

# VR-Projektbuch

Sebastian Wendt & Christian Stussak

Projekte der Nebenfachausbildung  
*Designinformatik* bei Prof. Dr. Peter Kolbe

1. Oktober 2003 – 21. Juli 2009





# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>1</b>
1.1. Aufbau des Projektbuches . . . . .	1
1.2. Stichpunktartige Projektchronik . . . . .	2
1.3. Thematische Projektübersicht . . . . .	3
<b>2. Konzeption einer virtuellen Homeworld</b>	<b>5</b>
<b>3. Interaktion durch Bewegungsdetektion</b>	<b>7</b>
3.1. Einsatz bei Präsentationen . . . . .	7
3.2. Verwendetes Verfahren zur Bewegungsdetektion . . . . .	8
3.3. Prototyp zur Präsentation und Erkundung von 3D-Modellen . . . . .	11
3.3.1. Positionierung der Auswahlboxen auf dem Kamerabild . . . . .	11
3.3.2. Dateiformat der Modelle . . . . .	12
3.4. Fazit . . . . .	12
<b>4. Sechseck-Fraktale</b>	<b>15</b>
4.1. Bauanleitung für Fraktale . . . . .	15
4.2. Ergebnisse . . . . .	16
4.3. Beispieldatei farn.iv . . . . .	18
<b>5. Multiverse</b>	<b>19</b>
5.1. Konzeption der Welt . . . . .	19
5.2. Visualisierungskonzept . . . . .	21
5.2.1. Gespann-Metapher . . . . .	21
5.2.2. Kommunikationsverhalten . . . . .	22
5.2.3. Der Geisteravatar . . . . .	23

5.2.4. Steuerkonzept . . . . .	23
5.2.5. Feedback vom Tag der offenen Tür 2005 . . . . .	24
5.3. Technische Aspekte . . . . .	25
5.3.1. Client in Macromedia Director . . . . .	25
5.3.2. Server in OCaml . . . . .	25
5.3.3. Protokoll . . . . .	26
5.4. Billboard-Sprechblasen . . . . .	28
5.5. WebCamMiaw . . . . .	29
5.6. Base64Jpeg-Xtra . . . . .	30
<b>6. Ensembles Sphärischer Distributionsräume &amp; Navigation</b>	<b>31</b>
6.1. Kugeln mit Bebauungen . . . . .	31
6.2. Navigation auf Großkreisen . . . . .	32
6.3. Ensemble von Kugeln . . . . .	33
6.3.1. Auftretende Probleme . . . . .	34
6.4. Planetenlandung . . . . .	35
6.5. Kamerasteuerung . . . . .	35
6.5.1. Tastenbelegung der Kamerasteuerung . . . . .	36
<b>7. Polyeder als Trägergebilde</b>	<b>39</b>
7.1. Platonische Körper . . . . .	40
7.1.1. Ausgewählte Eigenschaften Platonischer Körper . . . . .	40
7.1.2. PowerPlaton . . . . .	41
7.2. Abgestumpftes Ikosidodekaeder . . . . .	42
7.2.1. Eigenschaften . . . . .	42
7.2.2. Prototyp in OpenInventor . . . . .	44
7.2.3. Prototyp in Director . . . . .	46
7.2.4. Selektionsexperimente . . . . .	48
7.2.5. Prototyp im Handlungszellenframework . . . . .	50
<b>8. Filtrator</b>	<b>55</b>
8.1. Vokabular . . . . .	55
8.2. Bedienung . . . . .	56
8.2.1. Schnittstelle mit dem HZFW . . . . .	56

8.2.2. FiltratorGuide-XML-Erweiterung . . . . .	57
8.3. Prototypen . . . . .	57
8.3.1. Schalenform-Prototyp . . . . .	58
8.3.2. Kegelform-Prototyp . . . . .	58
8.3.3. Ringform-Prototyp . . . . .	59
8.3.4. Rechteckform-Prototyp . . . . .	59
8.4. Menüleiste . . . . .	59
8.5. Programmdokumentation Filtrator . . . . .	59
8.6. Fazit . . . . .	60

<b>A. Quellenverzeichnis</b>	<b>61</b>
<b>B. Anhang</b>	<b>62</b>
B.1. Konzeption einer virtuellen Homeworld . . . . .	62
B.2. Ausgewählte Eigenschaften Platonischer Körper . . . . .	72
B.3. Programmdokumentation Filtrator . . . . .	77

# 1. Einleitung

Während des Hauptstudiums im Wahlpflichtfach Designinformatik wurde eine Vielzahl von Projekten und Prototypen zu verschiedenen Themengebieten bearbeitet. Eine zeitliche Auflistung der einzelnen Tätigkeiten findet sich in Abschnitt 1.2, während Abschnitt 1.3 thematische Zusammenhänge verdeutlicht.

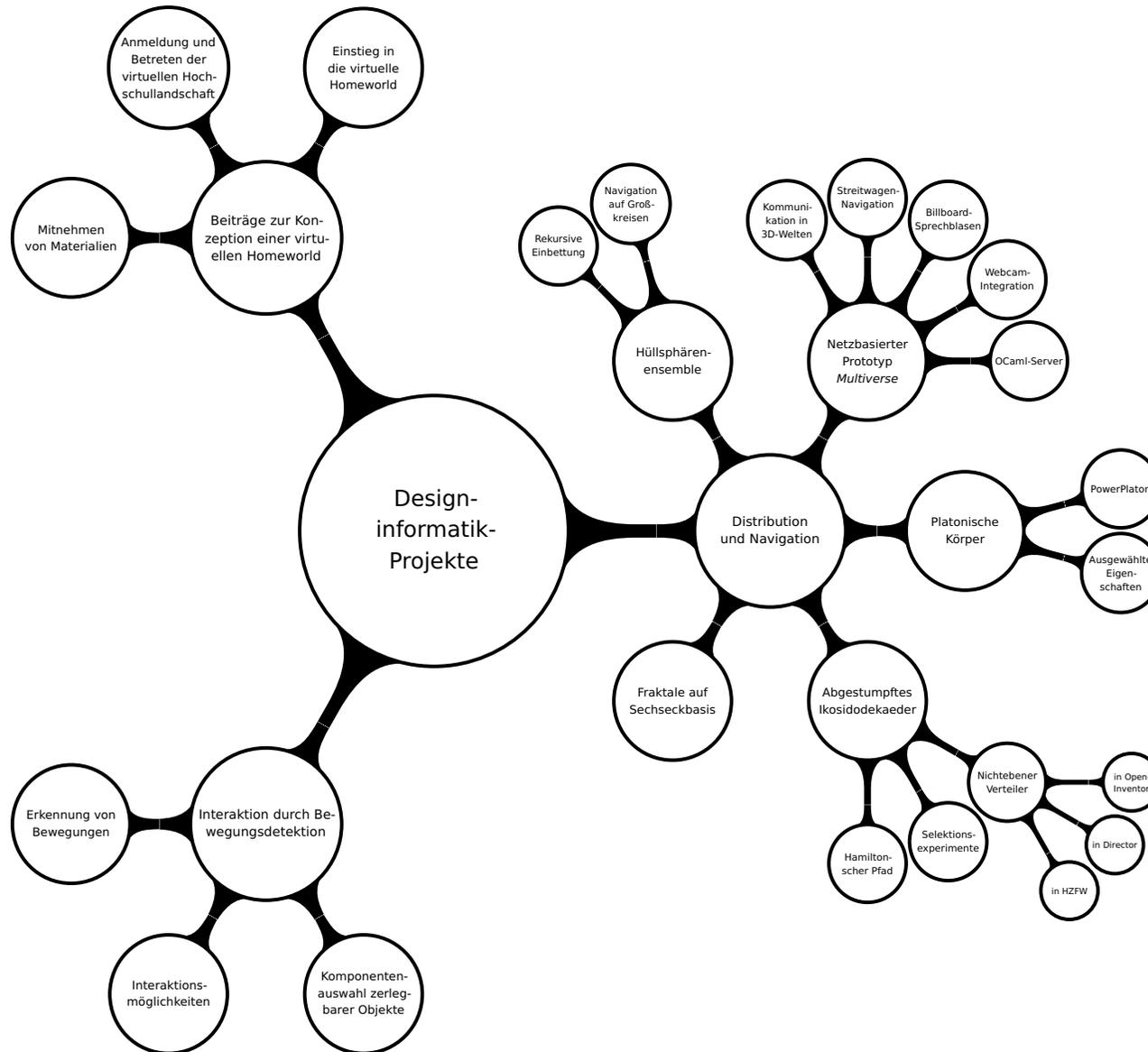
## 1.1. Aufbau des Projektbuches

Bei der Ausarbeitung des Projektbuches wurde versucht, auf die chronologische Abfolge der einzelnen Projekte Rücksicht zu nehmen. Dadurch entstand jedoch teilweise ein starker Themenwechsel zwischen den Kapiteln. Die letztendliche Gliederung ist zwar weitestgehend zeitlich geordnet, jedoch wurden die Projekte zur sphärischen Distribution und Navigation aufgrund des thematischen Zusammenhanges im hinteren Teil des Projektbuches mit den Projekten zur Polyeder-basierten Distribution und Navigation gruppiert.

## 1.2. Stichpunktartige Projektchronik

WS 03/04	<ul style="list-style-type: none"><li>• Beiträge zur Konzeption einer virtuellen Homeworld</li><li>• Interaktion durch Bewegungsdetektion (am Beispiel eines Komponentenfilters für die Mercedes A-Klasse)</li></ul>
SS 04	<ul style="list-style-type: none"><li>• Distribution mit Hilfe von und Navigation auf Ensembles von Hüllsphären</li><li>• Studien zur Steuerung des Systems mit Hilfe eines Joysticks und eines Datenhandschuhs</li><li>• sechseckbasierter Fraktal-Generator</li></ul>
WS 04/05	<ul style="list-style-type: none"><li>• Nutzung der Flächen von Polyedern als Distributionsraum</li><li>• Ausarbeitung zur Korrelation von bestimmten Eigenschaften Platonischer Körper</li><li>• <i>PowerPlaton</i>: Tool zur Generierung der 3D-Modelle Platonischer Körper mit bestimmten Eigenschaften</li><li>• Untersuchung des Archimedischen Körpers „Ikosidodekaederstumpf“ als Distributionsraum: Prototypen zur effizienten Navigation und Begehung (Hamiltonscher Pfad) in OpenInventor</li><li>• Portierung des Prototypen in Director und anschließende Integration in virtuelle Homeworld für den Internetauftritt des Studiengangs MM VR-Konzeption</li></ul>
SS 05	<ul style="list-style-type: none"><li>• weitere Experimente mit Ikosidodekaederstumpf: prototypische Implementierung eines Informationsfilters und Einbettung von Objekten/Subzellen mit entsprechender Drehung und Größenanpassung</li><li>• Entwicklung des netzbasierten Prototypen <i>Multiverse</i>: Studien zur effizienten mausgesteuerten Navigation und zur textbasierten Kommunikation in dreidimensionalen, virtuellen Mehrbenutzerszenarien</li></ul>
SS 07	<ul style="list-style-type: none"><li>• Integration des Ikosidodekaederstumpf-Distributionskörpers in das Handlungszellenframework</li><li>• Verfassen des Projektbuches</li></ul>

### 1.3. Thematische Projektübersicht





# 2. Konzeption einer virtuellen Homeworld

**Aufgabe** Es sollten verschiedene Aspekte einer virtuellen Homeworld konzipiert werden. Dazu zählten insbesondere der Einstieg in die virtuelle Homeworld, das Anmelden und Betreten einer virtuellen Hochschullandschaft und das Mitnehmen von Materialien beim Durchstreifen der virtuellen Homeworld.

**Bearbeitungszeitraum** Wintersemester 2003/2004

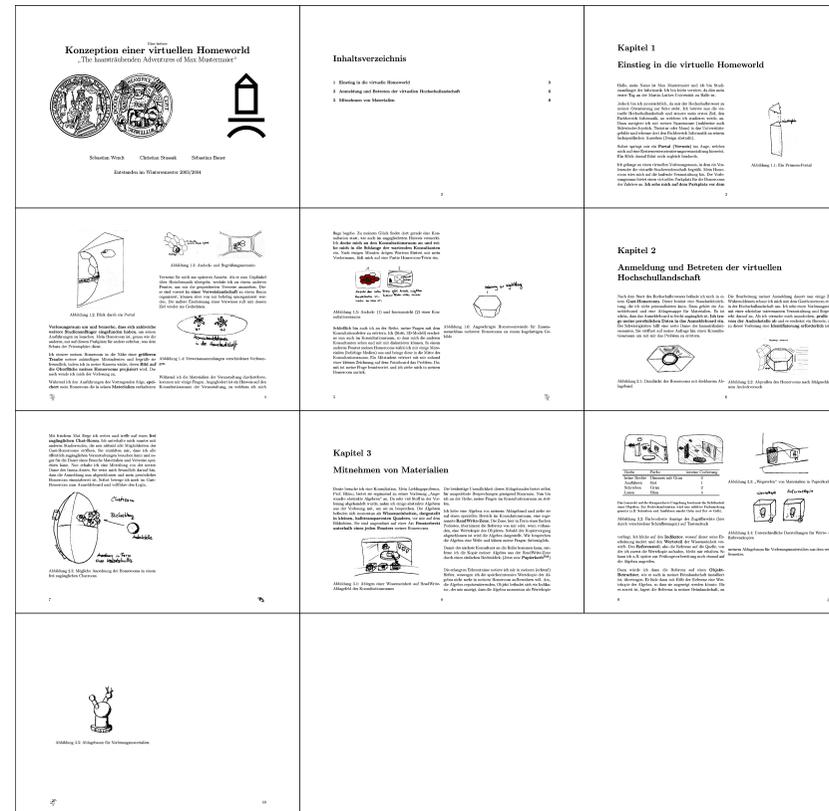
Die in wöchentlicher Verfeinerung, Erweiterung und Veränderung entstandene Konzeption ist im Anhang B.1 zu finden. Sie greift bereits einige interessante Konzepte auf, die inzwischen in die prototypische Entwicklung eingeflossen sind. Dazu gehören beispielsweise die Trennung von Körper- und Raumsichten, sowie in eingeschränkter Form die Unterstützung von szenischen Links durch Portale (Linkzellen).

**Ordner auf DVD**

Dokumente

**Systemvoraussetzungen**

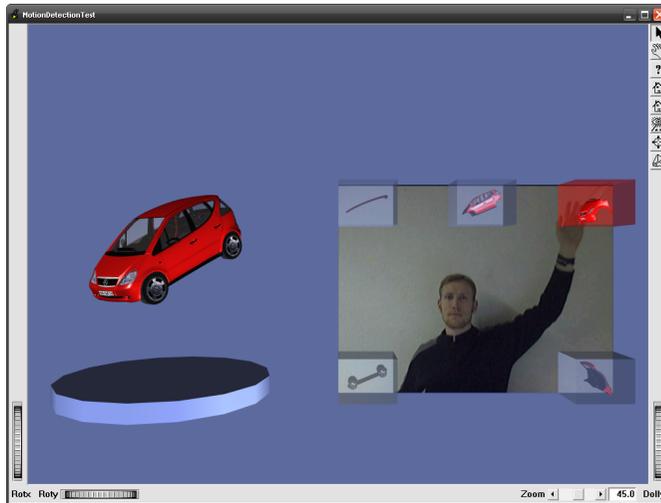
PDF-Betrachter



**Abbildung 2.1.:** Die ausgearbeitete Konzeption im Miniaturformat.



## 3. Interaktion durch Bewegungsdetektion



**Aufgabe** In OpenInventor sollte ein Prototyp entwickelt werden, der einfache Interaktionen zwischen Mensch und Computer durch Detektion von Bewegungen in einem Video, beispielsweise von einer Webcam, auslöst. Unter „einfachen Interaktionen“ war alles zu verstehen, was auch durch Klick auf einen Button oder Drücken einer Taste auf der Tastatur realisierbar wäre.

**Bearbeitungszeitraum** Wintersemester 2003/2004

### 3.1. Einsatz bei Präsentationen

Die Idee zu diesem Projekt entstand im Wesentlichen aus folgender Tatsache:

In Vorlesungen wird das Material heutzutage meist in Form von „Folien“ über einen Beamer präsentiert. Die Hauptinteraktion des Vortragenden mit dem Computer besteht hierbei im „Weiterschalten“ seiner Folien über Maus oder Tastatur.

Wird der Vortragende mit einer Videokamera aufgezeichnet und können seine Bewegungen effizient verfolgt werden, so wäre es möglich, das „Weiterschalten“ ohne direkten Kontakt zu einem Eingabegerät des Computers zu realisieren, nämlich durch bestimmte Bewegungen oder die Bewegung in vorher festgelegte Bereiche des Kamerabildes.

Im virtuellen Vorlesungsraum des Edutoriums sind darüberhinaus mehrere Anzeigetafeln vorhanden (Abbildung 3.1), wobei zwischen 1-, 2- und 3-Tafelmodus umgeschaltet werden kann. Es wäre auch denkbar, diese verschiedenen Modi durch Bewegungsdetektion zu steuern.

Als weiteres Einsatzgebiet ist die Präsentation und Erkundung von 3D-Modellen denkbar, indem der Benutzer einzelne Modellteile auswählt. Dies wurde als OpenInventor-Prototyp realisiert (siehe Abschnitt 3.3).



**Abbildung 3.1.:** Im Vorlesungsraum des Edutoriums könnte das Weiterschalten von Folien, als auch das Umschalten zwischen den verschiedenen Tafelmodi durch Detektion der Bewegungen des Vortragenden ausgelöst werden.

### 3.2. Verwendetes Verfahren zur Bewegungsdetektion

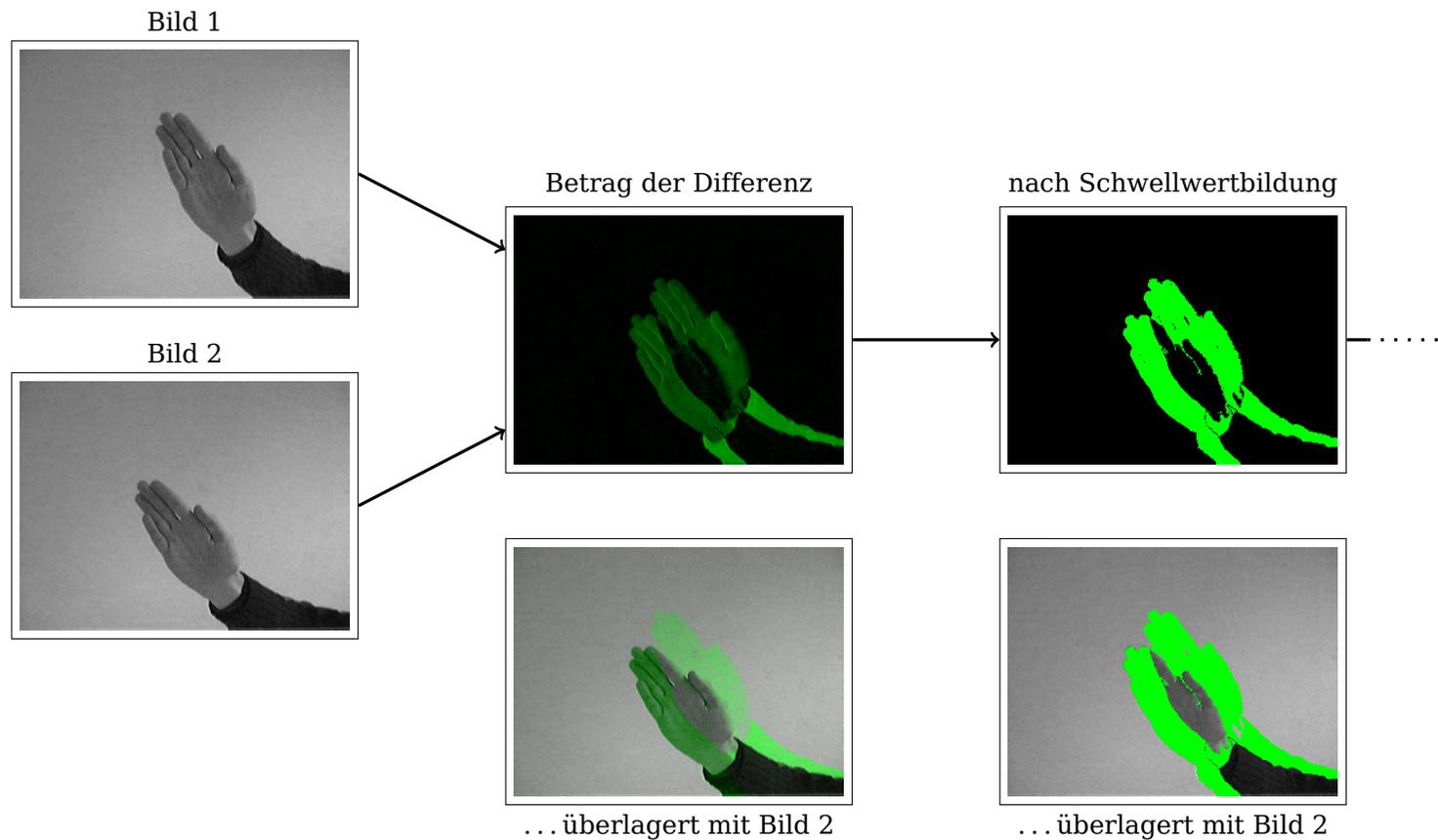
Ausgangspunkt der Detektion bildet eine Sequenz von Bildern, die in Echtzeit von einer Kamera geliefert wird. Bewegt sich ein Objekt vor der Kamera, so unterscheiden sich aufeinanderfolgende Bilder dieser Sequenz. Umgekehrt muss aber darauf geachtet werden, dass Effekte wie Bildrauschen oder kleine Änderungen der Lichtverhältnisse nicht als Bewegung interpretiert werden.

Häufig kommt das folgende, effizient umsetzbare Verfahren zum Einsatz (siehe auch Abbildung 3.2): Zwischen dem aktuellen und dem vorhergehenden Kamerabild wird pixelweise die Differenz gebildet, die die Veränderungen im Bild repräsentiert. Auf den Betrag der Differenzen wird eine Schwellwertfunktion angewendet, das heißt, dass allen Pixeln, deren Helligkeit einen gewissen Wert überschreitet, der Wert 1 (weiß) und allen anderen Pixeln der Wert 0 (schwarz) zugewiesen wird. Auf diese Weise wird versucht, die negativen Effekte, die durch

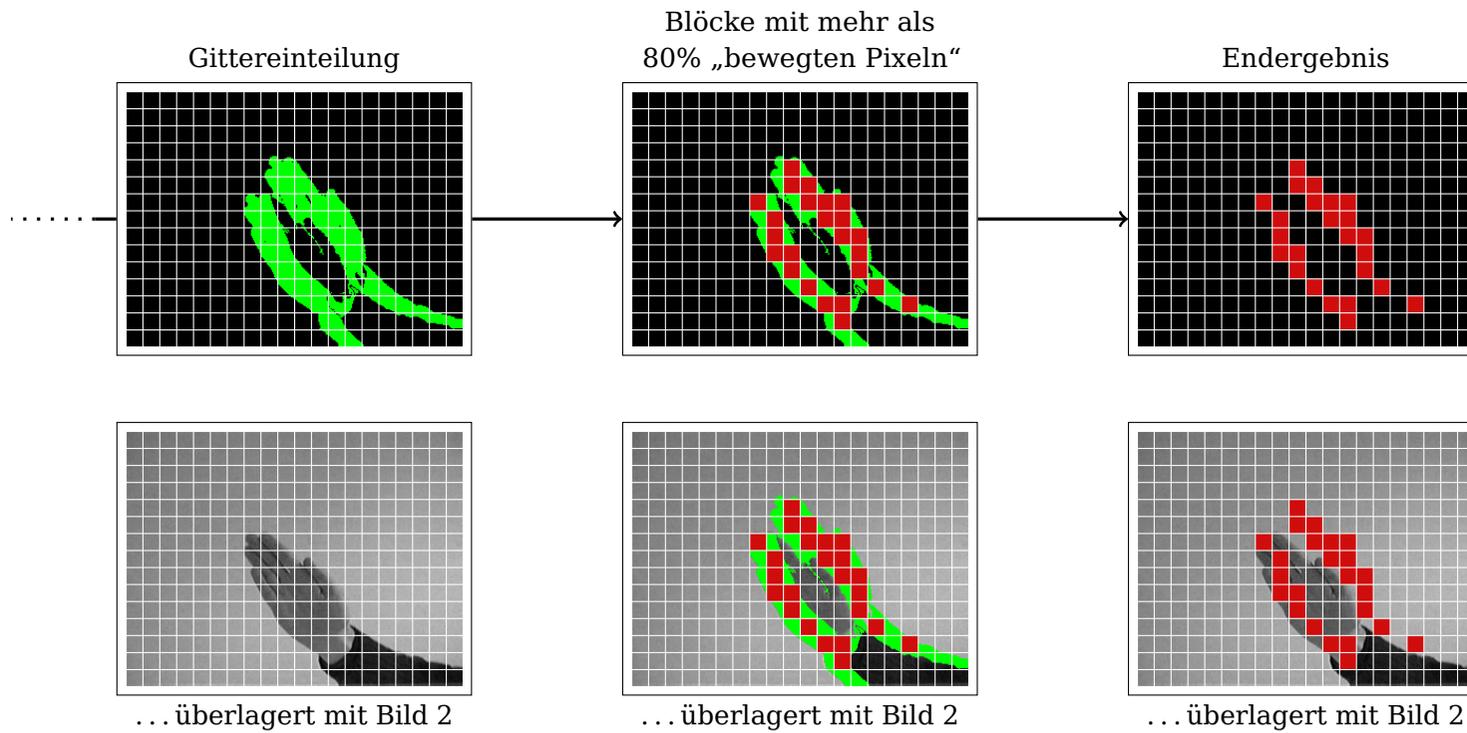
Bildrauschen oder Helligkeitsschwankungen entstehen, zu unterdrücken.

Bisher wurde noch nicht berücksichtigt, dass Kamerabilder meist im RGB-Format vorliegen. Eine Möglichkeit besteht darin, die Bilder vor der Bewegungsdetektion in Graustufenbilder zu konvertieren. Die weiteren Verarbeitungsschritte werden dadurch zwar beschleunigt, doch es entstehen Probleme, sobald sich Hintergrund und bewegtes Objekt hauptsächlich in der Farbe und kaum in der Helligkeit unterscheiden. Wird die Detektion für jeden Farbkanal separat durchgeführt und werden die Ergebnisse anschließend zusammengeführt, so steigt die Erkennungsrate auf Kosten der Geschwindigkeit. Aufgrund von Performanceproblemen wurde daher im OpenInventor-Prototypen ausschließlich auf Graustufenbildern gearbeitet.

Das durch die Schwellwertbildung erzeugte Schwarz-Weiß-Bild liefert die zum Vorgängerbild unterschiedlichen Pixel. Diese sollten aber nicht direkt als Bewegung interpretiert werden. Zum Beispiel ist ein einzelner weißer Pixel, in dessen näherer Umgebung nur schwarze Pixel vorhanden sind, wohl eher auf Bildrauschen zurückzuführen, als auf eine Bewegung. Daher untersucht man das Bild auf zusammenhängende oder benachbarte weiße Pixel, die ab einer gewissen Anzahl als Bewegung angesehen werden. Der in OpenInventor implementierte Ansatz (siehe Abbildung 3.3) teilt dazu das Schwarz-Weiß-Bild anhand eines regelmäßigen Gitters in Blöcke auf. Enthält ein Block mehr als einen bestimmten Prozentsatz (zum Beispiel 80%) an weißen Pixeln, so wird dies als Bewegung in diesem Block angesehen.



**Abbildung 3.2.:** Schritte zur Ermittlung der Pixel, die potentiell zu einer Bewegung gehören. Zum besseren Verständnis wurden die Zwischenergebnisse für die Überlagerung mit Bild 2 grün eingefärbt.



**Abbildung 3.3.:** Bestimmung der Blöcke, die letztendlich als Bewegung im Bild interpretiert werden.

### 3.3. Prototyp zur Präsentation und Erkundung von 3D-Modellen

Als Beispielanwendung für die Bewegungsdetektion wurde in OpenInventor ein Prototyp zur Besichtigung der Einzelteile beziehungsweise Komponenten von 3D-Modellen entwickelt. Die verwendete Szene besteht aus einem Podest, auf dem sich der aktuelle Präsentationsgegenstand dreht und einer Tafel, auf die der Videostream einer Webcam gemappt wird und an deren Rand verschiedene Boxen positioniert sind.

Die Boxen enthalten Teilkomponenten und gegebenenfalls die übergeordnete Komponente des auf dem Podest befindlichen Gegenstandes. Wird über mehrere aufeinanderfolgende Bilder des Videostreams hinweg eine ausreichend starke Bewegung in einer der Boxen festgestellt, so wird die zugehörige Komponente ausgewählt, auf dem Podest angezeigt und es werden die am Kamerabild sichtbaren Komponenten aktualisiert. Der Vorgang ist in Abbildung 3.4 illustriert.

#### 3.3.1. Positionierung der Auswahlboxen auf dem Kamerabild

Die dem Kamerabild überlagerten Boxen sollten sinnvollerweise am Rand des Bildes positioniert werden, um den Akteur nicht zu stark einzuschränken. Es hat sich herausgestellt, dass je nach Entfernung des Akteurs zur Kamera nicht mehr als 3 beziehungsweise 5 Boxen gleichzeitig angezeigt werden sollten, da es sonst häufig zum unbeabsichtigten Auswählen der falschen Box kommt (Abbildung 3.5).

