



**Hochschule für Technik  
und Wirtschaft Berlin**

*University of Applied Sciences*

IndustrialVR – Erstellung einer Virtual Reality Anwendung mit  
Unity auf Basis von 3D-CAD Daten.

**Projektbericht**

**Studiengang:**  
Internationale Medieninformatik

**Betreuer:**  
Prof. Dr.-Ing. Carsten Busch / Alexander Kramer

**Autor:**  
Chris Wodäge

Independent coursework  
Sommersemester 2018



# Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Stadt, den xx.xx.xxxx

Max Mustermann

# Abkürzungsverzeichnis

**CAD** Computer-aided design oder rechnerunterstütztes Konstruieren

**STL** Standard Transformation Language oder Surface Tessellation Language

**WEA** Windenergieanlage

# Abbildungsverzeichnis

1.1	Aufbau einer WEA. . . . .	3
1.2	Aufbau einer WEA im Detail. . . . .	4
3.1	Eine Plane, bestehend aus vier quadratischen Polygonen. . . . .	9
3.2	Ergebnis des Exportes der WEA aus der CAD-Anwendung bestehend aus 494.988 Polygonen. . . . .	10
3.3	Shadingfehler am Exportmodell. . . . .	12
3.4	Automatische Reduktion der Polygone auf (v.l.n.r.) 1536, 382 und 117. . .	13
3.5	Retopologisierung einer Teilfläche durch Erzeugung eines Primitives, An- passung der Vertices, Flächenextrusion und Einsetzen einer weiteren Un- terteilung (Edgeloop). . . . .	14
3.6	Vergleich des Rotorblattes aus dem CAD-Export (1130 Polygone) mit dem neu erstellten Rotorblatt(76 Polygone). . . . .	15
3.7	Ergebnis der Retopologisierung bestehend aus 12.182 Polygonen. . . . .	16
4.1	Szene in Unity. . . . .	19

# Inhaltsverzeichnis

<b>Eigenständigkeitserklärung</b>	<b>II</b>
<b>Abkürzungsverzeichnis</b>	<b>III</b>
<b>Abbildungsverzeichnis</b>	<b>IV</b>
<b>1 Einführung</b>	<b>1</b>
1.1 Einleitung . . . . .	1
1.2 Motivation . . . . .	2
1.3 Fachlicher Kontext . . . . .	3
<b>2 Datenverarbeitung in CAD</b>	<b>6</b>
2.1 Anwendungsbereiche von CAD . . . . .	6
2.2 3D-Modellierung in CAD . . . . .	6
2.3 Exportformat in CAD . . . . .	8
<b>3 Aufbereitung der 3D-CAD Daten</b>	<b>9</b>
3.1 Eine kurze Einführung zum Aufbau von 3D-Objekten . . . . .	9
3.2 Analyse des CAD-Exports . . . . .	10
3.3 Retopologisierung des CAD-Exports . . . . .	12
<b>4 Umsetzung in Unity</b>	<b>17</b>
4.1 VRTK – Virtual Reality Toolkit . . . . .	17
4.2 Szenenaufbau in Unity . . . . .	18
4.3 Implementierte Usecases . . . . .	20
<b>5 Fazit</b>	<b>21</b>

---

<b>Literatur</b>	<b>A</b>
<b>Quelltextverzeichnis</b>	<b>C</b>
<b>Anhang</b>	<b>D</b>
A    Quellen . . . . .	D
B    Software . . . . .	D

# Kapitel 1

## Einführung

### 1.1 Einleitung

Die vorliegende Arbeit beschäftigt sich mit der Fragestellung, ob und wie ein technisch konstruiertes CAD-Modell bspw. eine Maschine, ein Auto oder wie in diesem Fall eine Windenergieanlage (WEA) in das Echtzeitsystem Unity überführt werden kann. Dazu wird untersucht welche Möglichkeiten des Exportes professionelle Konstruktionsprogramme anbieten und wie die Qualität dieser 3D-Modelle beschaffen ist. Anhand eines Beispiels werden die Zwischenschritte einer Aufbereitung beschrieben und Vergleiche mit einer Software gezogen, die diese Aufbereitung automatisch übernimmt. An diesem Beispiel soll ebenfalls das Rigging und die Animierung in Maya sowie die Erstellung eines Prototypen in Unity demonstriert werden.



## 1.2 Motivation

Ein guter Freund, ein Ingenieur aus dem Maschinenbau und leidenschaftlicher VR-Spieler trat mit einer Idee an mich heran. Er wolle eine von ihm und seiner Firma konstruierte Maschine gerne einmal in VR betrachten. Er wisse aber nicht wie eine solche Anwendung umzusetzen sei bzw. ob es überhaupt möglich sei. Ich schlug ihm vor, dass wir als Technologie Unity als Laufzeit- und Entwicklungsumgebung nutzen können. Nach einer kurzen Recherche zur Kompatibilität von CAD-Formaten und den gängigen 3D-Formaten stellten wir fest, dass ein Import von CAD-Modellen generell möglich sein muss. Da die Firma aber wie zu erwarten der Weitergabe dieser Daten nicht zustimmte beschlossen wir auf ein Modell aus seinem Studium zurückzugreifen. Die von ihm und seinen Kommilitonen in einem Projekt konstruierte WEA stellt eine voll funktionsfähige Windenergieanlage zur Stromerzeugung dar. Dieses Modell konnten wir ohne Probleme in eine Unityszene importieren. Mehrere Probleme stellten wir allerdings fest. Zum ersten wies das Modell für eine Echtzeitanwendung eine extrem hohe Anzahl an Polygonen auf. Zum zweiten war der geometrische Grundaufbau der WEA, die sogenannte Topologie nicht geeignet um ein gutes Shading in Unity zu ermöglichen. Zum dritten wurde die WEA zu einem Objekt zusammengefasst. Es ist also nicht möglich Baugruppen auszublenden, einzelnen Teilen verschiedene Shader zuzuweisen oder einzelne Teile zu animieren. Die WEA für einen VR-Prototypen zu optimieren schien mir daher ein geeignetes und spannendes Thema für dieses ICW. Natürlich kann auf diesem Wege ebenfalls evaluiert werden ob ein VR-Prototyp in einem technisch-industriellen Kontext überhaupt Sinn macht. Die Funktionsweise und der Aufbau dieser WEA werden im folgenden Unterkapitel „1.3 Fachlicher Kontext“ beschrieben.

## 1.3 Fachlicher Kontext

Die Funktionsweise einer WEA richtet sich vor allem nach der Bauart. Aber alle Anlagen erzeugen aus die Windenergie Strom. Die im Wind enthaltene Leistung überträgt sich auf den Rotor der Anlage, versetzt diesen in eine Drehbewegung und treibt den in der WEA verbauten Generator an. Es wird Windenergie umgewandelt in mechanische Energie, welche dann in elektrische Energie umgewandelt wird.

