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
Matrikelnummer: 560597

Independent coursework
Sommersemester 2018

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit 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.

Berlin, den 28.09.2018

Chris Wodäge

Abkürzungsverzeichnis

CAD Computer-aided design oder rechnerunterstütztes Konstruieren

STL Standard Transformation Language oder Surface Tessellation Language

WEA Windenergieanlage

Abbildungsverzeichnis

Inhaltsverzeichnis

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, um aus diesem Modell eine VR-Anwendung zu entwickeln.

Zunächst wird in Kapitel 1 die Motivation zu dieser Arbeit kurz erläutert und mit Hilfe eines fachlichen Teils zur Funktion und zum Aufbau der WEA das Thema eingeleitet.

Weiterhin sollen in Kapitel 2 die Anwendungsbereiche von CAD-Programmen und die Möglichkeiten der 3D-Modellierung beschrieben werden. Darauf aufbauend wird auf den Transfer von CAD nach Autodesk Maya anhand des Exportformates eingegangen.

Das Kapitel 3 wird sich mit einigen Grundbegriffen der 3D-Modellierung beschäftigen. Diese Grundlagen sind essentiell für die folgende Analyse des CAD-Exports. Auch die Aufbereitung des 3D-Modells für den Import in Unity wird behandelt.

Kapitel 4 wird den Bereich um Unity abdecken. Hierzu wird das Open-Source Framework VRTK zur Erstellung einer VR-Anwendung untersucht, der Szenenaufbau in Unity beschrieben. Ferner werden die implementierten Anwendungsfälle aufgelistet und konkretisiert.

Zuletzt soll in Kapitel 5 mithilfe eines Fazits konkret auf die Optimierung des Workflows und die Automatisierung von Teilschritten der Arbeit eingegangen werden. Diese haben sich durch neue Entwicklungen ergeben, konnten aber aufgrund des Projektfortschritts und dem zusätzlichen Forschungsaufwand im Rahmen dieser Arbeit nicht mehr untersucht werden. Auch sollen abschließend noch Vorschläge und Ideen für weitere Funktionalitäten gegeben 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 in CAD 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 Unity als Technologie für die 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 nach Unity 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 mit einer Umkonvertierung des Dateiformates in eine Unity-Szene 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 einzigen Objekt zusammengefasst. Es war also nicht möglich Baugruppen auszublenden, einzelnen Teilen verschiedene Shader und Skripte zuzuweisen oder diese separat zu animieren. Die WEA für einen VR-Prototypen zu optimieren schien mir daher ein geeignetes und spannendes Thema für dieses ICW. Um die Thematik einzuleiten wird zunächst 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 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 Rotationsenergie, 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 automatisch darauf reagieren müssen um zu einer stabilen und sicheren Stromversorgung beizutragen.[1]

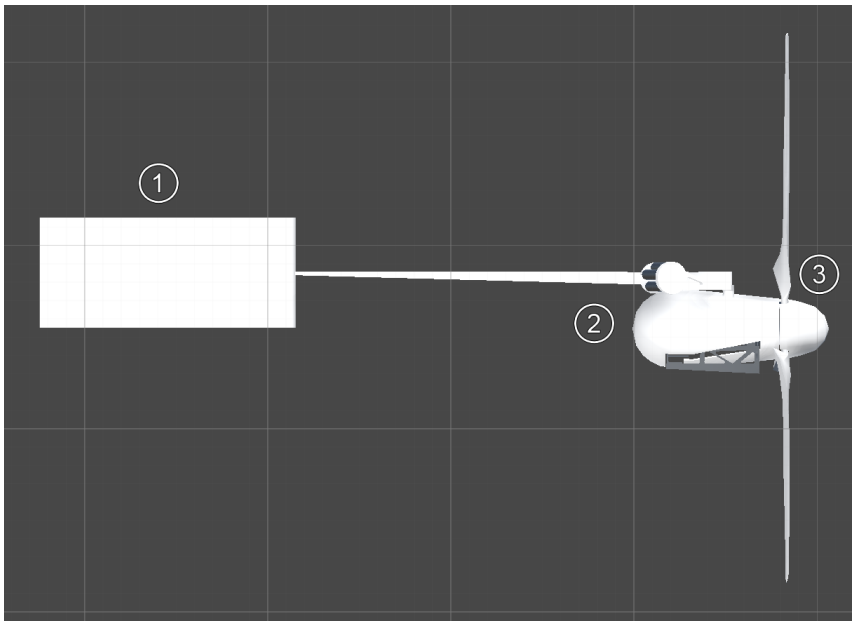

Abbildung 1.1: Äußerer Aufbau der WEA.