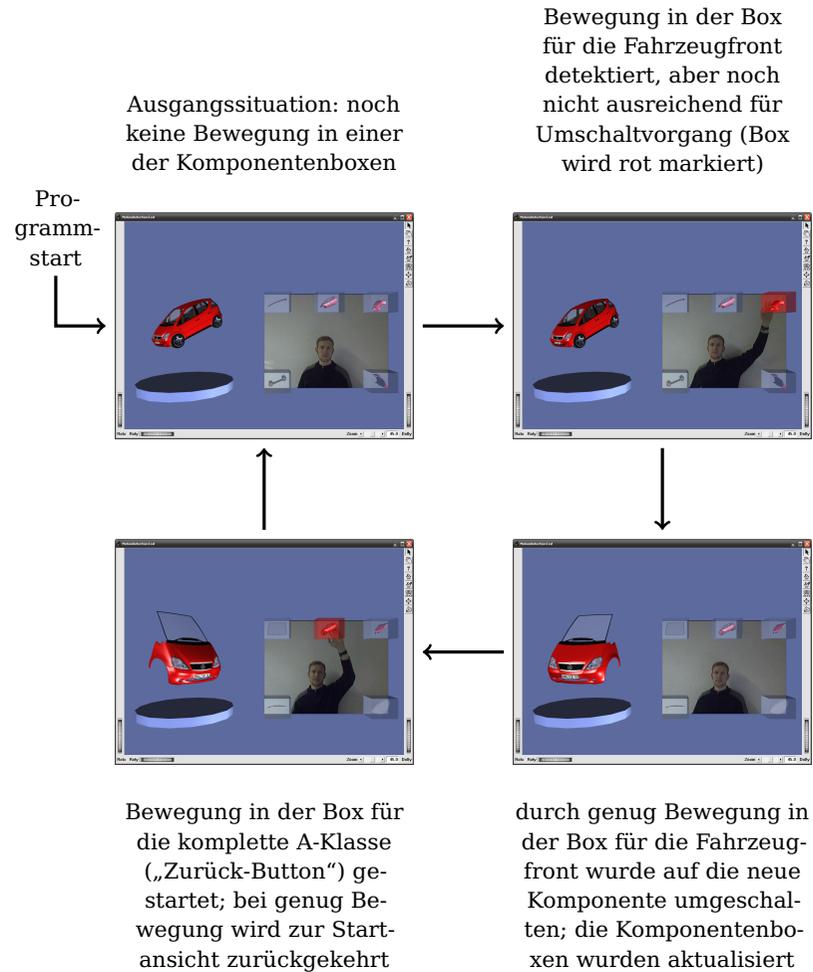


Abbildung 3.4.: Beispielhafter Programmablauf



sinnvolle Anzahl und Positionierung



zu hohe Anzahl oder ungünstige Positionierung kann zum unbeabsichtigten Auslösen der falschen Box führen (problematische Boxen in rot)

**Abbildung 3.5.:** Untersuchungen zur möglichen Anzahl und Anordnung der Auswahlboxen bei geringem und mittlerem Abstand zur Kamera.

### 3.3.2. Dateiformat der Modelle

Um ein 3D-Modell wie beschrieben zerlegen zu können, wurde ein OpenInventor-NodeKit namens `DecomposableObject` (=“zerlegbares Objekt“) entwickelt. Mit Hilfe dieses NodeKits wird die Hierarchie der Einzelkomponenten zusätzlich zum Szenegraph<sup>1</sup> festgelegt. Ferner enthält es die Positionen und Größen der Boxen auf dem Kamerabild. Die Werte müssen separat für die Nutzung einer Box zur Auswahl einer Unterkomponente (meist an den Seiten platziert) und für den Rücksprung zur Oberkomponente (meist oben mittig) angegeben werden. Die Nodekits können wie die meisten OpenInventor-Klassen in `iv`-Dateien beschrieben werden. Das zu verwendende Dateiformat lässt sich bei Bedarf leicht aus dem Beispiel der A-Klasse ablesen. Diese wird am Ende der Datei `d0AKlasse.iv` definiert.

### 3.4. Fazit

Die Erkennung von Bewegungen funktioniert mit dem verwendeten Algorithmus bereits verhältnismäßig gut. Probleme entstehen, sobald der Kontrast zwischen Hintergrund und bewegtem Objekt gering ist. Da aus Performancegründen nur mit Graustufenbildern gearbeitet wird, kann es bei ungünstigen Lichtverhältnissen dazu kommen, dass der typische Fall einer bewegten Hand vor einer hellen Tapete nicht erkannt wird. Am Algorithmus selbst kann aus heutiger Sicht mit den Methoden der digitalen Bildverarbeitung noch viel verbessert werden.

Unabhängig von der derzeitigen Erkennungsrate bietet die Programmsteuerung mit Hilfe der Bewegungsdetektion interessan-

<sup>1</sup>Dies ist notwendig, da der Szenegraph meist deutlich mehr Knoten enthält, als das Modell an *präsentationswürdigen* Komponenten besitzt.

te Möglichkeiten. Der OpenInventor-Prototyp zeigt, wie in hierarchischen, virtuell vergegenständlichten Strukturen effizient navigiert werden kann, solange sich auf einer Hierarchiestufe nur wenige Elemente („Autoteile“) befinden.

### **Ordner auf DVD**

Objektzerlegung durch MotionDetection

### **Systemvoraussetzungen**

Windows XP

OpenInventor 4.06

Webcam



## 4. Sechseck-Fraktale

**Aufgabe** Das Projekt entstand auf eigene Initiative, als die Arbeitsgruppe Designinformatik gerade sechseckige Prismen als Primitive entdeckt hatte, und die Möglichkeit untersuchte, auf Basis dieser Grundform komplexe Gebilde induktiv aufzubauen. Der Ansatz, Fraktale zur Erzeugung von Distributorobjekten zu verwenden, wurde gewählt, da aus einem früheren Projekt bereits Vorwissen im Umgang mit Fraktalen, den Möglichkeiten ihrer programmierpraktischen Umsetzung und im gestalterischen Entwurf, vorhanden war.

**Bearbeitungszeitraum** Sommersemester 2004

Das Programm erzeugt fraktale Strukturen auf Basis von OpenInventor-Beschreibungsdateien. Im Rahmen des Projekts waren dabei Sechseck-Fraktale von besonderem Interesse. Für den Betrieb notwendig ist lediglich der IV-Dateibetrachter, der um die beiden bereitgestellten Node-Typen *Generator* und *Pattern* erweitert ist.

*Pattern* bzw. Muster sind in der fraktalen Geometrie derjenige Teil einer Fraktalbeschreibung, der in sich selbst eingesetzt wird. *Generatoren* sind diejenigen Teile eines Musters, an denen diese Einsetzung stattfindet.

Mithilfe dieser einfachen Technik sind mannigfaltige Gebilde von fraktaler Schönheit mit nur wenigen Handgriffen aus einer gewöhnlichen Szenenbeschreibungsdatei erstellt. Im folgenden wird die Vorgehensweise erläutert.

### 4.1. Bauanleitung für Fraktale

Betrachten wir dazu den Aufbau von `farn.iv` (vollständige Definition in Abschnitt 4.3):

```
PatternNode {
    recursion_depth 8
    Separator {
        Transform { .. }
        Cylinder { .. }
    }
    Separator {
        Transform { .. }
        DEF GFarn GeneratorNode {
            substitute Cube { .. }
        }
    }
    .. # andere Blätter
}
```

Zuoberst befindet sich ein `PatternNode`, in dessen Rumpf sich die restliche Beschreibung befindet. Ein `PatternNode` charakterisiert den Gültigkeitsbereich der fraktalen Rekursion. Sein Inhalt wird rekursiv eingesetzt, an Stelle der Knoten vom Typ `GeneratorNode`.

Der Parameter `recursion_depth` gibt dabei an, auf der wievielten Verschachtelungsebene noch eine Einsetzung vorgenom-

men wird. Ein Wert von 0 gibt an, dass keine Einsetzung vorgenommen wird, ein Wert von 1 bewirkt einmaliges Einsetzen, usw. Dabei sind nur ganzzahlige Werte möglich.

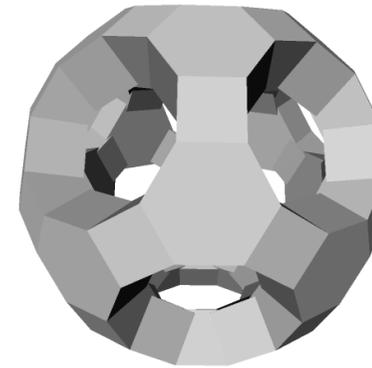
Der Parameter `substitute` des Knotens `GeneratorNode` schließlich gibt an, wodurch der Generator ersetzt werden soll, sobald die tiefste Rekursionsebene erreicht ist. Ein Primitiv mit einer Textur der rekursiven Fortsetzung ist dazu geeignet. Wird der Parameter ausgelassen, so wird ein Würfelprimitiv stattdessen eingesetzt. Sind nämlich beim Entwurf eines Fraktals außer Generatorknoten keine weiteren Geometrien im Muster angegeben, wäre die berechnete Struktur nicht sichtbar.

Ein verschachteltes Platzieren von Mustern wird nicht unterstützt, um den Berechnungsaufwand der rekursiven Funktionsaufrufe begrenzt zu halten. Es wird empfohlen, diese Fraktale gesondert zu berechnen, und das mithilfe der Exportfunktion abgespeicherte Muster in das übergeordnete zu laden. Beispielsweise empfiehlt es sich allein aus Komplexitätsüberlegungen, bei der Berechnung eines Baum-Fraktals, die Blätter-Fraktale an anderer Stelle vorzuberechnen.

## 4.2. Ergebnisse

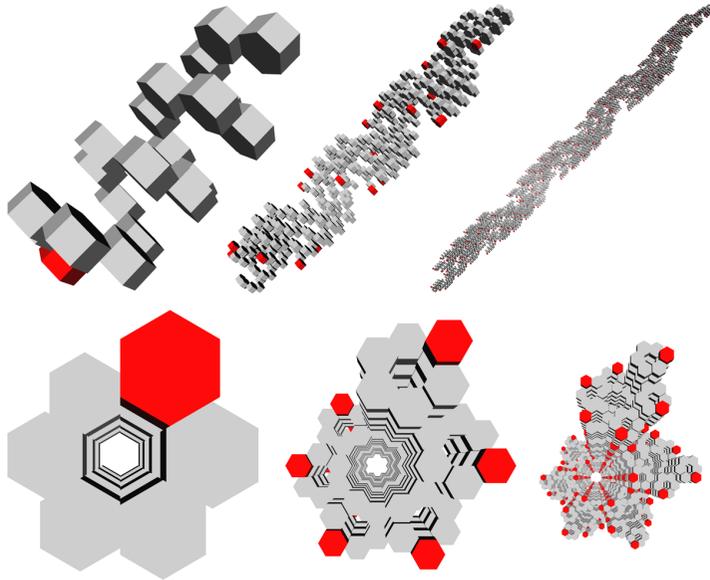
In Abbildung 4.2 sind einige der entwickelten Sechseckbasierten Fraktale zu sehen. Die Formen basieren auf relativ einfachen Rekursionsmustern, die in den Beschreibungsdateien auf dem Begleitdatenträger bereitgestellt sind.

Mit Sechsecken lassen sich also einige interessante Gebilde formen. Komplexe Kompositstrukturen aus mehreren Primitiven, wie zum Beispiel das abgestumpfte Ikosidodekaeder (Abbildung 4.1), enthalten ebenfalls sechseckige Seitenflächen und bieten

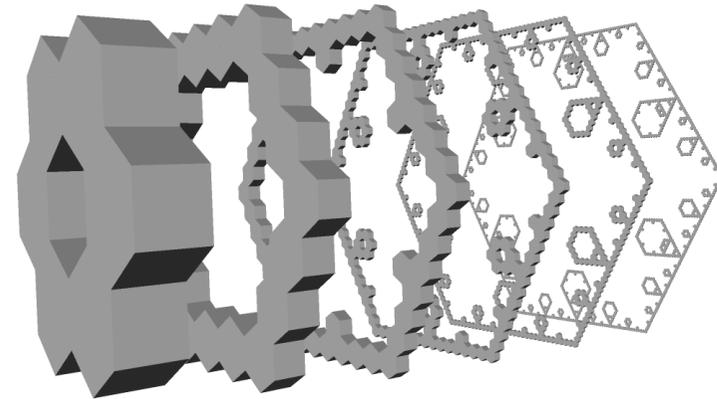


**Abbildung 4.1.:** mit dem Fraktal-Generator erzeugtes abgestumpftes Ikosidodekaeder

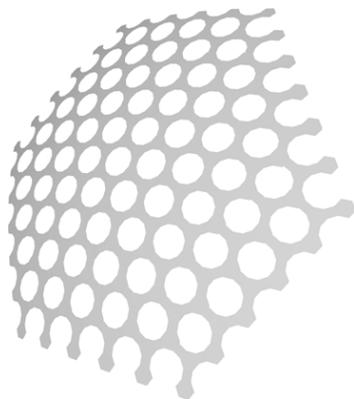
Möglichkeiten zur fortschrittlichen räumlichen Anordnung. Das abgestumpfte Ikosidodekaeder wurde in einem gesonderten Projekt untersucht (siehe Abschnitt 7.2).



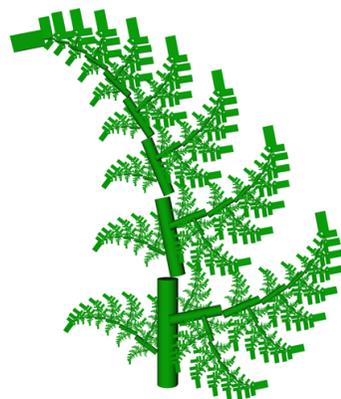
**(a)** Sechseck-Spiralen erster, zweiter und dritter Ordnung. Ansicht jeweils von der Seite und von oben.



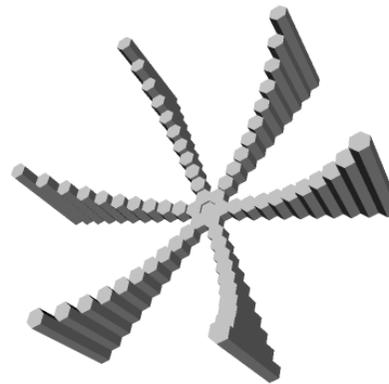
**(b)** Rekursionsstufen des Ringmusters



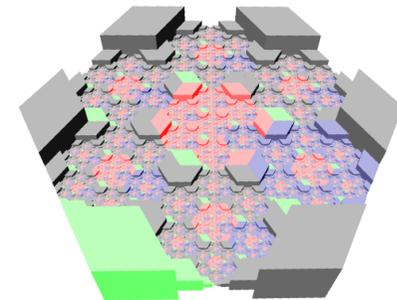
**(c)** Gitter aus 6- und 4-Ecken



**(d)** Farn



**(e)** sechseckige Parabel



**(f)** eingefärbte Schneeflockenunterteilung eines Sechsecks

**Abbildung 4.2.:** einige der generierten Gebilde

### 4.3. Beispieldatei farn.iv

```
#Inventor V2.1 ascii

Material { diffuseColor 0.0 0.6 0.0 }

PatternNode {
  recursion_depth 8
  Separator {
    Transform { translation 0.0 -0.75 0.0 }
    Cylinder { radius 0.05 height 0.5 }
  }
  Separator {
    Transform {
      scaleFactor 0.75 0.75 0.75
      translation 0.1 0.25 0.0
      rotation 0.1 0.0 1.0 -0.2
    }
    DEF GFarn GeneratorNode {
      substitute Cube {
        width 1.0
        height 2.0
        depth 0.1
      }
    }
  }
}
Separator {
  Transform {
    scaleFactor 0.5 0.5 0.5
    translation -0.5 -0.6 0.0
    rotation 0.1 0.0 1.0 1.4
  }
  USE GFarn
}
```

```
}
Separator {
  Transform {
    scaleFactor 0.25 0.25 0.25
    translation 0.25 -0.7 0.0
    rotation 0.1 0.0 1.0 -0.9
  }
  USE GFarn
}
}
```

#### Ordner auf DVD

Sechseck-Fraktal

#### Systemvoraussetzungen

Windows XP

OpenInventor 5.04

## 5. Multiverse



**Aufgabe** Entwurf und Implementierung eines Prototypen zur netzbasierten Kommunikation und Navigation in dreidimensionalen virtuellen Welten. Die Präsentation und das Feedback von den Benutzern zum Tag der offenen Tür der Burg Giebichenstein Hochschule für Kunst und Design 2005 standen im Mittelpunkt.

**Bearbeitungszeitraum** Juni 2005

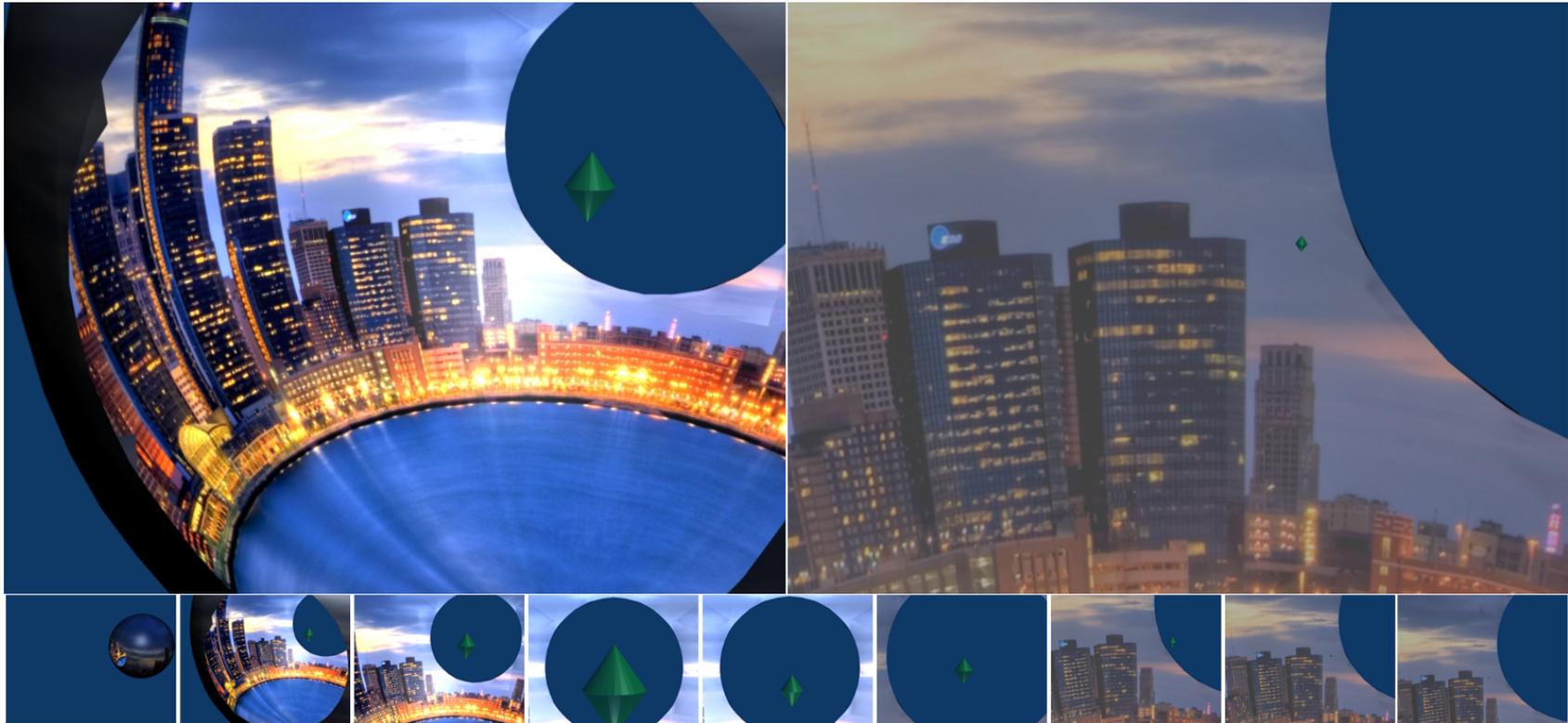
Das Projekt entstand auf eigene Initiative, um das in der Arbeitsgruppe Designinformatik zuvor praktisch unerprobte Potential von Interaktionsszenarien im Netzwerk zu erkunden, und Erfahrungen in Bezug auf mögliche Interaktionen zwischen den

Benutzern des virtuellen Szenarios zu sammeln. Im knapp bemessenen Bearbeitungszeitraum von nur 3 Wochen wurde Wert gelegt, sowohl auf die technischen Aspekte der Netzwerkkommunikation, als auch auf die Akzeptanz der angebotenen Interaktionsmittel durch die Benutzer. Konzeption, Programmierung und Präsentation des von Grund auf entwickelten Prototypen geschahen eigenverantwortlich.

### 5.1. Konzeption der Welt

Startet man das Programm, so befindet man sich nach erfolgreicher Anmeldung in der Welt des Multiverse. Die sichtbare ist nur eine von vielen möglichen Welten, die parallel im Multiversum existieren. Sie sind über so genannte Übergänge untereinander verbunden. Stellvertreten durch den Avatar kann man zur Erkundung der Welt aufbrechen. Näheres dazu im Abschnitt *Steuerkonzept*. Da der Prototyp die Netzwerkfähigkeit hervorhebt, wird man auf andere Benutzer treffen und mit diesen kommunizieren wollen. Näheres dazu im Abschnitt *Interaktion*. Zunächst jedoch eine kurze Beschreibung der Welt selbst.

Die Welt bzw. Welten sind angefüllt mit Objekten, die eine gemeinsame logische Grundstruktur aufweisen. Avatare und Übergänge sind ebenfalls solche Grundobjekte, auf die die gleichen Operationen wie auf alle Objekte anwendbar sind. Die Operationen (Kommandos) sind ausführlich im Abschnitt 5.3.3 erläutert.



**Abbildung 5.1.:** Wird der Übergang (Doppelkegel) aktiviert, so bewegt nicht der Avatar sich in die neue Welt, sondern diese baut sich um ihn herum auf. Die dabei stattfindende Skalierung vergrößert die vorherige Welt soweit, dass sie nach außerhalb des Sicht- und Wirkungsbereichs des Avatars verschoben wird.

Im Prototypen sind die Übergänge einfache Doppelkegel, die durch Anklicken aktiviert werden, und den Benutzer in die Zielwelt transportieren.

Die Konzeption der Übergänge sah vor, diese mit einer Vorschau auf das jeweilige Ziel auszustatten, welche die Form einer begehbaren Vorschau-Kugel haben sollte. Der Zielort würde sphärisch in die Kugel projiziert, die der Benutzer betreten muss, um den Übergang zu aktivieren. Bei Aktivierung würde die Projektion durch die tatsächliche Umgebung der Zielwelt ersetzt und die Welt außerhalb der Kugel könnte entfernt werden. Damit ist der Übergang vollzogen.

Der Prozess ist in Abbildung 5.1 visualisiert und auch als Animation auf dem Begleitdatenträger verfügbar.

## 5.2. Visualisierungskonzept

Das zentrale Element der Benutzeroberfläche ist der Avatar. Das auf seine Vorderseite projizierte Bild des Benutzers verleiht dem Avatar eine Identität. Es wird per Webcam aufgenommen und bei der Anmeldung an den Server gesendet<sup>1</sup>. Der Avatar verrät zudem, wohin der Blick beziehungsweise die Kamera des Benutzers zur Zeit gerichtet ist, so dass der Blickkontakt, der bei zwischenmenschlicher Kommunikation so wichtig ist, auch im Virtuellen vermittelt werden kann. Durch die Art der Kamerasteuerung behält der Benutzer dennoch eine gewisse Distanz für sich.

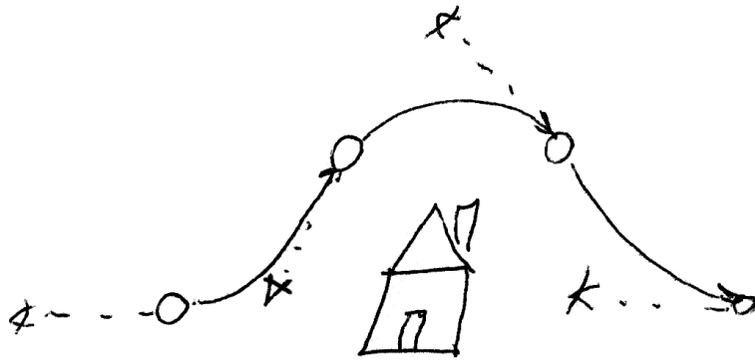
<sup>1</sup>aus Zeitgründen leider nicht fertiggestellt



**Abbildung 5.2.:** Avatar mit aufgeklebtem Webcambild (das Mapping ist im aktuellen Prototypen nicht umgesetzt).

### 5.2.1. Gespann-Metapher

Die Kamera befindet sich nicht am gleichen Punkt wie der Avatar (1.Person-Perspektive), sondern in einigem Abstand dahinter (3.Person-Perspektive). So wird das Gefühl des nicht unmittelbaren ausgesetzt-Seins, wie in einem Kinofilm, vermittelt. Der Benutzer kann die Kamera und den Avatar unabhängig voneinander steuern. Bewegt er den Avatar, so folgt ihm die Kamera in einem dynamischen, aber begrenzten Abstand. Richtet er seinen Blick bzw. die Kamera auf einen anderen Benutzer oder ein Objekt, so dreht sich auch der Avatar in die entsprechende Richtung. Diese Art der Navigation wurde unter dem Namen *Streitwagen-Navigation* bzw. *Gespann-Metapher* bekannt (Abbildung 5.3). Ganz wie der Mensch auch seine Augen bzw. seinen Kopf bewegen kann, ohne den Körper zu drehen, während er sich umsieht, hat auch der Benutzer die Möglichkeit, die Kamera um den Avatar frei zu rotieren, ohne dass dies für andere Benutzer in seiner Umgebung sichtbar wird. Die Fähigkeit, aus der Perspektive der dritten Person seine eigene Erscheinung zu inspizieren und gezielt korrigieren zu können, verleiht dem Benutzer ein Gefühl der Sicherheit bei der Interaktion mit



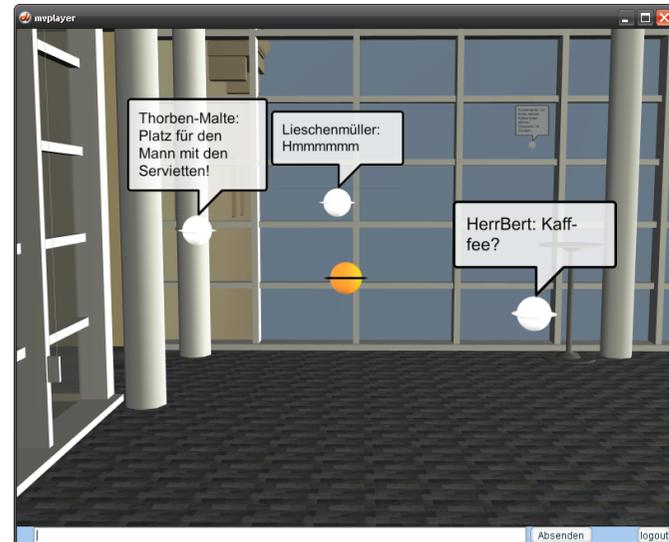
**Abbildung 5.3.:** Originalskizze zur Streitwagen-Navigation.

anderen Benutzern.

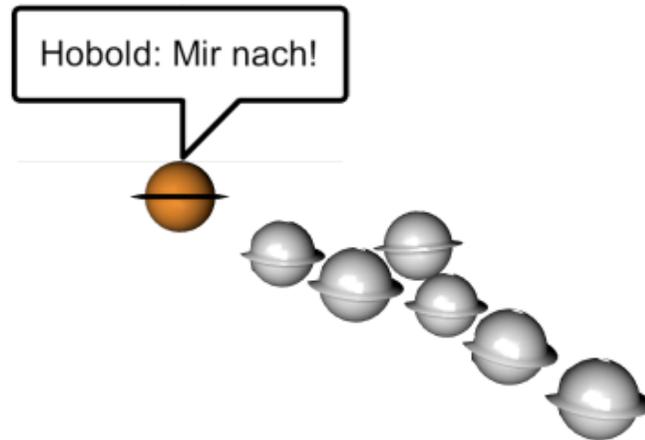
### 5.2.2. Kommunikationsverhalten

Durch den Einsatz comichafter Sprechblasen (siehe Abschnitt 5.4), ist es möglich Textkommunikation nahtlos in die 3D-Umgebung einzubetten. Da die Sichtweite begrenzt ist, entsteht ein Effekt der Ballung von Gesprächspartnern in der 3D-Welt (Abbildung 5.4), anstatt einer ungeordneten, mehrere Gespräche durcheinanderwerfenden Form, wie sie beispielsweise ein parallel laufender Chatroom bereitet hätte. Ein Wechseln des Aufmerksamkeitsfokus zwischen verschiedenen Interaktionsebenen (3D-Welt, 1D-Text) ist nicht notwendig. Die Integration ermöglicht zudem eine semiotische Bindung des Themas an eine dreidimensionale Umgebung, das heißt Chat-Rooms im eigentlichen Sinne des Raumbegriffs.

Der Benutzer ist durch die räumliche Nähe zu anderen Gesprächspartnern ebenfalls in der Lage, explizit Aufmerksamkeit oder Desinteresse zu bekunden, indem er seinen Avatar zu einem Gesprächspartner ausrichtet (Anschauen des Spre-



**Abbildung 5.4.:** Durch die begrenzte Sichtweite der Sprechblasen müssen sich Gesprächsteilnehmer, wie auch im realen Leben, räumlich nahe stehen. Die Texte in den Sprechblasen entfernter Avatare sind in der Regel nicht erkennbar (wie bei dem Avatar links oben).



