HAW HAMBURG

MASTERARBEIT

Entwicklung eines modularen und modifizierbaren 3D-Modells in Unity zur Echtzeit-Kollaboration mit Kunden

Welche technischen und gestalterischen Möglichkeiten und Grenzen bestehen bei der Entwicklung eines modularen und modifizierbaren 3D-Modells in Unity und wie wirkt sich dies auf die Effizienz des Feedbackprozesses aus?

vorgelegt am 08. Mai 2025

Chiara Körting

Erstprüfer: Prof. Dr. Roland Greule
Zweitprüfer: Prof. Dr. Eike Langbehn

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Medientechnik
Finkenau 35
20081 Hamburg

Zusammenfassung

Diese Masterarbeit untersucht, ob sich in Unity, unter Verwendung des ProBuilder Packages, ein komplexes, modular aufgebautes und modifizierbares 3D-Modell erstellen lässt, das in der Echtzeit-Kollaboration mit Kunden eingesetzt und bewertet werden kann. Ziel ist es, herauszufinden, ob ein vollständig innerhalb von Unity umgesetzter Modellierungs- und Präsentationsprozess eine effizientere Gestaltung des Feedbackprozesses ermöglicht, da der Wechsel zwischen externem Modellierungsprogramm und Game Engine entfällt. Zu diesem Zweck wurde ein Pflanzenmodell entwickelt, das eine Vielzahl an Variationen zulässt und sich dynamisch anpassen lässt. Fachfremde und fachkundige Teilnehmende bewerteten das Modell, den Entwicklungsprozess und den Feedbackprozess im Rahmen einer qualitativen und quantitativen Studie. Die Ergebnisse zeigen, dass die Erstellung eines modularen und modifizierbaren Modells in Unity grundsätzlich möglich ist, jedoch einige technische Herausforderungen mit sich bringt. Der Feedbackprozess wurde – insbesondere aus Nutzersicht – als transparent und effizient wahrgenommen. Experten bewerteten ProBuilder eher als ergänzendes Werkzeug für prototypisches Arbeiten und nicht als Ersatz für professionelle DCC-Software. Die Arbeit zeigt auf, dass sich Unity unter bestimmten Voraussetzungen als Plattform für Modellierungs- und Feedbackprozesse eignet. Zukünftige Forschungsansätze könnten die Methode auf andere Modelltypen wie architektonische Szenarien übertragen oder prozedurale Erweiterungen zur Optimierung untersuchen.

Abstract

This master's thesis investigates whether a complex, modular and modifiable 3D model can be created in Unity using the ProBuilder package, which can be used and evaluated in real-time collaboration with customers. The aim is to find out whether a modeling and presentation process implemented entirely within Unity enables a more efficient design of the feedback process, as there is no need to switch between the external modeling program and the game engine. For this purpose, a plant model was developed that allows a large number of variations and can be dynamically adapted. Non-expert and expert participants evaluated the model, the development process and the feedback process as part of a qualitative and quantitative study. The results show that the creation of a modular and modifiable model in Unity is possible in principle, but involves some technical challenges. The feedback process was perceived as transparent and efficient, especially from the user's perspective. Experts rated ProBuilder more as a complementary tool for prototypical work and not as a replacement for professional DCC software. The work shows that Unity is suitable as a platform for modeling and feedback processes under certain conditions. Future research approaches could transfer the method to other model types such as architectural scenarios or investigate procedural extensions for optimization.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In kollaborativen Design- und Entwicklungsprozessen gehören Änderungswünsche seitens der Kunden, die einen Auftrag für ein konkretes Designelement wie beispielsweise ein 3D-Modell erteilen, ebenso wie die damit verbundenen Änderungszyklen zur Regel. Auch in der Gestaltung virtueller Umgebungen spielt der Feedbackprozess eine zentrale Rolle, da Inhalte häufig iterativ angepasst werden müssen. Jeder Änderungswunsch bringt dabei neue Anforderungen an die Umsetzung und die technologischen Abläufe mit sich. Insbesondere der Wechsel zwischen unterschiedlichen Programmen – etwa einem 3D-Modellierungswerkzeug und einer Game Engine – kann diesen Prozess verlangsamen und fehleranfällig machen.

