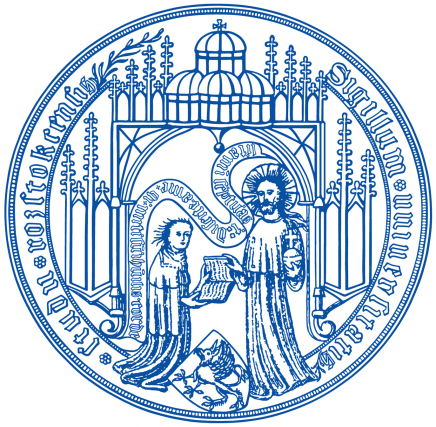
Werkzeugentwicklung für eine Community-getriebene Datenerfassung zur Modellierung von Gebäudestrukturen für die Steuerung mittels Augmented-Reality.

Masterarbeit

Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Informatik
Lehrstuhl für Informations- und Kommunikationsdienste



vorgelegt von:	Simeon Wiedenmann
Matrikelnummer:	209207232
Gutachter:	Prof. Dr. rer. nat. Clemens H. Cap
Zweitgutachter:	Prof. Dr.-Ing. habil. Peter Forbrig
Betreuer:	Dr.-Ing. Thomas Mundt
Abgabe:	Rostock, 22. August 2016

Danksagungen

Von ganzem Herzen möchte ich mich bei meinen Freunden und meiner Familie für Ihren Rückhalt und Ihre Unterstützung bedanken, welche es mir ermöglicht haben, mein Studium erfolgreich zu absolvieren.

Ich möchte mich auch für die Unterstützung bei der Bearbeitung meiner Masterarbeit durch den Lehrstuhl für Informations- und Kommunikationsdienste und insbesondere meinem Betreuer Dr. Thomas Mundt bedanken. In zahlreichen Gesprächen konnte ich wertvolles Feedback erhalten und somit die Entwicklung der Arbeit und meiner Person vorantreiben.

Inhaltsverzeichnis

Abkürzungsverzeichnis

$3D$	dreidimensional
API	Application Programming Interface
AR	Augmented-Reality
AT	Anwendungstyp
$BACnet$	Building Automation and Control Network
CSS	Cascading Style Sheets
DB	Datenbank
DOM	Document Object Model
EIB	Europäischer Installationsbus
GPL	GNU General Public License
GPS	Global Positioning System
GUI	Graphical User Interface
$HTML$	Hypertext Markup Language
IEC	International Electrotechnical Commission
IP	Internet Protokoll
ISO	International Organization for Standardization
$JSON$	JavaScript Object Notation
LON	Local Operating Network
MIT	Massachusetts Institute of Technology
OSM	OpenStreetMap
PL	Powerline
RF	Radio Frequency
SPS	Speicherprogrammierbare Steuerungen
TP	Twisted Pair
UI	User Interface
URL	Uniform Resource Locator
VGI	Volunteered Geographic Information
$VRML$	Virtual Reality Modeling Language
$WebGL$	Web Graphics Library
$X3D$	Extensible 3D
XML	Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Kurzzusammenfassung

Detaillierte Modelle von Gebäudeautomationssystemen ermöglichen neuartige, intuitive Bedienkonzepte für vernetzte Geräte in Augmented-Reality-Anwendungen. Die effiziente Erstellung hierfür benötigter, umfassender Gebäudemodelle stellt sich als schwierige Aufgabe dar. Diese Arbeit untersucht die Idee einer verteilten, Community-getrieben Datenerfassung zur Modellerstellung. Hierfür wird als Werkzeug ein Community-Editor konzipiert und prototypisch implementiert, mit welchem sich in erstellbaren Gebäudeplänen Räume und steuerbare Geräte mit zugehörigen Steuer-, Positions- und Bildinformationen erfassen lassen. Es wird herausgearbeitet, welche Eigenschaften ein solches Werkzeug vorweisen muss, um eine maximale Anzahl an Nutzern zu befähigen sich intuitiv an der Modellerzeugung zu beteiligen. Weiterhin wird untersucht, welche Probleme bei Community-getriebener Modellierung im Allgemeinen und im Konkreten auftreten können und es werden konzeptionelle Ansätze zur Lösung dieser Probleme skizziert. Ferner wird ein Konzept präsentiert, welches die generische Erzeugung von Schnittstellen zur Interaktion mit steuerbaren Geräten auf Basis der im Modell gewonnenen Informationen ermöglicht.

Abstract

Detailed models of building automation systems enable novel, intuitive user interfaces for interconnected devices in augmented reality applications. The efficient creation of comprehensive building models needed for such applications is a challenging task. This thesis examines the idea of a distributed, community-driven data acquisition to create such models. For this purpose the Community-Editor is designed and prototypically implemented. Rooms and controllable devices along with associated control-, positional- and image information can be included into building plans using this tool. It is pointed out which properties such a tool must possess in order to enable a maximum number of users to intuitively participate in the model generation process. Furthermore it is investigated which general and specific problems may arise during such community-driven modeling processes. Conceptual approaches for solving these problems are outlined. Moreover a concept is presented, which allows generic generation of user interfaces to interact with controllable devices based on the information obtained in the model.

1. Einleitung

Moderne Immobilien werden zunehmend mit Gebäudeautomationssystemen ausgestattet. Solche Technologien gestatten die Steuerung von Beleuchtungs-, Heizungs- und Klimaanlagen, Multimedia- und zahlreichen anderen Funktionalitäten über neueartige Methoden.

1.1. Vorstellung der Projektidee

Im Rahmen eines größeren Projektes wird angestrebt mittels Techniken der Augmented-Reality (AR) eine intuitive Steuerung unterschiedlichster aktiver Gebäudekomponenten von mobilen Geräten aus zu realisieren. Dazu wird dem Anwender etwa auf einem Smartphone, Tablet oder Head-Mounted Display seine aktuelle Umgebung angezeigt und steuerbare Geräte werden auf dem Bildschirm hervorgehoben. Diese können nun durch Gesten ausgewählt und intuitiv bedient werden. Die Masterarbeit „Gebäudesteuerung mittels Augmented Reality auf Smartphones“ [1] beschreibt das zugrunde liegende Konzept und die Architektur dieses Projektes.

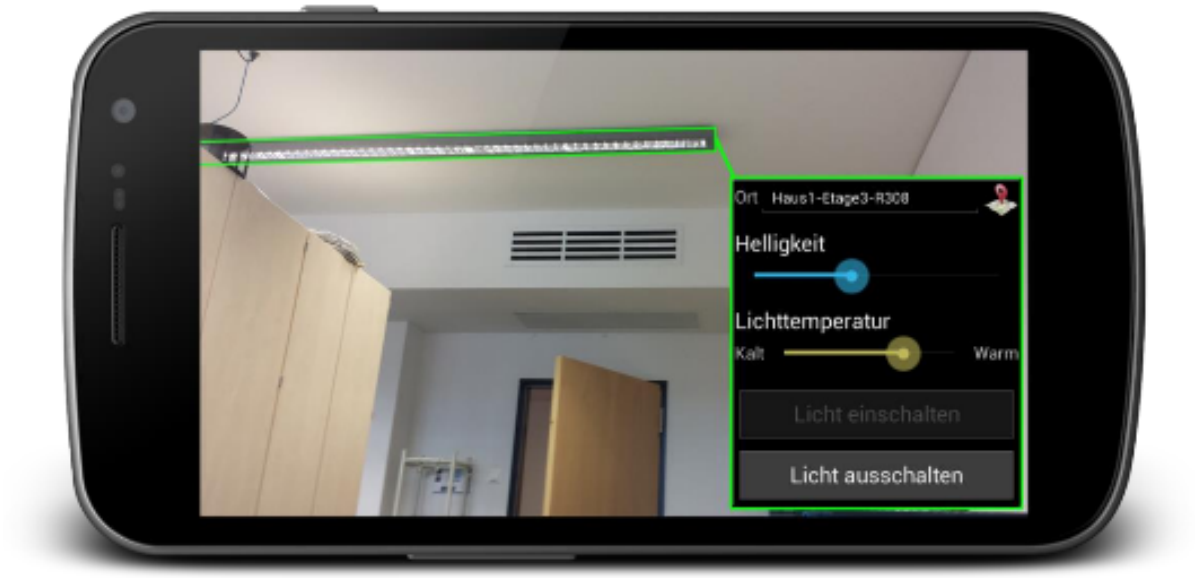


Abbildung 1.1.: Mockup der AR-Anwendung aus [2, S. 2]

Abbildung 1.1 zeigt ein Mockup für die intuitive Steuerung einer Zimmerbeleuchtung aus der Ferne mit Hilfe der AR-Anwendung. Der Anwender hält die Kamera seines mobilen Gerätes auf das zu steuernde Gerät. Unter Ausnutzung der Sensordaten des Gerätes kann seine Position und Orientierung erkannt werden. Durch den Abgleich dieser Informationen mit einem Gebäudemodell, welches Informationen über steuerbare Geräte bereit hält, kann die Computer-Vision-Erkennung der AR-Technik die Geräte im Sichtfeld der Kamera identifizieren und, z.B. durch eine farbige Umrandung wie in Abbildung 1.1, hervorheben. Nach Auswahl eines steuerbaren Gerätes kann der Nutzer mit diesem interagieren, um den gewünschten Zustand im Raum herbeizuführen.

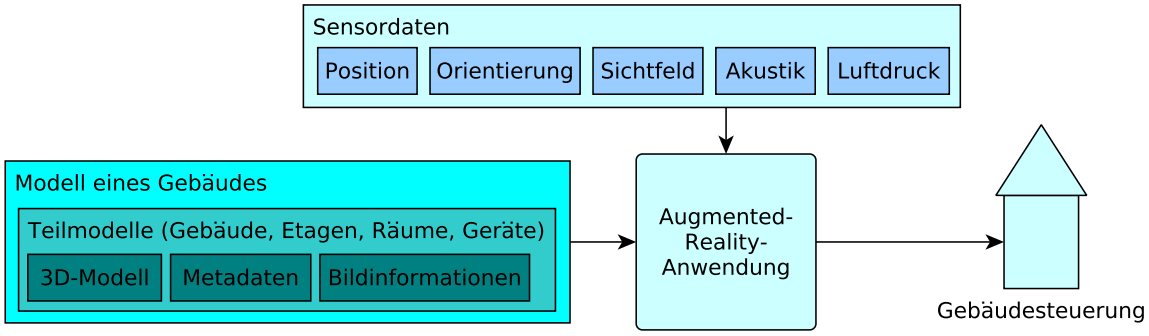


Abbildung 1.2.: Schematische Darstellung der AR-Anwendung zur Gebäudesteuerung

Eine auf [1] aufbauende Arbeit zum Thema „Gebäudesteuerung mittels Augmented Reality auf Smartphones – Feature Detection und Viewer“ [2], beschäftigt sich mit der Augmented-Reality-Komponente dieser Anwendung. Beide Arbeiten verweisen auf die Notwendigkeit eines dreidimensionalen (3D) Modells der zu steuernden Geräte und Ihrer Umgebung [2, S. 39]. So heißt es in [1, S. 32]: „Für das nichtvisuelle Tracking, die Objektidentifizierung und eventuell auch zur Unterstützung des modellbasierten Trackings wird eine 3D-Karte der Umgebung benötigt.“ Neben einem Gebäudeplan mit Räumen und den darin enthaltenen zu steuernden Geräten erfordert das System Informationen zu Steuermöglichkeiten, sowie Adressierungs-, Positions- und Bildinformationen der zu steuernden Gegenstände. Diese Informationen können in einem Modell des Gebäudes zusammengetragen werden. Eine auf dem Modell basierende Datenbank (DB) gestattet anschließend die Umsetzung der AR-Anwendung zur intuitiven Steuerung der Aktoren eines vernetzten Gebäudes.

Der Aufwand zur Erstellung von umfassenden 3D-Modellen einzelner Gebäuden und darin enthaltenen, steuerbaren Geräten, sowie die Sammlung von Positions-, Bildinformationen und Funktionsbeschreibungen der Geräte sind mit hohem Aufwand verbunden. Daher schlägt [2] vor, ein Werkzeug zur kollaborativen Erzeugung der Modelle in Form eines Community-Editors zu entwickeln [2, S. 45f].

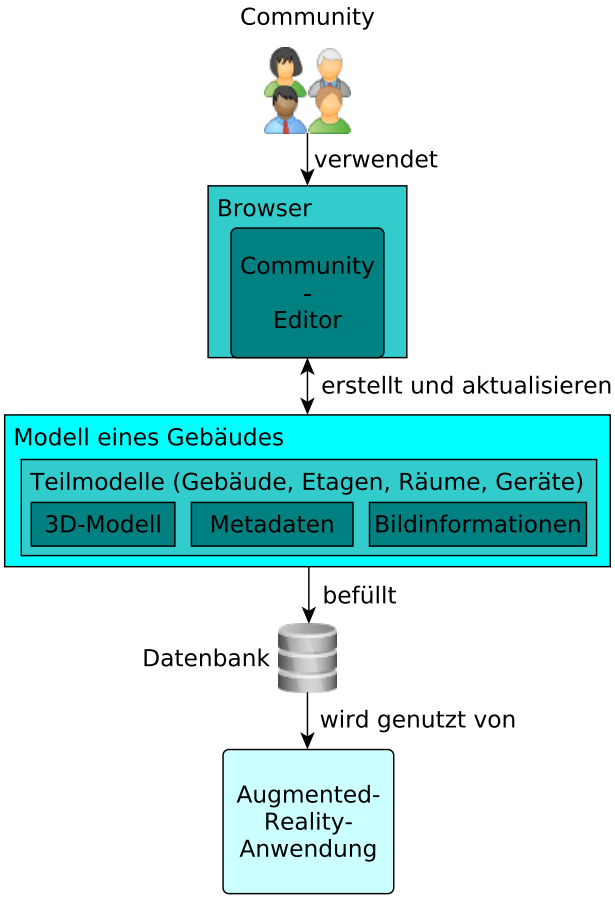


Abbildung 1.3.: Schematischer Ablauf der Erstellung und Verwendung von Gebäudemodelldaten

1.2. Zielsetzung und Einordnung der Masterarbeit in die Projektidee

Die Vorliegende Masterarbeit beschäftigt sich mit dem konzeptionellen Entwurf und der prototypischen Implementierung eines Werkzeuges zur Community-getriebenen Erstellung von Gebäudemodellen. Ein solcher Community-Editor dient der Sammlung, Verwaltung und Aktualisierung nötiger Modelldaten, welche von der beschriebenen AR-Anwendung zur Gebäudesteuerung benötigt werden, wie in Abbildung 1.3 verdeutlicht wird.

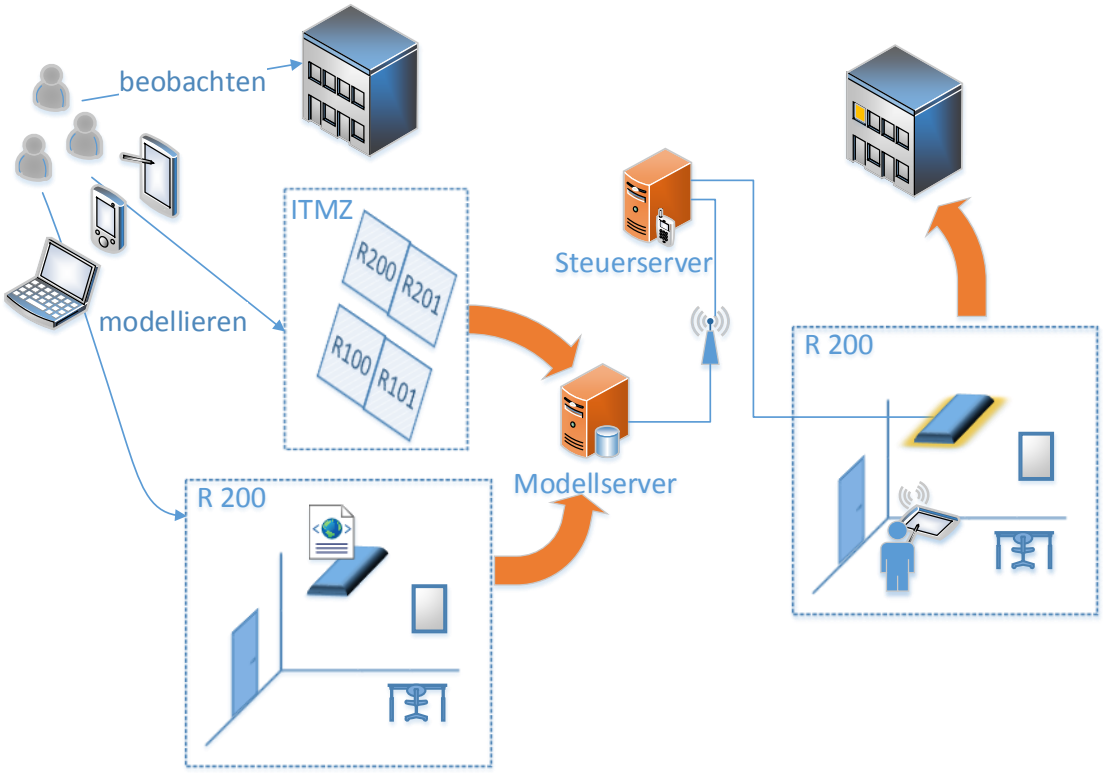


Abbildung 1.4.: Community-getriebene Modellierung eines Gebäudes und mobile Steuerung modellierter Geräte im Gebäude

In zu erstellenden Gebäudeplänen sollen hierdurch Modelle von Räumen und steuerbaren Geräten mit zugehörigen Adressen, Steuermöglichkeiten, Positions-, Lage- und Bildinformationen zusammengetragen werden können. Diese Informationen ermöglichen in anschließenden Projekten die Umsetzung eines komplexen Systems zur Steuerung von Gebäudekomponenten mittels AR durch mobile Geräte, wie es in [1] aufgezeigt wird. Abbildung 1.4 zeigt links die Modellierung eines Gebäudes durch eine Community. Rechts wird die dadurch ermöglichte mobile Steuerung aktiver Geräte, wie z.B. einer Lampe, dargestellt.

Mit Blick auf das geplante Einsatzgebiet des Editors ergeben sich wesentliche Herausforderungen, die im Folgenden kurz dargelegt werden.

Nutzer sollten ortsunabhängig und direkt an der Modellierung von Gebäuden mitwirken können. Daher empfiehlt sich die Entwicklung des Editors als webbasierte oder für mo-

bile Endgeräte spezialisierte Anwendung.

Damit der Editor flexibel eingesetzt werden kann und auch zukünftig neue Gerätearten und Steuerformate abbilden kann [1, S. 23], ist die Unabhängigkeit des Editors von konkreten Geräten und Gebäudeautomationssystemen erstrebenswert. Die Verwendung eines einheitlichen Adressierungsformates für verschiedene Busse schafft eine zusätzliche Abstraktion von den technischen Übertragungsmethoden für Steuerbefehle. Geräte unterschiedlichster Technologien und Hersteller können somit einheitlich modelliert und angesprochen werden.

Die Erstellung und Anordnung von Räumen und Geräten in einem Gebäude soll intuitiv möglich sein. Eine Darstellung zweidimensionaler Grundriss- und dreidimensionaler Raumansichten können den Nutzer hierbei unterstützen und schaffen Übersichtlichkeit.

Eine Versionierungsmöglichkeit für den Gebäudeplan ist eine wichtige Möglichkeit fehlerhafte Änderungen oder böswillige Manipulation am Modell mit minimalem Aufwand rückgängig machen zu können und wird angestrebt.

Um Editierungen am Modell einzelnen Akteuren zuordnen zu können, ein benutzerorientiertes Rechtemanagement für Modelländerungen sowie Nutzerspezifische Statistiken zur Dokumentation, Analyse und Motivation erstellen zu können, wird eine Möglichkeit zur Authentifizierung von Nutzern des Editors durch eine Login-Funktionalität benötigt.

Gebäude, Räume und Geräte besitzen unterschiedlichste Geometrien und Ausmaße. Zur realitätsgetreuen Abbildung dieser Objekte im Modell, muss die Übernahme vorhandener Bemaßungen und das Modellieren unterschiedlichster Raumgeometrien durch das Werkzeug ermöglicht werden.

Zum Beitrag der Arbeit zählen neben der Weiterentwicklung des Konzeptes eines umfassenden Gebäudemodells zur mobilen Steuerung von Geräten durch eine AR-Anwendung die Konzeptentwicklungen in den Bereichen Management einer Community-getriebenen Modellierung, Ontologiemanagement steuerbarer Geräte, sowie webbasierte Darstellung und Manipulation von 3D-Modellen.

Die prototypische Entwicklung eines Community-Editors soll abschließend die Tauglichkeit wesentlicher Kernaspekte unter Beweis stellen. Die mit diesem Werkzeug erstellbaren Modelldaten bieten in Zusammenhang mit der vorgestellten AR-Anwendung ein neues Abstraktionsniveau für die Bedienung interaktiver Gebäude aus Sicht der Anwender.

1.3. Wissenschaftliche Herausforderungen

Die verteilte, Community-getriebene Erfassung und Modellierung komplexer, heterogener Informationen zur maschinellen Weiterverarbeitung ist eine umfangreiche und vielschichtige Aufgabe. Die Entwicklung geeigneter Werkzeuge unter Ausnutzung des aktuellen Stands der Technik kann eine Community dabei unterstützen sich in diese Prozesse produktiv einzubringen. Im Rahmen dieser Arbeit wird exemplarisch an der Konzipierung und Entwicklung eines solchen Werkzeuges für den Bereich der Gebäudemodellierung aufgezeigt, welche Herausforderungen dabei auftreten können und wie geeignete Konzepte beim Design des Werkzeuges helfen können, diese Herausforderungen zu überwinden. Als Beitrag auf diesem Gebiet beschäftigt sich die hier vorliegende Arbeit mit folgenden wissenschaftlichen Fragestellungen:

1) Welche Eigenschaften muss ein Community-Editor vorweisen, damit seine Verwendung eine möglichst geringe Hürde darstellt?

2.1) Welche Probleme treten bei einer dezentralen, Community-getriebenen Modellierung komplexer Szenen im Allgemeinen und im Konkreten auf?

2.2) Welche Lösungsmöglichkeiten sind denkbar zur Bewältigung dieser Probleme?

3.1) Welche Eigenschaften muss ein Modell haben, um daraus automatisch generierte Nutzerschnittstellen erzeugen zu können?

3.2) Wie kann eine automatisch generierte Nutzerschnittstellenerzeugung aus Modelldaten erfolgen?

1.4. Motivation

Mit fortschreitendem Einsatz von Gebäudeautomationstechnik steigt auch die Anzahl steuerbarer Geräte im unmittelbaren Umfeld der Gebäudenutzer. Die Nutzerschnittstellen zur Interaktion mit automatisierten Geräten nehmen mit wachsendem Funktionalitätsumfang und steigender Gerätezahl stark in ihrer Komplexität zu. Komplizierte Bedienungsanleitungen [2, S. 3] verhindern einer intuitive Gerätenutzung und führen im schlechtesten Fall zu Nutzerfrustration, Ablehnung des Systems oder gar zu Beschädigungen der Geräte durch Fehlbedienung. Zentrale Steuerungselemente, wie etwa Lichtschalter oder Bedienungsarmaturen für die Gebäudetechnik, können zudem häufig nur aus unmittelbarer Nähe bedient werden. Die vorgestellte AR-Anwendung bietet eine alternative, flexible Steuerung dieser von mobilen Geräten wie z.B. Smartphones oder VR-Brillen aus. Die weite Verbreitung von Personal Devices wie Smartphones, welche in der Regel in unmittelbarer Körpernähe mitgeführt werden, sorgt für einen großen potentiellen Nutzerkreis der Anwendung. Zudem sind Nutzer mit dem Umgang ihrer

persönlichen Geräte bereits gut vertraut. Durch die AR-Anwendung werden sie zu intuitiven Universalfernbedienungen und ersetzen damit eine Vielzahl gerätespezifischer Fernbedienungen. Dies stellt einen Komfortgewinn aus Sicht der Nutzer dar. So lassen

Abbildung 1.5.: Viele individuelle Fernbedienungen zur Gerätesteuerung

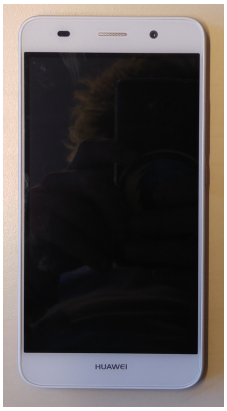


Abbildung 1.6.: Eine potentiellen Universalferbedinung

sich darüber auch schwer erreichbare Gebäudekomponenten, wie etwa Dachfenster oder an der Decke angebrachte Beamer, bequem bedienen. Durch die zunehmende Vielfalt an mobilen Geräten und deren Interaktionsmöglichkeiten, ist es zudem denkbar die Steuerung seiner Umwelt ohne Hände bzw. Arme vorzunehmen. Eine Interaktion des Nutzers mit den ihn umgebenden Geräten durch Sprachsteuerung oder gar Kopf- bzw. Blickgesten kann einen wichtigen Beitrag zur Inklusion von Menschen leisten, die z.B. auf Grund einer körperlichen Behinderung nicht in der Lage sind herkömmliche Interaktionsmethoden zu nutzen.

Für die AR-Anwendung ist ein umfassendes Gebäudemodell mit einer Vielzahl unterschiedlicher Informationen, wie etwa 3D-Modellen, Bildinformationen, Steuerungsinformationen und Metadaten einzelner Geräte nötig. Die zentrale Erfassung umfangreicher Datensätze, z.B. durch den Erwerb einer 3D-Scanndienstleistung, wie es manche Firmen anbieten [3], ist aufwendig, teuer, unflexibel und bedarf zusätzlicher Nachbereitung. Demgegenüber steht die in [2, S. 45] vorgeschlagene Idee einer webbasierten Community-getriebenen Modellierung. Dieser **dezentrale** Ansatz erlaubt eine Verteilung der Modellierungsaufgaben auf eine Vielzahl von Nutzern und **skaliert** mit wachsender Community. Modellierungen können parallel durch viele Personen gleichzeitig vorgenommen werden. Fehler bei der Modellierung fallen so schneller auf und Änderungen sind sehr dynamisch von jeder Person direkt einpflegbar. Nutzer können aktiv an der Inhaltserstellung des Systems teilhaben und so beispielsweise kreative Ideen zur intuitiven Steuerung von Geräten beitragen. Diese Art der Einbindung Freiwilliger hat sich bereits in anderen Feldern bewährt, wie z.B. der Erfolg von Community-Editoren zur Straßenkartenerzeu-

gung bei OpenStreetMap (OSM)[1] zeigt.

Eine webbasierte 3D-Darstellung des Gebäudemodells gestattet intuitives Mitwirken bei der Datengenerierung. Für das einfache Übertragen vorhandener Objekte im Raum in virtuelle 3D-Objekte ist kein Expertenwissen nötig. Dies vergrößert die Zahl potentieller mitwirkender Nutzer. Eine Verwendung des Werkzeugs aus einem Browser heraus sorgt zudem für eine geräteübergreifende, mobile Verfügbarkeit ohne zusätzliche Installation von Software und trägt ebenfalls dazu bei, den Nutzerkreis möglichst groß zu halten.

Häufig sind in großen Gebäuden unterschiedliche Gebäudeautomationsprotokolle und Geräte unterschiedlicher Hersteller vorhanden. Die Entwicklung einer technologieübergreifenden Steuerung und Adressierung aller Geräte schafft Unabhängigkeit von einzelnen Technolgien und Herstellern bei späteren Neuanschaffungen. Somit lassen sich bei größtmöglichem Funktionsumfang die Kosten gering halten und auch zukünftige Protokolle und Geräte ins System integrieren. Eine Klassifizierung von Geräten entsprechend ihrer Bedienmöglichkeiten gestattet zudem geräteübergreifende, funktionsunabhängige Bedienschnittstellen.

1.5. Aufbau der Masterarbeit

Anschließend an die Einführung folgt in Kapitel 2 ab S. 12 der aktuelle wissenschaftliche Stand der Technik für die im Rahmen dieser Arbeit relevanten Themenbereiche. Behandelt werden aktuelle Protokolle der Gebäudeautomation (2.1), Technologien webbasierter 3D-Modulation (2.2), Aspekte Community-getriebener Datengenerierung (2.3), Grundlagen aus dem Bereich Augmented-Reality (2.4), sowie relevante studentische Vorarbeiten zur Gebäudesteuerung (2.5).

Kapitel 3 ab S. 31 präsentiert ein bezüglich [2] und [1] erweitertes Modellkonzept (3.1) und beschäftigt sich mit der Konzipierung eines nutzerorientierten Modellierungswerkzeuges (3.2). Es werden Fragestellungen zum Management einer Community (3.3), sowie zur generischen Weiterverarbeitung der im Modell festgehaltenen Informationen (3.4) bearbeitet.

Die prototypische Implementierung präsentierter Konzepte wird in Kapitel 4 ab S. 61 beschrieben. Es werden Umsetzungen zur Modellkonstruktion (4.1), zur Realisierung eines nutzerorientierten Werkzeugs als Community-Editor (4.2), zum Verwalten des Community-Prozesses (4.3) und zur generischen Weiterverarbeitung der Modelldaten, insbesondere zur Nutzerschnittstellenerzeugung (4.4) dargestellt.

[1] `https://www.openstreetmap.org`

Zur Validierung der präsentierten Konzepte wird in Kapitel 5 ab S. 75 eine Testmethodik (5.1) mit Testszenarien (5.2) erläutert und ein Fragenkatalog zur Bewertung der Resultate (5.3) vorgestellt.

Eine Zusammenfassung der Arbeit und ein Ausblick auf zukünftige Herausforderungen der verteilten Erstellung generisch weiterverarbeitbarer Modelle werden in Kapitel 6 ab S. 79 geboten.