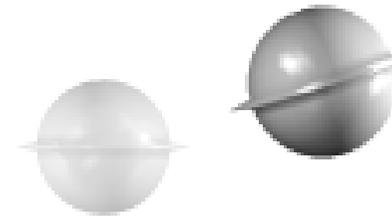
**Abbildung 5.5.:** Schwarmbildung bei virtuellen Führungen.

chenden) oder wegschaut beziehungsweise sich vom Gespräch entfernt. Ein interessanter Effekt ist auch bei virtuellen Führungen zu beobachten. Durch die Ballung der Avatare um den „Reiseleiter“ entsteht beim Flug zwischen verschiedenen Stationen der Führung eine Art Schwarm, der die Führung schon von weitem als solche erkennbar macht (Abbildung 5.5).

### 5.2.3. Der Geisteravatar

Um durch Netzwerkverzögerung entstehende Aussetzer und Sprünge in der Darstellung zu vermeiden, wurden zwei verschiedene Stellvertreter des Benutzers eingeführt. Der erste, lokale Avatar, reagiert unmittelbar auf die Steuerbefehle des Benutzers. Er ist in transparenter Ausführung dargestellt, woher der Name *Geisteravatar* entstammt.

Der eigentliche Avatar wird durch den Server gesteuert - angeregt durch Befehle des Clients an den Server - und hat für



**Abbildung 5.6.:** Lokaler *Geisteravatar* und echter, von allen Teilnehmern gleich gesehener Avatar, der aufgrund von Netzwerkverzögerungen dem *Geisteravatar* hinterherhinkt.

alle angeschlossenen Clients dieselbe Darstellung (Sprechblase, Position, Rotation, ...). Der Geisteravatar ist nur für den jeweiligen Benutzer sichtbar und zeigt die intendierte Position an. So entstehen keine Irritationen durch Verzögerungszeiten und gleichzeitig ist eine Vergewisserung über das von anderen gesehene Aussehen des eigenen Avatars möglich (Abbildung 5.6).

### 5.2.4. Steuerkonzept

Bei der Entwicklung des Steuerungskonzepts, stand, inspiriert von kommerziellen Spieleprodukten, die handelsübliche 2-Tastenmaus ohne Scrollrad im Mittelpunkt. Die Tastatur wurde hinzugenommen, lediglich um Textnachrichten in das Feld am unteren Bildschirmrand einzugeben.

Hinsichtlich der Präsentation zum Tag der offenen Tür war diese Beschränkung auch maßgebend, da der Client so auf beliebigen PCs präsentiert werden konnte.

Die linke Maustaste dient zur Umsicht und zum Aktivieren von Szenenobjekten, die rechte Maustaste steuert die Geschwindig-

keit der Vorwärts- und Seitwärtsbewegung.

## **Orientierung**

Bei gedrückter linker Maustaste ist es möglich, die Kamera um den Avatar im Mittelpunkt frei zu schwenken. Ein Winkelconstraint verhindert das Überschreiten der „Pole“, also der y-Achse über und unter dem Avatar. Während des Umsehens bleibt der Avatar unverändert, es werden keine Befehle an den Server gesandt. In einer Kommunikationssituation zieht der Benutzer keine ungewollte Aufmerksamkeit auf sein neugieriges Herumgucken und bleibt seinen Partnern zugewandt. Erst bei Loslassen der linken Maustaste wird der Avatar mit Blick nach vorn, entlang der neuen Kameraposition ausgerichtet.

Die linke Maustaste ist daher „ungefährlich“, da lediglich die clientseitige Kamera - für die anderen Teilnehmer unsichtbar - gesteuert wird.

## **Bewegung**

Durch Gedrückthalten der rechten Maustaste ist Beschleunigen/Abbremsen/Rückwärtsfliegen möglich. Der Cursor wird zunächst auf den Avatar zentriert und die Maustaste gedrückt. Durch Bewegen des Cursors in die obere Bildschirmhälfte wird eine Vorwärtsbeschleunigung eingeleitet. Durch Zurückbewegen auf die Mitte oder Loslassen der Maustaste wird die Beschleunigung gestoppt. Rückwärtsbeschleunigung wird gleichsam durch Ziehen der Maus in die untere Bildschirmhälfte erreicht. Die Bewegung des Avatars ist auf die x-z-Ebene begrenzt, das heißt er verändert seinen Horizont nicht. Der gewünschte Horizont kann, wie zuvor beschrieben, durch Schwenken der

Kamera mit gedrückter linker Maustaste eingestellt werden. Zusätzlich kann bei gedrückter rechter Maustaste das „Seitenruder“ gesteuert werden, indem die Maus bei gedrückter rechter Maustaste vorsichtig nach links bzw. rechts bewegt wird. Ein umständlicher Wechsel zwischen linker und rechter Maustaste ist für kleine Kurskorrekturen nicht erforderlich.

## **Interaktion**

Um schließlich mit der Welt auch interagieren zu können, kann durch einen Klick der linken Maustaste, ohne diese gedrückt zu halten, ein Objekt aktiviert werden. Die Art der Aktion ist serverseitig für jedes Objekt gespeichert.

Im Prototypen wird die Aktion dazu benutzt, um Übergänge zwischen Welten, dargestellt durch doppelkegelförmige Bögen, zu aktivieren.

Diese relative einfache Interaktionsmöglichkeit stellt keine prinzipielle Beschränkung verschiedener Aktionen dar. Der Server hat beispielsweise die Möglichkeit, bei Aktivieren eines Objektes, den Client anzuweisen ein Steuermenü zu erzeugen, das aus mehreren Unterobjekten besteht. Diese Unterobjekte können verschiedene, weiter verfeinerte Aktionen auslösen und nach Gebrauch per Anweisung vom Server wieder vernichtet werden.

### **5.2.5. Feedback vom Tag der offenen Tür 2005**

Durch Gespräche mit den Gästen und bereitwilligen Testern am Tag der offenen Tür konnten zahlreiche konstruktive Vorschläge zur Verbesserung gewonnen werden. Obwohl die Navigation gut durchdacht war, gab es hier zahlreiche Schwierigkeiten, da

sie eine Umgewöhnung von vertrauten Steuerkonzepten in 3D erforderte oder insgesamt ein Umdenken bei Bewegungsmustern mit der Maus bedeutete.

Kritisiert wurde der mangelnde Realismus, da es möglich ist, Wände zu durchfliegen. Dies hatte oftmals einen Orientierungsverlust zur Folge. Eine Kollisionsabfrage mit der Umgebung erfordert jedoch einen komplexeren Algorithmus zur sinnvollen Positionierung der Kamera in schmalen Gängen und anderen Innenbereichen der Szene.

Wünschenswert war auch eine flexiblere Kamerasteuerung, die es beispielsweise erlaubt, dem Avatar über die Schulter zu schauen, anstatt durch ihn die freie Sicht auf das Zielobjekt zu behindern.

Die Anregung, ein Fadenkreuz in der Bildschirmmitte zu platzieren, sowie die Bewegung der Kamera direkt an die der Maus zu koppeln, ohne einen Tastendruck zu fordern, wie es etwa in kommerziellen Spieleprodukten der Fall ist, wurde aufgenommen.

Trotz Integration einer alternativen Netzbibliothek<sup>2</sup> am Abend des ersten Präsentationstages, blieb die als „grottenlangsam“ kritisierte Übertragungsgeschwindigkeit ein Problem. Interessanterweise konnten aufgrund der Performanceschwäche der Director-Skriptsprache die erhofften Geschwindigkeitsvorteile nicht zutage treten.

---

<sup>2</sup>Wechsel vom XtraNet- zum Multiuser-Xtra

## 5.3. Technische Aspekte

### 5.3.1. Client in Macromedia Director

Der Client wurde in Macromedia Director entwickelt, da dieses über die Möglichkeit zur Darstellung von 3D-Szenarien verfügt. Durch das weitverbreitete Shockwave-Plugin ist auch eine Einbettung in Internetseiten möglich. Für die Netzwerkanbindung wurde anfangs das XtraNet-Xtra verwendet, da dieses für den anvisierten Einsatz im Internet vielversprechend erschien. Später wurde es durch das weniger fehleranfällige Multiuser-Xtra ersetzt. Der Client ist intern in eine Darstellungs- und eine Steuerkomponente, wie im MVC-Design [GHJV04, S. 5ff] beschrieben, getrennt.

Die Steuerungskomponente empfängt Befehle (Maus, Tastatur, Webcam) vom Benutzer und leitet diese Daten an den Server weiter. Die Darstellungskomponente empfängt Befehle vom Server und führt, entsprechend diesen, Änderungen an der 3D-Visualisierung durch. Um den Übertragungsaufwand einzugrenzen, führt der Client bereits einige Vorberechnungen auf den Benutzereingaben durch, so dass der Server die Welt Darstellung in einem abstrakteren, nicht-visuellen Modell behandeln kann.

### 5.3.2. Server in OCaml

Bei der Entwicklung des Servers stand im Vordergrund, innerhalb der 3 Wochen bis zum Präsentationstermin, ein System mit klar definierten Aufgaben schnell und ohne aufwendiges Debugging zu produzieren. OCaml wurde als Implementierungssprache gewählt, da es über die notwendigen Bibliotheken zur

```

D:\UserDaten\sean\desi-projekte\Incoming\server1707.exe
building world from a handful of nothing
2007.9.21.20:24:53:notice:Kurova listening to 0.0.0.0:1607
2007.9.21.20:25:13:debug:Client connecting from 127.0.0.1:4974
2007.9.21.20:25:13:debug:avatar Hobold restored from server
2007.9.21.20:25:13:debug:mz rendered
2007.9.21.20:25:29:debug:Hobold picked entity mz_innenhof
2007.9.21.20:25:40:debug:Hobold picked entity gate1_buoy_lover
2007.9.21.20:25:40:debug:gate buoy triggered
2007.9.21.20:27:13:debug:Hobold picked entity radio_relay
2007.9.21.20:31:37:debug:Client connecting from 127.0.0.1:1352
2007.9.21.20:31:37:debug:avatar Max Mustermann restored from server
2007.9.21.20:31:37:debug:mz rendered
2007.9.21.20:31:44:debug:Max Mustermann picked entity mz_innenhof
2007.9.21.20:31:49:debug:Max Mustermann picked entity mz_innenhof
2007.9.21.20:31:55:debug:Max Mustermann picked entity gate1_buoy_lover
2007.9.21.20:31:55:debug:gate buoy triggered
2007.9.21.20:31:58:debug:rendering ava Hobold
2007.9.21.20:34:10:debug:Max Mustermann picked entity radio_relay
2007.9.21.20:34:10:notice:logging out avatar Max Mustermann
2007.9.21.20:34:10:ERROR:Uncaught exception in xnet channel parser: Sys_error("Bad file descriptor")
2007.9.21.20:34:10:debug:writing thread closed
2007.9.21.20:34:10:notice:logging out avatar Max Mustermann
2007.9.21.20:34:10:debug:parser terminated

```

Abbildung 5.7.: OCaml-Server des Multiverse.

Entwicklung eines Kommunikationsprotokolls verfügte und als stark typisierte funktionale Sprache mit Typinferenzmechanismus keine Probleme mit Speicherzugriffsfehlern kennt. Dass eine Übersetzung in Maschinencode möglich ist, ist von Vorteil für jede Mehrbenutzeranwendung, die auf Performanz Wert legt.

Die hauptsächlichen Strukturen des Servers sind ein Parser für das Kommunikationsprotokoll (siehe Anhang), Puffer für die parallelisierte Kommunikation mit den angemeldeten Clients, sowie einem Teilsystem für Anmeldung- und Login-Verwaltung. Benutzerkonten enthalten den Zustand des Avatars beim Verlassen der Welt. Ein mehrstufiges Nachrichtendistributor-Konstrukt verwaltet die Teilwelten (intern „scaffolds“ genannt) innerhalb des Servers, die Objekte und deren Unterobjekte in den Teilwelten, sowie deren Interaktionen untereinander, zum Beispiel den Wechsel von Avataren zwischen den Welten. Wie bereits erwähnt, besteht serverseitig kein Unterschied zwischen „unbelebten“ Objekten, wie Häusern und Hintergründen, und Avataren. Der oben erläuterte Geisteravatar-Effekt ist ein Resultat dieses verallgemeinerten Objektkonzepts. Umgekehrt bedeutet dies, dass eine Impersonisation, also das Anschließen der Cli-

entsteuerung an ein Objekt, prinzipiell für alle Objekte möglich ist.

Der Server fungiert als Proxy für die angeschlossenen Clients. Nachrichten werden in einem sternförmigen Netzwerk ausgetauscht. Kopien einer Nachricht von einem Client sendet der Server nur an die Clients im selben Distributor weiter. Außerdem kann er Nachrichten filtern oder selbst Ereignisse erzeugen, um zum Beispiel die Kontrolle über einen Avatar zu übernehmen und mit den Benutzern zu interagieren. Mithilfe der Billboard-Sprechblasen (Abschnitt 5.4) ist es daher auch möglich, wichtige Nachrichten und Hinweise durch Impersonisation von unbelebten Objekten an die Benutzer in der Umgebung des impersonierten Objektes weiterzugeben.

### 5.3.3. Protokoll

Das Kommunikationsprotokoll ist ASCII-basiert und wurde für die Verwendung mit dem XtraNet-Xtra entworfen. Es besteht aus einem getrennten Vokabular für Nachrichten des Clients an den Server und für Befehle des Servers an den Client. Der Client besteht gemäß des MVC-Designkonzepts<sup>3</sup> aus getrennten Komponenten für das Entgegennehmen von Benutzereingaben und für die Darstellung der Welt.

Die Darstellung der Welt wird vom Server übermittelt, in Form einer Sequenz von Kommandos zum Erzeugen und Bewegen von Objekten u.a. Die Benutzereingaben können in Form von Maus- oder Tastatureingaben erfolgen. Die Mauseingaben werden bereits in einem lokalen Kameramodell in Transformationen umgewandelt und dem Server für die Berechnung des Welt-

<sup>3</sup>Model-View-Controller: Trennung von Logik, Darstellung und Steuerkomponenten

modells übermittelt. Der Server führt Berechnungen auf den übermittelten Anfragen durch und verteilt Antworten an alle Clients.

### Client -> Server

**LOGIN user pass** meldet den Client am Server an und lädt die Benutzerdaten (Aussehen des Avatars, letzte Position in der Welt).

**LOGOUT** meldet den Client ab und speichert die Benutzerdaten.

**TRANSFORM transformation** teilt dem Server die Veränderung der Transformation des eigenen Avatars mit.

**SPEAK message** übermittelt eine Textnachricht.

**PICK obj\_name** aktiviert ein Objekt (auf beliebige Objekte anwendbar).

**BINARY data** überträgt Binärdaten, zum Beispiel das eigene Bild, kodiert mit Base64Jpeg-Xtra.

### Server -> Client

**LET obj\_name data** weist den Client an, die übermittelten Geometriedaten (Sw3d-Format) unter dem angegebenen Namen verfügbar zu machen und darzustellen. Statt Geometriedaten kann auch ein Hashcode eines bereits lokal verfügbaren Modells übertragen werden. Der Server ist dafür verantwortlich, die Eindeutigkeit der vergebenen Namen zu gewährleisten.

**FORCE obj\_name transformation** setzt die Weltkoordinaten des benannten Objekts. Die Transformation wird ohne Animation übernommen.

**PUSH obj\_name transformation** setzt die Weltkoordinaten des benannten Objekts. Die Transformation wird durch eine clientseitige Animationsroutine visualisiert. Dieses Kommando wird hauptsächlich für die Animation von Avataren verwendet.

**SHADE obj\_name shading** weist dem benannten Objekt einen Shader im Director-Format zu.

**DESTROY obj\_name timeout** zerstört ein mit LET erzeugtes Objekt. Ein optionales Timeout gibt die maximale Zeit zwischen Eintreffen des Befehls und Entfernen des Objekts an, um eventuell laufende Animationen beenden zu lassen.

**LOOKAT transformation** richtet die Kamera am Client entsprechend der übergebenen Transformation aus. Wird benutzt, um Blickrichtungsconstraints beim Betreten der Welt oder geführte Navigation durchzusetzen.

**SPEAK obj\_name message** lässt ein Objekt durch das Balloon3D-Xtra sprechen, das heißt eine comicartige Sprechblase darstellen. Der Befehl kann vom Server auch benutzt werden, um Hinweis-Schilder an Objekte anzubringen.

### Ordner auf DVD

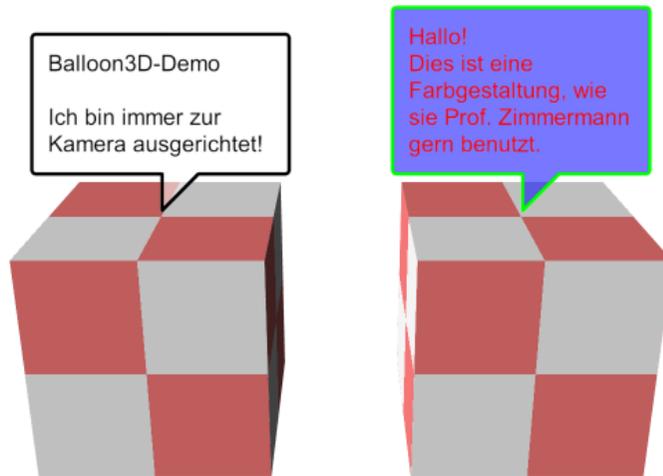
Multiverse

### Systemvoraussetzungen

Windows XP

DirectorMX 2004

## 5.4. Billboard-Sprechblasen



Bei der Konzeption des 3D-Chats *Multiverse* ergab sich die Notwendigkeit für personen- bzw. avatargebundene Textmeldungen, die im dreidimensionalen, virtuellen Raum platziert werden können. Um die Lesbarkeit der Textmeldungen sicherzustellen, ist deren Ausrichtung zur virtuellen Kamera erforderlich (Billboard). Dazu wurde eine Klasse `balloon3d` entwickelt, die Billboard-Sprechblasen realisiert. Sie dreht das ihr zugewiesene 3D-Modell stets so, dass dessen  $x$ - $y$ -Ebene dem Betrachter zugewandt ist. Die Position des Modells und die Verdrehung um die  $z$ -Achse werden dabei nicht verändert.

Die Billboard-Sprechblasen besitzen die folgenden Fähigkeiten:

- Ausrichtung zur Kamera
- Anzeigen von Text in der Sprechblase
- angezeigten Text ändern oder Text hinzufügen

- Anzeigen einer bestimmten Anzahl Textzeilen (ältere Zeilen werden dann nach oben aus der Sprechblase herausgeschoben)
- animiertes Ausblenden der Sprechblase, wenn der Inhalt eine bestimmte Zeit nicht verändert wurde
- animiertes Einblenden der Sprechblase, wenn die Sprechblase ausgeblendet war und neuer Text zum Anzeigen hinzugefügt wird
- Einstellen der Text-, Hintergrund- und Rahmenfarbe, sowie der Transparenz

Durch die Eigenständigkeit der Billboard-Sprechblasen sind sie potentiell auch in anderen Anwendungen außer *Multiverse* einsetzbar und können so beispielsweise zur Anzeige von Debugging-Informationen in virtuellen Szenarien verwendet werden.

### Ordner auf DVD

Multiverse\Balloon3d

### Systemvoraussetzungen

DirectorMX 2004

## 5.5. WebCamMiaw

Zur Personalisierung des virtuellen Stellvertreters im *Multiverse* war es geplant, diesen mit einem Foto des Chat-Teilnehmers zu versehen. Das Foto sollte mit einer an den PC angeschlossenen Webcam aufgezeichnet und anschließend an alle anderen Teilnehmer verteilt werden. Zur Unterstützung dieses Vorganges wurde in Director ein eigenständiges Modul, genannt *WebCamMiaw*<sup>4</sup>, entwickelt. Es wird über ein Lingo-Skript gestartet und übernimmt das Abrufen des Bildes von der Webcam und die anschließende Kodierung des Bildes mittels des *Base64Jpeg*-Xtras (siehe dazu Abschnitt 5.6).

Nach dem Start des *WebCamMiaw* wird der Videostream der Webcam angezeigt. Durch Klick des Benutzers auf eine beliebige Stelle im Anzeigefenster oder durch Drücken der Taste ENTER wird das Webcambild eingefroren. Der Nutzer hat nun die Möglichkeit den Schnappschuss zu verwenden oder einen neuen Versuch starten, wenn ihm die Aufnahme nicht gefällt. Hat der Nutzer seine Auswahl beendet, so wird das Bild mit Hilfe des *Base64Jpeg*-Xtras kodiert und zum Hauptfilm, der das *WebCamMiaw* gestartet hat, zurückgegeben. Der Ablauf ist in Abbildung 5.8 veranschaulicht.

### Ordner auf DVD

Multiverse\Webcam-Miaw und Base64JpegXtra

### Systemvoraussetzungen

DirectorMX 2004  
Webcam

<sup>4</sup>Die Bezeichnung *Miaw* leitet sich aus der in Director gebräuchlichen *Movie In A Window* Technik ab.

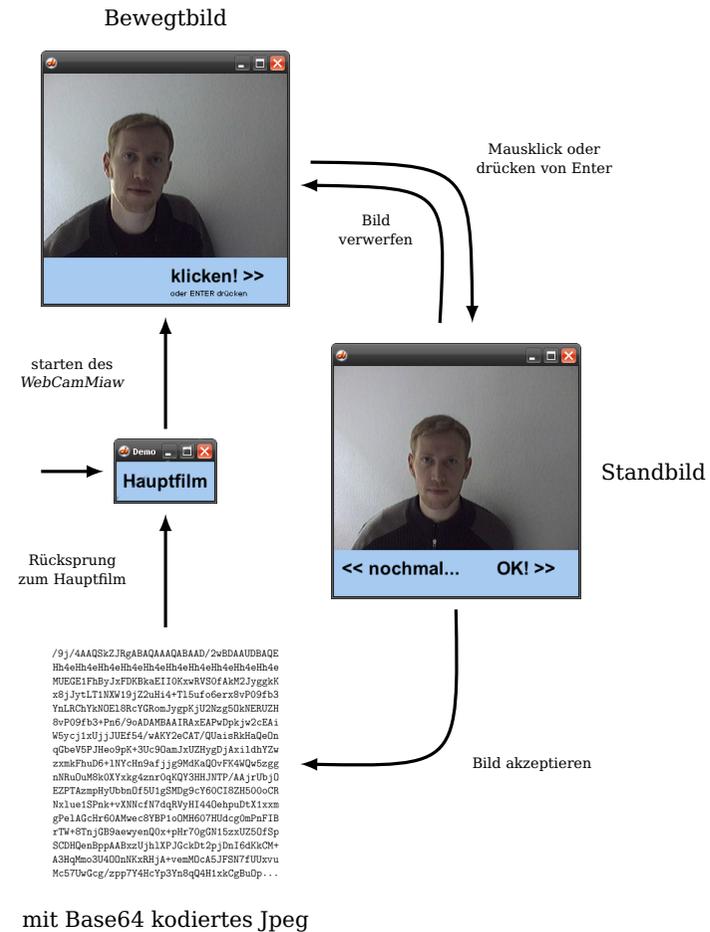


Abbildung 5.8.: Ablauf beim Einsatz des *WebCamMiaw*.

## 5.6. Base64Jpeg-Xtra

Mit Hilfe der in *Multiverse* genutzten Bibliothek *XtraNet* können Textnachrichten über ein Netzwerk ausgetauscht werden. Das Versenden von Binärdaten, wie beispielsweise Bildern, ist jedoch nicht ohne weiteres möglich. Das im Rahmen des Projektes *Multiverse* entwickelte *Base64Jpeg-Xtra* umgeht genau dieses Problem. Es wandelt Director-Bildobjekte ins platzsparende Jpeg-Format um und kodiert die entstandenen 8-Bit-Binärdaten mit Hilfe des so genannten Base64-Codes, der auch beim Versenden von Email-Anhängen genutzt wird.

Das Ergebnis enthält nur die druckbaren ASCII-Zeichen a-z, A-Z, 0-9, +, / und = und benötigt aufgrund des verringerten Zeichensatzes etwa 33% mehr Speicherplatz als die Daten des Jpeg-Bildes.

Das so umgewandelte Bild kann per *XtraNet* versendet und auf der Gegenseite wieder durch das *Base64Jpeg-Xtra* dekodiert und in ein Director-Bildobjekt umgewandelt werden.

### **Ordner auf DVD**

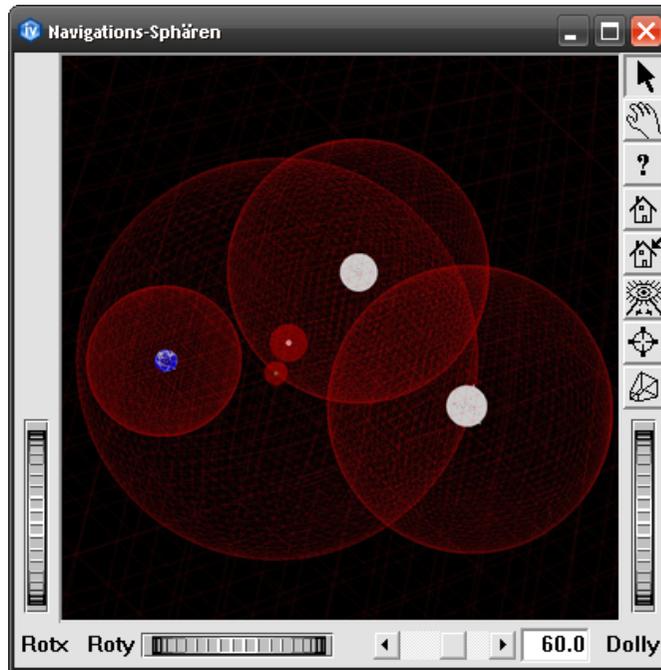
Multiverse\Webcam-Miaw und Base64JpegXtra

### **Systemvoraussetzungen**

Windows XP

DirectorMX 2004

## 6. Ensembles Sphärischer Distributionsräume & Navigation



**Aufgabe** Implementierung eines Prototypen zur sphärischen Distribution und Navigation

**Bearbeitungszeitraum** Sommersemester 2004

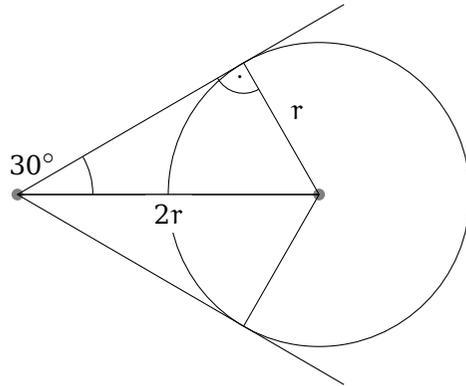
Distribution ist die Anordnung von Objekten, mit dem Ziel, durch diese Anordnung bestimmte Informationen schneller begreifen oder vorhandene Strukturen überhaupt erst erkennen zu können. Der vorliegende Prototyp experimentiert damit, wie

eine vorstrukturierte Objekt-Präsentation im 3-dimensionalen Raum aufgebaut sein kann, welche Bindungen zwischen bereitgestellten und distributorimmanenten Strukturen existieren, und welche Navigationsformen sinnvoll sind. Schwerpunkt liegt auf der Navigation, da Einbettungen in den 3-dimensionalen Raum mit Verdeckungen zu kämpfen haben, und eine Distribution über alle seine Freiheitsgrade nur in Kombination mit einer sinnvollen Navigation ein sowohl schnelles, als auch semantisch präzises Auffinden von Informationen ermöglicht.

Das Ergebnis versteht sich nicht als normative Lösung des Raumaufteilungsproblems, sondern als Teilschritt auf dem Weg zur Konzeption eines Navigations- und Distributionsraumes, in dem die Kombination beider Aspekte eine wirklich wegweisende Darstellungsform für Informationen in wissensbasierten Anwendungen bietet.

### 6.1. Kugeln mit Bebauungen

Einstiegspunkt des konzeptionellen Aufbaus bilden Trägerkörper, die durch ihren Aufbau die Anordnung der durch sie verteilten Elemente festlegen. Bereits am Anfang bestand die Idee, die Trägerkörper als Planeten aufzufassen, und sie entsprechend ihren Vorbildern aus der Realität zu gestalten. Es wurden daher Kugeln als Trägerkörper ausgewählt und mit Bebauungen versehen, die Gebäuden entsprechen sollten.



**Abbildung 6.1.:** Die Grenzlinie des Sichtfelds des Betrachters (links) bildet die Tangente zur Hüllsphäre.

Die Frage nach der Größe der Bebauungen wurde einerseits hinsichtlich der als Vorbild dienenden realen Maßstäbe, andererseits hinsichtlich der Brauchbarkeit als Verteiler abgewägt. Es standen sich Größenverhältnisse von 400.000:1<sup>1</sup> real und 3:1 auf den Konzeptskizzen gegenüber. Das Verhältnis in der Prototypenimplementierung liegt bei ca. 20:1 bis 10:1, um einerseits nicht den Charakter eines Planeten zu degradieren und andererseits die Bebauungen vom Orbit aus nicht unauffindbar klein werden zu lassen.

Im Zusammenhang mit der Bebauungsgröße bestand auch das Problem des optimalen Abstandes zum Planeten. Wie Abbildung 6.1 zeigt, ist dieser der Radius des Planeten selbst. Denn bei einem Kameraöffnungswinkel von 60° und einem Orbit, der dem doppelten des Planetenradius entspricht, nimmt der Planet bei der auf ihn gerichteten Kamera ein Maximum des Sichtfeldes ein. Es gibt keinen niedrigeren Orbit auf dem das der Fall ist.