Der äußere Aufbau ist gekennzeichnet durch ein Gehäuse (die Gondel) mit montierter Windrichtungsnachführung (siehe Abb. 1.1 – 2), bestehend aus einer am Windfahnenhebel angebrachten Windfahne (siehe Abb. 1.1 – 1). Der Aufbau ist drehend gelagert um auch bei wechselnden Windverhältnissen einen vollautomatischen Betrieb zu gewährleisten. Ein

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

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.[2]

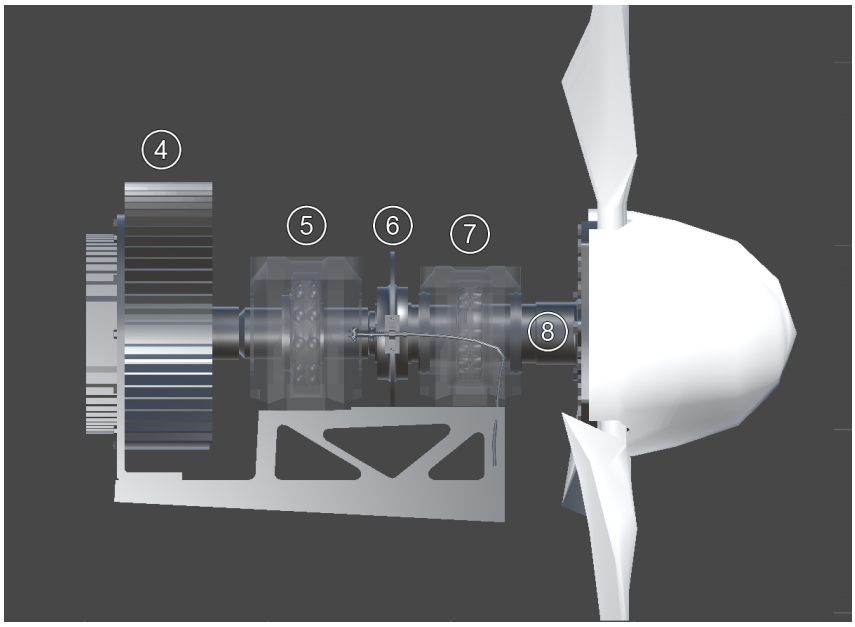

Abbildung 1.2: Innenliegende Baugruppen der WEA.

Für die Stromerzeugung ist ein Generator verbaut (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.[3] 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. Das Loslager ist ein Pendelkugellager (siehe Abb. 1.2 – 5) und dient zur Aufnahme der

[2]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. 10 ff.
[3]Vgl. Prof. Dr. G. Buch, Prof. Dr. M. Krug (2012): *Kurzskriptum zur Lehrveranstaltung „Elektrische Bordnetze“ im Studiengang Fahrzeugtechnik.* Hochschule München, S. 1.1.

Radialkräfte, also 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.[4]

[4]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.[5] 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.[6]

2.2 3D-Modellierung in CAD

In CAD werden Modelle in dreidimensionaler Form modelliert und persistiert. Ein dreidimensionaler Aufbau ermöglicht das Nachbilden einer realitätsnahen Darstellung, das

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

[6] Vgl. Wikipedia (2018): *CAD*. `https://de.wikipedia.org/w/index.php?title=CAD&oldid=178934444`, abgerufen am 23.08.2018.

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, der Erstellung von Arbeitsplänen, Montage- und Bedienungsanleitungen. Weitere Anforderungen sind unter anderem technisch visuelle Darstellungen, wie die Kollisionsbetrachtung oder die Zusammenbau-, Einbau- und die Montageuntersuchung. Die in dieser Arbeit thematisierte Betrachtung bezieht sich vor allem auf die Nutzung außerhalb der CAD-Umgebung. Der Vollständigkeit halber werden nachfolgend alle drei recheninternen 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 zur 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 Volumenmodellen versteht man Körper, die neben einer Hülle auch eine Materialdichte besitzen, woraus das CAD-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.[7]

