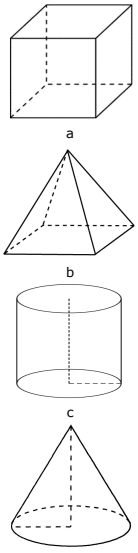


Abbildung 2: Die Beispiele der räumlichen geometrischen Figuren: Würfel (a), Pyramide (b), Zylinder (c), Kegel (d).

3D-Modellierung von Gebäuden aus räumlichen geometrischen Figuren

Valeriia Zakhlieniuk
Hochschule für Technik und Wirtschaft Berlin
s0551343@htw-berlin.de

Ilyes Soussi
Hochschule für Technik und Wirtschaft Berlin
s0548859@htw-berlin.de

Abstract
In dieser Arbeit wird ein Konzept für ein Tangible User Interface (TUI) vorgestellt. Die Hauptidee besteht in die Modellierung eines Gebäudes aus verschiedenen räumlichen geometrischen Figuren mit der Hilfe von einer Kamera. Es wird beschrieben, wie das System funktionieren sollte und die Ergebnisse der Modellierung des Konzeptes mit der Hilfe von der Bodystorming-Methode sind dargestellt. Auch werden die Kategorien und Taxonomien des TUIs beschrieben.

Author Keywords
Tengible user interfaces; Constructive Assembly System; 3D-Modellierung; Bodystorming-Methode; distant embodiment; noun metaphor.

ACM Classification Keywords
J. Computer Applications J.0 GENERAL

Einführung

‘*While for people with an HCI background, the physical aspect is often new ground, the physical has of course always formed an essential part of product design. (...) In the world of software, augmented reality is often presented as a way of introducing real, tangible objects in an otherwise virtual world. (...) Clearly, unlike software, electronic consumer products are tangible to start with. What is new is not so much the tangibility of the interaction as the richness of the interaction*‘. (Djajadiningrat, Overbeeke und Wensveen 2000, S.131)

Der Bildschirm war immer nur ein Fenster, durch das hindurch wir versuchen, in die virtuelle Welt hinein zu greifen, und den wir nicht als ein materielles Ding wahrnehmen. Virtuelle Welten und simulierte Realitäten laden uns dazu ein, ganz in sie einzutauchen, die Beschränkungen unserer physikalischen Welt aufzugeben [5]

Die Interaktion mit materiellen, greifbaren Objekten bietet verschiedenartigere Interaktionsmöglichkeiten und ist intuitiver und anschaulicher als die Interaktion mit rein visuellen Objekten. Der Benutzer kann beim Interagieren mit materiellen, greifbaren Objekten Fähigkeiten zur Manipulation von Objekten nutzen, die durch den Umgang mit Objekten der realen Umgebung erlernt wurden. Zu diesen Fähigkeiten zählt z.B. die Fähigkeit Bausteine zusammen zu setzen oder Modelle aus formbarer Masse zu erstellen [6].

Stand der Technik

Seit der Erfindung des ersten Tangible User Interfaces hat sich das Interesse an der TUI ständig und mit jedem Jahr mehr Sachanlagen zeigen sich gewachsen.

Neuere Entwicklungen gehen sogar noch einen Schritt weiter und übernehmen die dritte Dimension, indem es einem Benutzer, um Landschaften mit Lehm oder Sand zu bilden. Wieder verschiedenen Simulationen ermöglichen die Analyse der Schatten, Höhenkarten, Pisten und andere Merkmale der interaktiv formbaren Landmassen [7].

In den letzten Jahren auch viele Amateur- und semiprofessionelle Projekte, neben Wissenschaft und Wirtschaft gestartet wurden. Dank Source Tracking-Technologien und die ständig steigende Rechenleistung zur Verfügung, um Endverbraucher zu öffnen, ist die erforderliche Infrastruktur heutzutage zugänglich zu fast jeder. Ein Standard-PC, eine Webcam, und einige Handwerksarbeit zu ermöglichen, Sachanlagen mit einem minimalen Programmier- und Materialaufwand einzurichten. Dies öffnet Türen zu neuen Möglichkeiten der Wahrnehmung der Mensch-Computer-Interaktion und gibt Raum für neue Formen der Kreativität für die breite Öffentlichkeit, zu experimentieren und spielt mit.

Eine bemerkenswerter interaktive Installation ähnlich zu unserem Project ist Instant-Stadt, die Spiele, Musik, Architektur und kollaborative Aspekte vereint. Es ermöglicht dem Benutzer die dreidimensionalen Strukturen aufbauen und die Einrichtung einer Stadt mit rechteckigen Bausteine, die gleichzeitig zu den interaktiven Zusammenbau der musikalischen Fragmente von verschiedenen Komponisten [7].