<sup>1</sup>Erdumfang von 40.000km, zu Gebäuden von ca. 100m Ausdehnung

## 6.2. Navigation auf Großkreisen

Bei der Entwicklung der Navigationsformen war es durch die Kugelform der Träger-Körper naheliegend, eine sphärische Navigation auf so genannten *Navigations-Sphären* zu verwenden. Eine solche Sphäre ist bereits, wie oben beschrieben, durch den doppelten Planetenradius definiert.

Unter Nutzung von Kugel-Koordinaten wurde nun die Navigation, durch Umrunden des Planeten auf dessen Großkreisen, implementiert. Großkreise sind diejenigen Orbits, deren Mittelpunkt koinzident mit dem Planetenzentrum ist. Namentlich bezeichnete Großkreise sind die Meridiane (Längengrade) und der Äquator. Das Umrunden unter Variation des Azimut entspricht dem Parameter  $\Phi$ , das Variieren der Höhe dem Parameter  $\Theta$  in Kugelkoordinaten.

Das Überschreiten der Pole, das heißt eine polfreie Navigation, wurde aufgrund der effizienteren Erreichbarkeit aller Punkte auf der Kugel implementiert. Es gab jedoch schon Überlegungen, die  $\Theta$ -Navigation mit Constraints an den beiden Polen zu versehen, um diese Punkte als Konstanten zur Grundorientierung einzusetzen. Die Projekte zum Thema *Abgestumpftes Ikosidodekaeder* in Abschnitt 7.2 beschäftigen sich mit an den Polen beschränkter Navigation.

Die *Phi-Theta-Navigation* erfolgt durch Variation der oben beschriebenen Parameter  $\Phi$  und  $\Theta$  der tangentialen Navigation, und wird ergänzt durch den dritten Freiheitsgrad des Kugelkoordinatensystem im dreidimensionalen Raum: der radialen Koordinate  $R$ .  $R$  gibt den Abstand eines Punktes vom Zentrum des Koordinatensystems an.

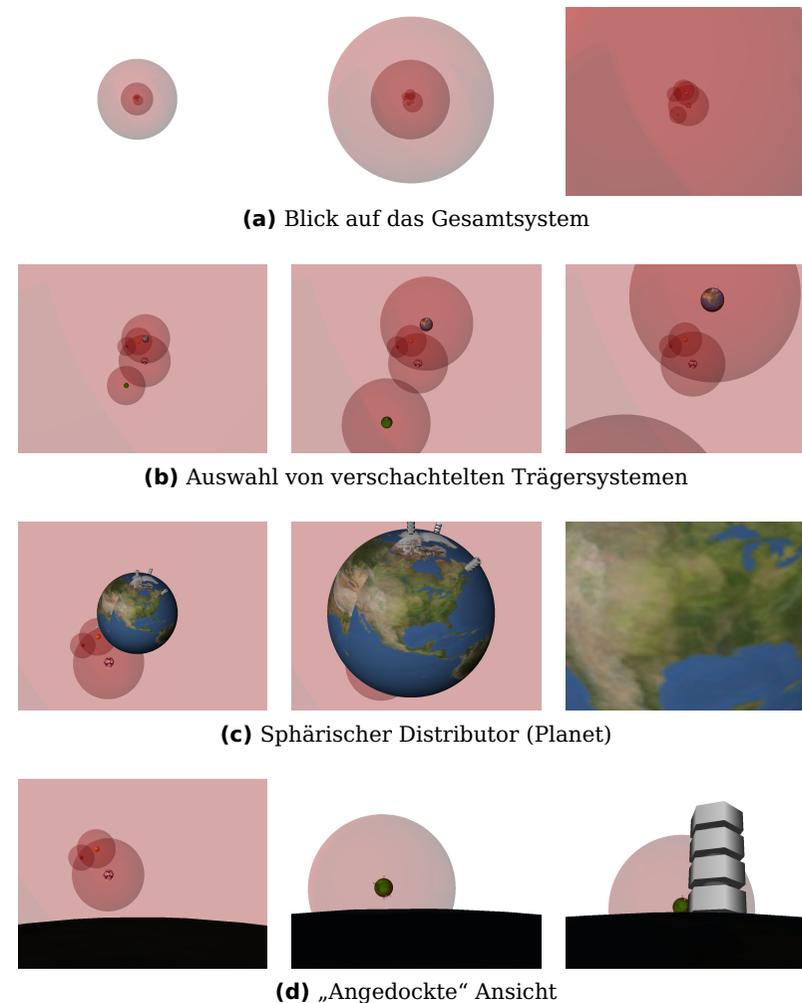
Da die Navigation auf Navigations-Sphären stattfinden sollte, die als Navigations-Constraints fungieren, erfolgt die Naviga-

tion in radialer Richtung auf bestimmten, vorher festgelegten Radien um den zentralen Träger-Körper. Jeder Radius wurde dabei doppelt so groß wie der nächstkleinere gewählt. Es lässt sich zeigen, dass die sichtbare Höhe und Breite des zentralen Körpers sich, bei dem Wechsel auf eine Navigations-Sphäre mit doppeltem Radius, in etwa halbiert. Der Effekt ist, dass der Benutzer eine annähernd lineare Größenänderung am Körper wahrnimmt. Abbildung 6.2 zeigt eine Serie von Bildern der sukzessiven Annäherung an einen Träger-Körper (hier eingebettet in ein hierarchisches System solcher).

### 6.3. Ensemble von Kugeln

Das Konzept des Träger-Körpers als Planeten, auf ein Analogon zum Sonnensystem auszudehnen, war der nächste logische Schritt der Konzeption. Da man keinen Wert auf eine Modellierung der physikalischen Rahmenbedingungen legte, war es möglich, auf ein Zentralgestirn zu verzichten und mehrere Planeten in fixer Konstellation anzuordnen. Dieses Ensemble konnte nun wiederum in ein System von Hüll- bzw. Navigations-Sphären eingebettet werden. Man sieht leicht, dass sich hier die Möglichkeit zur rekursiven Bildung von Trägersystemen ergibt.

Von Bedeutung für die Navigierbarkeit ist jedoch nicht nur die hierarchische Traversierbarkeit. Während der Entwicklung des Prototypen wurde bemerkt, dass Benutzer die visuelle Nähe zweier Träger-Körper zum Anlass nahmen, diesen anzuspringen. Diese Art der Navigation lag zunächst außerhalb der radialen R-Navigation und der hierarchischen Strukturierung der Distributions-Sphären, wurde jedoch rasch in den Prototypen aufgenommen. Jetzt ist es daher möglich, Nachbarn, das heißt



**Abbildung 6.2.:** Hierarchische Navigation durch ein Trägersystem.

Kinder des übergeordneten Ensembles direkt anzuspringen. Ist das eigentlich Ziel wiederum Teil eines Ensembles, so wird erst dessen Distributions-Sphäre angesprungen, von wo aus es durch hierarchische Navigation erreichbar ist.

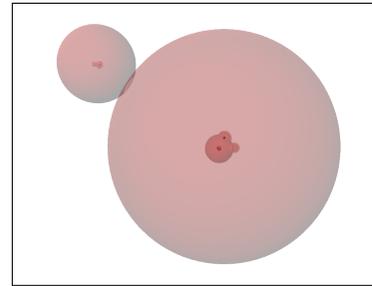
### 6.3.1. Auftretende Probleme

#### Visualisierung der Navigations-Sphären

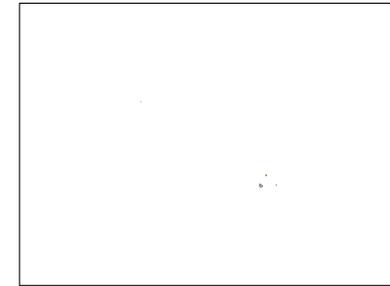
Infolge der freien Platzierbarkeit der Planeten beziehungsweise Distributions-Sphären trat aber auch das Problem der Überdeckung und Überschneidung der Hüll-Sphären innerhalb eines Ensembles auf. Da Navigations-Sphären als logische Strukturen angelegt waren, hatten sie einen beschränkenden Effekt auf die Navigation, der nicht visuell nachvollziehbar war, anders als zum Beispiel durch die Darstellung einer Planetenoberfläche. Die Visualisierung der Navigations-Sphären war mit der Einführung des rekursiven Distributoraufbaus also unabdingbar geworden. Ein weitere Verschärfung des Problem tritt dadurch ein, dass bei großen Rekursionstiefen ein Großteil des dargestellten Raumes durch die Navigations-Sphären eingenommen wird, während die massiven Träger-Körper, die Planeten, mitunter kleiner als 1 Pixel werden können (siehe Abbildung 6.3). Es ist zu beachten, dass selbst wenn die Visualisierung der Navigations-Sphären im Prototyp deaktiviert wird (Taste H), die Hierarchie der Träger-Systeme nicht übergangen werden kann, das heißt Planeten nicht direkt angesprungen werden.

#### Orientierungsverlust

Orientierungsverlust beim Springen von der niedrigsten Navigations-Sphäre eines Ensembles zu der höchsten einer



(a) gut sichtbare Hierarchiebeziehungen durch Visualisierung der Navigationssphären



(b) fehlende Visualisierung der Navigations-Sphären bei gleicher Entfernung

**Abbildung 6.3.:** Notwendigkeit der Visualisierung der Navigations-Sphären bei großer Rekursionstiefe.

der eingebetteten Distributions-Sphären ist ein Problem, da sich zumeist Mittelpunkt und Rotation des Systems ändern. Der Lösungsansatz versucht den Orientierungsverlust zu vermeiden. Der Sprung ist nach wie vor rasant, das heißt um ein Vielfaches schneller als die tangentiale  $\Phi$ - $\Theta$ -Navigation, es wird jedoch kein Hartschnitt, sondern eine, die Endtransformationen interpolierende, Kamerafahrt verwendet. Diese Art der Animation wird ebenfalls beim Sprung zwischen den Navigations-Sphären eingesetzt und erzeugt ein gleichmäßiges Benutzerempfinden.

#### Überschneidung

Die Überschneidung der Navigations-Sphären verschiedener Träger-Körper wurde frühzeitig als Problem erkannt, dennoch konnte kein befriedigender Lösungsvorschlag erbracht werden. Da nicht nur mehrere Ebenen von Navigations-Sphären, sondern sogar Träger-Körper selbst durch willkürliche Platzierung in den Raum der Navigations-Sphären eines anderen

Träger-Körper eindringen können (ganz zu schweigen von der Überschneidung der Träger-Körper untereinander), wäre hier ein aufwendiges Gesamtkonzept - eventuell in Form eines Platzierungs-Editors - zur Vermeidung der Überschneidungen nötig gewesen.

## 6.4. Planetenlandung

Gegen Ende des Projektzeitraumes strebte man an, andere, zuvor entwickelte Distributionskonzepte in den Prototypen zu integrieren. So wurden Türmchen von sechseckigen Prismen als Dummy-Bebauungen auf die Oberflächen der Planeten platziert. Da die Prismen entsprechend ihrer Konzeptionierung seitlich anzusteuern waren, musste hierfür die Navigation angepasst werden, so dass auch bei sehr niedrigen Orbits eine Orientierung sinnvoll möglich war. Hierzu wurde eine Landung des Betrachters auf dem Planeten simuliert. Seine anschließende Bewegung ist zwar weiterhin tangential, da die Perspektive jedoch den Horizont des Planeten zeigt, kann sie auch als lineare Bewegung auf der Planetenoberfläche interpretiert werden.

## 6.5. Kamerasteuerung

Insbesondere in Hinblick auf die Planetenlandung wurden außer der  $\Phi$ - $\Theta$ -Navigation noch zahlreiche andere Modi experimentell implementiert. Generell lassen sich alle Navigationsarten über den Nummernblock der Tastatur, alternativ auch über die entsprechend benannten Tasten im normalen Tastenbereich, bedienen. Es existieren 3 Modi der Navigation:

**Radial-Modus** Der Betrachter bewegt sich kreisförmig um ein Objekt. Entspricht  $\Phi$ - $\Theta$ -Navigation. Ist standardmäßig aktiviert und kann nach Wechsel der Navigationsart über Taste  $\times$  reaktiviert werden.

**Polar-Modus** Der Betrachter dreht sich um sich selbst. Wendet die  $\Phi$ - $\Theta$ -Navigation auf den Betrachter selbst an. Aktivierung über die Taste  $\div$ .

**Linear-Modus** Der Betrachter bewegt sich entlang der Achsen des kartesischen Koordinatensystems. Aktivierung über die Taste  $-$ .

Zwischen den drei Navigationsmodi kann frei umgeschaltet werden. Wird allerdings R-Navigation verwendet, das heißt die Hierarchie der Distributions-Sphären traversiert, so werden im Polar- und Linear-Modus vorgenommene Änderungen automatisch an eine gültige Position auf den Navigations-Sphären angepasst.

Bei der Traversierung der Hierarchie kann zwar zwischen den Navigations-Sphären eines Trägersystems per Tastatur gewechselt und auch in das übergeordnete System gesprungen werden (Bild<sup>↑</sup> bzw. Bild<sup>↓</sup>), die Auswahl eines untergeordneten Systems ist jedoch nur per Mausklick möglich. Neben den in Abschnitt 6.5.1 abgebildeten Tasten sind zur Planetenlandung noch die Tasten Entf, zum Andocken vom niedrigsten Orbit aus, sowie 0 zur Inversion der Y-Achse, von Bedeutung.

Die Maus kann zwar nicht zur  $\Phi$ - $\Theta$ -Navigation eingesetzt werden, ein freies Traversieren der Hierarchie ist jedoch durchaus möglich. So dient die *linke Maustaste* zur Auswahl der untergeordneten Elemente eines Träger-Systems und zum Abstieg auf einen niedrigeren Orbit des aktuellen Trägers, sowie schließlich zur Planetenlandung, wenn der niedrigste Orbit erreicht

ist. Die *zweite Maustaste*<sup>2</sup> wirkt in entgegengesetzter Richtung und wechselt auf höhere Orbits beziehungsweise in das übergeordnete System.

Nach Ende des Projektzeitraums wurde der Prototyp noch um eine Anbindung an Microsofts DirectX-Schnittstelle erweitert, so dass auch Joysticks zur Steuerung eingesetzt werden können (Abbildung 6.4a). Der P5-Datenhandschuh konnte aufgrund seiner Kompatibilität mit Zeigereingabegeräten ebenfalls erfolgreich zur Steuerung der  $\Phi$ - $\Theta$ -Navigation eingesetzt werden (Abbildung 6.4b).

### 6.5.1. Tastenbelegung der Kamerasteuerung

#### Alle Modi



Auswahl eines untergeordneten Trägersystems



Polar-Modus aktivieren (Rotation um Betrachterstandpunkt)



Radial-Modus aktivieren (Rotation um Objekt)



Linear-Modus aktivieren (Rotation um Objekt)



y-Achse umkehren

<sup>2</sup>Da OpenInventor die rechte Maustaste vorbelegt, entspricht Taste 2 zumeist der mittleren Maustaste.



(a) DirectX-8-kompatibler Joystick



(b) P5-Datenhandschuh

**Abbildung 6.4.:** Experimentelle Anbindung weiterer Eingabegeräte.

### Radius- und Polar-Modus

		$\Theta$ -Navigation
		$\Phi$ -Navigation
		R-Navigation
		Rotation um z-Achse

### Linear-Modus

		Navigation entlang x-Achse
		Navigation entlang y-Achse
		Navigation entlang z-Achse

### Ordner auf DVD

SphericalEnsemble

### Systemvoraussetzungen

Windows XP

OpenInventor 5.04



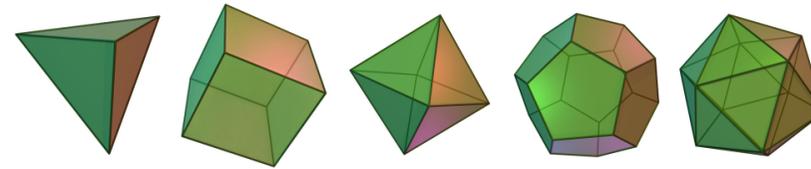
## 7. Polyeder als Trägergebilde

Nach der Untersuchung von sonnensystemartigen Trägergebilden, welche eine Navigation auf geschachtelten Navigationssphären erlaubten, rücken nun die Träger-Körper („Planeten“) dieser Sonnensysteme in den Vordergrund. Die weitere Verteilung von Informationen sollte auf Basis von konvexen Polyedern, also konvexen Körpern, die ausschließlich von ebenen Flächen begrenzt werden, geschehen. Dabei waren die Platonischen und Archimedischen Körper, wie sie in Abbildung 7.1 zu sehen sind, aufgrund ihrer Regelmäßigkeit von besonderem Interesse.

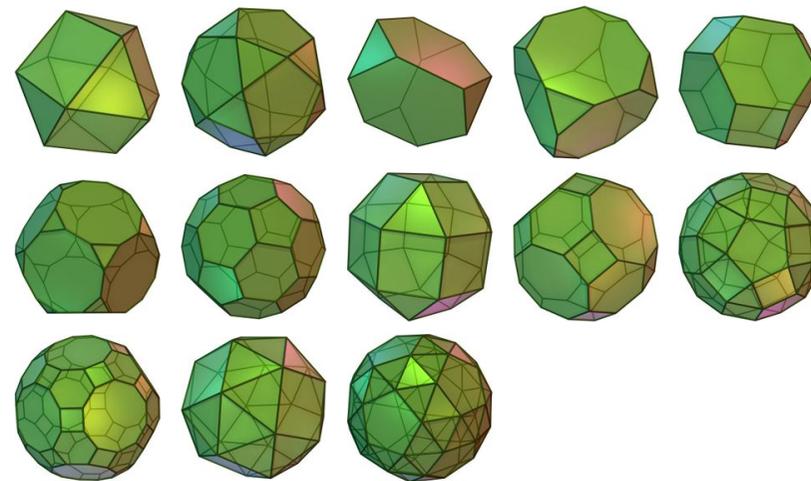
Auf den differenzierten N-Eck-Flächen dieser Körper können Informationen einerseits direkt, zum Beispiel durch ein Textur-Mapping von Bild- oder Video-Material, präsentiert werden. Andererseits können weitere Zellen beziehungsweise Trägergebilde auf den Flächen platziert oder in sie eingebettet werden.

Bei der Navigation um derartige Polyeder ist es von Vorteil, eine Polachse festzulegen, an der das Obere der Personalzelle ausgerichtet wird. Dieses zusätzliche Constraint verhindert einen Orientierungsverlust beim Nutzer, wenn dieser zum Beispiel um den Polyeder kreist, um eine Fläche zur genaueren Betrachtung oder zur Landung auszuwählen.

Die nachfolgenden Unterkapitel erläutern die einzelnen Prototypen, die zu diesem Thema entstanden sind.



(a) Platonische Körper



(b) Archimedische Körper

**Abbildung 7.1.:** Reguläre und halbrekuläre Polyeder (Bilder wurden [Coma, Comb] entnommen).

## 7.1. Platonische Körper

Platonische Körper, auch reguläre Polyeder genannt, sind dadurch charakterisiert, dass ihre Seitenflächen zueinander kongruente regelmäßige Vielecke sind, von denen in jeder Ecke des Körpers jeweils gleich viele zusammentreffen. Begonnen wurde mit der Untersuchung der geometrischen Eigenschaften dieser Körper: Der Zusammenhang zwischen der Kantenlänge der Seitenflächen und dem Außenkugelradius kann dazu genutzt werden, in effizienter Weise Hüll- und Navigationsphären für diese Trägergebilde zu erzeugen. Der Innenkugelradius ist hingegen bei der Einbettung anderer Körper in den Platonischen Körper verwendbar. Nachdem besagte Eigenschaften sowohl theoretisch (Abschnitt 7.1.1), als auch in einem Prototypen zur Generierung von 3D-Modellen Platonischer Körper (Abschnitt 7.1.2) praktisch untersucht wurden, verschob sich der Projektfokus auf das *Ikosidodekaederstumpf* genannte Polyeder. Die zugehörigen Projekte werden in Abschnitt 7.2 erläutert.

### 7.1.1. Ausgewählte Eigenschaften Platonischer Körper

**Aufgabe** Für die fünf Platonischen Körper sollte der Zusammenhang von Kantenlänge, Innen- und Außenkugelradius untersucht werden.

**Bearbeitungszeitraum** Wintersemester 2004/2005

Eine Ausarbeitung zu diesem Thema ist in Anhang B.2 zu finden.



**Abbildung 7.2.:** Die Ausarbeitung im Miniaturformat.

### Ordner auf DVD

Dokumente

### Systemvoraussetzungen

PDF-Betrachter

### 7.1.2. PowerPlaton

**Aufgabe** Zum Einsatz Platonischer Körper als Trägergebilde sollte ein Generator zur Erzeugung der 3D-Modelle dieser Körper entwickelt werden. Dabei sollte ein Skalierungsfaktor mit einberechnet werden, so dass das resultierende Modell eine bestimmte Kantenlänge oder einen bestimmten Radius der Innen- bzw. Außenkugel aufweist. Desweiteren sollte es möglich sein, den „Nordpol“ des Modells festzulegen.

**Bearbeitungszeitraum** Wintersemester 2004/2005

Das erstellte Programm, genannt *PowerPlaton*, besteht aus einem Steuerpult und einem Viewer. Das Steuerpult erlaubt die Auswahl des zu bearbeitenden platonischen Körpers über ein Pull-Down-Menü und die Einstellung der gewünschten Eigenschaften: Kantenlänge, Innenkugelradius und Außenkugelradius, per Schieberegler oder per Direkteingabe. Das im Viewer angezeigte Modell reagiert unmittelbar auf Veränderungen der Parameter. Im Anzeigefenster vorgenommene Drehungen, Skalierungen und Translationen wirken sich auch ausschließlich auf die Anzeige, nicht auf das zugrundeliegende Modell aus. Der fertig konfigurierte Platonische Körper kann anschließend ins IV- oder VRML-Format exportiert werden. Dabei wird der „Nordpol“ des Körpers je nach Einstellung auf einen Eckpunkt, eine Kantenmitte oder eine Flächenmitte ausgerichtet.

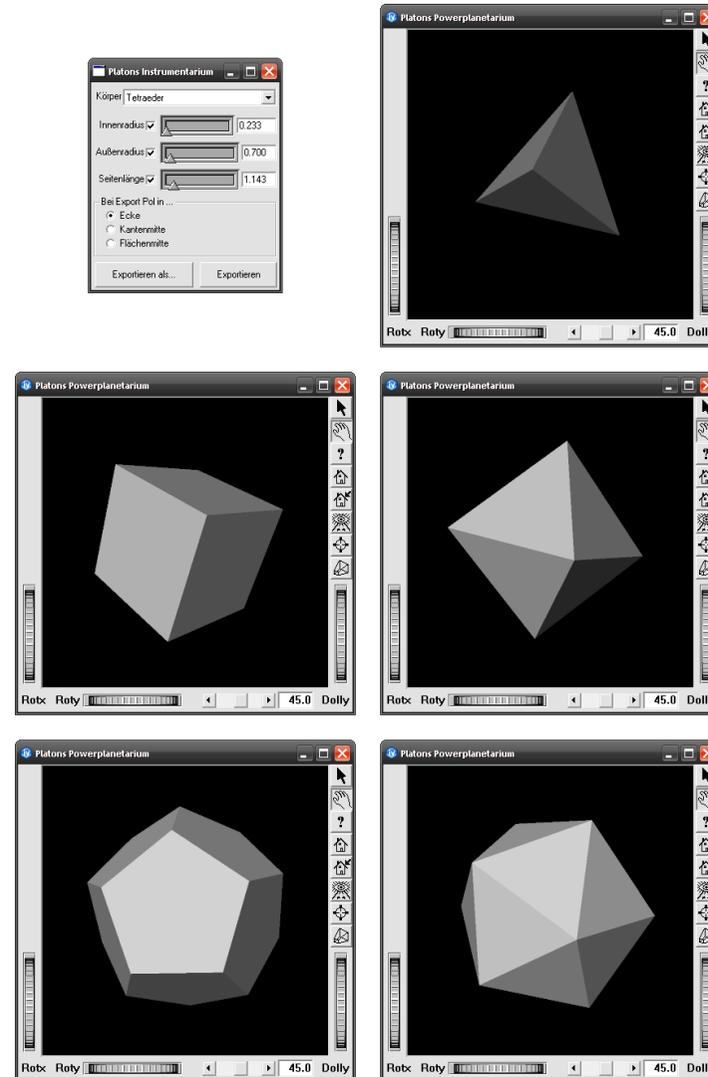
#### Ordner auf DVD

PowerPlaton

#### Systemvoraussetzungen

Windows XP

OpenInventor 5.04



**Abbildung 7.3.:** PowerPlaton-Bedienelemente und -Viewer mit allen fünf platonischen Körpern.

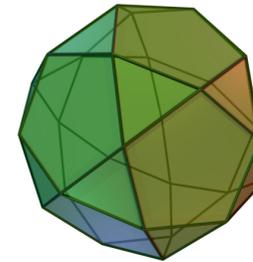
## 7.2. Abgestumpftes Ikosidodekaeder

Nach der fundierten Untersuchung Platonischer Körper beschränkte sich das Themengebiet nun auf einen einzelnen Archimedischen Körper – das abgestumpfte Ikosidodekaeder (auch Ikosidodekaederstumpf oder großes Rhombenikosidodekaeder genannt). Diese Einschränkung bot den Vorteil, sich auf die konkreten Eigenschaften eines Körpers konzentrieren zu können und damit die prototypische Entwicklung zu beschleunigen. Während der Arbeit am Ikosidodekaederstumpf entstanden mehrere Prototypen, teilweise in verschiedenen Autorensystemen (OpenInventor, Director), teilweise zu verschiedenen Teilaspekten dieses Trägergebildes.

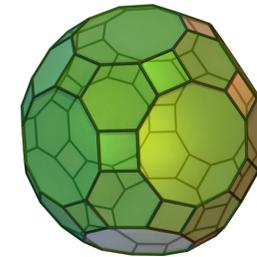
### 7.2.1. Eigenschaften

Das abgestumpfte Ikosidodekaeder besteht aus 30 regelmäßigen Vierecken, 20 regelmäßigen Sechsecken und 12 regelmäßigen Zehneck, wobei in jeder Ecke des Körpers jeweils ein Viereck, ein Sechseck und ein Zehneck zusammentreffen. Überraschenderweise kann, trotz der von Johannes Kepler eingeführten Namensgebung, dieser Körper nicht durch Abschneiden der Ecken eines Ikosidodekaeders erzeugt werden. Der resultierende Körper wäre zwar topologisch äquivalent zu dem hier betrachteten Archimedischen Körper, jedoch wären die Vierecke nicht quadratisch, sondern rechteckig (Abbildung 7.4).

Weitere für die implementierten Prototypen wichtige Eigenschaften sind die Zusammenhänge zwischen Kantenlänge und Innen- beziehungsweise Außenkugelradius des Körpers sowie zwischen den entsprechenden Radien der Seitenflächen. Die nachfolgenden Formeln sind [Wei03] beziehungsweise [Bre07]



(a) Ikosidodekaeder



(b) Ikosidodekaederstumpf

**Abbildung 7.4.:** Durch die unterschiedlichen Innenwinkel von Drei- und Fünfeck würden durch Abschneiden der Ecken des Ikosidodekaeders (a) Rechtecke anstelle von Quadraten entstehen und somit auch nicht der in (b) abgebildete Ikosidodekaederstumpf. Die Namensgebung ist daher etwas mißverständlich.

entnommen:

Kantenlänge	$a$
Innenkugelradius	$r = \frac{1}{2} \sqrt{25 + 10\sqrt{5}} a$ $\approx 3.44095a$
Außenkugelradius	$R = \frac{1}{2} \sqrt{31 + 12\sqrt{5}} a$ $\approx 3.80239a$
Viereck-Innenkreisradius	$r_4 = 0.5a$
Viereck-Außenkreisradius	$R_4 = \frac{1}{\sqrt{2}} a$ $\approx 0.70711a$
Sechseck-Innenkreisradius	$r_6 = \frac{\sqrt{3}}{2} a$

	$\approx 0.86603a$
Sechseck-Außenkreisradius	$R_6 = a$
Zehneck-Innenkreisradius	$r_{10} = \frac{\sqrt{5 + 2\sqrt{5}}}{2}a$
	$\approx 1.53884a$
Zehneck-Außenkreisradius	$R_{10} = \frac{\sqrt{5 + 1}}{2}a$
	$\approx 1.61803a$

### Kürzeste-Wege-Problem

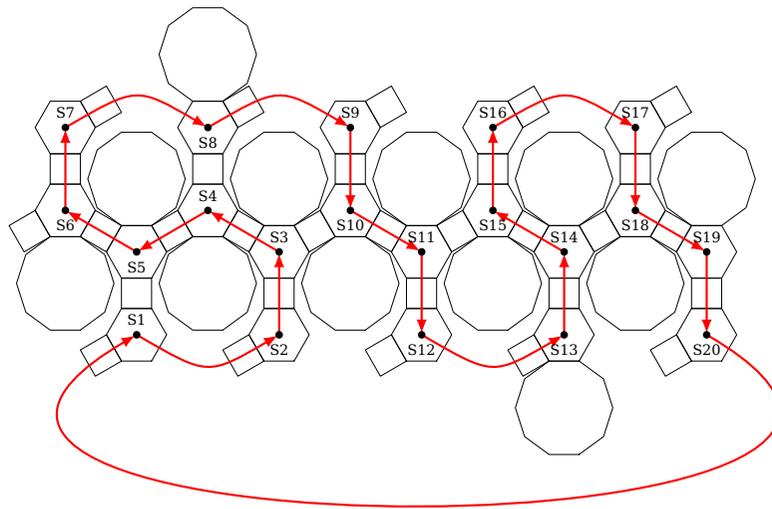
Bei der Navigation zwischen zwei beliebigen Flächen des abgestumpften Ikosidodekaeders ist es von Vorteil, wenn der Benutzer dabei einen möglichst kurzen Weg zurücklegt. Soll die Bewegung zwischen Start- und Zielfläche nicht einfach interpoliert werden, sondern entlang eines Pfades über jeweils benachbarte Flächen erfolgen, so ist auch hierbei eine minimale Pfadlänge zu präferieren. Der Algorithmus zur Berechnung eines minimalen Pfades ist für den Spezialfall des Ikosidodekaederstumpfes verhältnismäßig einfach: Ausgehend vom aktuellen Standpunkt wird diejenige Nachbarfläche angesprungen, deren Mittelpunkt dem der Zielfläche am nächsten steht. Dies wird solange wiederholt bis die Zielfläche erreicht ist.