2. Grundlagen

Im Folgenden wird der aktuelle Stand der Technik der verschiedenen, für diese Arbeit relevanten Bereiche dargestellt. Es werden verschiedene Gebäudeautomationsprotokolle und deren Übertragungs- und Adressierungsmöglichkeiten beleuchtet, um sicherstellen zu können, dass der zu konzipierende Community-Editor Geräte unterschiedlicher Hersteller und Technologien modellieren kann. Außerdem wird anhand dieser Protokolle der Funktionsumfang steuerbarer Geräte ermittelt, für welchen eine Methodik zur generische Nutzerschnittstellenerzeugung entwickelt wird. Um die Darstellung sowie die Erstellung von 3D-Modellen im Community-Editor realisieren zu können werden aktuelle Technologien der webbasierten 3D-Modulation vorgestellt. Die Konzeptentwicklung eines Community-getriebenen Werkzeuges zur Gebäudemodellierung kann auf Erfahrungen anderer Community-getriebener Projekte zurückgreifen. Bekannte Probleme und Konzepte der Community-getriebene Datengenerierung werden zu diesem Zweck zusammengefasst. Eine kurze Einführung in das Themengebiet der Augmented-Reality (AR) dient der Einordnung dieser Arbeit in das übergeordnete Projekt. Hieraus lassen sich Anforderungen an das zu erstellende Gebäudemodell sowie die zu generierenden Nutzerschnittstellen für die AR-Anwendung ableiten. Abschließend werden vorangegangene Arbeiten zur Gebäudesteuerung durch eine AR-Anwendung vorgestellt, welche ebenfalls Anforderungen an den zu entwickelten Community-Editor stellen und bereits erste Konzepte der Modellstruktur vorgeben.

2.1. Gebäudeautomation

Das Gebiet der Gebäudeautomation beschäftigt sich mit der Vernetzung von Geräten in Gebäuden. Neben privaten Haushalten spielt sie auch bei Zweckgebäuden eine wichtige Rolle. Es existieren bereits für eine Vielzahl an Anwendungsgebieten steuerbare Geräte in modernen Gebäuden. So zum Beispiel im Bereich der Heizungs-, Klima-, Belüftungs- und Beleuchtungsanlagen, Jalousien, Schließanlagen, Brandmeldezentralen, Bewegungsmelder, Unterhaltungselektronik u. v. m. [4, S. 21].

Eine zentrale Verkabelung einzelner Geräte ist unzweckmäßig und unflexibel. Spätere Veränderungen und Ergänzungen der Installation sind aufwendig und teuer. Kommunikationsprotokolle mit dezentralen Busanbindungen ermöglichen eine flexible, individuell anpassbare und kostengünstig erweiterbare Vernetzungen einzelner Teilkomponenten.

Als Motivation für die Installation solcher Systeme werden verschiedenste Aspekte angeführt [5, Kap. 3], darunter

- Energieeinsparungen bzw. Betriebskosteneinsparungen,
- Flexibilität der Gebäudeausstattung,
- Komfortzuwachs, sowie
- Sicherheitszuwachs.

Durch gezieltes Energiemanagement, wie z.B. automatisches Deaktivieren von Klimaanlagen bei geöffneten Fenstern oder eine belegungsplanabhängige Beheizung von Räumen, lassen sich Energieumsatz und Betriebskosten wesentlich reduzieren. Geringere Kosten bei nachträglichen Veränderungen und Erweiterungen der Gebäudetechnik sorgen ebenfalls dafür, dass sich die zunächst anfallenden Investitionskosten für Gebäudeautomationssysteme insbesondere in großen Zweckbauten zügig amortisieren [4, S. 27f].

Durch die Standardisierung einzelner Gebäudeautomationsprotokolle lassen sich unterschiedliche Geräte herstellerübergreifend miteinander vernetzen, sofern diese den jeweiligen Standard unterstützen. Proprietäre, zueinander inkompatible Protokolle und verschiedene Standards erschweren jedoch eine vollständige Integration von Geräten aller Hersteller innerhalb einer Installation [6, S. 30].

Die Vernetzung von Geräten ermöglicht zudem eine Steuerung aus der Entfernung, wie etwa das bequeme Öffnen und Schließen schwer zugänglicher Dachfenster oder gar das Aktivieren der Wohnzimmerheizung kurz vor Ankunft in der eigenen Wohnung. Auch die szenenhafte Steuerung mehrerer Geräte gleichzeitig bietet einen Komfortzuwachs. So können auf einen Knopfdruck die Jalousien heruntergefahren, Lampen ausgeschaltet, eine Leinwand herabgelassen und ein Beamer angeschaltet werden, um beispielsweise eine Präsentation zu starten. Durch die Kombination von Sensoren und Aktoren lassen sich vollautomatische Reaktionen realisieren. Eine Deaktivierung der Raumbeleuchtung nachdem die letzte Person diesen Raum verlassen hat oder eine Steuerung der Gebäudebeleuchtung in Abhängigkeit von der Umgebungshelligkeit sind mögliche Anwendungsbeispiele [6, S. 30].

Über Anwesenheitssimulationen der Beleuchtung können mögliche Einbrecher abgeschreckt werden. Automatisierte Fenster- und Türenregler können im Brandfall die Be- und Entlüftung sicherstellen und sind ebenfalls Beispiele für den Sicherheitszuwachs in Gebäuden durch die Vernetzung von Sensoren und Aktoren [4, S. 34].

Zu den gängigsten Protokollen der Gebäudeautomation [7] zählen

- KNX - ehemals Europäischer Installationsbus (EIB),

- LON - Local Operating Network,
- BACnet - Building Automation and Control Network,
- LCN - Local Control Network,
- SPS - Speicherprogrammierbare Steuerungen,
- Z-Wave.

Exemplarisch werden die ersten drei Protokolle näher vorgestellt, da diese in den Gebäuden der Universität Rostock zum Einsatz kommen. Für nähere Informationen zu den anderen Protokollen sei auf die bereits genannte Literatur [7] verwiesen.

2.1.1. KNX

Von der International Organization for Standardization (ISO) und der International Electrotechnical Commission (IEC) als internationaler Standard ISO/IEC 14543-3 [8] akzeptiert, ist KNX ein verbreitetes Feldbussystem zum Datenaustausch zwischen Geräten unterschiedlicher Hersteller in der Haus- und Gebäudesystemtechnik. Der Standard entwickelte sich aus dem Zusammenschluss des EIB mit European Home Systems (EHS) und BatiBUS Club International (BCI) [9]. Besonders im Bereich der Lichtsteuerung und Beschattung wird KNX häufig verwendet [4, S. 36].

Sensoren als Befehlsgeber und Aktoren als Befehlsempfänger werden an eine Busleitung angeschlossen und mit Spannung versorgt. Sogenannte Datenpunkttypen (DPT) [10] spezifizieren Kommunikationsschnittstellen und gestatten herstellerübergreifende Kompatibilität der Geräte. Die standardisierte Kommunikation der Busteilnehmer erfolgt durch den Austausch von Telegrammen über den KNX-Bus.

Zur eindeutigen Identifikation der Teilnehmer dienen physikalische Adressen, deren Aufbau in Abbildung 2.1 zu sehen ist und dem Schema <Bereich>.<Linie>.<Teilnehmer> entspricht. Sie orientieren sich an der Netzwerktopologie. Die Beispieladresse 1.4.19 identifiziert den 19ten KNX-Teilnehmer der Linie vier im ersten Bereich. In maximal 16 Bereichen mit bis zu 16 Linien und maximal 256 Teilnehmern pro Linie lassen sich somit bis zu 16*16*256 = 65 536 Geräte in einer KNX-Anlage verbinden.

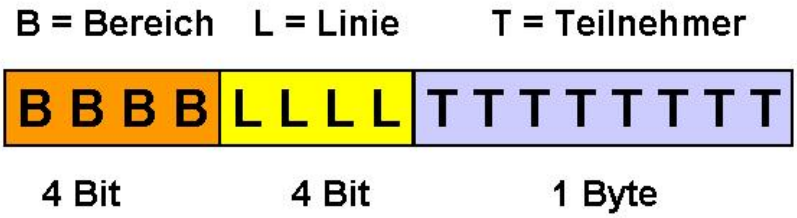


Abbildung 2.1.: KNX - Physikalische Adresse [11, S. 5]

Die logische Struktur wird durch frei programmierbare Gruppenadressen hergestellt und dient der Verbindung der Funktionen zwischen Sensoren und Aktoren. Diese lassen sich bei Bedarf, z.B. nach Umbaumaßnahmen, einfach anpassen und gestatten beispielsweise das Schalten mehrerer Geräte durch einen Steuerbefehl. In Abbildung 2.2 werden die zwei spezifizierten Formen der Gruppenadresse dargestellt. Ihr Aufbau entspricht dem Schema <Hauptgruppe>/<Untergruppe> bzw.
<Hauptgruppe>/<Mittelgruppe>/<Untergruppe>.

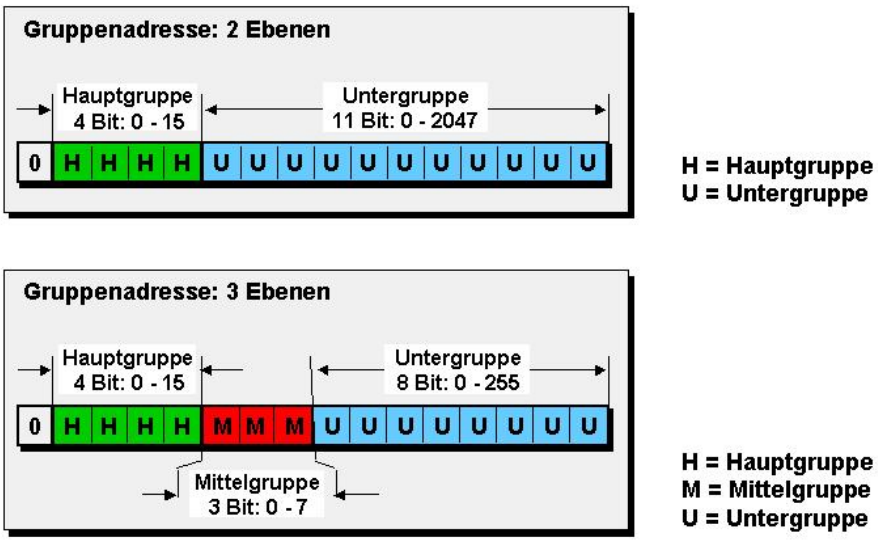


Abbildung 2.2.: KNX - Gruppenadresse [11, S. 6]

Telegramme haben als Quelladresse eine physikalische Adresse und als Zieladresse uni-cast oder multi-cast Gruppenadressen. Zur Übertragung dieser bietet der KNX-Standard folgende Übertragungsmedien an [12]:

- KNX Twisted Pair (TP)
- KNX Powerline (PL)
- KNX Radio Frequency (RF)

- KNX Ethernet/Internetprotokoll (IP)

2.1.2. LON

Das Local Operating Network (LON) wurde 1990 von der Firma Echelon entwickelt und ist nicht nur europäisch, sondern als ISO/IEC 14908 auch international standardisiert [13]. Es zählt zu den klassischen Bustechnologien im Gebäudeautomationsfeld und ist direkter Konkurrent zu KNX. Das Haupteinsatzfeld betrifft die Steuerung von Heizung, Klima und Lüftung [4, S. 37].

Ein dreikerniger Microcontroller, der Neuron-Chip ist die Kernkomponente von LON und befindet sich in jedem LON-Knoten. Er ermöglicht die dezentrale Steuerung von Geräten und dient als Kommunikationsschnittstelle zum gegenseitigen Datenaustausch der Geräte durch LonTalk-Telegramme [4, S. 161]. Sender können eine Empfangsbestätigung von den Empfängern verlangen, automatisch Neuübertragungen unbestätigter Telegramme veranlassen und Bestätigungspakete mit zusätzlichen Daten einfordern. Durch die Einführung sogenannter Netzwerkvariablen ermöglicht LON eine Interoperabilität zwischen standardisierten Geräten unterschiedlicher Hersteller [14, S. 11].

Jeder LON-Knoten ist weltweit eindeutig über eine 48 Bit lange Identifikationsnummer, die Neuron-ID identifizierbar [6, S. 38]. Zur logischen Adressierung existieren Domain-, Subnet- und Knotenadressen. Subnets sind logische Zusammenschlüsse von maximal 127 Knoten. Bis zu 255 solcher Subnets können in einer Domain zusammengefasst werden. Ein LON-Netzwerk lässt sich durch die Domain-ID identifizieren und kann entsprechend bis zu 255 * 127 = 32 385 Knoten enthalten. Mehrere Knoten können zudem in Gruppen verwaltet und über Gruppenadressierung gemeinsam angesprochen werden. Ein komplettes Adressfeld besteht aus Domain-, Ziel- und Absenderadresse. Je nach Adressierungsart enthält die Zieladresse eine sechs Byte lange Neuron-ID, eine ein Byte lange Gruppenadresse oder Subnet- und Knotenadresse, die zusammen zwei Byte umfassen. Die Absenderadresse besteht stets aus Subnet- und Knotenadresse des sendenden Knotens [14, S. 10f.].

Als grundlegende Kommunikationsobjekte zum Datenaustausch dienen Netzwerkvariablen. Funktionsblöcke definieren die Schnittstellen der verschiedenen Geräte. Diese enthalten Eingangsnetzwerkvariablen, welche Werte vom Netzwerk empfangen, sowie Ausgangsnetzwerkvariablen, die Werte ins Netzwerk senden. Über 1-zu-1-, 1-zu-n- und n-zu-1-Bindings können Eingangs- und Ausgangsnetzwerkvariablen aneinander gekoppelt werden [6, S. 39f.].

Spezielle LonWorks Transceiver ermöglichen den Anschluss der einzelnen Knoten an unterschiedliche Übertragungsmedien [14], darunter u.a.

- TP,
- PL,
- RF,
- Ethernet,
- RS-485,
- Lichtwellenleiter (LWL),
- Infrarot (IR).

2.1.3. BACnet

Ein weiteres Kommunikationsprotokoll zum Informationsaustausch zwischen Geräten und Systemen unterschiedlicher Hersteller ist das BACnet. Es wurde von der American Society of Heating, Refrigeration, and Air-Conditioning Engineers entwickelt und ist seit 2003 ISO-Norm mit der Nummer 16484-5 [4, S. 201]. Theoretisch für alle Bereiche der Gebäudeautomation geeignet, findet es häufig Anwendung als übergeordnetes System zum Management gebäudeübergreifender Leitrechner bei großen Anlagen, z.B. Krankenhäusern oder Regierungsgebäuden [4, S. 37].

Ein BACnet-Gerät wird als Menge von Objekten mit Eigenschaften abgebildet, deren Summe die Kommunikationsschnittstelle des Gerätes beschreibt. 54 Standardisierte Objekte ermöglichen herstellerübergreifende Interoperabilität zwischen Geräten [15]. Proprietäre Objekte können den Funktionsumfang eines Gerätes zu Lasten der Interoperabilität mit anderen BACnet-Geräten erweitern [4, S. 204-207]. Zugegriffen wird auf diese Objekte über Funktionen, die als Services bezeichnet werden [6, S. 33].

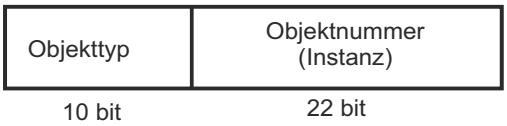


Abbildung 2.3.: BACnet - Aufbau des Object Identifiers [4, S. 249]

Um BACnet-Objekte innerhalb eines Gerätes eindeutig anzusprechen, existiert ein 32 Bit umfassender, numerischer Object Identifier, welcher sich in eine 10-Bit-Objekttyp- und eine 22-Bit-Instanznummer aufteilt. Abbildung 2.3 zeigt den Aufbau dieses Object Identifiers. Zudem besitzt jedes Gerät einen im Gesamtnetz eindeutigen Device Object Identifier. Unter Verwendung einer globalen Netzadresse und der Adresse im lokalen Netz sind Teilnehmer adressierbar. Eine zwei Byte lange Netznummer bestimmt das

Maximum von 62 535 Teilnetzen in einem BACnet.

Die Kommunikation kann über Anfragen und Antworten oder mittels eines Benachrichtigungsmechanismus erfolgen. Bei Letzterem abonniert der Client einen Wert, dessen Änderungen diesem dann automatisch mitgeteilt werden [6, S. 33].

Zur Übertragung von BACnet-Nachrichten kann auf folgende Protokolle zurückgegriffen werden [4, S. 207]:

- Master-Slave/Token-Passing (MS/TP)
- LonTalk
- ARCNET
- Ethernet

Telefonleitungen, RS-232, RS-485 und ZigBee lassen sich ebenfalls als Übertragungsmöglichkeiten verwenden [6, S. 31].

2.2. Webbasierte 3D-Modellierung

Dieses Unterkapitel beschäftigt sich mit der Beschreibung, Darstellung und Manipulation von 3D-Inhalten im Web. Mittels Beschreibungssprachen wie der Virtual Reality Modeling Language (VRML) oder Extensible 3D (X3D) lassen sich virtuelle 3D-Welten erzeugen. Diese können unter Ausnutzung von Grafikbibliotheken wie etwa die Web Graphics Library (WebGL) in Webbrowsern angezeigt werden und erlauben Interaktionen durch deren Benutzer. Auf WebGL basierende Frameworks wie X3DOM vereinfachen die Darstellung und Interaktion von 3D-Inhalten im Web und finden beispielsweise Einsatz in webbasierten Modellierungswerkzeugen.

2.2.1. Die 3D-Beschreibungssprache X3D

X3D ist ein offener Standard der ISO zur Beschreibung und zum Austausch von interaktiven 3D-Inhalten im Web [16]. Es ist der Nachfolger des webbasierten 3D-Formates der VRML [17] und ist hierzu vollständig abwärtskompatibel. VRML wurde bereits 1997 spezifiziert und als Standard eingeführt [17].

Das Web3D Consortium[1] entwickelt, pflegt und erweitert den X3D-Standard fortlaufend. So wurden bereits Vorschläge gemacht, um X3D vollständig in den Standard der Hypertext Markup Language (HTML) zu integrieren [18], [19].

[1] `http://www.web3d.org/about`

Basierend auf der Extensible Markup Language (XML) ist X3D Menschen- sowie Maschinenlesbar, jedoch existieren über die XML Notation aus dem Standard ISO/IEC 19776-1 [20] hinaus weitere Encoding-Formate. ISO/IEC 19776-2 [21] beschreibt das ClassicVRML-Encoding und ISO/IEC 19776-3 [22] ein Compressed-Binary-Encoding für das Dateiformat.

X3D-Beschreibungen können mit einem einfachen Texteditor von Hand erstellt werden oder unter Verwendung von Modellierungssoftware, wie z.B. Blender, erstellt und exportiert werden [23].

In einem Szenengraph lassen sich Objekte, Lichtquellen und Kameraperspektiven im Raum hierarisch strukturiert beschreiben [24, S. 2f.]. Somit bietet X3D ein Format zum Austausch von 3D-Informationen zwischen verschiedenen Anwendungen und über Plattformgrenzen hinweg. Beschriebene Szenen können ausgetauscht, verändert und erweitert werden, und beispielsweise als X3D-inline-Element zur Konstruktion komplexerer Szenen herangezogen werden, wie in Listing 2.2, Zeile 21 zu sehen ist.

Zur Interpretation und Darstellung von X3D-Dateien existiert unterschiedlichste Software, wie z.B. das instantreality Framework, welches vom Fraunhofer-Institut für Graphische Datenverarbeitung (Fraunhofer IGD) entwickelt wurde [25]. Darüber hinaus lässt sich X3D in Verbindung mit HTML5 und der Javascript Bibliothek X3DOM (siehe Abschnitt 2.2.2) auch interaktiv in modernen Browsern darstellen. Die durch X3D beschriebene Szene wird hierzu mittels des *<x3d>*-Tags im <body> der HTML-Seite eingefügt.

Listing 2.1 zeigt eine einfache X3D-Beschreibung eines abstrakten Hauses, welches aus einer blauen Box und einem roten Kegel zusammengesetzt wurde. Eine Szenenbeschreibung innerhalb einer HTML-Datei ist in Listing 2.2 zu sehen. Die Szene beschreibt eine grüne Kugel, welche mittels einer Translation in Zeile 12 um zwei Einheiten entlang der X-Achse und eine Einheit entlang der Y-Achse verschoben wurde. Zudem wird in Zeile 21 die X3D-Beschreibung des Hauses aus Listing 2.1 eingebunden und damit Teil der kompletten Szene. Abbildung 2.4 zeigt die Darstellung von Listing 2.2 im Browser.

```
<?xml version="1.0" encoding="UTF-8"?>
<X3D xmlns="http://www.web3d.org/specifications/x3d-namespace"
     width="400px" height="300px">

    <scene>
       <group DEF="house">

          <transform translation="0 -2 0" rotation="0 1 0 0.1745329">
            <shape>
              <appearance>
                <material diffuseColor='0 0 1'></material>
              </appearance>
              <box></box>
            </shape>
          </transform>

          <shape>
            <appearance>
              <material diffuseColor='1 0 0'></material>
            </appearance>
            <cone></cone>
          </shape>

       </group>
    </scene>
</X3D>
```

Listing 2.1: X3D-Beschreibung eines abstrakten Hauses durch die Datei house.x3d

```
<html>
  <head>
    <title>Display X3D in the browser</title>
    <script type='text/javascript'
            src='http://www.x3dom.org/download/x3dom.js'> </script>
    <link rel='stylesheet' type='text/css'
          href='http://www.x3dom.org/download/x3dom.css'></link>
  </head>
  <body>
    <x3d width='500px' height='400px'>
      <scene>
        <transform translation='2 1 0'>
          <shape>
            <appearance>
              <material diffuseColor='0 1 0'></material>
            </appearance>
            <sphere></sphere>
          </shape>
        </transform>

        <inline url="house.x3d"> </inline>
      </scene>
    </x3d>

  </body>
</html>
```

Listing 2.2: Beispiel einer X3D-Beschreibung in einer HTML-Datei x3d.html

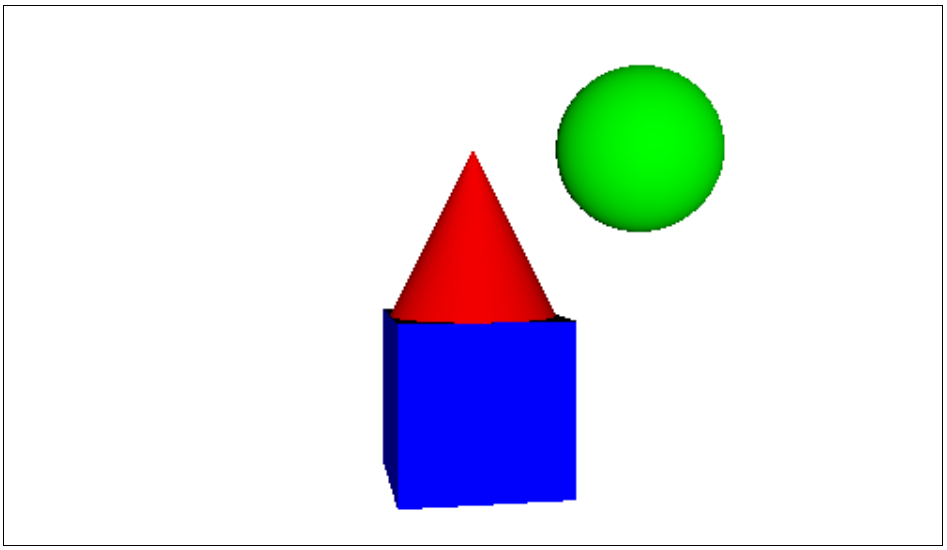


Abbildung 2.4.: Darstellung der HTML-Datei x3d.html aus Listing 2.2

Das *<x3d>*-Tag umgibt die 3D-Szenen und kann einen Raum, eine Kamera, Lichtquellen sowie die Blickrichtung der Kamera beschreiben. Die *<shape>*-Umgebung definiert dreidimensionale Objekte und kann neben einer Geometrie, wie z.B. *<box>* oder *<sphere>* auch die Erscheinung von Objekten innerhalb des *<appearance>*-Tags definieren. So lässt sich die Objektfarbe als Attribut des *<material>*-Tags beschreiben und *<imageTexture>* gestattet es Bildinformationen für eine Textur beschriebener Körper zu hinterlegen, wie in Abbildung 2.5 am Beispiel einer Deckenlampentextur zu sehen ist. Das *<transform>*-Tag ermöglicht die Translation, Rotation und Skalierung seiner Kinder-Elemente.

Neben primitiven Geometrien lassen sich auch komplexe Konstrukte durch Angabe einer Menge von Punkten, sowie deren Zugehörigkeit zu Polygonen definieren und mit entsprechenden Texturen belegen.

Eine Gruppierung von Elementen lässt sich mittels des *<group>*-Tags zusammenfassen. Durch ein für X3D-Elemente verfügbares *DEF*-Attribut lassen sich Elemente und deren Kinder eindeutig identifizieren und mittels des *USE*-Attribut entsprechend referenzieren. Das *<inline>*-Tag gestattet die Referenzierung externer X3D-Dateien. So lassen sich komplexe Szenen aus kleineren Szenen zusammen setzten.

Eine vollständige Liste aller X3D-Nodes findet sich unter [26] und eine gelungene Einführung in das Thema unter [24].

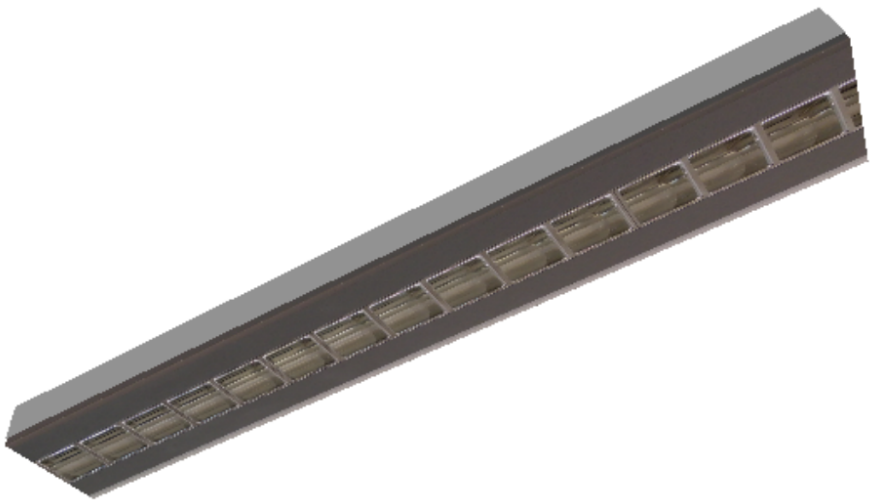


Abbildung 2.5.: X3D Modelleiner Deckenlampe mit vereinfachter Textur

Eignung von X3D zur Gebäudemodellerstellung

Die eingangs aufgeführte konzipierende Arbeit [1, S. 42f.] erwähnt bereits die Eignung von X3D für die Modellbeschreibung im Rahmen des Projektes. Durch die hierarchische Struktur von XML und die Möglichkeit Beschreibungen auszulagern ist eine flexible Erweiterbarkeit von Modellen möglich. Zudem kann eine Modellbeschreibungsdatei für jedes zu modellierende Gerät hinterlegt und diese Dateien entsprechend in komplexe Szenen eingebunden werden. Die Zerlegung komplexer Modelle in Teilmodelle wird im Konzeptkapitel 3 aufgegriffen und gestattet darüber hinaus eine einfache Katalogisierung existierender Modellkomponenten.

Darüber hinaus bietet X3D eine schnelle Möglichkeit einfache Objektstrukturen zu modellieren, jedoch sind auch komplexe, sehr detaillierte Beschreibungen möglich. Werden Objekte zudem noch mit entsprechend hochqualitativen Fototexturen belegt, können sehr präzise Modelle entstehen, welche gut von der Computervision wiedererkannt werden können.

X3D bietet viele Aspekte, welche sich zur Beschreibung von 3D-Modellen für das vorgestellte System eignen. Daher wird die Empfehlung von [1] aufgegriffen und X3D als Beschreibungssprache der Modelle des Community-Editors verwendet, ohne auf alternative Beschreibungsformate [27] wie etwa COLLADA [28] näher einzugehen.

2.2.2. WebGL

Die von der Khronos Group entwickelte WebGL ist ein lizenzgebührenfreier Web-Standard zur grafikbeschleunigten Darstellung von 3D-Inhalten in Webbrowsern [29], [30], [31]. Basierend auf dem Standard der Open Graphics Library (OpenGL) zum Rendern von 3D-Grafiken und unter Ausnutzung von JavaScript, sowie dem HTML5 Canvas-Element bietet die WebGL Anwendungsprogrammierschnittstelle (API) eine plattformübergreifende Lösung 3D-Informationen zu präsentieren [32]. Als Bestandteil moderner Browser sind keine zusätzlichen Plugins für die 3D-Darstellung erforderlich [33].

Die Entwicklung und Darstellung von 3D-Inhalten für das Web mit WebGL sind auf unterschiedlichsten Abstraktionsebenen möglich, da zahlreiche Frameworks wie etwa SpiderGL[2], XML3D[3], X3DOM[4], three.js[5] u.v.m. existieren. Eine Übersicht bekannter Frameworks findet sich unter [34]. Der in Kapitel 2.2.3 beschriebene Component-Editor basiert auf dem X3DOM Framework. Diese wird daher im Vergleich mit three.js näher vorgestellt.

[2] `http://spidergl.org/`
[3] `http://xml3d.org/`
[4] `http://www.x3dom.org/`
[5] `http://threejs.org/`

X3DOM

Das 2009 veröffentlichte Paper [35] „X3DOM – A DOM-based HTML5/ X3D Integration Model“ schlägt mit dem X3DOM-Framework (gesprochen X-Freedom) eine Architektur vor, welche es ermöglicht XML-basierte X3D-Inhalte in XHTML-Documente zu integrieren und diese durch das Standard Document Object Model (DOM) live verfügbar zu machen. Es versetzt moderne Browser in die Lage X3D-Inhalte direkt ohne Plugins anzeigen zu können [36].

Die Entwicklung des Open-Source-Frameworks [37] wird maßgeblich vom Fraunhofer IGD, dem Gründungsmitglied der Gruppe „Declarative 3D“ des World Wide Consortium (W3C), vorangetrieben und soll die Diskussionen um Web3D-Inhalte produktiv unterstützen. Auf lange Sicht ist die Aufnahme in den HTML5 Standard das Ziel. Hierfür strebt X3DOM einen menschenlesbaren 3D-Szenengraph an, der Entwicklung, Bearbeitung, Animation und Zugriff auf 3D-Inhalte mittels DOM Elementen ermöglicht [38].

Durch eine deklarative Entwicklung von Szenen [36] bietet es einen höheren Abstraktionsgrad als bei programmierten 3D-Szenen, wie es etwa direkt in WebGL nötig wäre [35], und ermöglicht zugleich den Einsatz externer Modellierungswerkzeuge, wie etwa Blender, zur Szenenerstellung.