Andere Beispiele für Constructive Assembly System sind die Toolkits "intelligente 3D-Modellierung" von Aish [1] und Frazer und Frazer [2] oder als neuere Beispiele, BlockJam [3] und Topobo [4].

Es ist schwierig, den Überblick zu behalten und mit Blick auf den schnell wachsenden Zahl all dieser Systeme und Werkzeuge, aber während viele von ihnen scheinen nur die verfügbaren Technologien zu nutzen und werden nur einigen anfänglichen Experimente und Tests mit ein paar grundlegende Ideen oder einfach nur reproduzieren bestehende Systeme, einige von ihnen öffnen sich in neue Schnittstellen und Wechselwirkungen und sind im öffentlichen Raum eingesetzt oder in Kunstinstallationen eingebettet [7].

Grundidee

Die Grundidee (Abb. 1) besteht darin, ein Modell eines Gebäudes aus verschiedenen räumlichen geometrischen Figuren zu erstellen. Dieses Modell wird durch die Kamera gescannt, in 3D-Darstellung umgewandelt und vom Computer dargestellt.

Dafür haben wir ein sogenanntes Feld (in der Tat: ein Kartonblatt und/oder ein Tablet), eine Kamera, die über dem Feld sich befindet, mehrere verschiedenen räumlichen geometrischen Figuren (Abb. 2) und ein Computer.

Wie soll das funktionieren? Der Benutzer nimmt eine Figur und platziert sie auf das Feld. Die Kamera scannt das Feld und überträgt das Videostrom zu dem Rechner. Die Anwendung, die wir schreiben werden, erkennt auf dem Videostrom die Position und den Art der Figur. Nachdem kann das 3D-Model vom Computer dargestellt werden.

Auf der Grund, dass es nur eine Kamera oben gibt, kann man nur eine Ebene der Figuren gleichzeitig aufbauen. Aber der Benutzer kann Ebene für Ebene ein Gebäude modellieren d.h. die Erstellung der verschiedenen Ebene eines Gebäudes ist möglich.

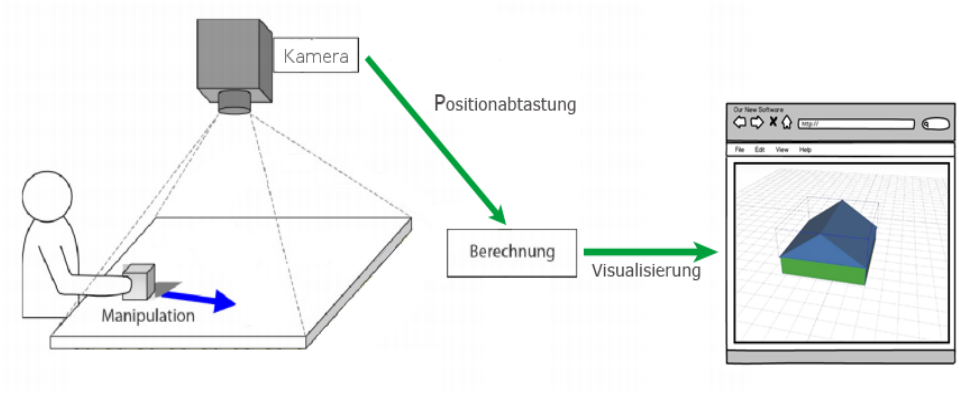


Abbildung 1: Skizze des Lösungsansatzes. Das Modell wird durch die Kamera gescannt, in 3D-Darstellung umgewandelt und vom Computer dargestellt.

Bodystorming

Das Bodystorming ist eine Methode, die den menschlichen Körper und die räumliche Umgebung für den Gestaltungsprozess nutzt, um mit ihrer Hilfe neue Herangehensweisen und Ideen für bestimmten Situationen zu erzeugen.

Wir haben die Bodystorming-Methode für unser Konzept zu dritt vorgenommen. Der Ort, an dem das neue Produkt gebraucht wird, kann ein beliebiger Ort mit einem Tisch sein. Wir haben das Szenario in einem Raum des Campus Wilchelminenhof durchgeführt. Die Person, die wichtig für das Verständnis des Produkts ist, wurde als „der Nutzer" genannt.

Die Tätigkeit, die modelliert wurde, lautet: das digitale Modell der (einfachsten) Gebäude aus verschiedenen räumlichen geometrischen Figuren aufzubauen.

Zur obengenannten Tätigkeit stellen wir die folgenden Forderungen:

- Die Komponenten als ein gesamtes System anschließen.
- Die Kamera über dem Feld platzieren.
- Das Feld ist leer.

Die Tätigkeit besteht aus den folgenden Schritten:

Schritt 1: Ein neues Project erstellen oder ein altes Projekt öffnen.

Schritt 2: Die Kamera kalibrieren.

Schritt 3: Eine Figur nehmen und auf das Feld platzieren.