Das zentrale Thema dieser Masterarbeit ist daher die Entwicklung eines komplexen, modularen und modifizierbaren 3D-Modells direkt in der Game Engine Unity – mit dem Ziel, den Feedbackprozess mit Kunden effizienter zu gestalten. In der Praxis erfolgt die Erstellung und Anpassung virtueller Objekte häufig durch das Zusammenspiel verschiedener Programme. Während die Modellierung in spezialisierten Digital Content Creation (DCC) Tools wie Blender stattfindet, wird die virtuelle Umgebung anschließend in Unity zusammengesetzt. Dieser Wechsel zwischen mehreren Softwarelösungen kann den Austausch mit Kunden verlangsamen und Änderungszyklen verlängern. Die Arbeit geht daher der Frage nach, ob durch die ausschließliche Nutzung von Unity sowohl ein technisch anspruchsvolles Modell realisiert als auch der Feedbackprozess vereinfacht werden kann – und wie sich diese Arbeitsweise auf dessen Wahrnehmung durch potenzielle Kunden und Experten auswirkt.

Zur Erreichung dieses Ziels wurde ein mehrstufiges methodisches Vorgehen gewählt. Im praktischen Teil wurde zunächst ein modular aufgebautes und modifizierbares 3D-Modell in Unity mit dem integrierten Unity Package ProBuilder entwickelt. In einer anschließenden Studie wurde dieses Modell zehn Probanden vorgestellt. Sie bewerteten das Modell, den damit verbundenen Feedbackprozess sowie den Entwicklungsprozess hinsichtlich Verständlichkeit, Effizienz und Praxistauglichkeit. Aufbauend auf dem Feedback wurde das Modell gezielt weiterentwickelt, um Optimierungsmöglichkeiten zu erproben und die praktische Umsetzbarkeit von Änderungswünschen zu testen.

Der Aufbau dieser Arbeit folgt einer klaren Struktur: Nach einer historischen Einordnung von 2D- und 3D-Visualisierungstechnologien wird der aktuelle Forschungsstand dargestellt und bestehende Lösungsansätze erläutert. Anschließend wird die Forschungsfrage präzisiert und durch Hypothesen ergänzt, die im weiteren Verlauf überprüft werden. Darauf folgt das Methodikkapitel, das die praktische Umsetzung sowie das Studiendesign beschreibt. Im Anschluss findet eine detaillierte Auswertung und Interpretation der erhobenen Ergebnisse statt. Abschließend werden die gewonnenen Erkenntnisse im Fazit zusammengeführt und in einen größeren Kontext eingeordnet.

2 Theoretische Grundlagen

Die visuelle Darstellung von Ideen und Konzepten ist seit jeher ein essenzieller Bestandteil menschlicher Kommunikation und Kreativität. 2D- und 3D-Visualisierungs- und Modellierungstechniken haben sich seit Mitte des 20. Jahrhunderts stets weiterentwickelt und leisten einen zentralen Beitrag zur heutigen Medienlandschaft. Dieses Kapitel gibt einen Überblick über die historische Entwicklung dieser Techniken, beleuchtet die wegweisenden Meilensteine und zeigt auf, wie technologische Innovationen die Art und Weise geprägt haben, wie wir heute virtuelle Welten erschaffen und erleben. Ziel ist es, den Übergang von traditionellen 2D-Ansätzen zu komplexen 3D- und Virtual Reality Technologien (VR) nachzuvollziehen und eine ausreichende Grundlage für den praktischen Teil dieser Arbeit zu schaffen.

2.1 Entwicklung von 2D- und 3D-Modellierungstechniken

Die Entwicklung der 2D- und 3D-Modellierungstechniken spielt eine zentrale Rolle in der Geschichte der Computergrafik. Was heute als Standardwerkzeug für die Erstellung von VR-Welten und Animationen gilt, nahm seine Anfänge in den experimentellen Projekten der frühen interaktiven Computergrafiksysteme. Dieses Kapitel beleuchtet die historische Evolution dieser Technologien und gibt einen Einblick in die verschiedenen Phasen, die die Modellierung von einer experimentellen Disziplin zu einer der vielseitigsten Anwendungen moderner Computertechnik gemacht haben.