Die Darstellung von X3D-Inhalten im Webbrowser ist mit Hilfe X3DOM einfach realisierbar. Eine Webseite, die in X3D beschriebene 3D-Szenen darstellen möchte, muss dafür lediglich zwei Dateien einbinden und sollte im xhtml Format geschrieben sein. Die etwa 950 kB umfassende JavaScript-Bibliothek x3dom.js[6] kann über das html-*<script>*-Tag und das zugehörige, knapp 7 kB große Stylesheet x3dom.css[7] über den *<link>*-Tag in die html-Datei eingebunden werden. Zahlreiche Beispiele, Tutorials und eine ausführliche Dokumentation [39] ermöglichen einen schnellen Einstieg, sowie die Umsetzung hoch komplexer 3D-Inhalte. Durch die Weiterentwicklung des Frameworks stehen neben WebGL inzwischen zusätzliche, alternative Methoden zum Rendering der Inhalte zur Verfügung [40].

Three.js

Ein weiteres, offenes WebGL-Framework zur Erstellung, Manipulation und Animation von 3D-Inhalten im Browser ist three.js. Die ursprünglich von Ricardo Cabello entwickelte JavaScript-Bibliothek ist seit 2010 in einem öffentlichen GitHup-Repository [41]

[6] `http://www.x3dom.org/download/1.7.1/x3dom.js`
[7] `http://www.x3dom.org/download/x3dom.css`

und wird aktiv weiterentwickelt.

API-Aufrufe zum Erstellen und Manipulieren von Kameras, Objekten, Szenen, Viewports, zur Verwendung von Beleuchtungsmodellen, zur Animation von Szenen sowie Vektor- und Matrixoperationen reduzieren den Aufwand gegenüber reiner WebGL-Programmierung durch ein höheres Abstraktionslevel [42]. Mit ca. 970 kB ist die three.js-Bibliothek[8] vergleichbar in Ihrer Größe zur x3dom.js-Bibliothek und lässt sich ebenfalls über das html-<script>-Tag in html-Seiten einbinden.

Wie auch X3DOM bietet three.js ein 2D-Rendering für den Fall, dass WebGL nicht unterstützt wird [42]. Zwar existieren Import- und Export-Hilfsmittel für verschiedene Datenformate, jedoch wird X3D aktuell nicht unterstützt [43].

Zahlreiche Beispiele [44], und Dokumentationen [45] sorgen dafür, dass das Framework in unterschiedlichsten Themenfeldern zum Einsatz kommt, wie etwa zum Anzeigen von Molekülmodellen [46] oder zur Visualisierung von Stromversorgungssystemen [47].

2.2.3. 3D-Modellierungswerkzeuge

xEdit

Die browserbasierte Rich Internet Application (RIA) xEdit ist ein erster Versuch ein einfaches, plattformübergreifendes Werkzeug zur Modellierung von Gebäudeszenen zu erschaffen [48]. xEdit gestattet die Erstellung eines X3D-basierten Szenengraphen durch die Manipulation der zugrunde liegenden XML-Struktur in einem Explorer. Es lassen sich so geometrisch primitive und komplexe Polygone definieren und gewisse Eigenschaften, wie z.B. die relative Lage zueinander beschreiben. Neben der Szenengraph-Modellierung können zusätzliche Informationen, wie beispielsweise eine Objekt-ID, in XML-Dateien ergänzt werden. Eine Render-Ansicht zeigt ein Modell der erstellten Szene. Ein detailliertes Verständnis für die XML-basierten Strukturen der zu modellierenden X3D-Szenen wird jedoch für die Benutzung vorausgesetzt. Dies begrenzt die Eignung des Editors auf Fachleute und technikaffine Personen. Für eine intuitive Gebäudemodellierung durch eine möglichst breite Community ohne Fachwissen ist xEdit somit ungeeignet.

Component-Editor

Der auf X3DOM (Siehe Kapitel 2.2.2), HTML und JavaScript basierende Component-Editor ermöglicht das Modellieren von 3D-Szenen im Browser und setzt dabei keine speziellen Fähigkeiten des Benutzers voraus. Zur Modellierung ist neben dem Browser keine separate Installation nötig. Der Editor steht quelloffen unter der Doppellizenz GNU General Public License (GPL) und der MIT-Lizenz des Massachusetts Institute of

[8] `http://threejs.org/build/three.js`

Technology (MIT) auf github zur Verfügung [49].

Einfache, individuell konfigurierbare 3D-Primitive stehen in einem Auswahlkatalog bereit. Diese können der Szene hinzugefügt und manipuliert werden. Zu den wichtigsten Funktionen des Editors gehören die Translation, Rotation, Skalierung, und Gruppierung einzelner Elemente. Mit Hilfe eines 2D-Editors können zudem komplexere Formen erzeugt und der Szene später dreidimensional hinzugefügt werden. Eine Gitterstruktur kann eingeblendet werden und die Anordnung von Elementen unterstützen. Flexibel einstellbare Kameraperspektiven und ein Bezugskoordinatensystem erleichtern das Erstellen von 3D-Inhalten aus unterschiedlichen Blickwinkeln. Die Modellierung kann außerdem unter Angabe unterschiedlicher Maßeinheiten, wie etwa Meter, Zentimeter oder Zoll, erfolgen. Erstellte Szenen lassen sich im Dateiformat JavaScript Object Notation(JSON) im- und exportieren. Der Component-Editor bietet damit die nötigen Voraussetzungen zur Integration als 3D-Modellierungswerkzeug in den Community-Editor.

2.3. Community-getriebene Datengenerierung

Die Anzahl der Nutzer im Internet steigt exponentiell und beträgt 2016 laut [50] bereits etwa 45% der Weltbevölkerung. Neben der Anzahl an Internetnutzern hat sich auch deren Rolle vom Konsument von Informationen hin zum Produzent von Inhalten im Web verändert [51]. Für Schlagworte wie „Social Media“ [52] und „Web 2.0“ [53], [54], [55] finden sich zahlreiche Definitionsversuche. Allen gemeinsam ist die Zunahme von Interaktionen der Nutzer durch Kollaboration, Teilen von Inhalten und sogar dem Erstellen von Inhalten (engl. User Generated Content) [56] in einem sozialen Prozess.

Diese Entwicklung des Web führte zu Community-basierten (engl. community based) [57] Anwendungen, deren wesentlicher Inhalt nicht länger zentral vorgegeben wird, sondern in Zusammenarbeit der Nutzer durch diese erstellt wird. Die Anwendungen selbst stellen Sammelorte und bisweilen Werkzeuge für die kollaborativen Prozesse dar [51]. Beispiele hierfür sind die Fotoplattform Flickr[9] oder das Videoportal YouTube[10]. Auch das weltweite, GPS-basierte Schatzsucherspiel Geocaching[11] wird erst durch das Verstecken kleiner Döschen durch Freiwillige möglich. Die Internet-Enzyklopädie Wikipedia[12] ist ein häufig genannter Vertreter für soziale Systeme zur kollaborativen Wissensbildung [58]. Jeder Nutzer des Internets kann sich an der Erstellung und Bearbeitung von textbasierten Artikeln der Wikipedia beteiligen.

Bezogen auf freiwillig erstellte geografische Informationen wird in diesem Zusammenhang

[9] `www.flickr.com`
[10] `www.youtube.com`
[11] www.geocaching.com
[12] `www.wikipedia.org`

auch von Volunteered Geographic Information (VGI) gesprochen [59]. Eine Besonderheit hierbei kann die Notwendigkeit sein, über lokales Wissen zu verfügen, um sich an der Erstellung und Bearbeitung von Kartenmaterial zu beteiligen [60]. Vertreter hierfür sind das Community-getriebene Projekte Cyclopath[13], welches sich der Erstellung von Kartenmaterial für Fahrradrouten widmet, sowie das OSM-Projekt. Es bietet eine von Nutzern erstellte Weltkarte zur freien Verfügung an. Studien hieran zeigen, dass Kollaboration nicht nur durch gemeinsames Bearbeiten identischer Objekte stattfindet, sondern bereits Kenntnis über Resultate anderer Nutzer Auswirkungen auf die Erstellung eigener Beiträge haben [61]. Um neue Nutzer zur Sammlung, Bearbeitung und Erweiterung der Kartendaten zu gewinnen, veranstaltet die Community sogar eigene „Mapping Partys "[62].

Die Daten offen lizenzierter Community-getriebenen Projekte ermöglichen eine Wiederverwertung [63] und führen oftmals zu kreativen, neuen Ideen [64]. Beispiele hierfür ist sind das Klarschiff-Portal[14] der Stadt Rostock oder die Navigations-Anwendung OsmAnd[15] für Android-Geräte. Klarschiff-hro.de ermöglicht es Bürgern Probleme wie Vandalismus oder Verschmutzung sowie Ideen zur Weiterentwicklung der Stadt auf einer OSM-basierten Karte zu melden und OsmAnd verknüpft Informationen aus Wikipedia mit OSM-Kartenmaterial zur Navigation und Darstellung von Karteninhalten.

In Community-getriebenen Strukturen treten neben positiven Aspekten, wie der zügigen, umfangreichen und vielfältigen Inhaltserstellung [65] auch Konflikte auf. Verschiedenste soziale Kontexte [66], Vandalismus, Trolle [67] und Meinungsverschiedenheiten, die zu sogenannten Edit Wars [68], [69] führen können, sind Beispiele für das Konfliktpotential solcher sozialer, kollaborativer Systeme. Mit wachsender Bedeutung und Größe steigen dabei die Koordinationskosten dieser sozialen Prozesse [58, S. 453].

Die Herausforderungen bei der Verwaltung Community-getriebener Systeme und dem Management der beteiligten Akteure sind ebenso zahlreich, wie deren Lösungsansätze und Forschungen auf diesem Gebiet. Beispielsweise setzt Wikipedia im Kampf gegen Vandalismus Versionierung und eine Rollback-Funktion ein [58] und das geocaching.com Portal nutzt einen Reviewer-Prozess zur Qualitätssicherung [70].

Im Zusammenhang mit Community-getriebenen Inhalten stellt der Begriff Wikinomics aus [71] negative Aspekte dieser Systeme in den Vordergrund und beschreibt Möglichkeiten zur wirtschaftlichen Ausbeutung aktiver Akteure durch zentrale Profiteure.

[13] `http://cyclopath.org/`
[14] www.klarschiff-hro.de
[15] `http://osmand.net/`

Abbildung 2.6.: Beispiel einer AR-Anwendung zur Projektion von Planeten in die eigene Handfläche

2.4. Augmented-Reality

Die Erweiterung der Realität (engl. Augmented-Reality) durch Ergänzung von digitalen Informationen zu realen Szenen lässt sich in zahlreichen Gebieten, wie beispielsweise Industrie, Unterhaltung, Bildung, Transport u.v.m. einsetzen [72]. Durch die zunehmende Verbreitung mobiler Geräte mit zahlreichen Sensoren und den technischen Fortschritt, werden AR-Anwendungen praxistauglich [73]. Es lassen sich Anwendungen entwickeln, die durch eine Videokamera, ein Abbild der Realität erfassen und dieses in einem Tracking-Prozess analysieren. Dabei werden die Position und Ausrichtung realer Objekte aus der Szene bestimmt sowie deren Bewegungen verfolgt. Ein anschließendes Rendering gestattet die gezielte Darstellung der manipulierten Szene, beispielsweise auf einem Display [72]. Es können realen Informationen der Szene entfernt oder zusätzliche computergestützte Informationen hinzugefügt werden [74]. Abbildung 2.6 zeigt die Android-Applikation „Planets AR“ zur AR-Darstellung von Planeten und Satelliten, die es scheinbar gestattet die Erde in der eigenen Hand zu halten.

Für die eingangs beschriebene Gebäudesteuerung sollen aktive Geräte im Blickfeld des Nutzers erkannt werden und durch das Rendering der AR-Anwendung mit Nutzerschnittstellen zur Bedienung der Komponenten überlagert werden, wie in Abbildung 1.1 prototypisch dargestellt ist. Ein wesentlicher Prozessschritt hierbei ist das Tracking, welches eine genaue Positionierung zusätzlicher Informationen ermöglicht. Abbildung 2.7 gibt einen Überblick über gängige Tracking-Methoden. Wie erkennbar ist, nutzen sowohl nicht visuelle als auch visuelle Tracking-Methoden ein Modell der Umgebung. Bereits aus stark vereinfachten 3D-Modellen können wichtige Informationen, wie Ausrichtung und Ausmaße von Objekten, gewonnen werden [75]. Detailliertere 3D-Modelle können

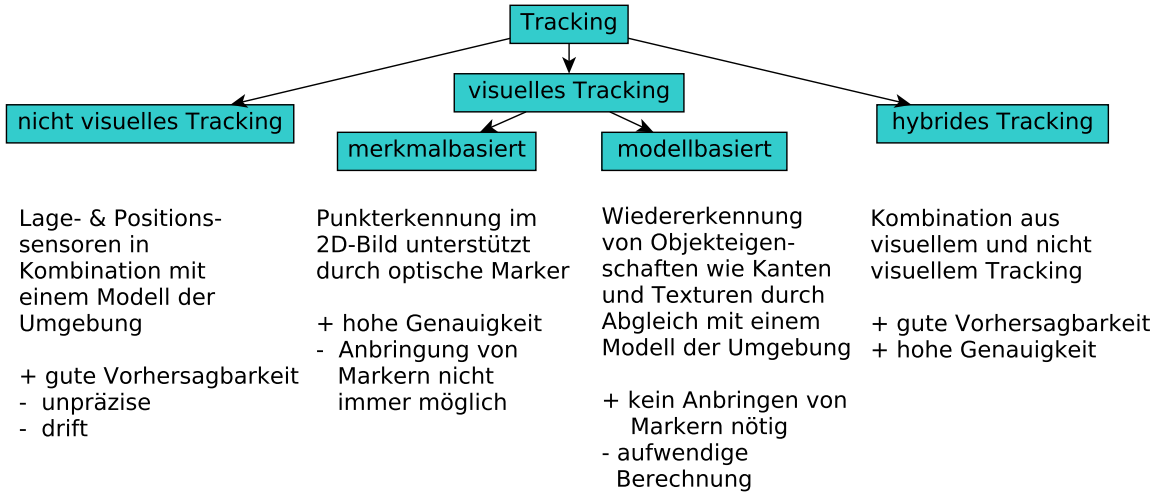


Abbildung 2.7.: Klassifizierung von Tracking-Methoden

zudem Fotografien der Objekte zum Training der Objekterkennung durch modellbasierte Trainingsdaten ersetzen [76]. Hieraus resultiert die Motivation der Werkzeugentwicklung für 3D-Gebäudemodelle.

2.5. Studentische Vorarbeiten

Wie bereits in der Einleitung erwähnt, ist diese Masterarbeit Teil eines größeren Projektes zur Konzipierung einer intuitiven Gerätesteuerung in vernetzten Gebäuden über eine AR-Anwendung. Es existieren bereits zwei Masterarbeiten und eine Projektarbeit hierzu.

Grundlegende Konzepte und eine Architektur der AR-Anwendung werden in [1] entwickelt. Es wird erstmals die Notwendigkeit eines Gebäudemodells angesprochen und mit X3D eine erste Empfehlung für eine hierfür geeignete 3D-Beschreibungssprache getätigt. Als vom Modell zu beinhaltende Informationen wird neben der 3D-Karte des Gebäudes eine Geräteliste sowie Befehls- und Funktionstypen der steuerbaren Geräte aufgeführt. Auch ein erstes Konzept zur Abbildung der Position und Ausrichtung realer Gebäude auf das Modells wird vorgestellt.

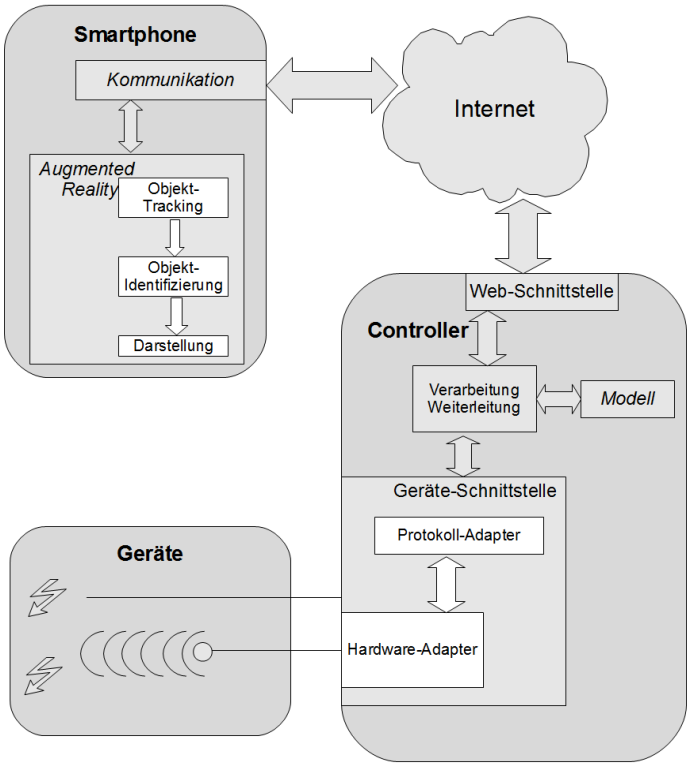


Abbildung 2.8.: Architekturkonzept der AR-Gerätesteuerung auf mobilen Geräten bestehend aus Geräten, Controller und Smartphone aus [1]

Die AR-basierte Smartphone-Komponente zur Einblendung von Interaktionsmöglichkeiten für steuerbare Geräte im Kamerabild (siehe Abbildung 1.1) wird in [2] entwickelt. Es werden unterschiedliche Ansätze zur visuellen Objekterkennung sowie der Umgang mit Trainingsdaten hierfür diskutiert und eine Evaluierung ausgewählter Erkennungsalgorithmen durchgeführt. Darüber hinaus wird der konzeptionelle Entwurf zur Architektur von Modell- und Steuerservern der AR-Anwendung und ein zugehöriges Discovery-Konzept für mobile Geräte vorgestellt. Es werden hierbei weitere Anforderungen an das Gebäudemodell gestellt, wie etwa die zusätzliche Sammlung von Traniningsdaten zur Objekterkennung. Außerdem wird die in [1] vorgeschlagene Abbildung der Position des Modells um eine dritte Dimension ergänzt.

Die Projektarbeit [48] beschäftigte sich mit der Entwicklung des in Abschnitt 2.2.3 bereits beschriebenen xEdit-Editors. Er ermöglicht eine erste Modellierung von Gebäuden, welche auf dem Erstellen von X3D-Szenengrafen basiert. Zur Modellierung wird folglich ein hohes Maß an Fachkenntnis vorausgesetzt, was für den Anwendungsfall eines Community-Editors ungeeignet ist.

3. Konzipierung eines Werkzeugs zur dezentralen Erstellung generisch weiterverarbeitbarer Gebäudemodelle

Das in der Einleitung beschriebene System zur Gebäudesteuerung basiert auf einem 3D-Modell, welches wichtige Informationen über Gebäude, Räume und Geräte bereithält. Neben einem Gebäudeplan mit Räumen und den darin enthaltenen zu steuernden Geräten erfordert das System Informationen zu Adressierungsmöglichkeiten, Klassifizierungen und Steuermöglichkeiten, sowie Position und Bildinformationen aktiver Gegenstände. Um eine Community-getriebene Generierung und fortlaufende Wartung dieser Informationen zu ermöglichen, wird im Rahmen dieser Arbeit ein webbbasierter Community-Editor konzipiert und entwickelt. Im folgenden Kapitel wird der konzeptionelle Modellaufbau erläutert. Es werden Anforderungen aufgeführt, die sich bei einer intuitiven, Community-getriebenen Modellierung generisch verarbeitbarer Modelle ergeben und Konzepte zur Lösung der dabei auftretenden Herausforderungen entwickelt.

3.1. Der Modellaufbau

Damit die Position eines Anwenders der eingangs beschriebenen AR-Anwendung in einem Gebäude mit steuerbaren Geräten in seiner unmittelbaren Umgebung in Verbindung gebracht werden kann, kommt ein dreidimensionales Modell zum Einsatz. Dieses enthält neben einem Gebäudeplan mit Etagen, Räumen und Geräten u.a. auch Positions-, Adressierungs-, Steuer- sowie Bildinformationen zu einzelnen Teilmodellen. Bereits ein aus einfachen Primitiven wie Quader und Kugeln zusammengesetztes, kantenbasiertes Modell kann zur Erkennung interessanter Objekte ausreichen [75]. Andere Trackingverfahren benötigen detailliertere Punktmodelle, welche auch Texturinformationen abbilden [77].

Um für die Endanwendung lediglich relevante Modellteile und somit geringere Datenvolumen an mobile Endgeräte zu übertragen, bietet sich eine Zerlegung komplexer Modelle in hierarchische Teilmodelle an, wie bereits von [1, S. 32] und [2, S. 54 ff.] empfohlen. Bei-

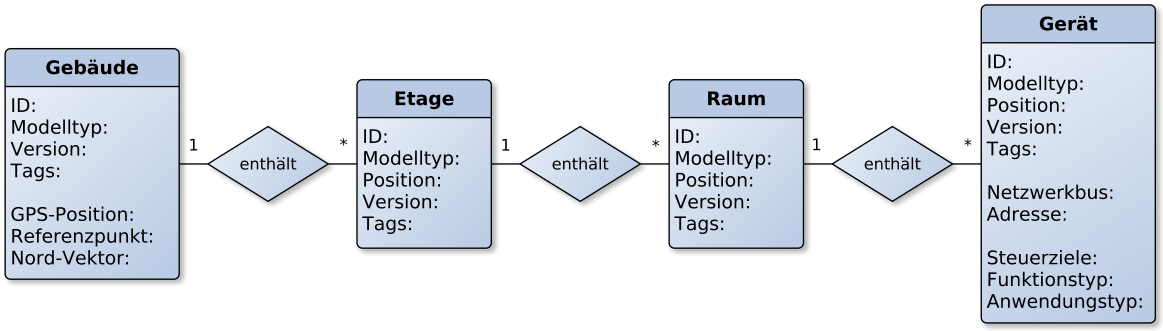


Abbildung 3.1.: Klassifizierung von Teilmodellen eines Gebäudemodells

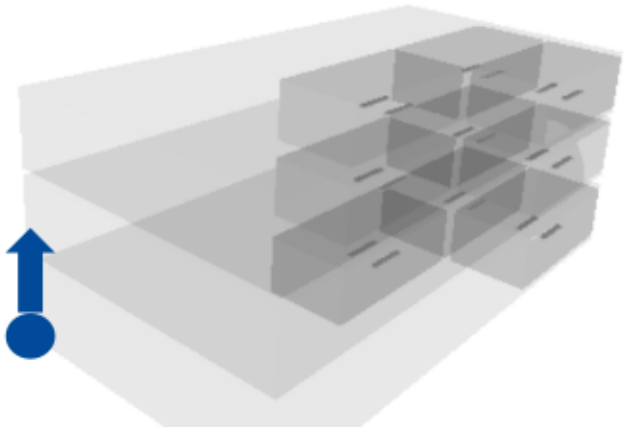


Abbildung 3.2.: Abstrakte Darstellung der modularisierten 3D-Karte eines Gebäudes mit Etagen, Räumen und Geräten samt Referenzpunkt und Nordvektor

spielsweise lassen sich Gebäude in Etagen- , Raum- und Gerätemodelle unterteilen, wie in Abbildung 3.1 erkennbar wird. Sollte es zukünftig notwendig werden weitere Aspekte im Modell mit zu berücksichtigen, lässt sich dieses durch neue Teilmodellklassen, wie etwa Treppenhäuser oder Fahrstuhlschächte, beliebig erweitern.

Eine positionsunabhängige Erstellung von Teilmodellen, sowie deren einfache Translationen und Wiederverwendung samt untergeordneter Informationen kann ermöglicht werden, indem statt absoluter Positionsangaben eine relative Positionierung der Teilmodelle erfolgt. Etagen können relativ zu Gebäudeplänen, Räume relativ zu Etagen und Geräte relativ zu Räumen positioniert werden. Damit Position und Ausrichtung des Gebäudemodells realitätsgetreu abgebildet werden können, erhält ein Gebäudeteilmodell zudem einen Referenzpunkt mit zugehöriger GPS-Koordinate des realen Gebäudepunktes, sowie einen Nordvektor [1, S. 46].

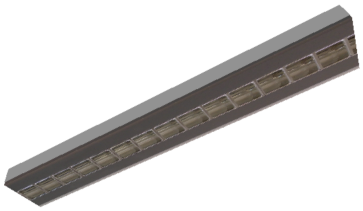
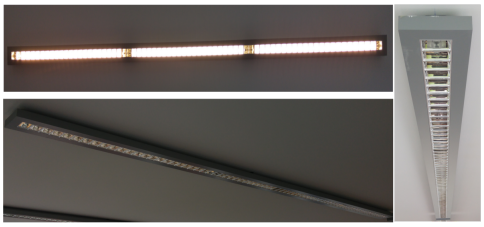
3D-Modell	Metadaten	Bildinformationen/ Foto-DB
		

Tabelle 3.1.: Komponenten eines Teilmodells am Beispiel einer Deckenlampe

Einzelne Teilmodelle bestehen wiederum aus Modellkomponenten, welche in Tabelle 3.1 exemplarisch am Gerätemodell einer Deckenlampe dargestellt werden. Zu den Modellkomponenten gehört ein 3D-Modell des Objektes, objektspezifische Metadaten und eine Foto-Datenbank. Diese enthält Bildinformationen zum spezifischen Objekt, welche es aus möglichst unterschiedlichen Perspektiven und in verschiedensten Belichtungssituationen darstellen. Sie wird für die Objekterkennung der Computergrafik benötigt [2, S. 39]. Eine Trennung konkreter Objekte von ihren 3D-Modellen gestattet die Wiederverwendung von 3D-Beschreibungen für identisch erscheinende Objekte. Beispielsweise können zwei verschiedene Lampentypen A und B über unterschiedliche Funktionalitäten verfügen, aber eine gleiche äußere Erscheinung besitzen und somit auf ein und dasselbe 3D-Modell zurückgreifen, wie in Tabelle 3.2 erkennbar wird. Die objektspezifischen Metadaten verweisen daher auf einen 3D-Modelltyp, welcher zudem über eine Versionsnummer und optionale Markierungs-Tags verfügt.

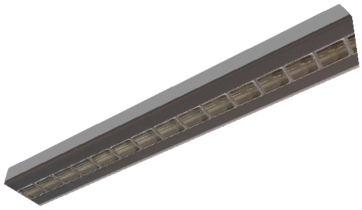
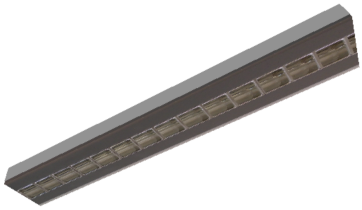
Gerät	Lampe A	Lampe B
Funktion	Ein, Aus	Ein, Aus, Dimm
3D-Modell	 Lampe.x3d	 Lampe.x3d

Tabelle 3.2.: Modelltyp zur Wiederverwendung von 3D-Modellen

Die Metadaten beinhalten außerdem eine ID zur Identifizierung des konkreten Objektes, Positionsinformationen, eine Versionsnummer und optionale Tag-Markierungen, sowie den Nutzernamen des letzten Bearbeiters und eine Liste von Nutzerkennzeichen, die temporär nicht berechtigt sind das betroffene Modell zu bearbeiten. Sofern es sich um

Schlüssel	Wert
ID	Ger_001
Teilmodellklasse	Gerät
Modelltyp	Gerät/D001/3d_Modell/Lampe.x3d
Position	<Gebäude 42><Etage 1><Raum 101><0.5m><1.0m><2.0m>
Version	1.1
Tags	To-Do
Letzte Editor	Nutzer0815
gesperrte Nutzer	-
Netwerkbus	KNX
Adresse	2.8.15
Steuerziele	self(Ger_001)
Funktionstyp	FT1(C1(Toggle), C2(Slide_Int))
Anwendungstyp	AT1(Beleuchtung)

Tabelle 3.3.: Beispiel einer Metadaten-Modellkomponente am Beispiel des Teilmodells einer Lampe

ID	Befehl	Parameter	Wertebereich	Beschreibung
C1	Toggle	Bool x	[true, false]	Kippschalter
C2	Slide_Int	Int x	[0, 100]	Mengenauswahl
C3	Slide_Float	Float x	[0.0, 100.0]	Fließauswahl

Tabelle 3.4.: Befehlstypen zur Klassifizierung der Funktionalitäten von Objekten

ein in der AR-Anwendung aktives Objekt handelt, werden Informationen zur physikalischen Adressierung des verknüpften Gerätes, eine Liste aller Steuerziele sowie ein Funktionstyp und eine Liste von Anwendungstypen (siehe Tabelle 3.5) definiert. Tabelle 3.3 zeigt beispielhaft die Metadaten eines Lampenteilmodells.

Funktionstypen gestattet eine Klassifizierung von Objekten entsprechend ihrer Steuerschnittstellen. Sie entsprechen einer eindeutigen Zusammenstellung von Befehlstypen, welche einzelne Funktionalitäten von Geräten entsprechend ihrer einstellbaren Parameter beschreiben. Tabelle 3.4 zeigt Beispiele hierfür. Der Anwendungstyp dient der intuitiven Filterung zur Wiederverwendung bereits modellierter Teilmodelle in Katalogen. Möchte ein Nutzer des Community-Editors beispielsweise in einem Raummodell ein Lampenmodell ergänzen, kann dieser im Gerätemodellkatalog bereits modellierter Geräte nach dem Anwendungstyp „Beleuchtung“ filtern und findet so mit minimalem Zeitaufwand passende Modelle.

3.2. Der Entwurf des Community-Editors als nutzerorientiertes Modellierungswerkzeug

Teilziel dieser Arbeit ist es einen Community-Editor zu konzipieren, welcher eine Gebäudemodellierung durch eine offene Community ermöglicht. Da eine hohe Anzahl an Nutzern in der gleichen Zeit umfangreichere Modelldatensätze erzeugen kann, als wenige Nutzer, ist die Benutzerfreundlichkeit des Editors von zentraler Bedeutung. Um dies zu erreichen, erfolgt eine Konzipierung des Community-Editors anhand nachfolgend erläuterter Eigenschaften:

Intuitivität: Die Verwendung des Werkzeuges setzt keine besonderen Fähigkeiten, wie etwa bestimmtes Fachwissen des Nutzers, voraus. Interaktionen, z.B. das Positionieren von Geräten in Räumen oder die Navigation durch ein Modell, sind weitestgehend natürlich möglich. Bekannte Bearbeitungskonzepte, etwa das Kopieren und Einfügen bestehender Teilaspekte, sind möglich. Eine logische Einteilung von Komponenten erfolgt anhand ihrer Anwendungsgebiete. Die Strukturen der Benutzeroberflächen, wie etwa Menüs, sind einheitlich sowie überschaubar gestaltet. Diese bieten dem Anwender Orientierungshilfe und Feedback für seine Aktionen.