Schritt 4: (optional) das Erkennungsergebnis auf dem Bildschirm beobachten.

Schritt 5: Schritt 3 und Schritt 4 wiederholen, bis alle nötigen Figuren befinden sich auf den erwünschten Stellen.

Schritt 6: Die Ebene speichern.

Schritt 7: (optional) die neue Ebene wählen und das Feld leer machen, gehen weiter zum Schritt 3.

Das beschriebene Szenario wurde zwei Mal durchgeführt: vor und nach der Entwicklung der Prototypen.

Während der ersten Durchführung haben wir als die Beobachter bemerkt, dass der Spieler konfus war: ihm war nicht komplett klar, wie die Figuren zu bewegen und zu platzieren und wie das Feld zu reinigen. Der Spieler hat erzählt, dass es ohne die physischen Figuren unklar ist, ob die Bewegung der Figuren reizen oder zu Schwierigkeiten führen kann und dass die Kamera auf jedem Fall festgelegt werden muss.

Die Entdeckungen von den Beobachtern und dem Spieler während der zweiten Durchführung können kurz zusammengefasst werden:

1. Mit dem Prototyp ist klarer für den Spieler, wie alles funktionieren soll.
2. Die Kamera sollte eindeutig festgelegt werden. Wenn der Nutzer die Kamera mit der Hand hält, benötigt sie eine ständige Kalibrierung, daneben der Hand des Benutzers müde wird.
3. Anschlüsse aller Komponenten und Führung der Anwendung sollten vermutlich keine Schwierigkeiten bereiten.
4. Die Zweifel an der Bequemlichkeit bei der Feldreinigung wurden benommen: die Kartonfiguren sind leicht und es gibt keine Schwierigkeit, sie für eine einzige Bewegung wegzumachen.
5. Während der Feldreinigung bewegt sich das Feld, falls er ein Kartonblatt ist.
6. Wenn der Benutzer die zweite oder nachfolgende Ebene bauen möchte, ist das Feld aus Kartonblatt auch unbequem, denn es ist immer nicht klar, wo man die Figuren platzieren darf. Deshalb ist die Verwendung eines Tablets anstatt des Kartons als ein Feld bevorzugt, weil mit Hilfe von einem Tablet man mögliche Plätze für die Figuren visualisieren kann.
7. Während des Tests brach eine Kartonfigur, d.h. eine sorgfältige Verwendung der Kartonfiguren ist notwendig wegen ihrer Kurzlebigkeit.

Kategorien und Taxonomien

Unser Projekt gehört zu Constructive Assembly Systeme d. h. Modulare und anschließbare Elemente sind wie das Modell der physikalischen Baukästen miteinander verbunden. Sowohl die räumliche Organisation als auch die Reihenfolge der Handlungen können vom System interpretiert werden.

Beispiele für Constructive Assembly Systeme ähnliche zu unsere System sind die Toolkits "intelligente 3D-Modellierung" oder als neuere Beispiele, BlockJam und Topobo.

Laut der Taxonomie von Fishkin [8] hat unser Project "distant embodiment" und "noun metaphor". Der Grund der Zugehörigkeit zu "distant embodiment" ist die Notwendigkeit, auf den Bildschirm zu schauen, um das Ergebnis zu beobachten. Und zu "noun metaphor"-Projekten gehört unser Projekt, weil für jedes Objekt eine und derselbe Handlung geschieht, – Aufstellung auf dem Feld, und als das Ergebnis werden die Farbe und die Lage des Objektes, d.h. seine Charakteristiken, berücksichtigt.

Laut die Relation von die Kategorisierung von Holmquist et al. [9] zu die Taxonomie von Fishkin befindet unser Projekt in die Token-Kategorie.

Wenn wir die Taxonomie von Fishkin in die Klassifikation von Underkoffler and Ishii [10] umwandeln, bekommen wir "Object as noun"-Klasse. Wichtig hier ist, dass wir nicht nur ein Attribut vom Object benutzen. Deshalb befinden wir nicht in "Object as attribute"-Klasse.

Vorstellung des Systems und Verwendete Technologien

Aus technischer Sicht haben wir unser Projekt in zwei Teilsysteme geteilt: 1) ein Teilsystem für die Erkennung und Verfolgung der räumlichen geometrischen Figuren; 2) ein Teilsystem für die 3D-Modellierung von Gebäuden.

Die Aufgabe des ersten Teilsystems besteht darin, die räumliche geometrische Figuren aus einem Videostrom, der aus einer Web-Kamera erhalten wird, zu erkennen und ihre Koordinaten auf dem Feld zu finden. Als die Kerntechnologie, die für das erste Teilsystem verwendet wurde, wurden Bibliothek OpenCV ausgewählt.

