kollisionserkennung* zwischen Bauteilen erkennen muss. Keinen Zweck erfüllt die Kollision dagegen in virtuellen Modellierungstools und bei Kamerafahrten durch Prototypen. Hier soll jedes Teil betrachtet werden können, unabhängig davon, ob die Kamera in Wirklichkeit an diese Stelle gelangen kann oder nicht.

Die *Kollisionserkennung* und -behandlung kann dabei auf verschiedene Arten implementiert werden. So kann eine Kollision direkt über die Meshes der einzelnen Objekte oder über ihre *Bounding-Boxes* geprüft werden. Die *Meshes* bieten dabei eine genauere *Kollisionserkennung*, die jedoch aufwendiger in ihrer Berechnung ist. Die Kollisionsbehandlung kann dabei verschiedene Ziele verfolgen. Geht es darum den Avatar beispielsweise an einer Wand abprallen zu lassen, so übt entweder die Wand Kraft auf den Avatar aus und drängt ihn somit zurück oder die Position des Avatars wird neu berechnet. Diese beiden Ansätze sind grundlegend verschieden. Berechnet sich die Bewegung über die Kraft, so muss jedes Objekt unter anderem eine Masse haben. Wird die Berechnung über die Positionsbestimmung durchgeführt, so muss nur der Avatar betrachtet werden. Je nach Szenenaufwand, Realitätsbezogenheit und letztendlich Ziel der Kollision kann ein Verfahren ausgewählt werden.

3.2.2. Passive Verfahren

Unter passiven Verfahren sollen die navigationsunterstützenden Methoden verstanden werden, die dem Benutzer zusätzliche Informationen über die Welt geben. Dabei wird ihm freigestellt zu entscheiden, inwiefern er diese in seine Entscheidungen über seine weitere Navigation mit einbezieht. Die passiven Verfahren dienen dazu die virtuelle Welt schneller und intuitiver bedienbar, sowie verständlicher zu machen. Dabei sind einige dieser Methoden davon abhängig, dass der Benutzer sie bewusst verarbeitet. Andere wirken ganz unbewusst auf ihn, da er sie aus dem realen Leben bereits

kennt und während der Benutzung als vollkommen natürlich empfindet. Der grobe Überblick über die Methoden, die im folgenden vorgestellt werden, bietet eine kleine Auswahl über die vielen Möglichkeiten, die eine virtuelle Szene bietet, um den Benutzer passiv zu unterstützen.

- **Grafische Abstraktion:** Nicht alle Objekte voll detailliert anzeigen - *verhindert Informationsüberflutung*
- **Elision-Technik:** Unwichtige Objekte ausblenden - *verhindert Informationsüberflutung und das Übersehen wichtiger Objekte*
- **Schattendarstellung:** Schattendarstellung von Objekten - *Ermöglicht bessere Höhen- und Abstandsschätzungen*
- **Gitterdarstellung:** Darstellung eines Gitters auf dem Boden - *Ermöglicht eine bessere Abstandsschätzung und grobe Positionsbestimmung*
- **Koordinatendarstellung:** Angabe von Kompass, Radar oder Koordinaten - *Ermöglicht eine genaue Positions- und Richtungsbestimmung*
- **„World in miniatur" - Metapher:** Eine kleine Miniaturdarstellung der Welt in den Händen halten - *Ermöglicht ein Manipulieren der Welt aus globaler und lokaler Perspektive in einem Fenster*
- **Multiple Fenster:** Darstellung mehrerer Fenster - *Ermöglicht das Trennen von Informationen und und somit das Zuordnen spezieller Aufgaben für jedes Fenster*
- **Übersichtskarte:** Darstellung einer Übersichtskarte - *Ermöglicht einen globalen Blick auf die virtuelle Welt*
- **Besondere Hervorhebungen:** Hervorhebungen wichtiger Objekte/ Orte mit Hilfe verschiedener Darstellungen - *Ermöglichen das Steuern der Aufmerksamkeit des Benutzers*
- **Markierungen:** Markierung einzelner Orte durch besondere existierende oder vom Benutzer hinzugefügte Markierungselemente - *Ermöglichen eine schnelle Rückkehr an einen Ort*
- **Transparenzen:** Objekte semitransparent einblenden - *Ermöglichen einen schnelleren Durchblick*

Im Folgenden sollen diese Verfahren im Einzelnen genauer betrachtet werden.

Grafische Abstraktion:
Die *Grafische Abstraktion* beschreibt ein Verfahren, um Objektabbildungen in bestimmten Situationen zu vereinfachen. Das bedeutet es werden nicht alle Objekte im gleichen Detailgrad dargestellt, sondern nach unterschiedlichen Kriterien bewertet und entsprechend abstrahiert oder detailliert angezeigt. Die Entscheidung darüber geschieht automatisch während der Anwendung und wird ständig aktualisiert. Dieses Verfahren orientiert sich an der Wahrnehmung des Menschen in der Realität, da auch hier entfernte Objekte nicht so detailliert wahrgenommen werden wie nahe Objekte. Demnach ist eine Gewöhnung an diese Darstellungsform nicht nötig. Zusätzlich kann dem Benutzer jedoch auch die Möglichkeit gegeben werden, durch Interaktion mit den Objekten den entsprechenden Abstraktionsgrad des Objektes selber zu wählen [KS98].

Durch die vereinfachte Darstellung einiger Objekte und der detaillierten Anderer, kann die Orientierung gelenkt und darüber hinaus dem Benutzer eine Flut an für ihn irrelevanten Informationen erspart werden. Neben dem Vorteil der Orientierungshilfe steigert die *grafische Abstraktion* auch die Performanz der Anwendung, da nicht alle Objekte inklusive all ihren Details dauerhaft neu gerendert werden müssen [KS98].

Den größten Zweck erfüllt die *grafische Abstraktion*, wenn große Entfernungen zwischen den Positionen der virtuellen Objekte liegen. Darum ist die Anwendung dieses Verfahrens in einer virtuellen Szene, die eine Welt außerhalb von Gebäuden darstellt, sowohl für die Performanz wie auch für die Orientierung besser geeignet, als eine Szene die das Innere widerspiegelt, denn innerhalb von Gebäuden sind die überschaubaren Strecken wesentlich kürzer.

Dabei werden als typische Beispiele von Vereinfachungen nach Krüger und Stahl in [KS98] die Folgenden betrachtet:

- Substitution von Farben: Ähnliche Farben durch eine Einzige oder durch Grauwerte ersetzt
- Vereinheitlichung von Strichstärken des Objekts
- Vereinfachung der Konturform des Objekts
- Filtern und Verschmelzen von Objektteilen zu einem größeren Objektteil
- Skalieren von Objektteilen

Dabei dürfen Objektteile nur soweit vereinfacht werden, dass trotzdem ein Bezug zu einem Objekt hergestellt werden kann. Eine Auswahl an

Algorithmen wird in [KS98] vorgestellt und soll hier als weiterführende Quelle dienen.

Als Ansatz für ein Verfahren zur Abstraktion von weit entfernten Gegenständen ist vor allem das *Level-of-detail*- Verfahren (siehe Abb. 3.20) gängig, das weit entfernte Gegenstände in einem anderen Detailgrad darstellt wie nahe gelegene Objekte [KS98]. Darum eignet sich diese automatische Methode jedoch kaum für das Innere eines Gebäudes.

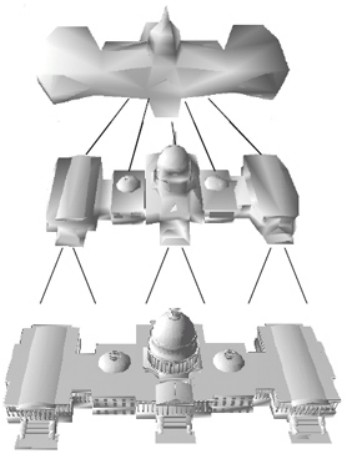


Abbildung 3.20.: Beispiel einer *grafischen Abstraktion*; Abbildung aus: [KS98]

Um speziell auch innerhalb von Räumen den Benutzer zu unterstützen bringen automatische Verfahren kaum einen Vorteil. Über eine Interaktion zwischen Anwendung und Benutzer kann dieser seine Priorität auf verschiedene Objekte legen. Diese werden demnach detailliert und andere abstrahiert dargestellt. Diese Möglichkeit kann natürlich auch außerhalb von Gebäuden genutzt werden, um den Benutzer seine Fokussierung wählen zu lassen. Anhand des Gebäude-Models soll diese Vorstellung erläutert werden.

In der Abbildung 3.21 ist zu sehen, wie der Benutzer die Abstraktion interaktiv beeinflussen kann. Durch die Gruppenauswahl entscheidet er sich für einen Fokus auf das ganze Objekt. Die Abstraktion wird demnach auf alle Teile des Gebäudes gleich angewandt. Wählt er die Fokussierung auf einen bestimmten Teil des Objekts, so wird dieser immer detailliert dargestellt, wobei andere Teile des Gebäudes weiterhin automatisch je nach Entfernung abstrahiert werden. Durch die Einzelauswahl eines Objekts wird

dieses abstrahiert und alle anderen Objekte werden nicht weiter betrachtet. Somit kann der Benutzer gezielt bestimmen, welche Teile detailliert oder abstrahiert dargestellt werden.

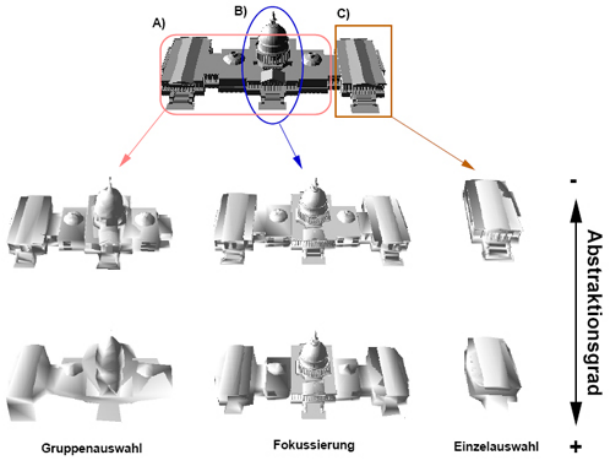


Abbildung 3.21.: Beispiel einer *grafischen Abstraktion* durch Interaktion des Benutzers; Abbildung aus: [KS98]

Elision-Technik:
Das gleiche Ziel verfolgt die in [PFW98] beschriebene *Elision*[26]-Technik. Sie wird speziell für die Darstellung von Graphen genutzt, kann aber auch auf die Vorstellung einer dreidimensionalen Umgebung übertragen werden. Das Grundprinzip besteht darin, dass unwichtige Teile nicht abstrahiert werden, sondern erst gar nicht sichtbar sein sollen, bis sie für den Benutzer relevant sind. Die Entscheidung, wann etwas sichtbar gemacht wird, ist bei einem dargestellten Graph selbstverständlich einfacher, da es immer einen aktuell gewählten Knoten gibt und seine Nachbar- und Kinderknoten somit die höchste Priorität für eine Darstellung erhalten. In einer dreidimensionalen Welt wird dies schwieriger und könnte wiederum über die Entfernung ausgewertet werden oder durch andere Kriterien, mit denen die Relevanz von Objekten beurteilt werden kann. Dies kann für jede Anwendung individuell unterschiedlich sein.

Die beschriebene Technik hat in ihrer Wirkung auf den Benutzer einen ähnlichen Effekt wie die vorgestellte *grafische Abstraktion*. Sie bietet eine Orientierungshilfe und eine Aussparung von Informationen, die für den

[26] engl. für Auslassung

Anwender zum aktuellen Zeitpunkt unwichtig sind. Zusätzlich wird die Auslastung des Systems verringert.

Schattendarstellung:
Wenn ein Benutzer eine Anwendung in einer virtuellen Szene verwendet, in der kein Schatten dargestellt wird, so kann er die Größe seines Avatars und der Objekte seiner Umgebung inklusive Höhen und Entfernungen nicht gut einschätzen. Das Licht bietet den großen Vorteil, dass es sich auf alle Objekte in der Umgebung auswirkt und so eine globale Tiefeninformation liefert und Abstände, vor allem zum Boden, aber auch zu den benachbarten Objekten, sichtbar macht [DS93]. Wie in Abbildung 3.22 zu sehen ist, kann der Benutzer erkennen, dass sich die rechte virtuelle Figur in einer Laufbewegung befindet. Doch nur durch die Schattendarstellung wird deutlich, dass er keinen Kontakt zum Boden hat. Ohne Schatten könnte er diesen ebenso berühren, dazu wäre seine Position lediglich leicht versetzt.

Abbildung 3.22.: *Schattendarstellung*: *Guild Wars*[27]

Der Benutzer kennt *Schattendarstellungen* aus dem realen Leben und kann diese Information sofort verarbeiten ohne wirklich darüber nachdenken oder es interpretieren zu müssen. Ein ambientes Umgebungslicht, das der Szene eine Grundhelligkeit verleiht ist meistens vorhanden, so dass der Benutzer zunächst auch ohne aktivieren eines speziellen Auslösers nicht komplett im Dunkeln steht. Zusätzliches Licht kann dem Anwender geboten werden, indem Auslöser, wie beispielsweise Lichtschalter, in der virtuellen Welt vorhanden und benutzt werden können. Somit ist diese Methode in einer Szene, die sich auf eine Umgebung außerhalb von Gebäuden bezieht ebenso relevant wie für eine, die das Innere eines Gebäudes darstellt.

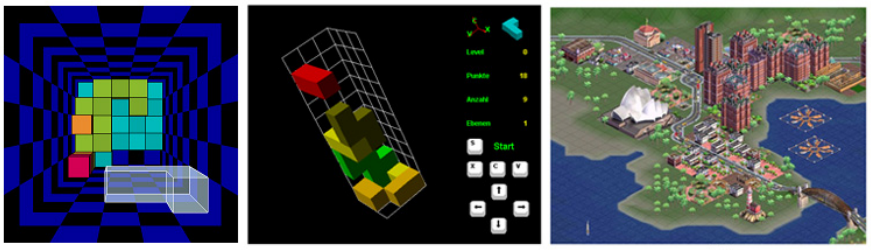
[27] Abbildung aus: http://img.generation-nt.com/photos/00006267.jpg, Stand: Nov. 2007

Gitterdarstellung:
Das Gitter wird als eine Art Netz auf den Boden der Welt projiziert und dient dem Benutzer jederzeit als Unterstützung für eine Größenabschätzung, Entfernungsmessung und Orientierung, da es immer sichtbar ist [DS93]. Ob der Benutzer auf das Gitter achtet oder sich mit Hilfe anderer Dinge orientiert bleibt dabei ihm selbst überlassen.

Die Darstellung ist besonders für solche Anwender intuitiv, die im täglichen Leben mit kariertem Papier arbeiten und Zeichnungen darauf anfertigen. Bauzeichner, 3D-Modellierer und ähnliche Berufsgruppen sind es gewohnt mittels Gitternetzstrukturen, sowohl auf Papier als auch in einer Anwendung am Computer zu arbeiten und Größen und Entfernungen mit Hilfe der Gitterstrukturen abzuschätzen.

Ein Gitter hilft sowohl innerhalb als auch außerhalb von Gebäuden zur besseren Einschätzung und Orientierung. Innerhalb von Gebäuden ist der Effekt jedoch geringer, da der Benutzer hier bereits andere markante Anhaltspunkte wie die Wände der Räume etc. wahrnimmt, die ihm bei der Einschätzung helfen.

Anwendung finden *Gitterdarstellungen* in virtuellen Anwendungen, wenn eine genaue Einschätzung notwendig ist, wie zum Beispiel in 3D-Tetris oder Simulationen wie SimCity (siehe Abb. 3.23). In Rollenspielen und ähnlichen Genres, in denen die realitätsnahe Darstellung ein wichtiger Aspekt ist, werden Gitterstrukturen in der Regel nicht angewendet.



(a) (b) (c)

Abbildung 3.23.: Anwendungsbeispiele: *Gitterdarstellungen*; (a)[28]und (b)[29]: verschiedene 3D-Tetris, (c) SimCity[30]

[28] Abbildung aus: http://jordy.gundy.org/wp-content/uploads/2006/09/3DTRIS_mini.png, Stand: Nov. 2007
[29] Abbildung aus: http://www.3dtetris.de/3dtetris.gif, Stand: Nov. 2007
[30] Abbildung aus: http://www.cincinnati.com/freetime/games/reviews/img/simcity3.jpg, Stand: Nov. 2007

Koordinatendarstellungen:
Um dem Benutzer eine Orientierungshilfe zu geben, kann auch eine *Koordinatendarstellung*, ein *Radar* oder ein *Kompass* dienen. Dabei hilft ein *Radar* einen globalen Überblick zu erhalten und die Nähe des eigenen Avatars zu anderen Objekten einzuschätzen. Die *Koordinaten* dienen mehr der Positionsbestimmung relevanter Orte. Der *Kompass* hingegen hilft dem Anwender zur Bestimmung von Richtungen und dem lokalen Standort seiner Figur.

Koordinatendarstellungen effizient zu nutzen muss der Benutzer erst lernen, da er die Koordinaten der Welt nicht kennt, bevor er nicht mindestens einmal hindurch gelaufen ist. Mit der Zeit fällt die Benutzung dieser Orientierungsmöglichkeiten jedoch leichter. Für die Anwendung muss er mitdenken und aktiv auf diese Hilfe achten. Besonders hilfreich sind diese Darstellungen für eine Lokalisierung von Objekten oder Orten (z. B. einem Startpunkt) [PC99].

Abbildung 3.24.: *Koordinatendarstellungen*: (a) Anwendungsbeispiel eines *Kompass* (*Guildwars*) in der Übersichtskarte (bzw GUI)[31], (b) ein *Radar Far Cry*[32], (c) versch. möglichen *Koordinatendarstellungen* in dem Flugzeugsimulator *X-Plane 8*[33]

In virtuellen Szenen werden diese *Koordinatendarstellungen* speziell in der GUI eingearbeitet (siehe Abb. 3.24) oder in die Geschichte der Anwendung integriert, indem sie beispielsweise dem Spieler als Gegenstand mitgegeben werden. In geografischen Informationssystemen (GIS) dagegen werden Nordpfeile, Kompassdarstellungen und Koordinaten als Pflicht integriert, um die Informationen exakt und richtig wiederzugeben.

[31] Abbildung aus: http://www.pcmasters.de/pics/Compilers/GuildWars1.jpg, Stand: Nov. 2007
[32] Abbildung aus: http://media2.playstadium.dk/img/spartan/farcry_360_1.jpg, Stand: Nov. 2007
[33] Abbildung aus: http://home.planet.nl/~vaals000/x-plane/m20j-002.jpg, Stand: Nov. 2007

„World in miniature"- Metapher:
Diese Metapher gibt dem Benutzer die Möglichkeit die Welt sowohl aus seiner Perspektive, als auch zusätzlich aus einer globalen Sicht zu sehen und zu manipulieren. In dieser Metapher hält der Spieler die Welt in seinen Händen [CS01]. Dies ist der Unterschied zu einer Übersichtskarte, die in einem extra Fenster angezeigt wird. Der Benutzer kann selber entscheiden, ob er die virtuelle Welt auch aus globaler Sicht manipulieren möchte. Er muss diese Funktion nicht benutzen. Die Verwendung dieser Metapher ist ähnlich wie die einer Übersichtskarte.

Multiple Fenster:
Multiple Fenster teilen den Bildschirm in verschiedene Bereiche, durch welche dem Benutzer meist verschiedene Ansichten [PFW98] wie zum Beispiel eine globale als auch lokale Sicht auf die virtuelle Szene ermöglicht wird. Die Verwendung von *multiplen Fenstern* ist vor allem für Übersichtskarten üblich, die dem Benutzer somit zeitgleich zur eigentlichen Perspektive eine andere Sicht auf das Geschehen oder ein Objekt liefern. Aber auch beispielsweise Inventarlisten oder Zusatzinformationen werden getrennt von dem „Blick" in die Welt dargestellt, um eine Überlappung mit der Welt zu verhindern.

Übersichtskarte:
Die *Übersichtskarte* wird sehr häufig eingesetzt und gehört in vielen Spielegenres zu der Standard-Ausrüstung. Entsprechend soll hierauf ein besonderes Augenmerk gelegt werden und eine gründlichere Betrachtung stattfinden.

Das Ziel besteht darin, dass der Anwender sich selbst orientieren kann. Gibt es also keinen festgelegten Pfad und dem Benutzer steht es frei sich in der Welt zu bewegen, so kann er sich so sehr gut selbst zurechtfinden [CRI03]. *Übersichtskarten* werden dabei häufig in extra Fenstern zur Verfügung gestellt und bieten somit den Vorteil der bereits erwähnten *multiplen Fenster*. Somit ist die Karte immer sichtbar. Diese Art der Kartendarstellung wird auch als *Minimap* bezeichnet. Karten müssen jedoch nicht immer eingeblendet sein, sondern können der Metapher entsprechen einen Stadtplan aus der Tasche zu nehmen. Der Vorteil daran ist, dass in dem Moment der ganze Bildschirm für die Karte genutzt werden und die Karte somit detailliertere Informationen enthalten kann. Dies hat jedoch zur Folge, dass der Benutzer nicht parallel auf die Karte und in die Welt blicken kann. Karten können zusätzlich je nach Anwendungsart auch interaktiv sein.

Befindet sich der Benutzer in der Egoperspektive und hat eine Karte zur Verfügung, so gibt diese ihm die Möglichkeit die Welt aus einer anderen Perspektive zu betrachten. Er betrachtet die Szene somit aus einer abstrakteren Perspektive und bekommt auf diese Weise Zusatzinformationen, die ihm alleine aus der Egoperspektive nicht ersichtlich sind [CRI03].

Die *Übersichtskarte* enthält nach [DS93]:

- *Wege*, die als separierende Linien dienen und auf denen sich der Benutzer bewegen kann.
- *Kanten*, die als separierende Linien dienen und durch die der Benutzer nicht hindurchgehen kann, wie beispielsweise Wände.
- *besondere Markierungen* sind Objekte mit scharfen Konturen im Gegensatz zu der Umgebung, wie beispielsweise Gebäude.
- *Einteilungen* dienen als logische oder physikalische Gebietseinteilungen.

Wie eine Karte dargestellt ist und funktioniert hängt vor allem von der speziellen Anforderung der Anwendung an den Benutzer ab. In dem Ansatz von Darken und Sibert werden in [DS93] drei Punkte als wichtig für die Darstellung angenommen:

1. Der Benutzer muss mindestens zu zwei Punkten aus der Welt, zwei Punkte in der Karte zuordnen können, um sich anhand dieser zu orientieren.
2. Die Karte muss zu der Welt ausgerichtet sein, also die Linie zwischen zwei Punkten im virtuellen Raum soll parallel sein zu einer Linie zwischen den beiden korrelierenden Punkten auf der Karte.
3. Oben auf der Karte wird immer das zu sehen sein, was vor dem Benutzer liegt.

Dies hat zur Folge, dass die Karte rotiert, wenn sich der Benutzer dreht. In diesem Fall sprechen Darken und Sibert von *blickabhängigem Verhalten* der Karte. Die Karte ist somit nicht eingenordet. Um dennoch die ganze Karte als Überblick angezeigt zu bekommen, ist der Benutzer nicht Mittelpunkt, sondern bewegt sich auf ihr. Wenn die Welt so groß ist, dass die Karte sowieso nur einen Ausschnitt zeigt, so kann es auch sinnvoll sein, wenn der Benutzer immer im Mittelpunkt der Karte steht und sich bei Drehung

und Bewegung des Avatars letztlich in der Karte die Welt um den Avatar dreht. Ansonsten steht der Avatar in der Welt und bewegt sich in ihr, nur die Karte rotiert mit, damit sein Blick nach vorne immer auf der Karte nach oben entspricht. Entsprechend sind auch seine Bewegungen auf der Karte niemals spiegelverkehrt zu denen in der Welt. Ein Navigieren nur mit Hilfe der Karte und während dem Schauen auf die Karte wird somit erleichtert; der Gesamtüberblick über die virtuelle Welt kann dabei jedoch verloren gehen.

Dieser Ansatz beschreibt dabei nur eine mögliche Art der Umsetzung. In vielen Spielen wird die Karte häufig auch statisch angezeigt. Ihr Verhalten wird demzufolge als *weltabhängig* bezeichnet. Ihre Koordinaten bleiben immer dieselben. Sie zeigt einen Überblick über die, falls möglich, gesamte Welt aus der Vogelperspektive. Der Benutzer befindet sich an einer Stelle auf der Karte und bei Drehungen und Bewegungen wird ausschließlich das Symbol des Avatars auf der Karte transliert, nicht aber die Karte selbst. Dies entspricht der Betrachtung, dass sich auch im realen Leben nicht die Welt um das Individuum dreht, sondern das Individuum sich in der Welt bewegt. Dabei ist die Orientierung über den gesamten Raum gut und bleibt die ganze Anwendung lang erhalten wobei der Blick auf die Karte konsistent bleibt. Ein Navigieren nur mit der Karte oder aufgrund dieser wird dadurch erschwert, dass der Benutzer, wenn er auf der Karte nach unten läuft spiegelverkehrt navigieren muss. Bei Darstellung dieser Kartenvariante führt der Anwender meistens keine Bewegung aus während er die Karte bedient, da dies sonst zu Verwirrung führen kann.