[7] 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 geparsed werden kann. Als alternivlos hat sich das Dateiformat „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[8]).[9]

```
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.

Da Autodesk Maya und andere (nicht CAD) 3D-Programme ausschließlich auf Basis von Flächenmodellen arbeiten, müssen keine weiteren Informationen bzgl. des Modells übertragen werden.

[8] Dave Touretzky (o.J.): *STL Files and Slicing Software*. Carnegie Mellon University Pittsburgh, Pennsylvania, S. 5.

[9] 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 (oder 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 sich damit 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.[10]

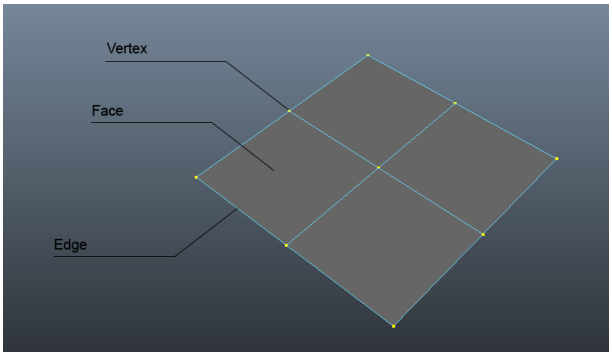

Abbildung 3.1: Eine Plane, bestehend aus vier quadratischen Polygonen.

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

[10] 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.[11] 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 Abschnitt erwähnt auf Autodesk Maya zurückgegriffen. Maya ist eine professionelle Software für Modellierung, Animation und Rendering von 3D-Objekten und Szenen.[12]

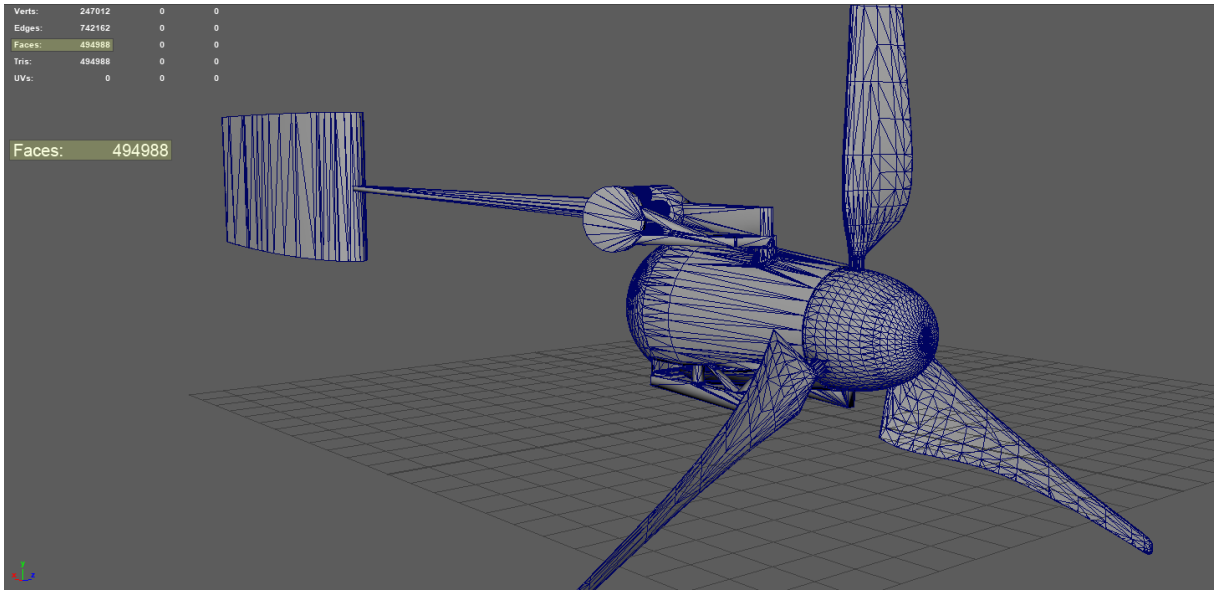

Abbildung 3.2: Ergebnis des Exportes der WEA aus der CAD-Anwendung bestehend aus 494.988 Polygonen.

[11] Vgl. Todd Palamer (2014): *Vgl. Mastering Autodesk Maya 2015*. 1.Aufl. Indianapolis: Sybex. S. 107 f.
[12] 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 Polygon Count vom 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.[13] Eine Empfehlung für eine maximale Beschränkung wird aber nicht gegeben.
- **Unterteilung in Einzelobjekte:** Die WEA setzt sich aus vielen Einzelobjekten zusammen. Diese sind vom größten 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.

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

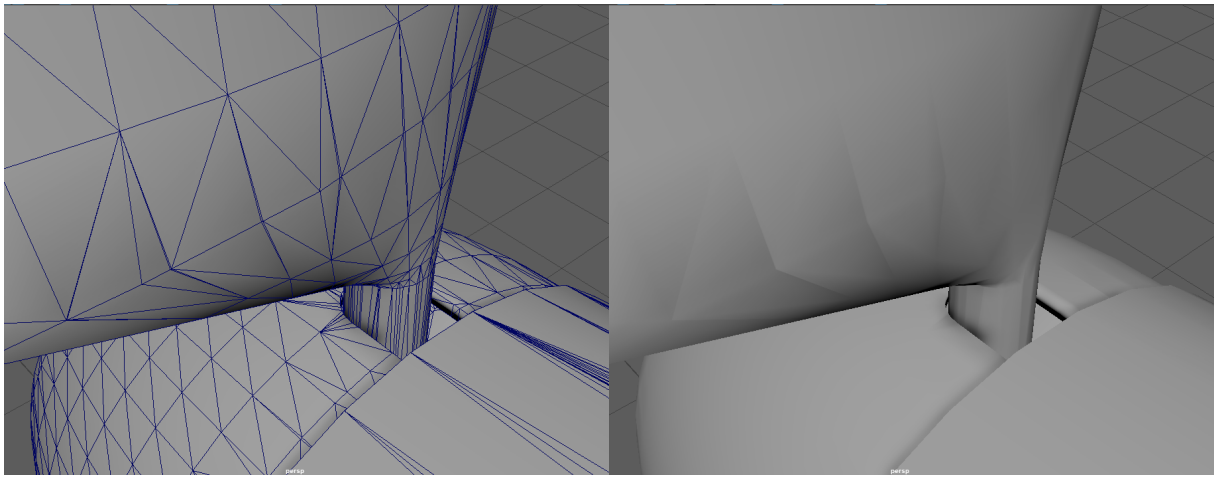

Abbildung 3.3: Shadingfehler am Exportmodell.

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.“[14]

[14] 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 aber sehr zeitaufwendig und das Ergebnis der Autoreduktion ist qualitativ minderwertig, wie die folgende Abbildung aus einem anderen Projekt zeigt (siehe Abb. 2.4).

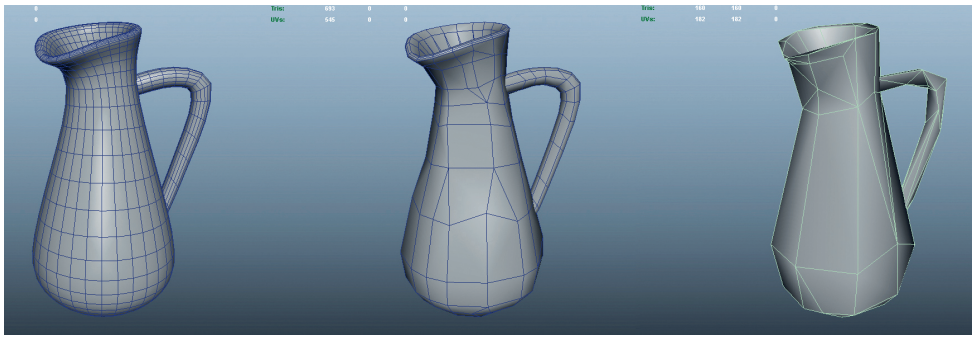