Die Open Source Computer Vision Library (OpenCV)[11] ist die am häufigsten verwendete und auch erfolgversprechendste Bildbearbeitungsbibliothek. Es ist eine sehr große und performante Programmbibliothek, die Algorithmen und Methoden nach den neuesten Forschungsergebnissen enthält, unter anderem in den Bereichen Bildverarbeitung, maschinelles Sehen und maschinelles Lernen. Als freie Software steht sie unter der BSD-Lizenz zur Verfügung.

OpenCV bietet ein plattformunabhängiges API an, das ein breites Anwendungsgebiet hat und somit Module mit Lösungen Methoden, und Algorithmen für viele Anwendungsfelder zur Verfügung stellt [12].

Visual C++ ist ein Compiler der Firma Microsoft zur Entwicklung von Software in der Programmiersprache C++ unter dem Betriebssystem Windows und anderen Betriebssystemen von Microsoft.

Aktuelle Versionen von Visual C++ verfügen über den erweiterten C++-Befehlssatz C++/CLI, der unter anderem die Nutzung der .NET-Programmierung vereinfachen soll.

Die Aufgabe des zweiten Teilsystems besteht in der Modellierung der physischen Objekte in einem digitalen Raum. Dieses Teilsystem stellt als eine web-basierte Anwendung. Als Server wurde Flask benutzt. Für die Rendering und Darstellung wurde die JS-Bibliothek three.js, die auf WebGL basiert, angewendet.

Flask ist ein in Python geschriebenes Web Application Framework. Der Fokus von Flask liegt auf Erweiterbarkeit und guter Dokumentation. Die einzigen Abhängigkeiten sind Jinja2, eine Template Engine, und Werkzeug, eine Bibliothek zum Erstellen von WSGI-Anwendungen.

WebGL (Web Graphics Library) ist eine Javascript-API zum Rendern interaktiver 3D und 2D Grafiken mittels eines kompatiblen Web-Browsers ohne Einsatz zusätzlicher Plugins. Mit WebGL steht eine API zur Verfügung, die an OpenGL angelehnt ist und deren Inhalte mittels eines <canvas> Elements dargestellt werden [15].

Die systemeigene WebGL-Grafikprogrammierung mit Shadern und den notwendigen mathematischen Funktionen (Matrizen, Quaternionen usw.) kann recht komplex sein. Um diese Komplexität zu entschärfen, wurde eine Reihe von Bibliotheken entwickelt, die die Programmierung vereinfachen, beispielweise, "Three.js".

Die Kommunikation zwischen den beiden Teilsystemen erfolgt über ein Messaging Protokoll MQTT und seine Implementierung Eclipse Paho.

MQTT (Message Queue Telemetry Transport) ist ein schlankes Messaging Protokoll, welches auf den Bereich Mobile und Internet of Things zugeschnitten ist.

Das von MQTT unterstützte Message Exchange Pattern ist das Publish/Subscribe Muster [16].

Bei Publish/Subscribe kommuniziert der Sender einer Nachricht nicht direkt mit dem Empfänger sondern wendet sich an einen Broker der die Zustellung der Nachricht übernimmt. Der Broker entkoppelt den Sender oder Producer vom Empfänger, der Consumer genannt wird. Ein Producer kann eine Nachricht selbst dann senden, wenn kein Consumer online ist. Tatsächlich weiß der Producer nicht, ob es keinen, einen oder mehrere Consumer gibt, die sich für seine Nachrichten interessieren. Der Broker ist für Consumer und Producer gleichermaßen der Server. Consumer und Producer sind Clients, die sich mit dem Broker verbinden [16].

Consumer sind oft nur an bestimmten Nachrichten interessiert, die ein Broker verteilt. Pro Thema kann ein Topic eingerichtet werden, an dem sich die Consumer anmelden (subscriben) können.

Ein Topic Filter beschreibt für jede Subscription, an welchen Topics ein Consumer interessiert ist.

Im Gegensatz zu Client/Server Protokollen wie HTTP arbeitet Publish/Subscribe ereignisorientiert. Ein Client muss nicht ständig beim Server anfragen, ob neue Daten vorliegen. Der Broker informiert die Consumer wenn neue Daten zu einem Topic vorliegen [16].

MQTT ist aber ein offener Standard mit einer großen Community. Dem Entwickler stehen zahlreiche Broker, Client Bibliotheken, Adapter und Weitere Werkzeuge wie Protokoll Analyzer zur Verfügung [16].

Das Paho-Projekt wurde Anfang 2012 unter der Schirmherrschaft der Eclipse Foundation gegründet. Ziel des Projekts ist es, quelloffene Implementierungen von Standard-Messaging-Protokollen im Machine-to-Machine-Bereich (M2M) zu liefern. Zurzeit wird die Referenzimplementierung des MQTT-Protokolls in Form von Clients für Java, C, C++, JavaScript, Lua, Python usw. bereitgestellt [16].