Die *Übersichtskarte* hilft im Gegensatz zu einer Koordinatendarstellung vor allem am Anfang für den groben Überblick, da sie eine globale Außenansicht darstellt. Sie enthält jedoch keine oder wenige Detailinformation [dSGA+00]. In [PC99] wird die Karte auch wie folgt beschrieben:

„A SORT OF MICROCOSM REFLECTING SOME USEFUL PROPERTIES OF THE MACROCOSM IN WHICH PEOPLE ARE INSERTED“

Manche Karten zeigen natürlich auch Grundrisse von Gebäuden, falls sich die virtuelle Figur vor allem innerhalb dieser bewegt [PC99]. Zur Orientierung speziell in Gebäuden können stattdessen auch Raumnummern hilfreich sein [CRI03].

Wenn der Nutzer eine *Minimap* angezeigt bekommt oder die Möglichkeit besteht eine *Übersichtskarte* „aus der Tasche zu ziehen“ und sie sich somit extra anzeigen zu lassen, so steht ihm trotzdem jederzeit frei diese Hilfe nicht zu nutzen. Besonders vorteilhaft ist die Karte, wenn sie auch Interaktionsmöglichkeiten bietet. Dies kann zum Beispiel genutzt werden um

Objekte anzuwählen, die sich in der virtuellen Szene nicht in Reichweite befinden. Auch die Anwendung der vorgestellten *Teleport*-Metapher bringt über die Karte den Vorteil schnell an einen entfernten Ort gelangen zu können. In Strategiespielen navigiert der Benutzer meist schneller über die Karte.

Abbildung 3.25.: Anwendungsbeispiele für *Übersichtskarten*: (a) Strategiespiel: *Anno 1701*[34], (b) Rollenspiel: *Gothic 2*[35], (c) Rollenspiel: *Gothic 3*,(d) Strategiespiel: *Warcraft 3*[36], (e) Strategiespiel: *Age of Empires 3*[37], (f) Rennspiel: *Need for Speed - Underground 2*[38], (g) Egoshooter: *Battlefield 2142*[39]

Karten finden sich dabei nicht nur in Simulationsspielen, sondern auch in diversen Genres und Anwendungen. Eine kleine Auswahl stellen die Abbildungen in 3.25 dar. Dabei zeigen (a) - (c) Möglichkeiten eine Karte darzustellen, die bildschirmfüllend ist und extra eingeblendet werden muss. Wohingegen die Karten in den Beispielen (d)-(g) *Minimaps* sind.

[34] Abbildung aus: http://aol.4players.de/grafik/premium/Screenshots/15/05/1695032-medium.jpg, Stand: Nov. 2007

[35] Abbildung aus: http://img392.imageshack.us/img392/2507/gothickarte6qw.jpg, Stand: Nov. 2007

[36] Abbildung aus: http://games4mac.de/media_g4m/reviews/warcraft3/3.jpg, Stand: Nov. 2007

[37] Abbildung aus: http://softkey.com.ua/images/upload/1/8ed376ace9f37d569f55e9289a2-4ec80.jpg, Stand: Nov. 2007

[38] Abbildung aus: http://images.ea.com/eagames/official/nfs/underground2/us/screenshots/contest_winners/reborngg/RebornGG_001.jpg, Stand: Nov. 2007

[39] Abbildung aus: http://www.gamestar.de/aktuell/blog/wp-content/uploads/2006/10-/knife-kill.jpg, Stand: Nov. 2007

Besondere Hervorhebungen:
Soll die Ausrichtung des Anwenders nicht automatisch gelenkt werden, wie über eine automatisierte Kamerasteuerung, so können einzelne Orte oder Objekte für ihn hervorgehoben werden, damit er auf sie aufmerksam wird, ohne dass er die Navigation abgeben muss. Somit hat diese Methode keinerlei direkte Auswirkung auf die Navigation des Benutzers. Er kann auf hervorgehobene Dinge achten und ihnen folgen, muss dies aber nicht. Die Entscheidung bleibt dabei ihm selbst überlassen.

Ob ein Punkt (sowohl ein Ort, als auch ein Objekt) für den Benutzer besonders auffällig sein soll, kann automatisch voreingestellt sein, indem die interessanten Orte/Objekte zu jeder Zeit der Anwendung hervorgehoben werden und somit immer ein Fokus auf ihnen liegt, egal wann der Anwender daran vorbeiläuft. Durch die Festlegung zuvor, kann keine individuelle Reaktion auf das Verhalten des Benutzers erfolgen. Eine ständige Hervorhebung würde sich genau dann anbieten, wenn der Ort/das Objekt für den Anwender immer interessant ist und nicht an Relevanz verliert, sobald er einmal dort war.

Dagegen kann es aber auch sein, dass die Objekte nur zu einem bestimmten Moment (einem bestimmten Spielstand oder abhängig von irgendeinem Ereignis/Auslöser) hervorgehoben werden sollen. Dazu muss die Methode dynamisch auf die Eingaben des Anwenders reagieren.

Es gibt dabei viele Möglichkeiten und Beispiele wie diese *Hervorhebungen* aussehen können. Zur Veranschaulichung werden einige Beispiele herausgegriffen (siehe Abb. 3.26), um die Methodik anhand dieser zu erläutern.

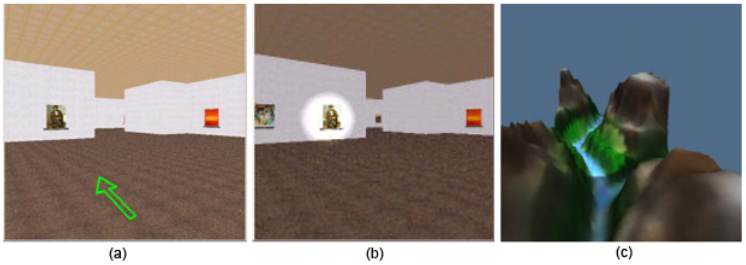


Abbildung 3.26.: *Besondere Hervorhebung* durch: (a) Pfeil auf dem Boden ([HBL02]), (b) Spotlight ([HBL02]), (c) Spotlight ([HW97])

Eine Variante ist dabei ein Spotlight. Dieses wird auf ein interessantes Objekt geworfen, falls dieses hervorgehoben werden soll (siehe Abb. 3.26) [HW97]. Eine andere Option wäre es, den Richtungsvektor zu dem nächs-

ten point-of-interest mit Hilfe eines Pfeils auf dem Boden darzustellen. Somit hat der Benutzer auch hier die Auswahl ob er dem Pfeil folgen will oder nicht. Diese Möglichkeit ist auch dann einsetzbar, wenn der Nutzer den interessanten Punkt noch nicht sieht [HBL02]. In dem Spiel *Unreal Tournament 3* (siehe Abb. 3.27) existieren zu diesem Zweck dreidimensionale Pfeile, die die Richtung angeben und fortlaufend zum Ziel fliegen. Somit wird dem Benutzer die Richtung gezeigt und er kann sich langsam dem Zielpunkt nähern.

Abbildung 3.27.: Anwendungsbeispiel: *Besondere Hervorhebung* in dem Shooter *Unreal Tournament 3*

Gut um ein kleines Objekt hervorzuheben ist immer ein dreidimensionales Objekt, wie beispielsweise die in [PFW98] vorgestellte *Snake*. Dies ist ein dreidimensionaler Pfeil, dessen Rumpf aus zwei Teilen besteht, so dass er auch die Möglichkeit bietet um die Ecke zu zeigen (siehe Abb. 3.28).

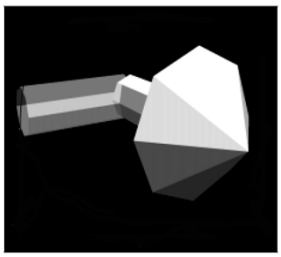


Abbildung 3.28.: *Besondere Hervorhebung*: *Snake*; Abbildung nach: [PFW98]

Mit Hilfe der *Snake* können auch kleine Punkte als wichtig angedeutet werden, die sonst durch ein zu kleines Spotlight nicht auffallen würden. Bei einem größeren Spotlight richtet sich dieses nicht mehr speziell auf das kleine Objekt. So kann es, obwohl es durch einen großen Lichtkegel hervorgehoben wird, als unwichtiges Element wahrgenommen und folglich übersehen werden. Eine zusätzliche Vereinfachung der Lokalisierung

von kleinen Objekten und Zuordnung dieser in der Gesamtszene kann mit Hilfe der Kamerasteuerung, wie in Abschnitt 3.2.1 erläutert, erfolgen.

Ist die Hervorhebung abhängig von einem bestimmten Ereignis, so kann der Benutzer über bestimmte Schlüsselaktionen die Situation in der Anwendung beeinflussen. Diese Orte/Objekte sind eventuell nach einmaligem Betrachten/Benutzen irrelevant und werden dann wieder „normal“ dargestellt. Für den Fall eines Museumsganges beispielsweise ist es sinnvoll immer den nächsten interessanten Punkt hervorzuheben, aber nicht alle gleichzeitig. So kann der Nutzer einen Pfad entlang geführt werden, ohne in der Navigation eingeschränkt zu sein. Das Spotlight bietet sich für einzelne Objekte an, die nur in einem Moment „angestrahlt“ werden. Wenn sie uninteressant geworden sind, stehen sie nicht mehr im „Rampenlicht“ und das Spotlight wird dort deaktiviert.

Markierungen:
Markierungen sind besonders herausragende Objekte die dem Benutzer bei der Orientierung helfen. Hierzu gehören vor allem große Gebäude oder auffällige Objekte, beispielsweise ein Kirchturm oder ein Springbrunnen. Diese sogenannten *Landmarks* sind fest in der Anwendung integriert [dSGA$^+$00], ohne dass der Benutzer dies beeinflussen kann. Ob er ihnen Beachtung schenkt bleibt wiederum ihm überlassen.

Im Gegensatz dazu können zusätzlich weitere Markierungspunkte entstehen, indem eine bewusste Interaktion und somit die Betätigung eines Auslösers stattfindet. Dazu kann der Benutzer grafische *Markierungen* setzten und sich somit einen speziellen Ort markieren oder aber sich einen ganzen Weg mit Hilfe wiederholender *Markierungen*[40] merken, um sich eine Rückkehr zu erleichtern. Die *Markierungen* können dabei unterschiedlichste Formen und Farben haben. Am besten bestehen sie dabei jedoch aus dreidimensionalen geometrischen Formen, die sonst nicht in der Welt integriert sind, um sie einfach unterscheiden zu können. Dreidimensionale Formen haben dabei den Vorteil, dass sie in der virtuellen Szene angeordnet werden können und von jeder Richtung aus sichtbar sein können.

Diese beiden Arten von *Markierungen* sind vergleichbar mit der Navigation im *World Wide Web*. Die *Landmarks* repräsentieren dabei die Links auf einer Homepage, wobei die sogenannten *breadcrumbs* die Einträge in der eigenen Favoritenliste darstellen [dSGA$^+$00].

Für *Markierungen* ist es wichtig, dass diese auch über weite Strecken zu sehen sind oder auf einer vorhandenen Übersichtskarte vermerkt werden,

[40] auch *breadcrumbs* genannt [DS93] [dSGA$^+$00]

damit sie dem Benutzer immer zugänglich sind und somit als globale Anhaltspunkte dienen [DS93]. Auf einer Karte sollen diese *Markierungen* aus den üblichen Darstellungen hervorstechen, um gut unterschieden zu werden. Zusätzlich können auch Miniaturwelten dienen, um *Markierungen* zu finden. Dazu muss der Benutzer natürlich zwischen den beiden verschiedenen Koordinatensystemen (der virtuellen Welt und der virtuellen Miniaturwelt) umschalten [TRC01].

Vor allem außerhalb von Gebäuden ist es gut dem Benutzer die Möglichkeit zu bieten *Markierungen* selbst zu erstellen. Innerhalb von Gebäuden erfüllen diese keinen großen Zweck, da sie beim Verlassen nicht mehr sichtbar sind oder bereits durch einen Raumwechsel an Bedeutung verlieren, außer bei einem Labyrinth beispielsweise.

Transparenzen:
In der Regel werden Gegenstände in virtuellen Szenen so dargestellt, wie sie aus der Realität bekannt sind, so dass sie wiedererkannt werden. Entsprechend sind die meisten Gegenstände *opak* (vollkommen undurchsichtig), es sei denn es handelt sich um Materialien wie Glas. Auch navigationsunterstützende *Transparenzen* geben dem Benutzer die Möglichkeit durch etwas hindurchschauen zu können. Einerseits bedeutet dies einen Durchblick zu ermöglichen, andererseits verrät diese Möglichkeit unter Umständen auch zuviel von dem, was sich hinter dem Gegenstand/Objekt befindet [CS01].

Besteht die Aufgabe beispielsweise aus der Suche nach Gegenständen, so erleichtert ihm eine semi-transparente Umgebung zwar diese Aufgabe, doch geht der Sinn einer Suche verloren, wenn keine Gegenstände verborgen sind. Dafür verhindert eine semi-transparente Umgebung an manchen Stellen jedoch ein unnötiges Laufen an Orte, die keine Bewandtnis haben [CS01].

Vorteilhaft kann die *Transparenz* eingesetzt werden, wenn sich der Benutzer zum Beispiel in der Drittpersonsicht befindet und den Avatar ein Gebäude betreten lässt. Wenn die Kamera in derselben Position bleibt, da keine besondere *Kamerasteuerung* in der Anwendung integriert ist, so ist der Benutzer blind für das, was mit dem Avatar in dem Haus geschieht. Statt dieses Problem mit Hilfe der *Kamerasteuerung* zu lösen, kann auch das Dach nach Betreten des Hauses transparent dargestellt werden und der Benutzer sieht den Innenraum, trotz dass sich die Kamera im selben Abstand zu dem Avatar befindet wie vorher. Dieses Problem ergibt sich in jedem Fall

sobald Kamera und Avatar durch einen festen Abstand getrennt sind und sich nicht im selben „Raum“ befinden. In diesem Fall kann *Transparenz* helfen, so dass der Benutzer seinen Avatar auch weiterhin in Gebäuden navigieren kann.

Das Umschalten einer opaken in eine transparente Wand geschieht dabei meist durch einen Auslöser den der Benutzer unbewusst betätigt. In dem erwähnten Beispiel entspricht dies dem Gehen durch die Tür.

Transparenzen können in der virtuellen Welt technisch gesehen auf alle Objekte angewandt werden, ganz gleich ob es der Realität entspricht oder nicht. Je nach Ziel der Anwendung macht dies natürlich nicht immer Sinn. Aussehen kann dies beispielsweise wie in Abbildung 3.29.

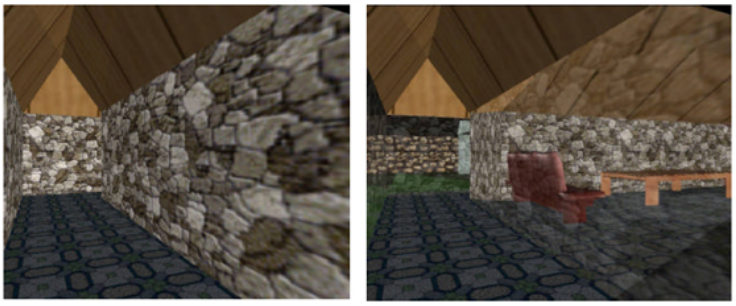


Abbildung 3.29.: Anwendungsbeispiel: *Transparenz*; Vergleich opake und transparente Wand; Abbildung nach: [CS01]

4. Testszenario

Wird eine Anwendung neu entwickelt, so existiert sie zunächst nur als Idee. Sie beschreibt das Ziel bzw. die Aufgabe, die die Anwendung später erfüllen soll. Diese Aufgabe wird wiederum in kleine Teilaufgaben zerlegt, die betrachtet und einzeln auf verschiedene Weisen umgesetzt werden können, um hinterher wieder zu einem ganzen System zusammengefügt zu werden. Zu jeder Idee gibt es entsprechend viele Umsetzungsmöglichkeiten, je nachdem wie komplex sie ist und wie viele Teilaufgaben sie beinhaltet.

In virtuellen Anwendungen ist die Navigation eine Teilaufgabe der eigentlichen Aufgabe der Anwendung. Sie dient einem bestimmten Ziel und kann auf verschiedene Art und Weise integriert werden. Durch die Betrachtung bestehender Navigationsmöglichkeiten lassen sich bereits existierende Stärken und Schwächen in Hinsicht auf unterschiedliche Aufgaben darlegen. Die Aufgaben, die eine Navigation in einer virtuellen Welt übernehmen kann, sollen im Folgenden in vier Kategorien eingeteilt werden. Somit können die Metaphern entsprechend den verschiedenen Aufgaben auf ihre Zweckmäßigkeit untersucht werden. Die Aufgabeneinteilung dient als Grundlage der Beurteilungen der Metaphern (Kapitel 6).

Die Aufgabenkategorien:

1. **Explore-Aufgaben:** Orientierung in einer unbekannten Szene
2. **Search-Aufgaben:** Suche nach Objekten in einer bekannten Szene
3. **Go-back-to-known-target-Aufgaben:** Zurückkehren an einen bereits bekannten Ort
4. **Inspect-Aufgaben:** Inspizieren eines Objektes

Um die Navigationsmöglichkeiten in Hinsicht dieser Aufgaben zu untersuchen, bedarf es eines Testszenarios. Dieses Szenario kann sehr umfangreich sein und wird in Abschnitt 4.1 vorgestellt. Bestimmte Aspekte aus diesem Szenario werden in Abschnitt 4.2 herausgegriffen und in dieser Arbeit

genauer betrachtet. Abschließend wird in Abschnitt 4.3 deutlich herausgestellt, welche funktionalen und nichtfunktionalen Anforderungen an die Software, die für diese Arbeit als Testumgebung entwickelt wird, gestellt werden, um eine geeignete Beurteilung zu ermöglichen.

4.1. Analyse der Möglichkeiten

Alle existierenden Navigationsmöglichkeiten in Kombination mit allen Benutzerperspektiven in Verbindung mit allen navigationsunterstützenden Verfahren und dies wiederum in jeder möglichen Kombination für jede der vier Aufgaben zu testen und dies in einer einzigen Anwendung ist nahezu, wenn nicht sogar überhaupt, unmöglich.

Im folgenden Abschnitt wird versucht ein Gesamtszenario zu beschrieben, welches es ermöglicht, nicht alle existierenden, aber zumindest die vorgestellten Verfahren zusammenzubringen, um sie untereinander vergleichbar testen zu können. Dies umfasst somit nicht die Kombination aller denkbaren Möglichkeiten, sondern es soll einen Gesamteindruck über die Kombinierbarkeit der einzelnen Aspekte vermitteln.

Eine kurze Zusammenfassung der Komponenten die betrachtet werden und in Kapitel 2 und 3 ausführlich vorgestellt wurden, macht das Zusammenspiel deutlicher. Dazu werden sie im Folgenden kurz in Erinnerung gerufen.

- **Die zu erfüllenden Aufgaben:**
  Explore-Aufgaben, Search-Aufgaben, Go-back-to-known-target-Aufgaben, Inspect-Aufgaben

- **Die Benutzerperspektiven:**
  Egoperspektive, Drittpersonansicht

- **Die Navigationsmöglichkeiten:**
  Laufen-Metapher, Pfad-Metaphern (vollautomatischer, halbautomatischer, mit Hilfe einer Führungsperson, Inspektionspfad), Teleport-Metapher, Ephemeral World Compression-Metapher, Speed couples flying-Metapher, Inverse Fog and Inverse Scaling-Metaphern, Posession-Metapher, Object Manipulation and Ghost Copy-Metapher

- **Die navigationsunterstützenden Verfahren:**
  Aktive Verfahren: Kamerasteuerung, Bewegungssteuerung, Kollisionserkennung

Passive Verfahren: Grafische Abstraktion, Elision-Technik, Schattendarstellung, Gitterdarstellung, Koordinatendarstellung, World in miniatur-Metapher, Multiple Fenster, Übersichtskarte, Besondere Hervorhebungen, Markierungen, Transparenzen

Aus diesen Komponenten lässt sich ein theoretisches Gesamtszenario zusammensetzen, welches alle Kombinationen dieser Komponenten beinhaltet. Hierzu bietet es sich zunächst an die Aufgabenstellungen in eine sinnvolle Reihenfolge zu bringen, so dass diese in einem Testszenario aufeinander folgend mit einer Versuchsperson getestet werden können.

Bei Betreten einer virtuellen Welt stellt sich dem Anwender zwangsläufig als erstes das Problem sich in der neuen Umgebung orientieren zu müssen. Demnach bietet es sich an die *Explore*-Aufgabe zu Anfang zu betrachten. Der Benutzer soll dabei versuchen sich in der unbekannten Szene zu orientieren und Zusammenhänge zwischen Objekten zu erkennen.

Nachdem er sich durch die Welt bewegt hat (oder noch währenddessen), bekommt er in einer virtuellen Anwendung eine Aufgabe. Diese kann bereits seit Starten der Anwendung klar sein, wie bei einer Montage oder sie wird im Laufe der Anwendung deutlich. Dabei kann sie wie in Adventures in den Spielverlauf integriert werden. Die meisten beinhalten dabei den Aspekt einer Suche. Sei es die Suche nach einem Gegenstand, einem Gegner oder nach einem bestimmten Ort. Aus diesem Grund soll die *Search*-Aufgabe der zweite Teil des Versuchs sein.

Hat der Benutzer sich einmal orientiert und bereits verschiedene Orte oder Gegenstände gefunden, so kann ihm nun eine *Go-back-to-known-target*-Aufgabe gestellt werden. Dabei soll er an einen speziellen Ort gelangen, den er bereits kennt.

Zuletzt bleibt noch die *Inspect*-Aufgabe. In dieser soll sich der Anwender ein spezielles Objekt genauer anschauen. Dieses könnte sich an dem Ort befinden, an den er durch die *Go-back-to-known-target*-Aufgabe zurückgekehrt ist. Auf diese Weise können die Aufgabentypen miteinander verbunden werden.

Somit kann ein Szenario ohne Unterbrechung geschaffen werden, welches eine intuitive Abfolge an zu erfüllenden Aufgaben darstellt und bei den Benutzertests für eine angenehme und nicht verwirrende Situation für die Testperson sorgt.

Für diese Aufgabentypen muss es entsprechende Aufgaben geben, die

die anderen Komponenten[1] mit einbinden. Dabei kann die Benutzerperspektive so integriert werden, dass diese dem Benutzer zum Umschalten freigegeben wird. Das bedeutet die Versuchsperson kann in jeder Situation die Egoperspektive oder die Drittpersonansicht benutzen und sich für diejenige entscheiden, die sie als sinnvoller für die gestellte Aufgabe erachtet. So kann ein direkter Vergleich beider Perspektiven von jeder einzelnen Testperson stattfinden. Die Auseinandersetzung mit beiden Perspektiven liefert dabei eine bessere Bewertung.

Um die Navigationsmöglichkeiten entsprechend den Aufgabentypen zu betrachten, muss die Versuchsperson zum Vergleichen der Navigationsmetapher, jede Aufgabe in jeder möglichen Metapher wiederholen.

Die *Explore*-Aufgabe verlangt von ihr sich durch die Welt zu bewegen und sie zu erkunden. Dieses muss sie in den folgenden Metaphern wiederholen: *Laufen*-Metapher, *Pfad*-Metaphern (vollautomatischer, halbautomatischer, mit Hilfe einer Führungsperson), *Teleport*-Metapher, *Ephemeral World Compression*-Metapher, *Speed couples flying*-Metapher, *Inverse Fog and Inverse Scaling*-Metapher und *Posession*-Metapher. (An dieser Stelle ist eine Betrachtung des *Inspektionspfades* und der *Object Manipulation and Ghost Copy*-Metapher nicht notwendig, da diese Verfahren zu einer Orientierung nicht beitragen können.) Da sich der Benutzer jedoch bereits nach ca. drei Rundgängen orientiert hat, kann er diesen Test nicht leisten. Aus diesem Grund werden mehr Probanden benötigt. Diese testen jeweils zwei bis drei verschiedene Metaphern hinsichtlich der Orientierung und sollen anschließend eine vergleichende Wertung abgeben.

Für die *Search*-Aufgaben können, wie für die *Explore*-Aufgaben, alle Metaphern bis auf den *Inspektionspfad* und die *Object Manipulation and Ghost Copy*-Möglichkeit verglichen werden. Dazu kann jede Testperson jede Metapher betrachten, indem die Szene so viele Objekte beinhaltet, wie Navigationsmöglichkeiten getestet werden sollen. Sobald die Versuchsperson ein Objekt gefunden hat, wird eine andere Navigationsmöglichkeit eingestellt, bis alle Navigationsmetaphern getestet wurden.

Dieses Vorgehen des Aufbaus entspricht auch dem für die *Go-back-to-known-target*-Aufgabe. Wenn genug Orte vorhanden sind, so kann der Benutzer jeweils mit einer Metapher zu einem Ort zurückkehren und diese nun vergleichend benutzen.

Für die *Inspect*-Aufgabe sieht die Testmöglichkeit etwas anders aus. Nicht alle Metaphern können hierzu eingesetzt werden. Grundsätzlich

[1] Benutzerperspektive, Navigationsmetaphern, navigationsunterstützende Verfahren

kann ein und dasselbe Objekt als Anschauung für alle Metaphern dienen.