Generell sind Windenergieanlagen dafür konzipiert einen optimalen Energieertrag zu liefern. Diese Anlagen sind dabei unterschiedlichen Windbedingungen ausgesetzt, weshalb diese sich automatisch darauf reagieren müssen um zu einer stabilen und sicheren Stromversorgung beizutragen.<sup>1</sup>

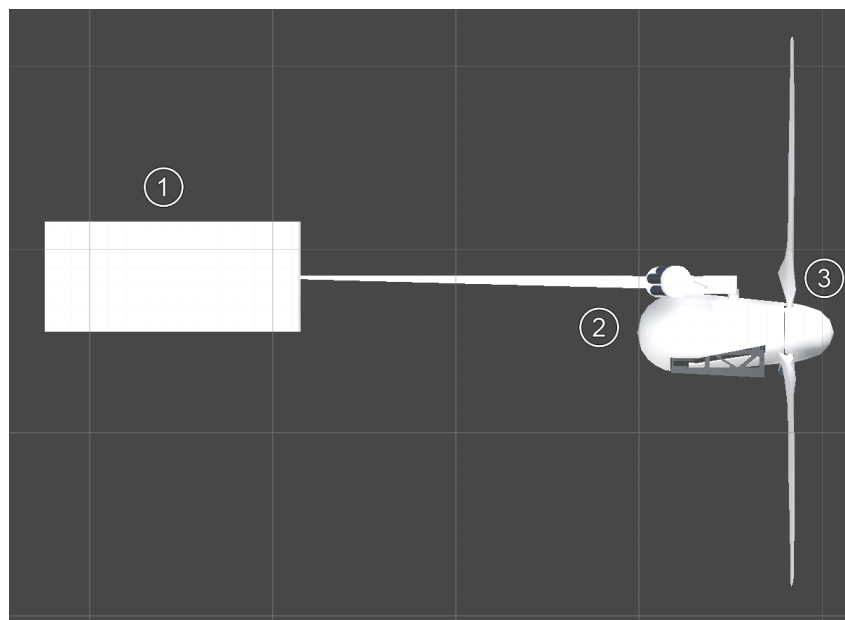


Abbildung 1.1: Aufbau einer WEA.

Der äußere Aufbau ist gekennzeichnet durch ein Gehäuse (Gondel) mit montierter Windrichtungsnachführung (siehe Abb. 1.1 – 2) sowie einer am Windfahnenhebel angebrachten Windfahne (siehe Abb. 1.1 – 1). Der Aufbau ist drehend gelagert um auch bei wechselnden

<sup>1</sup>Vgl. Bundesverband WindEnergie e. V. (2018): *Funktionsweise von Windenergieanlagen*.  
<https://www.wind-energie.de/themen/anlagentechnik/funktionsweise/>,  
abgerufen am 20.08.2018.

Windverhältnissen einen vollautomatischen Betrieb zu gewährleisten. Ein solches Regelungssystem ist für einen zuverlässigen Betrieb unabdingbar.

Der Rotor mit an der Narbe montierten Rotorblättern (siehe Abb. 1.1 – 3) ist nach aerodynamischen Prinzipien konstruiert und dient der Umwandlung der im Wind verfügbaren kinetischen Energie in mechanische Rotationsenergie.<sup>2</sup>

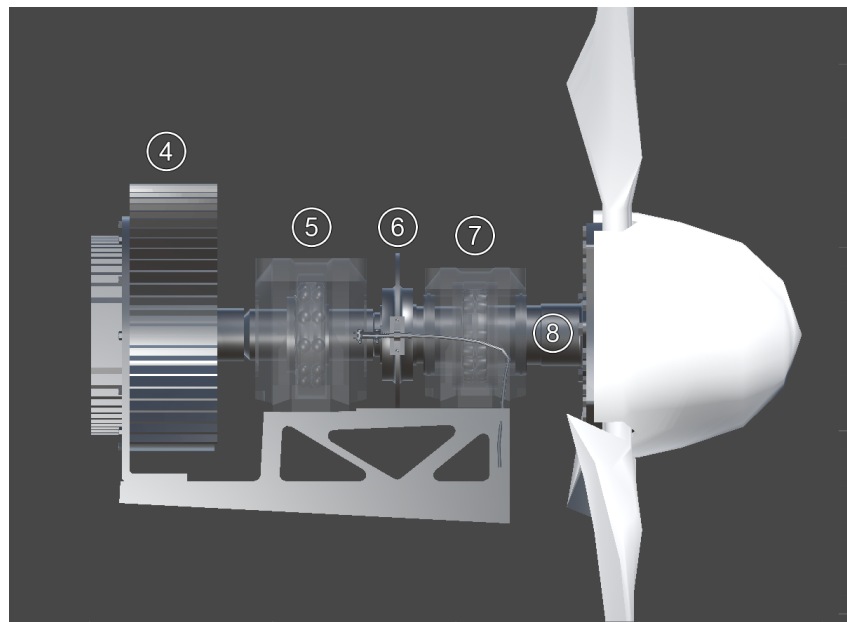


Abbildung 1.2: Aufbau einer WEA im Detail.

Ein Generator ist für die Stromerzeugung eingebaut (siehe Abb. 1.2 – 4). Dieser wandelt auf Basis des Induktionsgesetzes mechanische Rotationsenergie in elektrische Energie um. Es wird eine Spule in einem Magnetfeld in Rotation versetzt. Durch die Rotation wird an den Klemmen der Spule eine Sinusförmige Spannung induziert.<sup>3</sup> In diesem Beispiel erhält der Generator die verfügbare Rotationsenergie über eine Welle (siehe Abb. 1.2 – 8), die über eine Kupplung mit dem Anker (Rotor) des Generators verbunden ist. Zur Kühlung während des Betriebs sind auf der Außenseite Finnen aus Aluminium angebracht.

Die Lagerung wird über zwei Wälzlager realisiert und als Fest-Los-Lagerung bezeichnet.

<sup>2</sup>Vgl. Silvio Chemnitz, Sylvio Donner, Florian Hinze, Mats Mojem, Patrick Quandt, Oliver Seidler, Moritz Will, Jens Wuthe (2013): *Windpumpensysteme zur dezentralen Energieversorgung von Abwassersystemen*. TU Berlin, S. 10 ff.

<sup>3</sup>Vgl. Prof. Dr. G. Buch, Prof. Dr. M. Krug (2012): *Kurzschriftum zur Lehrveranstaltung „Elektrische Bordnetze“ im Studiengang Fahrzeugtechnik*. Hochschule München, S. 1.1.

Das Loslager ist ein Pendelkugellager (siehe Abb. 1.2 – 5) und dient zur Aufnahme der Radialkräfte, sprich die Kräfte die von außen auf die Welle wirken. Als Festlager dient ein Pendelrollenlager, (siehe Abb. 1.2 – 7) dass die kombinierten Axial- und Radialbelastungen aufnimmt. Unter Axialkraft versteht man die Belastung die längs der Achse wirkt. Müssen Wartungsarbeiten o.ä. an der WEA durchgeführt werden kann automatisches Anlaufen durch eine Bremse (siehe Abb. 1.2 – 6) verhindert werden. Mithilfe eines Bowdenzuges, also einem Seilzug der mechanische Kraft auf ein bewegliches Maschinenelement, in diesem Fall den Bremsbolzen überträgt, kann die WEA vom Boden aus verriegelt werden.