Technologische Umsetzung

Weiter stellen wir kurz alle wichtigen Teil der Implementierung und algorithmische Lösungen vor.

Verfolgung

Um die Objekte in Echtzeit zu verfolgen, sind zwei Schritte erforderlich. Erstens müssen wir der Art den Objekten (Würfel, Pyramide, Zylinder, Kegel) erkennen und dann kommt die Echtzeit Verfolgung.

Das Farbsehen kann mit RGB-Farbraum oder HSV-Farbraum bearbeitet werden. Der RGB-Farbraum beschreibt Farben in Bezug auf die Menge an Rot, Grün und Blau. Der HSV-Farbraum beschreibt Farben in Bezug auf den Farbton, die Sättigung und den Wert. In Situationen, in denen die Farbbeschreibung eine wesentliche Rolle spielt, wird das HSV-Farbmodell häufig gegenüber dem RGB-Modell bevorzugt.

Das HSV-Modell beschreibt Farben ähnlich wie das menschliche Auge dazu neigt, Farbe wahrzunehmen. RGB definiert die Farbe in Form einer Kombination von Primärfarben, und HSV beschreibt Farbe anhand bekannter Vergleiche wie Farbe, Lebendigkeit und Helligkeit.

Die Kamera verwendet das HSV-Modell, um die Farben zu bestimmen, weshalb vor jeder Operation der

Farbraum von BGR in HSV umgewandelt werden muss. Die HSV-Werte werden dann in dem Code verwendet, um die Position eines spezifischen Objekts zu bestimmen, für die die Anwendung sucht. Die Pixel werden einzeln geprüft, um zu bestimmen, ob sie mit einer vorbestimmten Farbschwelle übereinstimmen.

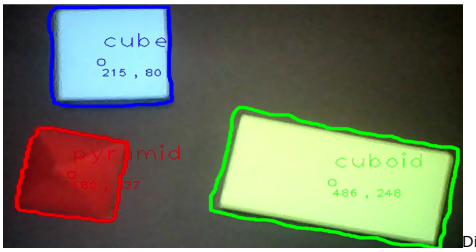
Die Operationen Erosion und Dilatation sind die Basisoperationen der morphologischen Bildverarbeitung.

Morphologische Informationen lassen sich in ähnlicher Weise, wie die Methode der Nichtlinearen Relaxation (Relaxation Labelling), verwendet. Also sie arbeiten auf bereits segmentierten Bildern. Danach können die Segmente oft nach Vordergrund und Hintergrund unterschieden werden. Auf die Vordergrundsegmente sollen die morphologischen Operationen angewendet werden. Man kann annehmen, dass alle Pixel von Vordergrundsegmenten durch eine „1" und alle anderen Pixel durch eine „0" gekennzeichnet sind. Im entstandenen Binärbild b ist die Menge aller Pixel von fs durch diejenigen Pixel von b gegeben, deren Wert 1 ist. Das Strukturelement s ist über die Menge von Orten (mk, nk) seiner K Pixel im Bildkoordinatensystem definiert. Eine morphologische Operation lässt sich als g = fs ◇ s zu formulieren. Der Operator „◇" kann eine der Mengenoperationen Schnitt, Vereinigung oder Differenz sein. fs ist die Menge der Segmentpixel (Eine Menge der strukturierenden Pixel heißt Strukturelement), auf der die Operation aus- geführt und das Ergebnis in g berechnet wird. Als Strukturelement kann man in der Bildverarbeitung oft ein Quadrat in Vierer- oder Achternachbarschaft. [13]

Die Grundoperation Erosion wird mit Hilfe einer Strukturmaske realisiert. Die Strukturmaske ist eine kleine Teilmenge des Gesamtbildes, die zum Prüfen des zu untersuchenden Bildes benutzt wird. Für jede Maske wird ein Bezugspunkt definiert, welcher das Platzieren der Maske an einer bestimmten Pixelposition erlaubt. Die eigentliche Operation besteht aus der pixelweisen Verschiebung der Strukturmaske über das Gesamtbild [13].

In der digitalen Bildverarbeitung wird die Dilatation im Allgemeinen mittels strukturierenden Elements angewandt. Mathematisch gesehen handelt es sich dabei im Falle von Binärbildern um die Bildung der Minkowski-Summe von Bild und strukturierendem Element.

Abbildung 3: : Ergebnis des ersten Teils der Anwendung (Verfolgung von Objekten)



Die Dilatation eines Bildes A mit einem strukturierenden Element X bezeichnet man als A●X Anschaulich bedeutet das im Fall der Binärbildmorphologie, dass man an jedem Bildpunkt von A das komplette Element X einfügt, den Bildpunkt quasi auf die Form des strukturierenden Elementes ausdehnt (dilatiert)[13].