2.1.1 Erste interaktive Computersysteme

Die Ursprünge der modernen Computergrafik und -modellierung lassen sich bis in die Mitte des 20. Jahrhunderts zurückverfolgen, als die Grundlagen für interaktive Systeme und visuelle Darstellungen geschaffen wurden. Ein Meilenstein in dieser Entwicklung war das Whirlwind-Projekt, das 1945 am Massachusetts Institute of Technology (MIT) unter der Leitung von Jay Forrester ins Leben gerufen wurde. Ursprünglich als Flugsimulationssystem für die United States Navy konzipiert, demonstrierte Whirlwind 1951 erstmals die Fähigkeit, Texte und Grafiken in Echtzeit auf einem Oszilloskop-Bildschirm darzustellen. Dieses System war das erste, das grafische Informationen in Echtzeit verarbeitete. Es konnte die Position eines Flugzeugs auf der Grundlage von Radardaten anzeigen, wobei eine einfache grafische Darstellung der Ostküste von Massachusetts als Hintergrund diente.

Ein entscheidender Fortschritt bei Whirlwind war die Einführung eines innovativen Eingabegeräts, des sogenannten Lightpen (dt. „Lichtstift“), durch Robert Everett. Dieses Gerät ermöglichte es, Informationen zu spezifischen Objekten auf dem Bildschirm abzufragen, indem es auf ein Symbol gerichtet wurde. Das System reagierte darauf mit der Anzeige zusätzlicher Daten wie Flugzeugidentifikation, Geschwindigkeit oder Flugrichtung und war somit eine frühe Form interaktiver Benutzeroberflächen.

Der Fortschritt in der Computergrafik setzte sich mit dem TX-2-Computer fort, der 1959 ebenfalls am MIT entwickelt wurde. Dieses System markierte eine weitere wichtige Etappe, insbesondere durch die

Verwendung von innovativer Transistortechnologie. Der TX-2 verfügte über einen 320 Kilobyte schnellen Speicher, was damals eine doppelt so große Speicherkapazität darstellte wie die der größten kommerziellen Computer. Zudem bestand er aus einem 9-Zoll-Kathodenstrahlröhren-Monitor und einem Lightpen, der als Eingabegerät diente.[1, 2]

2.1.2 Sketchpad

Inspiriert durch die Möglichkeiten des Lightpen und der Kathodenstrahlröhre des TX-2-Computers entwickelte Ivan Sutherland, ein amerikanischer Elektroingenieur und Informatiker, in den frühen 1960er Jahren im Rahmen seiner Doktorarbeit am MIT das bahnbrechende Programm Sketchpad, das als eines der ersten Systeme interaktiver Computergrafik galt. Dabei hatte er die Idee, dass Computer nicht nur für Berechnungen, sondern auch als Zeichenwerkzeuge genutzt werden könnten. Mit der Entwicklung von Sketchpad legte er den Grundstein für viele Technologien, die heute in der Computergrafik und in Computer-Aided Design (CAD) Systemen selbstverständlich sind.[3]

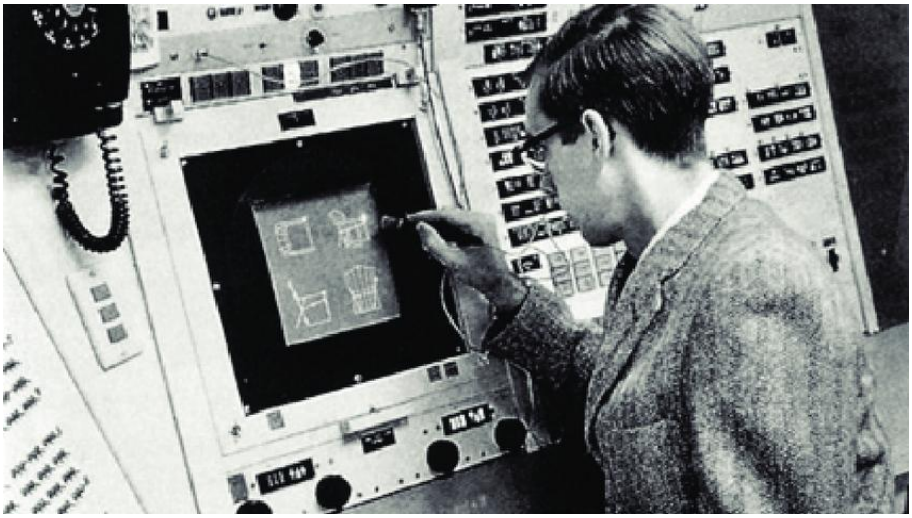


Abb. 2.1: Ivan Sutherlands Sketchpad. (Di Marco, 2019)