Zugänglichkeit: Eine Verwendung des Werkzeuges ist an verschiedenen Standorten unabhängig von konkreten Geräten und Betriebssystemen sowie ohne eine zusätzliche Installation von Spezialsoftware möglich.

Verständlichkeit: Auf die Verwendung von Fachbegriffen wird verzichtet. Beschriftungen werden durch sprachunabhängige Icons und Designs ersetzt oder ergänzt [78]. Notwendige textuelle Erläuterungen erfolgen zur internationalen Verständlichkeit in englischer Sprache oder lassen sich in weitere gängige Sprachen übersetzen.

Modularität: Komplexe Modelle lassen sich aus wiederverwendbaren, austauschbaren, verschiebbaren Teilmodellen und Modellkomponenten zusammensetzen, welche gesondert gespeichert und übertragen werden können. Die existierenden Modelle stehen in geordneten Auswahlkatalogen zur Wiederverwendung bereit und neu erstellte Teilmodelle und Modellkomponenten werden zu den Katalogen hinzugefügt. Die Erweiterung des Modells durch Einführung neuer Teilmodelle ist möglich.

Delegierbarkeit: Ein Anwender kann einzelne Bearbeitungsschritte am Modell, welche ihm zu umfangreich oder zu komplex erscheinen, als Aufgabe markieren und delegieren, sowie Fehler und Inkonsistenzen am Modell markieren.

Intelligenz: Die Benutzeroberfläche des Werkzeugs bietet Funktionspaletten und Informationen kontextabhängig an. Nutzereingaben werden durch passende Vorschläge zur

Dateneingabe, wie z.B. spezielle Tastaturlayouts oder Kalendervorlagen für Datumsangaben, unterstützt.

Durch die Verbindung verschiedener Konzepte lassen sich die vorgestellten Eigenschaften erreichen. Die Verwendung eines 3D-Editors zum Erstellen von komplexen 3D-Modellen für Gebäude, Etagen, Räume und Geräte durch das Zusammensetzen aus vorhandenen geometrischen Grundprimitiven, ermöglicht es den Modellierungsprozess intuitiv zu gestalten. Die 3D-Modelle können in ihrer Form, Größe, Lage und Position manipuliert werden und durch Rendering auf 2D-Bildschirmen perspektivisch dargestellt werden [79]. Durch verschiedene Betrachtungsperspektiven kann der Nutzer existierende Modelle von allen Seiten betrachten. Eine Orientierungsanzeige für die 3D-Raumansicht [80, S. 223] und ein 2D-Grundriss helfen bei der Orientierung im erstellten Gebäudemodell. Der Prozess der Modellierung kann durch gut strukturierte Kataloge mit wiederverwendbaren, existierenden 3D-Modellen stark vereinfacht werden. Neue Teilmodelle, wie etwa Räume oder Geräte durch kopieren, einfügen und anpassen existierender Teilmodelle zu erstellen, spart Arbeit und kann dazu beitragen, dass das resultierende Modell einheitlich wird.

Eine logische Einteilung der Gerätemodelle entsprechend ihres Anwendungstyps (AT) ermöglicht die Strukturierung des Modellkatalogs und erleichtert die Wiederverwendung einzelner Modelle. Geräte können dabei auch mehreren AT zugeordnet werden. Tabelle 3.5 zeigt die durch eine Application-ID identifizierbaren AT für Gerätemodelle.

App-ID	Kategorie	Beispiele
AT1	Beleuchtung	Lampen, Beamer, Jalousie
AT2	Klimatisierung	Heizung, Belüftung, Jalousie
AT3	Kommunikation	Wlan-AP, Lan-Buchse
AT4	Multimedia	Lautsprecher, Beamer
AT5	Sicherheit	Bewegungsmelder, CCTV, Türzugriffs-Felder
AT6	Eingabegeräte	Schalter
ATX	Sonstiges	falls kein anderer AT passt und für spätere Ergänzungen

Tabelle 3.5.: Anwendungstypen zur Kategorisierung von Gerätemodellen

Um dem Benutzer des Community-Editors Rückmeldung zu bieten, mit welchem Teil des Modells er sich gerade beschäftigt, bietet sich ein kleiner Informationsbereich auf der Grafischen Benutzerschnittstelle (GUI) an. Bei der Bearbeitung eines bereits in einem Gebäudemodell positionierten Raumes wird hier beispielsweise dessen Raumnummer und Etage angezeigt.

Die Entwicklung des Community-Editors als webbasierte Anwendung ermöglicht eine

plattform- und geräteübergreifende Verwendung, sowie eine direkte, ortsunabhängige Mitwirkung von Nutzern bei der Modellierung. Sie können sich mit Smartphones, Tablets und anderen Anwendungsgeräten an der Modellierung beteiligen. Neben einem Browser, dessen Existenz auf gängigen Anwendungsgeräten vorausgesetzt wird, ist auch für die 3D-Darstellung keine weitere Installation nötig [35]. Wartungen an der Software können für unterschiedliche Betriebssysteme zentral stattfinden. Demgegenüber steht ein für den Anwendungsfall hinnehmbarer Performancenachteil allgemeiner Webanwendungen im Vergleich zu gerätespezifischen Entwicklungslösungen.

Die Verwendung von aussagekräftigen Icons statt textuellen Beschriftungen schafft eine weitestgehende Unabhängigkeit des Werkzeugs von konkretem Sprachwissen. Unbedingt nötige Beschriftungen erfolgen prototypisch in der weltweit verbreiteten englischen Sprache, um internationale Einsetzbarkeit zu gewährleisten. In späteren Entwicklungen bietet sich die Erweiterung des Werkzeugs auf gängige Sprachen an, sodass Nutzer das Werkzeug in ihre präferierte Sprache verwenden können.

Der Aufbau eines komplexen Modells aus einzelnen Teilmodellen, wie beispielsweise Etagen, Räume und Geräte, ermöglicht Nutzern sich auf konkrete Teilaspekte zu konzentrieren und das Modell jederzeit durch Einführung neuer Modellkomponenten zu erweitern. Eine Filterung des Modells nach relevanten Daten wird so ermöglicht und ist hinsichtlich der Übertragung geringerer Datenmengen zu mobilen Endgeräten im Anwendungsfall der AR-Anwendung vorteilhaft [2, S. 54]. Gebäudemodelle können beispielsweise aus Etagen zusammengesetzt werden, welche Räume beinhalten in denen Geräte angeordnet sind. In Auswahlkatalogen können entsprechend des Vorgehens in [81] bereits existierende Teilmodelle zur einfachen Wiederverwendung als Vorlagen angeboten werden. Hierfür werden die Teilmodelle in separaten Dateien gespeichert.

Einzelne Teilmodelle, wie z.B. das Teilmodell einer Deckenlampe, setzen sich aus Modellkomponenten zusammen. Im konkreten Anwendungsfall aus einem 3D-Modell bestimmenden Modelltyp mit zugehöriger Foto-DB, sowie Metainformationen zum konkreten Objekt, wie beispielsweise Position und Steuermöglichkeiten.

Bestimmte Bearbeitungsschritte bei der Modellierung sind aufwendiger oder benötigen Hoheitswissen, welches nicht jedem Nutzer zugänglich ist. Beispielsweise ist das Eintragen einer physikalischen Adresse zur Steuerung eines konkreten Gerätes in einem Gebäudeautomationssystem ins Modell nur Nutzern möglich, die Zugang zum Wissen der Adressen dieser Geräte haben. Solche Situationen stehen im Widerspruch zur geforderten Eigenschaft der Intuitivität. Die Erstellung eines 3D-Modells, Fotos des Gerätes und andere Metadaten, wie etwa die Betreffende Raumnummer lassen sich jedoch bereits ohne dieses Wissen zusammentragen. Die Einführung spezieller Markierungs-Tags, welche auf unvollständige Informationen konkreter Teilmodelle hinweisen, ermöglichen dennoch eine Nutzung durch eine Vielzahl von Anwendern. Durch Delegierung zu umfangreicher

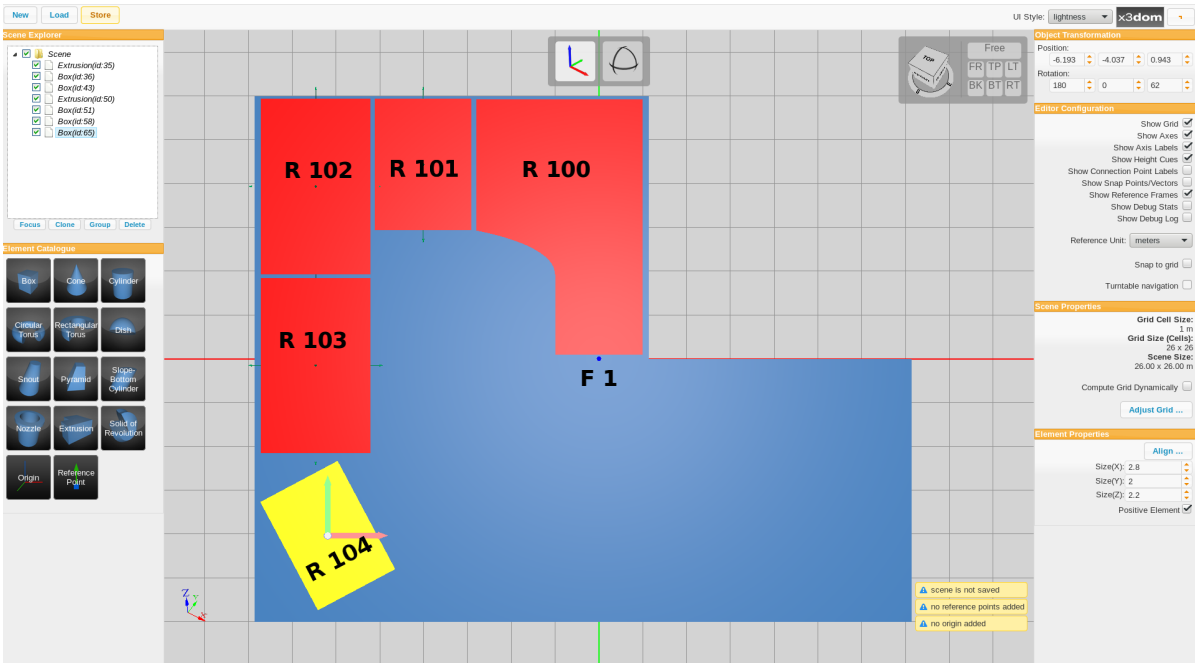


Abbildung 3.3.: Mockup der Positionierung von Räumen im Gebäudegrundriss

oder komplexer Modellierungsschritte kann jeder Nutzer seinen größtmöglichen Beitrag zur Erstellung des Gesamtsystems leisten.

Bei der Erstellung von Modellen existieren unterschiedliche Bearbeitungsschritte. Um diese möglichst effizient zu gestalten, können Werkzeugpaletten entsprechend der aktuellen Bearbeitung angeboten werden. Beim Anlegen neuer 3D-Modelle werden Modellierungswerkzeuge zum manipulieren der Objekte benötigt und geometrische Grundprimitive können in Form eines Auswahlkatalogs angeboten werden. Bei der Positionierung von Räumen im Gebäude ist stattdessen eine 2D-Grundrissansicht und ein Katalog mit existierenden Raummodellen hilfreich. Das Hinzufügen von Bildinformationen zu Modellen kann durch einen Auswahldialog hochzuladender Fotos erleichtert werden. Die Bedienoberfläche des Werkzeugs passt sich je nach aktuellem Bearbeitungsschritt an, um den Benutzer maximal bei seiner Arbeit zu unterstützen.

3.3. Das Community-Management

3.3.1. Herausforderungen einer Community-getriebenen Modellerstellung

Eine verteilte, Community-getriebene Modellierung bringt Herausforderungen mit sich, welche gelöst werden müssen. Die Erfahrung durch andere Community-getriebene Pro-

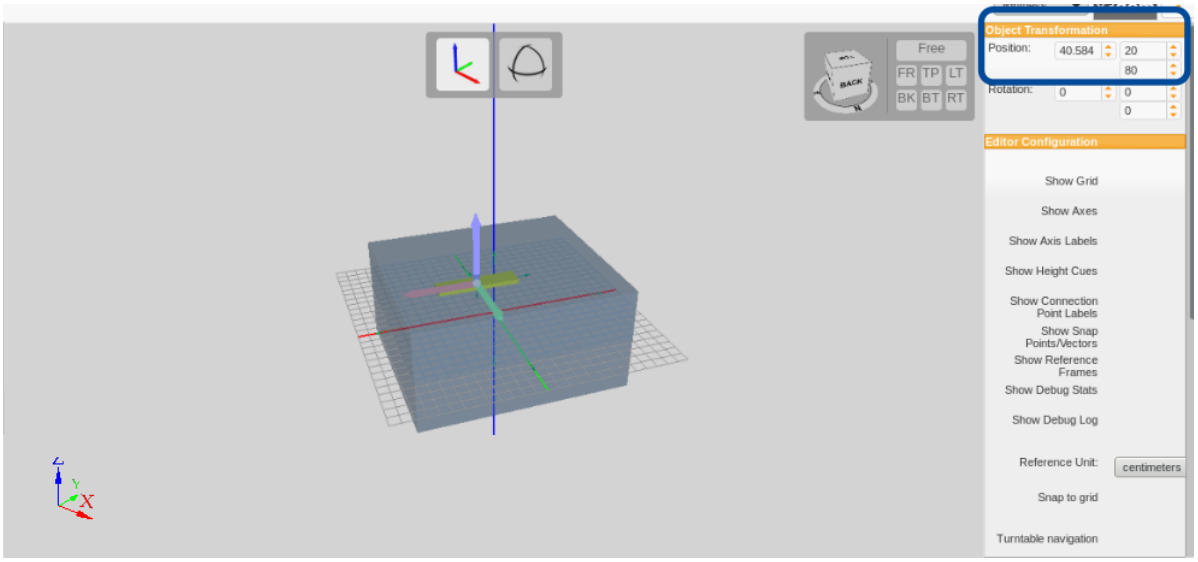


Abbildung 3.4.: Mockup der Positionierung eines Gerätes in einem Raum

zesse lässt vermuten, dass auch für das verteilte Erstellen von Gebäudemodellen die typischen Probleme auftreten werden. Dies sind u.a. sich wiederholende, gegensätzliche Änderungen am Modell, sogenannte Edit Wars [68] und Vandalismus z.B. durch das Entfernen von Inhalten oder bewusstem Hinzufügen fehlerhaften Inhalts [58, S. 456]. Studien zeigen am Beispiel der englischen Wikipedia, dass vandalisierende Änderungen aktuell nur einen geringen Anteil aller Beiträge ausmachen [58, S. 456]. Jedoch nimmt der Anteil an Beiträgen, welche sich mit Edit Wars, Vandalismus und Co auseinandersetzen stärker zu, als jener von Beiträge, welche neue Inhalte generieren [58, S. 455].

Bei kollaborativen Prozessen treffen zwangsläufig verschiedene Ansichten und Meinungen aufeinander und können zu Konflikten führen. Durch unterschiedliche Wahrnehmungen der Realität werden Modellierungen in unterschiedlicher Qualität und Genauigkeit erfolgen [82] und können zur Inkompatibilität zwischen Teilmodellen führen. Existieren Möglichkeiten der Nutzer über solche Situationen zu diskutieren, so sorgen insbesondere anonyme Beiträge oft für Provokation und verhindern eine zielführende Diskussion zur Lösung der Probleme [58, S. 458]. Folglich ist eine geeignete Koordination der Gruppenprozesse notwendig, um den Prozess einer gemeinsamen Erstellung und Bearbeitung hochqualitativer Modelldatensätze durch mehrere Personen erfolgreich zu gestalten. Ausgehend von den in Kapitel 1.3 gennanten Fragen, welche Probleme bei einer dezentralen, Community-getriebenen Modellierung komplexer Szenen im Allgemeinen und im Konkreten auftreten können und wie diese sich Bewältigen lassen, ergeben sich zahlreiche weitere Fragen. Unter anderem folgende:

- Welche Motivation treibt die Mitglieder der Community zum Modellierungsprozess an?

- Wie kann eine parallele, verteilte Bearbeitung der Modelle durch mehrere Benutzer ermöglicht werden?
- Wie kann die Kompatibilität zwischen verteilt erstellten Modellteilen sichergestellt werden?
- Welche Kommunikationsformen sind dem gemeinsamen Erstellen von Modellen förderlich, welche sind nötig und welche abträglich?
- Welche Regeln und Richtlinien können dazu beitragen die verteilte Modellierung zu koordinieren und aufkommende Konflikte produktiv zu lösen?
- Wie kann mit Edit Wars umgegangen werden?
- Wie kann gegen Vandalismus an Modelldaten vorgegangen werden?
- Wie verteilt sich die Arbeitslast in der Community auf Powernutzer, gelegentlich aktive Nutzer und einmalig aktive Nutzer?
- Sind alle nötigen Informationen zur Modellerstellung für die Community zugänglich?
- Welchen Einfluss hat Hoheitswissen[1] über abzubildende Informationen für die verteilte Modellierung?
- Lassen sich Konzepte zum Multiusermanagement von homogenen Datenbanken, wie z.B. Wikipedia, auf heterogene Datensammlungen zur komplexen 3D-Modellierung übertragen oder erweitern?
- Kann ein Community-getriebener Prozess sicherheitsrelevante Aspekte, wie etwa das Berechtigungsmanagement für eine Geräteansteuerung, hinreichend gut realisieren?

Auf Grund der begrenzten Bearbeitungszeit kann nur ein Bruchteil der aufgeworfenen Fragen behandelt werden. Die Beantwortung der Fragen nach paralleler Bearbeitung von komplexen Modellen sowie dem Umgang mit Vandalismus, Edit Wars und Hoheitswissen stehen zunächst im Vordergrund. Konzepte zur Nutzerverwaltung werden zusätzlich kurz aufgegriffen.

[1] Nicht frei zugängliches Wissen, beispielsweise die Kenntnis über technische Daten zu Geräten oder nicht öffentlich zugängliche Bereiche eines Gebäudes

3.3.2. Konzepte zum erfolgreichen Management Community-getriebener Modellerstellung

Ausgehend von der Annahme, dass die Mehrheit der Benutzer des Werkzeugs gutwillig sind und eine friedliche und zielstrebige Entwicklung eines Gebäudemodells anstreben, sollen unterschiedliche Nutzer ungestört eine parallele Erstellung und Bearbeitung hochqualitativer Teilmodelle durchführen und Schäden durch Vandalismus einfach behoben werden können [83].

Motivation

Damit sich eine möglichst große Community an der Erstellung von Gebäudemodellen beteiligt, bedarf es einer Motivation der Akteure, wie etwa in [57, S. 343ff.] aufgeführt. Analog zu bereits genannten Community-getriebenen Projekten profitieren Nutzer direkt selbst von ihren Beiträgen [84] zur Gebäudemodellierung, indem sie mitbestimmen welche Geräte von der in der Einleitung beschriebenen AR-Anwendung komfortabel bedient werden können. Wer beispielsweise die Deckenlampe in seinem Büro oder einem von ihm genutzten Raum im Modell ergänzt, ermöglicht es, diese Lampe anschließend von mobilen Geräten aus zu bedienen.

Durch nutzerbezogene Statistiken über aktive Beiträge zum Modell kann ein spielerischer Wettbewerb unter den Modellierenden erzielt werden, der sich positiv auf den Modellierungsprozess auswirken kann [85]. Nutzerprofile mit ansprechender Darstellung der eigenen Beiträge, Ranglisten aktiver Beiträge und Achievements, welche z.B. nach dem dritten, fünfzigsten, hundertsten aktiven Login freigeschaltet werden, können die Motivation aktiver Nutzer durch positive Rückkopplung für geleistete Beiträge zudem festigen.

Parallele Bearbeitung

Wollen mehrere Nutzer gleiche Teile des Modells zu einem Zeitpunkt bearbeiten, ist unklar, welche Veränderungen umgesetzt werden sollen. Verschiedene Strategien zur parallelen Bearbeitung werden in Abbildung 3.5 miteinander verglichen. Zwar existieren kollaborative Projekte im Internet, bei denen mehrere Nutzer gemeinsam einfachen Inhalt, wie beispielsweise Text[2] oder 2D-Zeichnungen[3] live bearbeiten können und Änderungen allen Nutzern live angezeigt werden, jedoch handelt es sich bei der Modellierung von maßstabsgetreuen 3D-Modellen von Räumen, Geräten und Co um eine komplexe Aufgabe. Einzelne Bearbeitungsschritte, wie etwa die Übernahme von existierenden Objektgrößen ins Modell sind zeitaufwendig und zudem von einer perspektivischen Betrachtung der

[2] https://www.piratenpad.de/
[3] https://www.twiddla.com/

Modelle abhängig. Könnten mehrere Nutzer gleichzeitig Größe, Form, Rotation und Position eines einzelnen 3D-Modells manipulieren, wäre es für den einzelnen Modellierer extrem schwer nachvollziehbar, welche parallel stattfindenden Änderungen Einfluss auf die von ihm gewünschte Manipulation am Modell haben.

Möchte ein Nutzer beispielsweise ein Gerätemodell von der Vorderseite detaillierter gestalten und rotiert daher das 3D-Modell entsprechend in eine für seine Bearbeitung hilfreiche Orientierung, könnte das die Bearbeitung eines anderen Nutzers behindern. Beginnen mehrere Nutzer die Orientierung des Modells gegeneinander zu verändern, wird die Modellbearbeitung unmöglich. Ob solches Verhalten aus ernst gemeinten Modellierungsversuchen oder durch gezielte störende Handlung von Trolls resultiert ist kaum zu unterscheiden.

Auch genaue Übernahme von Maßen wie etwa Winkeln oder Kantenlängen der realen Objekte ins Modellobjekt benötigen Zeit. Diese würde zu einer komplexen Herausforderung werden, wenn sich die Position, Dimension und Orientierung eines Live-3D-Modells ständig ändern könnte. Eine Live-Bearbeitung einzelner 3D-Modelle erscheint für den gewünschten Anwendungsfall der 3D-Modellierung nicht geeignet.

Eine weitere Mögichkeit zur realisierung paralleler Modellierung bietet die Strategie des sich durchsetztenden letzten Schreibzugriffs. Zwar lässt sich dieser Ansatz einfach realisieren und Nutzer können jeder Zeit alle gewünschte Modelle bearbeiten, jedoch überschreiben direkt nacheinander statt findende Editierungen sich gegenseitig. Dieses Verhalten ist schwierig nachzuvollziehen und bietet Potential Nutzer zu frustrieren. Bearbeiten viele Nutzer Zeitgleich identische Modelle, sind zielgerichtete, aufwendige Manipulationen kaum möglich.

Ein entgegengesetzter Ansatz des vollständigen Sperrens eines Gebäudemodells, sobald ein Nutzer eine Modellierungsaufgabe durchführt, verhindert zwar das Problem sich gegenseitig überschreibender Editierungen, jedoch kommt dieses Vorgehen für Multi-User-Anwendungen nicht in Frage. Eine parallele Bearbeitung ist nicht möglich. Das Modell kann höchstens zeitlich versetzt durch verschiedene Nutzer bearbeitet werden.

Als Mittelweg bietet sie eine Strategie des zeitlichen Sperrens minimaler Teilbereiche, der Modellkomponenten, an. Bei einem geringen Realisierungsaufwand kann dennoch ein hoher Grad paralleler Bearbeitungen am Modell erzielt werden, ohne das Risiko einzugehen, das Bearbeitungen durch Einflüsse anderer Nutzer gestört werden. Für die Konzipierung des Community-Editors zur 3D-Gebäudemodellierung wird daher dieses Konzept gewählt. Somit wird die parallele Bearbeitungen unterschiedlicher Modellkomponenten zur gleichen Zeit und identischer Modellteile nacheinander ermöglicht.

Der in Kapitel 3.2 erwähnte modulare Aufbau des Modells aus Teilmodelle und Modell-

	Letzter Schreibzugriff setzt sich durch	Live-Bearbeitung durch alle Nutzer	Sperrung einzelner Modell-komponenten	Vollständige Sperrung des gesamten Modells
Realisierungsaufwand	**sehr gering**	**groß**	**gering**	**gering**
Risiko gegensätzlicher Überschreibungen	**sehr hoch** steigt mit Nutzerzahl und Bearbeitungsdauer	**sehr hoch** steigt mit Nutzerzahl und Bearbeitungsdauer	**kein**	**kein**
mittlere Wartezeit eines Nutzers bis eine Modellierung möglich ist	**keine**	**keine**	**gering** steigt mit Nutzerzahl, sinkt mit Anzahl verschiedener Modell-komponenten	**sehr lang** steigt mit Nutzerzahl
Grad der Parallelisierung	**gering bis sehr hoch** sinkt bei längerer Bearbeitungsdauer und steigender Nutzerzahl	**gering bis sehr hoch** sinkt bei längerer Bearbeitungsdauer und steigender Nutzerzahl	**hoch**	– nur zeitlich versetzt möglich

Abbildung 3.5.: Vergleich möglicher Strategien zur Realisierung paralleler Modellierung

komponenten wird durch separate Dateien für jede Komponente realisiert. Diese lassen sich zu einem Zeitpunkt jeweils durch einen Nutzer bearbeiten. Auf diese Weise können viele Nutzer gleichzeitig an unterschiedlichen Modellkomponenten arbeiten, ohne sich gegenseitig zu behindern.

Ein Nutzer kann beispielsweise das Modell eines Raumes bearbeiten und ein anderer Nutzer das 3D-Modell einer Lampe erstellen. Die Bearbeitung des Raum- oder Lampenmodells durch einen weiteren Nutzer kann erst nach Freigabe durch den vorhergehenden Bearbeiter oder Ablauf einer Sperrzeit erfolgen. Die Sperrzeit beginnt mit dem Bearbeiten einer verfügbaren Datei und gestattet dem bearbeitenden Nutzer eine Frist, in der nur er das Modell editieren kann. Nach Ablauf der Sperrzeit werden die getätigten Änderungen in das Gesamtmodell übernommen und die Datei wieder für alle Nutzer freigegeben. So kann verhindert werden, dass einzelne Dateien dauerhaft blockiert werden, weil etwa eine Bearbeitung von Dateien vortäuscht und diese nicht wieder frei geben wird, oder weil ein in Bearbeitung eines Modells befindlicher Rechner abstürzt.

Durch die Zerlegung in Teilmodelle und Modellkomponenten (siehe Tabelle 3.1) kann ein kleinstmöglicher Bereich der Modellierung blockiert werden um die Einschränkungen für andere Nutzer minimal zu halten. Damit andere Nutzer nachvollziehen können, welche Teile des Modells sie aktuell bearbeiten können und welche sich in Bearbeitung durch

andere Nutzer befinden, werden gesperrte Modellbereiche durch Schloss-Icons markiert. Das erhöht die Intuitivität und Verständlichkeit der Benutzung.

Durch die verschiedenen Perspektiven, Motivationslagen [86] und Arbeitsweisen der Modellierenden können Kompatibilitätsprobleme zwischen den einzeln angefertigten Modellteilen auftreten. Auf solche Kompatibilitätsprobleme zwischen zwei Teilmodellen, beispielsweise unterschiedlich hohe Modelle benachbarter Räume, kann jeder Nutzer durch das setzten spezieller To-Do-Markierungen aufmerksam machen. An der Modellierung der beiden Teilmodelle noch nicht beteiligte Nutzer können dann dazu beitragen persönliche Konflikte zwischen bisherigen Autoren zu vermeiden und die Kompatibilität der betreffenden Teilmodelle herstellen.

Edit Wars und Vandalismus

Ausgehend von der Annahme, dass schadhafte Editierungen am Modell durch Edit Wars oder komplexeren Vandalismus nicht einfach von gutwilligen Editierungen unterschieden werden können [87], stellt das leichte Wiederherstellen ehemaliger Zustände durch Rückgängig machen von Änderungen eine gute Alternative zum Versuch dar, Schaden direkt zu verhindern [88, S. 63].

Eine Frustration von Nutzern durch fälschliches Blockieren produktiver Beiträge zum Modell gilt es zu vermeiden. Ein einfaches Zurücksetzten des gesamten Modells auf einen ehemaligen Zustand vor einem Vandalismusfall würde alle parallel zum Vandalismus stattgefundenen, gutmütigen Veränderungen ebenfalls rückgängig machen. Daher bietet sich eine Versionierung der einzelnen Teilmodelle an.

Durch identifizierbare Versionen der Teilmodelle mit generischen Versionsnummern und Zeitstempeln können Änderungen am Modell historisch nachvollzogen werden. Es wird somit unmöglich einmal versionierte Modellkomponenten mutwillig komplett aus dem Modell zu entfernen. Gleichzeitig bietet die Versionierung einzelner Teilmodelle Nutzern bei der Erstellung der Modelle eine Rückgängig-Funktion bei der Bearbeitung, wenn sie feststellen, dass ihre letzte Bearbeitung nicht den gewünschten Mehrwert erzielen konnte. Sie können ihre letzten Änderungen verwerfen und auf die vorherige Version des Teilmodells zurück greifen.

Die Idee nachvollziehbarer Entwicklung durch Versionen existiert bereits, beispielsweise für textbasierte Anwendungen wie Wikipedia, wo Änderungen an Artikeln Satz für Satz nachvollziehbar gemacht werden. Dieses Konzept wird auf die Modellkomponenten des heterogenen Gebäudemodells übertragen und angepasst.