Um die gefilterten Objekte zu verfolgen, müssen zuerst die Konturen des gefilterten Bildes mit der Funktion openCV findContours gefunden werden.

Konturen können einfach als eine Kurve erklärt werden, die alle kontinuierlichen Punkte (entlang der Grenze) mit derselben Farbe oder Intensität verbindet. Die Konturen sind ein nützliches Werkzeug für die Formanalyse und Objekterfassung und -erkennung.

Zweitens müssen wir die Momenten-methode verwenden, um unser gefiltertes Objekt zu finden, und ein Bildmoment ist ein bestimmter gewichteter Mittelwert (Moment) der Bildpixelintensitäten oder eine Funktion solcher Momente, die gewöhnlich gewählt werden, um eine attraktive Eigenschaft oder Interpretation zu haben. Zuletzt zeichnen wir die Objekte auf dem Bildschirm.

Server

Der folgende Quelltext stellt unsere Webanwendung dar, die auf der Startseite tui_start_page.html ausgibt:

```
from flask import render_template
from flask import Flask


app = Flask(__name__)


@app.route('/')
def main():
   return render_template("tui_start_page.html")
```

Verwendung von three.js

Für die Darstellung benötigen wir zuerst natürlich ein Canvas Element, welches die ID "ThreeJS" erhält:

```
<canvas id=" ThreeJS"></canvas>
```

Das Canvas Element repräsentiert natürlich den Bereich, innerhalb dessen wir 3D-Grafiken mit WebGL/three.js darstellen können. Das Canvas Element könnt ihr irgendwo platzieren. Dann beginnen wir das eben erstellte Canvas Element in JavaScript zu referenzieren:

```
var canvas = document.getElementById("ThreeJS");
```

Zunächst benötigen wir nur eine neue Instanz von Scene. Das Scene-Objekt ist eine Art Container, welche alle für einen 3D-Raum relevanten Objekte beinhaltet. Dazu gehören Kamera, Lichtquellen, 3D-Modelle usw.

```
var scene = new THREE.Scene();
```

Wir verwenden die perspektivische Kamera, weil sie dem menschlichen Sehen entspricht. Die Kamera hat die gleiche Funktion wie unsere Augen. Wir sehen mit ihr die Welt, also Scene. Eine Kamera benötigt im Wesentlichen eine Position und eine Blickrichtung. Anschließend positionieren wir die Kamera etwas und fügen sie der Scene hinzu.

```
var SCREEN_WIDTH = window.innerWidth, SCREEN_HEIGHT = window.innerHeight;
var VIEW_ANGLE = 45, ASPECT = SCREEN_WIDTH / SCREEN_HEIGHT, NEAR = 0.1, FAR = 20000;
camera = new THREE.PerspectiveCamera( VIEW_ANGLE, ASPECT, NEAR, FAR);
scene.add(camera);
camera.position.set(0,150,400);
camera.lookAt(scene.position);
```

Danach erstellen wir eine neue Instanz des Renderers. Der Renderer berechnet die Darstellung eines Ausschnitts der 3D-Welt, also was Kamera nun eigentlich in Scene sieht, und zeichnet sie auf ein Canvas Element.

```
if ( Detector.webgl )

        renderer = new THREE.WebGLRenderer( {antialias:true} );
else

        renderer = new THREE.CanvasRenderer();
```

```
renderer.setSize(SCREEN_WIDTH, SCREEN_HEIGHT);
container = document.getElementById( 'ThreeJS' );
container.appendChild( renderer.domElement );
```

Danach erstellen wir zwei Events: automatisch zu rendern und bei einem bestimmten Tastendruck Vollbildschirm umzuschalten.

```
THREEx.WindowResize(renderer, camera);
THREEx.FullScreen.bindKey({ charCode : 'm'.charCodeAt(0) });
```

Danach wurde eine Lichtquelle erstellt.

```
var light = new THREE.PointLight(0xffffff);
light.position.set(0,250,0);
scene.add(light);
```

Zur Scene muss die Objekte natürlich hinzugefügt werden. Alle 3D-Objekte bestehen im Wesentlichen aus Polygonen und ihrer Oberfläche/Textur. Alle Polygone eines 3D-Objektes bezeichnet man auch als Mesh. Mit Polygonen bezeichnen wir hier konkret Polygon mit drei Eckpunkten und mit Oberfläche/Textur innerhalb von three.js Materialien. Betrachten als ein Beispiel eine Kugel. Konkret erstellen wir eine Kugelgeometrie, welche 3 Parameter erhält. Das wären Breite, Höhe und Tiefe. Diesen Kugel übergeben wir einer neuen Instanz von THREE.Mesh, unserem eigentlichen 3D-Objekt. Als zweiten Konstruktorparameter erhält sie zusätzlich eine Instanz von Material. Die eigentliche Mesh Instanz könnte man dann frei im Raum bewegen, drehen, usw.

```
var sphereGeometry = new THREE.SphereGeometry( 50, 32, 16 );
var sphereMaterial = new THREE.MeshLambertMaterial( {color: 0x8888ff} );
var sphere = new THREE.Mesh(sphereGeometry, sphereMaterial);
sphere.position.set(x, y, z);
```