Die hier betrachtete WEA zählt zu den kleineren Modellen und hat eine Narbenhöhe von 10m. Der Turm, welcher in diesem Prototyp aus Darstellungsgründen nicht berücksichtigt wird weist eine Höhe von 9,73m auf.<sup>4</sup>

---

<sup>4</sup>Vgl. Silvio Chemnitz, Sylvio Donner, Florian Hinze, Mats Mojem, Patrick Quandt, Oliver Seidler, Moritz Will, Jens Wuthe (2013): *Windpumpsysteme zur dezentralen Energieversorgung von Abwassersystemen*. TU Berlin, S. 35–38, 44.

# Kapitel 2

## Datenverarbeitung in CAD

### 2.1 Anwendungsbereiche von CAD

Die Grundlage dieser Arbeit sowie des Prototypen bilden Daten, welche in 3D-CAD-Programmen erstellt wurden. Daher wird im Rahmen dieser Arbeit CAD als Synonym für 3D-CAD verwendet. Im Gegensatz zu 2D-CAD Systemen können in 3D-CAD Systemen Produkte realer konstruiert werden. Das kann sich positiv auf die Entwicklungszeit auswirken. Kollisionsbetrachtungen ermöglichen eine Fehlererkennung, bevor das erste Teil gefertigt wird.<sup>5</sup> Es ist möglich in CAD anhand von Materialeigenschaften physikalische Eigenschaften wie z.B. Festigkeit, Elastizität etc. zu simulieren. Aufgrund dieser Flexibilität sind CAD-Programme heute in fast allen technischen Zweigen vertreten, darunter Architektur, Bauingenieurwesen, Maschinenbau, Elektrotechnik bis hin zur Zahntechnik.<sup>6</sup>

### 2.2 3D-Modellierung in CAD

In einer 3D-Umgebung werden Modelle in dreidimensionaler Form modelliert und persistiert. Ein dreidimensionaler Aufbau ermöglicht das Nachbilden einer realitätsnahen

---

<sup>5</sup>Vgl. Dipl. Ing. (FH) Bettina Clauß, Helmut Prof. Dr.-Ing. von Eiff (2013): *CAD Grundkurs*. Hochschule Esslingen, S. 2.

<sup>6</sup>Vgl. Wikipedia (2018): *CAD*.

<https://de.wikipedia.org/w/index.php?title=CAD&oldid=178934444>,  
abgerufen am 23.08.2018.

Darstellung, das Rendern aus allen Blickwinkeln und Perspektiven sowie eine bessere räumliche Betrachtung. Das Hauptaugenmerk des Ingenieurs liegt dabei vor allem auf Funktionalitäten wie dem technischen Zeichnen, die Erstellung von Arbeitspläne, Montage- und Bedienungsanleitungen oder der technisch visuellen Darstellung, wie der Kollisionsbetrachtung oder der Zusammenbau-, Einbau- und Montageuntersuchungen. Die in dieser Arbeit thematisierte Betrachtung bezieht sich vor allem auf die Nutzung außerhalb der CAD-Umgebung. Der Vollständigkeit werden nachfolgend alle rechnerinternen Repräsentationen die in CAD vorliegen betrachtet.

- **Kantenmodelle:** Kantenmodelle (auch Drahtgitter oder Wireframe) repräsentieren ein Objekt anhand von Kanten. Diese Darstellung enthält keinerlei Informationen über die Flächen oder das Volumen eines Körpers. Kantenmodelle dienen häufig als Hilfsgeometrie, zu Erzeugung von Flächen oder als Darstellungsart von Volumen- oder Flächenmodellen.
- **Flächenmodelle:** Als Flächenmodelle werden „hohle“ Objekte bezeichnet, deren äußere Form durch Flächen beschrieben wird. Eine intuitive Anpassung der Objekthülle ist mit Hilfe von Kontrollpunkten oder Kontrollnetzen ohne Einschränkung möglich. Unter Zuhilfenahme von analytisch beschreibbarer (Translationsflächen, Regelflächen) sowie analytisch nicht beschreibbarer Flächen (B-spline-, NURBS-Flächen) lassen sich jegliche Formen modellieren.
- **Volumenmodelle:** Unter Volumenmodelle versteht man Körper, die neben einer Hülle auch eine Materialdichte besitzen, woraus das System automatisch eine Masse interpretiert. Auf diese Weise bleibt die geometrische Konsistenz bei Manipulation des Objektes erhalten. Aufgrund dieses zusätzlichen Parameters kann ein hoher Grad an Automatisierung sichergestellt werden. Es können Eigenschaften wie Trägheit, Schwerpunkt, Gewicht etc. durch das CAD-System automatisch abgeleitet und als Parameter für Simulationen übergeben werden.

Auf alle beschriebenen Modelle lassen sich räumliche Operationen wie Translation, Rotation und Skalierung anwenden. Zusätzlich existieren für jedes Modell spezielle Werkzeuge um diese zu verformen, zu zerschneiden, zu verdrehen oder anderweitig zu manipulieren.<sup>7</sup>

---

<sup>7</sup>Vgl. Wikipedia (2018): CAD.

<https://de.wikipedia.org/w/index.php?title=CAD&oldid=178934444>,  
abgerufen am 23.08.2018.

## 2.3 Exportformat in CAD

Für den Export aus SolidWorks einem 3D-CAD-Programm des französischen Softwareentwicklers Dassault Systèmes musste ein Dateiformat gefunden werden, dass auch von einem gängigen 3D-Programm gelesen werden kann. Als alternativlos hat sich STL erwiesen. Das Ergebnis des Exportes war nicht optimal (siehe „3.2 Analyse des CAD-Exports“). Nichtsdestotrotz hat die Übertragung schnell und ohne essentiellen Informationsverlust funktioniert. STL – Standard Transformation Language, Surface Tessellation Language oder Standard Triangulation Language ist einer der ältesten und im Kontext des 3D-Druckes das gängigste Format. STL beschreibt die Oberfläche eines Objektes näherungsweise durch Triangulation mit unterschiedlich großen Dreiecken. Der Datensatz besteht aus den Koordinaten der Eckpunkte jedes Dreiecks sowie dessen Normalenvektor (siehe Codeauszug 2.1<sup>8</sup>).<sup>9</sup>

```
1 solid <name>
2 facet normal ni nj nk
3     outer loop
4         vertex v1x v1y v1z
5         vertex v2x v2y v2z
6         vertex v3x v3y v3z
7     endloop
8 endfacet
9 ...
10 endsolid <name>
```

Codeauszug 2.1: STL ASCII Schema.

<sup>8</sup>Dave Touretzky (o.J.): *STL Files and Slicing Software*. Carnegie Mellon University Pittsburgh, Pennsylvania, S. 5.

<sup>9</sup>Vgl. Heiko Heckner, Marco Wirth (2014): *Vergleich von Dateiformaten für 3D-Modelle*. Julius-Maximilians-Universität Würzburg, S. 8.