Sutherland beschreibt das Sketchpad und dessen Funktionen in seiner Arbeit *Sketchpad: A Man-Machine Graphical Communication System* (1963) als ein System, das die Interaktion zwischen Mensch und Computer revolutionieren sollte. Anstatt wie bis dahin üblich jede Eingabe zu tippen, ermöglichte Sketchpad erstmals das direkte Zeichnen auf einem Bildschirm. Der Lightpen diente dabei als Eingabegerät, mit dem Koordinatenpunkte in Echtzeit übermittelt wurden. So konnten Benutzer Linien, Kreise, Polygone und andere geometrische Formen erstellen. Unterschiedliche Befehle und Tasten ermöglichten es, die Formen präzise zu manipulieren, etwa durch Verschieben, Skalieren oder Duplizieren. Besonders innovativ war die Möglichkeit, Zeichnungen auf Magnetband zu speichern, um sie später erneut

[1] Carlson, W. E. (2017). *Computer Graphics and Computer Animation: A Retrospective Overview*.
[2] Jackèl, D., Neunreither, S. & Wagner, F. (2006). Methoden der Computeranimation. In *eXamen.press*. https://doi.org/10.1007/3-540-33407-6
[3] Pyfer & James. (2014, 10. Juni). *SketchPad | Interactive Drawing, Vector Graphics & CAD*. Encyclopedia Britannica. Abgerufen am 19. März 2025, von https://www.britannica.com/technology/Sketchpad

aufzurufen und weiterzuverwenden. Grundelemente, wie beispielsweise Hexagone, konnten als wiederverwendbare Symbole gespeichert werden. Diese Symbole ließen sich in mehreren Instanzen einsetzen und individuell anpassen, was eine hierarchische Darstellung komplexer Strukturen ermöglichte. Sutherland integrierte zudem rekursive Funktionen in Sketchpad, die es beispielsweise erlaubten, abhängige Elemente automatisch zu löschen, wenn ein Hauptobjekt entfernt wurde, oder ähnliche Objekte zu einer Struktur zusammenzuführen. Darüber hinaus weist Sutherland in seiner Arbeit auf die grundlegende Erweiterbarkeit von Sketchpad hin und beschreibt es als System, das nicht nur für zweidimensionale Zeichnungen, sondern auch für dreidimensionale Anwendungen zukunftsweisend ist. Die Fähigkeit, Zeichnungen hierarchisch zu organisieren und miteinander zu verknüpfen, eröffnete neue Perspektiven für die Visualisierung und Modellierung komplexer Strukturen.[4]

Die Bedeutung von Sketchpad reichte weit über seine technischen Errungenschaften hinaus. Es markierte den Beginn der interaktiven Computergrafik und legte die Grundlage für moderne CAD-Programme und grafische Benutzeroberflächen. Für seine visionäre Arbeit wurde Ivan Sutherland später mit dem Turing Award und dem Kyoto-Preis ausgezeichnet.[5]

2.1.3 Entwicklung der Computergrafik in den 1960er und 1970er Jahren

In den darauffolgenden Jahren beschäftige Sutherland sich weiterhin mit dem Thema der interaktiven Computergrafik. Im Jahr 1968 gründete er gemeinsam mit David Evans die Firma Evans and Sutherland (E&S). Beide waren zu dieser Zeit Informatikprofessoren an der University of Utah und verfolgten das Ziel, Computervisualisierungstools so weiterzuentwickeln, dass sie die Simulation von Objekten und Umgebungen ermöglichten. Das Unternehmen etablierte seinen Hauptsitz im Forschungspark der Universität und entwickelte in den folgenden Jahren mehrere wegweisende Technologien. 1969 präsentierte E&S mit den Linienzeichnungssystem-Displays LDS-1 und LDS-2 die ersten Grafikgeräte, die über eine eigene Verarbeitungseinheit verfügten. Diese wurden später durch das E&S Picture System abgelöst, das als nächste Generation der LDS-Displays bis in die 1980er Jahre in der Produktion computergenerierter Bilder zum Einsatz kam. Neben diesen sogenannten Workstations spezialisierte sich das Unternehmen auch auf computergestützte Simulationssysteme. Besonders hervorzuheben sind die Flugsimulatoren CT5 und CT6, die sowohl im militärischen als auch im kommerziellen Bereich eingesetzt wurden. Viele der späteren Pioniere der Computergrafik begannen ihre Karriere bei E&S oder arbeiteten dort während ihrer Doktorandenzeit. Zu diesen zählen unter anderem Edwin Catmull, Mitbegründer von Pixar, John Warnock, Mitbegründer von Adobe, und Jim Clark, Gründer von Netscape und Silicon Graphics.[6]