### Hamilton-Kreis-Problem

Bei der Untersuchung des Ikosidodekaederstumpfes kam die Frage auf, ob dem Benutzer eine virtuelle Führung über die Sechseckflächen des Körpers, die zum damaligen Zeitpunkt als alleinige Präsentationsfläche dienten, ermöglicht werden kann. Die Route sollte außer den Sechsecken nur die Vierecke

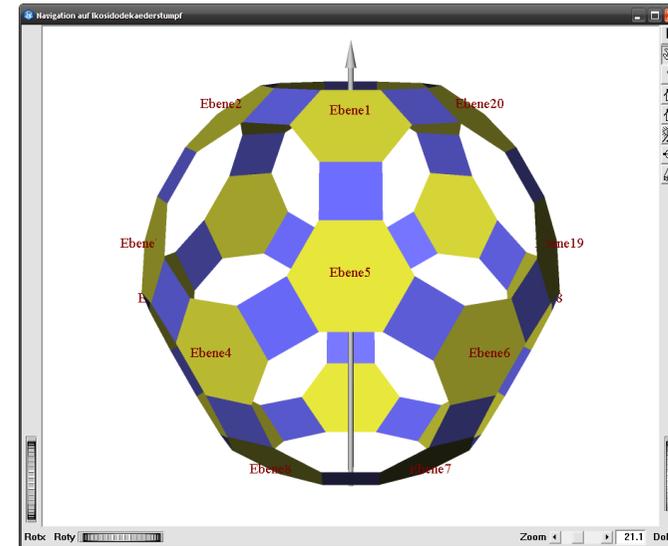
enthalten, keine Sechsecke auslassen oder doppelt enthalten und am Ende wieder beim Startpunkt ankommen. In der Graphentheorie wird eine solche Route Hamiltonscher Kreis genannt [Wei07b]. Die Bestimmung von Hamiltonschen Kreisen ist im Allgemeinen sehr schwierig<sup>1</sup>. Auf allen Platonischen und einigen Archimedischen Körpern (darunter auch der Ikosidodekaederstumpf) sind aber seit langem Hamiltonsche Kreise bekannt, die *alle* Flächen enthalten [Wei07b, Wei07a]. Durch gezieltes Entfernen von Flächen oder Flächentypen aus der Route kann jedoch ein Graph entstehen, der keinen Hamiltonschen Kreis mehr enthält. Dies ist zum Beispiel der Fall, wenn man versucht alle Vierecke und Sechsecke des Ikosidodekaederstumpfes durch einen Kreis zu verbinden. Da jeder Weg immer abwechselnd Vier- und Sechsecke enthält und es deutlich mehr Vier- als Sechsecke gibt, müssen einige Sechsecke zwangsläufig mehrfach durchlaufen werden. Erfreulicherweise kann man zwischen den Sechsecken des Ikosidodekaederstumpfes leicht einen Hamiltonschen Kreis finden, wie Abbildung 7.5 zeigt.

<sup>1</sup>Das Hamilton-Kreis-Problem ist NP-vollständig [Wei07b].



**Abbildung 7.5.:** Hamiltonscher Kreis auf den Sechsecken des abgestumpften Ikosidodekaeders (hier als abgewickeltes Gitternetz dargestellt), wobei nur Vierecke überquert werden.

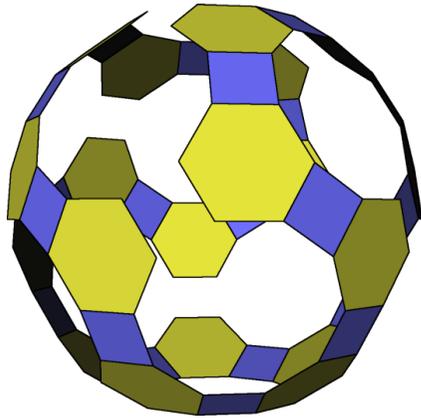
### 7.2.2. Prototyp in OpenInventor



**Aufgabe** Für die spätere Nutzung der einzelnen N-Eck-Flächen des Ikosidodekaederstumpfes als Distributionsräume sollten Navigationstechniken zwischen diesen Flächen erprobt werden.

**Bearbeitungszeitraum** Wintersemester 2004/2005

Das Projekt startete mit einer Recherche nach einem geeigneten 3D-Modell des abgestumpften Ikosidodekaeders, in dem jede N-Eck-Fläche des Körpers als einzelnes Objekt vorhanden war. Das beste verfügbare Modell [Har96] hatte leider die Eigenschaft, alle Flächen eines Typs zu einem geometrischen Modell zusammenzufassen. Nach der Separierung der einzelnen Flächen wurde ein Algorithmus entwickelt, welcher die zur



**Abbildung 7.6.:** dreidimensionale Ansicht des Hamiltonschen Pfades auf dem Ikosidodekaederstumpf

Navigation benötigten Nachbarschaftsbeziehungen der Flächen berechnet.

Mit diesen Informationen konnte nun ein Prototyp implementiert werden, der dem Benutzer eine Navigation zwischen beliebigen Sechseckflächen des Körpers erlaubte. Der dabei zurückgelegte Weg führt nur über Vier- und Sechsecke und überquert so wenige Flächen wie möglich (siehe Kürzeste-Wege-Problem, Abschnitt 7.2.1). Die Zehnecke des Körpers wurden ausgeblendet, um auch die Flächen auf der Rückseite des Ikosidodekaederstumpfes auswählen zu können. Auch eine Führung entlang des Hamiltonschen Kreises (Abschnitt 7.2.1) ist möglich. Um dem Nutzer bereits im Vorfeld die abzulaufende Route näher zu bringen, können im Prototypen alle Flächen, die nicht zum Hamiltonschen Kreis gehören, unsichtbar gemacht werden (Abbildung 7.6).

## Polbildung und Ausrichtung der Personalzelle

Zur besseren Orientierung bei der Navigation auf dem Ikosidodekaederstumpf wurde eine (sichtbare) Polachse, wie sie im Eingangsbild zu sehen ist, durch die Mittelpunkte zweier gegenüberliegender Zehnecke gelegt. Während die Blickrichtung stets auf den Körpermittelpunkt fixiert ist, orientiert sich der Up-Vektor der Personalzelle am *Nordpol* des Körpers. Die Personalzelle, und damit die Sicht des Betrachters, kann also nicht unbeabsichtigt seitlich geneigt werden.

## Steuerung des Prototypen

**linke Maustaste** Auswählen einer Sechseckfläche, die anschließend angefliegen wird

**Pfeiltasten** Vor- und Zurückbewegen auf dem Hamiltonschen Kreis

**H** nur Flächen des Hamiltonschen Kreises anzeigen

**A** Außenkugel des Ikosidodekaederstumpfes einblenden

**I** Innenkugel des Ikosidodekaederstumpfes einblenden

## Ordner auf DVD

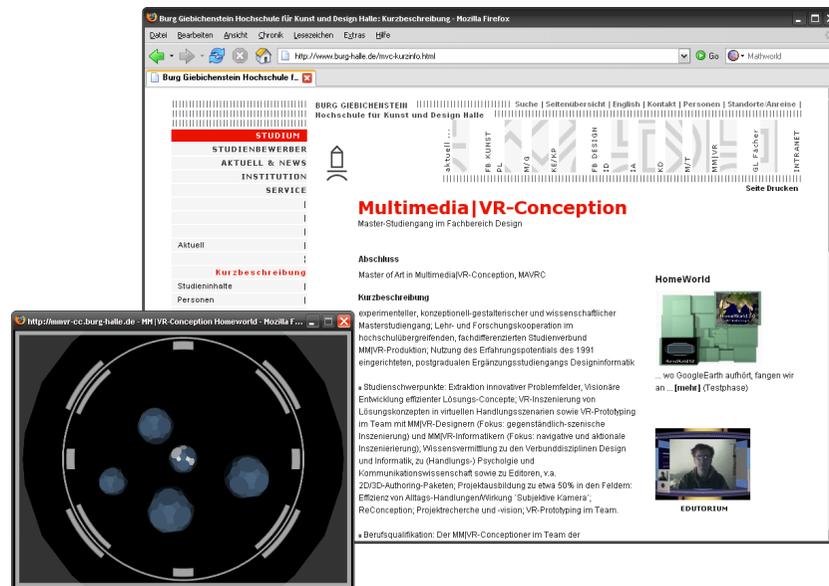
Ikosidodekaederstumpf\OpenInventor

## Systemvoraussetzungen

Windows XP

OpenInventor 5.04

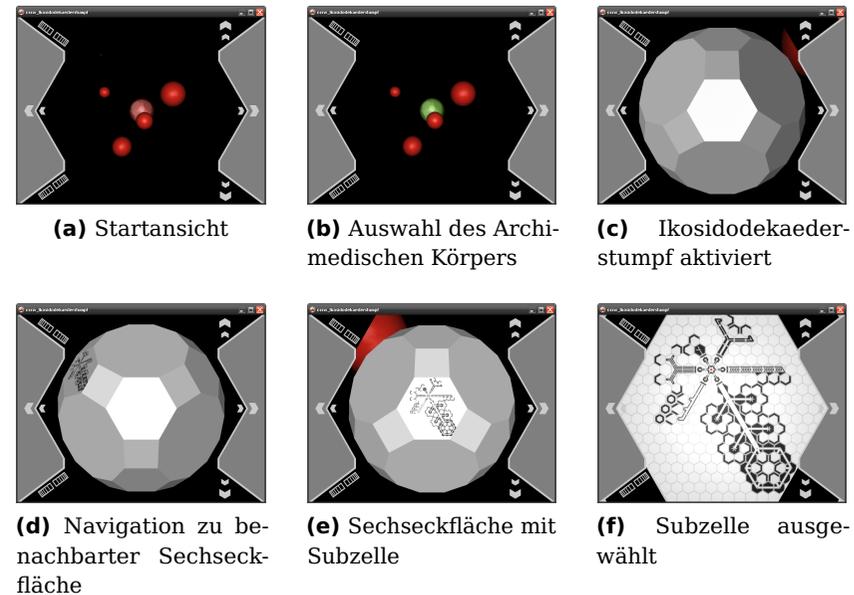
### 7.2.3. Prototyp in Director



**Aufgabe** Für den Onlineauftritt des Studienganges MM|VR-Konzeption sollten die in OpenInventor entwickelten Prototypen zur sphärischen Navigation und zur Navigation auf dem Ikosidodekaederstumpf in Director portiert und verschmolzen werden.

**Bearbeitungszeitraum** Wintersemester 2004/2005, Sommersemester 2005

Der entwickelte Prototyp war Teil eines größeren Projektes, welches den Anwärtern des Studienganges einen Einblick in die am Lehrstuhl erforschte Handlungszellenthematik und die im Entstehen begriffene virtuelle Homeworld geben sollte. Das oben dargestellte Szenario kann auf der Webseite [Kol] gestartet werden. Es besteht aus einer linearen Abfolge von Videos und



**Abbildung 7.7.:** Schritte vom Start des Prototypen bis zur Landung auf einer der Sechseckflächen.

Director-Filmen, und enthält eine optisch und funktional leicht überarbeitete Variante des internen Prototypen (Abbildung 7.7).

#### Personalzelleninterface

Im Gegensatz zu den früheren Prototypen in OpenInventor, deren Steuerung direkt über Tastatur und Maus erfolgt, wurde erstmals ein Teil der Benutzerführung in die Personalzelle integriert. Bei der sphärischen Navigation kann der Breitengrad  $\Theta$  über  $\blacktriangledown \blacktriangleleft$  und der Längengrad  $\Phi$  über  $\blacktriangleleft \blacktriangleright$  eingestellt werden. Die jeweils verkleinerten Bedienelemente erlauben eine

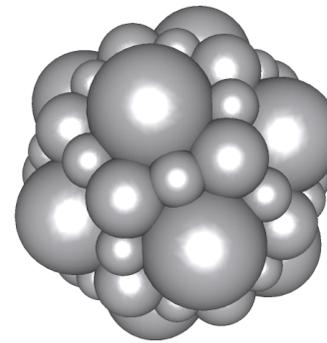
Feineinstellung des zugehörigen Winkels. Mit Hilfe von  ist ein Sprung in die jeweils übergeordnete Hierarchieebene des Distributionsraumes möglich.

### Unterschiede zwischen internem Prototypen und Onlineversion

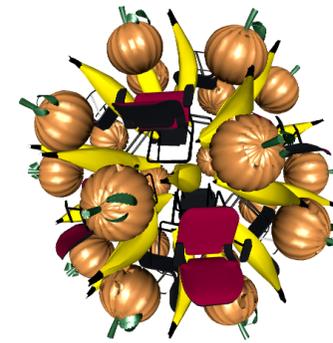
Beide Versionen unterscheiden sich hauptsächlich im 3D-Modell der Personalzelle und der Dummy-Geometrie, die den im Mittelpunkt stehenden Ikosidodekaederstumpf umgibt. Um den Nutzer besser durch das Szenario zu führen, wurde in der Onlineversion die Navigation auf dem Ikosidodekaederstumpf eingeschränkt. Das Springen zu beliebigen Sechseckflächen ist hier nicht mehr möglich. Stattdessen startet nach dem Anklicken des Körpers eine Animation entlang des Hamiltonschen Kreises, die bei einer texturierten Fläche endet. Anschließend ist die Landung auf dieser Fläche möglich. Danach wird ein weiterer Director-Film geladen, in dem die Reise durch die virtuelle Homeworld fortgesetzt werden kann.

### Ersetzen der N-Eck-Flächen

In diesem Prototypen wurden erste Experimente mit der Ersetzung der N-Eck-Flächen des Ikosidodekaederstumpfes durchgeführt. Die für die korrekte Position, Rotation und Skalierung benötigten Mittelpunkte und Innenkreisradien der Flächen waren bereits aus dem OpenInventor-Prototypen bekannt und konnten direkt in ein Programm umgesetzt werden. Abbildung 7.8 zeigt die Ergebnisse.



(a) alle Flächen durch Kugeln ersetzt



(b) verschiedene Flächentypen durch verschiedene Modelle ersetzt (Bananen, Kürbisse und Bürostühle), um die korrekte Rotation zu überprüfen

Abbildung 7.8.: Ergebnisse der Ersetzungsexperimente

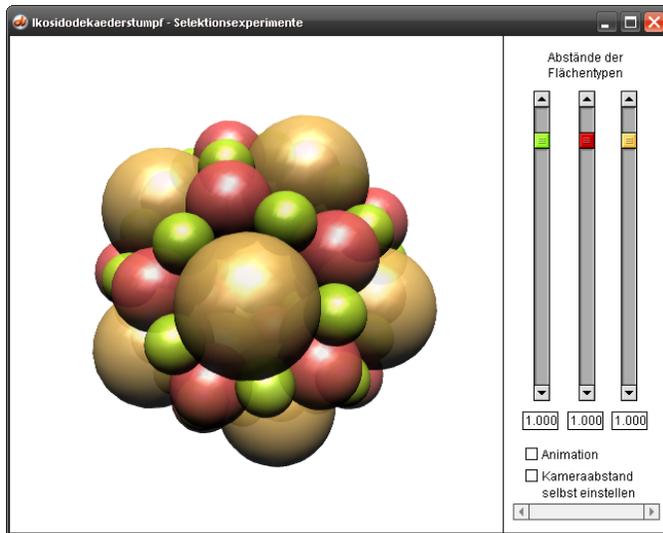
### Ordner auf DVD

Ikosidodekaederstumpf\Director (Web)

### Systemvoraussetzungen

Director MX 2004

## 7.2.4. Selektionsexperimente



**Aufgabe** Fortsetzung der Studien zur Distribution mittels Ikosidodekaederstumpf.

**Bearbeitungszeitraum** Sommersemester 2005

Im Director-Prototypen wurden erstmalig Flächen durch 3D-Modelle, also potentielle Inhalte, ersetzt. Es stellte sich heraus, dass die Belegung aller 62 Flächen zu einer unüberschaubaren Informationsflut führt, die der Benutzer kaum handhaben kann. Daher wurde untersucht, wie sich die Informationen auf effiziente Weise filtern lassen, ohne dabei auf die Belegung aller Flächen zu verzichten.

Da der Ikosidodekaederstumpf aus verschiedenen Typen von Flächen zusammengesetzt ist, bietet es sich an, verschiedene Arten von Informationen oder Objekten den Flächentypen zuzuordnen und jeweils nur einen bestimmten Typ anzuzeigen. Für

den Benutzer wäre es noch praktischer, wenn die Informationen nicht ausgeblendet, sondern nur in ihrer Prägnanz zurückgestellt würden. Der in diesem Prototyp realisierte Informationsfilter signalisiert daher die Priorität durch den Abstand der Flächen eines Typs zum Mittelpunkt des Ikosidodekaederstumpfes. Jeder Flächentyp wird durch eine Kugel repräsentiert, deren Radius dem Umkreisradius der zugehörigen Fläche entspricht. Die Einstellung des Abstandes erfolgt durch Schieberegler, wobei die Farbe jedes Reglers mit den von ihm kontrollierten Kugeln korrespondiert.

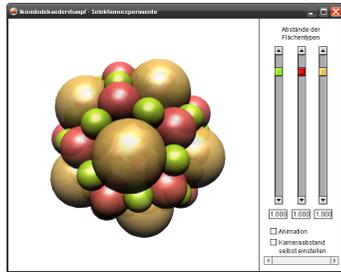
Die Manipulation des Abstandes bewirkt natürlich auch eine Änderung der Ausmaße des gesamten Distributionskörpers. Um dennoch die Übersicht zu behalten, wird der Betrachterstandpunkt gezielt so angepasst, dass der Distributionskörper stets das Anzeigefenster optimal ausfüllt. Der Abstand der Kamera vom Mittelpunkt des Körpers kann aber auch vom Benutzer selbst eingestellt werden. Diese Vorgehensweise ermöglicht viele interessante und nützliche Konstellationen, die eine effiziente Filterung der Inhalte erlauben. Einige davon sind in Abbildung 7.9 zu sehen.

### Ordner auf DVD

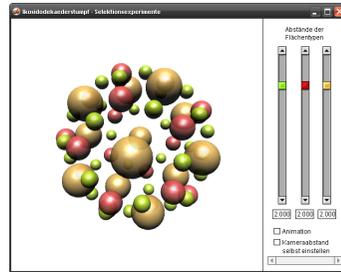
Ikosidodekaederstumpf\Selektionsexperimente

### Systemvoraussetzungen

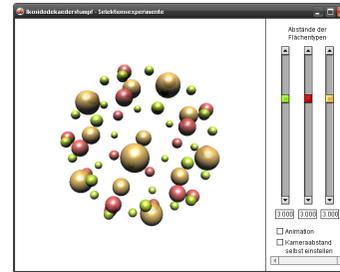
Director MX 2004



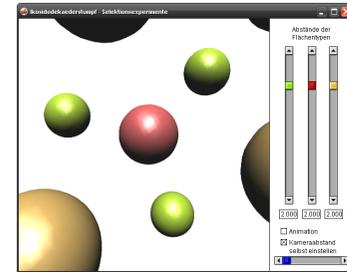
**(a)** Standardabstand der Flächen



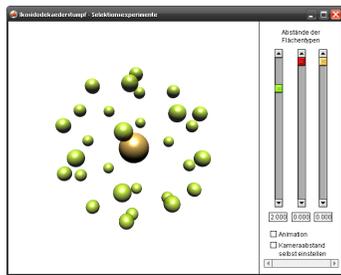
**(b)** zweifacher Abstand der Flächen



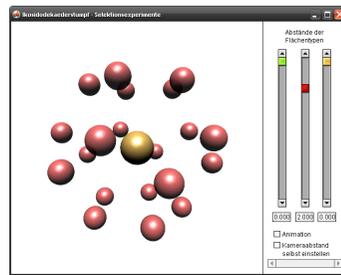
**(c)** dreifacher Abstand der Flächen



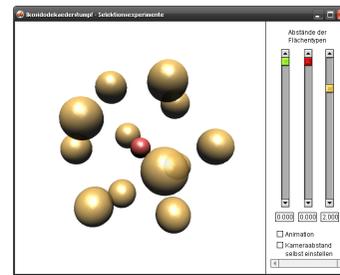
**(d)** zweifacher Abstand der Flächen, Kamera im Mittelpunkt



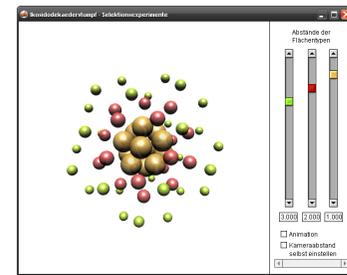
**(e)** kollabierte Sechs- und Zehnecke



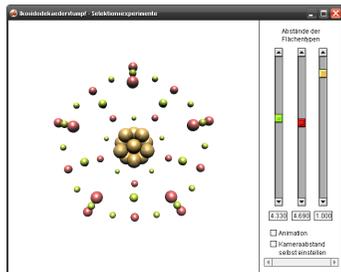
**(f)** kollabierte Vier- und Zehnecke



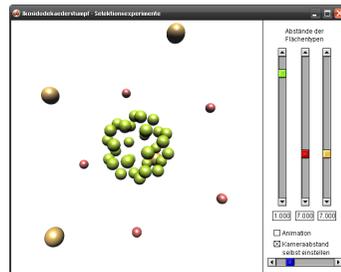
**(g)** kollabierte Vier- und Sechsecke



**(h)** dreifacher, zweifacher und einfacher Abstand



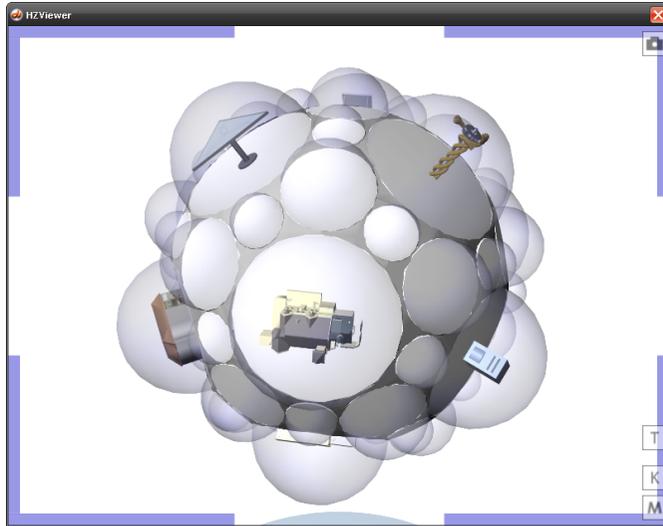
**(i)** Vier- und Sechsecke bilden die Umrisse eines Dodekaeders (in der Animation besser zu erkennen)



**(j)** Vierecke mit einfachem Abstand, Abstand der anderen Flächen größer als Kameraabstand

**Abbildung 7.9.:** verschiedene Konstellationen von Flächen- und Kameraabstand zum Mittelpunkt des Distributionskörpers.

### 7.2.5. Prototyp im Handlungszellenframework



**Aufgabe** Nachbildung und Integration des Distributionskörpers „Ikosidodekaederstumpf“ in das Handlungszellenframework.

**Bearbeitungszeitraum** Sommersemester 2007

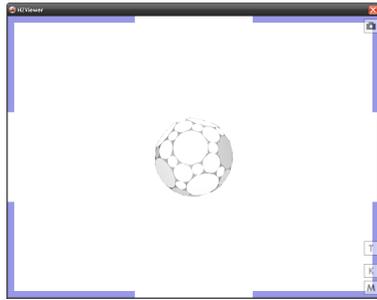
Der zuvor in vielen Einzelprojekten untersuchte Ikosidodekaederstumpf sollte nun in das in Director entwickelte Handlungszellenframework integriert werden. Erstmals können *Handlungszellen* in den Distributionskörper eingebettet, gezielt betrachtet und angesprungen werden.

Die Gestalt des Ikosidodekaederstumpfes wurde in diesem Prototypen leicht abgewandelt, wobei jede N-Eck-Fläche durch ihren einbeschriebenen Kreis ersetzt wurde. Der entstehende Raum zwischen den Flächen bildet das Grundgerüst des Körpers. So ist es selbst bei Ausblendung aller Kreisflächen

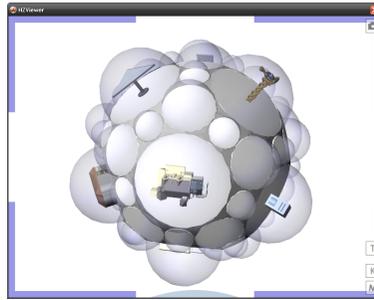
möglich, den abgewandelten Ikosidodekaederstumpf zu erkennen. Da es nicht immer wünschenswert ist, alle Flächen des Distributionskörpers mit Inhalt zu belegen, können nicht benötigte Flächen deaktiviert werden. Die optische Unterscheidung erfolgt anhand von blauen Halbkugeln, die wie Kuppeln über den belegbaren Flächen stehen.

#### Navigation auf dem Distributionskörper

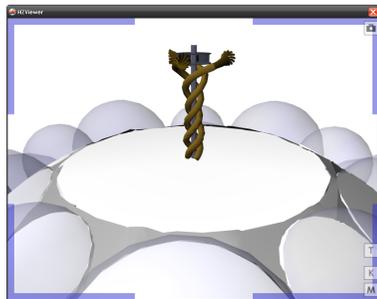
Die Navigation auf dem Distributionskörper kann schrittweise wie in Abbildung 7.10 dargestellt geschehen. Nach dem Andocken kann sich der Benutzer auf der Übersichtssphäre des Körpers bewegen. Das Überschreiten der Pole wird dabei durch ein Kugelconstraint mit eingeschränktem Breitengradintervall verhindert. Bei Auswahl einer Kuppel, wird diese ausgeblendet und die Personalzelle dockt an deren Übersichtssphäre an. Aufgrund der Rotation der Ikosidodekaederstumpfflächen muss das Koordinatensystem der Personalzelle dabei entsprechend angepasst werden. Wie um den Gesamtkörper, so ist auch hier eine sphärische Navigation um die eingebettete Zelle möglich. Allerdings wurden die zugänglichen Breitengrade der Kuppel dabei so eingeschränkt, dass angrenzende Kuppeln nicht durchstoßen werden können. Mit einem Klick auf die eingebettete Zelle wird diese angesprungen. Der Einfluß des Ikosidodekaederstumpf-Distributionskörpers endet mit dieser Aktion. Natürlich kann bei der Verwendung des Distributionskörpers jederzeit ein Kontextklick durchgeführt werden, der einen Rücksprung zur nächsthöheren Zelle der Handlungszellenhierarchie auslöst.



(a) andockt an Außenansicht des Distributionskörpers, eingebettete Zellen sind noch nicht geladen



(b) Distributionskörper „betreten“, eingebettete Zellen geladen



(c) andockt an die Übersichtssichtphäre einer Kuppel mit Blick auf die eingebettete Zelle



(d) andockt an Subzelle

**Abbildung 7.10.:** Schritte bei der Navigation auf dem Distributionskörper

## Der XML-Generator

In technischer Hinsicht wird ein wesentlicher Bestandteil des Distributionskörpers durch XML-Dateien spezifiziert, die vom Handlungszellenframework verarbeitet werden können. Die hohe Flächenzahl des Ikosidodekaederstumpfes führt zu einer hohen Komplexität der XML-Beschreibungen. Um effizient auf Anforderungsänderungen bezüglich des Verhaltens des Distributionskörpers reagieren zu können, wurde ein Generator implementiert, der in der Lage ist, die vollständige XML-Beschreibung eines leeren Ikosidodekaederstumpf-Distributionskörpers zu erzeugen. Die Programmoberfläche des Generators ist in Abbildung 7.11 dargestellt.

Die Zellenbeschreibung des Distributionskörpers wird in der Datei `auswahleder.xml` abgelegt. Alle zu den Flächen gehörenden Daten befinden sich im Ordner `data`. Die Datei `auswahleder.xml` dient als Einstiegspunkt und ist daher bei der Einbettung des Distributionskörpers in andere Handlungszellenzenarien von Bedeutung. Die Einbettung kann entweder durch Editieren der zugehörigen XML-Dateien oder mit Hilfe der Personalzellentoolbox<sup>2</sup> geschehen.

<sup>2</sup>Zum Zeitpunkt der Implementierung des hier beschriebenen Prototypen gab es noch einige technische Schwierigkeiten bei diesem Prozess, die



**Abbildung 7.11.:** Die Oberfläche des Generators, mit dem die XML-Beschreibung eines leeren Ikosidodekaederstumpf-Distributionskörpers erzeugt werden kann.

Der Generator ermöglicht weiterhin die Auswahl der mit Subzellen belegbaren Flächentypen. Beim Export werden nur die belegbaren Flächentypen mit Kuppeln versehen. Die übrigen Flächen können nicht angesprungen werden. Abbildung 7.12 zeigt alle möglichen Distributionsgrundkörper, die der Generator erzeugen kann.