# Kapitel 3

## Aufbereitung der 3D-CAD Daten

### 3.1 Eine kurze Einführung zum Aufbau von 3D-Objekten

In gängigen 3D-Programmen geschieht die Modellierung mit Hilfe von polygonalen Netzen (meshes), welche in der 3D-Computergrafik häufig anzutreffen sind. Polygonnetze, also Netze die sich aus einer Menge von miteinander verbundenen Polygonen zusammensetzen, bestehen zumeist aus Dreiecken und Vierecken. Es lassen somit einfache geometrische Figuren wie Zylinder, Kugeln, Prismen, Würfel, Tetraeder, Pyramiden etc. aber auch komplexere Strukturen erschaffen, um bspw. Modelle für Computerspiele oder Animationsfilme zu erstellen.<sup>10</sup>

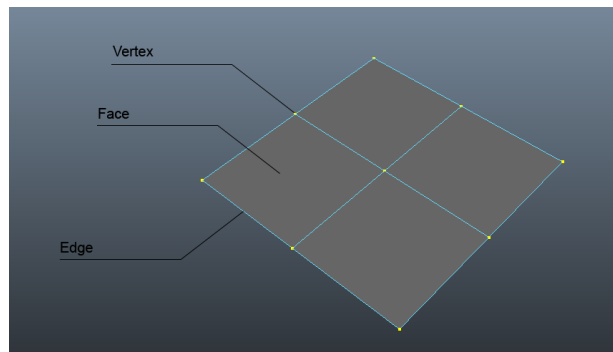


Abbildung 3.1: Eine Plane, bestehend aus vier quadratischen Polygonen.

In Autodesk Maya werden polygonale Flächen auch als Faces bezeichnet und haben drei oder mehr begrenzende Kanten, die Edges genannt werden. Die Edges wiederum sind

---

<sup>10</sup>Vgl. Michael Bender, Manfred Brill (2006): *Computergrafik*. 2.Aufl. München: Carl Hanser Verlag. S. 19.



über Punkte, den sogenannten Vertices (Plural) miteinander verbunden. Diese drei formgebenden Elemente können transformiert werden, um ein polygonales Netz zu deformieren. Genutzt werden dafür die verschiedenen Transform-Attribute, Translation, Rotation und Skalierung.<sup>11</sup> Zusätzlich können weitere Unterteilungen eingesetzt werden, um ein Polygonnetz feiner zu strukturieren. Es ist möglich Faces, Edges und Vertices miteinander zu verbinden, aufzutrennen oder um jeweils neue Elemente zu erweitern (extrudieren).

## 3.2 Analyse des CAD-Exports

Bevor das aus CAD exportierte Modell in Unity importiert wird sollte selbiges vorher in einem geeigneten 3D-Programm überprüft werden. Für diesen Zweck wird, wie im vorangegangenen Kapitel erwähnt auf Autodesk Maya zurückgegriffen. Maya ist eine professionelle Software für Modellierung, Animation und Rendering von 3D Objekten und Szenen.<sup>12</sup>

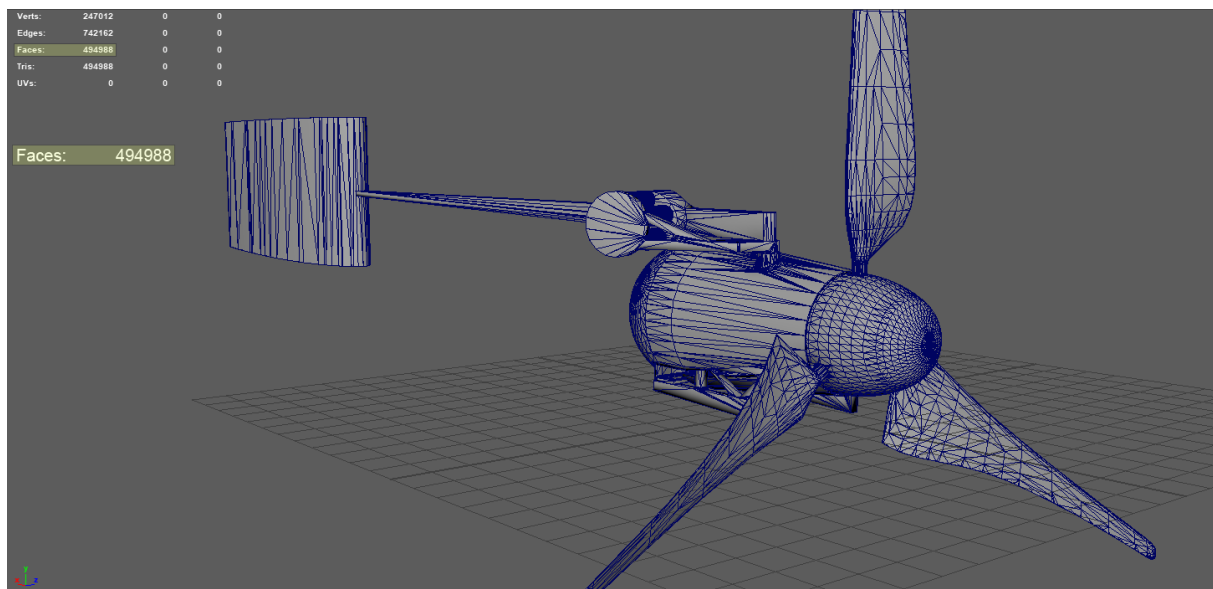


Abbildung 3.2: Ergebnis des Exportes der WEA aus der CAD-Anwendung bestehend aus 494.988 Polygonen.

<sup>11</sup>Vgl. Todd Palamer (2014): *Vgl. Mastering Autodesk Maya 2015*. 1.Aufl. Indianapolis: Sybex. S. 107 f.

<sup>12</sup>Vgl. Autodesk (2018): *MAYA*.

<https://www.autodesk.de/products/maya/overview>,  
abgerufen am 30.08.2018.

Das importierte Modell (siehe Abb. 2.1) weist mehrere Eigenschaften auf, welche es für einen Import sowie eine Weiterverarbeitung in Unity ungeeignet machen. Die betreffenden Eigenschaften sind nachfolgend aufgelistet. Generell sollten Modelle mit möglichst wenigen Polygonen auskommen, da ein hoher Detailgrad über Texturen generiert werden kann. Im Falle einer technischen Darstellung die neben der äußeren Verkleidung auch innenliegende technische Baugruppen wie Wälzlager, Getriebe o.ä. abbildet, kann dass die Anzahl der Polygone deutlich erhöhen. Schließlich müssen diese in adäquater Qualität dargestellt werden. Natürlich gilt aber auch hier der Grundsatz, dass eine zu geringe Anzahl zu lasten der Darstellungsqualität und eine zu hohe Anzahl zu lasten der Performance geht. Es gilt also einen guten Mittelweg zu finden.