Auch erstellen wir zusätzlich einen Boden, einen Himmel und einen Nebel-Effekt. Ohne eine davon wird die Hintergrundfarbe der Szene durch einen Webseitenhintergrund bestimmt.
Anschließend müssen wir dem Renderer nur noch sagen, welche Scene Instanz und Kamera berechnet werden soll und schon sehen wir auf dem Canvas die Objekte.

Konstruktive Festkörpergeometrie

Konstruktive Festkörpergeometrie (engl. Constructive Solid Geometry) ist eine Technik zum Modellieren von Körpern, die in der 3D-Computergrafik genutzt wird. Constructive Solid Geometry ermöglicht einem Designer komplexe Oberflächen und Körper zu erzeugen, indem er boolesche Operatoren benutzt, um Objekte zu kombinieren. Aus der CSG hervorgegangene Körper wirken oft sehr komplex, sind aber in Wirklichkeit nichts anderes als geschickt verknüpfte Objekte.
Wir benutzen diese Technologie, um Gebäude aus physischen Primitiven zu bauen. Gewöhnlich handelt es sich dabei um boolesche Operationen auf Mengen: Vereinigung, Differenz und Schnitt, aber zurzeit ist nur die Vereinigung der Objekte möglich. Vereinigung bedeutet, dass die zwei Objekte zu einem verschmolzen werden.

Kommunikation durch Eclipse Paho

Wie schon gesagt wurde, wird Eclipse Paho für die Kommunikation verwendet, weil diese Implementierung eine API für C++ und Python hat.
Die Nachrichten, die das erste Teilsystem auf das zweiten überträgt, werden in drei Typen aufgeteilt:

1. Ein neues Objekt wird hinzufügt: ein Nachricht wird gesendet nach der Erkennung eines neuen Objekts.
2. Ein Objekt wird verschoben: ein Nachricht wird gesendet nach der Verschiebung eines Objekts (regelmäßig bei der Verfolgung).
3. Ein Objekt ist gelöst: ein Nachricht wird gesendet nachdem ein Objekt verschwunden wurde.

Eine Nachricht besteht aus folgender Daten:

1. id – Identifikator;
2. action – Typ der Nachricht;
3. type – Typ des Objektes;
4. x - Wert eines Objektcenter auf der x-Achse;
5. y - Wert eines Objektcenter auf der y-Achse.

Als ein Broker benutzen wir ein Host "iot.eclipse.org" und ein Port 1883. Das Teilsystem für die Erkennung und Verfolgung der Objekte betrachten wir als ein Producer, dann das Teilsystem für die 3D-Modellierung nennt man als ein Consumer. Als Topic verwenden wir "tui/project".
Auf Sprache C++ stellt der folgende Quellcode die Verbindung, den Versand und die Unterbrechung dar:

```
sample_mem_persistence persist;
mqtt::client client(ADDRESS, CLIENTID, &persist);
mqtt::connect_options connOpts;
connOpts.set_keep_alive_interval(45);
```

```
connOpts.set_clean_session(true);


try {

        client.connect(connOpts);

        mqtt::message_ptr pubmsg = std::make_shared(PAYLOAD1);

        pubmsg->set_qos(QOS);

        client.publish(TOPIC, pubmsg);

        client.disconnect();
}
catch (...) {

        ...
}
```

Um in einer Python-Anwendung Nachrichten zu erhalten, müssen drei Funktion definiert werden: on_connect, on_message и on_subscribe.

```
def on_connect(mosq, obj, rc):
       mqttc.subscribe(MQTT_TOPIC, 0)


def on_message(mosq, obj, msg):
       print("Topic: " + str(msg.topic))

      print("QoS: " + str(msg.qos))

      print("Payload: " + str(msg.payload))
       # Weiter Quellcode


def on_subscribe(mosq, obj, mid, granted_qos):

      print("Subscribed to Topic: " + MQTT_TOPIC + " with QoS: " + str(granted_qos))

      # Weiter Quellcode
```