---

[4] Sutherland, I. E. (1964). Sketchpad a Man-Machine Graphical Communication System. *SIMULATION*, *2*(5), R-20. https://doi.org/10.1177/003754976400200514

[5] Sharma, M. (2023, 14. November). Evolution of Computer Graphics - Mitesh Sharma - Medium. *Medium*. https://medium.com/@miteshryp/evolution-of-computer-graphics-b970be31d2c9

[6] W. E. Carlson (2017, S. 90 ff.)

Parallel zu den technologischen Fortschritten von Evans und Sutherland nutzte der Kunstprofessor Charles Csuri an der Ohio State University (OSU) seit 1963 Computertechnologie, um Kunst und Wissenschaft zu verbinden. Mit einer Gruppe von Fakultätsmitgliedern untersuchte Csuri die Möglichkeiten der computergestützten Kunst und experimentierte mit frühen Animationstechniken. 1967 schuf er mit Hilfe eines IBM-360-Großrechners das Werk Hummingbird, eine experimentelle computeranimierte Sequenz, die als Meilenstein der digitalen Kunst gilt.

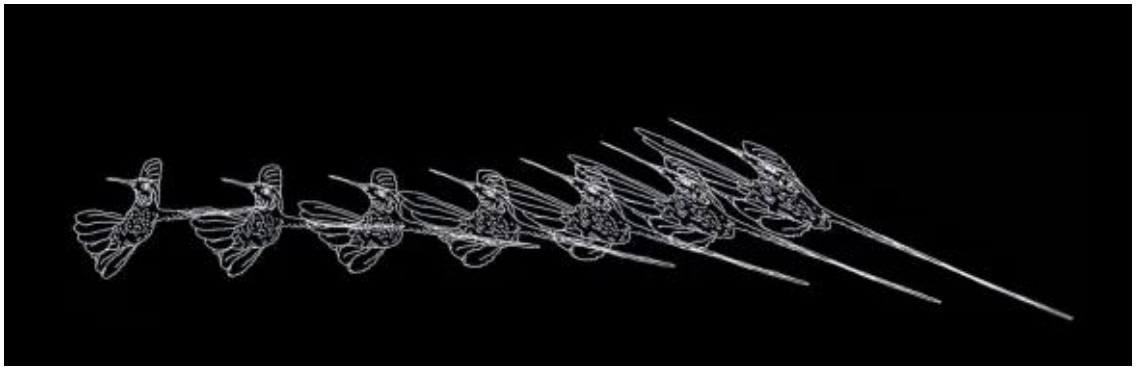


Abb. 2.2: Hummingbird von Charles Csuri, das als erste Animation der Welt gilt. (Sharma, 2023)

1971 gründete Csuri die Computer Graphics Research Group (CGRG), um die Forschung auf diesem Gebiet zu institutionalisieren und Fördermittel zu sichern. In den folgenden Jahren präsentierte die Gruppe zahlreiche Innovationen, darunter das von Tom DeFanti entwickelte Graphics Symbiosis System (GRASS), das Echtzeit-Animation und interaktive Manipulationen ermöglichte. Die von der CGRG entwickelten Technologien umfassten primitive Algorithmen, Grafik-Pipeline-Architekturen, spezialisierte Skriptsprachen für 3D-Grafikrendering, Techniken für Keyframe-Animationen und geometrische Modellierung sowie Echtzeit-Animationstechniken und Oberflächenalgorithmen. Diese Werkzeuge legten den Grundstein für viele moderne Anwendungen der Computergrafik und ermöglichten es, realistische Animationen, Modellierungen und Effekte zu erstellen.[7, 8]

2.1.4 Entwicklung der Computergrafik ab den 1980er Jahren

Mit dem Fortschritt der Technologie und der zunehmenden Verbreitung von Computern veränderte sich die Landschaft der Computergrafik ab den 1980er Jahren grundlegend. Neue Software, Studios und Geschäftsmodelle führten zu revolutionären Entwicklungen in der Bildbearbeitung, Animation und 3D-Modellierung.