- **Polygon count:** Das Modell besteht aus nicht ganz 500.000 Polygonen, was für eine Echtzeitanwendung sehr viel ist. Laut der Unity Dokumentation sollte der Polycount von dem Zielsystem und der angestrebten Qualität abhängig sein. Das Optimum liegt für Desktopanwendungen bei 1500 bis 4000 Polygonen pro Objekt. Falls sehr viele Objekte zur selben Zeit aktiv sind wird eine Reduktion der Polygone empfohlen.<sup>13</sup>
- **Unterteilung in Einzelobjekte:** Die WEA setzt sich aus vielen Einzelobjekten zusammen. Diese sind vom großen Rotorblatt bis zur kleinsten Schraube zu einem einzigen Objekt zusammengefasst. Es ist also nicht möglich einzelne Teilobjekte in Unity separat mit Materialien und Skripten auszustatten oder zu Animieren.
- **Flächenverteilung:** Ferner führt die ungleichmäßige Flächenverteilung im STL Export zu Darstellungsfehlern beim Shading (siehe Abb. 2.2). Eng zusammenliegende Edges beeinflussen das visuelle Endergebnis negativ.

---

<sup>13</sup>Vgl. Unity Documentation (2018): *Modeling characters for optimal performance*.  
<https://docs.unity3d.com/Manual/ModelingOptimizedCharacters.html>,  
abgerufen am 30.08.2018.

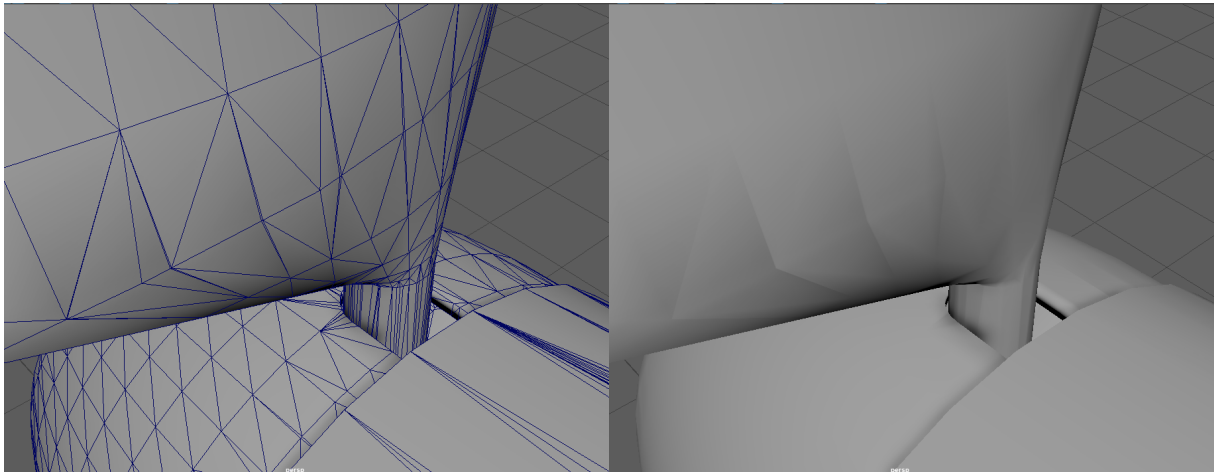


Abbildung 3.3: Shadingfehler am Exportmodell.

Um diese Probleme zu lösen ist eine Retopologisierung also eine Überarbeitung der geometrischen Grundstruktur und eine damit einhergehende Reduktion der Flächen nötig. Diese Vorgehensweise wird im folgenden Kapitel „3.3 Retopologisierung des CAD-Exports“ beschrieben.

### 3.3 Retopologisierung des CAD-Exports

Die Aufbereitung des Modells kann händisch oder automatisiert erfolgen. Die händische Retopologisierung ist aufwändiger und erfordert Kenntnisse in der Polygonmodellierung. Die Qualität und die Anzahl der Polygone des Modells können aber exakt gesteuert werden. Eine automatische Reduktion hat aufgrund von invalider Geometrie, sogenannter non-manifold geometry aber nicht funktioniert. Das folgende Zitat aus der Maya LT Dokumentation von Autodesk beschreibt in Kürze das Problem.

„Some types of polygon geometry will not work in Maya. Invalid geometry includes vertices that are not associated with a polygon edge and polygon edges that are not part of a face (dangling edges). While Maya does not let you create these types of geometry, it may be possible to import these types from other software applications.“<sup>14</sup>

<sup>14</sup>Autodesk (2015): *Two-manifold vs. non-manifold polygonal geometry*.

Der Import aus CAD enthielt diese invalide Geometrie. Bei einem solch komplexen Modell eine Fehlersuche einzuleiten ist daher sehr zeitaufwendig werden und das Ergebnis der Autoreduktion ist Qualitativ minderwertiger wie die folgende Abbildung aus einem anderen Projekt zeigt (siehe Abb. 2.4).

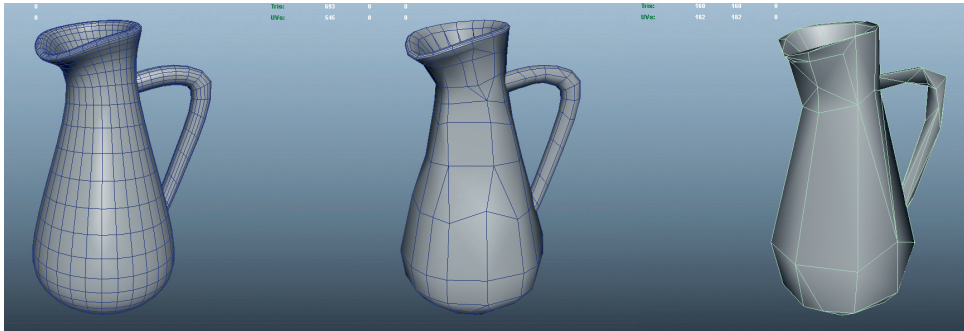


Abbildung 3.4: Automatische Reduktion der Polygone auf (v.l.n.r.) 1536, 382 und 117.

Daher erschien es praktikabler das Modell händisch aufzubereiten. Die Aufbereitung erfolgte in Maya anhand von Polygonmodellierung auf Basis von primitiven Körpern (Primitives). das können bspw. Würfel, Cylinder, Kugeln, Tori etc. sein. Mit Erzeugung dieser Primitives lässt sich durch Flächenextrusion die grobe Form nachbauen und mit Hilfe weiterer Unterteilungen verfeinern. Diese Anpassung des Primitives an die Form des CAD-Exportes geschieht für alle drei Raumdimensionen. Um eine sehr genaue Abbildung zu Erzeugen können die Vertices des erzeugten Primitives an die Vertices des CAD-Modells „angedockt“ werden. Es ist mit wenigen Werkzeugen möglich eine Abbildung des ursprünglichen Objektes mit guter Topologie und einer deutlich reduzierten Anzahl an Polygonen zu erzeugen. Die Vorgehensweise wird nachfolgend anhand eines Beispiels verdeutlicht (siehe Abb. 2.5).