Vorzuziehen ist jedoch die Variante ein großes und ein kleines Objekt zu betrachten, da die Metaphern bezüglich der Objektgröße verschiedene Vorteile aufweisen. So kann für die Betrachtung von Objekten die *Laufen*-Metapher, die *Pfad*-Metaphern (vollautomatischer, halbautomatischer, mit Hilfe einer Führungsperson, Inspektionspfad), die *Teleport*-Metapher und die *Object Manipulation and Ghost Copy*-Möglichkeit in Betracht gezogen werden. Mit Hilfe der *Ephemeral World Compression*-Metapher, der *Speed couples flying*-Metapher, der *Inverse Fog and Inverse Scaling*-Metapher kann sich der Avatar ausschließlich fortbewegen und nicht Dinge betrachten, weshalb diese hier bereits ausgeschlossen werden. Die *Posession*-Metapher dient dem Perspektivenwechsel und kann an dieser Stelle auch zu keinem sinnvollen Ergebnis führen.

Mit den bisherigen Überlegungen hängt die Testpersonenzahl von der Anzahl der für die Orientierung verwendeten Metaphern ab. Testet jede Versuchsperson nur eine Metapher entspricht dies genau der Anzahl der ausgewählten Metaphern. Bei der Vorgehensweise, dass jede Testperson mehrere Metaphern nutzen soll, um diese vergleichen zu können, vergrößert sich auch die Versuchspersonenanzahl entsprechend.

Von der Annahme ausgehend, dass maximal drei Durchgänge durch die Welt noch als Orientierungsphase bezeichnet werden kann, ist eine Zurückführung auf ein einfaches stochastisches Problem möglich. Dabei ist $\binom{n}{k}$ „die Anzahl der Möglichkeiten, aus einer Menge mit n Elementen k Elemente auszuwählen, wobei die Reihenfolge der ausgewählten Elemente nicht berücksichtigt wird“ [Bin07].

Bei Anwendung des Binomialkoeffizienten auf das dargestellte Problem ergibt sich daraus $\binom{10}{3}$. Somit existieren 120 Kombinationsmöglichkeiten, weshalb auch 120 Versuchspersonen benötigt werden. Für alle anderen Aufgabentypen kann jede Testperson Versuche mit allen Metaphern durchführen, somit werden keine zusätzlichen Testpersonen angesetzt. Die 120 Versuchspersonen stellen allerdings nur einen einzigen Testlauf dar. Aus diesem Grund müssen diese zusätzlich wiederholt werden, was die Anzahl der Testpersonen vervielfacht.

Unabhängig von der bisherigen Zahl der Testpersonen wurden die navigationsunterstützenden Verfahren noch nicht betrachtet. Hierzu gehört beispielsweise die *Übersichtskarte*, welche die Anzahl der Testpersonen wiederum verdoppeln würde, da eine Versuchsperson, die bereits eine Karte der Welt gesehen hat dieselben Aufgaben nicht ohne Darstellung

einer Karte wiederholen kann, ohne einen „unfairen“ Vorteil gegenüber anderen Testpersonen zu haben. Somit müssen alle Testpersonen in zwei Gruppen unterteilt werden. Die eine Gruppe erhält eine Karte, die andere nicht.

Viele weitere Verfahren wie zum Beispiel die *Schattendarstellung* können jedoch einfacher integriert werden und brauchen keine zusätzlichen Testgruppen. Bei bisheriger Betrachtung wird jedoch klar, dass ein solcher Benutzertest im Rahmen dieser Arbeit unmöglich umgesetzt werden kann.

4.2. Auswahl

Die Betrachtung der möglichen Kombinationen von

- Aufgabenstellung der Anwendung
- Benutzerperspektiven
- Navigationsmöglichkeiten
- navigationsunterstützenden Verfahren

aus dem Gesamtszenario aus 4.1 zeigt deutlich, dass in dieser Arbeit nicht jede Kombination zwischen den einzelnen Komponenten umgesetzt werden kann. Bei näherer Betrachtung sind aber auch nicht alle Verknüpfungen der einzelnen Techniken sinnvoll. Über einen Ausschluss nutzloser Verknüpfungen können bereits Einschränkungen für ein mögliches Testszenario vorgenommen werden.

Dabei bleiben die Aufgabentypen und die Reihenfolge des Gesamtszenarios erhalten. Die Benutzerperspektive wird dem Benutzer zur Auswahl gestellt und kann jederzeit umgeschaltet werden. Dies bietet so die Möglichkeit, die Perspektiven in jeder Aufgabe und an jedem Ort in der Szene zu testen.

Die navigationsunterstützenden Verfahren werden eingeschränkt eingebaut, da nur solche betrachtet werden sollen, die für Gebäudekomplexe eine Rolle spielen - wie bereits durch die Aufgabenstellung festgelegt wurde.

Eingeschränkt wird in dieser Arbeit aber vor allem auch die Anzahl der Navigationsmöglichkeiten, um die Testpersonenzahl weiter zu verringern. Um eine sinnvolle Auswahl zu treffen werden, die Navigationsmöglichkeiten sowohl in Hinsicht auf die vier Aufgabentypen, als auch in Hinsicht auf die Szenerie der virtuellen Welt, die einen Gebäudekomplex darstellen soll,

betrachtet. Unabhängig von diesen folgt in Abschnitt 4.2.2 die Darlegung der navigationsunterstützenden Verfahren.

4.2.1. Navigationsmetaphern:

Die Betrachtung der Navigationsmetaphern erfolgt anhand der vier festgelegten Aufgaben, da nicht jede Metapher für jede Aufgabe gleich zu bewerten ist.

Explore-Aufgabe:
Für die *Explore*-Aufgabe innerhalb eines Gebäudekomplexes soll die *Laufen*-Metapher betrachtet werden, da sie am meisten Bezug zu einer Navigation in der Realität aufweist. Zusätzlich ist die Betrachtung einer *Pfad*-Metapher sinnvoll, die den Benutzer durch die komplette Szene führt und bei einem Gebäudekomplex speziell dabei helfen kann auch an Orte zu gelangen, die sonst eventuell übersehen werden könnten. Dafür soll der vollautomatische Pfad integriert werden, da der halbautomatische Pfad grafische Abgrenzungen benötigen würde, die innerhalb eines Gebäudekomplexes nicht zwingend gegeben sind und nicht künstlich oder unpassend in der Szene dargestellt werden sollen. Da kein weiterer Avatar integriert werden soll, wird auch die Möglichkeit einer Führungsperson ausgeschlossen.

Die *Teleport*-Metapher ermöglicht eine schnellere Art der Fortbewegung und eine andere Art der Eingabemöglichkeit. Statt über die Pfeil-Tasten zu navigieren, kann der Zielpunkt durch die Maussteuerung direkt in der dreidimensionalen Szene gewählt werden.

Ausgeschlossen wird die *Ephemeral World Compression*-Metapher, die *Speed couples flying*-Metapher und die *Inverse Fog and Inverse Scaling*-Metaphern, da sie vor allem geeignet sind um weite Strecken zurück zu legen, was bei einem Gebäudekomplex jedoch nicht notwendigerweise gegeben ist. Dies gilt auch für die *Posession*-Metapher, welche für den Benutzer zu unintuitiv wäre, da er in dieser Metapher in die Rolle anderer Objekte schlüpfen würde. Wenn eine virtuelle Welt jedoch ausschließlich aus Häusern besteht, wäre es merkwürdig die Welt aus Sicht eines Hauses zu sehen.

Nach dieser Betrachtung steht die Auswahl der Navigationsmetaphern für die *Explore*-Aufgaben fest. Sie wurden somit auf drei Navigationsmöglichkeiten (*Laufen-*, *Teleport-*, *Pfad*-Metapher) beschränkt, die jeder Benutzer testen kann, wodurch es nur noch einer statt 120 Personen bedarf.

Search-Aufgabe:
Für die *Search*-Aufgabe ist es wichtig, dass der Benutzer seine Bewegungsfreiheit hat, um sich alles anschauen zu können. Aus diesem Grund wird die *Laufen*-Metapher auch hier betrachtet. Da der Benutzer durch die *Pfad*-Metaphern seine Freiheiten verliert und ein Führer bei der Suche nach Objekten die Aufgabenstellung zerstören würde, werden diese Möglichkeiten für die *Search*-Aufgabe nicht weiter berücksichtigt. Die *Teleport*-Metapher kann zusätzlich ausgeschlossen werden, da durch das „Springen“ eine zu große Möglichkeit besteht Objekte zu übersehen. Dies ergibt bereits ohne Tests die *Laufen*-Metapher als die bevorzugte Wahl. Alle anderen Navigationsmetaphern lassen sich bereits als die schlechtere Wahl bewerten und müssen hier nicht nochmals getestet werden. Damit bietet diese Aufgabe den Raum, sowohl verschiedene navigationsunterstützende Verfahren, als auch die Benutzerperspektiven genauer zu untersuchen.

Go-back-to-known-target-Aufgabe:
Nachdem die Welt erkundet ist, soll der Benutzer an einen Ort oder zu einem Gegenstand zurückkehren. Diese *Go-back-to-known-target*-Aufgabe legt nahe, nicht nur die *Laufen*-Metapher, sondern auch die *Teleport*-Metapher zu integrieren, denn durch das Teleportieren gelangt der Benutzer sehr schnell an andere Orte. Dies hilft ihm an einen speziellen Ort zu gelangen, wenn er nicht mehr auf die Umgebung achten muss, weil er die Welt schon kennt. Ob dies in diesem Kontext effizienter ist als die *Laufen*-Metapher soll anhand der Benutzertests in Kapitel 6 geklärt werden. Eine *Pfad*-Metapher zu integrieren erfüllt für diese Aufgabe keinen Zweck, da das Ziel des Benutzers nicht vordefiniert ist und dazu ein Pfad dynamisch erstellt werden müsste. Die Betrachtung der dynamischen Erstellung eines Pfades, der *Ephemeral World Compression*-Metapher und der *Speed couples flying*-Metapher wären in diesem Kontext sicher interessant, können aber auf Grund der kurzen Entwicklungszeit nicht betrachtet werden. Mit Hilfe der *Inverse Fog and Inverse Scaling*-Metapher lässt sich dagegen ausschließlich eine Orientierung gewinnen, erleichtert dem Benutzer allerdings keine schnellere Navigation zu bekannten Orten und hat für diese Aufgabe darum keine Bewandtnis.

Inspect-Aufgabe:
Für die *Inspect*-Aufgabe dienen einige Metaphern auf den ersten Blick der besseren Aufgabenbewältigung und werden im Folgenden aus diesem Grund bevorzugt integriert. Hierzu gehört die *Laufen*-Metapher und der

Inspektionspfad. Mit diesen beiden lassen sich Objekte von allen Seiten betrachten. Ein „Springen“ um das Objekt, wie es bei der *Teleport*-Metapher der Fall ist, kann schon bei Vergleich mit dem Laufen als die schlechtere Wahl ausgeschlossen werden, da hierdurch die Orientierung verloren gehen kann. Die *Object Manipulation and Ghost Copy*-Metapher ist zu den beiden ausgewählten Möglichkeiten noch eine interessante Betrachtung, soll aber auch ausgeschlossen werden, da es nur indirekt eine Navigationart ist, denn der Benutzer interagiert eigentlich mit dem Objekt. Die Navigation selbst wird letztendlich nur durch eine automatische Kamerasteuerung übernommen.

4.2.2. Navigationsunterstützende Verfahren:

Die navigationsunterstützenden Verfahren können während allen vier Aufgabentypen angewandt werden und müssen somit nicht speziell den einzelnen Aufgaben zugeteilt werden. Aus den vorgestellten Verfahren werden einige aktive und passive ausgewählt, die für die Betrachtung eines Gebäudekomplexes am sinnvollsten erscheinen.

Aktive Verfahren:
Die vorgestellte Möglichkeit der *Kamerasteuerung* wird nicht integriert, da die passive Methode (*Transparenz*) die hier erforderliche Aufgabe erfüllt (siehe auch **Passive Verfahren: Transparenz**).

Eine weitere vorgestellte aktive Methode ist die *Bewegungssteuerung*. Sie wird an dieser Stelle vernachlässigt, da ausschließlich ebener Boden betrachtet wird und demnach keine Treppensteigung und ähnliches simuliert werden muss.

Die letzte vorgestellte aktive Möglichkeit ist die *Kollisionserkennung*. Sie ist essentiell für das Betreten und Verlassen von Gebäuden und wird entsprechend integriert.

Passive Verfahren:
Für den Benutzer muss es eine Unterstützung geben, wenn er in der Drittpersonansicht den Avatar in ein Gebäude hineingehen lässt. Für eine Unterstützung an dieser Stelle wurden eine aktive und eine passive Methode vorgestellt. Mit Hilfe beider Möglichkeiten kann der Benutzer seinen Avatar weiterhin gezielt navigieren. Die aktive Methode besteht darin die Kamera hinab zu dem Avatar gleiten zu lassen und die Perspektive somit zu verändern. Die passive Methode beschreibt die Möglichkeit das Haus oder Teile von diesem, wie z. B. das Dach, transparent werden zu lassen.

Für die entwickelte Testumgebung wird an dieser Stelle die passive Methode, also die *Transparenz*, getestet. Diese Hilfe dient als Zusatz zu den beiden betrachteten Perspektiven und soll nicht nur für diesen Zweck eine Weitere einführen.

An passiven Verfahren wird zusätzlich die *Schattendarstellung* integriert, da die Anwendung somit wesentlich realistischer wirkt und auch aus diesem Grund in den gängigen Anwendungen zu finden ist.

Eine Möglichkeit die virtuelle Welt als Überblick zu sehen, sollte der Orientierung nützlich sein und wird deshalb in Form einer *Übersichtskarte* integriert (und nicht als *World in miniatur*-Metapher, da keine Manipulation in der virtuellen Welt erforderlich ist). Die Karte soll dabei auf zwei verschiedene Arten umgesetzt werden. An dieser Stelle soll die Darstellungsmöglichkeit einer statischen mit einer sich mit den Bewegungen des Benutzers drehenden Karte vergleichen werden. Die *Übersichtskarte* wird dabei in der GUI integriert, um jederzeit sichtbar zu sein. Dazu benötigt es die Möglichkeit der *multiplen Fenster*, um den Blick in die virtuelle Welt gleichzeitig nicht zu versperren.

Markierungen in der Welt gibt es zudem auch in Form der statischen *Landmarks*, die voreingestellt sind. Diese werden durch große oder auffällige Gebäude repräsentiert.

Nicht integriert werden alle anderen vorgestellten Verfahren. Somit hat der Benutzer in der Testumgebung keine Möglichkeit zusätzlich eigene *Markierungen* zu setzen. Auch *besondere Hervorhebungen* gibt es nicht, da auf keine speziellen Objekte hingewiesen werden soll mit denen es Interaktion geben könnte. Eine *Gitterdarstellung* würde in einer solchen Szenerie nicht in das Gesamtszenario passen und wird entsprechend nicht dargestellt. Die *grafische Abstraktion* bringt durch die Einfachheit der Szene weder Vorteil der Rechnerperformanz noch einen besseren Überblick, da nur wenige Objekte in der Szene vorhanden sind und keine Reizüberflutung zu erwarten ist. Dies gilt auch für die *Elision*-Technik. Da die Szene ziemlich klein ist, bringen *Kompass* und *Koordinaten* keinen ersichtlichen Vorteil für den Benutzer, da er die komplette Szene nach einem kurzen Gang bereits überblicken kann.

Somit soll das zu betrachtende Szenario folgende Elemente enthalten:

- **Navigationsmöglichkeiten:**
  - **Explore-Aufgaben:**

    Laufen-Metapher, vollautomatischer Pfad, Teleport-Metapher

  * **Search-Aufgaben:**
    Laufen-Metapher
  * **Go-back-to-known-target-Aufgaben:**
    Laufen-Metapher, Teleport-Metapher
  * **Inspect-Aufgaben:**
    Laufen-Metapher, Inspektionspfad

- **Die Benutzerperspektiven:**
  Egoperspektive, Drittpersonansicht
  (durch Umschaltmöglichkeit für den Benutzer)
- **Die navigationsunterstützenden Verfahren:**
  Aktive Verfahren: Kollisionserkennung
  Passive Verfahren: Transparenzen, Multiple Fenster,
  Übersichtskarte (statisch und drehend), Markierungen,
  Schattendarstellung

4.3. Anforderungen

Um die Szenerie mit all den in Kapitel 4.2 ausgewählten Komponenten als Testumgebung zusammenzusetzen, bedarf es einiger Anforderungen an das System und an alle sonstigen mit einfließenden Komponenten. Diese werden dazu in funktionale und nichtfunktionale Anforderungen unterteilt.

Funktionale Anforderungen:

1. **Anwendungsfälle:**
   - Der Benutzer soll den Avatar in der virtuellen Szene, die als Testumgebung dient, navigieren können.
   - Die verschiedenen Navigationsmöglichkeiten sowie die Perspektive müssen zur Laufzeit umgestellt werden können.

2. **Schnittstellen:**

   - Durch die Standard-Eingabegeräte Tastatur und Maus kann der Benutzer mit dem System kommunizieren.
   - Das Zusammenarbeiten mit einem geeigneten *Renderer* sowie einer geeigneten *Physik-Engine* muss möglich sein. (Da die Anwendung, die im Zusammenhang mit dieser Arbeit entwickelt wird, ausschließlich eine Testumgebung darstellt, ist der Anspruch nicht so hoch wie bei aktuellen Computerspielen und erfordert nicht solch enorm grafische Leistung. Demnach wird dies sowohl von dem *Renderer* als auch der *Physik-Engine* nicht in dem Maße erwartet).
   - Dabei soll der *Renderer* folgende Anforderungen erfüllen:
     - Der *Renderer* soll *open source* sein, um somit für die optimale Einbindung eigener Entwicklungen zu dienen.
     - Es soll eine gute Dokumentation zu dem *Renderer* vorhanden sein, um eine schnelle Einarbeitung in den fremden Code zu ermöglichen.
     - Die Code-Syntax soll einfach zu handhaben sein, da das Primärziel der Testumgebung das Testen der Navigation und nicht die Darstellung einer aufwendigen Szene ist.
     - Der *Renderer* soll aktuell (nicht „out-of-date“) sein, so dass Support und Ansprechpartner vorhanden sind.
     - Der *Renderer* muss *Microsoft Windows* tauglich sein, da mit einer bereits bekannten Entwicklungsumgebung gearbeitet wird.
     - *OpenGL* Unterstützung soll gewährleistet sein, da es hierfür bereits eine breite Akzeptanz gibt.
     - Es muss die Möglichkeit geben GUI-Elemente durch den *Renderer* zu unterstützen, um verschiedene Informationen benutzerfreundlich darzustellen.
     - Die Möglichkeit eigene Objekte (Texturen und Meshes) aus Maya zu exportieren und rendern zu können muss von dem *Renderer* unterstützt werden, um dank persönlicher Erfahrung mit Maya eine schnellere Entwicklung einer Szenerie zu fördern.

  - Die Kamerasteuerung muss flexibel sein, um verschiedene Metaphern integrieren zu können.
  - Lichtquellen müssen eingefügt und Schatten muss dargestellt werden können.
  - Eine Himmeldarstellung (Skybox o. ä.) muss integrierbar sein, um die Szene zu vervollständigen und den Benutzer nicht ins „Leere“ schauen zu lassen.
  - Eine Szenenbegrenzung muss möglich sein, damit die Szene nicht unendlich groß und auch nicht abgeschnitten werden muss.
  - Eine einfache Szenenhierarchie muss erstellbar sein, um abhängige Bewegungen einfacher integrieren zu können.
- Die *Physik-Engine* soll folgende Anforderungen erfüllen:
  - Zusammenarbeit mit dem *Renderer* muss möglich sein.
  - Das Erkennen der Objekte in der Szene als Physik-Objekte muss möglich sein.
  - Eine Kollisionserkennung zwischen Physik-Objekten muss möglich sein.
  - Es soll eine gute Dokumentation existieren.

3. **Dimensionierung:**
   - Es kann zu jeder Zeit nur ein Benutzer die Anwendung bedienen und somit den Test durchführen.

Nichtfunktionale Anforderungen:

1. **Zuverlässigkeit:**
   Das System sollte während den Tests kein unerwartetes Verhalten aufweisen. Im Notfall soll das System mit Hilfe eines Neustarts auf den Anfangszustand zurückgesetzt werden können.
2. **Aussehen und Handhabung:**
   Das System soll in einem Fenster eine virtuelle Szene darstellen. Zusätzlich soll eine grafische Bedienoberfläche vorhanden sein, in die eine Übersichtskarte der Szene integriert ist, sowie dem Benutzer Hilfestellungen per Text anzeigt.

3. **Benutzbarkeit:**
   Das System soll mit Hilfe einer kurzen Einführung leicht verständlich sein und keine speziell zu erlernenden Funktionen bieten.

4. **Leistung und Effizienz:**
   Das System muss sofort auf die Eingaben des Benutzers reagieren.

5. **Betrieb und Umgebungsbedingungen:**
   Das System wird unter der Aufsicht des Entwicklers und der Bedienung einer Testperson als Testumgebung benutzt. Dabei werden Maus und Tastatur als Eingabegeräte verwendet.

6. **Performanz:**
   Eine dreidimensionale Szene mit wenigen und nicht komplexen Gebäuden soll flüssig für den Benutzer in der Bedienung sein.

7. **Verfügbarkeit:**
   Das zu entwickelnde System ist ausschließlich als Testumgebung im Rahmen dieser Arbeit gedacht.

8. **Skalierbarkeit:**
   Eine weitere Anpassung an andere Gegebenheiten muss nicht berücksichtigt werden, da das System ausschließlich im Rahmen dieser Arbeit betrachtet wird.

9. **Erweiterbarkeit:**
   Die Möglichkeit weitere Navigations- oder Eingabemöglichkeiten hinzuzufügen soll gegeben sein.

10. **Wartbarkeit:**
    Das System muss nach Abschluss der Tests nicht weiter gewartet werden und wird demnach speziell für die Tests ausgelegt.

5. Technische Umsetzung

Für die Umsetzung des Szenarios, wie es im vorangegangenen Kapitel beschrieben wurde, werden verschiedene Werkzeuge (siehe Abschnitt 5.1) benötigt. Darunter finden sich Werkzeuge zur Erstellung der virtuellen Umgebung und dem Anzeigen der erstellten Umgebung (dem Rendering). Zusätzlich spielen Werkzeuge für die Implementierung einer Kollisionserkennung eine Rolle, sowie der Einbindung der Eingabegeräte für die Mensch-Maschine-Kommunikation. Das Zusammenspiel dieser wird durch einen Blick auf die Softwarearchitektur (Abschnitt 5.2) deutlich. Nach Betrachtung der Struktur des Systems wird die Implementierung der einzelnen Bausteine (Benutzerperspektive (Abschnitt 5.3), Navigationsmetaphern (Abschnitt 5.4) und navigationsunterstützende Verfahren (Abschnitt 5.5)) für die beschriebene Testumgebung beschrieben. Diese werden letztendlich zu einem Gesamtsystem integriert (Abschnitt 5.6). Somit liegt die Hauptaufgabe dieses Kapitels in der Beschreibung der Implementierung und der Integration, so dass am Ende ein in sich geschlossenes System dargestellt ist, das als Testumgebung für die in Kapitel 6 dokumentierten Benutzertests dient.

5.1. Auswahl der Werkzeuge

Im Folgenden werden die verschiedenen Werkzeuge betrachtet, mit dessen Hilfe die Testumgebung entstanden ist. Zu diesen gehören natürlich auch die genutzte Programmiersprache *C++* und die Entwicklungsumgebung *Microsoft Visual Studio 2005*. Da diese hier jedoch keine weitere Erklärung benötigen, sollen sie an dieser Stelle nur genannt werden. Alle weiteren Werkzeuge inklusive ihrer Nutzung in der Entwicklung werden genauer beschrieben. Dazu gehört das Tool für die Modellierung der Szene (Abschnitt 5.1.1), die *Grafik-Engine* für die Darstellung der Szene (5.1.2), die *Physik-Engine* für die Kollisionserkennung (5.1.3) und die Eingabegeräte, um dem Benutzer eine Kommunikation mit der Anwendung zu ermöglichen (5.1.4).

5.1.1. Modellierung

In dieser Arbeit stellt das Vergleichen der verschiedenen Navigationsmöglichkeiten die Hauptaufgabe dar. Aus diesem Grund reicht eine einfache virtuelle Umgebung, um in ihr zu navigieren. Sie erhebt keinen Anspruch darauf besonders schön, vielfältig oder in irgendeiner Weise anspruchsvoll gestaltet zu sein. Durch die Nutzung der vorhandenen Vorkenntnisse mit *Alias Maya 6.0* war es möglich eine Umgebung schnell entstehen zu lassen, darum fiel die Wahl des Modellierungstools auf dieses Werkzeug.