Adobe

Die Gründung von Adobe Systems im Jahr 1982 leitete eine neue Ära der Kreativtechnologie ein, wobei die Vorstellung von Adobe Photoshop im Jahr 1988 ein Meilenstein war. Photoshop, entwickelt von den

[7] Carlson, W., Hackathorn, R. & Parent, R. (2021). Computer Graphics and Animation at The Ohio State University. *IEEE Computer Graphics And Applications*, *41*(3), 8–17. https://doi.org/10.1109/mcg.2021.3070624
[8] Sharma (2023)

Brüdern Thomas und John Knoll, wurde ursprünglich als Softwarepaket zusammen mit Scanner-Hardware vertrieben. Aufgrund seiner leistungsstarken Bildbearbeitungsfunktionen etablierte es sich schnell als Standardwerkzeug für Fotografen, Grafikdesigner und digitale Künstler.

Mit der Einführung der Creative Suite im Jahr 2003 bot Adobe ein Bündel seiner wichtigsten Produkte – darunter Photoshop, Illustrator und InDesign – in einem einzigen Paket an. Die Creative Suite, die bis 2012 weiterentwickelt wurde, ermöglichte es Kreativprofis weltweit, effizienter zu arbeiten und Projekte nahtlos in verschiedenen Anwendungen zu verwenden.[9]

Pixar

Die Pixar Animation Studios, ursprünglich eine Abteilung von Lucasfilm, entwickelte in den 1980er Jahren Softwaretechnologien, die die Animation revolutionierten. Zu den wichtigsten Innovationen gehörte der REYES-Renderer („Renders Everything You Ever Saw"), der die Grundlage für die spätere RenderMan-Technologie legte. Weitere Entwicklungen umfassten CAPS, ein Animationssystem für Disney, Marionette, eine Software zur Modellierung und Animation von Figuren sowie Ringmaster, ein Werkzeug für das Produktionsmanagement. 1989 wurde RenderMan veröffentlicht, ein Produkt, das bis heute als Industriestandard für die Erstellung realistischer 3D-Bilder gilt.

Den wichtigsten Meilenstein setzte Pixar jedoch 1995 mit der Veröffentlichung von Toy Story, dem ersten vollständig computeranimierten Spielfilm. Der Film war ein kommerzieller Erfolg und markierte den Beginn einer neuen Ära in der Filmindustrie.[10]

Blender

Die Geschichte von Blender, einer der führenden Open-Source-3D-Softwarelösungen, begann 1989, als Ton Roosendaal das Animationsstudio NeoGeo in den Niederlanden gründete. Nachdem Blender 1994 ursprünglich als interne Anwendung für NeoGeo entwickelt wurde, erschien 1995 die erste öffentliche Version, Blender 1.0. Nach der Schließung von NeoGeo gründete Roosendaal 1998 das Unternehmen Not a Number (NaN), um Blender kommerziell zu vermarkten. Da das Unternehmen 2002 in finanzielle Schwierigkeiten geriet, etablierte er 2002 die gemeinnützige Blender Foundation, mit der er über eine Crowdfunding-Kampagne das nötige Budget sammelte, um Blender zu einer Open Source Software zu machen.

Seitdem wird Blender von einer globalen Gemeinschaft von Entwicklern kontinuierlich verbessert und hat sich zu einer der leistungsstärksten und flexibelsten Plattformen für 3D-Animation, Modellierung und Rendering entwickelt.[11]

[9] Weston, Z. (2023, 2. Oktober). Title: The Evolution of Adobe: A Journey Through History. *Medium*. https://medium.com/@zacharywestonintech/title-the-evolution-of-adobe-a-journey-through-history-21b1c029174

[10] W. E. Carlson (2017, S. 324 ff.)

[11] Blender Foundation. (o. D.). *History — Blender.org*. blender.org. Abgerufen am 1. Januar 2025, von https://www.blender.org/about/history/

2.2 Entwicklung von 2D- & 3D Visualisierungstechniken

Die Visualisierung von Daten und Szenen in zwei und drei Dimensionen spielt eine Schlüsselrolle in der Darstellung immersiver Erlebnisse. Die Entwicklung begann mit einfachen 2D-Visualisierungen und erweiterte sich hin zu beeindruckenden 360°-Ansichten und virtuellen Realitäten. Dieses Kapitel beschreibt die Entwicklung der Visualisierungstechniken und beleuchtet die wichtigsten Meilensteine.