Existieren für einzelne Teilmodelle Versionen, auf welche die Community Zugriff hat, kann zwar Vandalismus leicht behoben werden, jedoch bringt diese Methode auch neue

Version	Nutzer	Tags
1.0	B	-
1.1	A	-
1.2	B	-
1.3	B	-
1.1	A	-
1.3	B	-
1.1	A	-
1.1	C	Edit War
	-A, B, C gesperrt	
1.3	S	Edit War-Cooldown
	-A, B, C noch Zeit t gesperrt	

Tabelle 3.6.: Schematischer Ablauf der Bearbeitung eines Teilmodells Lampe_XY mit Edit War zwischen Nutzern A und B

Probleme mit sich. Beginnen unterschiedliche Nutzer oder Nutzergruppen wechselseitig eine durch sie bevorzugte Version eines Teilmodells immer wieder als die aktuelle Version auszuwählen und hervorzuheben [58, S. 456,459], so überschreiben sich die Änderungen in einem unproduktiven Prozess fortlaufend. Solche Edit Wars behindern die produktive Entwicklung des Modells. Wikipedia versucht gegen sich gegenseitig überschreibende Zurücksetzungen von Versionen mittels einer 3 Reverts Rule vorzugehen. Diese besagt, dass ein Artikel nicht mehr als dreimal in 24 Stunden zurückgesetzt werden soll [58, S. 454]. Diese Idee wird unter Ausnutzung der Resultate von [58, S. 458] weiterentwickelt. Gegen Konflikte Community-getriebener Prozesse durch eine steigende Zahl involvierter Parteien vorzugehen, lässt sich über die Einführung zweier spezieller Markierungen, sogenannter Tags, realisieren. Mit Hilfe eines Edit War-Tags, sowie eines Edit War-Cooldown-Tags können betroffene Teilmodellversionen von beliebigen Nutzern markiert werden, welche einen Edit War feststellen. Eine Angabe der im Edit War involvierten Nutzer ermöglicht es die Bearbeitung der Versionen durch diese Nutzer vorübergehend zu sperren. Mit solchen Tags markierte Versionen ziehen die Aufmerksamkeit unbeteiligter Nutzer auf sich, welche helfen können den Konflikt zu beenden.

Tabelle 3.6 zeigt den Umgang mit Edit Wars durch das Zurücksetzen von Teilmodellversionen am Beispiel eines Teilmodells Lampe_XY. Der zeitliche Verlauf ist von oben nach unten in der Tabelle dargestellt. Nach anfänglicher, produktiver Versionsentwicklung des Teilmodells Lampe_XY entsteht zwischen Nutzer A und B einen Edit War. Ein weiterer Nutzer C markiert die aktuelle Version als Edit War unter Angabe der involvierten Parteien A und B. Die Parteien A, B und C können nun die Version des betreffenden Teilmodells nicht länger bearbeiten. Dadurch, dass Nutzer C sich freiwillig in diese Sperrsituation bringt und einen unbeteiligten Nutzer zur Entscheidung in dieser

Angelegenheit einberuft, wird diese Rolle für involvierte Parteien des Edit Wars uninteressant. Ein unbeteiligter Nutzer S kann schlichtend in den Konflikt eingreifen und setzt das Teilmodell auf eine von ihm gewählte Versionsnummer. Erst eine gewisse Zeit t nach der schlichtenden Entscheidung ist die Bearbeitung des Teilmodells durch die im letzten Edit War genannten Nutzer wieder möglich. In dieser Zeit werden gesperrte Nutzer gezwungen ihre Aufmerksamkeit auf andere Teilmodelle zu verlagern.

Ein gemeinsames agieren mehrerer Nutzer im Edit War ist unrentabel. Im Beispiel müssten bereits mindestens Nutzer B, C und S gemeinsam vorgehen um ihre Version durchzusetzen. Zwei der drei beteiligten Angreifer werden bei dem Vorgehen vorübergehend gesperrt und Nutzer A könnte nach der Edit War-Cooldown-Zeit einen erneuten Edit War mit den Parteien B, C, S melden. Dies würde weitere Nutzer zum Schlichtungsprozess hinzuziehen und schlussendlich könnte sich die angreifende Partei gemäß der getätigten Annahme einer Mehrheit produktiver Nutzer nicht länger durchsetzen. Gegenüber der 3 Reverts Rule bietet das Konzept den Vorteil, dass beliebig häufig auf alte Versionen zurückgesetzt werden kann, solange die Zurücksetzungen nicht als Edit Wars markiert werden.

Login

Durch die Umsetzung eines Login-Mechanismus zur Modellierung können Bearbeitungen einzelner Nutzer mit ihrem Benutzernamen in Verbindung gebracht werden. Die Einführung eines Logins stellt zwar eine gewisse Einstiegshürde für neue Nutzer dar, jedoch bietet sie entscheidende Vorteile. So ist es möglich, sitzungsübergreifende Statistiken anzufertigen, die Rückschlüsse auf das Nutzerverhalten ermöglichen. Das kann interessant sein, um z.B. zu untersuchen, ob auch bei der Modellierung von Gebäuden ein Großteil der Modelle durch einen kleinen Teil der Nutzer erstellt wird, wie etwa bei der Entwicklung des Gebietes um London im OSM-Projekt [60].

Vollständig anonyme Beiträge, welche von anderen Nutzern in Konfliktsituationen provozierend empfunden werden können [58, S. 458], werden so ebenfalls verhindert. Fällt ein Nutzerprofil wiederholt auf, weil dadurch getätigte Bearbeitungen gehäuft als Vandalismus oder Edit Wars markiert werden, so können mit dem Profil verknüpfte Bearbeitungen am Modell einfach gefunden und rückgängig gemacht werden. Zudem können Nutzerprofile besondere Hervorhebungen, wie beispielsweise auffällige Avatars als Belohung für aktives Modellieren erhalten, und den Nutzern kann ihre eigene, getätigte Arbeit in Form von Ranglisten sowie Statistiken vor Augen geführt werden.

Nutzerprofile mit Kommunikationsoptionen ermöglichen zudem einen gezielten Dialog zwischen Nutzern, sowie den Aufbau von Reputationen, welche einen Gruppenprozess mitgestalten. Eine Zusammenfassung der geleisteten Beiträge eines Nutzers innerhalb seiner Sitzung vor dem Logout eines Nutzerprofils drückt Wertschätzung für die frei-

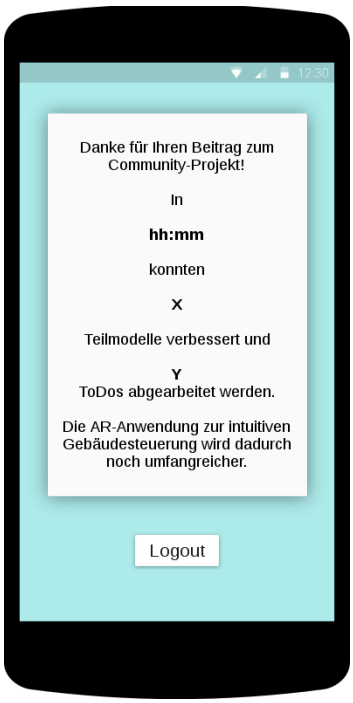


Abbildung 3.6.: Zusammenfassung des erreichten Beitrags eines Nutzers vor dem Logout

willige Arbeit des Nutzers aus, wie in [89] empfohlen wird. Sie kann als motivierend empfunden werden und dazu beitragen, dass Nutzer erneut aktiv werden.

Aufmerksamkeitsmanagement

Bei der verteilten Erstellung von komplexen Datensätzen treten immer wieder Situationen auf, welche besonderer Aufmerksamkeit bedürfen. Die Aufmerksamkeit aktiver Community-Mitglieder auf bestimmte Probleme zu lenken, trägt dazu bei, dass der Community-Prozess effizient und zielführend stattfinden kann. Hierfür werden sogenannte To-Do-Tags verwendet, mit welchen sich Teilmodelle markieren lassen. Ein Beispiel für die Anwendung dieses Tags ist das Management von Situationen, die spezielles Hoheitswissen erfordern. Nutzer, welche nicht über benötigtes Hoheitswissen verfügen, können entsprechende Komponenten mit einem To-Do-Tag markieren, die Aufmerksamkeit anderer Nutzer auf diese Situation lenken und somit den Modellierungsprozess dennoch möglichst weit vorantreiben. Nutzer mit entsprechendem Zugang zu den benötigten Spezialinformationen können durch Abarbeitung solcher Markierungen ihr besonderes Wissen effizient in die Modellerstellung einbringen.

Solche To-Do-Markierungen können auch in wenigen Sekunden, etwa von Nutzern unter Zeitdruck, angelegt werden. Sie lassen sich zur Erinnerung einsetzten, damit markierte

Ergänzungen nicht in Vergessenheit geraten und zu einem späteren Zeitpunkt von beliebigen Nutzern bearbeitet werden können. Die in Kapitel 3.3.2 aufgeführten Edit War- und Edit War-Cooldown-Tags dienen ebenfalls dem Aufmerksamkeitsmanagement der Community.

Eine einsehbare Liste aller offenen To-Do-Markierungen bietet Nutzern die Chance ihre Arbeitskraft dort einzubringen, wo sie benötigt wird. Die Bearbeitung solcher To-Do-Flags kann gesondert in den Nutzerstatistiken hervorgehoben, durch spezielle Achievements belohnt und in der Arbeitszusammenfassung betont werden. Das schafft zusätzliche Motivation für Nutzer, sich an die markierten Komponenten zu setzten, statt neue Teilmodelle zu erstellen. Eine Dringlichkeitsangabe als Zusatzinformation bei dem Setzen solcher Markierungen ermöglicht zudem eine priorisierte Auflistung noch offener Aufgaben und kann den Community-getriebenen Modellierungsprozess effizient unterstützen [90].

3.4. Die generische Weiterverarbeitung der Modelldaten zur Geräteansteuerung

Die durch den Community-Editor zusammengetragenen Modelldaten sollen eine Steuerung von vernetzten Geräten in Gebäuden durch eine AR-Anwendung ermöglichen. Dafür ist es insbesondere nötig, dass Adressierungs- und Steuerinformationen aktiver Modellkomponenten in Form von Nutzerschnittstellen (engl. User Interface (UI)) aufgearbeitet und dem Endanwender zur Verfügung gestellt werden. Über die Nutzerschnittstelle kommunizieren die Nutzer mit der AR-Anwendung, welche Steuerbefehle an zugehörige Steuerserver absendet. Diese ermöglichen über entsprechend angebundene Übertragungsmedien eine Kommunikation mit den aktiven Geräten, wie in Abbildung 3.7 schematisch dargestellt ist.

3.4.1. Modelleigenschaften zur generischen Erstellung von Nutzerschnittstellen

Untersuchungen zeigen, dass das Erstellen grafischer Nutzerschnittstellen in mobilen AR-Anwendungen zahlreiche Herausforderungen mit sich bringt [91, S. 71-80]. Um einheitliche UIs sicherzustellen, ist es erstrebenswert diese generisch aus den im Modellierungsprozess gesammelten Daten zu erzeugen. Ähnlichkeiten zwischen Geräten und vergleichbaren Funktionalitäten sollen sich auch in deren Bedienung widerspiegeln, um vertraute und intuitive Interaktionen zu gewährleisten. Beispielsweise soll die An- und Ausschaltefunktionalität einer einfachen Lampe hierfür das gleiche UI-Widget besitzen, wie bei einem Lampenmodell das zusätzlich noch eine Einstellung der Leuchtfarbe anbietet. Folglich ist eine Ontologie der Geräte entsprechend ihrer Steuermöglichkeiten

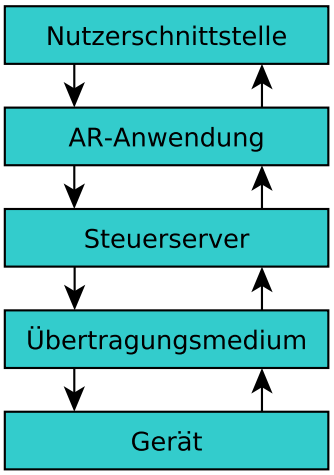


Abbildung 3.7.: Kommunikationsstack der AR-Anwendung

erforderlich.

Digitale Geräte lassen sich über Funktionsparameter steuern. Die Funktionsparameter einschließlich zugehöriger Datentypen und Wertebereiche der Steuerfunktionen von Geräten sind wesentliche Informationen, die bei der Modellierung aktiver Komponenten erfasst werden müssen, um generische Nutzerschnittstellen zu ermöglichen. Hieraus lassen sich Bereichsgrenzen, wie beispielsweise Minimum und Maximum der Funktionsparameter, sowie die Anzahl möglicher Steuerzustände des betreffenden Gerätes bestimmen. Da die einzelnen Funktionsparameter maschinenlesbar in Form von Bits über ein Kommunikationsprotokoll unter Geräten ausgetauscht werden, existiert eine endliche Anzahl an Zuständen. Die Bedienung eines Gerätes kann daher durch eine passende Auswahl aus allen möglichen Steuerzuständen erfolgen.

Die Namen der Steuerfunktionen und ihrer Parameter können zur Beschriftung der Steuerschnittstellen herangezogen werden. Für komplexere Steuerfunktionen, welche sich aus der Kombination mehrere Parameter zusammensetzen, ist eine Gruppierung dieser Parameter nötig, um die Zusammengehörigkeit auch im UI abzubilden.

Um den Benutzer bei der Bedienung des UI zu unterstützen, sind Informationen darüber hilfreich, welche Parameter für eine erfolgreiche Gerätesteuerung optional und welche obligatiorisch anzugeben sind. Auch Angaben über den Kontext von Parametern können bei der Generierung intuitiver UIs unterstützend sein. So lassen sich beispielsweise auf mobilen Geräten kontextabhängige Tastaturlayouts bei der Parametereingabe oder -auswahl einblenden [92], [93], [94], wie in Abbildung 3.8 am Beispiel einer numerischen

Aktion	Beispiel
an-/ ausschalten	Lampe, Beamer
muten/ unmuten	Lautsprecher
ver-/ entriegeln	Türschloss, Fensterriegel
öffnen/ schließen	Türblatt, Fenster
aktivieren	Kaffemaschine, Waschmaschine
zurücksetzen	Beamerkalibrierung, Beleuchtungsfarben
hell/ dunkel dimmen	Lampe, Jalousie, Beamer
hoch/ runter regeln	Jalousie, Tafel, Rollo, Leinwand, Heizung
vor/ zurück regeln	Beamerfokus, Schublade
rechts/ links regeln	Lautsprecherpfade
laut/ leise regeln	Lautsprecher
hinzufügen/ entfernen	Bassanteil, Rotanteil im Beamer
drehen, kippen	Jalousielamellen, Beamerbild
auswählen	Beamersignalquelle, Radiofrequenzbereich
Farbe einstellen	RGB-Lampe, Beamer, Raumbelegungsanzeige
Temperatur einstellen	Heizung, Klimaanlage
Datum angeben	Heizung
Uhrzeit einstellen	Flurbeleuchtung, Radiowecker
Text eingeben	Zugangskontrollsystem
Zahl eingeben	Fahrstuhl
Datei auswählen	Lautsprecher
Bereich auswählen	Zugangskontrollsystem

Tabelle 3.7.: Übersicht existierender Steueraufgaben im Gebäudeautomationsumfeld

Uhrzeiteingabe zu sehen ist.

3.4.2. Generische Nutzerschnittstellen erzeugen

Eine Möglichkeit zur generischen Schnittstellenerzeugung besteht darin eine Abbildung von Steuermöglichkeiten verschiedener Geräte auf vorhandene Bedienschnittstellen zu erzeugen. Hierfür wurden zunächst existierende Steuermöglichkeiten verschiedener Geräte im Gebäudeautomationsumfeld in Tabelle 3.7 exemplarisch zusammengetragen und klassifiziert. Zudem erfolgte eine Recherche gängiger Eingabekontrollelemente unterschiedlicher Technologien, welchen in Tabelle 3.8 aufgeführt werden. Hieraus wurde anschließend eine strukturierte Zuordnung von Steuermöglichkeiten zu Eingabekontrollelementen vorgenommen, welche in Tabelle 3.9 zu sehen ist.

Es ist eine Vielzahl an Steuermöglichkeiten für die unterschiedlichsten Geräte der einzelnen Gebäudeautomationsprotokolle denkbar. Die genannten Steueraufgaben haben

daher exemplarischen Charakter zur Verdeutlichung der Methodik und besitzen keinen Anspruch auf Vollständigkeit. Um eine vollständige Liste aktueller Steuermöglichkeiten für die Geräte unterschiedlicher standardisierter Gebäudeautomationsprotokolle zu erhalten, können für die einzelnen standardisierten Protokolle alle lizenzierten Geräte der verschiedenen Hersteller nach Steuermöglichkeiten durchgesehen und aufgelistet werden. Übersichten zertifizierter Produkte finden sich beispielsweise für KNX [95], Z-Wave [96], BACnet [97] oder LON [98] auf den Internetauftritten der Organisationen und Firmen.

Abbildung 3.8.: Parametereingabe am Beispiel einer Uhrzeit mit numerischem Tastaturlayout

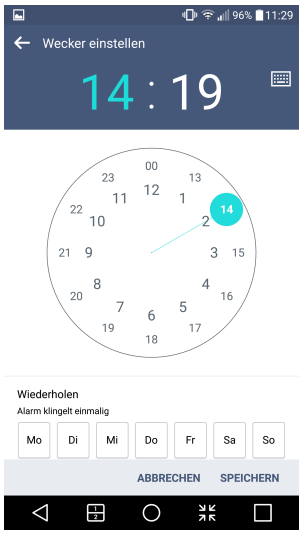


Abbildung 3.9.: Parameterauswahl am Beispiel einer Uhrzeit mit Ziffernblatt-Widget

Die Steuerung von Geräten erfolgt durch das setzen einzelner Parameterwerte. Dies kann durch eine direkte Eingabe von Wert und Größe wie etwa Temperatur, Datum, Zeit, Farbe, Text, Änderungen solcher Größen oder Prozentangaben geschehen. Alternativ zur direkten Eingabe kann auch eine Auswahl aus vorgegebenen Möglichkeiten stattfinden. Der Übergang zwischen direkten Eingaben und Auswahlelementen ist gelegentlich fließend. Inhaltsabhängige Eingabefelder mit kontextabhängigen Tastaturlayouts oder spezifische Widgets, wie etwa ein Ziffernblatt zur Eingabe der Uhrzeit, sowie die Kombination beider Möglichkeiten, wie in den Abbildungen 3.8 und 3.9 dargestellt, sind üblich.

Die Parameterauswahl lässt sich, wie in Abbildung 3.10 dargestellt, entsprechend der Anzahl an Auswahlmöglichkeiten unterteilen. Gewisse Prozesse, wie etwa das Brühen einer Tasse Kaffee benötigen lediglich eine Aktivierung und enden nach einer gewissen Zeit t selbständig. Der Spezialfall einer diskreten Auswahl aus zwei möglichen Zuständen findet sich in zahlreichen Beispielen aus Tabelle 3.7, wie etwa dem An- und Ausschalten einer Lampe, oder dem Ver- und Entriegeln eines Türschlosses wieder. Steueraufga-

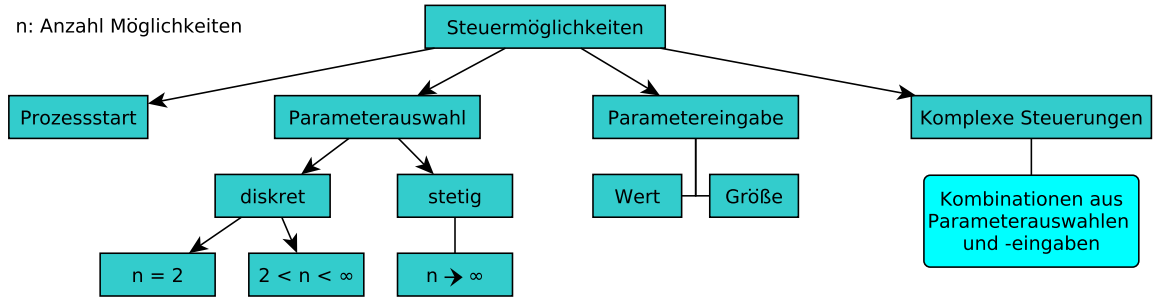


Abbildung 3.10.: Klassifizierung von Steuermöglichkeiten

ben wie beispielsweise Dimmen von Lampen, Regeln von Rollos oder Lautstärke- bzw. Basseinstellungen an einem Lautsprecher lassen sich durch eine Auswahl endlich vieler Möglichkeiten beschreiben. Die Anzahl der möglichen Zustände n wird hierbei durch den Wertebereich des entsprechenden Parameters begrenzt. Lässt sich eine Lampe beispielsweise durch einen 8-Bit Parameter in ihrer Helligkeit regeln, so existieren $2^8 = 265$ Zustände zur Helligkeitssteuerung. Für sehr große n lässt sich eine stetige Parameterauswahl simulieren, indem die Schritte zwischen den tatsächlich endlichen n Zuständen so klein werden, dass sie von Nutzern nicht länger als Schritte erkannt werden.

Umfangreichere Steueraufgaben, die k Parameterwerte mit $k > 1$ benötigen oder gestatten, lassen sich aus k Steuermöglichkeiten für einen Parameterwert zusammensetzen. Beispielsweise lässt sich die Steuerung der Leuchtfarbe einer RGB-Lampe, wie in Abbildung 3.11 dargestellt, mit drei Parametern für Rot, Grün und Blau realisieren, die jeweils einen Farbwert definieren. Eine Mehrfachauswahl, wie etwa Wochentage, an denen die Heizung eines Raumes aktiv werden soll, lässt sich analog aus sieben an/aus Steuermöglichkeiten für die sieben Wochentage umsetzten.

Eingabekontrollelemente als Komponenten eines UI stehen für unterschiedliche Technologien in unterschiedlicher Anzahl und Form, beispielsweise als Widgets, zur Verfügung. Zusätzlich zu den existierenden Vorlagen können je nach gewählter Technologie beliebig weitere, individuelle UI-Komponenten entwickelt werden [99], deren Komplexität und Erscheinung nicht vorhersehbar sind. Je nach Anwendungsfall lassen sich Erscheinung und Verhalten der Komponenten anpassen, um z.B. ein responsives Design zu realisieren [100] oder Webanwendungen nativ wirken zu lassen [101, S. 123].

In Tabelle 3.8 sind exemplarisch für Mac OS, Android und HTML einige, gängige UI-Kontrollelemente inklusive einer möglichen grafischen Umsetzung der Komponenten aufgeführt. Weitere Widgets, Technologien und Frameworks zur UI-Erzeugung könnten in der unvollständigen Liste nach belieben ergänzt werden. Die vorgestellte Auswahl soll

Abbildung 3.11.: Umsetzung komplexerer Steueraufgaben aus mehrfachen Parameterauswahlmöglichkeiten am Beispiel der Farbeinstellung einer RGB-Lampe

jedoch reichen, um daran die beschriebene Methodik zu verdeutlichen.

Aufbauend auf der vorgestellten Einteilung von Steueraufgaben lässt sich eine Zuordnung konkreter Steueraufgaben zu UI-Widgets durchführen, welche als Vorschrift zur generischen UI-Erstellung herangezogen werden kann. Somit können aus Steuerinformationen automatisch Interaktionsschnittstellen für Nutzer generiert werden. Wie in Abschnitt 3.4.3 beschrieben, ist eine solche Abbildung nicht eindeutig, da beispielsweise ein und dieselbe Steueraufgabe auf unterschiedliche Widgets abgebildet werden kann. Die Abbildung ist zudem nicht notwendigerweise injektiv, da unterschiedliche Steueraufgaben auf identische Widgets abgebildet werden können. Auch müssen nicht zwangsläufig alle Widgets durch die Abbildungsvorschrift generisch erzeugbar sein, sodass die Abbildung ebenfalls nicht notwendigerweise surjektiv ist. Tabelle 3.9 stellt folglich exemplarisch eine von vielen möglichen Abbildungen von Steueraufgaben auf UI-Widgets dar, um die Methode zur generischen Schnittstellenerzeugung zu veranschaulichen. Sie gestattet die automatische Erstellung von UIs für bekannte Steueraufgaben auf der Basis ausgewählter UI-Widgets.

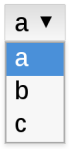
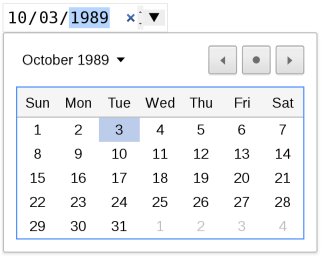
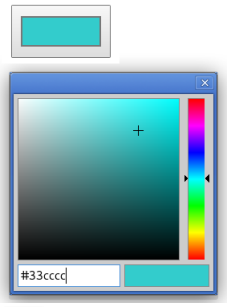
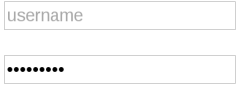
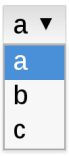
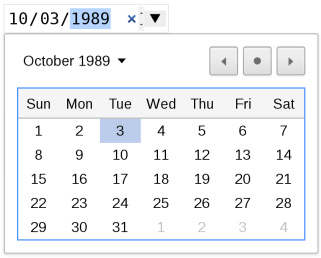
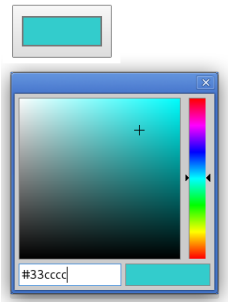
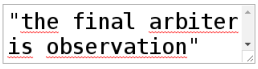
Apple [94]	Android [93]	HTML5 [92]	Beispiel
Push Button	Push Button	Button	
Radio Button	Radio Button	Radio Button	
Checkbox	Checkbox	Checkbox	
Combination Box	Spinner	Select Element	
Date Picker	Date Picker	Date Control	
Time Picker	Time Picker	Time Control	
Colors Window	–	Color Well	
Switches	Toggle Button	–	
Slider	Slider	Slider Control	
Stepper	Number Picker	Spinner Control	
Text Input Field with several input-types	Text Field with several input-types	Text Field with several input-types	

Tabelle 3.8.: Auflistung gängiger UI-Widgets verschiedener Technologien

Funktionsparameter	UI-Widgets	Beispiel
Aktivierung		
Prozessstart	Push Button	
Auswahl [1]		
n = 2	Toggle Button	
$2 < n \leq 7$	Radio Button	
$7 < n < 20$	Spinner	
n >20	Slider	
Angabe		
Datum	Date Picker	
Temperatur	Text Field + Label	
Zeit	Time Picker	
Farbe	Color Well	
Text	Textarea	
Änderung	Number Picker	
Prozentangabe	Slider	
Datei	File Upload	

[1] n Anzahl der Auswahlmöglichkeiten

Tabelle 3.9.: Exemplarische Abbildung von Steueraufgaben auf UI-Widgets

3.4.3. Limitierungen der präsentierten Methode zur generischen Nutzerschnittstellenerzeugung

Während die vorgestellte Methodik eine generische Schnittstellenerzeugung aus Modelldaten durch die Abbildung bekannter Steueraufgaben auf eine ausgewählte Menge UI-Widgets generell ermöglicht, führt das Resultat nicht zwangsläufig zu angemessenen, intuitiven oder gar brauchbaren Resultaten. Eine einzelne Auflistung aller Steuerzustände eines digital steuerbaren Gerätes ist stets möglich, da der Zustand solcher Geräte über eine endliche Menge an Bits codiert ist. Am Beispiel einer RGB-Lampe, deren Rot, Grün und Blaulichtanteil jeweils mit 8 Bit codiert sind, lässt sich jedoch leicht nachvollziehen, dass eine GUI aus $2^{3 \cdot 8} = 2^{24}$ Buttons zur Auswahl aller möglicher Farbkombinationen in einer wenig intuitiven und unübersichtlichen UI resultieren würde. Zudem sind zukünftige Steuermöglichkeiten und UI-Widgets nicht vorhersagbar und bekannt. Somit können sie in einer aktuellen Abbildungsvorschrift noch keine Berücksichtigung finden. Jedoch lassen sich komplexe Steuermöglichkeiten stets durch Kombinationen einfacher Steuermöglichkeiten erstellen. Auf diesem Weg können über die vorgestellte Methode auch komplexe UIs erstellt werden.

Da es eine Vielzahl verschiedener UI-Elemente mit zum Teil gleicher Funktionalität gibt, ist die Abbildung von Steuerfunktionen auf Bedienschnittstellen keinesfalls eindeutig. Selbst eine primitive Aufgaben wie die Auswahl eines von zwei möglichen Zuständen lässt sich auf zahlreiche UI-Widgets, wie etwa eine Checkbox, einen Toggle Button, zwei Radio Buttons, einen Spinner mit zwei Elementen u.v.m. abbilden. Welches Widget für einen spezifischen Einsatzfall besonders geeignet ist, da es als intuitiver wahrgenommen wird und somit eingesetzt werden sollte, ist eine komplexe Frage für weiterführende Forschungen in diesem Gebiet. Für den in der Einleitung vorgestellten Anwendungsfall wäre beispielsweise eine Anlehnung der zu verwendenden UI-Widgets an tatsächlich im Gebäude vorhandene physikalische Hardwareinteraktionsschnittstellen, wie z.B. Lichtschalter, denkbar.

Eine konkrete Steueraufgabe lässt sich nicht nur durch unterschiedliche UI-Widgets darstellen, sondern auch unterschiedlich interpretieren. So wäre denkbar, dass die Leuchtfarbe der bereits erwähnten RGB-Lampe durch drei Slider gesteuert wird, welche den Anteil an rotem, grünem und blauem Licht angeben. Stehen alle drei Slider auf ihrem Minimum, könnte das als deaktivieren der Lampe interpretiert werden, oder aber eine Standardleuchtfarbe erwartet werden. Für den ersten Fall müsste das UI keine weitere An- bzw. Ausschaltinteraktion der Lampe bieten. Alternativ könnte die Leuchtfarbe auch über ein Color Well eingestellt werden und bei Auswahl der Farbe schwarz die Deaktivierung der Lampe erfolgen.