Abbildung 3.4: Automatische Reduktion der Polygone auf (v.l.n.r.) 1536, 382 und 117.

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, Zylinder, 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).

https://knowledge.autodesk.com/support/maya-lt/..., abgerufen am 05.09.2018.

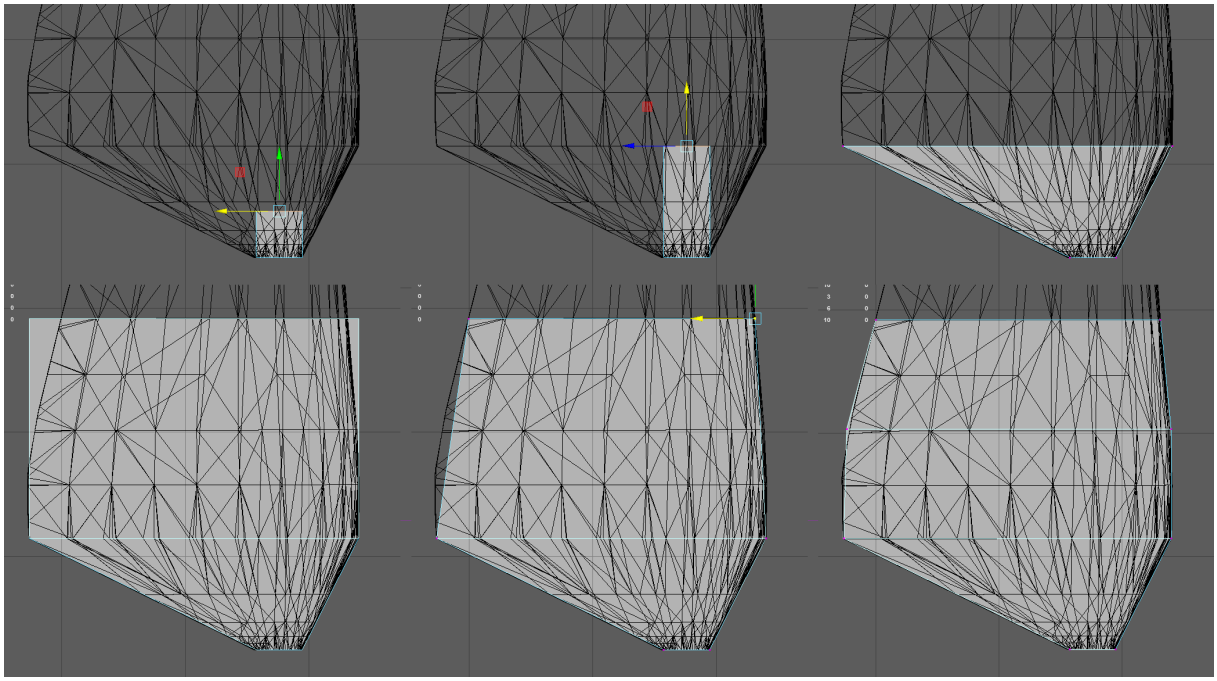

Abbildung 3.5: Retopologisierung einer Teilfläche durch Erzeugung eines Primitives, Anpassung der Vertices, Flächenextrusion und Einsetzen einer weiteren Unterteilung (Edgeloop).

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).

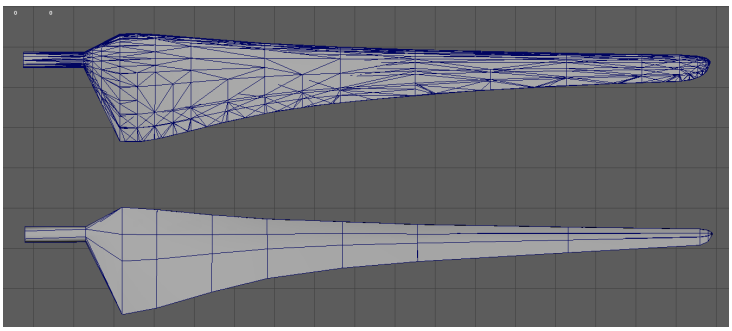

Abbildung 3.6: Vergleich des Rotorblattes aus dem CAD-Export (1130 Polygone) mit dem neu erstellten Rotorblatt(76 Polygone).

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 Sparmaßname unbedingt greifen. Zudem gilt natürlich auch hier der Grundsatz des sparsamen Umgangs mit Hardwareressourcen, wie das generell in der Softwareentwicklung der Fall sein sollte.

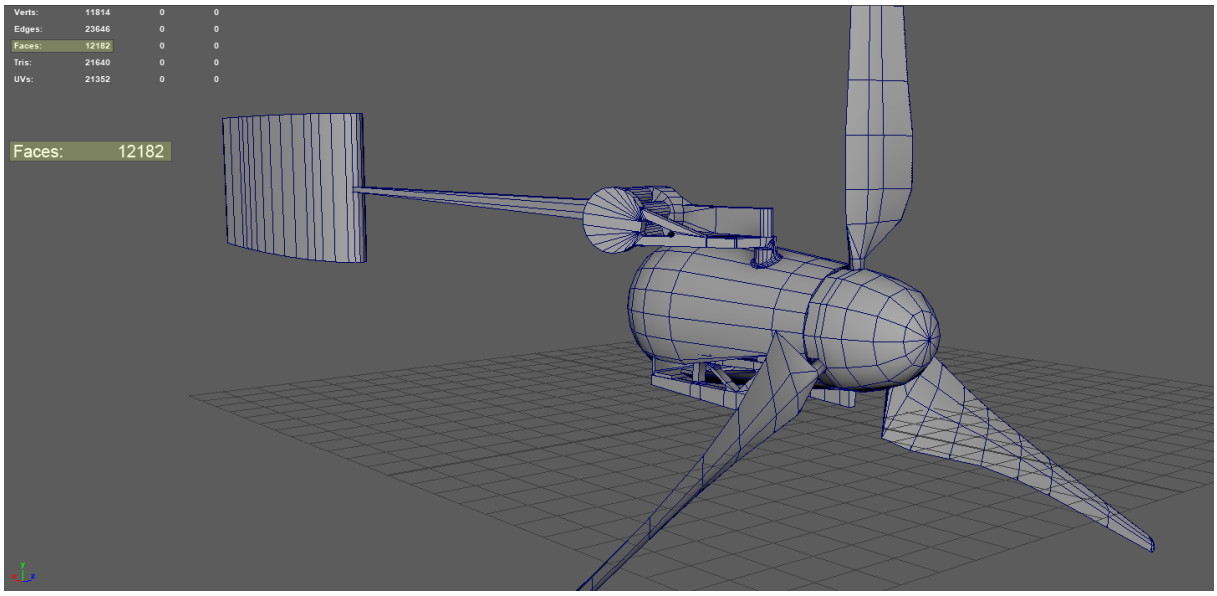

Abbildung 3.7: Ergebnis der Retopologisierung bestehend aus 12.182 Polygonen.

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.[15] Ein großer Vorteil des VRTK-Frameworks ist laut seinem Initiator Harvey Ball die Plattformunabhängigkeit. VRTK verfügt über eine Abstraktionsschicht die es ermöglicht, dass Komponenten für die Mechanik eines Spiels auf jedem unterstützen 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.[16] Unterstützt werden gängige VR-Platformen 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

[15] Vgl. GitHub (2018): *VRTK Code frequency.* `https://github.com/thestonefox/VRTK/graphs/code-frequency`, abgerufen am 06.09.2018.

[16] 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, Hebeln, Türen, Schubladen etc. möglich. Diese können, wenn nötig an gegebene Anforderungen angepasst.[17]

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 Unterkapitel „4.3 Implementierte Anwendungsfälle“ 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 mithilfe 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. Auch können diese andere Komponenten verdecken und so eine Betrachtung der WEA erschweren. Gerade an Stellen, an denen sich viele Komponenten auf kleinem Raum