Mit Hilfe von *Maya* entstanden verschiedene Objekte (siehe Abb. 5.1), die zu einer virtuellen Szene zusammengesetzt wurden und als Testumgebung für einen Gebäudekomplex dienen. Um die Szene klein zu halten, wurde eine Stadtmauer modelliert, die als Begrenzung dient. Um diese anschaulicher zu gestalten enthält sie außerdem einige Türme. Zusätzlich wurden verschiedene einfache Gebäude gestaltet. Sie besitzen teilweise Türen, um auch Szenarien innerhalb von Gebäuden testen zu können. Ergänzend zu den einfachen Häusern mit ausschließlich einem Raum existiert ein komplexeres Haus, dessen Inneres mehrere Zimmer, Türen und Fenster aufweist. Eine Kirche inmitten der anderen Gebäude soll speziell als *Landmark* dienen. Die Kirche und eine kleine Skulptur, die vor ihr steht, dienen zudem als Anschauungsobjekte für die *Inspect*-Aufgabe. Auch andere Aufgaben benötigen weitere spezielle Objekte; so wurde für die *Pfad*-Metapher ein kleiner Wagen mit Fähnchen modelliert, in dem der Avatar sitzen kann. Nach der Erstellung kleiner Boxen für die *Search*-Aufgabe, wurden diese als zu suchende Objekte aufeinander gestapelt in der virtuellen Szene verteilt. Insgesamt bilden die Häuser somit den benötigten Gebäudekomplex. Alle Objekte sind dabei einfarbig und ohne aufwendige Texturen gehalten.

Neben der virtuellen Umgebung benötigt der Benutzer einen Avatar als Darstellung seines Charakters in der Drittpersonansicht. Auch dieser wurde als kleine Spielfigur mit *Maya* erschaffen (siehe Abb. 5.1 (g)).

Alle Objekte zusammen ergeben die Testszene, die die Testpersonen durchlaufen und betrachten mussten (siehe Kapitel 6).

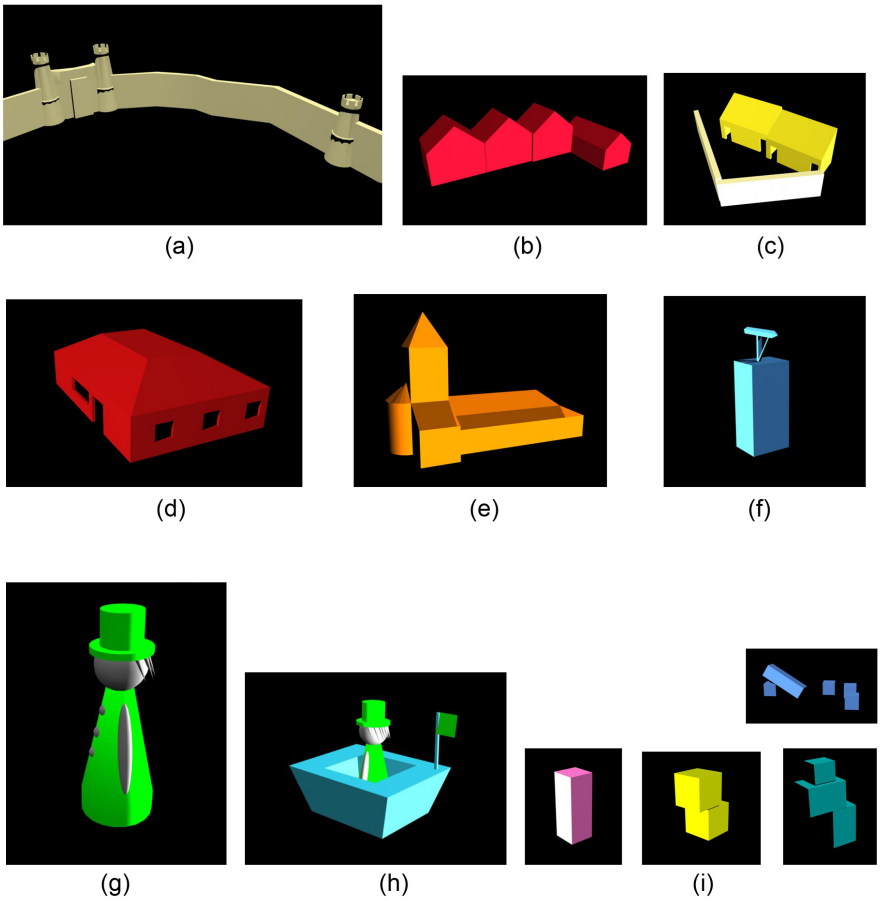


Abbildung 5.1.: Modelle der Szene: (a) Die Stadtmauer mit Türmen, (b) Beispiel für Häuser, (c) Beispiel für Kombination Haus mit Wand, (d) großes Haus, (e) Kirche, (f) Skulptur, (g) Spielfigur, (h) Pfad-Metapher Wagen, (i) Suchobjekte

5.1.2. Rendering

Das Zurückgreifen auf eine bereits existierende *Grafik-Engine*, zur Darstellung der modellierten Szene, ermöglicht es die in Abschnitt 2.2 erläuterten Vorteile nutzen zu können.

Die Wahl fiel hier auf *Ogre*[1]. *Ogre* ist ausschließlich eine *Grafik-Engine* und nicht etwa eine komplette *Game-Engine*. Somit entspricht sie einem Modul einer solchen und ist dabei alleinig für die Grafik zuständig. Entsprechend ist das Framework nicht so umfangreich wie bei einer ganzen *Game-Engine* und erschlägt den Entwickler nicht mit zuvielen Umsetzungsmöglichkeiten.

Für die Auswahl von *Ogre* waren dabei einige Punkte entscheidend. *Ogre* ist ein *Open Source* Projekt und wird ständig weiterentwickelt. Daher gibt es entsprechend viele Plugins, die beispielsweise auch eine Zusammenarbeit mit *Maya* vereinfachen. Zudem gibt es Einführungstutorials, Dokumentationen, Foren und ein Wiki, so dass die Einarbeitung schnell und einfach ermöglicht wird. Zusätzlich sind bereits Kenntnisse in der Universität vorhanden auf die zurückgegriffen werden konnte. Somit konnte schnell zur Hauptaufgabe dieser Studie, die Untersuchung von Navigationsmetaphern, übergegangen werden. *Ogre* ist zudem plattformunabhängig und unterstützt sowohl *OpenGL*- als auch *Direct3d*-Rendering. Verschiedene Techniken der Schattendarstellungen sind bereits integriert und können je nach gewünschter Darstellung eingesetzt werden. Außerdem arbeitet das System mit Szenenhierarchien und die Anbindung einer *Physik-Engine* ist möglich, welche für die Kollision benötigt wird (siehe 5.1.3). *Ogre* erfüllt somit all die Anforderungen, die in Abschnitt 4.3 aufgestellt wurden.

5.1.3. Kollisionserkennung

Die, für die Kollisionserkennung dieser Arbeit, benötigten physikalischen Berechnungen sind ein weiterer Teil einer *Game-Engine*. Dabei liegt es nahe die *Physik-Engine Open Dynamic Engine (ODE)* zu nutzen, da diese mit *Ogre* zusammenarbeitet und auch teilweise in das Framework von *Ogre* integriert ist.

Der große Vorteil, die Kollisionserkennung von *ODE* zu nutzen liegt darin, dass sogenannte *Kollisions-Objekte*, die für eine Kollision benötigt werden, aus speziellen *Meshes* und nicht nur aus einfachen geometrischen Formen erzeugt werden können. Somit kann zu jedem selbsterstellten *Mesh* ein zugehöriges Kollisions-Objekt generiert werden. Dieses wird

[1] http://www.ogre3d.org/, Stand: Nov. 2007

zusammen mit dem *Mesh*, das die grafische Darstellung des Objektes ist, an demselben Ort positioniert, ausgerichtet und anschließend nur noch über einen *body* gemeinsam bewegt. Somit können Kollisionen direkt gegen die exakten *Meshes* der einzelnen Objekte getestet werden. Aus diesem Grund gibt es beispielsweise kein Problem mit Türen. Da diese Öffnungen auch in dem Kollisions-Objekt vorhanden sind, kann der Avatar des Spielers durch diese hindurchgehen, ohne dass eine Kollision erkannt wird.

Bei einem Kollisionstest von Objekten über ihre *Bounding-Boxes* würde dies ein Problem darstellen, da die Öffnung innerhalb der Box liegt und hier zusätzlich ein anderer Weg gefunden werden müsste, die Öffnung als durchdringbar zu markieren. Mit Hilfe der Kollisions-Objekte kann dieses Problem einfach umgangen werden und es muss kein zusätzlicher Strahlentest selber implementiert werden.

Alle Objekte die kollidieren können sollen, müssen somit auch als Kollisions-Objekte erzeugt werden. Demnach müssen sowohl der Avatar als auch die Objekte als Kollisions-Objekte vorhanden sein, da der Avatar mit den einzelnen statischen Objekten kollidieren muss.

5.1.4. Eingabegeräte

Bereits in der Aufgabenstellung wurde festgelegt, dass zur Mensch-Maschine-Kommunikation Standard-Desktop-Eingabegeräte genutzt werden. Diese beinhalten vor allem Maus und Tastatur, die auch in dieser Arbeit eingesetzt werden. Dabei dient die Maus vor allem zur Steuerung des Mauscursors (wenn vorhanden) oder der Drehung des Avatars - je nach Navigationsmetapher. Für die Pfadimplementierung wurde zusätzlich das Scrollrad eingesetzt. Von der Tastatur wurden sowohl die Verwendung der üblichen Pfeil-Tasten, als auch die Verwendung der WASD-Tastenkombination zur Fortbewegung des Avatars ermöglicht.

Weitere Tastenfunktionen werden benötigt, um die Navigationsmetaphern umzuschalten, sowie die Übersichtskarte ein- oder auszublenden. Hierzu werden die F-Tasten benutzt, da sie nur von dem Entwickler und nicht durch die Versuchsperson während der Tests bedient werden sollten. Die Umschaltung der Perspektive dagegen wurde auf die Taste „p“ festgelegt, so dass sie für den Benutzer auch während der Anwendung jederzeit intuitiv zu finden ist. Um während der laufenden Anwendung Screenshots zu machen oder das Programm zu verlassen wurden zusätzliche Tastaturbelegungen integriert.

5.2. Software-Architektur

Das System wurde so integriert, dass die folgende Architektur entstand (siehe 5.2).

Für den Test bekommt der Benutzer eine Aufgabestellung, die er verarbeitet und über die Eingabegeräte seine Reaktion an das System übermittelt. Die Signale der Eingabegeräte werden teilweise direkt über die *Grafik-Engine* und teilweise über das eigene System abgefragt und weiterverarbeitet. Die *Physik-Engine* aktiviert sich sobald eine Kollision stattfindet und meldet diese an das eigene System, das wiederum die Kollisionsbehandlung übernimmt. Das Rendering wird während der gesamten Anwendung von der *Grafik-Engine* übernommen.

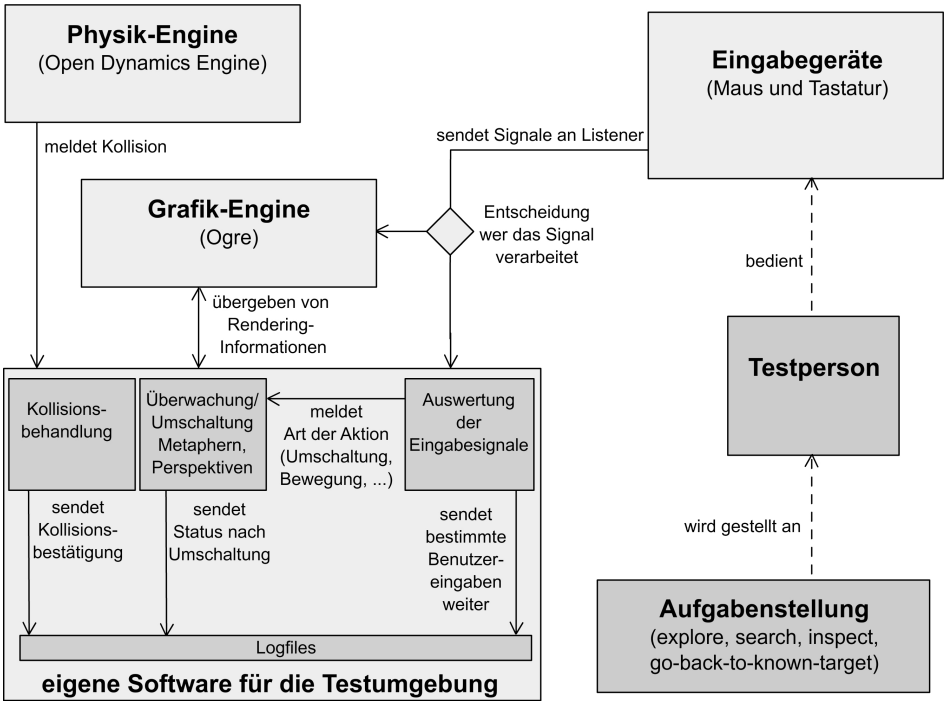


Abbildung 5.2.: Softwarearchitektur; Anhand der Grafik ist zu erkennen, wie die verschiedenen Bestandteile miteinander verknüpft sind. So arbeiten in dem letztendlich entwickelten Gesamtsystem sowohl Teile der *Grafik-Engine* (*Ogre*), sowie kleine Teile der *Physik-Engine* (*ODE*) zusammen mit der eigens entwickelten Software und den Eingabegeräten.

Die selbst entwickelte Software arbeitet dabei mit einem Szenengraphen (siehe Abb. 5.3). Dieser wird möglichst einfach aufgebaut, so dass nach und nach die einzelnen Navigationsmetaphern leicht integrierbar sind.

Dazu existiert ein Wurzelknoten (*root*), der so viele Kinderknoten beinhaltet wie modellierte Objekte in der Szene vorhanden sind (Abb. 5.3 gelbe Felder Ebene 1), so dass diese leicht einzeln angesprochen werden können. An jedem dieser Knoten ist eine Geometrie des entsprechenden Objektes angehangen (Abb. 5.3 Darstellung durch gelbes Geometriefeld vertretend für alle Geometrien dieser Ebene). Zudem existieren weitere Kinderknoten des *root*-Knotens, die spezielle Aufgaben erfüllen (Abb. 5.3 orange Felder Ebene 1). Der *PersonViewModel*-Knoten ist für die Bewegung des Avatars und der Kamera verantwortlich. Aus diesem Grund enthält er zwei weitere Knoten: Den *AvatarNode*-Knoten, an den die Geometrie des Avatars angehangen ist und den *CamNode*-Knoten. Um den Abstand zwischen Kamera und Avatar zu definieren, wurde der *CamNode*-Knoten transliert und rotiert. Der *CamNode*-Knoten hat noch einen Kinderknoten, den *PitchNode*. Über ihn kann die Rotation der Kamera gesteuert werden, ohne den Abstand zwischen Kamera und Avatar zu verändern. An diesem Knoten ist letztendlich die Kamera angehangen. Zu diesen Knoten gibt es noch einen Weiteren, der speziell für den Inspektionspfad notwendig ist. Dies ist der *OrbitNode*-Knoten. Da die Kamera bei dieser Metapher anders reagieren muss als bei den Übrigen, ist ein einfacheres Umschalten durch umhängen der Kamera zwischen *orbitCamNode*-Knoten und *PitchNode*-Knoten möglich. Weitere Erläuterungen zu diesem Verfahren in Abschnitt 5.4.2.

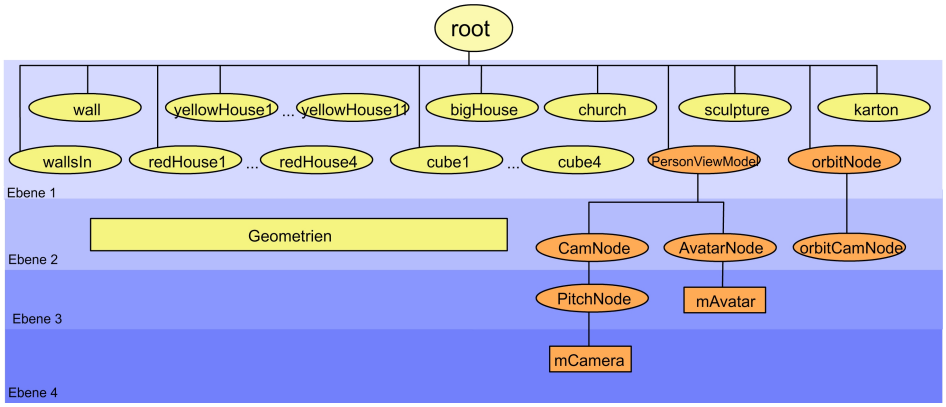


Abbildung 5.3.: Szenengraph: Darstellung der verschiedenen Hierarchieebenen der entwickelten Software

5.3. Benutzerperspektiven

Die Implementierung der Benutzerperspektiven kann durch die Taste „p“ umgeschaltet werden, unabhängig davon zu welchem Zeitpunkt und an welchem Ort sich der Benutzer gerade in der virtuellen Welt befindet. Die Umsetzung der beiden Benutzerperspektiven wird in den folgenden Abschnitten beschrieben.

Drittpersonansicht:
Um dem Benutzer eine Drittpersonansicht zu ermöglichen, wird eine virtuelle Figur benötigt. Diese muss sich immer im Blickfeld der Kamera befinden (siehe Abb. 5.4 (a)). Aus diesem Grund wird die Kamera-Position abhängig von der Avatar-Position schräg über ihr positioniert und bietet einen Blick sowohl auf den Avatar, als auch auf seine umliegende Umgebung. Wird der Avatar bewegt, so bewegt sich die Kamera mit ihm. Ihre Position wird dabei in einer festgelegten Entfernung und einem festgelegten Winkel zu dem Avatar berechnet.

Egoperspektive:
In der Egoperspektive wird kein Avatar dargestellt. Die Kamera wird im Gegensatz zu der Drittpersonansicht in eine Position, die dem Kopf des Avatars entspricht, gesetzt. So wird dem Benutzer der Blick aus Sicht einer Figur in der virtuellen Welt ermöglicht (siehe Abb. 5.4 (b)). Da die Kamera keine Geometrie besitzt, kann sie so ohne weiteres nicht mit anderen Objekten kollidieren. Dies ist aber auch in der Egoperspektive notwendig. Aus diesem Grund wird der Avatar nur ausgeblendet, ist aber eigentlich noch vorhanden und wird in die Kollisions-Berechnungen mit einbezogen. So kann die Kollisionsabfrage immer abhängig von dem Avatar berechnet werden und erlaubt eine einheitliche Kollisionsbehandlung.

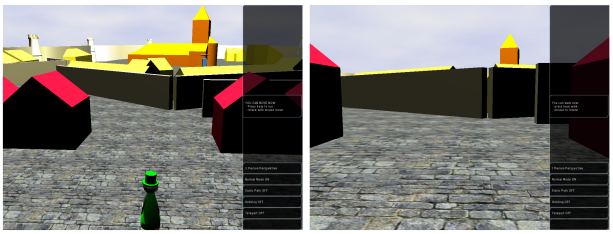


Abbildung 5.4.: Darstellung derselben Szene mit beiden Benutzerperspektiven: (a) Drittpersonansicht, (b) Egoperspektive

5.4. Navigationsmetaphern

In diesem Abschnitt werden die Implementierungen der in Kapitel 4 ausgewählten Navigationsmetaphern erläutert. Mit Hilfe der Tasten „F1“ bis „F3“ ist ein Umschalten zwischen den einzelnen Metaphern während der Anwendung möglich.

5.4.1. Laufen-Metapher

Das Laufen ist die voreingestellte Navigationsmöglichkeit, wenn die Anwendung gestartet wird. Sie wird als Standard- oder „normale“-Fortbewegung betrachtet, weil sie der Fortbewegung in der Realität am nächsten kommt. Ist keine andere Navigationsmetapher ausgewählt, so ist automatisch diese aktiv.

In dieser Arbeit findet die *Laufen*-Metapher ausschließlich auf ebenem Boden statt. Somit ist keinerlei Betrachtung von Steigungen notwendig. Dabei ist die einfachste Möglichkeit der Umsetzung die Position des Avatars um die Veränderlichkeit der Höhenachse (üblicherweise die y-Achse) zu beschränken. Zudem gibt es keinen *run-mode*, da die konstruierte Umgebung relativ klein und überschaubar ist und ein schnelleres Bewegen keinen Vorteil auf kurzen Strecken bietet. Das Laufen versteht sich somit in der Umsetzung dieser Arbeit als ein **konstantes** Fortbewegen auf einem Boden ohne Höhenveränderung der Spielfigur. Dazu wird die Geschwindigkeit zu Anfang durch einen konstanten Wert festgelegt und ist nicht durch den Benutzer beeinflussbar.

Für die Bewegungsrichtung sind die Pfeiltasten sowie die „WASD“-Tasten zu benutzen. Zudem ist ein Drehen des Körpers mit Hilfe der Maus möglich. Rechts- und Linksbewegungen bewirken hierbei eine Rotation um die lokale y-Achse. Vor- und Zurückbewegungen lassen den Avatar seinen Blick gegen den Himmel oder den Boden wenden (Rotation um die lokale x-Achse).

5.4.2. Pfad-Metapher

Die *Pfad*-Metapher wurde in dem implementierten Szenario auf zwei Arten integriert. Sie dient einmal dem Ziel der Orientierung (*vollautomatischer Pfad*) und einmal der Inspektion (*Inspektionspfad*). Dabei kann über die Tasten „F1“ (*Inspektionspfad*) und „F2“ (*vollautomatischer Pfad*) ein Ein- bzw. Ausschalten der entsprechenden Metapher erfolgen.

Der Inspektionspfad:
Dieser Pfad wurde entsprechend der *Orbit*-Metapher implementiert. Somit liegt der Fokus der Kamera immer auf dem zu inspizierenden Objekt und sie dreht sich in einer bestimmten Entfernung um dieses herum.

Nach Umschalten auf diese Metapher wird dem Benutzer ein Maus-Cursor eingeblendet und er kann die Kamera und den Avatar mit Hilfe der Mausbewegung nicht mehr drehen. Stattdessen steuert die Maus den eingeblendeten Cursor. So ist es ihm möglich ein Objekt in der Szene anzuwählen. Dies wird mit Hilfe eines Strahls durch die Kamera in die virtuelle Szene berechnet. Das erste getroffene Objekt entspricht dabei dem Ausgewählten. Hat eine Auswahl stattgefunden, wird dem Benutzer zur Information der Name des Objektes in der GUI angezeigt. Zudem wird entsprechend der *Bounding-Box* des Objektes berechnet, ob es sich dabei um ein großes oder kleines Objekt handelt. Für diese Berechnung wurde zuvor ein Schwellwert festgelegt der groß und klein definiert. Dem entsprechend wird der Abstand zwischen Kamera und Objekt festgelegt. Bei kleineren Objekten ist dieser entsprechend geringer (siehe auch Abb. 3.6, S. 26). Die Höhe zwischen Kamera und Boden wird dabei so gewählt, dass leicht von oben auf das Objekt herab geblickt werden kann. Da sich die Kamera nicht unbedingt weiterhin auf dem Boden befindet, wird eine neue Perspektive eingeführt. Die Berechnung für diese Perspektive beeinflusst ausschließlich die Kamerabewegung und nicht etwa einen Avatar - wenn vorhanden. Dieser bleibt stehen und die Kamera bewegt sich losgelöst von ihm auf dem Pfad um das Objekt.

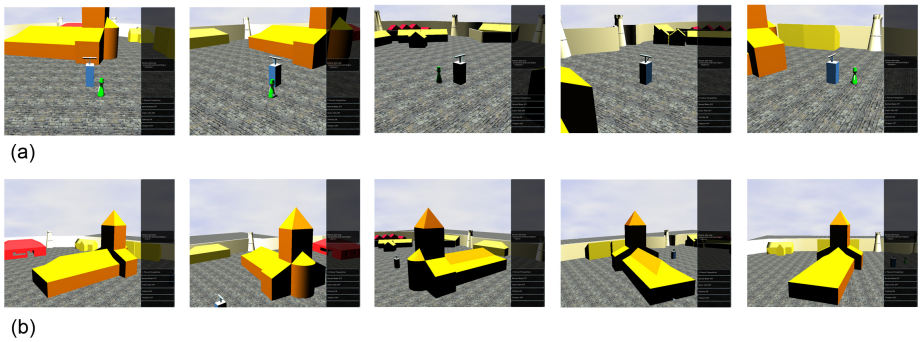


Abbildung 5.5.: *Inspektionspfad*, *Orbit*-Metapher: (a) Drehung um kleines Objekt (Skulptur), (b) Drehung um großes Objekt (Kirche)

Mit Hilfe des *OrbitNode*-Knotens kann dies einfach umgesetzt werden (siehe Abb. 5.6).