Ein weitere Hürde für intuitive UIs ist die Interpretation und Darstellung komplexer Parameter. Beispielsweise kann ein Datum als drei aufeinanderfolgende Integer dargestellt

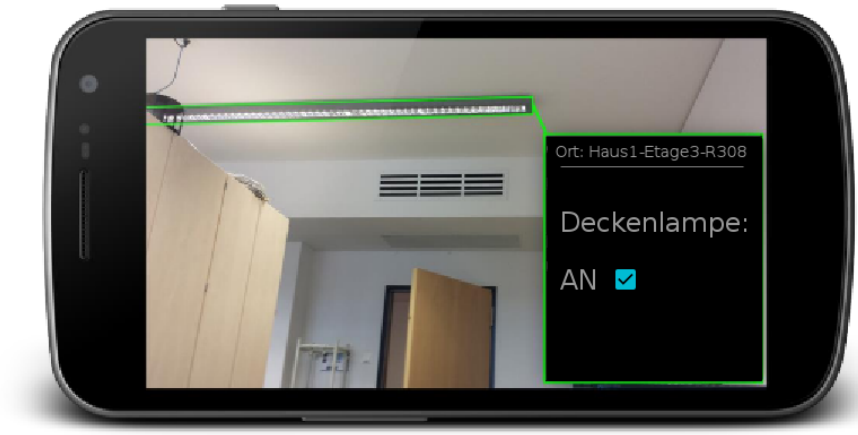


Abbildung 3.12.: Mockup der AR-Anwendung zur Lampensteuerung mit Checkbox

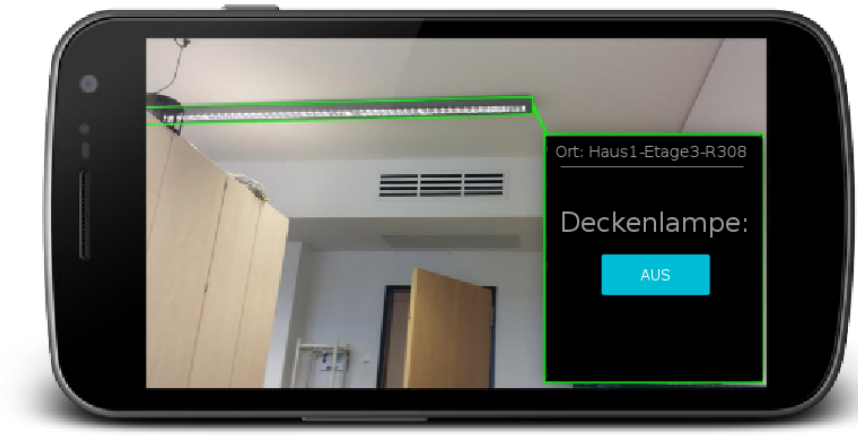


Abbildung 3.13.: Mockup der AR-Anwendung zur Lampensteuerung mit Toggle Button

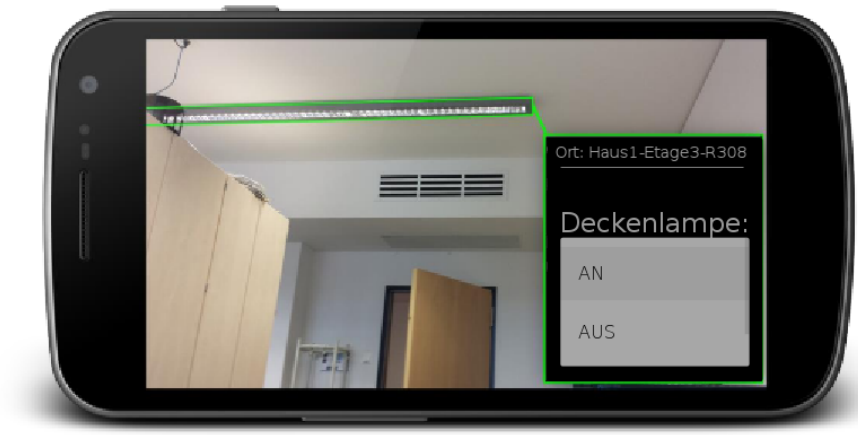


Abbildung 3.14.: Mockup der AR-Anwendung zur Lampensteuerung mit Spinner

Abbildung 3.15.: Mockup der AR-Anwendung zur Lampensteuerung mit Radio Buttons

Abbildung 3.16.: Mockup der AR-Anwendung zur Lampensteuerung mit Schalter

werden, welche jedoch über eine innere Logik und unterschiedliche Interpretationsmöglichkeiten verfügen. Die Angabe eines Datums als 10.11.12 könnte beispielsweise entsprechend der Schemata TT.MM.JJ, MM.TT.JJ, JJ.MM.TT, JJ.TT.MM, TT.JJ.MM oder MM.JJ.TT interpretiert werden. Handelt es sich bei dem angegebenen Datum um den zehnten November 2012, den elften Oktober 2012 oder gar um den elften Oktober 1912? Interpretiert der Nutzer eines UI eine Angabe anders, als die Logik der Schnittstelle, ist eine Fehlbedienung der Anwendung die Folge.

Das Problem generischer Erstellung intuitiver Nutzerschnittstellen ist somit folglich nicht vollständig lösbar. Die vorgestellte Methodik gestattet jedoch eine im Zweifel komplizierte und individuell verbesserbare Möglichkeit zur generischen Schnittstellenerzeugung für unterschiedliche Steueraufgaben aus Modelldaten.

4. Umsetzung vorgestellter Konzepte und Anforderungen

Das folgende Kapitel hat zum Ziel die Umsetzung der im Konzeptkapitel vorgestellten Ideen unter Beachtung der eingangs erwähnten Anforderungen zu demonstrieren. Es werden wesentliche Aspekte aus Kapitel 3 aufgegriffen und prototypische Umsetzungen skizziert.

4.1. Der Modellaufbau

Die beschriebene AR-Anwendung verlangt ein umfangreiches Gebäudemodell mit Positions-, Steuer-, Bild- und 3D-Informationen zu Objekten. Wie in Kapitel 3.1 beschrieben bietet sich eine Zerlegung des Modells in Teilmodelle und Modellkomponenten an. Diese lassen sich getrennt verwalten und ermöglichen eine einfache Übertragung von Modellausschnitten, gefiltert nach relevanten Bereichen.

Diese Zerlegung wurde bei der Implementierung auch für die Datenhaltung und die GUI-Darstellung angewendet. 3D-Beschreibungen und Fotos werden nach Teilmodellklassen sortiert auf dem Dateisystem des Modellservers gehalten. Abbildung 4.1 zeigt exemplarisch die Datenhaltung für verschiedene Versionen der 3D-Beschreibung zweier Geräte D001 (Lampe) und D002 (Beamer) sowie die zugehörigen Bildinformationen.

Für jedes konkret modellierte Objekt verweist eine individuelle Metadatenbeschreibung auf den entsprechenden Modelltypen und beschreibt objektspezifische, relevante Informationen. Diese kann statt in einer XML-Datei, wie in [2] vorgeschlagen, als JSON-Datei umgesetzt werden. Bei gleichem Informationsgehalt lassen sich somit geringere Dateigrößen erzielen, was für die Übertragung dieser Informationen auf mobile Geräte im Rahmen der AR-Anwendung Relevanz hat. Kleinere Dateien lassen sich günstiger und schneller an mobile Geräte mit begrenzter Bandbreite übertragen. Die Haltung dieser individuellen Metadatenbeschreibungen bietet sich in einer dokumentenorientierten Datenbank, wie beispielsweise MongoDB an. Eine Erweiterung der individuellen Objektbeschreibungen durch zukünftige Änderungen ist auf diese Art nicht von einem Datenbankschema abhängig.

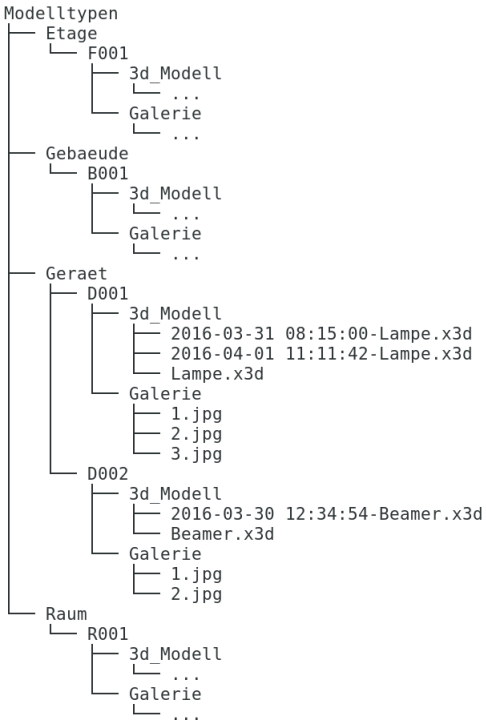


Abbildung 4.1.: Datenhaltung der 3D-Beschreibungen und Fotos

Zum Aufbau komplexer Modelle lassen sich die hierarischen Teilmodelle relativ zueinander positionieren. Zunächst erfolgt eine Zerlegung in die Teilmodellklassen Gebäude, Etage, Raum und Gerät (siehe Abbildung 3.1). Ein Gebäude kann mehrere Etagen mit darin positionierten Räumen enthalten. In einem Raum sind entsprechende Geräte enthalten. Sollte zukünftig ein höherer Detailgrad für Teilmodellklassen, wie etwa nicht technische Objekte oder Flure nötig werden, können zusätzliche Teilmodellklassen ergänzt werden. Denkbar wäre beispielsweise die Einführung einer Unterklasse von Räumen, um Flure gesondert zu modellieren.

Zur relativen Positionierung der Teilmodelle wird die Eigenschaft von X3D-Beschreibungen ausgenutzt, dass sich X3D-Dateien ineinander einbetten und über Transform-Elemente positionieren lassen. Listings A.1 bis A.4 im Anhang zeigen die relative Positionierung verschachtelter Teilmodelle. Abbildung 3.2 vermittelt einen Eindruck des hieraus resultierenden, abstrakten Modells. Hierfür ist es zweckmäßig jedes 3D-Modell in einer extra Datei zu beschreiben, welche mehrfach wiederverwendet werden kann.

Für die Erstellung intuitiver Auswahlkataloge mit bestehenden Teilmodelle wird ebenfalls auf diese Zerlegung der 3D-Modelle zurückgegriffen. Basierend auf dem in [81] beschriebenen Vorgehen können einzelne 3D-Modelle somit aus einer Vorschau ausgewählt werden, die sich per Drag-and-Drop auf die aktuelle Bearbeitungsfläche ziehen und in

Schlüssel	Wert
ID	Ger_003
Teilmodellklasse	Gerät
Modelltyp	Gerät/D003/3d_Modell/Vase.x3d
Position	<Gebäude 42><Etage 1><Raum 101><0.5m><1.0m><0.8m>
Version	1.0
Tags	-
Letzte Editor	Nutzer1419
gesperrte Nutzer	-
Netwerkbus	-
Adresse	-
Steuerziele	Ger_001
Funktionstyp	-
Anwendungstyp	ATX(Sonstiges)

Tabelle 4.1.: Metadatenbeschreibung einer Vase zur Steuerung einer Lampe

einem komplexeren Modellteil wiederverwenden lassen.

Auch eine Steuerung aktiver Geräte durch nicht technische Objekte kann durch die Trennung von Modelltyp und Steuerinformationen einfach realisiert werden. In der AR-Anwendung kann beispielsweise eine Vase erkannt werden und mit dem Steuerinterface für eine Deckenlampe verknüpft werden. Neben Fotoinformationen und 3D-Modell der Vase verweist dazu die Metadatendatei der Vase durch die Angabe der Lampe als Steuerziel auf die Steuerinformationen der Lampe. Tabelle 4.1 zeigt dies in Anlehnung an die Metadatenbeschreibung der Lampe aus Tabelle 3.3.

Eine prototypische Umsetzung der Beschreibung von 3D-Modellen erfolgte durch die in Kapitel 2.2.1 vorgestellte XML-basierende Beschreibungssprache X3D. Diese stellt das Standardformat für 3D-Beschreibungen im Web dar und ermöglicht ein auf WebGL basiertes Rendering durch das X3DOM-Framework. Hiermit werden X3D-Elemente direkt im DOM verfügbar, was eine flexible Darstellung und Bearbeitung der 3D-Beschreibungen im Browser ermöglicht.

Zur Umsetzung der 3D-Modellierung bietet sich die Einbindung des in Kapitel 2.2.3 beschriebenen, Open-SourceComponent-Editors an. Basierend auf dem X3DOM-Framework stellt er ein technologisch passendes Werkzeug zur 3D-Bearbeitung dar, das durch viele nützliche Features, wie z.B. die Angabe von Maßeinheiten oder eine Orientierungshilfe für gewählte Kameraperspektiven auf die Szene, überzeugt.

4.2. Umsetzung des Modellierungswerkzeugs als Community-Editor

Um den Nutzer-getriebenen Modellierungsprozess von Gebäuden bestmöglich zu unterstützen, erfolgte die Entwicklung des Modellierungswerkzeugs entsprechend der in Kapitel 3.2 vorgestellten Eigenschaften Intuitivität, Zugänglichkeit, Verständlichkeit, Modularität, Delegierbarkeit und Intelligenz.

4.2.1. Intuitivität

Die GUI des Community-Editors unterstützt eine intuitive Handhabung durch die Unterteilung in logische Bereiche, wie in Abbildung 4.2 erkennbar ist. Ein zu bearbeitendes Modell lässt sich im Katalogbereich (im Bild grün-weiß hinterlegt) auswählen. Dieser Katalog-Bereich ist entsprechend der Teilmodellklassen geordnet. Informationen über das aktuell ausgewählte Teilmodell werden dem Nutzer in einem Informationsbereich (blau) angezeigt. Über eine Tab-Leiste (beige) lässt sich die Arbeitsfläche (rot) an die gewünschte Bearbeitung anpassen.

Bildinformationen zu Teilmodellen lassen sich in der Galerie ergänzen. Zur Modellierung der 3D-Beschreibungen wird die Component-Editor-Arbeitsfläche (siehe Abbildung 4.3) angezeigt. Sie bietet dem Nutzer ein Orientierungs-Panel zur Hilfe bei der Perspektivwahl auf die Modellszene und eine Anzeige über die Ausrichtung des Koordinatensystems. In einem zusätzlichen Konfigurations-Panel wird das Einblenden eines magnetischen Hilfsgitters und die Einstellung einer Maßeinheit zur Übernahme vorhandener Bemaßungen ins Modell ermöglicht. Für die Eingabe von Metadaten wird eine entsprechende Formularseite (siehe Abbildung 4.4) eingeblendet. Ihr Layout orientiert sich an der Teilmodellklasse. Bei der Eingabe gültiger Formulardaten wird der Nutzer durch UI-Widgets unterstützt, indem existierende Vorauswahlen, wie beispielsweise mögliche Räume angeboten werden.

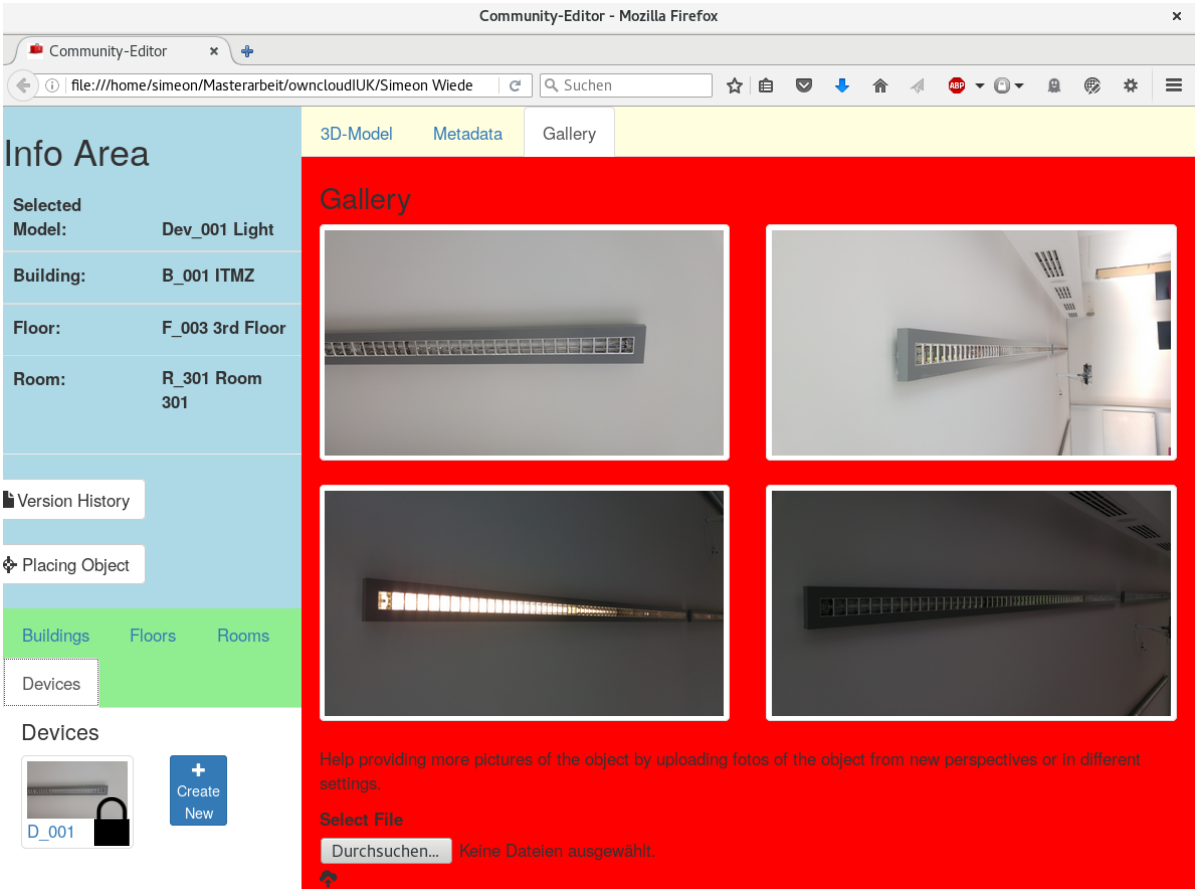
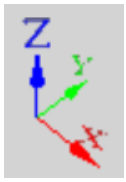
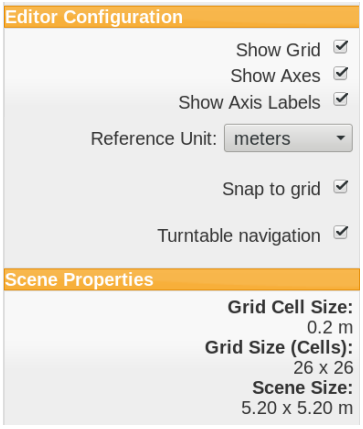


Abbildung 4.2.: GUI des prototypischen Community-Editors

Orientierungs-Panel



Koordinatensystem-ausrichtung



Konfigurations-Panel

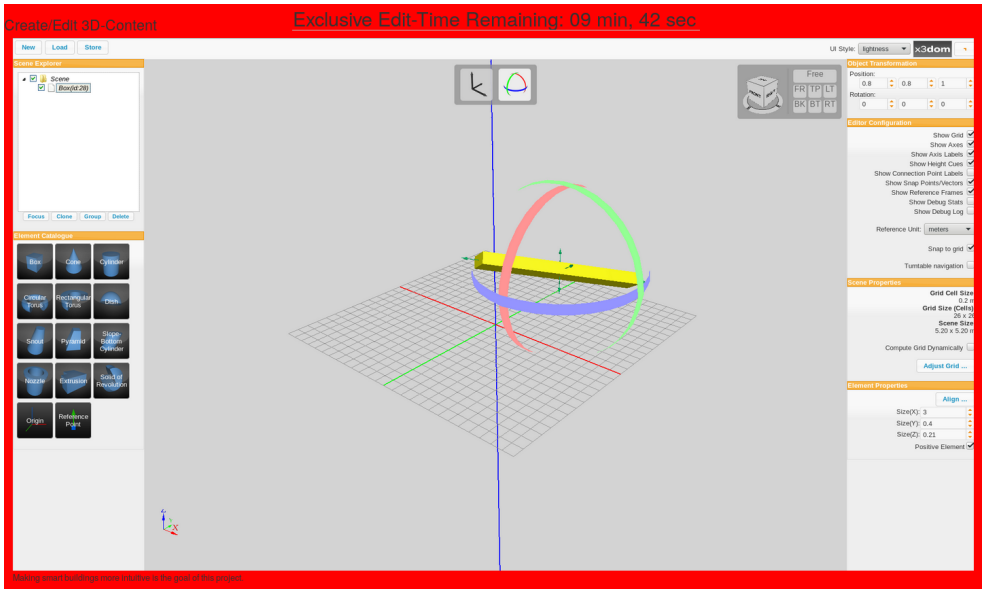


Abbildung 4.3.: Component-Editor-Arbeitsfläche zur 3D-Modellierung

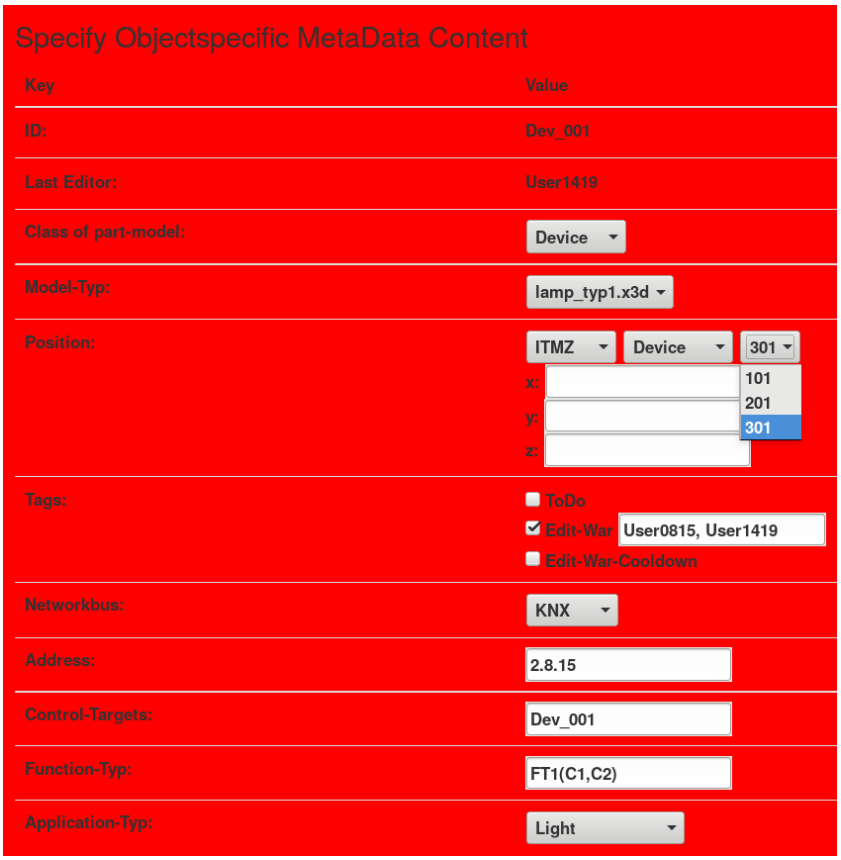


Abbildung 4.4.: Metadaten-Ansicht der GUI

4.2.2. Zugänglichkeit

Die Entwicklung des Community-Editors als Webanwendung gestattet es Nutzern des Editors ohne zusätzliche Installation von Spezialsoftware in einem modernen Browser an der Modellerstellung mitzuwirken. Hierfür kommen die typischen Web-Technologien Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) sowie JavaScript zum Einsatz.

Die Verwendung des Bootstrap-Frameworks sorgt zusätzlich für ein Responsive Design, sodass neben Workstations auch mobile Geräte wie Laptops, Tablets oder Smartphones zur Modellierung eingesetzt werden können. Eine Verwendung des Editors ist folglich weitestgehend unabhängig von Einsatzort, Gerätetyp und verwendetem Betriebssystem möglich und benötigt lediglich einen aktuellen Browser und eine Internetverbindung.

Unter Ausnutzung der WebGl-Schnittstelle aktueller Browser erfolgt die Darstellung der 3D-Inhalte. Zur Beschreibung dieser kommt, wie in Abschnitt 2.2.1 auf Seite 23 erläutert, die Standardbeschreibungssprache für 3D-Informationen im Web, X3D, zum Einsatz. Interaktion mit und Manipulation an den 3D-Objekten des Modells werden durch die Verwendung des vorgestellten X3DOM-Frameworks realisiert, welches X3D-Objekte im DOM verfügbar macht.

4.2.3. Verständlichkeit

Die englischesprachige GUI sowie sprachunabhängige Icons tragen dazu bei, dass die Verständlichkeit des Community-Editors möglichst hoch ist. Eine Verwendung des Editors erfordert somit lediglich die Kenntnis der Weltsprache. Übersetzungen der GUI in andere, gängige Sprachen ist für die Zukunft denkbar. Ebenso der weitere Einsatz von Kombinationen aus Beschriftung und Icons, wie es etwa bei der Schaltfläche zum Hinzufügen eines neuen Modells in Abbildung 4.7 zu sehen ist.

4.2.4. Modularität

Die Modularität des Editors durch die bereits beschriebene Zerlegung in logische Bereiche, wie etwa Informations- und Arbeitsbereiche bietet einen Wiedererkennungswert für den Benutzer und unterstützt diesen bei der Handhabung des Werkzeugs. Eine Navigationsleiste gestattet dem Anwender den Wechsel auf eine gewünschte Modellkomponente. Das Anbieten von Katalogen mit bereits existierenden Teilmodellen ermöglicht zudem die einfache Wiederverwendung existierender Module. Dies spart Arbeit und sorgt für Einheitlichkeit im Gesamtmodell.

4.2.5. Delegierbarkeit

Die Modellierung von Objekten, welche gesonderte Zugangsvoraussetzungen oder Hoheitswissen erfordern, aber auch Arbeiten am Modell, welche den verfügbaren Zeitrahmen eines Nutzers aktuell übersteigen, können durch das Setzen einer To-Do-Markierung in den Metadaten des entsprechenden Teilmodells delegiert werden. Die Darstellungen delegierter Teilmodelle werden entsprechend ihrer Markierungen mit einer kleinen schwarz-weißen Fahne für To-Do-Markierungen versehen, wie in Abbildung 4.7 gezeigt wird. Auf die Notwendigkeit der Bearbeitung dieser Modelle durch weitere Nutzer kann so gezielt hingewiesen werden.

4.2.6. Intelligenz

Der hierarchische Modellaufbau aus Teilmodellklassen ermöglicht es Nutzerinteraktionen im Kontext aktuell ausgewählter Teilmodellklassen auszuwerten und ein intelligentes Verhalten des Community-Editors umzusetzen. Hat ein Nutzer aktuell ein Gerät zur Bearbeitung im Community-Editor ausgewählt und klickt nun auf die Schaltfläche zur Platzierung von Objekten, so liegt die Vermutung nahe, dass die Platzierung dieses Gerätes in einem Raum gewünscht ist. Folglich wird dem Nutzer eine Abfrage zur Raumauswahl und anschließend eine Arbeitsfläche zur 3D-Positionierung des Gerätes im gewünschten Raum, ähnlich zu Abbildung 3.4, angezeigt. Handelt es sich jedoch beim aktuell ausgewählten Teilmodell um einen Raum, kann die gewünschte zugehörige Etage erfragt und eine übersichtliche 2D-Grundansicht zur Positionierung des Raumes in dieser Etage eingeblendet werden, wie in Abbildung 3.3 skizziert. Für die Positionierung von Gebäuden hingegen können Werkzeuge zur Angabe einer GPS-Koordinate und eines Nordvektors eingeblendet werden.

Bei der Bearbeitung der Metadaten eines Teilmodells unterstützen geeignete UI-Widgets die syntaktisch und semantisch korrekte Eingaben von Werten. Beispielsweise wird zur Angabe des Modelltyps ein Select-Element eingeblendet. Damit kann der Nutzer aus einer Liste aller existierenden Modelltypen das gewünschte Modell einfach auswählen. Durch Eingabefelder mit deklariertem Input-Typ können mobile Geräte nach Auswahl des entsprechenden Formularfeldes automatisch passende Tastaturlayouts einblenden. Auf diese Weise kann ein intelligentes Verhalten des Editors die Modellierungsarbeit der Nutzer unterstützen.

4.3. Das Community-Management

Wie in Kapitel 3.3.1 angedeutet, ist das Management einer Gruppe zur gemeinsamen Modellerstellung eine komplexe Herausforderung. Im Folgenden werden für ausgewählte Aspekte prototypische Umsetzungen dargelegt.

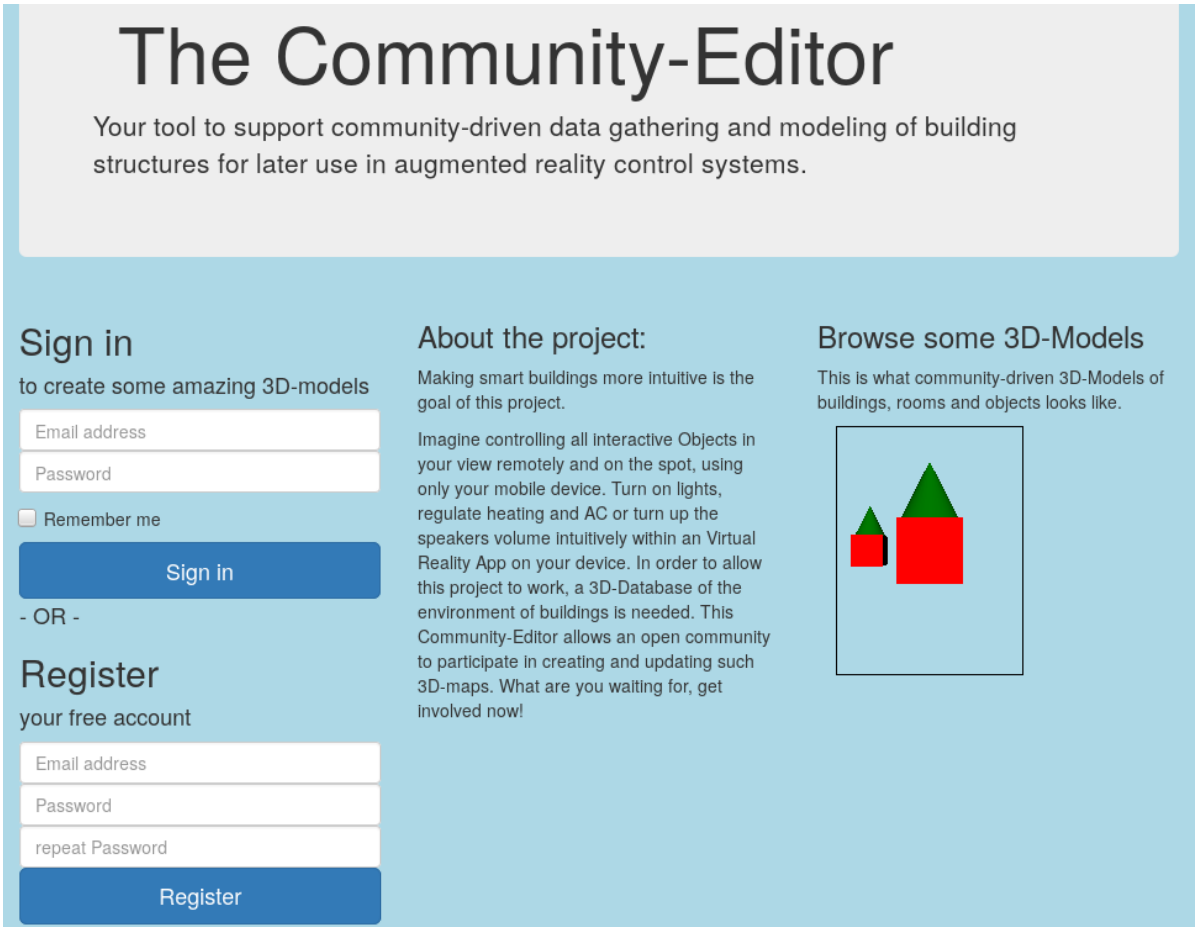


Abbildung 4.5.: Landingpage des Community-Editors

4.3.1. Nutzerverwaltung

Möglichst viele Nutzer für die Modellierung zu begeistern ist eine erste Herausforderung. Auf der in Abbildung 4.5 dargestellten Landingpage des Community-Editors wird daher das Projekt kurz und knapp vorgestellt und Beispiel-3D-Modelle aus dem Modellkatalog werden interaktiv angezeigt. Somit können neue Nutzer direkt an das Konzept herangeführt werden. Zudem besteht die Möglichkeit der Registrierung sowie des Logins, um als registrierter Nutzer am Modellierungsprozess mitzuwirken. In den Metadaten eines Teilmodells wird das Nutzerkennzeichen des letzten Bearbeiters gespeichert. Auf diese Weise kann sichergestellt werden, dass alle Aktionen am Modell eindeutig zu Autoren zugeordnet werden können.