2.2.1 2D-Visualisierungen

Um die Ursprünge moderner VR-Technologie zu verstehen, ist es unverzichtbar, die Entwicklung der 2D-Visualisierungstechniken nachzuvollziehen. Da diese Methoden die Grundlage für viele der Innovationen bildeten, die später zur Entstehung dreidimensionaler und immersiver Technologien führten. Bis zum Ende des 19. Jahrhunderts beschränkten sich visuelle Darstellungen hauptsächlich auf Kunstwerke und Fotografien, die als einziges Mittel zur Abbildung von Menschen, Orten oder Objekten dienten. Künstler experimentierten dabei mit verschiedenen Ansätzen, darunter großformatige Wandbilder, Panoramen und sogar frühe stereoskopische Bilder.

Ein Schlüsselmoment in der Geschichte der Visualisierung war die Erfindung des Panoramabildes durch den Maler Robert Barker im Jahr 1787. Er präsentierte Rundumsichten Edinburghs und Londons, die er nach den griechischen Begriffen „pan“ (dt. „alles“) und „horama“ (dt. „Sicht“) benannte. Barker galt somit zwar als Begründer der 360°-Darstellung, das erste Patent für Panoramen wurde jedoch 1791 dem amerikanischen Ingenieur Robert Fulton zugesprochen. Dieses Patent umfasste nicht nur das kreisförmige Bild selbst, sondern auch die baulichen und organisatorischen Elemente wie die Architektur und die Bewegung der Zuschauer durch den Ausstellungsraum.[12]

Abb. 2.3: Panorama von Robert Baker. (Government Art Collection, 2025)

Ein weiterer wichtiger Meilenstein war die Entdeckung des Phänomens der Persistence of Vision durch den englischen Arzt Peter Mark Roget im Jahr 1824. Dies beschrieb die Trägheit des Auges, die das menschliche Gehirn dazu bringt, einzelne Bilder zu einer Bewegung zu verbinden und bildete die Grundlage für zahlreiche Entwicklungen im Bereich der Bewegtbildtechnik. Zu den frühen Erfindungen zählte hierbei das Thaumatrop, das 1825 von William Henry Fitton entwickelt wurde. Dieses einfache

[12] Wölfel, M. (2023, S. 32). *Immersive virtuelle Realität*. Springer Vieweg.

Instrument bestand aus einer Scheibe mit zwei unterschiedlichen Bildern auf Vorder- und Rückseite, die durch schnelle Rotation zu einem zusammenhängenden Bild verschmolzen.[13]

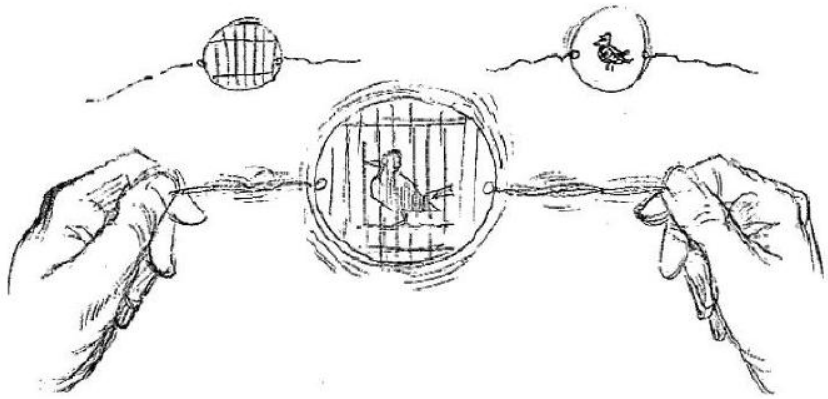


Abb. 2.4: Darstellung eines Thaumatrops. (Williams, 2002, S. 13)