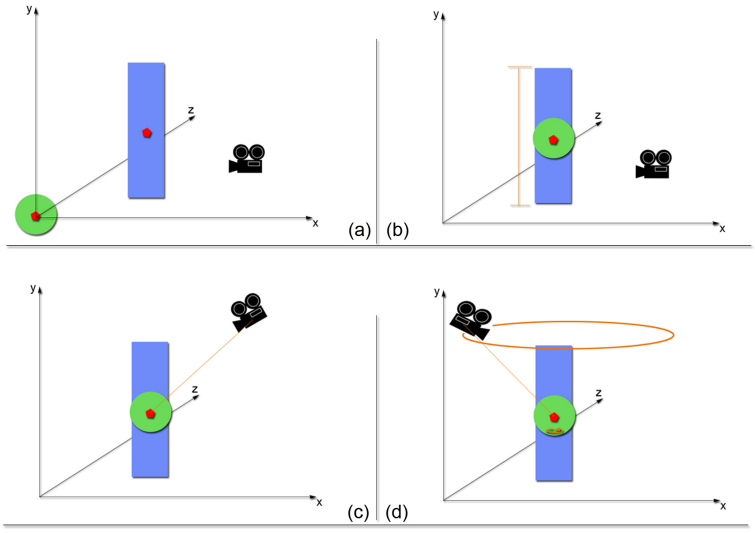


Abbildung 5.6.: *Inspektionspfad*, *Orbit*-Metapher: Das zu inspizierende Objekt ist dabei blau dargestellt und der *orbitNode*-Knoten zur Anschauung grün visualisiert. Positionen sind immer rot markiert und Berechnungen werden orange veranschaulicht. (a) Ausgangssituation, (b) Umpositionierung *OrbitNode*-Knoten und Berechnung der Größe des Objektes, (c) Umpositionierung der Kamera, (d) Bewegung der Kamera auf dem Pfad, durch Rotation des *OrbitNode*-Knotens um seine eigene Achse

In der Ausgangssituation (siehe Abb. 5.6 (a)), steht das zu betrachtende Objekt an einer Position in der virtuellen Welt. Die Kamera in Egoperspektive blickt auf das Objekt und der *OrbitNode*-Knoten hat die Position (0,0,0). Nach Selektion eines Objektes (siehe Abb. 5.6 (b)) wird die Position des *OrbitNode*-Knotens auf die des ausgewählten Objektes gesetzt. Zusätzlich wird die Größe des Objektes berechnet. Der Kinderknoten *OrbitCamNode* wird anschließend (siehe Abb. 5.6 (c)) um einen entsprechenden Abstand (je nach Objektgröße) transliert und die Kamera wird an diesen angehangen. So ist die Kamera losgelöst von dem Avatar. Der Fokus der Kamera ist dabei immer auf die Position des *OrbitNode*-Knoten gerichtet. Ist die Kamera positioniert ((siehe Abb. 5.6 (d))), kann der Benutzer durch die rechts- und links-Tasten (bzw. „a“-Taste und „d“-Taste) die Kamera auf dem Pfad um das Objekt rotieren. Dies geschieht durch eine Rotation des *OrbitNode*-Knotens um seine eigene Achse. Die Kamera bewegt sich aufgrund der Szenenhierarchie im gleich bleibenden Abstand mit.

Der vollautomatische Pfad:
Bei dieser *Pfad*-Metapher wird ein Pfad durch die gesamte Szene festgelegt. Ist dieses Verfahren aktiv, kann der Benutzer nur noch steuern in welche Richtung er blicken und ob er weitergehen oder stehen bleiben möchte. Die Richtung in die er sich fortbewegt ist vorbestimmt. Entspricht die Richtung des Pfades dabei nicht der Richtung, die die gedrückte Taste beschreibt, wird die Benutzung für den Anwender unintuitiv, da sich die Kamera (in Egoperspektive) bzw. der Avatar (in Drittpersonansicht) entlang des Pfades bewegen, dies aber nicht unbedingt der gewählten Taste entspricht. Beispielsweise könnten die Pfeiltasten „vor“ und „zurück“ belegt werden, um den Pfad entlang zu gehen. Führt der Pfad jedoch nach links, so entspricht die Taste „vor“ einem Links-Gehen. Aus diesem Grund wurde ein Wagen integriert, indem sich der Avatar (bzw. die Kamera) während der Fahrt befindet (siehe Abb. 5.7).

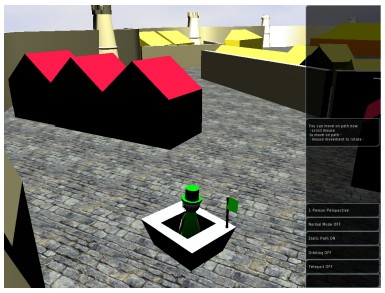


Abbildung 5.7.: Vollautomatischer Pfad in der Testumgebung: Sicht des Benutzers in Drittpersonansicht

Um den Pfad sowohl vorwärts, als auch rückwärts intuitiver abzufahren, wurde das Scrollrad der Maus als Steuerung integriert. Ein Vorwärtsscrollen beschreibt dabei das Bewegen zum nächsten Eckpunkt, wobei das Rückwärtsscrollen zum vorherigen Eckpunkt führt. Scrollt der Benutzer nicht, bleibt er stehen. Dies entspricht einer Fahrt auf unsichtbaren Schienen ohne die Möglichkeit aussteigen zu können.

Der Benutzer kann seine Blickrichtung weiterhin durch die Mausbewegung ändern und die Kamera in dem Wagen drehen. Ein Fähnchen an dem Wagen zeigt dem Benutzer dabei an, wo vorne und wo hinten ist. Gelangt der Wagen an einen Eckpunkt und dreht sich in Richtung des nächsten Zielpunktes, so verändert sich die Blickrichtung der Kamera jedoch nicht, da diese losgelöst von der Wagenbewegung berechnet wird.

Um den Benutzer durch die gesamte Szene zu führen, wurden 24 Eckpunkte festgelegt, die den gesamten Pfad bilden (siehe Abb. 5.8, S. 92).

Der Wagen fährt diesen ab. Die Berechnung der Bewegung zwischen den Eckpunkten findet wie folgt statt:

- *Initialisieren:*
  - Liste 1 anlegen mit allen noch abzulaufenden Eckpunkten
  - Leere Liste 2 anlegen für alle abgelaufenen Eckpunkte
- *Berechne Bewegungrichtung:*
  - Festlegen von Start- und Zielposition (aktueller/erreichter Punkt und nächster Punkt aus Liste 1)
  - Berechnen der Distanz zwischen aktuellem Punkt und Zielpunkt
  - Verschieben des Startpunkts in Liste 2
- *Eingaben des Benutzers erhalten:*
  - Nach vorne Scrollen: goForward = 1 (siehe Abb. 5.8 (b) blaue Pfeile)
  - Nach hinten Scrollen: goForward = -1 (siehe Abb. 5.8 (b) rote Pfeile)
- *Abfrage welche Richtung gefahren wird:*
  - wenn goForward = 1 und Distance != 0, dann Schritt Richtung Zielpunkt gehen und Distanz um die Schrittlänge verringern
  - wenn goForward = -1 und Distance != 0, dann Schritt Richtung Startpunkt gehen und Distanz um die Schrittlänge addieren
  - wenn goForward = 1 und Distance = 0, dann Zielpunkt erreicht (siehe Abb. 5.8 (b) in blaue Richtung) —> nextLocation
  - wenn goForward = -1 und Distance = 0, dann Zielpunkt erreicht (siehe Abb. 5.8 (b) in rote Richtung) —> nextLocation
- *Bei Erreichen eines Eckpunktes (nextLocation):*
  - In konstante Richtung (siehe Abb. 5.8 (b) Bsp.: 1->2->3 (blau->blau) *Berechne Bewegungrichtung*
  - Bei Richtungswechsel (siehe Abb. 5.8 (b) Bsp.: 1->2->1 (blau->rot)) Liste 1 und 2 tauschen und *Berechne Bewegungrichtung*
- Wenn eine der beiden Listen leer ist, so ist der Start- bzw. Zielpunkt erreicht.

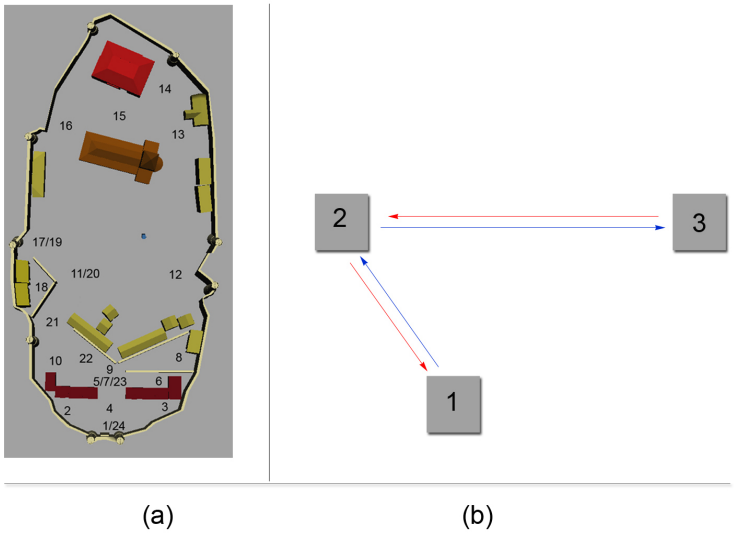


Abbildung 5.8.: Vollautomatischer Pfad in der Testumgebung: (a) Eckpunkte des Pfades, (b) Laufrichtungen zwischen den Eckpunkten, blau - vorwärts; rot - rückwärts

5.4.3. Teleport-Metapher

Durch das Aktivieren der *Teleport*-Metapher ist es für den Benutzer möglich sich ohne zurücklegen des Weges an einen selbst gewählten Zielpunkt zu bewegen. Dabei wurden zwei Ansätze umgesetzt. Die eine berücksichtigt die Auswahl des Zielpunktes direkt in der virtuellen Welt, die andere die Auswahl über die *Übersichtskarte*. In der Erreichung des Zielpunktes sind beide Verfahren gleich.

Nach Aktivierung der *Teleport*-Metapher bekommt der Benutzer zunächst einen Cursor angezeigt, den er mit der Maus bewegen kann. Mit den Pfeiltasten rechts- und links (oder „a“-Taste und „d“-Taste) kann er den Avatar drehen. Nach einem Mausklick wird getestet, ob die Karte oder die virtuelle Welt getroffen wurde. Durch das Anklicken der virtuellen Welt, wird ein Strahl durch die Kamera in die Welt geschickt. Entspricht der Schnittpunkt mit dem ersten Objekt der Szene einem Punkt auf dem Boden, so ist dieser Punkt der Zielort. Wurde jedoch die *Übersichtskarte* getroffen, so wird der Strahl durch die *RenderToTargetCam*-Kamera[2] geschickt.

[2]Diese rendert die virtuelle Szene von oben und ist für die Darstellung der Übersichtskarte verantwortlich (siehe 5.5.2)

Der Schnittpunkt zwischen Strahl und Ebene ergibt dabei jeweils den Zielpunkt. Nach Auswahl des Zielpunktes wird die Position der Kamera, und wenn vorhanden auch die des Avatar, an diese Position gesetzt. Die Ausrichtung des Avatars/bzw. der Kamera ändert sich durch das teleportieren nicht.

Die *Teleport*-Metapher sollte zudem hinsichtlich der Nutzbarkeit innerhalb von Gebäuden getestet werden. So konnten die Benutzer innerhalb des großen Hauses teleportieren sowie außerhalb aller Häuser. Für die restlichen Häuser wurde die *Teleport*-Metapher nicht eingesetzt, da sie zu klein sind, um diese Metapher innerhalb zu testen.

5.5. Navigationsunterstützende Verfahren

Die in Abschnitt 4.2 ausgewählten navigationsunterstützenden Verfahren werden im Folgenden in Hinsicht auf ihre Implementierung in dieser Arbeit beschrieben. Dabei wird wie schon in Abschnitt 3.2 auch nach aktiven und passiven Verfahren unterschieden.

5.5.1. Aktive Verfahren

Das ausgewählte aktive Verfahren, welches notwendigerweise implementiert werden musste, war die *Kollisionserkennung*. Ohne diese verliert die Betrachtung von Gebäuden und die Trennung von innerhalb und außerhalb ihre Bedeutung.

Eine Kollision besteht in der Implementierung aus der Kollisionserkennung und der Kollisionsbehandlung. Die Kollision wird in der konstruierten Welt genutzt, um den Avatar nicht durch Objekte gehen zu lassen, sowie um zu entscheiden, ob sich der Avatar innerhalb oder außerhalb eines Gebäudes befindet. Diese beiden Fälle wurden auf unterschiedliche-Weisen implementiert.

Kollision mit Objekten (Meshes):
Für die *Kollisionserkennung* mit Objekten wird die *Physik-Engine* verwendet. Diese Abfrage ist sehr präzise, da jedes Objekt der Szene anhand ihres *Meshes* betrachtet wird. Würde die *Kollisionserkennung* nicht auf die *Meshes* sondern auf die *Bounding-Box* der einzelnen Objekte reagieren, so könnte kein Gebäude betreten werden, da die Tür (bzw. die Öffnung) sich auch innerhalb der *Bounding-Box* befindet (siehe Abschnitt 5.1.3). Jede Kollision die dabei erkannt wird, kann ausschließlich von dem Avatar ausgelöst

werden, da alle übrigen Objekte in der modellierten Szene statisch sind. Dementsprechend entfallen zusätzliche Abfragen, um herauszufinden welche Objekte beteiligt sind.

Aus diesem Grund wurde die *Kollisionsbehandlung* so implementiert, dass zunächst die aktuelle Position vor einer Bewegung gespeichert wird. Sobald ein Auslöser für eine Bewegung erfolgt, wird dem Avatar/bzw. der Kamera die neue Position zugewiesen und diese auf Kollision geprüft. Bei Kollisionserkennung erfolgt eine Rücksetzung der Position auf die gespeicherte. Ist dies nicht der Fall, wird die neue Position beibehalten. Eine neue Anzeige der Szene erfolgt, sobald die Position geprüft und die Kollision entschieden ist, so dass sich kein Hin- und Herspringen des Avatars/bzw. der Kamera ergibt.

Kollision über Bounding-Boxes:
Ist die Drittpersonansicht gewählt, ist es wichtig zu erfahren, ob sich der Avatar innerhalb eines Gebäudes aufhält. Denn damit der Benutzer seinen Avatar nach Betreten des Hauses weiterhin navigieren kann, muss eine Veränderung (beispielsweise eine Umpositionierung der Kamera oder das Erscheinen von semitransparenten Wänden) stattfinden, bei Verlassen muss dies entsprechend zurückgesetzt werden. Das Betreten muss somit auf jeden Fall registriert werden. Befindet sich der Avatar in einer *axis-aligned Bounding-Box* eines Gebäudes, so hält er sich auch innerhalb dieses Gebäudes auf. Eine solche Abfrage kann nicht über die *Meshes* der Objekte erfolgen, denn in diesen kann er sich nicht aufhalten, da er mit ihnen kollidiert. Durch die *Bounding-Box* kann dagegen das Betreten eines Gebäudes festgestellt werden, indem ein Kollisionstest zwischen Avatar und Gebäude erfolgt.

5.5.2. Passive Verfahren

Die Implementierungen der ausgewählten passiven Verfahren werden im Folgenden näher beschrieben und sollen deutlich machen, wie sie in dieser Arbeit eingesetzt werden.

Transparenz:
Die *Transparenz* soll eingesetzt werden, um dem Benutzer (in Drittpersonperspektive) die Sicht auf seinen Avatar weiterhin zu ermöglichen, wenn dieser in einem Gebäude bewegt wird. Dazu wird bei Betreten die Farbe des Gebäudes semitransparent (siehe Abb. 5.9 (b)). Nach Verlassen des Hauses wird diese Materialeigenschaft wieder zurückgesetzt (siehe Abb.

5.9 (a)). Um sagen zu können, ob der Benutzer durch diese Veränderung der Szene unterstützt oder verwirrt wird, wurde die Semitransparenz nur bei einem Haus implementiert. Die Materialien aller anderen Gebäude hingegen bleiben während der ganzen Anwendung gleich und die Objekte erscheinen opak. Somit muss die Abfrage nach dem Betreten auch nur auf dieses eine Haus hin geprüft werden, was die Leistung des Systems nicht zusätzlich belastet.

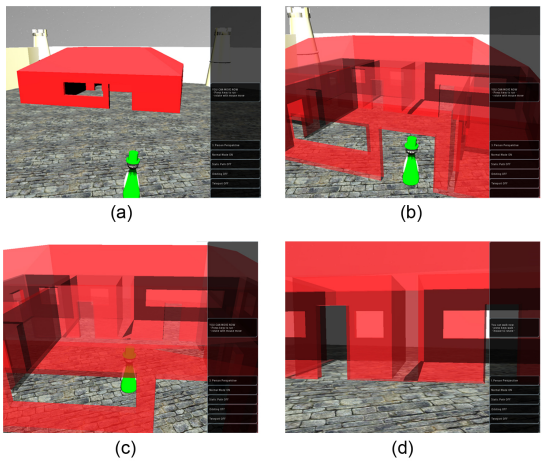


Abbildung 5.9.: *Transparenz*: (a) ursprüngliche Farbe des Hauses, (b) bei Betreten des Hauses, (c) während der Avatar sich in dem Haus befindet, (d) *Transparenz* in der Egoperspektive

Multiple Fenster:
Um zu der virtuellen Szene Informationen und eine Übersichtskarte anzuzeigen, werden zusätzliche Fenster benötigt. So kann die virtuelle Szene, in der sich der Avatar aufhält, getrennt von weiteren Informationen dauerhaft angezeigt werden. Diese sind in eine GUI integriert und bieten so ein einheitliches Bild für den Benutzer. Darin enthalten ist der Hinweis welche Navigationsmetapher zurzeit aktiv ist und mit welcher Eingabe er sich in dieser Metapher fortbewegen kann, sowie die Übersichtskarte (siehe Abb. 5.10). Die GUI wurde dabei mit Hilfe von *CEGUI*[3] entwickelt.

[3]Crazy Eddie's GUI System, eine freie Bibliothek für die Entwicklung von Fenstern und Widgets für Grafik-Engines

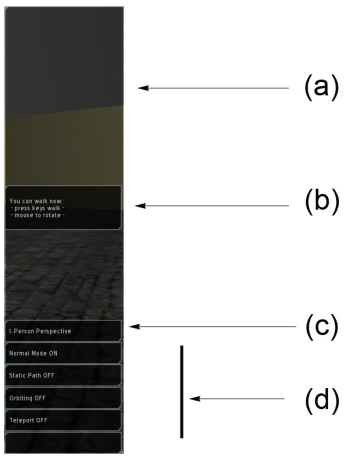


Abbildung 5.10.: *Multiple Fenster*: (a) Platz zur Einblendung der *Übersichtskarte* (siehe auch Abb. 5.11), (b) Informationen zu der ausgewählten Metapher: Eingabemöglichkeiten sowie Fehlermeldungen, (c) Information über die gewählte Perspektive, (d) Informationen zu allen vorhandenen Navigationsmetaphern

Übersichtskarte:
Da die *Übersichtskarte*, wenn aktiviert, für den Benutzer immer sichtbar sein soll, wurde sie in die GUI integriert. Die *Übersichtskarte* ist das einzige navigationsunterstützende Verfahren, das an- und abschaltbar ist. Um den Nutzen der Karte zu beurteilen, müssen die Tests sowohl mit Hilfe der Karte, als auch ohne sie ausgeführt werden können. Bei Abschaltung wird sie jedoch nicht deaktiviert, sondern lediglich ausgeblendet, so dass sie jederzeit wieder „eingeschaltet“ werden kann. Die Karte dient dabei der Orientierung des Benutzers und bietet zusätzlich für die *Teleport*-Metapher auch eine Interaktionsmöglichkeit.

In Abschnitt 3.2.2 wurden zwei Umsetzungsmöglichkeiten für eine, in die GUI integrierbare, *Übersichtskarte* vorgestellt. Sie kann entweder statisch dargestellt werden oder je nach Bewegung des Avatars entsprechend seiner Orientierung rotieren. Da jede Variante einen bestimmten Vorteil besitzt, den die andere nicht aufweist, wurden beide implementiert. So bietet die rotierende Karte den Vorteil immer dieselbe Orientierung, die der Benutzer inne hat, anzuzeigen. Die statische Karte dagegen bringt eine konstante Sicht auf die Welt, da jedes Objekt immer an derselben Stelle dargestellt wird. Werden diese Eigenschaften betrachtet, so liegt es nahe die beiden Varianten abhängig von der Benutzerperspektive zu integrieren. In der Drittpersonansicht wird die statische Karte angezeigt (siehe Abb.

5.11 (a)), die eine globale Übersicht verschafft und die Vorteile der Drittpersonansicht unterstützt. Hat der Benutzer dagegen die Egoperspektive ausgewählt, so bekommt er eine rotierende Karte als Hilfestellung (siehe Abb. 5.11 (b)).

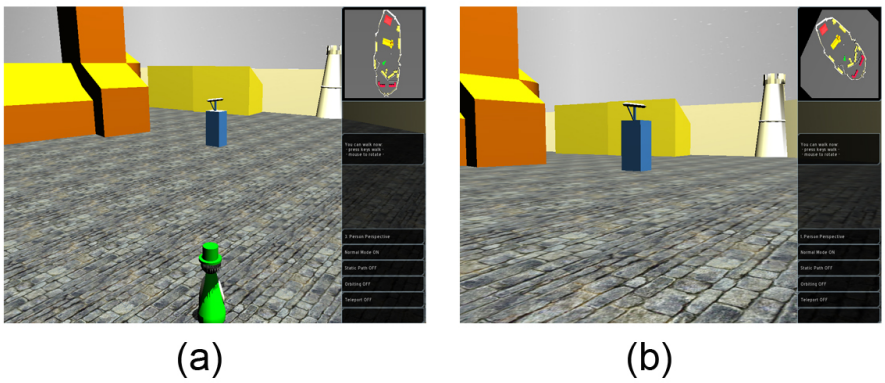


Abbildung 5.11.: *Übersichtskarte*: (a) statisch (Drittpersonansicht), (b) rotierend (Egoperspektive)

Implementiert ist die Übersichtskarte für beide Perspektiven ähnlich. Dazu wurde eine weitere Kamera in die Szene eingebaut *(rttCam: RenderToTargetCamera)*. Sie ist mittig über der Stadt positioniert und auf die Stadt ausgerichtet. Pro Frame wird nicht nur aus der Kamera gerendert, die die Sicht des Benutzers darstellt, sondern auch aus der rttCam-Kamera. So sind die Übersichtskarte und der Standort der Spielfigur zu jeder Zeit aktuell. Der Standort der Spielfigur wird dabei durch einen grünen Pfeil dargestellt, der die Orientierung der Spielfigur wiedergibt. Entsprechend zeigt seine Spitze immer in die Blickrichtung der Figur.

Beim Umschalten von der Drittpersonansicht in die Egoperspektive wird die *rttCam*-Kamera an der Orientierung des Benutzers ausgerichtet und rotiert mit dieser mit. Wird in die Drittpersonansicht zurückgewechselt, so wird die Kamera auf ihre ursprüngliche Position (die genordete Orientierung) zurückgesetzt.

Bei der *Teleport*-Metapher erfüllt die *Übersichtskarte* einen weiteren Zweck. Der Benutzer kann durch Klicken auf diese seine Zielposition bestimmen (siehe Abschnitt 5.4.3). Sie ermöglicht ihm somit das Erreichen auch weit entfernter und nicht mehr sichtbarer Orte. Dazu wird ein Strahl durch die *rttCam*-Kamera geschickt. Dieser hat auf jeden Fall einen Schnittpunkt mit dem Boden, welcher die Zielposition ergibt.

Markierungen:
In der virtuellen Szene stellen verschiedene Objekte *Landmarks* dar. Dazu gehören vor allem die Kirche, die Statue und das große Haus (siehe Abb. 5.1, S. 81). An diesen Markierungen kann sich der Benutzer besonders gut orientieren, da sie genau einmal in der Welt vorhanden sind und ihre Darstellungen und Eigenschaften nicht denen der anderen Gegenständen entsprechen. Der Benutzer kann jedoch eigenständig keine weiteren Markierungen hinzufügen.

Schattendarstellung:
Für alle Objekte in der Szene sollte ursprünglich Schatten berechnet werden. In *Ogre* gibt es hierzu verschiedene Möglichkeiten, je nachdem wie aufwendig dieser berechnet werden soll. Die Hardware des Systems war jedoch damit überlastet, zu allen Objekten den entsprechenden Schatten zu rendern, da zu viele Polygone in der Szene vorhanden waren. Aus diesem Grund wurde die virtuelle Szene ohne Schatten dargestellt. Ausschließlich der Avatar wurde mit einem Schatten gerendert, der auf der *Shadow-Volumes*-Technik beruht. So kann bei Bewegung des Avatars seine Höhe eingeschätzt und die Entfernung zu anderen Objekten besser beurteilt werden.

5.6. Integration aller Bausteine

Durch das Zusammenspiel aller in diesem Kapitel erläuterten Bausteine lies sich schließlich die komplette Testumgebung schaffen. Die virtuelle Szene besteht aus den modellierten Objekten und bietet so die Umgebung für die folgenden Tests. Die GUI hilft dem Benutzer sich zurecht zu finden und bietet ihm Informationen über die Navigationsmetaphern und deren Benutzung, sowie eine *Übersichtskarte*. Ein Perspektivenwechsel ist mit Tastendruck möglich, wodurch sich gleichzeitig die Darstellung der Karte ändert. Diese kann zudem auch ausgeblendet werden. Zusätzlich können auch die Navigationsmetaphern per Tastendruck umgeschaltet werden. Die speziell für die *Search*-Aufgabe angefertigten Objekte können per Tastendruck ein- und ausgeblendet werden. Die navigationsunterstützenden Verfahren dagegen aktivieren sich während der Anwendung automatisch oder sind konstant aktiv.