Durch an Nutzerkennzeichen gebundene Aktionen lassen sich individuelle Statistiken erstellen. Mit deren Hilfe lässt sich die in Abbildung 3.6 vorgeschlagene, motivierende Wertschätzung und Zusammenfassung zum Ende einer Nutzersitzung realisieren. Eine

Umsetzung der im Konzept vorgeschlagenen spielerischen Elemente zur Nutzermotivation wurde im zeitlich begrenzten Rahmen der Arbeit nicht durchgeführt.

4.3.2. Parallele Bearbeitung

Wenn viele Nutzer gemeinsam ein Modell erstellen, ist es wesentlich für eine produktive Modellentwicklung, dass parallele Arbeiten am Modell stattfinden können ohne sich gegenseitig negativ zu beeinflussen. Um dies umzusetzen kommt erneut die Zerteilung des Modells in Teilmodell und Modellkomponenten zum Tragen. Für jede 3D-Beschreibung und jede objektspezifische Metadatenbeschreibung existiert eine Datei auf dem Modellserver, die sich zu einem Zeitpunkt von maximal einem Nutzer bearbeiten lässt. Hierfür kann je nach Servertechnologie ein passendes Datei-Management-Tool, wie etwa lockfile[1] aus dem Node Package Manager (npm) für Node.js-Server, verwendet werden.

Editiert ein Nutzer eine Modellkomponente, so wird die Datei für eine Bearbeitung durch weitere Nutzer eine gewisse Zeit lang gesperrt. Diese wird dem aktiven Nutzer in der GUI angezeigt, wie in Abbildung 4.3 am oberen Rand zu sehen. Andere Nutzer sehen für solche Modellkomponenten ein Schloss-Icon in der Katalogauswahl, wie in Abbildung 4.2 unten links für das Gerät D_001 dargestellt.

Unterschiedliche Modellkomponenten können von verschiedenen Nutzern daher parallel bearbeitet werden, während die Bearbeitung identischer Modellkomponenten nur zeitlich versetzt ermöglicht wird. Davon nicht betroffen sind die Foto-DB-Modellkomponenten. Hier können auch mehrere Nutzer gleichzeitig Fotos zu einem Objekt ergänzen.

4.3.3. Versionierung der Modellkomponenten

Um gegen Vandalismus und Beschädigungen am Modell vorgehen zu können, werden für einzelne Modellkomponenten unterschiedliche Versionen gespeichert. Beschädigungen können somit durch die Auswahl älterer Versionen einfach und zügig rückgängig gemacht werden. In einer einfachen Implementierung wird bei jedem Speichern einer Modellkomponente auf dem Modellserver eine neue Version der Datei angelegt. Der Zeitstempel des letzten Schreibvorgangs der Datei auf dem Modellserver wird genutzt, um die Chronologie der Versionen nachvollziehen zu können. Abbildung 4.1 zeigt am Beispiel des 3D-Modells einer Lampe D001 die Umsetzung.

Zusätzlich wird eine symbolische Verknüpfung mit dem entsprechenden Dateinamen für jede Modellkomponente angelegt, welche stets auf die jeweils neueste Version dieser verweist. Angelegt werden kann solch eine Verknüpfung für die Beispiellampe aus Abbildung 4.1 auf Unixsystemen wie folgt:

[1] `https://github.com/npm/lockfile`

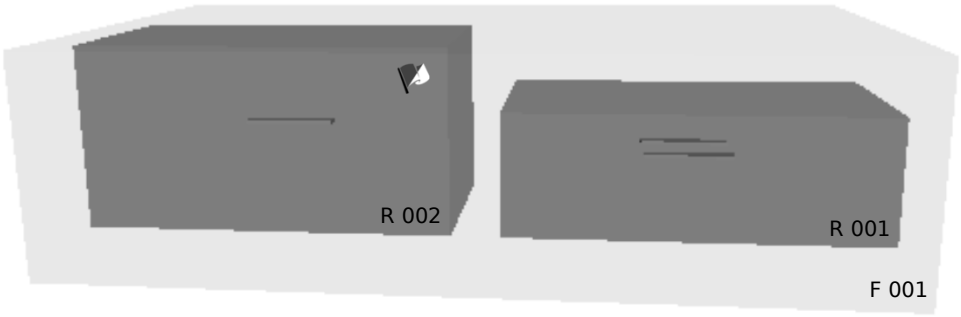


Abbildung 4.6.: Markierung eines inkompatiblen Raummodells mit zu großer Deckenhöhe durch eine schwarz-weiße To-Do-Fahne

ln −s 2016−04−01\ 11:11:42−Lampe.x3d Lampe.x3d

Auf diesem Weg kann eine dauerhafte Verknüpfung auf die jeweils aktuellste Version einer Komponente umgesetzt werden. In der GUI des Editors kann beispielsweise für die Metadaten einer konkreten Lampe der Modelltyp-Eintrag Lampe.x3d gewählt werden, statt die Versionshistorie nach der aktuellsten Datei zu durchsuchen. Wird später eine überarbeitete Version des 3D-Modells der Lampe erstellt, sorgt die symbolische Verknüpfung dafür, dass ein neues 3D-Modell auch für das konkrete Lampenobjekt übernommen wird, ohne individuell Nachgetragen werden zu müssen.

Eine Übersicht aller existierender Modellkomponentenversionen zu einem ausgewählten Teilmodell kann über eine entsprechende Schaltfläche im Infobereich angezeigt werden.

4.3.4. Behandlung von Inkompatibilitäten und Edit Wars durch Aufmerksamkeitsmanagement

Unterschiedliche Situationen der Modellkonstruktion benötigen unterschiedlich viel Aufmerksamkeit durch Nutzer. Analog zur Delegierbarkeit von Modellierungsaufgaben (vgl. Abschnitt 4.2.5) können inkompatible Modellaspekte, oder Edit Wars markiert werden, um die Aufmerksamkeit weiterer Nutzer auf sich zu lenken.

Offene Modellierungsarbeiten, Probleme und Inkompatibilitäten im Modell, beispielsweise ein Raum mit unpassend hoch modellierter Deckenhöhe, können durch das Setzen des To-Do-Tags in den Metadaten hervorgehoben werden, wie in Abbildung 4.6 dargestellt. Tritt der Fall auf, dass mehrere Nutzer ein Teilmodell mehrfach Editieren und sich dabei wiederholt gegenseitig überschreiben, so kann ein beliebiger Nutzer durch das setzten des Edit War-Tags in den Metadaten des betroffenen Teilmodells auf diesen Umstand aufmerksam machen. Die Angabe am Edit War beteiligter Nutzerkennzeichen

erfolgt über ein Eingabefeld der GUI und ermöglicht es die Schreibrechte der betroffenen Nutzer für das Teilmodell vorübergehend zu entziehen. Eine gelb-rot gestreifte Fahne markiert von Edit War betroffene Teilmodelle, wie in Abbildung 4.7 am Beispiel des Gebäudes B_002 zu sehen ist. Ein schlichtender Nutzer kann eine entsprechende Version des Teilmodells wählen und die Markierung auf Edit War-Cooldown setzen. Dabei wird ein Timer gestartet, welcher eine Wartezeit t nach unten zählt, bevor die gesperrten Nutzerkennzeichen Schreibrechte für das Modell erhalten und aus der Liste gesperrter Nutzer gestrichen werden. Für die Dauer des Cooldowns wird keine Fahne mehr im Modell angezeigt, jedoch sehen gesperrte Nutzer das Teilmodell mit entsprechendem Schloss-Icon.

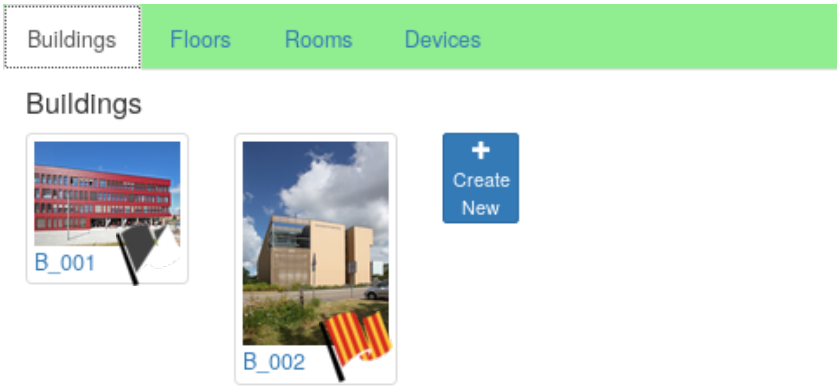


Abbildung 4.7.: Hervorhebung eines Teilmodells mit offenen To-Dos (schwarze-weiße Fahne) sowie eines von einem Edit War (rot-gelb gestreifte Fahne) betroffenen Teilmodells

Noch nicht umgesetzt wurde mangels Zeit eine Übersichtsseite zur Auflistung aller mit solchen Aufmerksamkeits-Tags markierter Teilmodelle im Editor. Markierte Teilmodelle lassen sich jedoch durch die Datenbankhaltung der Metadaten-Dateien mittels geeigneter Abfragen herausfiltern. Eine zukünftige Erweiterung um die Möglichkeit der Angabe einer Priorität und einer Beschreibung zu Markierungen könnten helfen eine solche Liste nach Bedarf zu ordnen, um so die Aufmerksamkeit engagierter Nutzer zielgerichtet auf bestimmte Teilmodelle zu lenken.

4.4. Die generische Weiterverarbeitung der Modelldaten

Ein mittels des Community-Editors erstelltes Gebäudemodell enthält 3D-Daten, Metadaten und Bildinformationen zu Gebäude-, Etagen-, Raum- und Geräteteilmodellen. Werden diese Informationen auf einem Modellserver zugänglich gespeichert, gestatten sie eine maschinelle Weiterverarbeitung im Rahmen einer AR-Anwendung zur intuitiven Gebäudesteuerung. Die Weiterverarbeitung der Modelldaten u.a. zur generischen Erstellung von Nutzerschnittstellen für die Bedienung steuerbarer Geräte werden im Folgenden

beschrieben.

Befindet sich ein Nutzer der AR-Anwendung in einem modellierten Gebäude, so kann sein mobiles Gerät unter Angabe der eigenen Position eine Anfrage relevanter Modelldaten an den Modellserver stellen. Dieser antwortet mit der Übertragung der 3D-Modelle, Metadaten und Bildklassifikatoren von Teilmodellen in unmittelbarem Umfeld des Nutzers. Die auf dem mobilen Gerät laufende AR-Anwendung kann unter Ausnutzung der Modell- und Fotoinformationen steuerbare Geräte im Sichtfeld der Kamera des mobilen Gerätes tracken und diese auf einem Display hervorgehoben rendern. Wählt der Nutzer ein solches Gerät aus, so wird aus dem zugehörigen Metadaten-File des korespondierenden Teilmodells der Funktionstyp des Gerätes ermittelt. Dieser stellt eine Liste von Befehlstypen dar. Jeder Befehlstyp entspricht einer Funktion zur Steuerung des Gerätes und beschreibt die hierfür benötigten Funktionsparameter. Das Prinzip lässt sich gut an einem einfachen Beispiel verdeutlichen.

Sei toggleLamp(boolean active) die Signatur der Funktion zur Steuerung einer einfachen Deckenlampe, die an- und ausgeschaltet werden kann. Diese Funktionalität entspricht dem Befehlstypen C1 aus Tabelle 3.4. Für jeden Funktionsparameter des Befehlstypen wird nun anhand der Zuordnungen aus Tabelle 3.9 ein UI-Widget zur Repräsentation gewählt. Im Beispiel der Lampe mit $n = 2$ möglichen Zuständen des booleschen Funktionsparameters active ein Toggle Button.
Das gewählte UI-Widget wird als Nutzerschnittstelle in die GUI der AR-Anwendung gerendert, wie in Abbildung 3.13 zu sehen ist. Funktionalitäten, welche die Belegung mehrerer Parameter erfordern, werden aus einer Kombination von UI-Widgets zusammengesetzt, wie ein weiteres Beispiel zur Steuerung einer Lampe mit Farbwahl in Abbildung 3.11 verdeutlicht.

Auf diese Weise wird aus den Modelldaten eine generische Nutzerschnittstelle erzeugt, welche als Front-End der AR-Anwendung die Eingaben des Nutzers zur Gerätesteuerung entgegennimmt.

Zur Verarbeitung der eingegebenen Steuerwerte sendet die AR-Anwendung diese an einen für das konkrete Teilmodell zuständigen Steuerserver. Die Adressierung erfolgt zunächst unabhängig vom konkreten Gebäudeautomationsprotokoll des betreffenden Gerätes entsprechend der hierarischen Positionierung der Geräte im Modell. Basierend auf den Uniform Resource Locators (URL) kann die Beispiellampe D001 in Raum R308 auf Flur 003 des Gebäudes B001 aktiviert werden, indem dem Steuerserver für die Lampe die Parameterbelegung „activate=true“ wie folgt mitgeteilt wird:

```
http://steuerserver.de/B001/F003/R308/D001?active=true\
```

Der Steuerserver erfragt anschließend vom Modellserver für das entsprechende Teilmodell die in dessen Metadaten hinterlegten Informationen zur Ansteuerung des realen Gerätes. Nach Erhalt der Informationen über das vom Gerät verwendete Netzwerkprotokoll sowie dessen Adresse im zugehörigen Netzwerk, übernimmt der Steuerserver seine Aufgabe als Gateway und übersetzt den empfangenen Steuerbefehl in das Zielprotokoll. Angebunden an das entsprechende Übertragungsmedium des Zielprotokolls, beispielsweise einen KNX-Bus, kann der Steuerserver nun den tatsächlichen Steuerbefehl an das reale Gerät absenden und die Lampe leuchtet auf.

5. Validierung der entwickelten Konzepte

Der im Rahmen dieser Arbeit vorgestellte Community-Editor wurde mit dem Ziel entwickelt, möglichst intuitiv nutzbar zu sein und den Prozess Community-getriebener Modellierung von Gebäuden bestmöglich zu unterstützen. Er soll Nutzern helfen die dabei auftretende Probleme wie etwa Vandalismus oder Edit Wars zu lösen und die Erstellung eines maschinell weiterverarbeitbaren Gebäudemodells ermöglichen. Insbesondere sollen aus Gerätebeschreibungen generische Nutzerschnittstellen erzeugt werden, welche eine Interaktion mit diesen in AR-Anwendungen ermöglichen.

Es gilt die Frage zu beantworten, ob sich durch einen verteilten Community-Prozess mit Hilfe des Community-Editors Modelle von hinreichend hoher Güte erstellen lassen, um es produktiv in AR-Anwendungen zur intuitiven Gerätesteuerung verwenden zu können. Im folgenden werden Nutzertests konzipiert und präsentiert mit deren Hilfe sich untersuchen lässt, inwiefern sich die vorgestellten Konzepte des Community-Editors in der Praxis bewähren.

5.1. Testmethodik

Um den Einsatz des Editors im Feld evaluieren zu können, bieten sich Nutzertests an. Hierfür werden einer Gruppe von Testnutzern identische Aufgaben (siehe Kapitel 5.2) gestellt, die sie unter Verwendung des Community-Editors lösen sollen. Jede Aufgabe behandelt einen unterschiedlichen Aspekt des Modellierungsprozesses. Zum Teil werden die zur Modellierung benötigten Informationen den Nutzern zur Verfügung gestellt. Für andere Aufgaben werden solche Informationen bewusst vorenthalten oder in älteren Versionen der Teilmodelle versteckt, um Situationen abzubilden, in denen Hoheitswissen zur Modellierung nötig ist oder Vandalismus aufgetreten ist. Die Testkandidaten werden gebeten ihre Gedanken bei der Benutzung auszusprechen und werden bei der Durchführung der Aufgaben von einem anwesenden, stummen Protokollant beobachtet. Die Resultate der verschiedenen Nutzer werden anschließend an Hand eines zuvor festgelegten Fragenkatalogs (siehe Kapitel 5.3) verglichen und ausgewertet. Abschließend wird die Eignung des im Test erstellten Modells in der AR-Anwendung auf die Probe gestellt. Eine Durchführung und Auswertung umfangreicher Nutzertests konnten im zeitlich beschränkten Rahmen dieser Masterarbeit nicht durchgeführt werden. Jedoch

erleichtert die vorhandene Beschreibung der Testmethodik, die Ausarbeitung von Testszenarien und des Fragenkataloges die Durchführung der Tests in Folgearbeiten. Auf diese Art kann ermittelt werden, welche wesentlichen Funktionalitäten und Ziele bei der Konzipierung des Community-Editors erreicht werden konnten, welche Schwachstellen sich zeigen und wo Verbesserungen am Konzept nötig bzw. sinnvoll sind. Hieraus können relevante Aspekte für die weitere Entwicklung des Community-Editors zur verteilten Gebäudemodellerstellung ermittelt werden.

5.2. Testszenarien

Die im folgenden beschriebenen Testaufgaben stellen keinen Anspruch auf Vollständigkeit und lassen sich um weitere Aufgaben ergänzen. Die getroffene Auswahl orientiert sich daran wesentliche Aspekte der Modellerstellung und -bearbeitung mittels des Community-Editors abzubilden. Dazu zählen das Erstellen von maßstäblichen 3D-Komponenten, die Bearbeitung von Metadaten, das Hinzufügen von Bildinformationen zu Teilmodellen, die Delegierung von Modellierungsaufgaben, die Verwendung der Versionshistorie, die Positionierung von Teilmodellen sowie die paralleler Modellierung.

Auswahl der Testaufgaben:

1) Das Erstellen eines neuen Raummodells mit vorgegebenen Bemaßungen.

2) Die Ergänzung der Metadaten für ein existierendes, steuerbares Gerät, zu welchem dem Testnutzer Informationen zur Position, dem verwendeten Netzwerkprotokoll, der zugehörigen Netzwerkadresse, dem Steuerziel und dem Funktionsumfang vorliegen.

3) Das Hinzufügen von Fotos zu den Bildinformationen des Teilmodells aus 2).

4) Die Vervollständigung relevanter Metadaten für ein weiteres, existierendes Gerät, für welches dem Testnutzer nur ein Bruchteil der nötigen Informationen unmittelbaren vorliegen. Alle weiteren Informationen liegen ohne Hinweis in Form einer älteren Version der Metadaten dieses Teilmodells vor.

5) Die Platzierung des Gerätes aus 2) in einem bestimmten Raum aus 1) entsprechend einer realen Vorlage.

6) Die Suche nach einem von einem Edit War betroffenen Teilmodell und die Markierung eines solchen.

7) Das Zurücksetzten eines vandalisierten Teilmodells auf eine Version vor der Beschädigung.

8) Das Editieren eines in Bearbeitung befindlichen Teilmodells.

5.3. Fragenkatalog zur Bewertung der Resultate

Die Auswertung der durch die verschiedenen Testnutzer erzielten Resultate beim Durchführen der beschriebenen Testaufgaben soll objektiv möglich sein. Im Idealfall sind die durch verschiedene Nutzer erstellten Teilmodelle identisch zueinander. Um jedoch auch für Abweichungen der einzelnen Resultate eine Vergleichbarkeit zu schaffen, wurde folgender Fragenkatalog zur Bewertung der Resultate ausgearbeitet.

1.1) Konnte ein neues Raummodell erstellt werden?

1.2) Wurden die vorgegebenen Bemaßungen der Vorlage im Modell eingehalten?

2.1) Wurden alle vorhandenen Informationen in die Metadaten des Teilmodells übernommen?

2.2) Wurde ein vorhandenes 3D-Modell des Gerätes als zugehöriger Modelltyp ausgewählt?

2.3) Wurde ein Anwendungstyp zum Gerät ausgewählt?

3.1) Wurde ein Foto hochgeladen?

3.2) Zeigt das hochgeladene Foto das betreffende Teilmodell?

3.3) Ist das Teilmodell auf dem hochgeladenen Foto vollständig zu erkennen?

4.1) Sind die unmittelbar vorliegenden Informationen in den Metadaten des Teilmodells enthalten?

4.2) Wurde die Versionshistorie des Teilmodells aufgerufen?

4.3) Konnten die nicht unmittelbar vorliegenden Informationen aus der Versionshistorie des Teilmodells ermittelt werden?

4.4) Falls nicht alle Informationen ergänzt werden konnten, wurde die Möglichkeit der Delegierung genutzt und das Teilmodell mit dem To-Do-Tag markiert?

5.1) Wurde ein Gerät in einem Raum plaziert?

5.2) Wurde das richtige Gerät im richtigen Raum plaziert?

5.3) Stimmt die Position des Gerätes im Modell mit der aus der realen Vorlage überein?

6.1) Wurde ein Teilmodell mit der Edit War-Markierung versehen?

6.2) Zeigt die Versionshistorie des markierten Teilmodells mindestens drei sich gegenseitig überschreibende Änderungen des Teilmodells?

6.3) Wurden die Nutzernamen der am Edit War beteiligten Nutzer dabei angegeben?

7.1) Wurde das Teilmodell auf eine ältere Version zurückgesetzt?

7.2) Wurde die Version des Teilmodells aus der Historie gewählt, die die maximale Anzahl an Informationen enthält?

8.1) Wie oft wurde vergeblich versucht die betroffene Modellkomponente zu bearbeiten?

8.1) Konnte nach Ablauf der Sperrzeit eine Bearbeitung durchgeführt werden?

8.2) Wurde die Sperrzeit genutzt um andere Modellkomponenten des Teilmodells zu bearbeiten?

Abschließend wird für das Resultat der einzelnen Testnutzer jeweils durch einen Live-Test evauliert, ob sich das in 2. und 3. bearbeitete Gerät in dem in 1. erstellten Raum mittels der AR-Anwendung von einer Testhardware aus erkennen lässt, die Steuerschnittstellen zur Bedienung des Gerätes aus den Modelldaten generiert werden konnten und der Zustand des Gerätes sich hierüber manipulieren lässt. Gelingt dieser Abschlusstest, ist die Eignung des Community-Editors zur Modellerzeugung prototypisch nachgewiesen.

Eine Analyse der Protokolle zum Gedankengang und dem Verhalten der Testnutzer während der Bearbeitung der Testaufgaben bietet eine Möglichkeit die Benutzerfreundlichkeit des Editors zu bewerten. Diese ist ebenfalls von Bedeutung, da der Community-Editor den Anspruch stellt, ein möglichst intuitives Werkzeug zu sein. Seine Verwendung soll eine kleinstmögliche Hürde darstellen. Jedoch steht die Auswertung der Tests bezogen auf Funktionalität gegenüber einer Auswertung der Benutzerfreundlichkeit im Hintergrund. Detailliertere Auswertungen zur Benutzbarkeit können, basierend auf dem vorgeschlagenen Vorgehen, in anschließenden Arbeiten durchgeführt werden.

6. Zusammenfassung und Ausblick

6.1. Zusammenfassung

Im Rahmen dieser Arbeit wurde der Community-Editor als webbasiertes Werkzeug zur verteilten, Community-getriebenen Erstellung von Gebäudemodellen konzipiert. Er gestattet eine maßstabsgetreue Modellierung unterschiedlichster Geometrien und bietet die Möglichkeit Räume durch zweidimensionale Grundrissansichten und Geräte durch dreidimensionalen Raumansichten zu positionieren. Verschiedenste steuerbare Geräte unterschiedlicher Gebäudeautomationssysteme lassen sich in einem Modell abbilden und können über URLs protokollübergreifend, einheitlich adressiert werden. Durch eine Login-Funktionalität sowie die Versionierung einzelner Teile des Modells können sich mehrere Benutzer an der Modellierung beteiligen. Aus den mittels des Community-Editors erstellen Gerätemodellen lassen sich generisch grafische Nutzerschnittstellen zur Interaktion mit steuerbaren Geräten in der AR-Anwendung erstellen.

Auf die Frage, welche Eigenschaften ein solcher Editor vorweisen muss, damit seine Verwendung eine möglichst geringe Hürde darstellt, wurden Intuitivität, Zugänglichkeit, Verständlichkeit, Modularität, Delegierbarkeit sowie Intelligenz als Antworten erarbeitet. Intuitivität bei der Handhabung des Editors konnte durch das Zerlegen der Arbeitsfläche in einen Informationsbereich sowie getrennte Bereiche zur Bearbeitung von 3D-Modellen, Metadaten und Bildinformationen umgesetzt werden. Eine breite Zugänglichkeit zur Erstellung von Modellkomponenten lies sich durch die Entwicklung des Editors als webbbasierte Browseranwendung mit responsivem Design realisieren. Der Einsatz des Community-Editors ist somit ortsunabhängig, auf verschiedensten Geräten und ohne Installation möglich. Eine webbasierte Darstellung und Manipulation von 3D-Inhalten ist mithilfe des auf dem WebGL-Framework X3DOM basierenden, quelloffenen Component-Editor möglich. Verständlichkeit des Editors über die Grenzen des deutschen Sprachraumes hinaus konnte durch die englischsprachige GUI sowie die Verwendung sprachunabhängiger Icons realisiert werden. Es wurde ein modularer Modellaufbau entwickelt, welcher die Zerlegung komplexer Modelle in Teilmodelle bestehend aus Modellkomponenten gestattet, die sich relativ zueinander positionieren lassen. Durch Auswahlkataloge konnte somit eine einfache Wiederverwendbarkeit bereits erstellter Module erreicht werden. Die Einführung von To-Do-Markierungen für Modellkomponenten ermöglicht es ausstehende Modellierungsaufgaben an andere Nutzer zu delegieren. Ein intelligentes Verhalten des Editors konnte durch die Einführung von Teilmodellklassen

für Gebäude, Etagen, Räume und Geräte realisiert werden, in deren Abhängigkeiten sich die Darstellung und Funktionalität des Editors anpasst.

Die Bearbeitung der Fragestellungen, welche möglichen Probleme bei einer dezentralen, Community-getriebenen Modellierung komplexer Szenen im Allgemeinen und im Konkreten auftreten und wie sich diese bewältigen lassen, stellt einen weiteren Schwerpunkt der Arbeit dar. Untersuchungen bereits existierender Community-getriebener Anwendungen ergaben, dass Vandalismus, Edit Wars, Hoheitswissen, Anonymität der Akteure, die Notwendigkeit zur Motivation der Community sowie die effiziente Realisierung einer parallelen Bearbeitung des Gebäudemodells zu den wesentlichen Herausforderungen zählen. Es wurden Lösungsansätze zu diesen Probleme entwickelt, die helfen den Prozess der Community-getriebenen Modellierung zu managen. So gestattet der Einsatz eines Versionierungskonzept für Modellkomponenten die einfache Beseitigung von Schäden durch Vandalismus. Mithilfe spezieller Edit War-Tags können unbeteiligte Nutzer zur Schlichtung in Edit Wars beitragen und beteiligte Akteure für eine Edit War-Cooldown-Zeit dazu zwingen ihre Aufmerksamkeit auf andere Modellaspekte zu verlagern. Die Möglichkeit zur Delegierung von Modellierungsaufgaben durch To-Do-Tags lässt es zu, dass nicht frei zugängliche Informationen durch weitere Nutzer im Modell nachgetragen werden, sofern mindestens ein Mitglied der Community Zugang zu dem benötigten Wissen hat. Eine vollständig anonyme Bearbeitungen des Modells wurde durch die Einführung einer Login-Funktion realisiert. Auf diese Weise lassen sich alle Editierungen an Gebäudemodellen mit einem verantwortlichen Nutzerkennzeichen in Verbindung bringen. Die Auswertung von Nutzerstatistiken zur Wertschätzung der Beiträge einzelner Community-Mitglieder wurde als Konzept präsentiert, die Motivation der aktiven Nutzer aufrecht zu erhalten. Eine parallele Bearbeitung der Gebäudemodelle durch mehrere Nutzer konnte durch die Modularisierung des Modells anteilig umgesetzt werden. Verschiedene Modellkomponenten können durch unterschiedliche Akteure gleichzeitig bearbeitet werden. Durch die Entwicklung eines zeitlich begrenzten Reservierungskonzeptes für konkrete Modellkomponenten konnte sichergestellt werden, dass die Arbeit eines Nutzers an der betroffenen Komponente nicht durch weitere Nutzer gestört werden kann.

Als dritten Gegenstand der Arbeit wurden die Fragen untersucht, welche Eigenschaften ein Modell vorweisen muss, um daraus automatisch generierte Nutzerschnittstellen für Geräte erzeugen zu können und wie eine solche Generierung aus den Modelldaten erfolgen kann. Es konnte herausgearbeitet werden, dass eine endliche Anzahl an möglichen Steueraufgaben existiert, welche sich Befehlstypen zuordnen lassen. Durch die Gruppierung eindeutiger Befehlstypen als Funktionstypen wurde eine Möglichkeit geschaffen, für identische Steueraufgaben einheitlich Nutzerschnittstellen zu erzeugen. Hierfür müssen in den Metadaten der Teilmodelle die Funktionsparameter steuerbarer Geräte sowie deren Wertebereiche enthalten sein. Es wurde eine Methode präsentiert, welche die Abbildung der Steueraufgabe auf UI-Widgets in Abhängigkeit des Funktionsparameters sowie dessen Wertebereichs ermöglicht und ein Vorschlag für eine mögliche Zuordnung

von Befehlstypen auf UI-Widgets präsentiert. Komplexe Steueraufgaben mit mehreren Funktionsparametern konnten durch das Zusammensetzten einzelner UI-Widgets erzielt werden. Es wurde darüber hinaus aufgezeigt, dass keine eindeutige Abbildung von Steueraufgaben auf Nutzerschnittstellen durchgeführt werden kann, sondern eine Vielzahl unterschiedlicher Nutzerschnittstellen denkbar ist, deren Interpretation und Intuitivität kontextabhängig betrachtet werden muss.