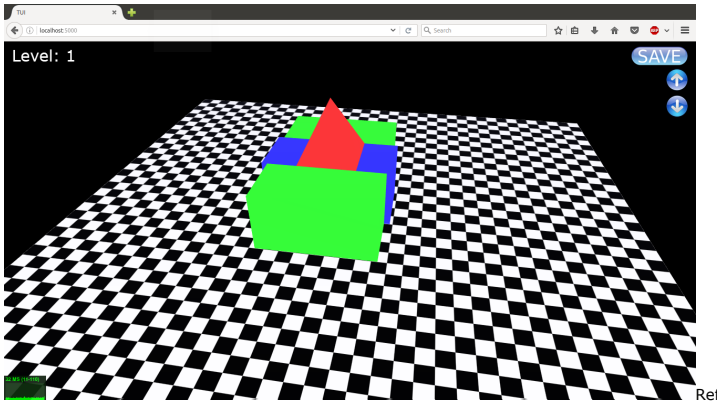
Der folgende Quellcode stellt die Verbindung:

```
mqttc = mqtt.Client()
mqttc.on_message = on_message
mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe
mqttc.connect_async(MQTT_HOST, MQTT_PORT, MQTT_KEEPALIVE_INTERVAL)
```

Beim Empfand einer Nachricht wird die Funktion on_message ausgeführt.

Fazit
In dieser Arbeit wird ein Prototyp für die Modellierung eines Gebäudes aus verschiedenen räumlichen geometrischen Figuren mit der Hilfe von einer Kamera. Der Prototyp besteht aus zwei Teilsysteme: ein Teilsystem für die Erkennung und Verfolgung der Objekten und ein Teilsystem für die 3D-Modellierung von Gebäuden. Für das erste Teilsystem wurde Bibliothek OpenCV als die Kerntechnologie ausgewählt. Für das zweite Teilsystem, das in Abbildung 4 dargestellt ist, wurden Flask als Server und JS-Bibliothek three.js benutzt. Die Datenübertragung zwischen den beiden Teilsystemen erfolgt über ein Messaging Protokoll MQTT und seine Implementierung Eclipse Paho.

Abbildung 4: Prototyp des Teilsystems für die 3D-Modellierung von Gebäuden



References

1. R. Aish. 1979. 3D input for CAAD systems. Computer Aided Design, vol. 11, no. 2. 66–70
2. J. Frazer and P. Frazer. 1982. Three-dimensional data input devices. In Proceedings of Computer Graphics in the Building Process, Washington: National Academy of Sciences
3. H. Newton-Dunn, H. Nakano, and J. Gibson. 2003. Blockjam: A tangible interface for interactive music. In Proceedings of the 2003 Conference on New Interfaces for Musical Expression (NIME-03). 170–177
4. H. S. Raffle, A. J. Parkes, and H. Ishii. 2004. Topobo: A constructive assembly system with kinetic memory. In Proceedings of the ACM CHI'04. 647– 654
5. Eva Hornecker. 2004. Tangible User Interfaces als kooperationsunterstützendes Medium. Fachbereich 3 (Mathematik & Informatik) der Universität Bremen. Retrieved November 7, 2016 from http://elib.suub.uni-bremen.de/diss/docs/E-Diss907_E.pdf
6. Marion Koelle. Tangible User Interfaces Ein kurzer Überblick über Forschungsfeld und Literatur. Retrieved November 7, 2016 from https://www.medien.ifi.lmu.de/lehre/ws1011/mmi2/mmi2_uebungsblatt1_loesung_koelle.pdf
7. Tangible User Interface. Eigenschaften von Sach Benutzeroberflächen. Beispiele. Der letzte Stand der Technik. Retrieved November 7, 2016 from http://leistentag.com/article/tangible-user-interface
8. Kenneth P. Fishkin. 2004. A taxonomy for and analysis of tangible interfaces. In: Personal and Ubiquitous Computing Volume 8 Issue 5, London, Great Brittain, September 2004. 347–358
9. Holmquist L, Redström J, Ljungstrand P. 1999. Token-based access to digital information. In: Proceedings of the 1st international symposium on handheld and ubiquitous computing (HUC'99), Karlsruhe, Germany, September 1999. 234–245
10. Underkoffler J, Ishii H. 1999. Urp: a luminous-tangible workbench for urban planning and design. In: Proceedings of the CHI'99 conference on human factors in computing systems, Pittsburgh, Pennsylvania, May 1999. 386–393

11. OpenCV. Homepage: http://opencv.org/
12. Homepage der OpenCV-Module. Retrieved February 12, 2017 from http://docs.opencv.org/modules/refman.html
13. Wolfgang Abmayr: Einführung in die digitale Bildverarbeitung, Teubner, 1994
14. Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions. Retrieved February 5, 2017 from http://flask.pocoo.org/
15. WebGL (Web Graphics Library). Retrieved February 5, 2017 from https://developer.mozilla.org/de/docs/Web/API/WebGL_API
16. MQTT, Das M2M und IoT Protokoll. Retrieved February 5, 2017 from https://www.predic8.de/mqtt.htm