Zu all den genannten Bausteinen, gibt es noch eine *Log*-Funktion, die während der gesamten Anwendung aktiv ist und übergreifend im Gesamtsystem eingesetzt wird. Dabei werden bestimmte Eingaben des Benutzers

protokolliert:

- Das Umschalten zwischen den einzelnen Navigationsmetaphern und den Benutzerperspektiven, zur Berechnung der Zeit, die jede einzelne Testperson in den verschiedenen Metaphern und den beiden Benutzerperspektiven verbracht hat.
- Das Betreten und Verlassen des großen Hauses, zur Berechnung des zeitlichen Aufenthaltes.
- Das Vorwärts- und Rückwärtsabfahren über den Pfad, zur Feststellung der Bewegungsaktivität während der *Pfad*-Metapher.
- Das Teleportieren über die Karte und die virtuelle Welt, zur Feststellung der bevorzugten Möglichkeit zur Auswahl der Zielposition.
- Das Ein- und Ausschalten der *Übersichtskarte*, als Vermerk der Gruppenzugehörigkeit der Testperson.
- Das Einschalten der Such-Objekte, als Vermerk des Beginns der *Search*-Aufgabe.
- Die Auswahl des Objektes zum Inspizieren, zum Vermerk der betrachteten Objekte.

Zudem können über die Umschaltungen der Navigationsmetaphern die benötigten Zeiten für die einzelnen Aufgabenbewältigung berechnet werden. Die Daten sind für eine Beurteilung der Effizienz der einzelnen Metaphern wichtig und werden dazu in eine Text-Datei ausgegeben und manuell ausgewertet.

6. Benutzertests

Um die Implementierungen der Navigationsmetaphern bezüglich der gestellten Aufgaben zu testen, mussten Benutzertests durchgeführt werden. Diese sollten die zuvor aufgestellten Hypothesen (siehe Abschnitt 6.1) erhärten oder widerlegen. Die Testumgebung wird in Abschnitt 6.2 erläutert, um den speziellen Aufbau für die Tests deutlich zu machen. Durch eine Beschreibung der Testabläufe (siehe Abschnitt 6.3) wird verdeutlicht, wie die Ergebnisse zustande gekommen sind. Diese werden in Abschnitt 6.4 genauer betrachtet, anschaulich dargestellt und interpretiert.

6.1. Hypothesen

Vor der Durchführung der Benutzertests wurden Hypothesen aufgestellt, die das zu erwartende Ergebnis darstellen. Dabei werden die Navigationsmetaphern und die Benutzerperspektiven in Hinsicht der vier Aufgaben betrachtet. Darüber hinaus werden die navigationsunterstützenden Verfahren unabhängig von den Aufgaben eingeschätzt. Die hier aufgeführten Hypothesen stellen dabei sowohl teilweise eine Zusammenfassung der in Kapitel 6 ausgeführten Ziele, als auch ergänzend eine Ausführung der an die Ergebnisse gestellten Erwartungen dar.

6.1.1. Navigationsmetaphern und Benutzerperspektiven

Da sich die Wahl der effizientesten Navigationsmetapher und der besseren Benutzerperspektive in den vier Aufgabentypen sehr unterscheiden, werden sie einzeln betrachtet und Hypothesen entsprechend den Aufgaben aufgestellt.

Aufgabenübergreifend:
Bei der Perspektive kann unabhängig von der Aufgabe davon ausgegangen werden, dass vermutlich die Egoperspektive innerhalb und die Drittpersonansicht außerhalb von Gebäuden bevorzugt wird. Bei einem Durchqueren von Öffnungen wird wahrscheinlich die Drittpersonansicht gewählt, da

sich somit leichter Höhe und Breite einschätzen lassen. Zudem wird die Einsetzung der *Teleport*-Metapher bei Verwendung innerhalb von Gebäuden bei jeder Aufgabe als unerwünscht angenommen. Statt dessen wird wahrscheinlich die *Laufen*-Metapher bevorzugt.

Explore-Aufgabe:
Bei der Orientierung ist zu erwarten, dass die *Pfad*-Metapher sehr gut bewertet wird, da diese den Benutzer in der gesamten Szene herumführt und ihm so die Sicherheit gibt, keinen wichtigen Ort ausgelassen zu haben. Dabei wird davon ausgegangen, dass die Testpersonen sich vor allem Vorwärts auf dem Pfad bewegen und stehenbleiben um sich umzuschauen.

Zudem wird wahrscheinlich die *Laufen*-Metapher als gut bewertet, da sich mit ihrer Hilfe die komplette Szene selbstständig erforschen lässt. Einige Benutzer könnten dies dem Pfad vorziehen, um ihren Weg selbst zu bestimmen und die Sicherheit, jeden Ort der Szene gesehen zu haben, dafür gerne aufgeben.

Die *Teleport*-Metapher scheint am wenigsten geeignet, weil die Chance etwas zu übersehen zu hoch ist. Für Gruppe B wird davon ausgegangen, dass die Wahl der Zielposition in der virtuellen Szene direkt ausgewählt werden soll, da die Wege, die zurückgelegt werden nur einen kleinen „Sprung“ darstellen sollen und die Chance über die Karte zu weit weg zu geraten wesentlich höher ist.

Die gesamte Szene wird wahrscheinlich mit Hilfe der *Pfad*-Metapher am Schnellsten durchlaufen. Bei der *Teleport*-Metapher wird dagegen wohl am Längsten navigiert, da der Benutzer sich eventuell nie sicher ist, ob er alle vorhandenen Orte erreicht hat. Die *Laufen*-Metapher wird aller Wahrscheinlichkeit nach gerade ein Zwischenergebnis erzielen.

Als meistgewählte Perspektive wird die Drittpersonansicht erwartet, da sie einen besseren Überblick verschafft und somit eine effizientere Orientierung unterstützt.

Search-Aufgabe:
Für eine Navigation während der Suche nach Objekten wurde bereits im Vorfeld der Benutzertests die *Laufen*-Metapher festgelegt, die als Einzige zum Einsatz kommt. Es wird erwartet, dass alle Objekte in der Szene innerhalb von wenigen Minuten gefunden werden.

Als Perspektive wird voraussichtlich die Drittpersonansicht gewählt, da sie einen besseren Überblick bietet. Somit können Objekte schneller gefunden werden, weil der Sichtbereich wesentlich größer ist.

Go-back-to-known-target-Aufgabe:
Von den beiden betrachteten Navigationsmetaphern (*Laufen*- und *Teleport*-Metapher) ist die anzunehmende bessere Möglichkeit, an einen bekannten Ort zurückzukehren, die *Teleport*-Metapher. Da bei dieser Aufgabe die Schnelligkeit an den Zielort zu gelangen wichtiger ist als der Weg dorthin, hat die *Teleport*-Metapher durch das „Springen" den entscheidenden Vorteil vor allem auch weite Strecken rasch zurückzulegen. Bei der *Teleport*-Metapher wird von den Probanden aus Gruppe B aller Wahrscheinlichkeit nach die Auswahl der Zielposition über die Übersichtskarte bevorzugt.

Die Perspektive, die wahrscheinlich den größeren Vorteil bringt ist die Drittpersonansicht. Anzunehmen ist, dass sie auch nach einer Teleportation einen Überblick über die neue Position verschafft und so Verwirrung oder (plötzliche) Desorientierung verhindert.

Inspect-Aufgabe:
Soll der Benutzer ein Objekt inspizieren, so wird davon ausgegangen, dass die *Orbit*-Metapher bevorzugt verwendet wird. Dem Benutzer wird hierdurch ein besserer Blick auf das Objekt ermöglicht und das Umkreisen und Betrachten eines Objektes wird wahrscheinlich wesentlich erleichtert. Dabei wird erwartet, dass das Umkreisen eines Objektes mit Hilfe der *Orbit*-Metapher wesentlich schneller geht, als ein Inspizieren mit Hilfe der *Laufen*-Metapher.

Werden Objekte inspiziert, benötigt der Benutzer wahrscheinlich keinen Überblick, sondern ausschließlich einen Blick auf das Objekt. Der eigene virtuelle Charakter könnte aus der Drittpersonansicht sogar im Weg stehen und die Sicht auf das Objekt versperren. Aus diesem Grund ist für das Inspizieren die Egoperspektive oder einer von dem Avatar unabhängigen Perspektive wahrscheinlich sinnvoller.

Zusätzlich stellt sich an dieser Stelle noch die Frage, ob es angenehmer und weniger ablenkend für den Benutzer wäre, wenn die Szene um dieses Objekt während dem *Orbiting* ausgeblendet würde. Zu erwarten ist, dass der Benutzer die Szene nicht ausgeblendet haben möchte, um die Relationen zwischen den Objekten weiterhin zu sehen. Doch in einem Vergleich mit der Möglichkeit die *Orbit*-Metapher in einem extra Fenster darzustellen, indem alleine das Objekt angezeigt würde, erhielte das extra Fenster den Vorzug. Weiterhin anzunehmen ist, dass in diesem Fenster die Szene auch nicht als Hintergrund zu sehen sein sollte.

6.1.2. Navigationsunterstützende Verfahren:

Die navigationsunterstützenden Verfahren sind bis auf die *Übersichtskarte* aufgabenübergreifend. Aus diesem Grund werden die Verfahren allgemein betrachtet und nur auf die Aufgaben bezogen, wenn unter diesen unterschiedliche Ergebnisse zu erwarten sind.

Aktive Verfahren:
Die *Kollision* wird der Wahrscheinlichkeit nach von dem Benutzer gar nicht bewusst wahrgenommen, da er sie aus der Realität kennt. Auf diese Weise wird der Anwendung mehr Realitätsnähe verschafft.

Passive Verfahren:
Durch die Darstellung des Schlagschattens bei dem Avatar geht der Benutzer wahrscheinlich davon aus, dass sich dieser auf dem Boden befindet und nicht schwebt. Zudem werden Formen von Objekten dank ihrer Selbstverschattung womöglich besser erkannt. Da zu den anderen Objekte keine Schatten berechnet werden, wird vermutet, dass eine schlechtere Einschätzung erfolgt, was sich jedoch nicht messen lässt.

Die Kirche und die Statue werden unbewusst als *Landmarks* erkannt. Sie helfen während der Orientierung und zur Bewältigung der Aufgabe, die richtige Abbildung zu der Szene herauszufinden. Dabei wird vermutet, dass die Testpersonen aus Gruppe A mehr auf die *Landmarks* angewiesen sind, als die Probanden der Gruppe B, da diese die *Übersichtskarte* zur Verfügung haben.

Ist die Drittpersonansicht gewählt und die virtuelle Figur wird in ein Gebäude hineingelenkt, so wird eine Umschaltung auf ein semitransparentes Material des Hauses wahrscheinlich erwünscht. In Egoperspektive dagegen sollen die Gebäude vermutlich wie auch bei einer Betrachtung von außerhalb dieser opak erscheinen. In Drittpersonansicht ist die Semitransparenz wahrscheinlich sinnvoller, da sich die Kamera nicht im selben „Raum“ befindet, wie die virtuelle Figur, der Benutzer diese so aber weiterhin sehen und gezielt navigieren kann. In Egoperspektive ist dies wahrscheinlich weniger effizient, da sich die Kamera im Haus befindet. Es wird weiterhin vermutet, dass die Testpersonen in Egoperspektive wegen der Unübersichtlichkeit nicht lange in dem Haus bleiben, im Gegensatz zur Drittpersonansicht.

In der GUI werden die textuellen Hinweise aller Wahrscheinlichkeit nach erkannt und helfen der Testperson voraussichtlich bei der Verwen-

dung der Navigationsmetaphern. Die zusätzlich eingeblendete Perspektive (*Übersichtskarte*) wird mit hoher Wahrscheinlichkeit wahrgenommen und stört den Blick in die Szene nicht.

Die *Übersichtskarte* hilft höchstwahrscheinlich vor allem zur Orientierung und Erfüllung der *Explore*-Aufgabe. Entsprechend ist zu erwarten, dass die Probanden der Gruppe B bei der Aufgabe, die richtige Abbildungen zu der virtuellen Szene zuzuordnen, besser abschneiden. Für die *Search*-Aufgabe wird angenommen, dass die Karte hilft die Szene systematischer abzugehen, um alle Objekte zu finden. Auch in der *Go-back-to-known-target*-Aufgabe ist die *Übersichtskarte* voraussichtlich für die *Teleport*-Metapher aufgrund ihrer Interaktionsmöglichkeit besonders hilfreich, weil der Benutzer mit einem Klick auf sie noch weitere Entfernungen zurücklegen kann. Voraussichtlich navigieren in den drei genannten Aufgaben all die Probanden effizienter, die eine *Übersichtskarte* erhalten haben. Bei der *Inspect*-Aufgabe wird die Karte dagegen wahrscheinlich gar nicht betrachtet, da sie keinerlei detaillierte Informationen über das aktive Objekt enthält. Somit unterscheiden sich die Ergebnisse der Gruppe A wohl nicht von denen der Gruppe B.

Die Kartendarstellung wird vermutlich von der Benutzerperspektive abhängig beurteilt. Zu erwarten ist, dass in der Egoperspektive die mitdrehende Karte hilfreicher ist. Die statische Karte wird wohl dagegen bei Drittpersonansicht bevorzugt.

6.2. Aufbau der Testumgebung

Der Aufbau der virtuellen Szene, die der Benutzer durchlaufen muss ist so gewählt, dass verschiedene Möglichkeiten der Navigation in einer einzigen Szene getestet werden können. In Abbildung 6.1 (S.106) ist der komplette Aufbau dargestellt.
Die Stadtmauer begrenzt dabei den Raum, in dem sich die virtuelle Figur/bzw. die Kamera bewegen kann. Gäbe es eine solche Abgrenzung nicht, wäre die Szene unendlich groß, was jedoch nicht einem Gebäudekomplex entspricht.

Die roten Häuserreihen an dem Startpunkt dienen der Wiedererkennung und Orientierung. In Sackgassen kann getestet werden, wie der Benutzer eine 180 Grad Drehung in der entsprechenden Metapher empfindet. Mit Hilfe der Mauern können Bereiche speziell abgegrenzt werden. So können verschiedene Bereiche auch auf kleinem Raum entstehen, ohne dass eine große Szene benötigt wird. Zusätzlich muss der Benutzer um die-

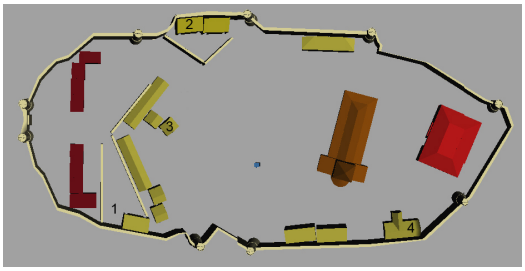


Abbildung 6.1.: Aufbau der Testumgebung, Startpunkt links zwischen den Türmen am Stadttor; Die Zahlen zeigen die Orte für die zu suchenden Objekte. Dabei befindet sich (1) außerhalb, (2) innerhalb, (3) innerhalb des Hauses mit zu kleiner Tür, (4) außerhalb aber unter einem Vorsprung

se herum navigieren und kann die Szene nicht mit einem einzigen Blick überschauen (ohne auf die *Übersichtskarte* angewiesen zu sein) oder auf einem geraden Weg komplett durchqueren. Für die Betrachtung der *Teleport*-Metapher ist dies wichtig, denn so können kurze Strecken verlängert werden (da die Luftlinie zwischen einem Ort vor und hinter der Mauer nicht derselben Strecke entspricht, wie der wirklich abzulaufende Weg). Die gelben Häuser, die direkt an die Stadtmauer angrenzen, sind eher unauffällig. Im Gegensatz zu diesen sind die gelben Häuser in der Mitte der Stadt chaotisch angeordnet und werden dem Benutzer wahrscheinlich mehr auffallen. Hieran kann betrachtet werden, welche Objekte für den Benutzer wirklich als *Landmarks* dienen. Die Kirche und die Statue wurden erstellt, um speziell als Landmarks zu dienen. Zusätzlich sind diese für die *Inspekt*-Aufgabe vorgesehen. Sie sind dabei so in der virtuellen Szene angeordnet, dass die Kamera bei der *Orbit*-Metapher nicht mit der Stadtmauer kollidiert und sich immer innerhalb dieser befindet. Das große rote Haus am Ende der Szene dient mehreren Beobachtungen. Es kann einerseits als zusätzlicher *Landmark* angesehen werden, andererseits sollen speziell zwei Beobachtungen hieran erfolgen. Es ist das einzige Haus, in dem sich mehrere zusammenhängende Räume befinden. Entsprechend soll das Verhalten innerhalb eines Gebäudes vor allem an diesem Haus betrachtet werden. Zudem wird das Haus beim Betreten semitransparent. Alle anderen Häuser bleiben dagegen opak, wodurch ein Vergleich der beiden Darstellungen möglich ist. Eines der gelben Häuser unterscheidet sich zusätzlich von den anderen, da die Türöffnung nicht groß genug ist, dass der Avatar durch sie hindurchpassen könnte. So lässt sich testen, ob es dem Benutzer möglich ist eine richtige Abschätzung in beiden zur

Verfügung stehenden Perspektiven zu treffen. Die *Übersichtskarte* wurde bei den Benutzertests für zehn Testpersonen eingeblendet und für zehn ausgeblendet, so dass ein Vergleich möglich ist, ob sie bei der Orientierung helfen. Zudem konnten die Personen die mit der Karte gearbeitet haben eine Aussage darüber treffen, ob die Entscheidung über eine statische oder eine rotierende Karte von der Benutzerperspektive abhängig gemacht werden sollte und in welcher Ansicht die Karte besser zu bedienen ist. Die Gegenstände, die der Benutzer in der *Search*-Aufgabe suchen muss, wurden sowohl innerhalb, als auch außerhalb von Gebäuden platziert.

6.3. Testablauf

Wird der Aufbau der Szene hinsichtlich der vier Aufgaben betrachtet, sind alle Aspekte vorhanden, um den Benutzer die Aufgaben durchführen zu lassen. Für den Benutzer erscheinen die Aufgabentypen wie folgt:

- Zunächst bewegt er sich in einer vollkommen unbekannten virtuellen Welt.
- Nach der ersten Aufgabe (*Explore*) befindet er sich in einer bekannten Umgebung mit unbekannten Objekten.
- Nach der zweiten Aufgabe (*Search*) befindet er sich in einer bekannten Umgebung mit bekannten Objekten.
- Nach der dritten Aufgabe (*Go-back-to-known-target*) befindet er sich in einer bekannten Umgebung mit bekannten Objekten und hat verschiedene Orte und Objekte mehrfach gesehen.
- Nach der vierten Aufgabe (*Inspect*) kennt er die komplette Umgebung und hat spezielle Objekte zusätzlich näher betrachtet.

Für den Test saß die Entwicklerin jeweils mit einem Proband in einem ruhigen Umfeld zusammen. Abgesehen von einigen durch das Gespräch bedingte Variationen in der Reihenfolge der Fragen, entsprachen die Tests weitestgehend dem folgenden Ablauf:

Zuerst musste der Proband einen kurzen Fragebogen zu seiner Person und seinen Erfahrungen mit Computerspielen ausfüllen. Dann wurde er von der Entwicklerin Gruppe A oder Gruppe B zugeteilt. Dies geschah abhängig von seinem Erfahrungsstand, so dass letztendlich beide Gruppen

ungefähr ausgeglichen waren. Versuchspersonen der Gruppe A bekamen im Gegensatz zu denen aus Gruppe B keine *Übersichtskarte* angezeigt. Dabei kannten die Testpersonen den Unterschied zwischen den beiden Gruppen nicht, damit die Probanden der Gruppe B nicht versucht waren speziell auf die Karte zu achten. Während des gesamten Tests bediente die Testperson die Anwendung und die Entwicklerin stellte zwischendurch Fragen, die sie entsprechend im Fragebogen (Anhang A) ausfüllte.

Nach dem Starten der Testumgebung wurde dem Benutzer die *Explore*-Aufgabe gestellt. Er sollte sich die Welt anschauen, um sich zunächst eine Orientierung zu verschaffen. Dazu wurde ihm die *Laufen*-Metapher inklusive der möglichen Eingaben über Maus und Tastatur erklärt. Zusätzlich wurde er darauf hingewiesen, dass ein Perspektivenwechsel mit der Taste „p“ möglich sei und es ihm während der Anwendung jederzeit freisteht diese zu wechseln. Nachdem die Testperson meinte die Szene erkundet zu haben (spätestens aber nach fünf Minuten), bekam sie sechs Ansichten (von oben auf die Szene) gezeigt. Aus diesen sollte die Versuchsperson bestimmen, welche dargestellte Abbildung der durchlaufenen Szene entspricht. Während den Überlegungen durfte die Szene nicht noch einmal betrachtet werden. Im Anschluss an diese Frage schaltete die Entwicklerin die *Pfad*-Metapher ein. Mit deren Hilfe konnte der Proband die Szene nun erneut betrachten. Auch die Benutzung dieser wurde ihm dazu erklärt. Hatte er den Pfad einmal komplett „abgefahren“, durfte er sich die Abbildungen erneut anschauen und seine Meinung noch einmal verbessern. Anschließend wurde auf die *Teleport*-Metapher umgeschaltet und der Benutzer konnte auch mit dieser ein letztes Mal durch die Szene navigieren, um sich zu orientieren.

Nach der Beantwortung einiger weiterer Fragen zum Thema *Explore*, wurde zu der *Search*-Aufgabe übergegangen. Hierzu schaltete die Entwicklerin wieder auf die *Laufen*-Metapher um und ließ zusätzlich die versteckten Objekte an ihren entsprechenden Orten erscheinen. Während der Benutzer diese suchte, konnte ein Vergleich der Navigation zwischen innerhalb und außerhalb von Gebäuden angesprochen werden. Der Benutzer war gezwungen Gebäude zu betreten und somit auch innerhalb zu navigieren, da einige Objekte innerhalb von Gebäuden versteckt liegen. Hier wurden vor allem Fragen angesprochen, die sich auf die navigationsunterstützenden Verfahren, sowie auf die Benutzerperspektiven bezogen. Hierzu gehörte die Fragestellung, ob die *Transparenz* dem Benutzer geholfen hat; wenn ja, ob dies perspektivenabhängig war. Zusätzlich konnte die Einschätzung der eigenen Maße in den beiden zur Verfügung stehenden Perspektiven beurteilt werden. Da sich ein zu suchendes Objekt innerhalb

des Hauses befindet, dessen Öffnung zu klein ist, war zu beobachten, ob der Benutzer versuchte hineinzugehen oder „sich“ in einer gewählten Perspektive einschätzen konnte. Die *Search*-Aufgabe wurde beendet, indem der Benutzer entweder alle Objekte gefunden hatte oder der Test aus Zeitgründen abgebrochen wurde.

Als nächstes wurde die Testperson zu der *Go-back-to-known-target*-Aufgabe befragt. Die Erfahrung mit der *Laufen*-Metapher und der *Teleport*-Metapher hatte er bereits zuvor gemacht. Wenn er wollte, konnte er diese nun mehrfach gezielt ausprobieren. Auf jeden Fall aber sollte er zu der Statue navigieren und dabei erklären, welche Metapher er bevorzugte, um an einen bereits bekannten Ort zurückzukehren.

Zuletzt war das Inspizieren der Statue und der Kirche vorgesehen. Für die *Inspect*-Aufgabe sollte sich die Testperson zunächst mit Hilfe der *Laufen*-Metapher die beiden Objekte anschauen. Hatte er dies schon während der Orientierungsphase gemacht, weil er zum Beispiel wissen wollte, ob Türen an der Kirche vorhanden sind, so konnte er auf die Erfahrung zurückgreifen. Danach wurde auf die *Orbit*-Metapher bzw. den Inspektionspfad eingegangen. Nachdem er diese Möglichkeit ausprobiert hatte, konnte er die Navigation zwischen den beiden Methoden vergleichend bewerten. Dazu wurden vor allem zwei Fragen formuliert: Die eine bezog sich darauf, ob der Benutzer sich aus der Szene herausgerissen fühlte, weil er seine Spielfigur verlassen hatte bzw. die Kamerahöhe verändert wurde und er somit offensichtlich nicht mehr auf dem Boden stand. Die andere Frage dagegen betrachtete die Möglichkeit ein spezielles weiteres Fenster in die GUI zu integrieren. Zusätzlich wurde sich nach den üblichen Beurteilungen von Bedienbarkeit und Vorzug einer Metapher erkundigt.

Anschließend war der gesamte Test beendet und der Proband wurde entlassen.

Zusätzlich zu dem erwähnten Testablauf, wurde während des gesamten Zeitraumes auf die *Übersichtskarte* eingegangen. Die Fragen unterschieden sich entsprechend nach den Gruppen A und B. Bei den Probanden aus Gruppe A wurde sich lediglich erkundigt, ob das Gefühl bestand, dass eine Karte helfen würde. Die Testpersonen aus Gruppe B sollten dagegen beantworten, welche Art der Karte (statisch oder rotierend) sie bevorzugten und ob dies von der Benutzerperspektive abhing. Zusätzlich wurde ihnen natürlich die Frage gestellt, ob die Karte zur Erfüllung einer speziellen Aufgabe, zum Beispiel der Orientierung, geholfen hatte und wie die Karte für die *Teleport*-Metapher empfunden wurde.

6.4. Ergebnisse

Alle Versuchspersonen benötigten durchschnittlich zwischen 30-60 Minuten für den kompletten Test. Insgesamt wurde der Test mit 20 Probanden durchgeführt. Die Hälfte der Personen bekamen dabei eine *Übersichtskarte* angezeigt.

Unter den Testpersonen befanden sich 14 Männer und sechs Frauen, von denen zwei zwischen 50 und 55 Jahren alt waren. Die anderen 18 Probanden hatten ein Durchschnittsalter von 26 Jahren. Von allen Probanden gab es zehn Personen, die aktuell regelmäßig am Computer spielen, was jedoch vor allem in der Selbstverständlichkeit der Bedienung sichtbar wurde.