6.2. Ausblick

Erste Erfahrungen mit dem Community-Editor zeigen verwertbare Resultate. So konnten beispielsweise einfache, verschiebbare 3D-Modelle erstellt, positioniert und anhand von Bildmaterial mit Textur versehen werden. Basierend auf der fiktiven Funktionsbeschreibung einer Lampe ließen sich unterschiedliche Nutzerschnittstellen zur Interaktion mit dieser erzeugen. Praktische Tests der erzielten Resultate in der AR-Anwendung waren aufgrund des mangelnden Versuchsaufbaus der AR-Anwendung und im zeitlich begrenzten Rahmen dieser Arbeit nicht möglich. Diese müssen zur abschließenden Bewertung der Tauglichkeit der präsentierten Konzepte zukünftig durchgeführt werden.

Eine Verbesserung des Community-Editors, z.B. durch das Erstellen eines unterhaltsamen Tutorials, das Einbinden von Gamification-Elementen zur Motivation der Nutzer oder die Übersetzung der GUI in weitere Sprachen ist zukünftige denkbar. Die Umsetzung solcher Funktionalitäten war jedoch kein für diese Arbeit priorisierter Aspekt. Nutzern bei Änderungen am Modell die Möglichkeit zu geben, erklärende Notizen zu verfassen, kann den Community-getriebenen Prozess der Modellerstellung weiter verbessern. Gleiches gilt für die im Konzept bereits erwähnte, aber noch nicht implementierte, priorisierbare Liste von Modellkomponenten, welche die Aufmerksamkeit weiterer Nutzer erfordern. Gegenwärtig erfolgt zudem nur bei Teilbereichen der Metadaten von Modellkomponenten eine Plausibilitätsprüfung der Nutzereingaben. So könnte die Erarbeitung geeigneter Prüfverfahren die Qualität der erstellbaren Modelle steigern, beispielsweise in dem die Angabe zyklischer Verweise von Steuerzielen bei Teilmodellen ausgeschlossen wird.

Beim Bearbeiten der wissenschaftlichen Fragestellungen ergaben sich weitere relevante Fragen, welche zukünftig erforscht werden können. So kann etwa untersucht werden, wie sich die Steuerung komplexer Szenen, d.h. mehrerer Geräte gleichzeitig, im Modell abbilden lassen und ob sich das präsentierte Konzept zur generischen Nutzerschnittstellenerzeugung hierfür erweitern lässt. Die Weiterentwicklung des Konzeptes bezüglich der genannten mehrdeutigen UI-Erstellung ist ebenfalls denkbar. Es kann die Frage untersucht werden, wie zwischen mehreren möglichen Nutzerschnittstellen eine für einen konkreten Einsatzfall möglichst passende Auswahl zur Generierung getroffen werden kann. Für den in der Einleitung vorgestellten Anwendungsfall wäre beispielsweise eine

Anlehnung der zu verwendenden UI-Widgets an tatsächlich im Gebäude vorhandene physikalischer Interaktionsschnittstellen, wie beispielsweise Lichtschalter, denkbar. Gebäudeweit einheitliche Abbildungen zur generischen Schnittstellenerzeugung sind ein möglicher Ansatzpunkt.

Die Konzipierung des Community-Editors wurde darauf ausgelegt, möglichst vielen Nutzern einen leichten Zugang zur Modellierung zu ermöglichen. Es stellt sich daher die Frage, ob der dargelegte Nutzen einer Login-Funktionalität den Nachteil der erhöhten Einstiegshürde für neue Interessenten aufwiegt [102]. Zudem zeigen andere Community-getriebene Anwendungen eine starke Tendenz der Inhaltserstellung durch wenige, sehr aktive Mitglieder der Community [103]. Ob eine Spezialisierung des Editors für solche Poweruser den Erzeugungsprozess für Gebäudemodelle effizienter unterstützen kann, als eine Auslegung des Editors für möglichst viele Nutzer, ist daher ein weiterer Ansatzpunkt für zukünftige Untersuchungen.

Die Benutzung des Community-Editors wirft zudem auch sicherheitsrelevante Fragen auf, welchen in Zukunft nachgegangen werden sollte. Erstellte Modelldaten liefern öffentlich zugängliche Informationen über steuerbare Geräte, deren Steuerumfang und Adressierung. Zugriff auf Gebäudeautomationsbusse, wie beispielsweise KNX, lassen sich in öffentlich zugänglichen Gebäuden kaum verhindern. Potentielle Angreifer können diese Informationen Missbrauchen und erheblichen Schaden verursachen, indem sie die Steuerung der Geräte eines Gebäudes unter ihre Kontrolle bringen und beispielsweise die Heizungen des Gebäudes dauerhaft aktivieren.

Literaturverzeichnis

[1] Steffen Kutz. Gebäudesteuerung mittels Augmented Reality auf Smartphones. Masterarbeit, Universität Rostock Fakultät für Informatik und Elektrotechnik Institut für Informatik Lehrstuhl für Informations- und Kommunikationsdienste, 03 2013.

[2] Marcus Kröller. Gebäudesteuerung mittels Augmented Reality auf Smartphones – Feature Detection und Viewer. Masterarbeit, Universität Rostock Fakultät für Informatik und Elektrotechnik Institut für Informatik Lehrstuhl für Informations- und Kommunikationsdienste, 09 2013.

[3] FARO Europe GmbH & Co. KG. 3D-Vermessung. `http://www.faro.com/de-de/produkte/3d-vermessung`, 2011. [Accessed: 2016-04-11.

[4] H. Merz, T. Hansemann, and C. Huebner. Gebäudeautomation: Kommunikationssysteme mit EIB/KNX, LON und BACnet. Fachbuchverl. Leipzig im Carl-Hanser-Verlag, 2007.

[5] Bernd Aschendorf. Energiemanagement durch Gebäudeautomation: Grundlagen - Technologien - Anwendungen. Springer Vieweg, 2014.

[6] Vlado Altmann. Untersuchung des Einsatzes von Web Service-Technologien in Automationsnetzwerken. PhD thesis, Universität Rostock Fakultät für Informatik und Elektrotechnik, 2014.

[7] Bernd Aschendorf. Systemvergleich. In Energiemanagement durch Gebäudeautomation. Springer, 2014.

[8] Home Electronic Systems (HES) Architecture - Part 3-1: Communication layers - Application layer for network based control of HES Class 1, September 2006. International Standard ISO/IEC 14543-3-1:2006.

[9] KNX Association cvba. KNX ist jetzt der internationale Standard ISO/IEC 14543-3. Pressemitteilung.

[10] KNX Assoziation. KNX ADVANCED COURSE Home and Building Management Systems Interworking. Dokumentation, KNX Assoziation. Interworking E1209b.

[11] KNX Assoziation. KNX Kommunikation - Grundkurs. online.

[12] KNX Assoziation. Grundlagenwissen zum KNX Standard. KNX Association.

[13] Control network protocol Part 1 - Protocol stack, November 2012. International Standard ISO/IEC 14908-1:2012.

[14] Christoph Brönnimann. Technische Grundlagen zur LonWorks Technologie. Technical report, LonMark Schweiz, März 2015.

[15] Hans Kranz. BACnet Gebaeudeautomation 1.12 : Grundlagen in deutscher Sprache. cci Dialog, Karlsruhe, 2013.

[16] Extensible 3D (X3D) Part 1 - Abstract Specification, November 2013. International Standard ISO/IEC 19775-1:2013.

[17] The Virtual Reality Modeling Language Part 1 - Functional specification and UTF-8 encoding, 12 1997. ISO/IEC 14772-1:1997.

[18] X3D and HTML5. `http://www.web3d.org/wiki/index.php/X3D_and_HTML5`. Accessed: 2016-08-05.

[19] HTML5 Recommendation Additions for Integrating X3D Graphics. `http://www.web3d.org/x3d/content/html5/HTML5RecommendationAdditionsForX3D.html`. Accessed: 2016-08-05.

[20] Extensible 3D (X3D) encodings Part 1 - Extensible Markup Language (XML) encoding, July 2005. International Standard ISO/IEC 19776-1:2005.

[21] Extensible 3D (X3D) encodings Part 2 - Classic VRML encoding, December 2015. International Standard ISO/IEC 19776-2:2015.

[22] Extensible 3D (X3D) encodings Part 3 - Compressed binary encoding, October 2015. International Standard ISO/IEC 19776-3:2015.

[23] building and deploying X3D content. `http://www.web3d.org/hack-web3d-vr`. Accessed: 2016-08-05.

[24] Jan-Martin Kirves. Einführung in X3D - Ein kurzes Tutorial. Universität Göttingen, 2013. Seminarfolien SoSe 2013.

[25] Fraunhofer-Institut. The instantreality-framework. `http://www.instantreality.org/`. Accessed: 2016-08-05.

[26] Extensible 3D (X3D) Part 1: Architecture and bases - Node index. `http://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/nodeIndex.html`. Accessed: 2016-08-06.

[27] Vergleich von Dateiformaten für 3D-Modelle. `http://cedifa.de/wp-content/uploads/2014/05/07_3D-Modell-Formate.pdf`. Accessed: 2016-08-05.

[28] COLLADA - 3D Asset Exchange Schema. `https://www.khronos.org/collada/`. Accessed: 2016-08-05.

[29] WebGL Specification. `https://www.khronos.org/registry/webgl/specs/1.0.3/`, October 2014.

[30] WebGL 2 Specification. `https://www.khronos.org/registry/webgl/specs/latest/2.0/`, July 2016.

[31] OpenGL ES 2.0 for the Web. `https://www.khronos.org/webgl/`. Accessed: 2016-08-06.

[32] Michael P. Peterson. Online Maps with APIs and WebServices. Springer Science & Business Media, Berlin Heidelberg, 2012. aufl. edition, 2012.

[33] K. Matsuda and R. Lea. WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL. Always learning. Addison-Wesley, 2013.

[34] WebGL - Frameworks. `https://www.khronos.org/webgl/wiki/User_Contributions`. Accessed: 2016-08-06.

[35] Johannes Behr, Peter Eschler, Yvonne Jung, and Michael Zöllner. X3DOM: A DOM-based HTML5/X3D Integration Model. In Proceedings of the 14th International Conference on 3D Web Technology, Web3D '09, pages 127–135, New York, NY, USA, 2009. ACM.

[36] Jacek Jankowski, Sandy Ressler, Kristian Sons, Yvonne Jung, Johannes Behr, and Philipp Slusallek. Declarative Integration of Interactive 3D Graphics into the World-wide Web: Principles, Current Approaches, and Research Agenda. In Proceedings of the 18th International Conference on 3D Web Technology, Web3D '13, pages 39–45, New York, NY, USA, 2013. ACM.

[37] X3DOM. A framework for integrating and manipulating X3D scenes as HTML5/DOM elements. `https://github.com/x3dom/x3dom`. Accessed: 2016-08-06.

[38] Johannes Behr, Yvonne Jung, Timm Drevensek, and Andreas Aderhold. Dynamic and Interactive Aspects of X3DOM. In Proceedings of the 16th International Conference on 3D Web Technology, Web3D '11, pages 81–87, New York, NY, USA, 2011. ACM.

[39] official x3dom documentation. `http://doc.x3dom.org/tutorials/index.html`. Accessed: 2016-08-06.

[40] J. Behr, Y. Jung, J. Keil, T. Drevensek, M. Zoellner, P. Eschler, and D. Fellner. A Scalable Architecture for the HTML5/X3D Integration Model X3DOM. In Proceedings of the 15th International Conference on Web 3D Technology, Web3D '10, pages 185–194, New York, NY, USA, 2010. ACM.

[41] three.js - JavaScript 3D library. `https://github.com/mrdoob/three.js/`. Accessed: 2016-08-06.

[42] Brian Danchilla. Three.js Framework. In Beginning WebGL for HTML5, pages 173–203. Springer, 2012.

[43] Jos Dirksen. Learning Three.js: the JavaScript 3D library for WebGL. Packt Publishing Ltd, 2013.

[44] three.js examples. `http://threejs.org/examples`. Accessed: 2016-08-06.

[45] three.js documentation. `http://threejs.org/docs/index.html`. Accessed: 2016-08-06.

[46] Hongjian Li, Kwong-Sak Leung, Takanori Nakane, and Man-Hon Wong. iview: an interactive WebGL visualizer for protein-ligand complex. BMC Bioinformatics, 15(1):1–6, 2014.

[47] Wenling Hu, Hao Yuan, Jiangong Wang, and Liang Wang. The research and application of power system visualization based on HTML5. In Electric Utility Deregulation and Restructuring and Power Technologies (DRPT), 2011 4th International Conference on, pages 1562–1565. IEEE, 2011.

[48] Martin Müller John Trimpop Chris Kaping, Konrad Jacobi. Schriftliche Ausarbeitung des Softwareprojekts X3DIT zum Thema Gebäudesteuerung mittels Augmented Reality auf Smartphones. Softwareprojekt, Universität Rostock, September 2013.

[49] Thomas Paviot Max Limper. Component-Editor - A simple 3D component editor, based on X3DOM. `https://github.com/x3dom/component-editor`, Juni 2015. Accessed: 2016-06-17.

[50] Internet Users in the world. `http://www.internetlivestats.com/internet-users/`. Accessed: 2016-08-03.

[51] Axel Bruns. Towards produsage: Futures for user-led content production. 2006.

[52] Jan H Kietzmann, Kristopher Hermkens, Ian P McCarthy, and Bruno S Silvestre. Social media? Get serious! Understanding the functional building blocks of social media. Business horizons, 54(3):241–251, 2011.

[53] Tim O'reilly. What is Web 2.0: Design patterns and business models for the next generation of software. Communications & strategies, (1):17, 2007.

[54] Tim o'Reilly. What is web 2.0. Ö'Reilly Media, Inc.", 2009.

[55] Graham Cormode and Balachander Krishnamurthy. Key differences between Web 1.0 and Web 2.0. First Monday, 13(6), 2008.

[56] Tom Gruber. Collective knowledge systems: Where the Social Web meets the Semantic Web. Web Semantics: Science, Services and Agents on the World Wide Web, 6(1):4 – 13, 2008. Semantic Web and Web 2.0.

[57] David J Coleman, Yola Georgiadou, Jeff Labonte, et al. Volunteered geographic information: The nature and motivation of produsers. International Journal of Spatial Data Infrastructures Research, 4(1):332–358, 2009.

[58] Aniket Kittur, Bongwon Suh, Bryan A. Pendleton, and Ed H. Chi. He Says, She Says: Conflict and Coordination in Wikipedia. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '07, pages 453–462, New York, NY, USA, 2007. ACM.

[59] Michael F. Goodchild. Citizens as sensors: the world of volunteered geography. GeoJournal, 69(4):211–221, 2007.

[60] Taha Yasseri, Giovanni Quattrone, and Afra Mashhadi. Temporal analysis of activity patterns of editors in collaborative mapping project of OpenStreetMap. In Proceedings of the 9th International Symposium on Open Collaboration, page 13. ACM, 2013.

[61] Klaus Stein, Dominik Kremer, and Christoph Schlieder. Spatial Collaboration Networks of OpenStreetMap, pages 167–186. Springer International Publishing, Cham, 2015.

[62] M. Haklay and P. Weber. OpenStreetMap: User-Generated Street Maps. IEEE Pervasive Computing, 7(4):12–18, Oct 2008.

[63] OpenStreetMap. `https://www.openstreetmap.org/about`. Accessed: 2016-08-04.

[64] Daniel Sui, Sarah Elwood, and Michael Goodchild. Crowdsourcing geographic knowledge: volunteered geographic information (VGI) in theory and practice. Springer Science & Business Media, 2012.

[65] Ann Majchrzak, Christian Wagner, and Dave Yates. The impact of shaping on knowledge reuse for organizational improvement with Wikis. MIS Quarterly, February, 2012.

[66] Georg Glasze and Chris Perkins. Social and Political Dimensions of the OpenStreetMap Project: Towards a Critical Geographical Research Agenda, pages 143–166. Springer International Publishing, Cham, 2015.

[67] Pnina Shachaf and Noriko Hara. Beyond vandalism: Wikipedia trolls. Journal of Information Science, 36(3):357–370, 2010.

[68] Róbert Sumi, Taha Yasseri, András Rung, András Kornai, and János Kertész. Edit wars in Wikipedia. arXiv preprint arXiv:1107.3689, 2011.

[69] Róbert Sumi, Taha Yasseri, András Rung, András Kornai, and János Kertész. Characterization and prediction of Wikipedia edit wars. 2011.

[70] Carman Neustaedter, Anthony Tang, and Judge K Tejinder. The role of community and groupware in geocache creation and maintenance. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 1757–1766. ACM, 2010.

[71] Don Tapscott and Anthony D Williams. Wikinomics: How mass collaboration changes everything. Penguin, 2008.

[72] A. Mehler-Bicher, M. Reiß, and L. Steiger. Augmented Reality: Theorie und Praxis. Oldenbourg Wissenschaftsverlag, 2011.

[73] Clemens Arth, Raphael Grasset, Lukas Gruber, Tobias Langlotz, Ro Mulloni, Dieter Schmalstieg, and Daniel Wagner. The History of Mobile Augmented Reality Developments in Mobile AR over the last almost 50 years, 2015.

[74] Ronald T Azuma. A survey of augmented reality. Presence: Teleoperators and virtual environments, 6(4):355–385, 1997.

[75] Feng Zhou, Henry Been-Lirn Duh, and Mark Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, pages 193–202. IEEE Computer Society, 2008.

[76] Bernd Heisele, Gunhee Kim, and Andrew Meyer. Object Recognition with 3D Models. In BMVC, pages 1–11. Citeseer, 2009.

[77] Gerhard Reitmayr and Tom Drummond. Going out: robust model-based tracking for outdoor augmented reality. In Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality, pages 109–118. IEEE Computer Society, 2006.

[78] Tom Page. Skeuomorphism or flat design: future directions in mobile device user interface (UI) design education. International Journal of Mobile Learning and Organisation, 8(2):130–142, 2014.

[79] Christian Stein, Max Limper, and Arjan Kuijper. Spatial Data Structures for Accelerated 3D Visibility Computation to Enable Large Model Visualization on the Web. In Proceedings of the 19th International ACM Conference on 3D Web Technologies, Web3D '14, pages 53–61, New York, NY, USA, 2014. ACM.

[80] Bernhard Preim and Raimund Dachselt. Interaktive Systeme: Band 2: User Interface Engineering, 3D-Interaktion, Natural User Interfaces. Springer-Verlag, 2015.

[81] Fraunhofer IGD. X3DOM Documentation - The Virtual Catalogue. `http://doc.x3dom.org/tutorials/applicationTutorials/virtualCatalogue/index.html`. Accessed: 2016-07-20.

[82] Jane Hunter, Abdulmonem Alabri, and Catharine van Ingen. Assessing the quality and trustworthiness of citizen science data. Concurrency and Computation: Practice and Experience, 25(4):454–466, 2013.

[83] Michael F. Goodchild and Linna Li. Assuring the quality of volunteered geographic information. Spatial Statistics, 1:110 – 120, 2012.

[84] Harald Holone, Gunnar Misund, and Hakon Holmstedt. Users are doing it for themselves: Pedestrian navigation with user generated content. In The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST 2007), pages 91–99. IEEE, 2007.

[85] Olga Yanenko and Christoph Schlieder. Game principles for enhancing the quality of user-generated data collections. In Proceedings of the 17th Annual Association of Geographic Information Laboratories for Europe (AGILE) Conference on Geographic Information Science, 2014.

[86] Nama R Budhathoki and Caroline Haythornthwaite. Motivation for open collaboration crowd and community models and the case of OpenStreetMap. American Behavioral Scientist, 57(5):548–575, 2013.

[87] Philip Langer and Manuel Wimmer. A benchmark for conflict detection components of model versioning systems. In Proceedings of the International Workshop on Comparison and Versioning of Software Models (CVSM 13) at SE 13.

[88] Anja Ebersbach, Markus Glaser, Richard Heigl, and Alexander Warta. Wiki: web collaboration. Springer Science & Business Media, 2008.

[89] Tristan Harris. How better tech could protect us from distraction. TED talk, Jun 2016.

[90] Dane Bertram, Amy Voida, Saul Greenberg, and Robert Walker. Communication, Collaboration, and Bugs: The Social Nature of Issue Tracking in Small, Collocated Teams. In Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work, CSCW '10, pages 291–300, New York, NY, USA, 2010. ACM.

[91] Carlos Santos, Brunelli Miranda, Tiago Araujo, Nikolas Carneiro, Anderson Marques, Marcelle Mota, Jefferson Morais, and Bianchi Meiguins. Guidelines for Graphical User Interface Design in Mobile Augmented Reality Applications. In International Conference on Virtual, Augmented and Mixed Reality, pages 71–80. Springer, 2016.

[92] Robin Berjon, Steve Faulkner, Travis Leithead, Silvia Pfeiffer, Edward O'Connor, and Erika Doyle Navara. HTML5 - A vocabulary and associated APIs for HTML and XHTML, October 2014.

[93] Android Developer Guide. `https://developer.android.com/guide/topics/ui/controls.html`. Accessed: 2016-07-28.

[94] OS X Human Interface Guidelines. `https://developer.apple.com/library/prerelease/content/documentation/UserExperience/Conceptual/OSXHIGuidelines/ControlsAll.html`. Accessed: 2016-07-28.

[95] KNX Certified Products. `https://www.knx.org/ae/knx/knx-products/knx-certified-devices/index.php`. Accessed: 2016-07-29.

[96] Z-Wave Products. `http://products.z-wavealliance.org/`. Accessed: 2016-07-29.

[97] BACnet Testing Laboratory Product Listings. `http://www.bacnetinternational.net/btl/`. Accessed: 2016-07-29.

[98] LonMark International Certified Product Catalog. `http://www.lonmark.org/product/`. Accessed: 2016-07-29.

[99] Android Developer Guide - Custom Components. `https://developer.android.com/guide/topics/ui/custom-components.html`. Accessed: 2016-07-29.

[100] jQuery UI Bootstrap. `https://jquery-ui-bootstrap.github.io/jquery-ui-bootstrap/index.html`. Accessed: 2016-07-29.

[101] Kyle Lutes. Cross-platform mobile app software development in the curriculum. Journal of Issues in Informing Science and Information Technology (IISIT), 9:115–124, 2012.

[102] Paul B Laat. Profiling vandalism in Wikipedia: A Schauerian approach to justification. Ethics and Information Technology, 18(2):131–148, 2016.

[103] Peter Mooney. Understanding the activity of contributors to VGI projects. How, why, where, and when do they contribute geographic information. In Proceedings of the 26th International Cartographic Conference (ICC), Dresden, 2013.

A. Anhang

A.1. Beispiel-X3D-Beschreibungen für Teilmodelle

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.3//EN"
  "http://www.web3d.org/specifications/x3d-3.3.dtd">
<X3D profile='Immersive' version='3.3'
     xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
     xsd:noNamespaceSchemaLocation=
      'http://www.web3d.org/specifications/x3d-3.3.xsd'>

  <head>
    <meta content='16 Mai 2016' name='created'/>
    <meta content='16 Mai 2016' name='modified'/>
    <meta content='Simeon Wiedenmann' name='creator'/>
    <!-- Alternate encodings: VRML97, X3D ClassicVRML Encoding,
         X3D Compressed Binary Encoding (CBE), X3DOM, JSON -->
    <!--
    <meta content='HelloWorld.wrl' name='reference'/>
    <meta content='HelloWorld.x3dv' name='reference'/>
    <meta content='HelloWorld.x3db' name='reference'/>
    <meta content='HelloWorld.xhtml' name='reference'/>
    <meta content='HelloWorld.json' name='reference'/>
    -->

  </head>

  <Scene>
    <WorldInfo title='Buildingmodel in x3d!'/>

    <!-- 3D-Description of Building -->
    <Group DEF='BuildingType001'>
      <Viewpoint position='-14.81127 24.56596 -2.70658'
                 orientation='-0.93717 -0.23931 0.25387 1.74091'
                 DEF='BuildingView'
                 description="Viewpoint for building"
      />

      <!-- Floors within the Building -->

      <!-- #1 Floor -->
      <Transform translation='0.2 0.2 0.2'>
```

```
        <!-- Floor floor001 of Object-type floor.x3d-->
41      <inline url="Floor.x3d" DEF='floor001'/>
      </Transform>
43
      <!-- #2 Floor  -->
45    <Transform translation='0.2 0.2 2.8'>
        <inline url="Floor.x3d" DEF='floor002'/>
47    </Transform>

49    <!-- #3 Floor  -->
      <Transform translation='0.2 0.2 5.6'>
51      <inline url="Floor.x3d" DEF='floor003'/>
      </Transform>
53
        <!-- Floor Volume -->
55    <Shape>
        <!-- room dimensions X * Y meters with a high of Z meters-->
57      <Box size='19.2 10.2 5.4'/>
        <Appearance>
59        <Material DEF='RoomVolume'
                    diffuseColor='0.5 0.5 0.5' transparency='.95'>
61        </Material>
        </Appearance>
63    </Shape>

65  </Group>

67 </Scene>
</X3D>
```

Listing A.1: X3D-Beschreibung eines aus drei Etagen bestehenden Gebäudes

```
<Scene>
  <WorldInfo title='Floor-Model in x3d!'/>

  <!-- 3D-Description of Floor  -->
  <Group DEF='FloorType001'>
    <Viewpoint position='-10.78309 17.88484 -1.97050'
               orientation='-0.93717 -0.23931 0.25387 1.74091'
               DEF='FloorView'
               description="Viewpoint for floor"
    />

    <!-- Rooms within the Floor  -->
    <!-- #1 Room  -->
    <Transform translation='0.2 0.2 0'>
      <!-- Device Device01 of Object-type LampTypeA.x3d-->
      <inline url="Room.x3d" DEF='Room01'/>
    </Transform>
    <!-- #2 Room  -->
    <Transform translation='6.4 0.2 0'>
      <inline url="Room.x3d" DEF='Room02'/>
    </Transform>
    <!-- #3 Room  -->
    <Transform translation='6.4 3.4 0'>
      <inline url="Room.x3d" DEF='Room03'/>
    </Transform>

    <!-- Floor Volume -->
    <Shape>
      <!-- room dimensions X * Y meters with a high of Z meters-->
      <Box size='18.8 9.8 2.4'/>
      <Appearance>
        <Material DEF='Floor001Volume'
                  diffuseColor='0.6 0.6 0.6' transparency='.9'>
        </Material>
      </Appearance>
    </Shape>
  </Group>
</Scene>
```

Listing A.2: Gekürzte X3D-Szenen-Beschreibung einer Etagen mit drei Räumen

```
<Scene>
  <WorldInfo title='Room-Model in x3d!'/>

  <!-- 3D-Description of Room  -->
  <Group DEF='RoomTypeR100'>
    <Viewpoint position='-0.35904 8.09576 3.01026'
               orientation='-0.99832 -0.05555 -0.01620 1.21531'
               DEF='RoomView' description="Viewpoint for room"
    />

    <!-- Devices within the Room  -->
    <!-- #1 Device  -->
    <Transform translation='0 0 0'>
      <!-- Device Device01 of Object-type LampTypeA.x3d-->
      <inline url="LampTypeA.x3d" DEF='Device01'/>
    </Transform>
    <!-- #2 Device  -->
    <Transform translation='0 1 0'>
      <inline url="LampTypeA.x3d" DEF='Device02'/>
    </Transform>

    <!-- Room Volume -->
    <Shape>
      <!-- room dimensions X * Y meters with a high of Z meters-->
      <Box size='6 3 2'/>
      <Appearance>
        <Material DEF='RoomVolume'
                  diffuseColor='0.8 0.8 0.8' transparency='.8'>
        </Material>
      </Appearance>
    </Shape>
  </Group>
</Scene>
```

Listing A.3: Gekürzte X3D-Szenen-Beschreibung Raumes mit zwei Geräten

```
<Scene>
  <WorldInfo title='Lamp-Device-Model in x3d!'/>
  <Group DEF='LampTypeA'>

    <Viewpoint position='-0.07098 1.60045 0.59510'
               orientation='-0.99832 -0.05555 -0.01620 1.21531'
               DEF='DeviceView' description="Viewpoint for device"
    />

    <!-- 3D-Description of Device LampA -->
    <Transform translation='0 0 0'>

      <Shape>
        <Box size='1.5 0.2 0.06'/>
        <Appearance>
          <Material DEF='LampColor' diffuseColor='0.8 0.8 0.8'/>
          <!--
          <ImageTexture DEF='ImageOfDeviceTypeLampA'
                        url='"textureOfLampA.png" '
          />
          -->
        </Appearance>
      </Shape>

    </Transform>
  </Group>
</Scene>
```

Listing A.4: Gekürzte X3D-Szenen-Beschreibung eines Gerätes

A.2. Beispielliste durch Gebäudeautomationssysteme vernetzbarer Geräte

- Lampen
- Heizungen
- Klimaanlagen
- Belüftung
- Beamer
- Leinwände
- Smartboards
- Rollos/Jalousie
- Fenster-Motor zur Fensterstellung (z.B. für Fenster hoch hoben)
- Lautsprecher
- Monitor / TV
- Wlan-Router/ WLAN-Access-Points
- Localisierungsbeacon (iBeacon)
- Indoor Sateliten
- Netzwerksteckdose/ Netzwerktechnik allgemein
- Telefonanlage
- Drucker, Fax
- Türschlösser/ Zutrittskontrolle
- Alarmanlage
- Steckdosen/ Elektroversorgung
- Bewegungsmelder
- Rauchmelder
- Notausgangsleuchte/Schild

- CCTV
- Toilettenspühlung/ Sanität
- Raumduft-Anlagen
- Giesanlage (Zimmerpflanzen)
- Lichtschalter
- Aufzug
- Rohrpostsystem
- Kassen
- RFID Sicherheitsantennen (wie z.B. Mediamarktausgang)
- Wasserzähler
- Stromzähler
- Raumzutrittszähler (Schranken in Museen)

Eidesstaatliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit selbständig verfasst wurde und nur die aufgeführten Quellen und Hilfsmittel genutzt wurden. Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, sind entsprechend kenntlich gemacht und die Arbeit ist in gleicher oder ähnlicher Form noch nicht Bestandteil einer Studien- oder Prüfungsleistung.

Rostock, der 22. August 2016