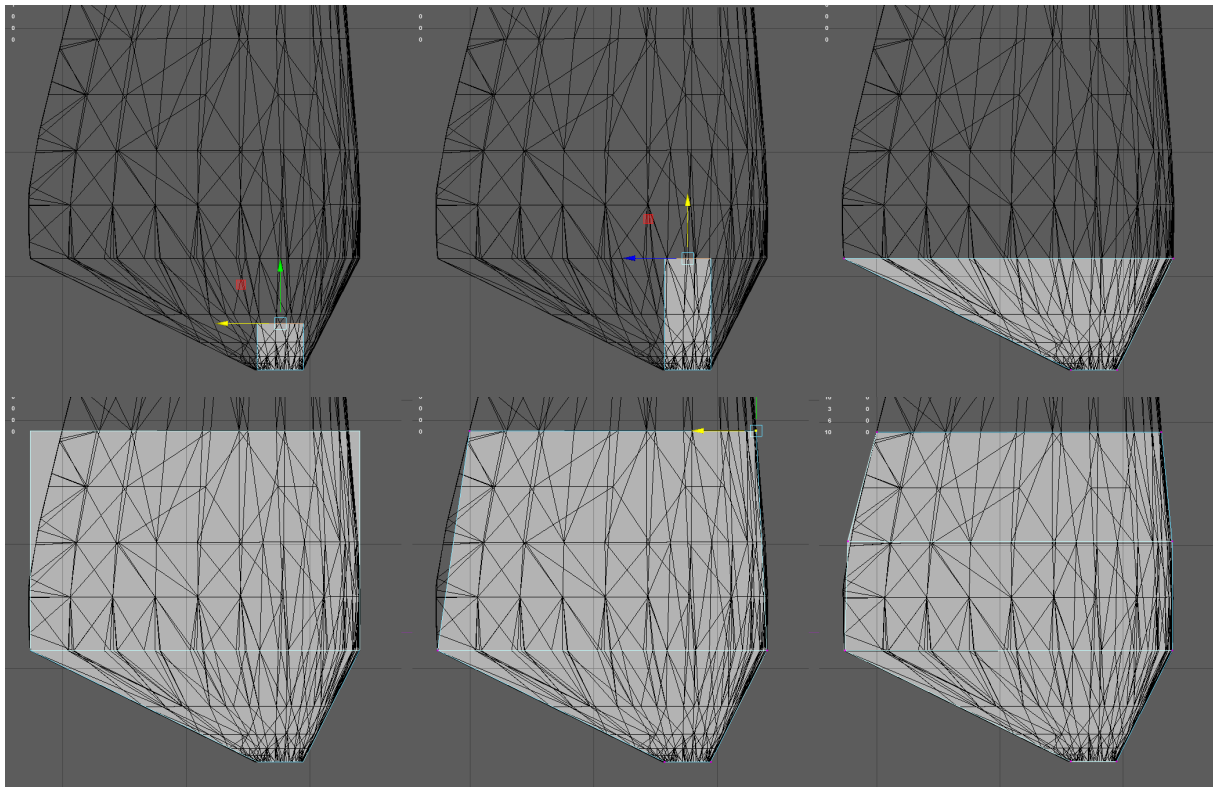


Abbildung 3.5: Retopologisierung einer Teilfläche durch Erzeugung eines Primitives, Anpassung der Vertices, Flächenextrusion und Einsetzen einer weiteren Unterteilung (Edgeloop).

Die Gesamte Retopologisierung des CAD-Exportes fand mit mehr oder weniger diesen Werkzeugen statt. Die Abbildung verdeutlicht noch einmal den Unterschied. Das alte Rotorblatt weist eine Anzahl von 1130 Polygonen auf. Das neue Modell setzt sich aus 76 Polygonen zusammen. Es konnten also etwas mehr als 93% der Flächen eingespart werden (siehe Abb. 2.6).

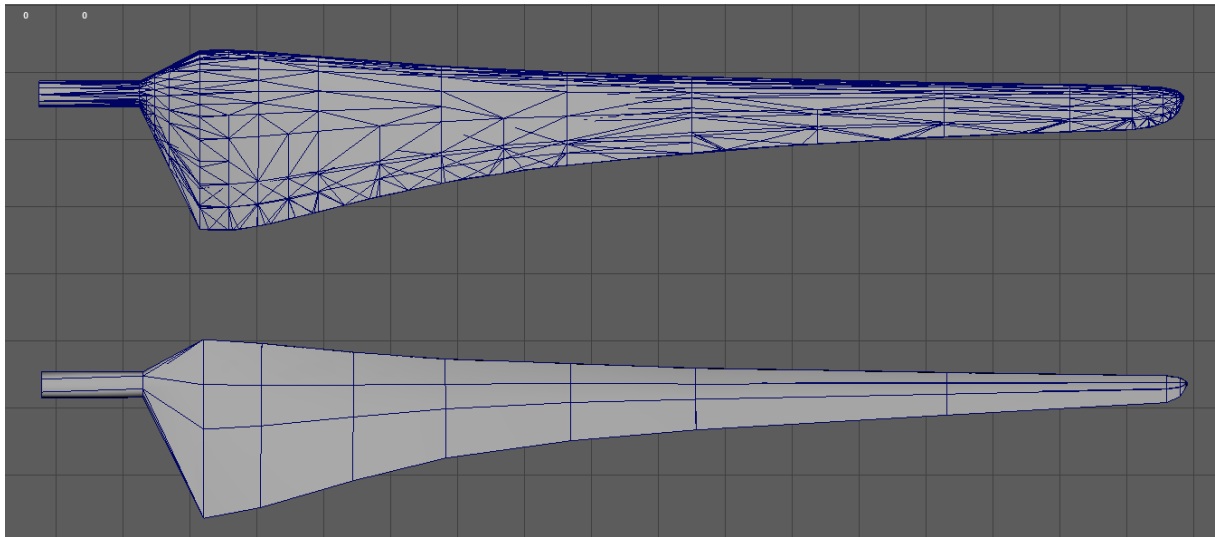


Abbildung 3.6: Vergleich des Rotorblattes aus dem CAD-Export (1130 Polygone) mit dem neu erstellten Rotorblatt (76 Polygone).

Für ein einzelnes Objekt sind über 1000 Polygone kein Problem. Da es sich bei diesem Rotorblatt aber um eines von vielen Bauteilen handelt, kann die Polygondichte sehr schnell ausufern und die Hardware überfordern. Wenn sich das Objekt mit weniger Polygonen ähnlich gut beschreiben lässt, sollte diese Sparmaßnahme unbedingt greifen. Zudem gilt natürlich auch hier der Grundsatz des sparsamen Umgangs mit Hardwareressourcen, wie dies generell in der Softwareentwicklung gewünscht wird.