Entsprechend der Erfahrung der Testpersonen mit virtuellen Anwendungen, insbesondere Computerspielen, sind sie auf die verschiedenen Navigationsmöglichkeiten bereits konditioniert. Durch die Gewöhnung setzen sie automatisch verschiedene Prioritäten und Anforderungen an die Navigation, welche sich in den Wünschen und Verbesserungsvorschlägen widerspiegeln, die vor allem auch als Ausblick (siehe Kapitel 7) dienen.

Die Ergebnisse sind ebenso wie die Hypothesen nach den Aufgaben gegliedert, um einen Vergleich zu erleichtern.

6.4.1. Navigationsmetaphern und Benutzerperspektiven

Aufgabenübergreifend kann für die Benutzerperspektive festgehalten werden, dass 19 der 20 Testpersonen innerhalb von Gebäuden die Egoperspektive und außerhalb die Drittpersonansicht bevorzugt haben. Dies hatte den Grund, dass innerhalb von Gebäuden eine Detailsicht erforderlich war und der Avatar eine untergeordnete Rolle spielte, wogegen außerhalb die Orientierung, also der Überblick wichtiger war.

Bei einem Durchqueren von Türen (bzw. Öffnungen), also dem Moment des Übergangs zwischen dem Perspektivenwechsel, wird von 19 Personen die Drittpersonansicht gewählt, da die Höhe besser eingeschätzt werden kann und die Testperson besser sieht, ob ein hindurchgehen möglich ist. In Egoperspektive ergibt sich der Irrglaube sich ducken zu können. Bei den erfahrenen Anwendern ergibt sich dies aus anderen Spielen, in denen dies möglich ist. Bei unerfahrenen Anwendern kommt der Bezug zu dem Ducken aus der Realität.

Die Einsetzung der *Teleport*-Metapher innerhalb von Gebäuden war von zwei Personen erwünscht. Sie sahen den Vorteil darin die Tür bei der Navigation nicht „treffen“ zu müssen. Beide Testpersonen waren unerfahren mit virtuellen Szenen und hatten Probleme mit der Bedienung, weshalb

das Klicken und anschließende Teleportieren für sie einfacher zu handhaben war.

Explore-Aufgabe:
Die Wahl der Navigationsmetapher fiel fast einstimmig auf die *Laufen*-Metapher. So entschieden sich 18 Leute für diese Metapher und zwei für die *Pfad*-Metapher. Keiner bevorzugte die *Teleport*-Metapher. In der *Laufen*-Metapher wurde im Durchschnitt am längsten navigiert (siehe Abb. 6.2). Der Grund für die Entscheidung die *Laufen*-Metapher zu nutzen, war vor allem die Realitätsnähe und die Möglichkeit, an jeden beliebigen Ort gehen zu können und nicht darauf warten zu müssen, irgendwann vielleicht an diesen geführt zu werden.

Die *Pfad*-Metapher wurde von den meisten als gut bewertet, da sie einen guten Überblick bietet und die Sicherheit nichts zu verpassen, indem alle Ecken und markante Stellen erreicht werden. Zudem erlangten 13 Testpersonen mit Hilfe des Pfades schneller eine Orientierung über die Szene, als durch die anderen beiden Metaphern (siehe Abb. 6.2 S.112). Neun Personen scrollten dazu den Pfad häufig vor und zurück. Trotz dieser Vorteile bietet sie zu viele Nachteile, als dass diese Möglichkeit am meisten genutzt wurde. Als Negativ wurde empfunden, dass sie dem Anwender die komplette Kontrolle über den Weg entzieht. So wollten alle zumindest die zusätzliche Möglichkeit erhalten aus dem Wagen „aussteigen“ und alleine navigieren zu können. Dies würde eine Kombination mit der *Laufen*-Metapher darstellen. Die Möglichkeit beim Laufen eventuell etwas zu verpassen, wurde dagegen nicht als nachteilig empfunden. Es wurde mehr als störend empfunden alles sehen zu müssen, obwohl die meisten Anwender nicht alles interessiert. Zusätzlich sollte bei der *Pfad*-Metapher die Steuerung ganz übernommen werden, so dass nur die Geschwindigkeit vom Benutzer vorher eingestellt oder währenddessen umgestellt werden kann. Für alle Versuchsteilnehmer war es verwirrend den Weg nachzuvollziehen, da dieser nicht dargestellt wurde. Diese Darstellung wurde von den Testpersonen als ein Pflichtteil der *Pfad*-Metapher beschrieben.

Die Probanden mit *Übersichtskarte* empfanden den Pfad allgemein negativer als die andere Gruppe, da sie zu einer Orientierung bereits die Karte hatten und sich selbst somit einen Pfad durch die komplette Szene leicht überlegen konnten. Zudem sei bei einer kleinen Szene die Chance gering, dass der Benutzer etwas verpassen würde, weshalb ein Pfad vor allem für komplexere und größere Umgebungen geeignet ist. Eine Kombination aus der *Pfad*- und der *Laufen*-Metapher schien bei allen als gute Lösung

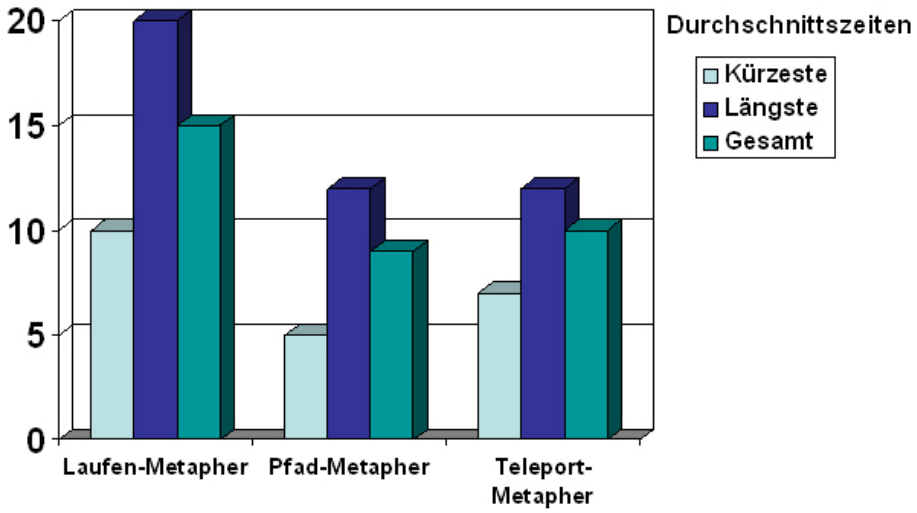


Abbildung 6.2.: Durchschnittszeiten bei der *Explore*-Aufgabe

vorstellbar, um die Vorteile beider Verfahren miteinander zu vereinbaren. Ein Proband schlug dabei vor, eine Umsetzung wie in dem Spiel *Unreal Tournament* zu nutzen. Hier sei es möglich eine Leuchtkugel zu versenden, die an den Zielort fliegt. Der Weg ist somit auch festgelegt, aber die Freiheit der Navigation bleibt dem Benutzer erhalten, da er diesem folgen kann oder auch nicht. Dies entspricht letztendlich der *Pfad*-Metapher mit einem Führer.

Die *Teleport*-Metapher wurde für die Orientierung von 100% der Testpersonen als nicht effizient empfunden. Die Gründe dafür lagen darin, dass sich die Umgebung nach jedem Sprung komplett veränderte und der Benutzer sich somit jedes Mal in alle Richtungen neu orientieren musste und nicht das Gesamtszenario erfassen konnte. Durch das „Springen“ fehlt zudem der Bezug zur Realität. Aus diesem Grund navigierten sechs Probanden nur durchschnittlich fünf Minuten ín dieser Metapher und wollten sie dann wechseln. Zudem navigierten 5 Personen aus Gruppe B lieber über die direkte Auswahl in der Welt, drei Personen über die *Minimap* und zwei versuchten beides gleich oft. Zusätzlich gab es den Wunsch den Gehweg einzublenden, um den aktuellen Standort und die Zielposition miteinander in Verbindung zu setzen. Zur Lösung dieses Problems wurde eine Umsetzung aus dem Spiel *Testdrive Unlimited* erwähnt. Soll darin teleportiert werden, so wird der Zielort über eine Karte angeklickt. Daraufhin

zoomt die Kamera heraus und zeigt eine Ansicht von oben auf die Szene. Der Avatar verschwindet an der aktuellen Stelle und taucht an der Zielposition wieder auf, woraufhin die Kamera wieder „hinunterfliegt“ und die ursprüngliche Perspektive einnimmt.

Während der *Laufen*-Metapher entschieden sich 16 Versuchspersonen für die Drittpersonansicht, zwei für die Egoperspektive, eine benutzte direkt die voreingestellte Perspektive (sie ging davon aus, dass diese sicher die meisten Vorteile bietet) und eine Person schaltete ständig zwischen den Perspektiven um, benutzte beide viel und konnte sich nicht entscheiden. Die Mehrheit entschied sich für die Drittpersonansicht, da diese mehr Übersicht bietet und das speziell für die Orientierung dienlich ist. Entsprechend schlecht an der Egoperspektive ist, dass das Sichtfeld sehr eingeschränkt ist. Ein Versuchsteilnehmer hatte aber das Gefühl, sich in Egoperspektive schneller bewegen zu können und bevorzugte folglich diese.

Search-Aufgabe:
Bei der *Search*-Aufgabe wurde nur die *Laufen*-Metapher verwendet. Die restlichen Metaphern wurden bereits bei der Auswahl des Szenarios in Abschnitt 4.2 ausgeschlossen. Nachfragen an die Testpersonen und die Ergebnisse der Tests bestätigten diese Entscheidung. Mit der *Teleport*-Metapher würden eventuell auch Orte übersprungen werden, an denen vielleicht Objekte versteckt sind. Durch eine *Pfad*-Metapher, die zu den versteckten Objekten führt, verliert die Aufgabenstellung ihre Bedeutung.

Alle vier Orte, die als Verstecke dienten wurden von 18 Testpersonen gefunden. Ausschließlich bei zwei musste abgebrochen werden, da sie sicher waren alle Orte besucht aber kein Objekt gefunden zu haben.

Die Benutzerperspektive ist dabei abhängig davon, ob die Kamera sich innerhalb oder außerhalb von Gebäuden bewegt, wie es bereits am Anfang dieses Abschnittes beschrieben wurde.

Go-back-to-known-target-Aufgabe:
Für diese Aufgabe ergab sich eine allgemeine Meinung über die Nutzung der Navigationsmetaphern. So wurde hier von 18 Testpersonen die *Teleport*-Metapher bevorzugt, von ausschließlich zwei dagegen die *Laufen*-Metapher. Dabei wurde der Vorteil der *Teleport*-Metapher darin gesehen, schneller weite Entfernungen zurücklegen zu können. Alle Probanden der Gruppe B hatten bei dieser Metapher die Möglichkeit die *Übersichtskarte* mit einzubeziehen. Diese entschieden sich einheitlich für die Navigation über die Karte, denn somit können noch weiter entfernte Orte erreicht wer-

den und die Navigation wird wesentlich erleichtert. Zusätzlich wünschten sich die Benutzer statt einem Mauszeiger einen 3D-Punkt in der Welt, ähnlich einem Laserpointer, so dass die Zielposition genauer sichtbar ist. Alternativ war auch die Beschränkung der Zielpositionen auf bestimmte, gekennzeichnete Orte erwünscht. Das Laufen wurde dagegen von den zwei Probanden bevorzugt, da es realistischer ist und keine Verwirrung über die Position entstehen kann.

Die Wahl der Benutzerperspektive lag zu 100% bei der Drittpersonansicht, da bei der Egoperspektive nicht genau erkannt wurde, wo die Zielposition ganz genau sein würde. Zusätzlich trat es häufiger auf, dass die Probanden einen Zielpunkt nahe an Mauern oder Gebäude gewählt haben. Eine Teleportation in Egoperspektive direkt vor eine Wand führt zum Verlust der Orientierung. In Drittpersonansicht ist klar, dass der Avatar vor einer Wand steht und er es wird folglich seine Ausrichtung gedreht, so dass er in die andere Richtung blickt.

Inspect-Aufgabe:
Für die *Inspect*-Aufgabe konnten die Testpersonen zwischen der Laufen-Metapher und der *Orbit*-Metapher wählen. 15 entschieden sich dabei für das Orbiting und fünf für das Laufen. Dabei wurde als Vorteil des Orbiting genannt, dass sich der Benutzer um die Navigation und die Perspektive keine Gedanken machen muss. Der Pfad wird um das Objekt gelegt und die Benutzerperspektive entsprechend der Größe des Objektes angepasst. Dadurch sieht der Benutzer mehr und spart Zeit. Zudem bekommt er einen guten dreidimensionalen Eindruck, wenn er das Objekt von allen Seiten in demselben Abstand anschauen kann. Die fünf Probanden, die sich für die *Laufen*-Metapher entschieden hatten, empfanden die *Orbit*-Metapher als nachteilig, weil sie die Spannung aus dem Spiel nimmt und den Benutzer aus seiner Figur herausreißt. Jedoch alle Probanden navigierten mit der *Orbit*-Metapher schneller und empfanden dies auch so. Ein Einsatz des *Orbiting* beispielsweise im Architektur-Bereich, wurde für eine mögliche Anwendung als gut empfunden, jedoch nicht die Verwendung in einem Spiel. Zudem kann die Figur bei der Betrachtung im Weg stehen, da diese nicht mit bewegt wird. Zusätzlich wollten 12 Testpersonen gerne eine Zoom-Funktion integriert haben, um den Abstand zu dem Objekt selbst bestimmen zu können.

Die Probanden, die sich für die *Laufen*-Metapher entschieden, wollten gerne zwischen beiden Perspektiven wechseln können, um einen Blick auf das Objekt sowohl von oben, als auch von Nahem haben zu können.

Da es bei der *Inspect*-Aufgabe um das Betrachten des Objektes geht, wurde die Frage gestellt, ob es angenehmer und weniger ablenkend für den Benutzer wäre, wenn die Szene um dieses Objeke währenddessen ausgeblendet würde. 16 Versuchspersonen wollten die Szene nicht ausgeblendet haben und empfanden diese nicht als störend, zwei wollten sie gerne ausgeblendet haben und zwei weitere konnten sich eine Darstellung dessen nicht vorstellen und blieben unentschieden. Der Vorteil, wenn die Szene eingeblendet bleibt wurde vor allem darin gesehen, dass alle sichtbaren Objekte in Relation zueinander betrachtet werden konnten und somit der Gesamteindruck der Szene bestehen bleibt. Somit wirkt das ganze Szenario realistischer, natürlicher und reißt den Benutzer nicht so sehr aus dem Spielgeschehen. Ein Kompromiss wäre ein spezielles Fenster, in dem nur das Objekt dargestellt würde. Hier teilten sich die Meinungen und es kann keine einheitliche Aussage gefunden werden. So konnten sich acht Testpersonen vorstellen den Hintergrund auch in diesem Fenster weiter eingeblende zu haben, sowie neun dann meinten sie würden einen neutralen Hintergrund bevorzugen. Drei Probanden konnten sich dieses Fenster nicht vorstellen und konnten keine Aussage treffen. So tauchte eine Ansicht auf, dass nur noch das extra Fenster betrachtet würde. Das eigentliche Fenster bliebe währenddessen ungenutzt und wäre somit verschwendet. Bei komplexeren Objekten war dies dagegen besser vorstellbar. So könnte ein drehendes 3D-Modell des Objektes speziell in einem extra Fenster als Video abgespielt werden. Auch für eine Auflistung wie in Inventarlisten, vor allem in Rollenspielen, wurde diese Möglichkeit als nützlich empfunden.

6.4.2. Navigationsunterstützende Verfahren

Aktive Verfahren:
Die Testpersonen bemerkten erst dann bewusst, dass eine *Kollisionserkennung* stattfand, wenn sie durch diese bei der Navigation behindert wurden. Ansonsten ist sie ihnen gar nicht aufgefallen, wurde als natürlich und normal empfunden und steigerte somit die Realitätsnähe der Anwendung. Verwunderung entstand, wenn durch das Teleportieren ein Weg durch die Wand hindurch möglich war. Also unterstützte die Kollision die Versuchspersonen, weil sie ihre Erwartungen erfüllte.

Passive Verfahren:
Die *Schattendarstellung* half den Testpersonen bei der Einschätzung der Entfernung des Avatars zum Boden, was jedoch bei allen Objekten der

Szene erwünscht gewesen wäre. Es entstanden aber keine Probleme, da kein weiteres bewegtes Objekt in der Szene vorhanden war und Positionen von statischen Objekten ausschließlich einmal richtig eingeschätzt werden müssen. Demnach sind Schatten bei bewegten Objekten wesentlich wichtiger.

Während der *Explore*-Aufgabe sollten die Testpersonen die Szene einer der sechs Szenen zuordnen, die ihnen auf einem Blatt gezeigt wurden (siehe Anhang A). Während den Überlegungen der Testpersonen ordneten sie Objekte der Szene, die besonders auffällig waren und sie sich gut merken konnten, den abgebildeten Objekten zu. Dementsprechend wurde klar, dass die Kirche, die Statue und das Haus, welches beim Betreten semitransparent wird, als *Landmarks* dienten. Dabei halfen die *Landmarks* allen Probanden beider Gruppen ebenso viel, im Gegensatz zu den Erwartungen. Auch während die Testpersonen in der Szene navigierten sprachen sie immer wieder über diese Objekte und orientierten sich an ihnen. Die kleineren Objekte, die im Gegensatz dazu am Rand stehen, wurden in den meisten Fällen nicht beachtet, teilweise sogar ganz übersehen.

Die *Transparenz* wurde meist abhängig von der Benutzerperspektive beurteilt. So haben 12 Probanden das Umschalten auf eine semitransparente Farbe des Hauses in der Egoperspektive als schlecht, in der Drittpersonansicht allerdings als gut bewertet. Zwei Versuchspersonen hatten eine umgekehrte Meinung. Zudem fanden drei Testpersonen die *Semitransparenz* in beiden Perspektiven gut und drei Personen in beiden Perspektiven schlecht. Die Vorteile wurden somit vor allem in der Drittpersonansicht erkannt, also genau dann, wenn Kamera und Avatar sich nicht im selben „Raum“ befinden. Eine Einsetzung der *Transparenz* bringt den Vorteil, trotz allem gut navigieren zu können. Wenn die Kamera allerdings oberhalb eines Gebäudes positioniert ist und auf dieses hinabblickt, so sollte besser nur die Decke semitransparent werden - also ausschließlich die Hindernisse, die sich zwischen der Kamera und dem Avatar befanden. In der Egoperspektive bietet dieses Verfahren deshalb keinen Vorteil, weil sich die Kamera in dem Gebäude befindet und eine Durchsicht nur die Realitätsnähe verringert. All diejenigen, die das Verfahren der *Transparenz* negativ bewertet haben, waren im ersten Moment irritiert, hatten das Gefühl durch Wände laufen zu können oder dachten an ein Glashaus. Die Hälfte der Testpersonen wollten sich aus diesem Grund in Egoperspektive gar nicht erst in dem Haus aufhalten.

Die GUI wurde von den Testpersonen durchweg als positiv empfunden. Die dargestellten Informationen erinnerten daran, wie sie in der gewählten Metapher navigieren können. Die zehn Probanden, die eine *Übersichtskarte*

angezeigt bekamen, empfanden dies als „normal“, da sie diese Darstellung bereits aus anderen Anwendungen kannten. Keiner fühlte den Blick in die Szene dadurch eingeschränkt.

Die *Übersichtskarte* wurde nur zehn der zwanzig Versuchspersonen zur Verfügung gestellt, um ihre Effektivität zu testen. Allgemein kann aber nicht festgestellt werden, dass die Testpersonen aus Gruppe B in den einzelnen Aufgaben aus diesem Grund effizienter navigierten. So waren die Durchschnittszeiten der Gruppen für die *Explore*- und *Search*-Aufgaben nahezu identisch (siehe Abb. 6.3). Aus diesem Grund kann keine aussagekräftige Schlussfolgerung gezogen werden, da zehn Versuchspersonen mit sehr unterschiedlichen Vorkenntnissen noch keinen guten Durchschnitt bilden. So sind die kleinen Unterschiede darauf zurückzuführen, dass es in Gruppe A drei Testpersonen gab, die bisher noch in keiner virtuellen Szene navigiert sind. Entsprechend fällt ihnen die *Laufen*-Metapher sehr viel schwieriger. Wobei sie das Scrollen mit der Maus gewohnt waren, so dass die *Pfad*-Metapher kein besonderes Problem darstellte.

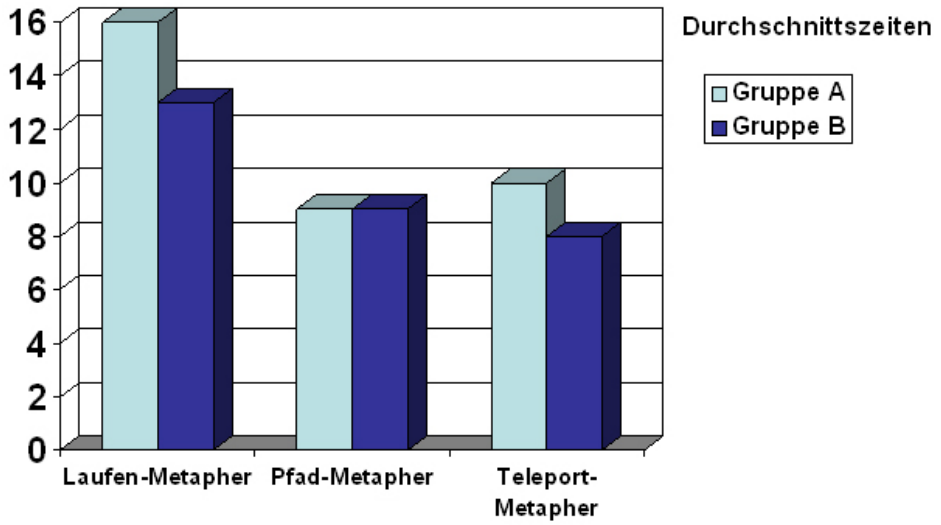


Abbildung 6.3.: Durchschnittszeiten abhängig der *Übersichtskarte*: Alle Metaphern beziehen sich dabei auf die Nutzung in der *Explore*- und der *Search*-Aufgabe

Zu diesem Test wurde während der *Explore*-Aufgabe allen zwanzig Probanden ein Blatt mit sechs Abbildungen gezeigt. Sie sollten entscheiden, welche der Darstellung der virtuellen Szene entspricht. Diese Aufgabe bekamen sie nachdem sie die Szene mit Hilfe der *Laufen*-Metapher erkun-

det hatten. Nur zwei Personen fanden auf Anhieb die richtige Abbildung. Diese beiden Personen waren aus Gruppe A und hatten somit keine *Übersichtskarte* zur Verfügung. Nach einem zweiten Durchlauf mit Hilfe der *Pfad*-Metapher durften alle Testpersonen erneut eine Abbildung benennen. Nach diesem Versuch hatten nun insgesamt zwölf Versuchspersonen die richtige Abbildung gefunden. Davon waren genau die Hälfte aus Gruppe A und die andere Hälfte aus Gruppe B. Daran kann festgestellt werden, dass zum einen eine Orientierung erst nach mehrmaligem durchlaufen einer Szene stattfindet. Zudem hilft eine *Übersichtskarte* nur bedingt der Orientierung. Hilfreicher dagegen sind *Landmarks*. Sieben von den zehn Testpersonen aus Gruppe A sagten, dass ihnen eine Karte bestimmt geholfen hätte. Lediglich 5 aus Gruppe B vermerkten, dass ihnen die Karte tatsächlich weiter geholfen hatte. Drei Probanden aus Gruppe A und fünf aus Gruppe B waren der Ansicht, die Karte würde nicht helfen. Warum die Karte nicht helfen würde (Gruppe A) bzw. nicht geholfen hat (Gruppe B) wurde dadurch begründet, dass die Szene zu klein ist und die Erforschung durch ein Hindurchnavigieren durch diese mehr bringt. Diejenigen, die eine Karte angezeigt bekamen, begründeten ihre falsche Wahl dadurch, dass die Karte auch nur für eine grobe Orientierung diene und kleine Unterschiede darum nicht erkannt werden. Positiv an der *Übersichtskarte* wurde gewertet, dass jeder Ort zu jederzeit sichtbar und durch den Pfeil die Ausrichtung des Benutzers jederzeit zu erkennen ist. Zwei Testpersonen wünschten sich statt des Pfeils einen Sichtkegel.

In der *Search*-Aufgabe wurde beobachtet, ob die Probanden die Karte zum systematischen Abgehen der Szene verwendeten, doch neun der zehn Personen schauten die Karte dazu nicht speziell an.

Besondere Vorteile in der *Übersichtskarte* wurden von allen Probanden aus Gruppe B bei der *Teleport*-Metapher gesehen. Wie schon in dem Abschnitt zu der *Go-back-to-known-target*-Aufgabe erwähnt, bietet die Interaktion mit der Karte eine schnellere Navigation durch die Szene. Dies entspricht den Testergebnissen. So konnten die Probanden aus Gruppe B wesentlich schneller und somit effizienter an einen bekannten Ort zurückkehren.