[17] Vgl. GitHub (2018): *VRTK - Virtual Reality Toolkit.* `https://github.com/thestonefox/VRTK`, abgerufen am 05.09.2018.

befinden ist dieses Problem aufgetreten. Daher können die Hinweisschilder jederzeit ausgeblendet werden.

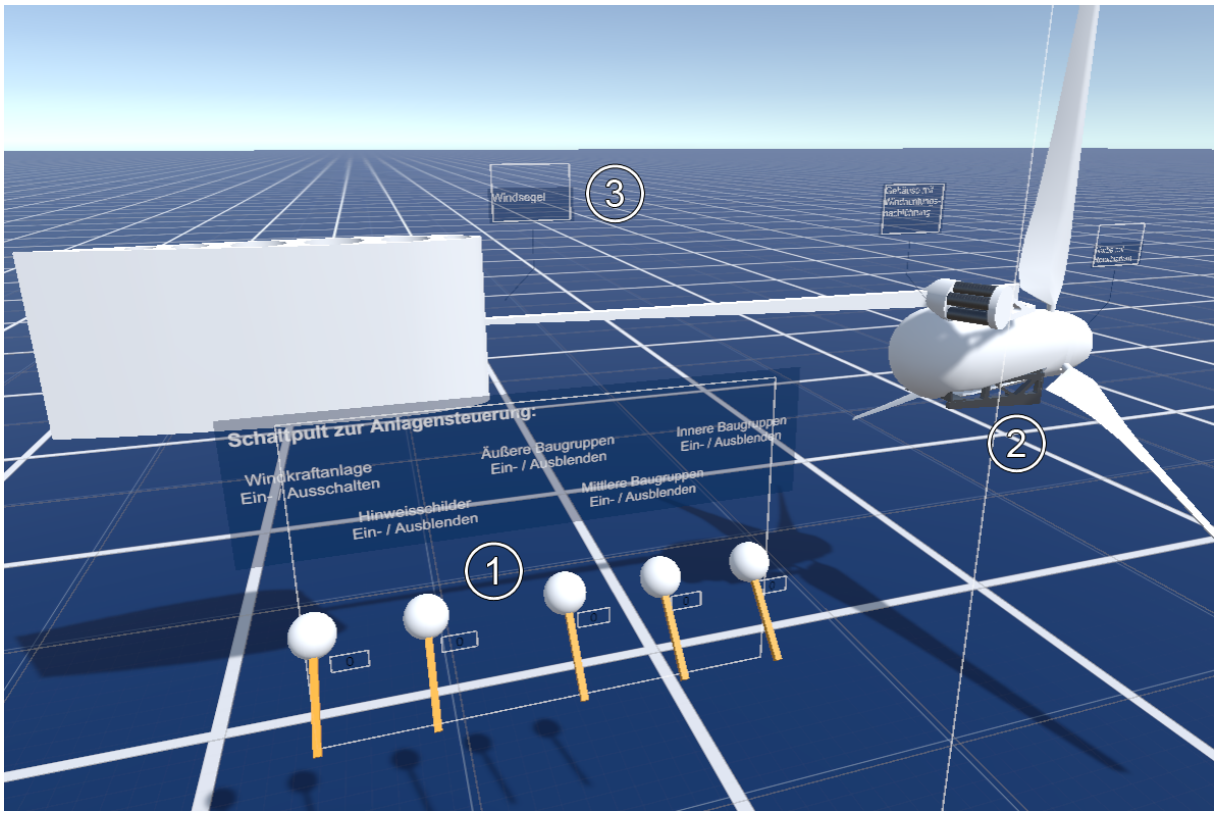

Abbildung 4.1: Szene in Unity.

Um die Szene zu strukturieren wurde die hierarchische Gruppierung vom VRTK-Framework übernommen. Diese der Szene untergeordneten GameObjects enthalten alle wichtigen Komponenten, welche für das Funktionieren der 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]:** Enthält obligatorische Skripte wie die Konfigurationsskripte für die Controllersteuerung.

> **[SceneScripts]:** Enthält optionale Skripte wie Teleporter, AvatarHandVisibility, Bodyphysics etc.

> **SceneObjects:** Enthält alle Szenenobjekte wie die Anlagensteuerung und die WEA.

4.3 Implementierte Anwendungsfälle

Im Folgenden werden die implementierten Anwendungsfälle erläutert und deren Funktionalität beschrieben.

Rotation der WEA: Die WEA kann mittels Trigger-Button rotiert werden, um diese von verschiedenen Seiten zu betrachten.

WEA Ein-/Ausschalten: Der Anwendungsfall WEA Ein-/Ausschalten beschreibt die Start-/Stopp-Animation der WEA. Per Default befindet sich der Rotor der WEA in Rotation, was die Idle-Animation ist. Bei Betätigung des Schalters wird eine Brems-Animation gestartet. Im bewegungslosen Zustand wird eine Situation ohne Wind oder ein Wartungsmodus simuliert. Während der Idle-Animation erzeugt die WEA Strom.

Hinweisschilder Ein-/Ausblenden: Dieser Anwendungsfall beschreibt das Ein-/Ausblenden der Hinweisschilder. Per Default sind die Hinweisschilder aktiviert. Die Funktionalität dient vor allem dazu die Übersicht zu verbessern, falls eine Betrachtung der Bauteile durch die Hinweisschilder erschwert wird.

Der Schalter gibt, je nach Winkel einen normalisierten Float-Wert zwischen 0,0 und 1,0 zurück. Dieser Wert wird an das Skalierungsattribut der Hinweisschilder übergeben und steuert so die Größe und dementsprechend die Sichtbarkeit.

Baugruppen Ein-/Ausblenden: Dieser Anwendungsfall bezieht sich auf mehrere Funktionalitäten, die das Ein- und Ausblenden von Bauteilen zur Folge haben. Dies dient dazu Innere Baugruppen der WEA sichtbar zu machen, die sonst durch äußere verdeckt sind.