## **Einbettung von Handlungszellen**

Die Einbettung von Handlungszellen in den Distributionskörper kann, wie auch beim Distributionskörper selbst, durch Editieren der XML-Dateien und durch Verwendung der Personalzellentoolbox<sup>3</sup> realisiert werden.

In der zu einer Fläche gehörenden XML-Beschreibung (Verzeichnis data) muss die einzubettende Zelle lediglich in die bereits vorhandene Andockstelle eingefügt werden. Die Skalierung der eingefügten Zelle geschieht automatisch anhand der Formanlage. Handlungszellen, bei denen die Größe der Formanlage und die tatsächlichen Ausmaße der 3D-Modelle nicht harmonieren, können daher zu groß oder zu klein erscheinen.

Mit Hilfe der Personalzellentoolbox kann nach Andocken an eine leere Kuppel einfach eine neue Handlungszelle zur Einbettung ausgewählt werden. Die eingefügte Zelle muss gegebenenfalls noch skaliert und verschoben werden, da die Personalzellentoolbox die bereits vorgefertigte Andockstelle ignoriert und die neue Zelle an der Stelle des Mausklicks mit einer festen Größe einfügt.

---

aber im Handlungszellenframework selbst verwurzelt waren.

<sup>3</sup>Auch hier bereitet das Abspeichern des veränderten Distributionskörpers Probleme.

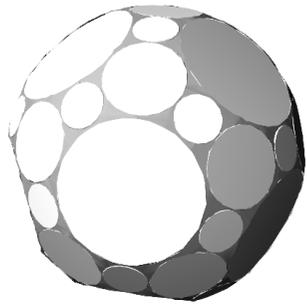
## **Ordner auf DVD**

Ikosidodekaederstumpf\HZFW

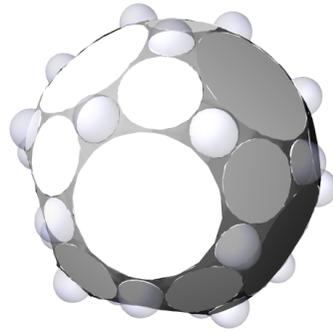
## **Systemvoraussetzungen**

Director MX 2004

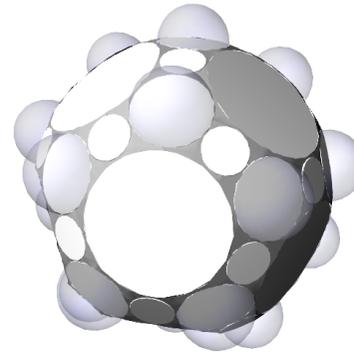
Handlungszellenframework (min. svn-rev. 1856)



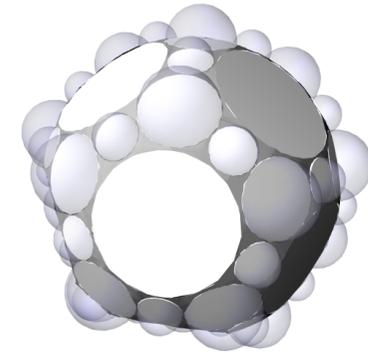
**(a)** keine Flächen belegbar



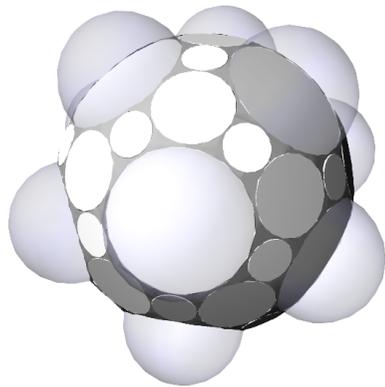
**(b)** 4-Eck-Flächen belegbar



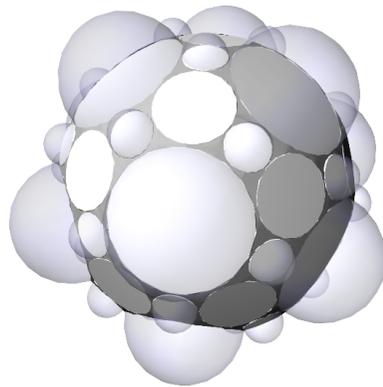
**(c)** 6-Eck-Flächen belegbar



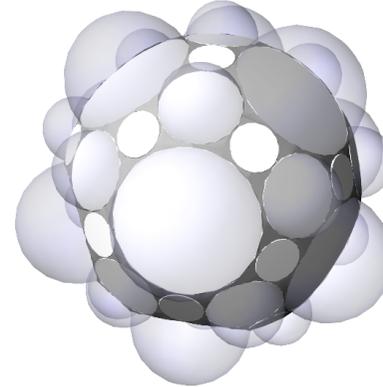
**(d)** 4- und 6-Eck-Flächen belegbar



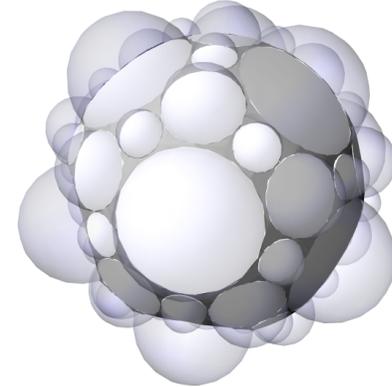
**(e)** 10-Eck-Flächen belegbar



**(f)** 4- und 10-Eck-Flächen belegbar



**(g)** 6- und 10-Eck-Flächen belegbar

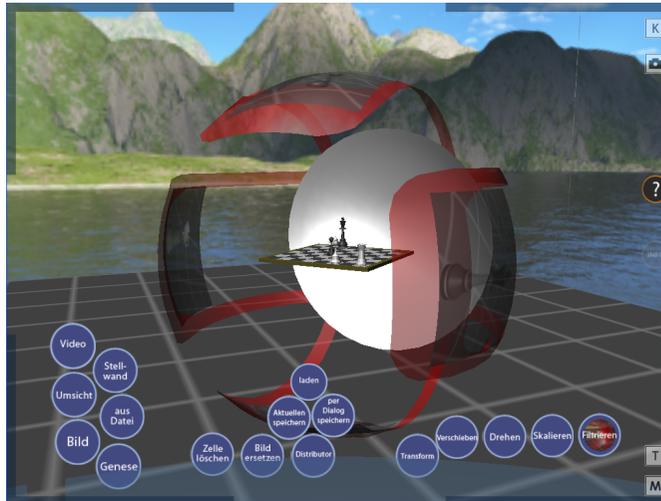


**(h)** alle Flächen belegbar

**Abbildung 7.12.:** Mit dem Generator erzeugte leere Distributionskörper, bei denen jeweils verschiedene Flächentypen mit Inhalt belegt werden können. Dies wird durch die blau gefärbten Kuppeln signalisiert.



## 8. Filtrator



**Aufgabe** Entwicklung eines Werkzeugs innerhalb des Handlungszellenframework (HZFW), das es ermöglicht, Szenen und die Zellen darin zu dekonstruieren, d.h. ihre Komponenten selektiv von der Darstellung auszuschließen, um die Struktur untersuchen zu können. Zusätzlich sollte der sog. VRTC-Testwürfel in das HZFW-XML-Format konvertiert und die Umgebungsdarstellung erweitert werden.

**Bearbeitungszeitraum** März–Juli 2009

### 8.1. Vokabular

Da Filtrator-Werkzeug ist Bestandteil des HFZW, das für Entwickler diverse Schnittstellen zur Steuerung der 3D-Umgebung bietet. Um den Filtrator beschreiben zu können ist es daher notwendig, zunächst eine Einführung in das HZFW-Vokabular zu geben.

**Zelle** Als hierarchischer Grundbaustein der Szene repräsentiert jede Zelle eine Raumeinheit mit 3D-Inhalten, die je eine innere und eine äußere Ansicht über Formanlagen bereitstellen. Diese werden angezeigt, wenn der Benutzer die Zelle entweder *betritt* oder an ihr *andockt*.

**Formanlage** Jede Zelle kann eine Raum- und eine Körper-Formanlage besitzen, die ihre innere bzw. äußere Ansicht mit statischer Geometrie und Interaktionsmöglichkeiten in Form von Andockstellen beschreibt.

**Distributor** Parallel zu den Formanlagen enthält der Distributor einer Zelle Referenzen auf andere Zellen (innerhalb von Andockstellen, die hierarchische oder graphenförmige<sup>1</sup> Einbettungsbeziehungen der Zellen untereinander beschreibt.

**Andockstelle** oder kurz ADS, dienen als Navigationspunkte für den Betrachter, der an ihnen andocken kann, und enthalten Referenzen auf weitere Zellen.

<sup>1</sup>mit Linkzellen

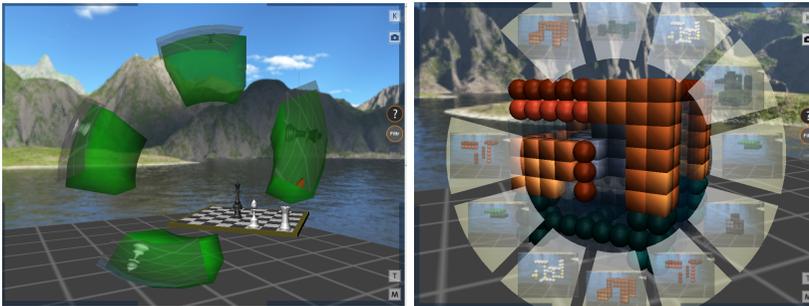
**Geometrie** Referenzen auf 3D-Modelle werden in Geometrie-Elementen beschrieben.

Das HZFW implementiert die oben beschriebenen Elemente als Klassen, die über XML-Dateien parametrisiert werden. Eine detaillierte Beschreibung des XML-Dialekts findet sich in der noch zu erscheinenden Diplomarbeit [Deu09].

## 8.2. Bedienung

Zur Visualisierung des Filtrators wurden zwei verschiedene Ansätze konzipiert und prototypisch umgesetzt, von denen der zweite in der finalen Implementierung umgesetzt ist.

1. Eine Einbettung in die Szene als Zelle, die zwischen die zu filternde Zelle und ihre Umgebung eingeschoben wird.
2. Eine menüförmige Darstellung die am Betrachter fixiert ist.



**Abbildung 8.1.:** Zellen- und Menü-Prototyp

Das Bedienkonzept ist simpel: wird der Filtrator durch Klick auf die Schaltfläche „Filtr“ am rechten Rand aktiviert, werden weitere Schaltflächen eingeblendet. Anklicken der Schaltflächen blendet einige Bestandteile der aktuellen Szene aus und

erlaubt das Untersuchen von verschachtelten oder dicht gepackten Objekten, die dem normalen Betrachter der Szene nicht zugänglich sind.

Da der Filtrator ohne spezielle Vorbereitung auch mit bestehenden Zellen im XML-Format zusammenarbeiten soll, muss er auf bestehende Daten zurückgreifen und sie anschaulich visualisieren. Zusätzlich wird eine Erweiterung des Formats unterstützt, die eine präzisere Kontrolle über die Visualisierung erlaubt.

### 8.2.1. Schnittstelle mit dem HZFW

Die Filterfunktion wird über anzeigen und verstecken der Andockstellen einer Zelle realisiert. Auf diese Weise lassen sich sowohl verteilte als auch dicht gepackte Räume einer Dekomposition unterziehen, um ihre Struktur zu untersuchen. Jeder Schaltfläche des Filtrators wird dabei eine Andockstelle zugeordnet, die daraus auch das Vorschaubild bezieht. Handelt es sich bei der in die ADS eingebetteten Zelle um eine Bilderzelle wird direkt deren Inhalt als Vorschaubild verwendet. Mit dem erweiterten XML-Format können Andockstellen in sogenannten FiltratorOptionen gruppiert und mit gesondertem Vorschaubild versehen werden.

Ist der Filtrator aktiviert, werden beim Andocken bzw. Betreten einer Zelle alle verfügbaren Andockstellen als Schaltflächen dargestellt, die sich damit entweder einzeln oder gruppiert anzeigen und verbergen lassen<sup>2</sup>. Ein Cache verhindert das wiederholte Erzeugen der Schaltflächen für bereits besuchte Zellen.

Der Filtrator merkt sich, welche Zellen der Benutzer mit ihm

<sup>2</sup>Einzel- oder Exklusivauswahl mit Zusatz-Button 1

in der aktuellen Sitzung gefiltert hat. Wird der Filtrator deaktiviert, navigiert der Benutzer wie gewohnt durch mit dem HZFW. Schaltet er den Filtrator erneut ein, stellt dieser den vormaligen Filterzustand wieder her<sup>3</sup>.

Der Benutzer kann wie gewohnt durch die Szene navigieren, wenn der Filtrator aktiviert ist; bei jedem An- und Abdockvorgang werden die Steuerelemente automatisch aktiviert. Enthält eine Szene keine Andockstellen, so wird eine Warnmeldung nur dann ausgegeben, wenn der Benutzer versucht den Filtrator (erneut) zu aktivieren. Die Steuerelemente werden in einer speziellen Cache abgelegt, was unnötige Ladezeiten beim erneuten Andocken oder Betreten einer Zelle vermeidet.

### 8.2.2. FiltratorGuide-XML-Erweiterung

Gruppen von Andockstellen können an Filtrator-Steuerelemente über sogenannte FiltratorGuides zugewiesen werden. FiltratorGuides werden innerhalb des Distributors definiert und enthalten Referenzen auf die Namen der Andockstellen in diesem Distributor. Dazu ist es erforderlich, Andockstellen, die gruppiert werden sollen mit dem Attribute name auszustatten.

Ein Beispiel erläutert die Syntax:

```
<Distributor>
  <Andockstelle name="ads1"> .. </Andockstelle>
  ..
  <FiltratorGuide retain_default_mapping="true">
    <FiltratorOption name="ADS 1 und 2">
      <ADSReference name="ads1" />
      <ADSReference name="ads2" />
    </FiltratorOption>
  </FiltratorGuide>
</Distributor>
```

<sup>3</sup>Löschen der History mit Zusatz-Button 3

```
      <ReferenzBild url="ads1+2.png" />
    </FiltratorOption>
  </FiltratorGuide>
  ..
</Distributor>
```

Hier werden die Andockstellen „ads1“ und „ads2“ gruppiert, und mit einem gesonderten Referenzbild versehen. Die Schaltfläche trägt den Titel „ADS 1 und 2“, der als Tooltip angezeigt wird.

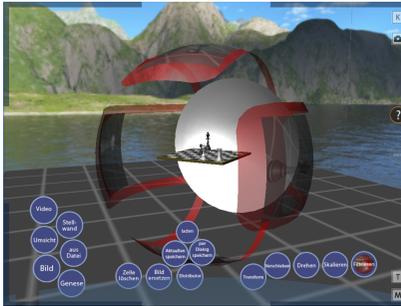
Standardmäßig werden aus allen Andockstellen Schaltflächen generiert. Sobald jedoch ein FiltratorGuide definiert ist, werden Schaltflächen nur aus den enthaltenen FiltratorOptions erzeugt. Ist daher ein leerer FiltratorGuide definiert, werden gar keine Schaltflächen angezeigt. Wird dagegen wie im Beispiel das Attribut retain\_default\_mapping gesetzt, werden auch sämtliche Andockstellen wieder als Schaltflächen dargestellt.

Es ist darauf zu achten, dass die Namen in ADSReference auch dem Namen einer Andockstelle im selben Distributor entsprechen. Der Filtrator validiert vor Erstellung der Schaltflächen die Existenz der Andockstellen und verwirft solche Elemente, die keine Entsprechung haben.

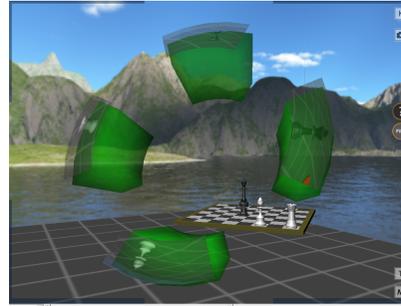
Weiterhin darf nur ein FiltratorGuide pro Distributor definiert werden.

### 8.3. Prototypen

Dieser Abschnitt erläutert die Besonderheiten der verschiedenen Designprototypen auf dem Weg zum finalen Design, und legt die Gründe für die Änderungen am Design dar.



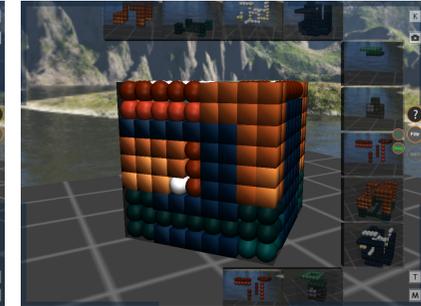
(a) Schalenform



(b) Kegelform



(c) Ringform



(d) Rechteckform

### 8.3.1. Schalenform-Prototyp

Ringförmige Schalen, die die zu filternde Zelle umschließen, und jeweils eine Andockstelle kontrollieren, konstituieren den ersten Prototyp. Augenmerk wurde gerichtet auf die Visualisierung des sichtbar-/unsichtbar Zustandes jeder Andockstelle, was durch die Drehung jeder Schale um die zentrale Achse dargestellt wird. Ist ein Filter aktiviert, so legt sich nach der Metapher der Kamera-Linsenfilter die Schale über die Frontöffnung und gestattet den Blick auf die gefilterte Zelle.

Die Auswahlmöglichkeit zwischen exklusiver (nur 1 Komponente sichtbar) und additiver (Komponenten zuschaltbar) wurde so ebenfalls auf ansprechende Weise umgesetzt.

Der umschließende Charakter des Prototyps behinderte leider seinen universellen Einsatz bei nicht-kompakten und nicht isoliert-stehenden Objekten, wie z.B. auf der Oberfläche eines Tisches.

### 8.3.2. Kegelform-Prototyp

Ein Steuerring aus freischwebenden Schaltflächen über dem Objekt ermöglicht dem zweiten Prototyp, auch nicht-isolierte Objekte zu filtern.

Die Steuerelemente sind ringförmig um den imaginären Sichtstrahl des Betrachters auf seiner Standardandockposition an der Zelle angeordnet. Wie beim ersten Prototyp variiert die Anzahl der Steuerelemente mit der Anzahl der Andockstellen. Zusätzlich konnten Vorschäubilder leicht auf die nun planare Steueroberfläche aufgebracht werden. Weiterhin wird ein farbliches Feedback zur Unterscheidung des exklusiven und additiven Filtermodus eingebaut.

Probleme mit nicht-kompakten Objekten, die sich mit der Filtrator-Geometrie schneiden, bleiben jedoch auch in diesem Prototyp.

### 8.3.3. Ringform-Prototyp

Vollständig kollisionsfrei mit der Szenengeometrie ist der Menüform-Prototyp. Die Steuerelemente sind 3D-Objekte befinden sich jedoch unmittelbar vor der Kamera, so dass es zu keinen Verdeckungen kommt. Nachteilig sind jedoch die mangelnde Kompatibilität mit rechteckigen Vorschaubildern und der eingeschränkte Sichtbereich in der Bildschirmmitte.

### 8.3.4. Rechteckform-Prototyp

Eine effektive Ausnutzung des Bildschirmplatzes und eine potentiell unbegrenzte Anzahl von Steuerelementen bietet die Rechteckform. Die Vorschaubilder werden spiralförmig angeordnet und pro Spiralwindung in der Größe halbiert, so dass sich entlang der geometrischer skalierten Spiralschalen eine unendliche Anzahl auf konstantem Platz unterbringen lassen. Das innere der Spirale erlaubt genügend Platz, um durch Szenarien bei eingeschaltetem Filtrator zu navigieren und Details zu betrachten. Die schlichten rechteckigen Schaltflächen erlauben eine verzerrungsfreie Darstellung der gängigen Referenzbilder, wie sie für Zellen angelegt sind.

Der Rechteckform-Prototyp ist in der finalen Version implementiert. Der Ringform-Prototyp und andere lassen sich durch Ändern des Attributs `layout_pattern` in der PersonalZell-XML-Datei auf das in der Klasse `FiltrationsManipulatorLayoutState` definierte Layout wiederherstellen. Mit etwas Aufwand (ändern der PersonalZell-XML-Datei und Austauschen der Shockwave-Modelldateien), lassen sich auch die anderen Prototypen nachträglich wieder in Betrieb nehmen.

## 8.4. Menüleiste

Eine Menüleiste neben dem „Filtr“-Button gibt Zugriff auf Meta-Steuerelemente mit Funktionen wie folgt:



Umschalten zwischen additivem und exklusivem Auswahlmodus. (Standard: exklusiv) Das Umschalten zwischen beiden Modi ist durch geänderte Farbgebung gekennzeichnet. Im additiven Modus wird jede ADS einzeln an und ausgeschaltet; im exklusiven Modus wird stets nur die zuletzt ausgewählte ADS angezeigt.



Objekt-Modus: Szene wird durch statischen Hintergrund ersetzt; Tastatursteuerung wird auf Objekt übertragen. Ein kontrastreicher Hintergrund stellt in der Szene schwach beleuchtete Objektteile besser heraus.



Löschen der Filterkonfiguration: alle Andockstellen werden wieder sichtbar gemacht.

## 8.5. Programmdokumentation Filtrator

Der Arbeitsablauf zum Konvertieren von Inventor-IV-Dateien in ein HZFW-kompatibles Format, die Beschreibung der Programmierschnittstelle mit Erläuterungen zu Methoden in Prosaform, sowie das Kontrollflussdiagramm für sämtliche Filtrator-Klassen wird beschrieben in der Dokumentation für Programmierer in Anhang B.3.

## 8.6. Fazit

Der Filtrator ist ein Werkzeug dar, das noch eher als zur Manipulation, als optisches Instrument zum Erkunden des Innenlebens einer Szene eingesetzt werden kann. Für Debugging-Zwecke eignet es sich, wenn die Entwickler-Hotkeys zur Anzeige der Distributorstangen (Zifferntasten 1–3) in zu dicht gepackten Szenen keinen Überblick mehr bieten. Da das HZFW bereits Werkzeuge zum Löschen von Szenenelementen kennt ist diese Funktionalität im Filtrator auch gar nicht notwendig. Leider ist die Versorgung mit ReferenzBildern seitens der Zell-Entwickler noch dürftig, so dass das Vorschaubild-Feature oftmals leer ausgeht. Aber auch für diesen Fall sorgt die mitgelieferte Tooltip-Klasse für Orientierung. Alles in allem ein kompaktes kleines Helferlein.

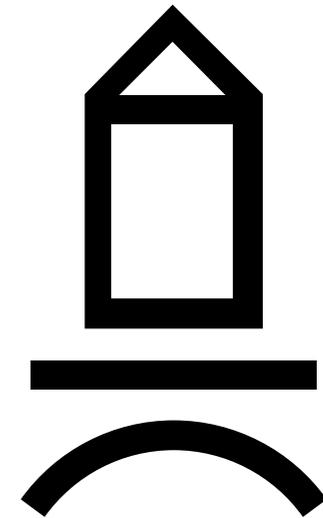
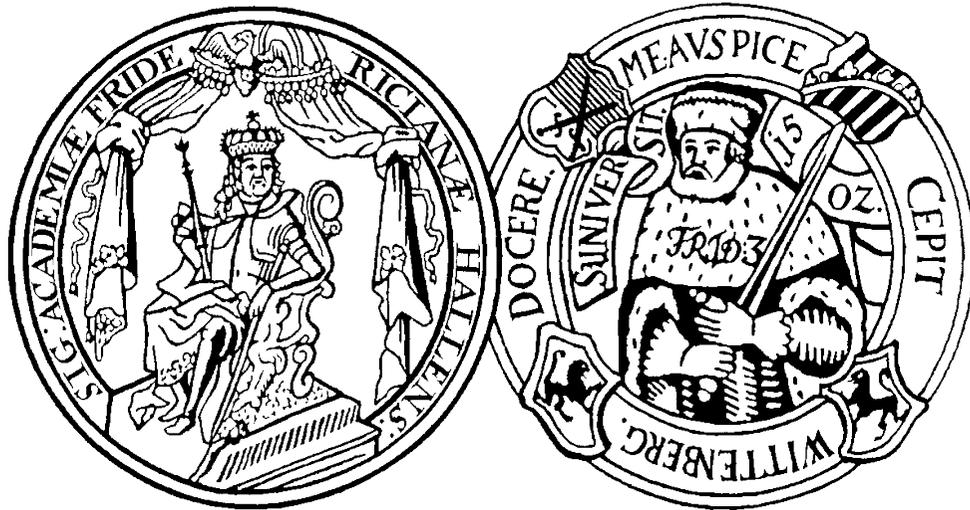
## A. Quellenverzeichnis

- [Bre07] *Brefeld, Werner: Platonische Körper und Archimedische Körper.* Version: 2007. <http://www.brefeld.homepage.t-online.de/polyeder.html>
- [Coma] *Community, Wikipedia (Hrsg.): Archimedean Solid.* [http://en.wikipedia.org/wiki/Archimedean\\_solid](http://en.wikipedia.org/wiki/Archimedean_solid), Abruf: 26. Oktober 2007. Wikipedia
- [Comb] *Community, Wikipedia (Hrsg.): Platonic Solid.* [http://en.wikipedia.org/wiki/Platonic\\_solid](http://en.wikipedia.org/wiki/Platonic_solid), Abruf: 26. Oktober 2007. Wikipedia
- [Deu09] *Deutschel, Marcel: Kurzbeschreibung des MM/VR-Conception-Studienganges.* erscheint 2009 Martin-Luther-Universität Halle
- [GHJV04] *Gamma, E. ; Helm, R. ; Johnson, R. ; Vlissides, J.: Entwurfsmuster.* 1. Addison-Wesley, 2004. – ISBN 978-3827321992
- [Har96] *Hart, George W.: Archimedean Semi-Regular Polyhedra.* Version: 1996. <http://www.georgehart.com/virtual-polyhedra/archimedean-info.html>, Abruf: 26. Oktober 2007. Virtual Polyhedra
- [Kol] *Kolbe, Peter: Kurzbeschreibung des MM/VR-Conception-Studienganges.* <http://www.burg-halle.de/mvc-kurzinfo.html>, Abruf: 26. Oktober 2007
- [Wei03] *Weisstein, Eric W.: Great Rhombicosidodecahedron.* Version: 2003. <http://mathworld.wolfram.com/GreatRhombicosidodecahedron.html>, Abruf: 26. Oktober 2007. Mathworld - A Wolfram Web Resource
- [Wei07a] *Weisstein, Eric W.: Archimedean Dual Graph.* Version: 2007. <http://mathworld.wolfram.com/ArchimedeanDualGraph.html>, Abruf: 26. Oktober 2007. Mathworld - A Wolfram Web Resource
- [Wei07b] *Weisstein, Eric W.: Hamiltonian Circuit.* Version: 2007. <http://mathworld.wolfram.com/HamiltonianCircuit.html>, Abruf: 26. Oktober 2007. Mathworld - A Wolfram Web Resource

Eine heitere

# Konzeption einer virtuellen Homeworld

„The haarsträubenden Adventures of Max Mustermaier“



Sebastian Wendt

Christian Stussak

Sebastian Bauer

Entstanden im Wintersemester 2003/2004

# Inhaltsverzeichnis

1	Einstieg in die virtuelle Homeworld	3
2	Anmeldung und Betreten der virtuellen Hochschullandschaft	6
3	Mitnehmen von Materialien	8

# Kapitel 1

## Einstieg in die virtuelle Homeworld

Hallo, mein Name ist Max Mustermaier und ich bin Studienanfänger der Informatik. Ich bin leicht verwirrt, da dies mein erster Tag an der Martin-Luther-Universität zu Halle ist.

Jedoch bin ich zuversichtlich, da mir der Hochschulbrowser zu meiner Orientierung zur Seite steht. Ich betrete nun die virtuelle Hochschullandschaft und steuere mein erstes Ziel, den Fachbereich Informatik, an welchem ich studieren werde, an. Dazu navigiere ich mit meiner Spacemouse (wahlweise auch Sidewinder-Joystick, Tastatur oder Maus) in das Universitätsgebilde und erkenne dort den Fachbereich Informatik an seinem fachspezifischen Aussehen (Design abstrakt).

Sofort springt mir ein **Portal (Verweis)** ins Auge, welches mich auf eine Erstsemesterorientierungsveranstaltung hinweist. Ein Klick darauf führt mich sogleich hindurch.

Ich gelange an einen virtuellen Vorlesungsraum, in dem ein Vorlesender die virtuelle Studierendenschaft begrüßt. Mein Home-room wies mich auf die laufende Veranstaltung hin. Der Vorlesungsraum bietet einen virtuellen Parkplatz für die Homerooms der Zuhörer an. **Ich sehe mich auf dem Parkplatz vor dem**

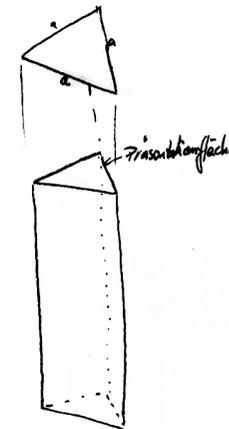


Abbildung 1.1: Ein Prismen-Portal

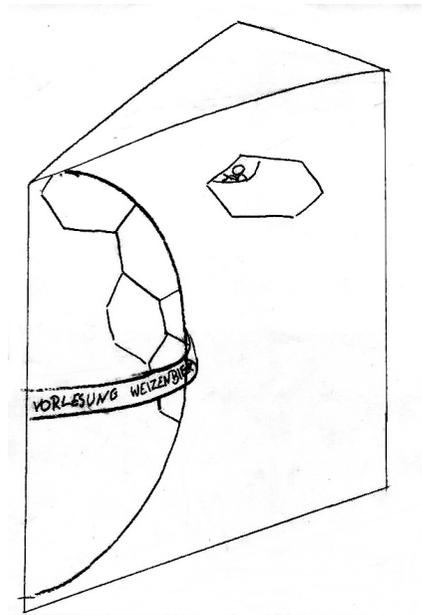


Abbildung 1.2: Blick durch ein Portal

Vorlesungsraum um und bemerke, dass sich zahlreiche weitere Studienanfänger eingefunden haben, um seinen Ausführungen zu lauschen. Mein Homeroom ist, genau wie die anderen, nur auf diesem Parkplatz für andere sichtbar, was dem Schutz der Privatsphäre dient.