Für die *Inspect*-Aufgabe wurde die Karte von den Versuchspersonen gar nicht betrachtet, da es lediglich um das Objekt und dessen Inspektion ging, wodurch die Karte keinen speziellen Vorteil bringen konnte.

Da die Karte in den beiden Perspektiven etwas anders dargestellt wurde, mussten die zehn Probanden auch dies beurteilen. Sechs Testpersonen wollten lieber immer eine statische Karte angezeigt bekommen, wogegen vier sowohl die statische als auch die dynamische *Übersichtskarte* hilfreich

empfanden. Einheitlich war die Meinung, dass die statische Karte in beiden Perspektiven eingesetzt werden kann, die drehende Karte dagegen nur in Egoperspektive. Positiv an der statischen Karte wurde empfunden, dass es besser ist, wenn sich die Figur in der Welt bewegt, als die Welt um die Figur, da dies realistischer ist. Wenn sich die Karte dagegen mit der Ausrichtung der Kamera rotiert, so sollte sie auch entsprechend transliert werden, wenn sich die Position der Kamera verändert, so dass der Benutzer immer Mittelpunkt der Karte ist und die Karte sich entsprechend um ihn bewegt.

7. Fazit

Mit dieser Arbeit wurde eine Testumgebung geschaffen, um die effiziente Navigation in virtuellen Szenen zu testen. Nach den Recherchen zu Navigationsmetaphern für virtuelle Szenen, deren Anwendungsgebieten und Zielgruppen, folgte eine Auswahl an Metaphern, die sinnvoll für Gebäudekomplexe eingesetzt werden können. Durch die Implementierung in eine Testumgebung konnten diese näher betrachtet und Ergbnisse gewonnen werden. Die Ergebnisse der Tests sollen in Abschnitt 7.2 resümierend dargelegt werden. Mit Hilfe des Ausblicks (Abschnitt 7.3) sollen die Ideen, die während der Entwicklung entstanden sind, festgehalten werden und als Denkanstöße dienen.

7.1. Bewertung

Die Aufgabenstellung dieser Diplomarbeit wurde in all ihren Anforderungen erfüllt, indem verschiedene Navigationsmetaphern implementiert und anschließend vergleichend miteinander getestet wurden. Wie erfordert lag die Priorität vor allem auf der Navigation in Gebäudekomplexen. Die Metaphern wurden nach und nach in das System integriert, so dass das Gesamtsystem jederzeit erweiterbar blieb und neue Metaphern unabhängig von den bereits Integrierten hinzugefügt werden konnten. Auf diese Weise konnten Navigationsmetaphern integriert werden, ohne das System mit jeder Metapher umstellen zu müssen. So bleibt das System auch weiterhin erweiterbar, auch wenn aufgrund der Zielsetzung diese Eigenschaft nicht vorrangig zu betrachten war.

Der Wunsch ausgefallene Metaphern zu testen und mit schon bekannten und einfachen Verfahren zu vergleichen entstand während der Arbeit. Um komplexere Navigationsmetaphern mit bereits bekannten Möglichkeiten zu vergleichen, mussten diese zunächst integriert werden. Dazu wurden vier Navigationsmöglichkeiten implementiert. Für weitere Metaphern wäre eine Integration zeitlich zu umfangreich geworden. Sicher wäre dies jedoch zusätzlich interessant gewesen. Ein großer Teil des Zeitaufwandes entstand vor allem daraus, fremde *Engines* miteinander zu kombinieren

und ein eigenes System mit Hilfe dieser zu erstellen.

Die zwanzig Nutzertests wurden intensiv vorbereitet, genau durchgeführt und lieferten hypothesenunterstützende Ergebnisse, sowie einige Überraschungen (siehe Abschnitt 6.4). An einigen Stellen wurden jedoch Probleme des Systems festgestellt, die in dieser Phase nicht mehr behoben wurden, weil die Tests bereits größtenteils abgeschlossen waren. Dabei betrafen diese Probleme lediglich Möglichkeiten, wie beispielsweise aus der Szene hinausteleportieren zu können. Dies wurde in den folgenden Tests direkt angesprochen und dem Benutzer explizit verboten, so dass solche Problem ohne zusätzlichen Code eingegrenzt werden konnte. Die Effizienz der einzelnen Metaphern konnte sowohl über die Testfragen, als auch über die Betrachtung der mitgeloggten Daten gut ausgewertet werden und bietet somit eine Grundlage für weitere Ausblicke wie sie auch in Abschnitt 7.3 betrachtet werden.

7.2. Zusammenfassung

Zusammenfassend kann zu den Navigationsmetaphern festgestellt werden, dass ihre Anwendung je nach Aufgabe und Ziel der Anwendung bedacht werden muss. Effiziente Navigation kann nicht global betrachtet werden. Die Anwendung sollte in kleine Aufgaben unterteilt werden. Entsprechend diesen Aufgaben muss eine optimale Kombination aus verschiedenen Navigationsmetaphern und unterstützenden Verfahren zur Lösung eingesetzt werden, um eine effiziente Lösung zu erhalten. So kann beispielsweise eine Methode besonders effizient sein, wie die *Pfad*-Metapher bei der *Explore*-Aufgabe, doch trotz allem wird sie von den Anwendern nicht angenommen. Sobald eine Methode so implementiert wird, dass den Benutzer irgendetwas daran stört, wird er wenn möglich eine andere Navigationsmöglichkeit wählen. Ist eine *Laufen*-Metapher integriert, so wird der Benutzer im Notfall immer auf diese zurückgreifen. Sie wird vom Benutzer nicht angezweifelt oder in Frage gestellt, weil sie der Realität am nächsten kommt und angenommen wird, sobald sie in den Kontext der Anwendung passt. Entsprechend sollte die subjektive Meinung eines Benutzers nie unterschätzt werden. Die Erfahrung des Benutzers mit dreidimensionalen Umgebungen entscheidet zudem darüber, ob die Navigationsmetaphern als sinnvoll und intuitiv betrachtet werden. Für erfahrene Benutzer ist es wichtig die Freiheiten in der Navigation nicht zu sehr einzuschränken, wohingegen unerfahrene Benutzer nicht zu sehr „alleine gelassen“ werden sollten. Entsprechend ist eine Zielgruppenbestimmung notwendig,

um eine weitreichende Akzeptanz der Navigation bei den Benutzern zu erzielen.

Innerhalb von Gebäudekomplexen ist die Navigation insofern besonders zu betrachten, da vor allem auch eine Navigation innerhalb von Gebäuden möglich ist, was sonst nicht jede Anwendung erfordert. Aus diesem Grund ist es wichtig navigationsunterstützende *Kamerasteuerung*, *Transparenz* oder eine andere entsprechend vorteilhafte Methode einzusetzen.

Wird die Wahl der Benutzerperspektive betrachtet so stellt sich ganz klar heraus, dass sich die Drittpersonansicht in den meisten Fällen bewährt. Navigation bedeutet auch immer Orientierung im Raum, dazu eignet sich die Drittpersonansicht besser als die Egoperspektive. Sobald Interaktion mit der Umgebung oder speziellen Objekten integriert ist und eine Detailsicht notwendig wird, ist die Egoperspektive zu bevorzugen.

Bei Gebäudekomplexen besteht durch die Anforderung innerhalb und außerhalb von Gebäuden navigieren zu müssen das Problem, dass sowohl die Detailsicht, als auch die Übersicht gewährleistet sein muss. Gelöst werden kann dieses Problem, indem dem Benutzer eine Umschaltung zwischen den beiden Perspektiven zur Verfügung gestellt wird. Muss diese von dem Anwender manuell geleistet werden, so besteht die Gefahr, dass er dies nicht tut, sondern die Voreinstellung benutzt. Darum bietet es sich an, bei bestimmten Schlüsselaktionen den Perspektivenwechsel zu automatisieren.

Die implementierten navigationunterstützenden Verfahren wurden allgemein als positiv empfunden. Bei einer *Übersichtskarte* ist festzustellen, dass diese besser statisch integriert wird, denn so wurde sie von jeder Testperson akzeptiert. Die rotierende Karte wird dabei von der Hälfte als irreführend angesehen.

Für das Zurechtfinden in Gebäudekomplexen bietet sich eine Karte erst dann an, wenn diese entsprechend groß sind. Bei kleinen Umgebungen helfen vor allem auch *Landmarks* bei der Orientierung.

Überraschend bei all den Ergebnissen war vor allem, dass die *Pfad*-Metapher, trotz ihrer für den Benutzer bietenden Sicherheit alles in der Szene zu sehen, nicht besonders gut angenommen wurde. Zudem hatten einige Probanden Schwierigkeiten mit der Steuerung. Ein Vorwärtsscrollen bedeutete ein weitergehen, bei einer üblichen Anwendung des Scrollens, also beispielsweise eines Scrollbalkens, bedeutet jedoch ein Rückwärtsscrollen ein „weiter“,

da Dokumente nach unten fortgesetzt werden. Auch die Reaktion auf die Karte war teilweise erstaunlich. Die Benutzer merkten vor einem Hinweis nicht unbedingt, dass sich das Verhalten der Karte je nach Perspektive veränderte. Sie beachteten sie gar nicht, weil die Szene zu klein und einfach dargestellt war.

7.3. Ausblick

Statt speziell zu den verschiedenen Aufgaben die Navigationsmetaphern zu testen, würde es sich anbieten eine Aufgabe zu isolieren und die Navigationsmöglichkeiten genauer in Hinblick auf verschiedene Zielgruppen zu testen.

Außerdem könnten weitere Metaphern getestet werden, wie die in Kapitel 3 vorgestellten (z. B. *Ephemeral World Compression*-Metapher und die *Speed couples flying*-Metapher). Dazu müsste eine komplexere Szene erschaffen und die Metaphern dort integriert werden. In dieser könnte auch eine erneute Einsetzung der *Übersichtskarte* erfolgen und mit den Ergebnissen aus Kapitel 6 verglichen werden.

Zudem wäre es möglich, weitere Eingabemöglichkeiten zu testen. Dazu könnten die hier verwendeten Standard-Desktop-Eingabegeräte anders integriert werden oder es könnten andere Geräte verwendet werden. Das Hauptaugenmerk dieser Arbeit lag jedoch nicht speziell auf den verschiedenen Möglichkeiten der Integration der Bedienung und der technischen Umsetzung.

Somit sind viele denkbare Ansätze möglich die Navigation noch weiter zu erforschen und lassen noch viel Spielraum, um die richtigen Lösungen für eine effiziente Navigation sowohl in Gebäudekomplexen, als auch in jeder anderen Umgebung mit allen erdenklichen Aufgaben zu finden und umzusetzen. Dabei kann sich zwar an der Realität orientiert werden, doch sollten nie die vielen Möglichkeiten außer Acht gelassen werden, die eine virtuelle Realität bietet.

Literaturverzeichnis

[BC03] Grigore Burdea and Philippe Coiffet.
Virtual reality technology.
Wiley Verlag, 2003.

[Bin07] Wikipedia: Binomialkoeffizient.
http://de.wikipedia.org/wiki/N_%C3%BCber_k,
Stand: Nov. 2007.

[CRI03] Luca Chittaro, Roberto Ranon, and Lucio Ieronutti.
guiding visitors of web3d worlds through automatically generated tours. In *Web3D '03: Proceeding of the eighth international conference on 3D Web technology*, pages 27–38, New York, NY, USA, 2003. ACM Press.

[CS01] Luca Chittaro and Ivan Scagnetto.
is semitransparency useful for navigating virtual environments? In *VRST '01: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 159–166, New York, NY, USA, 2001. ACM Press.

[Dri07] Wikipedia: Drittpersonansicht.
http://de.wikipedia.org/wiki/Drittpersonansicht,
Stand: Sept. 2007.

[DS93] Rudy P. Darken and John L. Sibert.
a toolset for navigation in virtual environments. In *UIST '93: Proceedings of the 6th annual ACM symposium on User interface software and technology*, pages 157–165, New York, NY, USA, 1993. ACM Press.

[dSGA+00] C. Russo dos Santos, P. Gros, P. Abel, D. Loisel, N. Trichaud, and J.P. Paris.
metaphor-aware 3d navigation. *infovis*, 00:155, 2000.

[Ego07] Wikipedia: Egoperspektive.
http://de.wikipedia.org/wiki/Egoperspektive,
Stand: Sept. 2007.

[HBL02] Stephen Hughes, Peter Brusilovsky, and Michael Lewis.
Adaptive Navigation Support in 3D E-Commerce Activities.
School of Information Sciences, University of Pittsburgh, Pittsburgh PA 15260, 2002.

[Her06] Michael Herczeg.
Interaktionsdesign: Gestaltung interaktiver und multimedialer Systeme.
Oldenbourg Verlag, 2006.

[HW97] Andrew J. Hanson and Eric A. Wernert.
constrained 3d navigation with 2d controllers. In *IEEE Visualization*, pages 175–182, 1997.

[IKMT98] Takeo Igarashi, Rieko Kadobayashi, Kenji Mase, and Hidehiko Tanaka.
path drawing for 3d walkthrough. In *ACM Symposium on User Interface Software and Technology*, pages 173–174, 1998.

[Int07] Wikipedia: Interaktion.
http://de.wikipedia.org/wiki/Interaktion,
Stand: Sept. 2007.

[JRH07] Projektmanagement GmbH und Co. KG Joachim R. Henning.
Cap Vermell Hotel.
http://www.jrh-projektmanagement.de/27-0-cap-vermell-hotel.html, Stand: Okt. 2007.

[KS98] Antonio Krüger and Christoph Stahl.
Intelligente Navigation in 3D-Welten: Zur Rolle graphischer Abstraktion.
http://w5.cs.uni-sb.de/ stahl/research/magdeburg98.pdf, 1998.

[Nav07] Wikipedia: Navigation.
http://de.wikipedia.org/wiki/Navigation,
Stand: Sept. 2007.

[PC99] Fabio Pittarello and Augusto Celentano.
A Multimodal Approach for Orientation and Navigation in 3D Scenes.
citeseer.ist.psu.edu/368238.html, 1999.

[PFW98] Greg Parker, Glenn Franck, and Colin Ware.
visualization of large nested graphs in 3d: Navigation and interaction. *Journal of Visual Languages and Computing*, 9(3):299–317, 1998.

[Rou99] Richard Rouse.
what's your perspective?
SIGGRAPH Comput. Graph., 33(3):9–12, 1999.

[SC03] William R. Sherman and Alan B. Craig.
Understanding virtual reality: interface, application, and design.
Morgan Kaufmann, 2003.

[SSC02] Mel Slater, Anthony Steed, and Yiorgos Chrysanthou.
Computer graphics and virtual environments: from realism to real-time.
Addison Wesley, 2002.

[Sta02] Kay M. Stanney.
Handbook of virtual environments: design, implementation, and applications.
Erlbaum Verlag, 2002.

[STV06] Patrick Salamin, Daniel Thalmann, and Frédéric Vexo.
The Benefits of Third-Person Perspective in Virtual and Augmented Reality?
Virtual Reality Laboratory Ecole Polytechnique Fédérale, Lausanne, 2006.

[Tel07] Wikipedia: Teleportation.
http://en.wikipedia.org/wiki/Teleportation,
Stand: Sept. 2007.

[TRC01] Desney S. Tan, George G. Robertson, and Mary Czerwinski.
exploring 3d navigation: combining speed-coupled flying with orbiting. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 418–425, New York, NY, USA, 2001. ACM Press.

[Vel91] Max Velmans.
consciousness from a first-person perspective. *PERSPECTIVE.BEHAVIORAL AND BRAIN SCIENCES 14*, 14:4, 1991.

[VF03] Kai Vogeley and Gereon R. Fink.
neural correlates of the first-person-perspective. *Trends in Cognitive Sciences*, Vol.7 No.1:pp. 38–42(5), 2003.

[VS99] Francisco J. Varela and Jonathan Shear.
first-person methodologies: What, why, how? In *THE VIEW FROM WITHIN*, pages 1–15.
Imprint Akademic, 1999.

A. Umfragebogen

Umfragebogen:

Umfragebogen: Gruppe A/B

Geschlecht: männlich weiblich

Alter: _____

Spielst du regelmäßig am PC? Ja nein
Wie oft? _____

Welches Spielegenre bevorzugst du?

1. Shooter
2. Rennspiele
3. Rollenspiel
4. Strategiespiele
5. Adventure
6. Jump'n Run
7. Beat'em up (Prügelspiele)
8. Sonstiges: _____

Aufgabenbogen: explore Gruppe A/B

[A= Aufgabe für Proband, F= Frage an Proband, B= Aufgabe für Beobachter] [Gruppe „A“ ohne Karte, „B“ mit Karte]

A1) Du spielst eine kleine Spielfigur, die verschiedene Aufgaben erfüllen muss. Hierzu sollst du zunächst die Welt erkunden, schau dir möglichst alles genau an. Du kannst frei durch die Welt laufen und die Perspektive von Egoperspektive auf Drittpersonansicht umschalten, indem du auf die Taste „p“ drückst. Probiere die Perspektiven aus, indem du sie ab und an umschaltest.

B1) Auf die Uhr schauen, Zeitlimit: max. ca. 10 min
B2) Stadtkarten-Blatt zeigen

F1) Welche der sechs Karten stimmt am besten mit der virtuellen Szene überein? [ ]
(eine Ziffer 1-6 bitte eintragen)
F2) Glaubst du, du bist überall in der Welt gewesen? [ja] [nein]

B3) War die Testperson überall? [ja] [nein]

F3) Welche Perspektive war dir am liebsten? [1.] [3.]

B4) Welche Perspektive wurde am Meisten benutzt (aus Log-Datei) [1.] [3.]

F4) Hattest du in einer der Perspektiven Schwierigkeiten, oder Unannehmlichkeiten?
Falls „ja“, in welcher?
Beschreibung?_____
_____

B5) „F2“, um die Navigation in „staticPath“ umzuschalten.
A2) Nun wirst du auf einem festen Pfad durch die Welt geführt. Mit „p“ kannst du die Perspektive wie gewohnt umschalten. Schau dir alles in der Szene an, bis du wieder am Startpunkt angekommen bist.

F5) Welche der sechs Karten stimmt am besten mit der virtuellen Szene überein? Willst du deine Entscheidung verändern? [ ]
(eine Ziffer 1-6 bitte eintragen)
F6) Welche Perspektive war dir am liebsten? [1.] [3.]

B6) Welche Perspektive wurde am Meisten benutzt (aus Log-Datei) [1.] [3.]

F7) Hattest du in einer der Perspektiven Schwierigkeiten, oder Unannehmlichkeiten?
Falls „ja“, in welcher?
Beschreibung?_____
_____

F8) Hast du dich intensiver mit der Umgebung beschäftigt, weil du nicht auf den Weg achten musstest? [ja] [nein]
F9) Hast du Sachen entdeckt, die du beim dem freien Gehen nicht gesehen hast?
Wenn ja, was? _____

B7) „F3“, um die Navigation in „teleport“ umzuschalten.
A3) Nun sollst du noch die Variante ausprobieren dich durch klicken auf die Karte, zu teleportieren. „p“ funktioniert wie bisher.
F10) Wie findest du das Teleportieren im Gegensatz zu den anderen beiden Möglichkeiten um die Welt zu erkunden?
Kannst du einen Grund angeben? _____

F11) Hast du zwischenzeitlich die Orientierung verloren? [ja] [nein]
Wenn nein: War dies nur kein Problem, weil du die Umwelt schon kanntest?

F12) Welche Perspektive war dir am liebsten? [1.] [3.]

B8) Welche Perspektive wurde am Meisten benutzt (aus Log-Datei) [1.] [3.]

F13) Hattest du in einer der Perspektiven Schwierigkeiten, oder Unannehmlichkeiten? Falls „ja“, in welcher?
_____
_____

Aufgabenbogen: search Gruppe A/B

B9) Schalte die versteckten Objekte an für die Search-Aufgaben „F6“ + „F3“ teleport aus!

A4) Die kleine Spielfigur, die du spielst, hat ihre Spielklötzchen verloren, geh sie suchen.
Du kannst nun frei in der Szene umhergehen. Mit der Taste „P“ kannst du die Perspektive umschalten. Verwende die die Perspektive, die dir am besten hilft entsprechend den nachfolgenden Aufgaben.

B10) Auf die Uhr schauen: max. ca. 10min

F14) Welche Perspektive hast du am Häufigsten verwendet? Drinnen, draußen? 1. 3.
Begründung:

_____
_____
_____

B11) Welche Perspektive wurde am Meisten benutzt (aus Log-Datei) 1. 3.

A5) In ein Haus konntest du nicht hineingehen (das mit dem lila Spielklötzchen), hast du dies in 1. und 3. Person Perspektive versucht? Wenn nein, versuch es noch mal.

F15) Gab es eine Perspektive, in der du das sofort erkannt hast, dass du nicht hinein kommst, oder hast du in beiden ausprobieren müssen? 1. 3.

_____
_____
_____

Aufgabenbogen: known target Gruppe A/B

A6) Gehe nun zu der Statue zurück. Die Perspektive kannst du wie gewohnt mit „p“ umschalten. Du kannst nun frei in der Szene gehen oder dich mittels klicken auf den Boden (für Gruppe B auch auf die Karte) in der Welt „teleportieren“.

F16) Welche Perspektive hast du beim Teleport lieber genutzt? 1. 3.
Begründung:

_____
_____

F17) Welche Perspektive hast du beim freien Gehen lieber genutzt? 1. 3.
Begründung:

_____
_____

F18) Welche Navigation schien dir effizienter?

_____
_____

B12) Welche Perspektive wurde beim Teleport am Meisten benutzt (aus Log-Datei) 1. 3.

B13) Welche Perspektive wurde bei freiem Gehen am Meisten benutzt (aus Log-Datei)

1. 3.

Aufgabenbogen: inspect Gruppe A/B

A7) Gehe zur Statue. Schaue dir die Statue und die Kirche von allen Seiten genau an. Mit „p“ die Perspektive verändern.

B14) Welche Perspektive wurde am Meisten benutzt (aus Log-Datei) 1. 3.

B15) „F1“, um auf die Navigationsmöglichkeit „Path Orbit“ umzuschalten

A8) Du hast nun die Möglichkeit durch rechts und links Bewegung in einem festen Weg um das Objekt herumzu“fliegen“ und es zu betrachten. Versuche es.

F19) War dir „freies Gehen“ oder der „Orbiting“ um das Objekt lieber, um das Objekt genau anzuschauen?
Begründung:

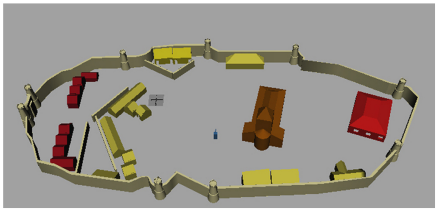
F20) Wäre es dir lieber die ganze Welt ausgeblendet zu haben, und nur das Objekt zu sehen? (Lenkt der Hintergrund ab?)
Begründung:

F21) Sollte das zu inspizierende Objekt lieber in einem extra Fenster angezeigt werden?
Begründung:

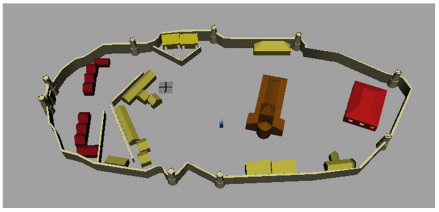
Wenn ja, soll der Hintergrund dann ausgeblendet werden, oder soll die Welt im Hintergrund zu sehen bleiben?
Begründung:

Explore-Aufgabe-Zusatz:

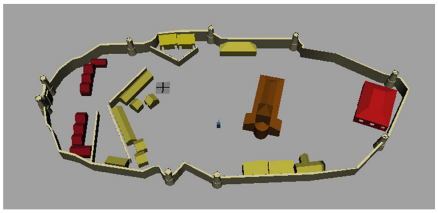
Stadtkarten



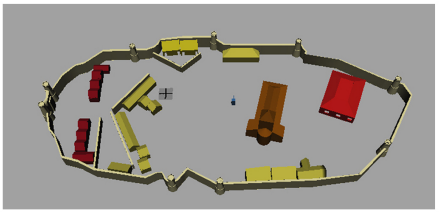
Stadt 1



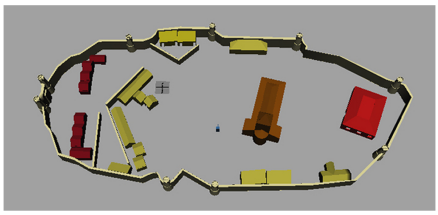
Stadt 2



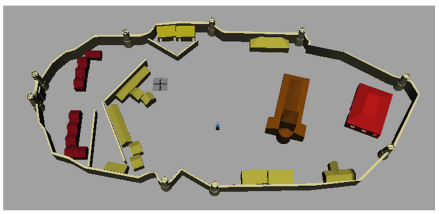
Stadt 3



Stadt 4



Stadt 5



Stadt 6

B. CD-ROM

Inhalt:

- *SetupNavigation.msi/.exe* : Installer, um die Testumgebung zu installieren
- Benutzertests
  - Log-Dateien (Gruppe A und Gruppe B)
  - Fragebogen als pdf-Datei
- Diplomarbeit : Diplomarbeit als pdf-Datei
- Source : *NavigationProjekt* : Quelldateien inkl. eigenen Meshes
- Quellen : Benutzte Quellen der Diplomarbeit