Durch Betätigung des Schalters werden entsprechende Bauteile ausgefaded. Der Schalter übergibt einen normalisierten Float-Wert zwischen 0,0 und 1,0 an den Alpha-Kanal des Materials, welches dem auszublendenden Objekt zugewiesen ist. 1,0 entspricht dem Farbwert 255 (opak), 0,0 entspricht dem Farbwert 0 (volltransparent). Alle Werte zwischen 0,0 und 1,0 entsprechen Graustufen (halbtransparent). Auf diese Weise können Baugruppen ebenenweise zu- und abgeschaltet werden(siehe Abb. 4.2).

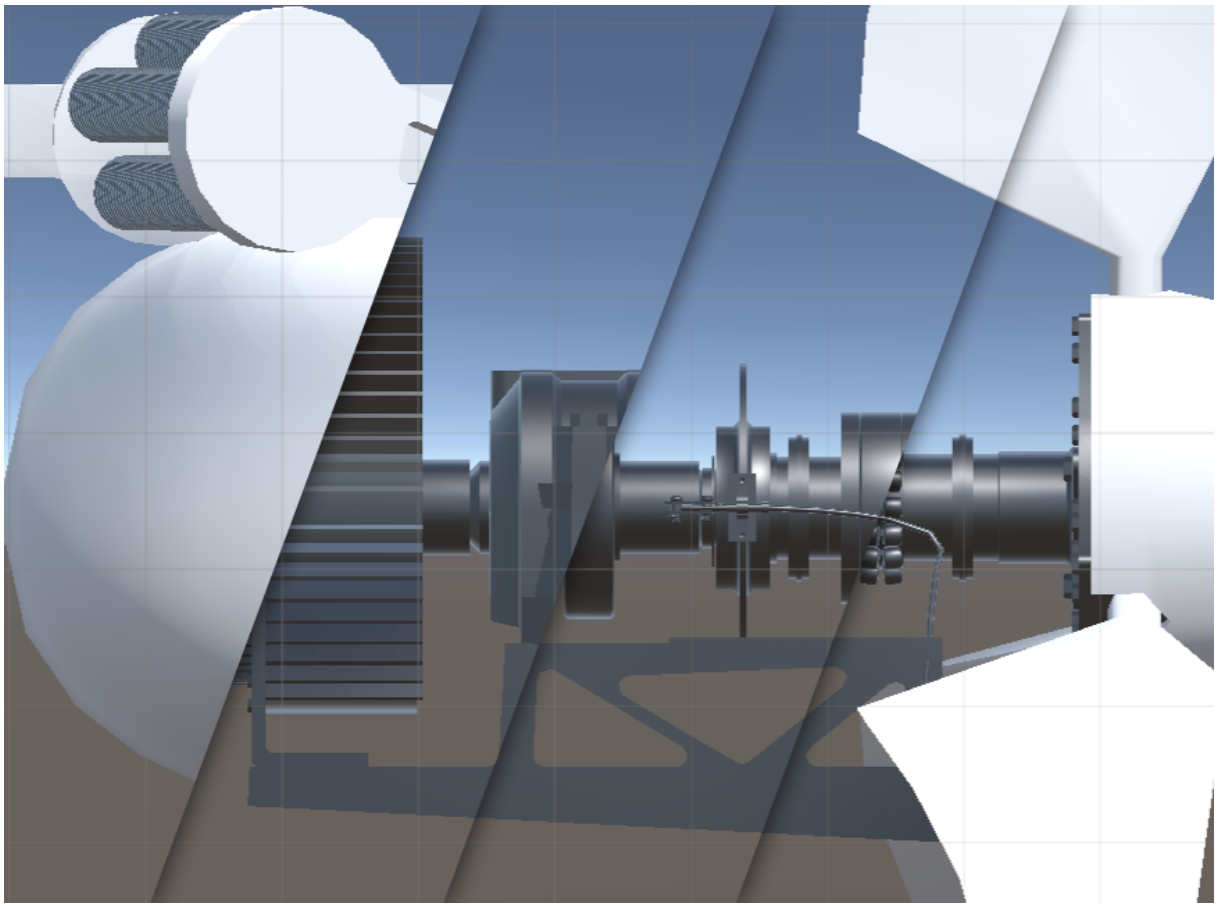

Abbildung 4.2: Zu- und abschaltbare Ebenen der WEA.

4.4 Anforderungsmanagement

Der Test der VR-Anwendung hat neue Anforderungen offengelegt, welche in weiteren Iterationen implementiert werden sollten.

Feedback von interaktiven Elementen: Um den VR-Nutzer auf Funktionalitäten hinzuweisen, die nicht über eine klassische UI verwendet werden, sollten interaktive Objekte zumindest bei Kollision mit einem Controller eingefärbt oder mit einer Outline gekennzeichnet sein. Das kann Frustration vermeiden, da der VR-Nutzer nicht jedes Objekt auf Interaktion testen muss.

Verschiebbare Kontrollelemente: In der aktuellen Implementierung ist die Steuerung der WEA über ein Schaltpult gelöst. Gerade in kleinen Räumen kann es durchaus Sinn machen das Kontrollpult zu verschieben oder zu drehen.

Mehrstufige Bedienelemente: In der aktuellen Implementierung lassen sich Baugruppen der WEA stufenweise ab- und zuschalten. Dies ist im Unterkapitel „4.3 Implementierte Anwendungsfälle“ (siehe Abb. 4.2) beschrieben. Dies geschieht mit drei separaten Schalthebeln, welche alle jeweils eine Ebene ab- oder zuschalten. Problematisch an dieser Umsetzung ist, dass verdeckte, nicht sichtbare Ebenen abgeschaltet werden können, es aber kein optisches Feedback gibt. Besser wäre eine Umsetzung mit einem mehrstufigen Schalter, der nicht nur ein binäres an / aus unterstützt, sondern einen Schaltzustand pro schaltbarer Ebene. Technisch könnte das ähnlich einer Combobox mit einer Switch-Anweisung gelöst werden.

Interaktion mittels Pointer: Zusätzlich zur Interaktion durch Contoller-Kollision sollte eine Interaktion mittels Laserpointer implementiert werden, um mit weiter entfernten Objekten zu interagieren.

Kapitel 5

Fazit