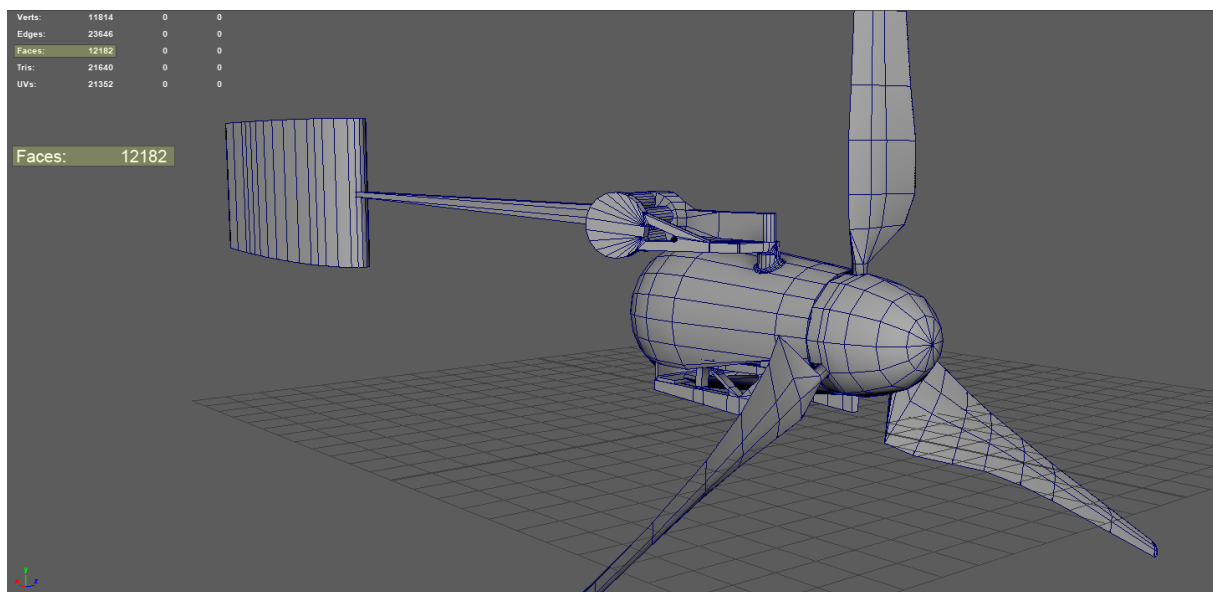


Abbildung 3.7: Ergebnis der Retopologisierung bestehend aus 12.182 Polygonen.

Das Endergebnis der Retopologisierung kann in der folgenden Abbildung betrachtet werden (siehe Abb. 2.7). Im Gegensatz zum CAD-Export, der aus 494.988 Polygonen besteht, enthält das neue Modell nur 12.182 Polygone. Es lässt sich konstatieren, dass in Summe ca. 97,5 aller Flächen eingespart wurden.

# Kapitel 4

## Umsetzung in Unity

### 4.1 VRTK – Virtual Reality Toolkit

Für die Entwicklung des Prototypen wurde das Virtual Reality Toolkit (VRTK) verwendet. VRTK ist ein Open-Source Framework für Unity mit dessen Hilfe es möglich ist, in kurzer Zeit eine voll funktionsfähige VR-Anwendung zu entwickeln. Das Projekt startete im April 2016 und wird seit dem weiterentwickelt.<sup>15</sup> Ein großer Vorteil des VRTK-Frameworks ist laut seinem Initiator Harvey Ball (thestonefox) die Plattformunabhängigkeit. VRTK verfügt über eine Abstraktionsschicht die es ermöglicht, dass Komponenten für die Mechanik eines Spiels auf jedem unterstützten SDK funktionieren. Die Anwendung läuft auf SteamVR oder Oculus oder PSVR ohne zusätzlichen Programmieraufwand. Falls ein Projekt auf Basis von SteamVR entwickelt wird, man aber auf weitere Plattformen portieren möchte, so müssen Codeblöcke für andere SDKs umgeschrieben werden oder eben eine eigene Abstraktionsschicht entwickelt werden. Einige beliebte SteamVR Titel lassen sich laut Ball auch nicht ohne weiteres nach Oculus Home portieren. VRTK soll dieses Problem lösen.<sup>16</sup> Unterstützt werden gängige VR-Plattformen wie die Oculus Rift, Windows Mixed Reality oder die HTC VIVE. Auch eine Anbindung an Steam über das SteamVR Plugin ist möglich und wurde im Rahmen dieser Arbeit verwendet. Falls gerade keine VR-Hardware

---

<sup>15</sup>Vgl. GitHub (2018): *VRTK Code frequency*.

<https://github.com/thestonefox/VRTK/graphs/code-frequency>,  
abgerufen am 06.09.2018.

<sup>16</sup>Vgl. Ian Hamilton (2017): *VRTK's Open Source Tools Help New Developers Get Started In VR*.

<https://uploadvr.com/vrtk-stone-fox-unity-tool/>,  
abgerufen am 06.09.2018.



zur Verfügung steht, kann die Anwendung über den integrierten VR-Simulator getestet werden. Das VRTK-Framework implementiert die Grundlegenden Funktionalitäten, welche für eine VR-Anwendung benötigt werden. Dazu zählen unter anderem die Möglichkeit der Fortbewegung in einer Szene oder Interaktionen wie die Berührung, das Greifen und die Benutzung von Objekten. Auch die Interaktion mit UI-Elementen durch Berührung oder mittels Pointer sind implementiert. Weiterhin ist die Benutzung von Kontrollelementen wie Buttons, Hebel, Türen, Schubladen etc. möglich und kann, wenn nötig an gegebene Anforderungen angepasst.<sup>17</sup>

## 4.2 Szenenaufbau in Unity

Die Szene setzt sich aus verschiedenen Komponenten zusammen, die sich grob in drei Kategorien einteilen lassen.

- **Schaltpult:** Das „Schaltpult zur Anlagensteuerung“ (siehe Abb. 4.1 – 1) erfüllt verschiedene Funktionalitäten, welche die Steuerung der WEA betreffen. Weitere Erläuterungen sind im nachfolgenden Kaptiel „4.3 Implementierte Usecases“ beschrieben.
- **WEA:** Das eigentliche Modell der WEA (siehe Abb. 4.1 – 2) ist in der Szene platziert. Anhand dieses Modells soll die Funktionsweise verdeutlicht werden. Aktuell sind vor allem Mechanische Komponenten implementiert und animiert. In weiteren Iterationen könnte bspw. das Thema Stromerzeugung mit Hilfe eines Generator in die Visualisierung mit einbezogen werden.
- **Hinweisschilder:** Die Hinweisschilder (siehe Abb. 4.1 – 3) dienen dazu, den Anwender über die Benennung der Komponenten zu informieren. Problematisch an dieser Darstellungsweise ist, dass zu viele Schilder auf einer zu kleinen Fläche sich gegenseitig überlagern können und dadurch die Lesbarkeit beeinträchtigt ist. Gerade an Stellen, an denen sich viele Komponenten auf kleinem Raum befinden ist dieses Problem aufgetreten.

---

<sup>17</sup>Vgl. GitHub (2018): *VRTK - Virtual Reality Toolkit*.  
<https://github.com/thestonefox/VRTK>,  
abgerufen am 05.09.2018.