Ich steuere meinen Homeroom in die Nähe einer **größeren Traube** meiner zukünftigen Mitstudenten und begrüße sie freundlich, indem ich in meine Kamera winke, deren **Bild auf die Oberfläche meines Homerooms projiziert** wird. Danach wende ich mich der Vorlesung zu.

Während ich den Ausführungen des Vortragenden folge, **speichert** mein Homeroom die in seinen **Materialien** enthaltenen

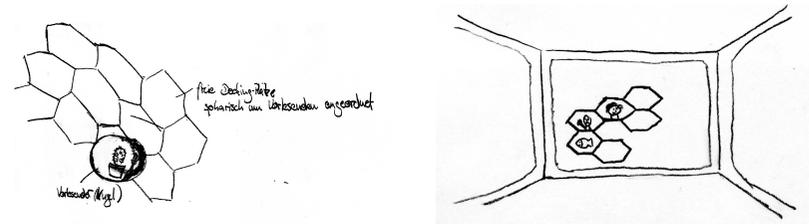


Abbildung 1.3: Andock- und Begrüßungsszenario

Verweise für mich zur späteren Ansicht. Als er zum Geplänkel über Bioinformatik übergeht, wechsele ich zu einem anderen Fenster, um mir die gespeicherten Verweise anzusehen. Diese sind vorerst **in einer Verweislandschaft** zu einem Baum organisiert, können aber von mir beliebig umorganisiert werden. Die äußere Erscheinung eines Verweises ruft mir dessen Ziel wieder ins Gedächtnis.

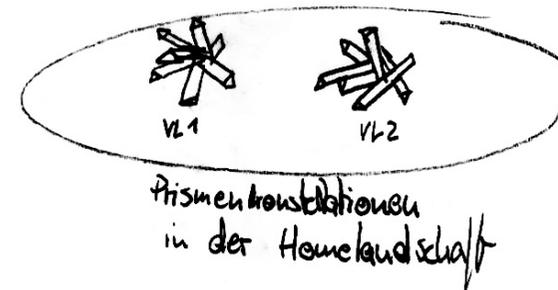


Abbildung 1.4: Verweisansammlungen verschiedener Vorlesungen

Während ich die Materialien der Veranstaltung durchstöbere, kommen mir einige Fragen. Angegliedert ist ein Hinweis auf den Konsultationsraum der Veranstaltung, zu welchem ich mich



flugs begeben. Zu meinem Glück findet dort gerade eine Konsultation statt, wie auch im angegliederten Hinweis vermerkt. Ich **docke mich an den Konsultationsraum an und reihe mich in die Schlange der wartenden Konsultanten** ein. Nach einigen Minuten drögen Wartens flüstert mir mein Vordermann, lädt mich auf eine Partie Homeroom-Tetris ein.

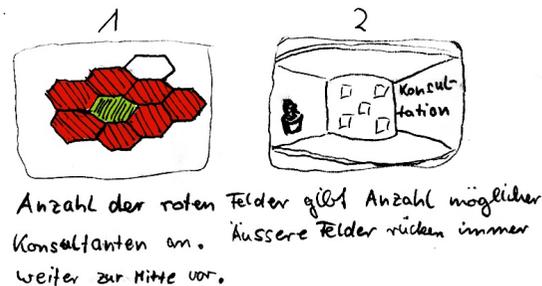


Abbildung 1.5: Andock- (1) und Innenansicht (2) eines Konsultationsraums

Schließlich bin auch ich an der Reihe, meine Fragen mit dem Konsultationsleiter zu erörtern. Ich (Stele, 3D-Modell) erscheine nun auch im Konsultationsraum, so dass mich die anderen Konsultanten sehen und mit mir diskutieren können. In einem anderen Fenster meines Homerooms wähle ich mir einige Materialien (beliebige Medien) aus und bringe diese in die Mitte des Konsultationsraums. Ein Mitstudent erörtert mit mir anhand einer kleinen Zeichnung auf dem Paintboard das Problem. Damit ist meine Frage beantwortet und ich ziehe mich in meinen Homeroom zurück.

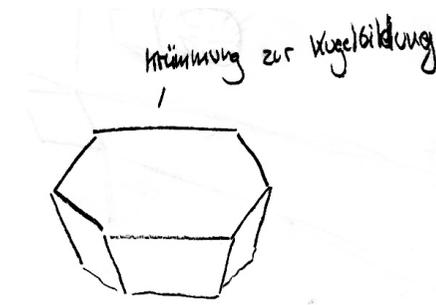


Abbildung 1.6: Angeschrägte Homeroomwände für Zusammenschluss mehrerer Homerooms zu einem kugelartigen Gebilde

## Kapitel 2

# Anmeldung und Betreten der virtuellen Hochschullandschaft

Nach dem Start des Hochschulbrowsers befinde ich mich in einem **Gast-Homeroom**. Dieser besitzt eine Standardeinrichtung, die ich nicht personalisieren kann. Dazu gehört ein Anmeldeboard und eine Ablagemappe für Materialien. Es ist schön, dass das Anmeldeboard so leicht zugänglich ist. **Ich trage meine persönlichen Daten in das Anmeldeboard ein.** Bei Schwierigkeiten hilft eine nette Dame des Immatrikulationsamtes. Sie eröffnet auf meine Anfrage hin einen Konsultationsraum um mit mir das Problem zu erörtern.

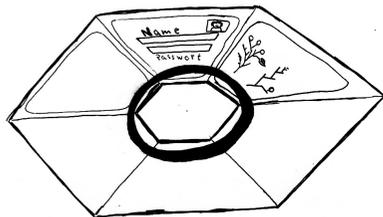


Abbildung 2.1: Draufsicht des Homerooms mit drehbarem Ablageband

Die Bearbeitung meiner Anmeldung dauert nun einige Zeit. Währenddessen schaue ich mich mit dem Gasthomeroom etwas in der Hochschullandschaft um. Ich sehe einen Vorlesungsraum mit einer scheinbar interessanten Veranstaltung und fliege direkt darauf zu. Als ich versuche mich anzudocken, **pralle ich vom der Andockstelle ab** und es erscheint ein Hinweis, dass zu dieser Vorlesung eine **Identifizierung erforderlich** ist.

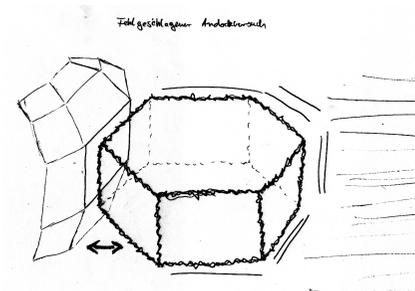


Abbildung 2.2: Abprallen des Homerooms nach fehlgeschlagenem Andockversuch

Mit frischem Mut fliege ich weiter und treffe auf einen **frei zugänglichen Chat-Room**. Ich unterhalte mich munter mit anderen Studierenden, die mir alsbald alle Möglichkeiten des Gast-Homeroms eröffnen. Sie erzählten mir, dass ich alle öffentlich zugänglichen Veranstaltungen besuchen kann und sogar für die Dauer eines Besuchs Materialien und Verweise speichern kann. Nun erhalte ich eine Mitteilung von der netten Dame des Imma-Amtes. Sie weist mich freundlich darauf hin, dass die Anmeldung nun abgeschlossen und mein persönlicher Homeroom einsatzbereit ist. Sofort bewege ich mich im Gast-Homeroom zum Anmeldeboard und vollführe den Login.

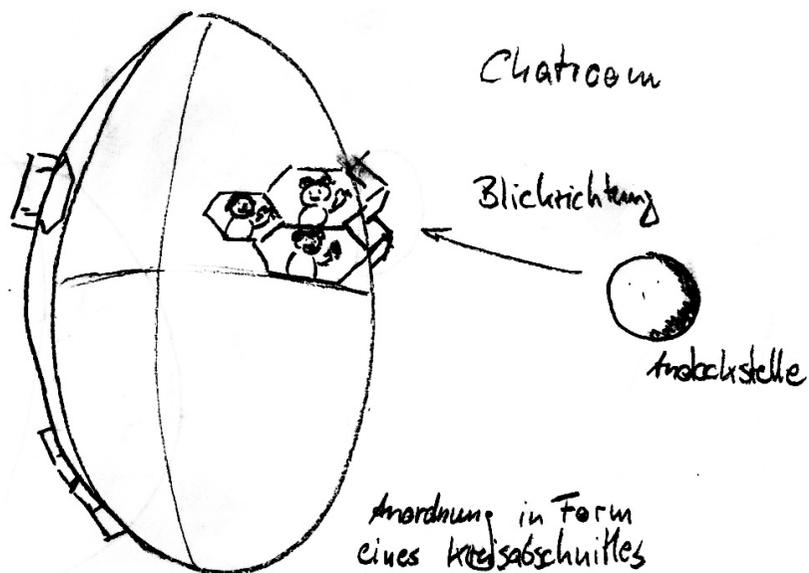


Abbildung 2.3: Mögliche Anordnung der Homerooms in einem frei zugänglichen Chatroom



# Kapitel 3

## Mitnehmen von Materialien

Heute besuche ich eine Konsultation. Mein Lieblingsprofessor, Prof. Rhino, bietet sie ergänzend zu seiner Vorlesung „Angewandte abstrakte Algebren“ an. Da sehr viel Stoff in der Vorlesung abgehandelt wurde, nahm ich einige abstrakte Algebren aus der Vorlesung mit, um sie zu besprechen. Die Algebren befinden sich momentan als **Wissenseinheiten, dargestellt in kleinen, halbtransparenten Quadern**, vor mir auf dem Bildschirm. Sie sind angeordnet auf einer Art **Fensterbrett unterhalb eines jeden Fensters** meines Homerooms.

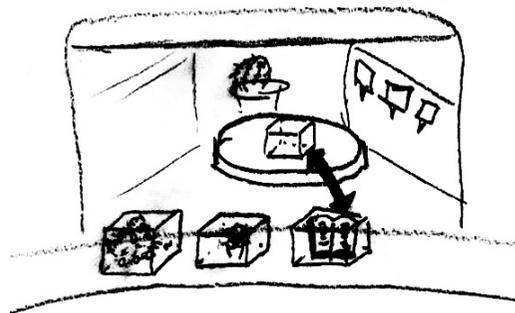


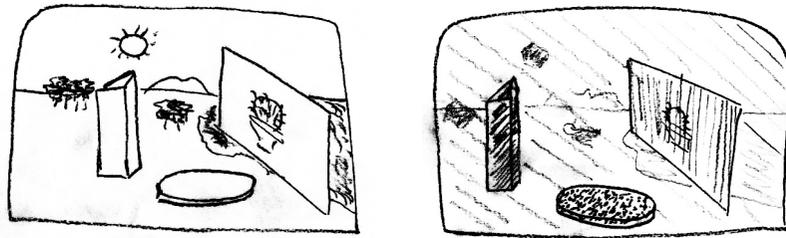
Abbildung 3.1: Ablegen einer Wissenseinheit auf ReadWrite-Ablagefeld des Konsultationsraums

Die beidseitige Unendlichkeit dieses Ablagebandes bietet selbst für ausgedehnte Besprechungen genügend Stauraum. Nun bin ich an der Reihe, meine Fragen im Konsultationsraum zu stellen.

Ich hebe eine Algebra von meinem Ablageband und ziehe sie auf einen speziellen Bereich im Konsultationsraum, eine sogenannte **ReadWrite-Zone**. Die Zone, hier in Form eines flachen Podestes, übernimmt die Referenz von mir oder, wenn vorhanden, eine Wertekopie des Objektes. Sobald der Kopiervorgang abgeschlossen ist wird die Algebra dargestellt. Wir besprechen die Algebra eine Weile und klären meine Fragen ihrbezüglich.

Damit der nächste Konsultant an die Reihe kommen kann, entferne ich die Kopie meiner Algebra aus der ReadWrite-Zone durch einen einfachen Rechtsklick. (Jetzt neu: **Papierkorb™**)

Die erlangten Erkenntnisse notiere ich mir in meinem (echten!) Hefter, weswegen ich die speicherintensive Wertekopie der Algebra nicht mehr in meinem Homeroom aufbewahren will. Am, die Algebra repräsentierenden, Objekt befindet sich ein Indikator, der mir anzeigt, dass die Algebra momentan als Wertekopie



Recht	Farbe	interne Codierung
keine Rechte	Dimmen mit Grau	0
Ausführen	Rot	1
Schreiben	Grün	2
Lesen	Blau	4

Das Leserecht auf die übergeordnete Umgebung bestimmt die Sichtbarkeit eines Objektes. Zur Rechtekombination wird eine additive Farbmischung genutzt (z.B. Schreiben und Ausführen mischt Grün und Rot  $\Rightarrow$  Gelb).

Abbildung 3.2: Farbcodierte Anzeige der Zugriffsrechte (hier durch verschiedene Schraffierungen) auf Tastendruck

vorliegt. Ich klicke auf den **Indikator**, worauf dieser seine Erscheinung ändert und den **Werteteil** der Wissensinheit verwirft. Den **Referenzteil**, also die Referenz auf die Quelle, von der ich zuerst die Wertekopie aufnahm, bleibt mir erhalten. So kann ich z.B. später zur Prüfungsvorbereitung noch einmal auf die Algebra zugreifen.

Dazu würde ich dann die Referenz auf einen **Objekt-Betrachter**, wie er auch in meiner Heimlandschaft installiert ist, übertragen. Er lüde dann mit Hilfe der Referenz eine Wertekopie der Algebra, so dass sie angezeigt werden könnte. Bis es soweit ist, lagert die Referenz in meiner Heimlandschaft, an

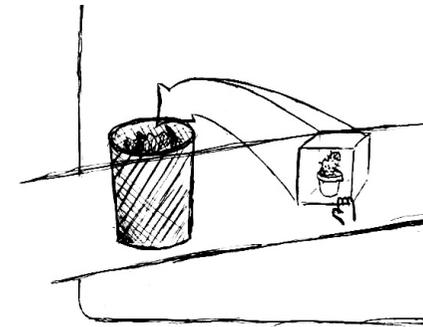


Abbildung 3.3: „Wegwerfen“ von Materialien in Papierkorb™

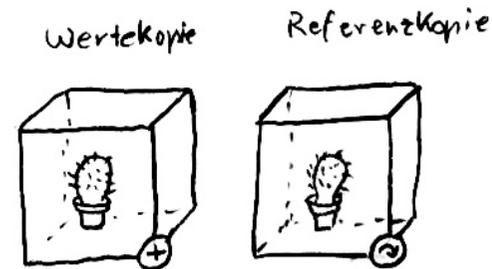


Abbildung 3.4: Unterschiedliche Darstellungen für Werte- und Referenzkopien

meinem Ablagebaum für Vorlesungsmaterialien aus dem ersten Semester.





Abbildung 3.5: Ablagebaum für Vorlesungsmaterialien



# Ausgewählte Eigenschaften platonischer Körper

Christian Stussak Sebastian Wendt

26. Oktober 2004

## Inhaltsverzeichnis

1	Einleitende Worte	1
2	Tetraeder	2
3	Hexaeder	3
4	Oktaeder	4
5	Dodeaeder	4
6	Ikosaeder	7
7	Ausblick	8

## 1 Einleitende Worte

In dieser Ausarbeitung sollen ausgewählte Eigenschaften platonischer Körper näher beleuchtet werden. Insbesondere sind die Zusammenhänge Kantenlänge - Innenkugelradius sowie Kantenlänge - Außenkugelradius von Interesse. Zuerst wollen wir jedoch die platonischen Körper allgemein vorstellen.

Platonische Körper sind reguläre konvexe Polyeder (Vielflächner). Ein Polyeder ist regulär, wenn alle seine Oberflächen aus demselben regelmäßigen Vieleck bestehen und in jeder Ecke gleich viele dieser Vielecke zusammenstoßen. Zur Vollständigkeit sei hier noch erwähnt, dass regelmäßige Vielecke ausschließlich von Kanten gleicher Länge begrenzt werden und der Winkel zwischen zwei benachbarten Kanten immer derselbe ( $< 180^\circ$ ) ist.

Obwohl die Körper nach Platon benannt wurden, waren es doch andere Mathematiker bzw. Philosophen vor ihm, die feststellten, dass es nur genau fünf regulär konvexe Polyeder gibt, nämlich Tetraeder (vier gleiche gleichseitige Dreiecke), Hexaeder (vier gleiche Quadrate), Oktaeder (acht gleiche gleichseitige Dreiecke), Dodekaeder (zwölf gleiche regelmäßige Fünfecke) und Ikosaeder (zwanzig gleiche gleichseitige Dreiecke).

In jeder Ecke eines Polyeders müssen mindestens drei Vielecke zusammenstoßen um eine räumliche Ecke zu bilden. Da andererseits das reguläre Polyeder konvex ist, muss die gesamte Winkelsumme aller  $n$ -Ecke, die in jeder Körperecke zusammenstoßen, stets echt kleiner als  $360^\circ$  sein. Es können also nur 3, 4 oder 5 regelmäßige Dreiecke, 3 Quadrate oder 3 regelmäßige Fünfecke sein. Diese fünf möglichen Fälle lassen sich aber durch die oben angegebenen Körper realisieren.

Jeder platonische Körper besitzt eine Innenkugel mit Radius  $r$ , auf der die Mittelpunkte sämtlicher Flächen des Körpers liegen, und eine Außenkugel mit Radius  $R$ , auf der sämtliche Körperecken liegen. Im folgenden werden wir  $r$  und  $R$  in Abhängigkeit von der Kantenlänge  $a$  berechnen.

## 2 Tetraeder

Zur Berechnung der Radien betrachten wir die Schnittebene durch den Tetraeder. Die eingezeichnete Kante  $\overline{M_1E_1}$  mit der Länge  $h = \frac{\sqrt{3}}{2}a$  ist gleichzeitig die Höhe eines der gleichseitigen Dreiecke. Der Schwerpunkt eines solchen Dreiecks, der wiederum Berührungspunkt der Innenkugel ist, liegt gerade im Schnittpunkt aller Seitenhöhen, also bei  $\frac{2}{3}h$  im Punkt  $N$ .

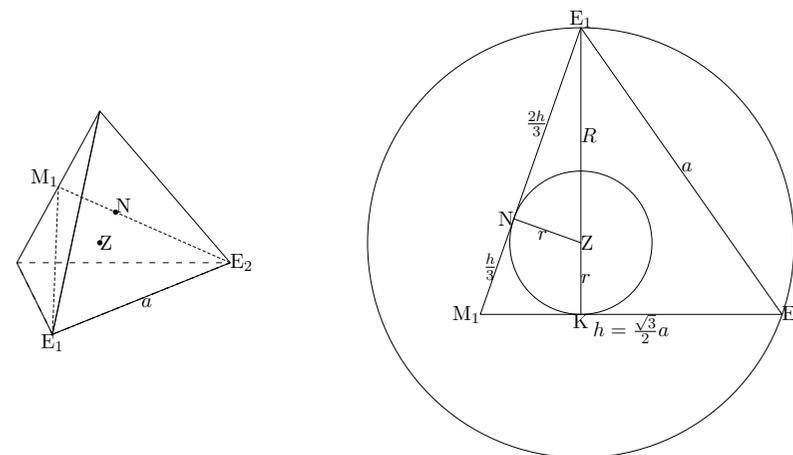


Abbildung 2.1: Tetraeder und zugehöriger Schnitt

Wir können nun mit Hilfe des Satzes des Pythagoras folgende Gleichungen aufstellen:

$$\triangle ZNE_1 : \quad R^2 = r^2 + \left(\frac{2}{3}h\right)^2 \quad (2.1)$$

$$\triangle M_1KE_1 : \quad h^2 = (R+r)^2 + \left(\frac{h}{3}\right)^2 \quad (2.2)$$

Gleichung (2.1) lässt sich nach  $R$  umstellen und Gleichung (2.2) können wir vereinfachen.

$$R = \sqrt{r^2 + \left(\frac{2}{3}h\right)^2} = \sqrt{r^2 + \frac{a^2}{3}} \quad (2.3)$$

$$(R+r)^2 = \frac{3}{4}a^2 - \frac{1}{9} \cdot \frac{3}{4}a^2 = \frac{2}{3}a^2 \quad (2.4)$$

$$\Rightarrow R = \sqrt{\frac{2}{3}a} - r \quad (2.5)$$

Wir setzen (2.3) mit (2.5) gleich und stellen nach  $r$  um.

$$\left(\sqrt{\frac{2}{3}a} - r\right)^2 = \frac{2}{3}a^2 - 2\sqrt{\frac{2}{3}a}r + r^2 = r^2 + \frac{a^2}{3} \quad (2.6)$$

$$r = \frac{\sqrt{6}}{12}a \quad (2.7)$$

Mit (2.5) folgt nun auch der Außenkugelradius  $R$ .

$$R = \frac{\sqrt{6}}{4}a \quad (2.8)$$

### 3 Hexaeder

Die Eigenschaften des Hexaeders (Würfel) lassen sich besonders einfach erläutern. Der Innenku-

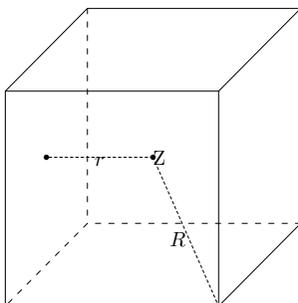


Abbildung 3.1: Hexaeder

gelradius ist die Hälfte des Abstandes zweier gegenüberliegender Seiten.

$$r = \frac{a}{2} \quad (3.1)$$

Der Abstand vom Körpermittelpunkt  $(0,0,0)^T$  zu einer Ecke, z.B.  $(\frac{a}{2}, \frac{a}{2}, \frac{a}{2})^T$ , bildet den Außenkugelradius. Dieser beträgt  $\sqrt{\frac{a^2}{4} + \frac{a^2}{4} + \frac{a^2}{4}} = \frac{\sqrt{3}}{2}a$ , womit  $R$  folgt.

$$R = \frac{\sqrt{3}}{2}a \quad (3.2)$$

### 4 Oktaeder

Betrachten wir in Abbildung 4.1 das Dreieck mit den Ecken  $E_1$ ,  $Z$  und  $E_3$ . Wir erkennen, dass

$$a^2 = R^2 + R^2 = 2R^2 \quad (4.1)$$

und damit:

$$R = \frac{a}{\sqrt{2}} \quad (4.2)$$

Der Innenkugelradius ist der Abstand des Koordinatenursprungs  $Z$  zu einer Seitenfläche, sagen wir zum Dreieck  $E_2E_3E_4$ . Dieses Dreieck spannt eine Ebene mit der Formel

$$(\vec{v} - \vec{v}_0) \cdot \vec{n}_0 = \left(\vec{v} - \begin{pmatrix} \frac{a}{\sqrt{2}} \\ 0 \\ 0 \end{pmatrix}\right) \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \frac{1}{\sqrt{3}} = 0 \quad (4.3)$$

auf. Dies folgt, wenn wir  $\vec{v}_0 = E_4$  als Punkt der Ebene nutzen und  $\vec{n}_0 = \frac{E_2+E_3+E_4}{|E_2+E_3+E_4|}$  als Normalenvektor annehmen. Für den Abstand einer Ebene vom Ursprung gilt

$$|\vec{v}_0 \cdot \vec{n}_0| = \left|a \cdot \frac{1}{\sqrt{6}} + 0 + 0\right| = \frac{a}{\sqrt{6}}, \quad (4.4)$$

womit  $r$  folgt.

$$r = \frac{a}{\sqrt{6}} \quad (4.5)$$

### 5 Dodeaeder

Der Dodekaeder ist wohl der am schwersten zu berechnende platonische Körper. Dies liegt hauptsächlich an den pentagonalen Seitenflächen. Aus diesem Grund werden wir auch mit den Seitenflächen beginnen.

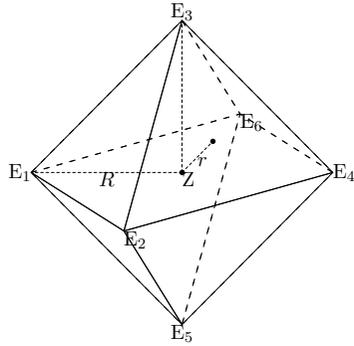


Abbildung 4.1: Oktaeder

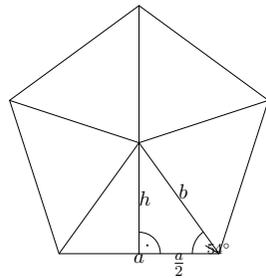


Abbildung 5.1: Pentagon

Aus den Winkelsätzen im rechtwinkligen Dreieck ergeben sich  $b$  und  $h$ .

$$b = \frac{a}{2} \tan 54^\circ = \frac{a}{2} \cdot \frac{1 + \sqrt{5}}{\sqrt{10 + 2\sqrt{5}}} \quad (5.1)$$

$$h = \frac{a}{2 \cos 54^\circ} = a \frac{2}{\sqrt{10 - 2\sqrt{5}}} \quad (5.2)$$

Legen wir eine Ebene geeignet in den Dodekaeder, so erhalten wir Abbildung 5.3.

Die rechtwinkligen Dreiecke liefern uns wieder ein Gleichungssystem.

$$\triangle ZNE_1 : \quad R^2 = r^2 + b^2 \quad (5.3)$$

$$\triangle ZM_3E_1 : \quad R^2 = x^2 + \left(\frac{a}{2}\right)^2 \quad (5.4)$$

$$\triangle E_1KM_1 : \quad (b + h)^2 = x^2 + \left(x - \frac{a}{2}\right)^2 \quad (5.5)$$

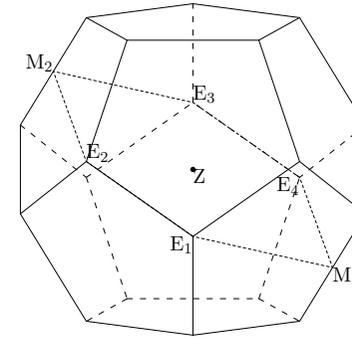


Abbildung 5.2: Dodekaeder

Gleichung (5.5) können wir mithilfe der Lösungsformel für Nullstellen quadratischer Gleichungen umstellen nach

$$x = \frac{a}{4} + \frac{1}{4} \sqrt{-a^2 + 16bh + 6h^2 + 8b^2} \quad (5.6)$$

und die Formeln für  $b$  und  $h$  einsetzen.

$$x = \frac{a}{4} + \frac{1}{4} \sqrt{-a^2 + 16 \frac{1 + \sqrt{5}}{10 + 2\sqrt{5}} a^2 + \frac{24}{10 - 2\sqrt{5}} a^2 + \frac{6}{4} \cdot \frac{6 + 2\sqrt{5}}{10 + 2\sqrt{5}} a^2} \quad (5.7)$$

Durch langwieriges Vereinfachen erhalten wir

$$x = \frac{3 + \sqrt{5}}{4} a. \quad (5.8)$$

Wir setzen  $x$  in (5.4) ein, um den Außenkugelradius  $R$  zu ermitteln.

$$R^2 = \left(\frac{3 + \sqrt{5}}{4}\right)^2 a^2 + \frac{a^2}{4} = \frac{9 + 3\sqrt{5}}{8} a^2 \quad (5.9)$$

$$R = \sqrt{\frac{9 + 3\sqrt{5}}{8}} \quad (5.10)$$

Zusammen mit (5.3) ergibt sich

$$\begin{aligned} r^2 &= R^2 - b^2 = \frac{9 + 3\sqrt{5}}{8} a^2 - \frac{4}{10 - 2\sqrt{5}} a^2 \\ &= \left(\frac{9 + 3\sqrt{5}}{8} - \frac{2}{5 - \sqrt{5}}\right) a^2, \end{aligned} \quad (5.11)$$

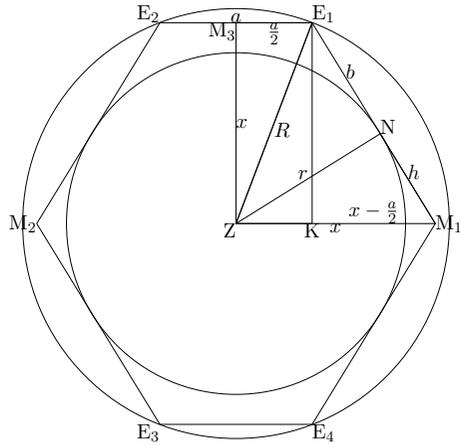


Abbildung 5.3: Schnitt durch den Dodekaeder

was wir nach  $r$  umstellen können.

$$r = \sqrt{\frac{9 + 3\sqrt{5}}{8} - \frac{2}{5 - \sqrt{5}}a} \quad (5.12)$$

## 6 Ikosaeder

Die Berechnungen im Ikosaeder sind analog zu denen im Dodekaeder. Im Schnittbild unterscheiden sich lediglich die Längen der eingezeichneten Kanten. Allerdings vereinfacht sich die Berechnung dahingehend, dass der Berührungspunkt der Innenkugel (N) mit einem der gleichseitigen Dreiecke wieder in dessen Schwerpunkt bei  $\frac{2}{3}$  der Höhe liegt.