Abschließend lässt sich konstatieren, dass es mit einem gewissen Aufwand möglich ist ein CAD-Modell in eine VR-Anwendung zu überführen. Der größte Aufwand konzentrierte sich hauptsächlich auf die Aufbereitung des CAD-Modells. Die Arbeitszeit verteilte sich zu ca. 50% auf die Aufbereitung, zu ca. 25% auf die Implementierung in Unity und zu ca. 25% auf den Projektbericht. Natürlich richtet sich die Aufbereitungszeit nach der Komplexität des Modells. Das in diesem Projekt verwendete CAD-Modell der WEA zeichnete sich durch eine eher geringe Komplexität aus. CAD-Modelle können theoretisch eine vielfach höhere Komplexität annehmen. Ein vorher untersuchtes Modell einer „Blasformmaschine“, zur Herstellung von Hohlkörpern wie Kanister o.Ä. schlug bei ähnlichem Detailgrad mit einer Polygonanzahl von ca. 5.000.000 zu Buche. Das entspricht einer zehnfach höheren Komplexität als die der WEA, die nur ca. 500.000 Polygone aufweist. Die Zeit zur Aufbereitung wäre dementsprechend höher. Daher ist es durchaus angebracht automatisierte Verfahren zu verwenden. Dies kann z.B. über die neue Software PiXYZ geschehen, die auf Microsofts Entwicklerkonferenz „Build 2018“in Seattle vorgestellt wurde. PiXYZ wurde speziell für diesen Fall entwickelt.[18] Die Ergebnisse dieser Software können durchaus unter ähnlichen Kriterien in einem weiterführenden Forschungsprojekt untersucht werden. Das Video „1) Take CAD to mixed reality with Unity“ im Unity Blog (siehe Fußnote) macht bzgl. der Qualität des CAD-Imports einen sehr guten ersten Eindruck. Generell wäre es wünschenswert den Arbeitsschritt der Retopologisierung zu automatisieren, da es sich

[18] Vgl. Dustin Burg im Unity Blog (2018): *Unity at Microsoft Build: 7 takeaways from the show.* `https://blogs.unity3d.com/2018/05/11/unity-at-microsoft-build-7-takeaways-from-the-show/`, abgerufen am 12.09.2018.

um eine reine Fleißarbeit handelt, die wenig Kreativität erfordert. Zudem sind spezielle Kenntnisse in der Polygonmodellierung in Maya oder einem äquivalenten 3D-Programm erforderlich. Die freigewordene Arbeitszeit kann in anspruchsvollere und kreativere Tätigkeiten, wie der Entwicklung neuer Features in Unity investiert werden. Ferner können weitere Anwendungsfälle entwickelt und implementiert werden. In Verbindung mit weiteren Visualisierungsmethoden können alle möglichen Fakten vermittelt werden. Folgende Umsetzungen wären denkbar:

- **Erzeugte Leistung:** Im Betriebszustand kann anhand einer Leistungsanzeige, einer Glühlampe o.Ä. die erzeugte Energie visualisiert werden. Im Wartungszustand würde z.B. keine Energie erzeugt werden, was diese Anzeige verdeutlichen könnte.
- **Induktion nach dem Generatorprinzip:** Wie aus mechanischer Energie elektrische wird kann im Generator der WEA durch Darstellung von rotierenden Spulwicklungen in einem Magnetfeld sowie durch entsprechende sinusförmige Verläufe, ähnlich einem Oszillograph visualisiert werden.
- **Manipulation der Windrichtung:** Die Beeinflussung der Windrichtung und die korrekte Reaktion der Windrichtungsnachführung auf die Änderung kann visuell am animierten Modell dargestellt werden.
- **Darstellung von Strömungen:** Anhand von Partikelemittern kann das Strömungsverhalten der Rotorgeometrie bzw. Rotorblattgeometrie simuliert und evaluiert werden. Dies setzt natürlich die Implementierung eines physikalischen Modells aus dem Bereich der Strömungslehre voraus.

Dies sind nur vier mögliche Anwendungsfälle. Natürlich muss immer geprüft werden, was die Anforderungen an das System sind. So kann man sich die Frage stellen, ob der Aufwand der Implementierung eines physikalischen Modells aus der Strömungslehre tatsächlich Sinn macht. Schließlich sollte das Strömungsverhalten einer WEA oder bspw. eines Flugzeuges schon digital während der Konstruktionsphase oder anhand eines Kunstoff- oder Holzmodells in einem Windkanal getestet werden.

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): *Windpumpsysteme zur dezentralen Energieversorgung von Abwassersystemen*. TU Berlin.

[3] Prof. Dr. G. Buch, Prof. Dr. M. Krug (2012): *Kurzskriptum 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 Dateiformatenfür 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.

[16] Dustin Burg im Unity Blog (2018): *Unity at Microsoft Build: 7 takeaways from the show*.
`https://blogs.unity3d.com/2018/05/11/unity-at-microsoft-build-7-takeaways-from-t`
abgerufen am 12.09.2018.

Quelltextverzeichnis

Anhang

A Quellen

A1 *Windpumpsysteme zur dezentralen Energieversorgung von Abwassersystemen* [PDF]

A2 *Kurzskriptum 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]