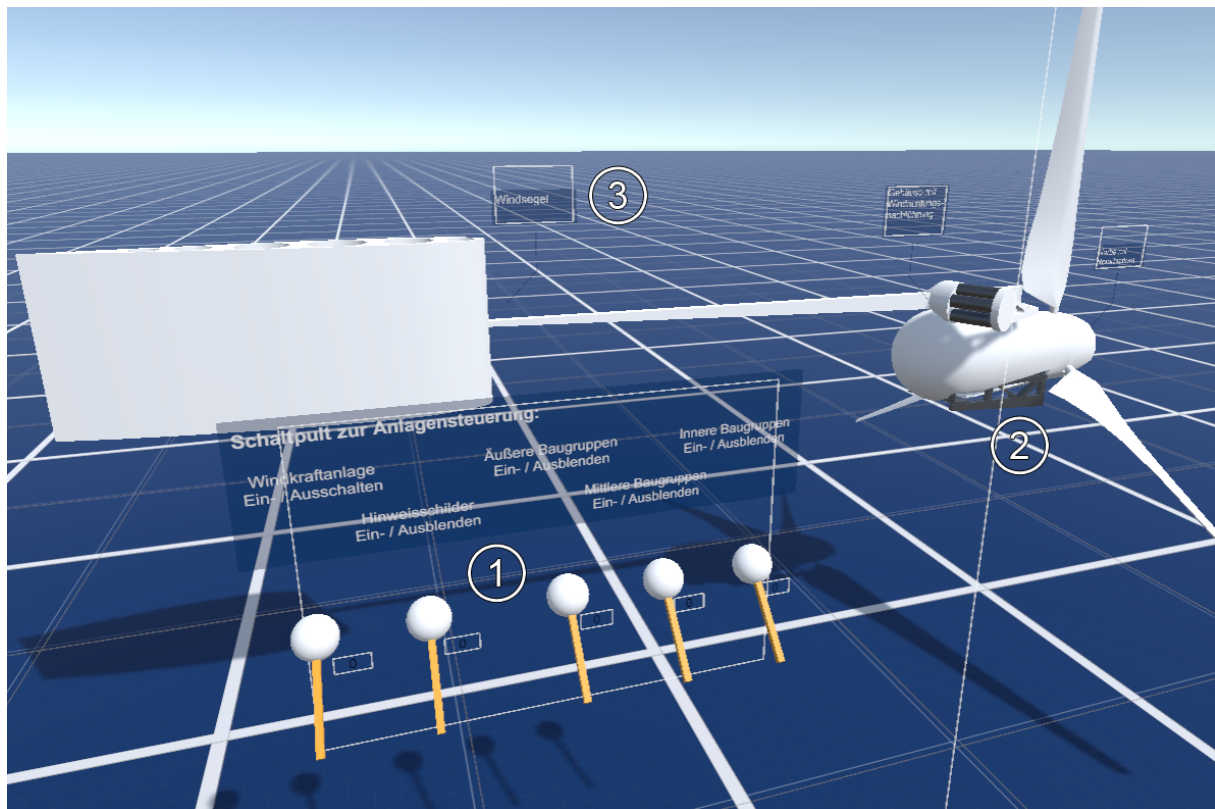


Abbildung 4.1: Szene in Unity.

Um die Szene zu strukturieren wurde die hierarchische Gruppierung vom VRTK-Framework übernommen. Diese der Szene untergeordneten GameObjects enthalten alle wichtigen Komponenten, welche für VR-Anwendung erforderlich sind. Die Folgende Struktur soll den hierarchischen Szenenaufbau verdeutlichen.

> **Szene**

- > **[VRTK\_SDKManager]:** Lädt Konfigurationen der SDKs (Windows MR, Oculus, SteamVR, UnityXR, VR-Simulator) und legt die Startreihenfolge fest.
- > **[VRTK\_SDKSetups]:** Enthält die Konfigurationsobjekte der oben genannten SDKs inklusive CameraRig, welches Kopf (Kamera) und Hände (Controller) beinhaltet.
- > **[VRTK\_Scripts]**

> [SceneScripts]

> SceneObjects

## 4.3 Implementierte Usecases

abc

# Kapitel 5

## Fazit

# Literatur

- [1] Bundesverband WindEnergie e. V. (2018): *Funktionsweise von Windenergieanlagen*. <https://www.wind-energie.de/themen/anlagentechnik/funktionsweise/>, abgerufen am 20.08.2018.
- [2] Silvio Chemnitz, Sylvio Donner, Florian Hinze, Mats Mojem, Patrick Quandt, Oliver Seidler, Moritz Will, Jens Wuthe (2013): *Windpumpensysteme zur dezentralen Energieversorgung von Abwassersystemen*. TU Berlin.
- [3] Prof. Dr. G. Buch, Prof. Dr. M. Krug (2012): *Kurzsriptum zur Lehrveranstaltung „Elektrische Bordnetze“ im Studiengang Fahrzeugtechnik*. Hochschule München.
- [4] Dipl. Ing. (FH) Bettina Clauß, Helmut Prof. Dr.-Ing. von Eiff (2013): *CAD Grundkurs*. Hochschule Esslingen.
- [5] Wikipedia (2018): *CAD*. <https://de.wikipedia.org/w/index.php?title=CAD&oldid=178934444>, abgerufen am 23.08.2018.
- [6] Dave Touretzky (o.J.): *STL Files and Slicing Software*. Carnegie Mellon University Pittsburgh, Pennsylvania.
- [7] Heiko Heckner, Marco Wirth (2014): *Vergleich von Dateiformaten für 3D-Modelle*. Julius-Maximilians-Universität Würzburg.
- [8] Michael Bender, Manfred Brill (2006): *Computergrafik*. 2.Aufl. München: Carl Hanser Verlag.
- [9] Todd Palamer (2014): *Mastering Autodesk Maya 2015*. 1.Aufl. Indianapolis: Sybex.

- 
- [10] Autodesk (2018): *MAYA*.  
<https://www.autodesk.de/products/maya/overview>,  
abgerufen am 30.08.2018.
  - [11] Unity Documentation (2018): *Modeling characters for optimal performance*.  
<https://docs.unity3d.com/Manual/ModelingOptimizedCharacters.html>,  
abgerufen am 30.08.2018.
  - [12] Autodesk (2015): *Two-manifold vs. non-manifold polygonal geometry*.  
<https://knowledge.autodesk.com/support/maya-lt/...>,  
abgerufen am 05.09.2018.
  - [13] GitHub (2018): *VRTK Code frequency*.  
<https://github.com/thestonefox/VRTK/graphs/code-frequency>,  
abgerufen am 06.09.2018.
  - [14] Ian Hamilton (2017): *VRTK's Open Source Tools Help New Developers Get Started In VR*.  
<https://uploadvr.com/vrtk-stone-fox-unity-tool/>,  
abgerufen am 06.09.2018.
  - [15] GitHub (2018): *VRTK - Virtual Reality Toolkit*.  
<https://github.com/thestonefox/VRTK>,  
abgerufen am 05.09.2018.

# Quelltextverzeichnis

2.1 STL ASCII Schema. . . . .	8
-------------------------------	---

# Anhang

## A Quellen

- A1 *Windpumpensysteme zur dezentralen Energieversorgung von Abwassersystemen* [PDF]
- A2 *Kurzschriftum zur Lehrveranstaltung „Elektrische Bordnetze“ im Studiengang Fahrzeugtechnik* [PDF]
- A3 *CAD-Grundkurs* [PDF]
- A4 *Vergleich von Dateiformaten für 3D-Modelle* [PDF]
- A5 *STL Files and Slicing Software* [PDF]

## B Software

- B1 *IndustrialVR Prototyp* [Unity-Anwendung]
- B2 *IndustrialVR Projekt* [<https://github.com/ChrisWodaeye/IndustrialVR.git>]