Es ergeben sich ähnliche Gleichungen wie beim Dodekaeder:

$$\triangle ZNE_1 : \quad R^2 = r^2 + \left(\frac{2}{3}h\right)^2 \quad (6.1)$$

$$\triangle ZM_3E_1 : \quad R^2 = x^2 + \left(\frac{a}{2}\right)^2 \quad (6.2)$$

$$\triangle M_1KE_1 : \quad h^2 = x^2 + \left(x - \frac{a}{2}\right)^2 \quad (6.3)$$

Aus Gleichung (6.3) lässt sich mit  $h = \frac{\sqrt{3}}{2}a$  wieder  $x$  berechnen.

$$x = \frac{a}{4} + \sqrt{\frac{a^2}{16} - \frac{a^2}{8} + \frac{h^2}{2}} = \frac{a}{4} + \sqrt{\frac{a^2}{16} - \frac{a^2}{8} + \frac{3a^2}{8}} = \frac{a}{4} + \sqrt{\frac{5}{16}}a = \frac{1 + \sqrt{5}}{4}a \quad (6.4)$$

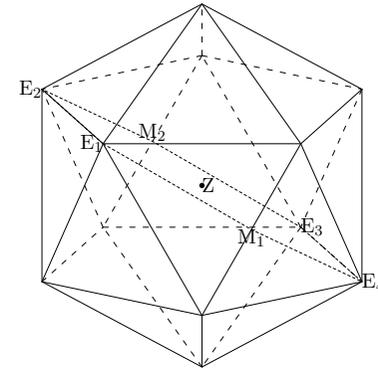


Abbildung 6.1: Ikosaeder

Um  $R$  zu berechnen setzen wir  $x$  nun in Gleichung (6.2) ein.

$$R^2 = \left(\frac{1 + \sqrt{5}}{4}a\right)^2 + \left(\frac{a}{2}\right)^2 = \frac{1 + 2\sqrt{5} + 5}{16}a^2 + \frac{1}{4}a^2 = \frac{10 + 2\sqrt{5}}{16}a^2 \quad (6.5)$$

$$R = \sqrt{\frac{10 + 2\sqrt{5}}{16}}a \quad (6.6)$$

Gleichung (6.1) liefert nun Zusammenhang mit  $r$ .

$$r^2 = R^2 - \left(\frac{2}{3}h\right)^2 = \frac{10 + 2\sqrt{5}}{16}a^2 - \frac{1}{3}a^2 = \frac{30 + 6\sqrt{5} - 16}{48}a^2 = \frac{7 + 3\sqrt{5}}{24}a^2 \quad (6.7)$$

$$r = \sqrt{\frac{30 + 6\sqrt{5} - 16}{48}}a = \frac{7 + 3\sqrt{5}}{24}a \quad (6.8)$$

## 7 Ausblick

Außer dem Zusammenhang von Kantenlänge, Innenkugelradius und Außenkugelradius besitzen platonische Körper noch eine Vielzahl weiterer interessanter Eigenschaften.

Beispielsweise sind bestimmte platonische Körper dual zueinander. Ein Körper lässt sich so in den Anderen einbetten, dass die Ecken des inneren Körpers die Seitenmittelpunkte des äußeren Körpers berühren: Tetraeder  $\leftrightarrow$  Tetraeder, Hexaeder  $\leftrightarrow$  Oktaeder, Dodekaeder  $\leftrightarrow$  Ikosaeder. Weitere Einbeschreibungen platonischer Körper ineinander sind möglich. In [2] findet sich im Unterabschnitt Geometrie/Eigenschaften ein Applet, welches diese Beziehungen anschaulich macht.

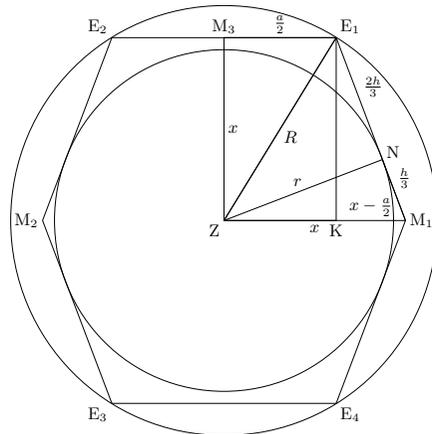


Abbildung 6.2: Schnitt durch den Ikosaeder

Aus den 5 platonischen Körpern lassen sich durch abstopfen der Ecken die 13 archimedischen Körper konstruieren. Dabei werden die Ecken so abgeschnitten, dass alle entstehenden Kanten wieder gleich Länge haben. Dieses Verfahren kann mehrfach wiederholt werden. So entsteht z.B. aus einem Ikosaeder bzw. Dodekaeder ein Ikosidodekaeder, der wiederum zum einem Ikosidodekaederstumpf abgewandelt werden kann.

Die platonischen Körper stellen nur einen sehr kleinen Teil allgemeiner Polyeder dar. Eine umfassende Beschreibung weiterer Polyeder und deren Eigenschaften findet sich in [3] und [4].

## Literatur

- [1] Hebisch, Udo, „Die Platonischen Körper“, <http://www.mathe.tu-freiberg.de/~hebisch/cafe/platonische.html>
- [2] Rockstroh, Michael, „Platonische Körper“, <http://btmdx1.mat.uni-bayreuth.de/~rockstroh/Platon.htm>
- [3] Hart, George W., „Virtual Polyhedra“, <http://www.georgehart.com/virtual-polyhedra/vp.html>
- [4] Maier, Peter H., „Körper und Raum - Die dritte Dimension: ein Lehrbuch der Polyedergeometrie und eine Klassifikation konvexer Polyeder“, Franzbecker-Verlag, 2003

# Programmdokumentation Filtrator

Sebastian Wendt

21. Juli 2009

Diese Dokumentation zum HZFW-Erweiterungspaket „Filtrator“ umfasst die Beschreibung sämtlicher Klassen und deren öffentliche Schnittstelle. Da die Programmiersprache Lingo keinen Zugriffsschutz unterstützt, sind die übrigen Methoden der jeweiligen Klassen im Quelltext als privat annotiert. Da deren Funktion Bestandteil der hier erläuterten Schnittstelle ist, wird der geneigte Entwickler auf die Funktionsbeschreibung per Kommentar im Quelltext der jeweiligen Klasse verwiesen.

Im Anschluss an die Beschreibung befindet sich eine Postervorlage des Kontrollfluss- und Klassendiagramms der Filtrator-Erweiterung, das die inneren Abläufe in schematischer Form darstellt, und eine halb-formale Darstellung zwischen Freitext und Quelltext bietet.

## 1 Klassenindex

Die Filtrator-Erweiterung zum HZFW besteht aus den folgenden Klassen:

- FiltrationsManipulator** steuert die Anzeige der Steuerelemente inklusive Schaltflächen zur Komponentenauswahl
- FiltrationsManipulatorLayoutState** verwaltet ein einzelnes Set von Schaltflächen zur Komponentenauswahl
- FiltrationsManipulatorLayoutStateFactory** erzeugt und initialisiert einen FiltrationsManipulatorLayoutState mit Laufzeitkonstanten (Zeiger auf Visor, Display)
- FiltratorStateLogger** ist Hilfs-Klasse zu FiltrationsManipulator und verwaltet einen Stack von deferierten Funktionsaufrufen. Wird eingesetzt zum Speichern und Wiederherstellen der Filtrationskonfiguration (welche ADS angezeigt werden), während der Filtrator deaktiviert ist.
- FiltratorGuide** XML-Objekt, das als Kind von Distributor mehrere Gruppen von Andockstellen in FiltratorOptions zusammenfassen kann
- FiltratorOption** XML-Objekt, das als Kind von FiltratorGuide eine Gruppe von Andockstellen zusammenfasst
- Tooltip** Utility-Klasse, die Mouseover-Tooltips erzeugt

## 2 FiltrationsManipulator

Die Koordination der Funktionalität der Filtrator-Erweiterung ist in dieser Klasse implementiert. Diese Funktionalität lässt sich wie folgt kompakt zusammenfassen:

Ist der Filtrator aktiviert wird bei Andocken an eine Zelle (genauer: an die Personalzellanockstelle der Körpermanlage) die Anzeige der Körpermanlage deaktiviert und

dafür die standardmäßig versteckten Kinderzellen (genauer: die Körpermanlagen der Zellen an den Andockstellen des Distributors) angezeigt. Dies ermöglicht die Filtration über Umschalten der Anzeige der Distributor-Andockstellen (ADS).

Ist der Filtrator aktiviert, die Zelle an der die Personalzelle andockt jedoch keine Kinderzellen enthält, so wird die Körpermanlage nicht verborgen.

Dieses Verhalten basiert auf der Annahme, dass sich die Kinderzellen in bestimmtem Umfang durch die Körpermanlagengeometrie repräsentiert werden. Insbesondere bei „Hacks“ zur Umgehung des Standard-Andockverhaltens, z.B. durch Einsatz von Dualer Geometrie kann sich das Benutzererlebnis verändern.

Geometrielemente innerhalb Dualer Geometrie werden vom Filtrator nicht verborgen. Es wird hier davon ausgegangen, dass der Ersteller der Zelle mit dem Einsatz der Dualen Geometrie eine andere Absicht verfolgte als die Darstellung einer Vorschau des Zelleninhalts, der ursprünglichen Funktion von Körpermanlagen.

Wird der Filtrator deaktiviert, so werden alle gefilterten Elemente wieder sichtbar gemacht. Ein erneutes Aktivieren des Filtrators stellt die vorherige Filterung wieder her.

## Methodenbeschreibung

- new( aoParent )** Erzeugt eine neue Instanz der Klasse, die von der Klasse Anlage erbt, und dementsprechend auf ihr Elternobjekt verweisen muss.
- show()** Macht den Filtrator sichtbar und stößt die Aktualisierung seiner selbst und der Szene per update an.
- hide()** Verbirgt den Filtrator und aktualisiert die Szene entsprechend.
- update()** Aktualisiert den angezeigten Schaltflächen und die Darstellung der Szene entsprechend der obigen Beschreibung und der zuletzt eingestellten Filterkonfiguration (d.h. welche ADS aus- bzw. eingeblendet sind).
- reset()** Löscht die gespeicherte Filterkonfiguration und aktualisiert die Szene.

Die übrigen Funktionen sind Teil der privaten Klassenschnittstelle, und erfüllen Teilaufgaben wie das Auslesen der Konfiguration von Datei. Ihre Verwendung ist anhand kommentierten Quelltextes dokumentiert.

Die ausführliche Beschreibung dieser und aller andere Funktionen der Filtrator-Erweiterung anhand von Kontrollfluss- und Klassendiagrammen befinden sich in Anhang B.

## 3 FiltrationsManipulatorLayoutState

Instanzen diese Klasse werden üblicherweise durch eine FiltrationsManipulatorLayoutStateFactory erzeugt und in einem Wörterbuch (Property-Liste)<sup>1</sup> in der Klasse FiltrationsManipulator verwaltet. In einem LayoutState zusammengefasst sind vorkonfigurierten Anordnungen von texturierten Schaltflächen (Buttons) zur Auswahl von Andockstellen. Jede Instanz entspricht einer mit aktiviertem Filtrator besuchten Zelle. Eine Schaltflächenkonfiguration wird beim Aufruf von hide nicht vernichtet, sondern per Transformationsänderung vor dem Benutzer verborgen. Die so verborgenen Schaltflächen lassen sich wesentlich schneller wiederherstellen, als solche, die per hide-Aufruf auf die einzelnen Schaltflächen verborgen wurden, ganz zu schweigen von der erneuten Anordnung und Texturzuweisung der Schaltflächen.

Die Schaltflächen bestehen aus jeweils 2 Teilen: einem Display und einem Visor. (Die Namensgebung entstammt des ersten Designprototypen des Filtrators.) Das Display be-

<sup>1</sup>in plButtonsetDict

herbergt das VorschauBild, das als Textur aufgebracht wird. Der Visor ist das bewegliche Schaltflächenelement das den Aktivierungszustand darstellt.

Displays und Visors werden während der mehrstufigen Konstruktion des LayoutStates als (rekursive/deep) Kopien von Prototyp-Display und -Visor erstellt. Die Prototypen werden dabei nicht modifiziert.

### Methodenbeschreibung

**new( avDisplacement )** konstruiert ein leeres Button-Set mit dem angegebenen Displacement-Vektor für die Methode show und hide.

**show()** zeigt ein konstruiertes Button-Set durch Subtraktion des Displacement-Vektors von der aktuellen Transformation an.

**hide()** verbirgt ein Button-Set durch Addition des Displacement-Vektors zur Transformation seiner Komponenten.

**create\_buttons( n, mdl\_proto\_display, mdl\_proto\_visor, asOptionalUniqueSuffix )** erzeugt n Schaltflächen (Displays und Visors) durch rekursive Kopie von den angegebenen Prototypen. Da die Director-Funktion cloneDeep einen eindeutigen Namen als Parameter benötigt, kann optional eine Zeichenfolge angegeben werden, der an eine laufende Nummer suffigiert wird. Als Standard wird die Stringform der Objektinstanz: string(me) verwendet.

**layout\_buttons( sLayoutPattern )** ordnet die frisch erzeugten Schaltflächen nach einem der folgenden Muster an:

- circular
- clockwise
- rectangular
- squared, nibbles

Während der aktuelle Platzierungsalgorithmus „nibbles“ eine Reihe von erwünschten Designkriterien (siehe Projektbuch) aufweist, entstammen die übrigen teilweise früheren Prototypen und sind nicht hinreichend getestet. Durch Ändern des Attributs layout\_pattern an FiltrationsManipulator in der XML-Beschreibung der Personalzelle kann ein beliebiger anderer ausgewählt werden.

**assign\_RefPics( oParent, loFiltratorOptions )** Appliziert die in den einzelnen FiltratorOptions (vorberechnet in FiltrationsManipulator) vorhandenen Referenzbilder auf die Texturen der Displays mit Hilfe der Klasse Rahmen. Diese Klasse benötigt zur Initialisierung einen Zeiger auf eine Vaterklasse, deren Rolle üblicherweise die Instanz des FiltrationsManipulators einnimmt.

**assign\_visor\_color( OptionalColor )** setzt die emissive-Farbe der Shader der Visors auf den angegebenen Wert und löscht diffuse und spekulare Farbe. Vor dem Setzen wird die diffuse Farbe in einer Klassenvariable gespeichert und wiederhergestellt, wenn die Methode ohne Parameter aufgerufen wird.

**get\_index\_of\_button\_model( mdlCandidate )** überprüft, ob das angegebene Modell identisch mit einer der Displays oder Visors des Button-Sets ist, und liefert dessen Indexnummer zurück. Wird keine Identität festgestellt, wird nichts zurückgegeben.

## 4 FiltrationsManipulatorLayoutStateFactory

Ein FiltrationsManipulatorLayoutState benötigt während verschiedene Phasen der Initialisierung bis zum finalen Layout 4 Konstanten, die in dieser Factory-Klasse vorgehalten

werden. Diese Klasse wird einmalig im FiltrationsManipulator instanziiert und danach mit der Erzeugung neuer LayoutStates kommissioniert.

Die 4 Konstanten sind

- Vektor für Modell-Displacement
- Zeiger auf Display-Prototyp-Modell
- Zeiger auf Visor-Prototyp-Modell
- Zeiger auf Host-Objekt für Rahmen

Der Displacement-Vektor ist eine Konstante deren Wert in der finalen Version auf vector( 0, 0, 177 ) eingestellt ist. Dies ist die Verschiebung zur Betrachterkamera, die ausreicht um einen LayoutState vor dem Benutzer zu verbergen. Wie bereits erläutert, ist diese Vorgehensweise diejenige, die das schnellste Wiederherstellen von LayoutStates erlaubt. Da Director – basierend auf Beobachtungen des Laufzeitverhaltens – Occlusion Culling betreibt, sind so verborgene LayoutStates auch nicht der Performanz der Anzeige abträglich.

Die beiden Modellzeiger werden aus der XML-Beschreibung des FiltrationsManipulators bezogen und ändern sich danach nicht mehr. Um sie nicht bei jeder Konstruktion eines LayoutStates übergeben zu müssen, werden sie hier gespeichert.

Das Host-Objekt für den letzten Parameter ist üblicherweise die Instanz der Klasse FiltrationsManipulator. Der Zeiger wird benötigt, um die Klasse Rahmen zu initialisieren, an die die Abbildung von Texturen auf die Display-Modelle delegiert wird.

### Methodenbeschreibung

**new( vDisplacementVector, mdlProtoDisplay, mdlProtoVisor, oRahmenHost )** setzt die Konstanten die an die Produkt-Instanzen der Klasse FiltrationsManipulatorLayoutState weitergegeben werden.

**make\_buttonset( IFiltratorOptions )** Erzeugt einen vollständig initialisierten LayoutState aus einer Liste von FiltratorOptions, wie sie aus einer XML-Zellbeschreibung ausgelesen werden kann. Die Klasse FiltrationsManipulator verfügt über eine private Methode compile\_ADS\_map( oFiltratorContextCell ), die aus einer Zelle in der Szene diese Informationen extrahiert und, falls keine FiltratorOptions angegeben sind, auch eine Liste von Andockstellen in Instanzen der Klasse verpacken kann.

## 5 FiltratorStateLogger

Diese Klasse verwaltet einen Stack von deferierten Funktionsaufrufen mit variabler Zielinstanz, Funktionsname und Parametern. Sie wird von FiltrationsManipulator eingesetzt, um hide und show-Aufrufe an die Andockstellen einer Zelle zu protokollieren und zu speichern während der Filtrator deaktiviert ist.

### Methodenbeschreibung

**new()** erzeugt einen Stack der Maximaltiefe 2, auf dem jeweils das erste und das zuletzt geschriebene gespeichert werden. Da die Funktionen zum Wiederherstellen von Zuständen genau diese beiden Fälle adressieren, werden alle Zwischenzustände verworfen.

**log\_state( Domain, oComponent, yStateHandler, oOptionalArgs )** Speichert einen Funktionsaufruf, ohne diesen auszuführen. Ein Funktionsaufruf besteht aus

**Domain** die Zelle bzw. die PZADS an deren Zelle die Distributor-ADS verändert werden. Kann jeder beliebige Typ sein, der als Schlüssel einer Property-Liste dient.

**Component** die zu verändernde Andockstelle. Domain und Component bilden zusammen eine eindeutige Beschreibung des Kontextes, wie sie für die Zuordnung von Filterzuständen zu Personalzellposition erforderlich ist. Die Funktionsaufrufe müssen von Component verarbeitet werden.

**StateHandler** der auf Component auszuführende Funktionsaufruf.

**OptionalArgs** optionale Argumente, die dem Funktionsaufruf übergeben werden.

**replay\_state( Domain, oComponent )** stellt den *zuletzt* gespeicherten Zustand auf der angegebenen Komponente per call-Aufruf wieder her. Existiert kein Zustand, geschieht nichts.

**restore\_original\_state( Domain, oComponent )** stellt den *zuerst* gespeicherten Zustand auf der angegebenen Komponente per call-Aufruf wieder her. Dieser Zustand wird auch bei einem Reset nicht gelöscht. Existiert kein Zustand, geschieht nichts.

**reset\_all()** löscht für alle Domänen und Komponenten alle Zustände außer dem ersten (d.h. maximal 1). Gleichzeitig werden diese Originalzustände wiederhergestellt. Dies entspricht der Funktionalität des Reset-Spezialbuttons auf der Filtrator-Oberfläche. Ein optionaler boolescher Parameter erlaubt das komplette Löschen des Stacks, nach dem auch die Originalzustände nicht wiederhergestellt werden können.

## 6 FiltratorGuide

Diese Klasse die ausschließlich zum Auslesen der Daten des erweiterten XML-Dialekts. Ein FiltratorGuide ist dabei lediglich eine Gruppe von FiltrationOptionen. Die Instanziierung der Klasse sollte ausschließlich in der readXMLPropList-Methode der Klasse Distributor geschehen.

Für den Zugriff durch Programmierer werden dabei die folgenden Klassenvariablen abgelegt:

**psName** speichert den optionalen Namen des FiltratorGuides. Dieser wird nicht im Programm angezeigt und dient nur dem Debugging.

**pbRetainDefaultMapping** speichert ob das Attribut retain\_default\_mapping (oder Varianten) gesetzt wurde, und zeigt dem FiltrationsManipulator an, dass bei der Erstellung der Schaltflächen sowohl sämtliche Andockstellen als auch FiltratorOptionen zu berücksichtigen sind. Standardmäßig steht dieses Attribut auf false und es werden entweder nur FiltratorOptionen (wenn vorhanden) bzw. Andockstellen als Schaltflächen angezeigt.

**plFiltratorOptions** enthält die Liste der ausgelesenen und instanziierten FiltratorOptionen. Der Zugriff erfolgt direkt ohne get-Methoden.

## 7 FiltratorOption

Diese Klasse die ebenfalls als Container für Referenzen auf Andockstellen, die als Unter-Tags ADSReference mit dem einzigen Attribut name direkt von dieser Klasse ausgelesen werden. Ebenso liest die Klasse FiltratorOption ein optionales ReferenzBild aus, instanziiert und speichert dieses.

Der Zugriff auf die Klassenvariablen ist für Entwickler freigegeben. Die Variablenamen lauten:

**psName** speichert den optionalen Namen des FiltratorOption. Dieser wird im Programm als Tooltip angezeigt und sollte in der XML-Datei gesetzt werden.

**plsADSReferences** speichert die Liste von Strings, die als Verweise auf ADS-Namen gelten sollen, wie sie aus der XML-Datei ausgelesen wurden. Diese Variable ist nur vor dem Aufruf vom bind\_ADS relevant und sollte nicht extern verarbeitet werden.

**poReferenzBild** speichert die Instanz des optionalen ReferenzBildes bzw. keinen Wert.

**ploVerifiedADS** enthält nach Aufruf von bind\_ADS die Liste der Andockstellen, deren Namen mit denen der ADSReferences übereinstimmen.

## Methodenbeschreibung

**new( aoParent )** Konstruktor, der bei der Instanziierung in der Klasse FiltratorGuide gerufen wird. Instanzen der Klasse werden auch von FiltrationsManipulator erstellt, um ADS in vor der Übergabe an die Klasse LayoutState gemäß dem Interface der Klasse FiltratorOption zu verpacken.

**readXMLPropList( IXMLPropList )** sollte nur rekursiv durch FiltratorGuide aufgerufen werden.

**bind\_ADS( loADS, yOptionalOverride )** vergleicht die übergebene List der Objektreferenzen auf Andockstellen (die derselben Zelle entstammen sollte) mit den den in der Klassenvariable plsADSReferences, und speichert diejenigen ADS-Referenzen in ploVerifiedADS, die darin vorkommen. Wird der zweite Parameter mit dem Wert #justDoIt besetzt, so wird der erste Parameter direkt in die Liste der verifizierten ADS kopiert. Dies ist nützlich, um Andockstellen ad-hoc in FiltratorOptionen zu verpacken.

**initialize\_RefPic\_from\_IClells( loEmbeddedCells, ICBInfo )** registriert ein Callback, das auf das Laden der Vorschaubilder wartet.

**completeReferenzBild\_CB( IShowInfoList )** empfängt die Callback-Aufrufe, die über initialize\_RefPic\_from\_ICells registriert wurden und überprüft, ob das Laden der Vorschaubilder abgeschlossen ist.

## 8 Tooltip

Die Tooltip-Klasse stellt unabhängig vom Rest des Filtrators Funktionen im HTFW für die Anzeige kleiner Hinweistexte beim Überfahren mit der Maus bereit. Dazu wird mit einfachen Director-Mitteln eine Textur erzeugt und als Kamera-Overlay platziert. Der Quellcode umfasst lediglich 120 Zeilen und kann leicht erweitert werden, um z.B. verschiedene Schriftarten oder auch Bilder anzuzeigen.

## Methodenbeschreibung

**new()** Konstruktor, der ein Tooltip in Schriftgröße 24, weiß erstellt

**set\_tooltip( sTooltip )** Stellt den anzuzeigenden Text ein und aktualisiert die Anzeige. Ist der Text identisch mit dem aktuell gesetzten, wird nichts getan.

**set( yWhat, Value )** Setzt die Eigenschaften fontSize, fontScale, color und tooltip. Letztere Eigenschaft setzt den Tooltip-Text neu, achtet dabei aber nicht auf Identität mit dem bestehenden. Der Faktor fontScale dient dazu die Texturbreite, auf die der Text dargestellt wird, abzuschätzen, und die Platzierung von Zeilenumbrüchen zu steuern. Es gilt die Formel

$$\text{Texturbreite} = \text{Zeichenanzahl} * \text{fontSize} * \text{fontScale}$$

**get( yWhat )** Ruft die Eigenschaften ab, die mit set gesetzt werden können.

`update()` Aktualisiert den Tooltip nach Aufruf der Methode `set_feed_mouse_position(pointMouseLoc)` bewegt den Tooltip zum Mauscursor. Diese Methode sollte direkt an die Aktualisierung der Mausposition gebunden werden.  
`set_overlay(textureOverlay)` private Methode - setzt die Tooltip-Textur direkt und lässt sich daher leicht für die Einspeisung von Bild-Tooltips missbrauchen.

## 9 Workflow zum Konvertieren von IV-Modellen nach HZFW-XML

Zur Demonstration der Funktionen des Filtrators wurde als traditionsreiches Beispiel wurde aus dem VRTC<sup>2</sup> die Mercedes A-Klasse und sein anonymisierter Stellvertreter vom OpenInventor IV-Format nach Shockwave-W3D konvertiert. In der vorliegenden Version wurden starke Vereinfachungen getroffen, da die zwei Hauptbestandteile Außenhaut (ca. 150 Teile) und Bremssystem (ca. 250 Teile) außerhalb der erfahrungsgemäßen Belastungsgrenzen der Director-Implementierung des HZFW liegen.

Die Arbeitsschritte lassen sich wie folgt nachvollziehen:

1. Konvertieren von IV nach WRL mit IVTools ([inventor-tools.sf.net](http://inventor-tools.sf.net))
2. Import in Blender3D ([www.blender3d.org](http://www.blender3d.org))
3. Benennung der Modelle zum Gruppieren
4. Export nach FBX-Format
5. Import in Maya 8.0 (nur dieses unterstützt das SW3D-Exporter-Plugin)
6. (Notieren und Zurücksetzen der Komponententransformationen)
7. Komponentenweiser Export nach SW3D-Format
8. Erstellen der Szenenbeschreibung in XML, Wiedereinfügen der Komponententransformationen

Eine Automatisierung des letzten Punktes, z.B. durch ein Ruby-Skript kann den Arbeitsaufwand erheblich reduzieren. Ein solches Skript ist dem Verzeichnis „Komponenten“ des vorliegenden Mercedes-Modells beiliegend, das lediglich die Eingabe einer JSON-artigen Struktur mit den Modellbasistransformationen erfordert (alle Modelle sollten beim Export zentriert sein, aber ihre Transformationen in diese Struktur eingetragen werden - siehe Datei).

### 9.1 Skalierung rekursiver Modelle

Für die Skalierung von SW3D-Modellen stehen im HZFW prinzipiell 2 Freiheitsgrade zur Verfügung:

1. das Verhältnis SW3D-Modell zu Zellengröße
2. das Verhältnis Kind- zu Elternzelle

Um diese einzustellen kennt das HZFW-XML-Format aus einer Zusammenlegung mehrerer Konzeptionsstränge 7 Parameter, auf die es die Information aufzuteilen gilt.

- Attribut `groesse` der Klasse `Zelle` (FBE)
- Attribut `groessenFaktoren` der Klasse `Formanlage` (Vektor)
- Attribut `HZFWEinheiten` der Klasse `Geometrie` (FBE)
- Attribut `Sw3dEinheiten` der Klasse `Geometrie` (Skalar)

<sup>2</sup>virtuelles Teilecenter

- Attribut `massStabHZzuSW` der Klasse `PersonalZellAndockstelle` („x:y“)
- Attribut `groesse` der Klasse `Andockstelle` (FBE)
- Attribut `massStabHZzuSW` der Klasse `Andockstelle` („x:y“)

Da eine Dokumentation nicht verfügbar ist, wurden die folgenden Richtlinien per Experiment ermittelt. Besteht die Möglichkeit einen der Entwickler des HZFW zu konsultieren, sollte dieser Möglichkeit jederzeit der Vorzug gegeben werden-

- Entfernen der Attribute `massStabHZzuSW` (Werte und Bezugsgrößen hier sind am unzuverlässigsten vorherzusagen).
- Entfernen des Attributs `Sw3dEinheiten` (entspricht dem reziproken von HZFWEinheiten).
- Entfernen des Attributs `groessenFaktoren`.
- Entfernen des Attribut `groesse` an `Andockstelle`. Im Einklang mit dem Prinzip der Datenkapselung, sollte eine Zelle ihre bevorzugte Einbettungsgröße selbst verwalten. Soll von diesem Wert abgewichen werden, um z.B. ein Auto als Modellauto einzubetten, kann das Attribut wieder gesetzt werden.
- Setzen des Attributs `groesse` an Zelle auf  $6^4$ . Dieser Wert ist zu justieren, um die Einbettungsgröße der Zellen ineinander zu bestimmen.
- Setzen des Attributs `HZFWEinheiten`, so dass die tiefsten Zellen der Rekursionshierarchie die geeignete Größe haben (ggf. mit einem normierten SW3DXML-Einheitswürfel vergleichen).

Ist die Skalierungshierarchie fertiggestellt, sind noch die Geh- bzw. Seh-Constraints in den entsprechenden Anlagen auf sinnvolle Intervalle einzustellen und die Blickzentren auszurichten. Dies entspricht dem normalen Arbeitsablauf beim Erstellen einer Szene in HZFW-XML.

## A Kontrollflussdiagramm

Um die Lesbarkeit der Textzellen zu erhalten wurde das Kontrollflussdiagramm auf mehrere Seiten verteilt. Diese lassen sich in der Druckversion mit Schneidwerkzeug und Kohäsionsbindemittel wieder zu einer Gesamtansicht zusammenfügen.