Eine weitere bedeutende Innovation war das Phenakistiskop, das 1832 von Joseph Plateau in Frankreich und Simon Stampfer in Österreich nahezu zeitgleich erfunden wurde. Dieses sogenannte Lebensrad nutzte den Stroboskopeffekt, bei dem abwechselnde Licht- und Dunkelphasen die Wahrnehmung von Bewegung erzeugen. Das Gerät bestand aus zwei Scheiben: Eine zeigte eine Abfolge von Bildern, die eine Szene darstellten, während die andere mit Schlitzen versehen war, durch die die Bilder intermittierend sichtbar wurden. Diese Technik ermöglichte es, bewegte Szenen realistisch darzustellen.[14]

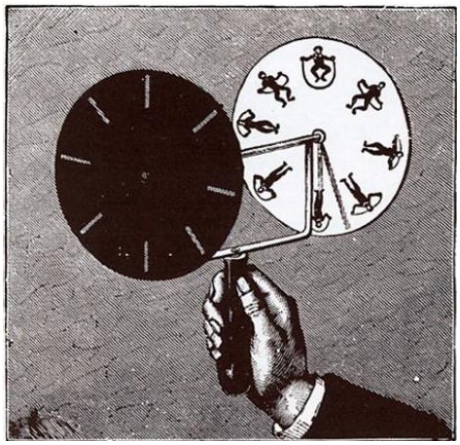


Abb. 2.5: Darstellung eines Phenakistiskops. (media storehouse., 2025)

Eine Weiterentwicklung des Phenakistiskops stellte das Zoetrop dar, das 1834 von William George Horner konzipiert wurde. Dieses Gerät nutzte ein bemaltes Band, das in einer rotierenden Trommel mit Schlitzen platziert wurde. Durch die Drehung und den Wechsel zwischen Bildern und dunklen Streifen entstand die Illusion von Bewegung. In den 1860er Jahren wurde das Zoetrop als Spielzeug populär und inspirierte weitere Erfinder wie den Franzosen Emile Reynaud, der 1877 das Praxinoskop vorstellte.

[13] Williams, R. (2002). *The animator's survival kit: A Manual of Methods, Principles and Formulas for Classical, Computer, Games, Stop Motion and Internet Animators*. Faber & Faber. (S. 13 ff.)

[14] Schmidt, U. (2013). *Professionelle Videotechnik: Grundlagen, Filmtechnik, Fernsehtechnik, Geräte- und Studiotechnik in SD, HD, DI, 3D*. Springer Vieweg. (S. 2 f.)

Durch ein Spiegelprisma optimierte Reynaud die Lichtausbeute und ermöglichte es später, Bilder auf größere Flächen zu projizieren. Obwohl das Praxinoskop technisch überlegen war, blieb es aufgrund seiner Empfindlichkeit und der komplizierten Handhabung begrenzt einsetzbar.[15, 16]

Abb. 2.6: Zoetrop. (Trickbüro Basil Vogt, o. D.)

Alle aufgeführten Entwicklungen im Bereich der 2D-Visualisierung legten den Grundstein für die Erforschung von Bewegtbildern und deren Transformation in immersive, dreidimensionale Welten, wie sie in der heutigen Virtual Reality genutzt werden.

2.2.2 Die Anfänge der 360°- und VR-Technik

Im Jahr 1838 präsentierte Sir Charles Wheatstone, Professor für experimentelle Naturwissenschaften am King's College in London, seine Erkenntnisse über das räumliche Sehen. Er stellte fest, dass das menschliche Gehirn in der Lage ist, zwei Bilder desselben Objekts – aufgenommen aus leicht unterschiedlichen Blickwinkeln und jeweils einem Auge zugeordnet – zu einem dreidimensionalen Bild zu kombinieren. Auf dieser Grundlage entwickelte Wheatstone einen Apparat, der aus zwei Spiegeln bestand, die in einem 45°-Winkel zueinander angeordnet waren. Dieses System lenkte die Bildpaare so um, dass jedes Auge nur das für es vorgesehene Bild sah, wodurch ein Tiefeneindruck erzeugt wurde. Zudem erlaubte der mechanische Aufbau verschiedene Anpassungen, um die Wahrnehmung weiter zu optimieren.[17]

[15] Williams (2002, S. 13 ff.)
[16] Lenk, S. (o. D.). *Zoetrope [Das Lexikon der Filmbegriffe]*. Abgerufen am 7. Dezember 2024, von https://film-lexikon.uni-kiel.de/doku.php/z:zoetrope-852
[17] Wölfel (2023, S. 32)

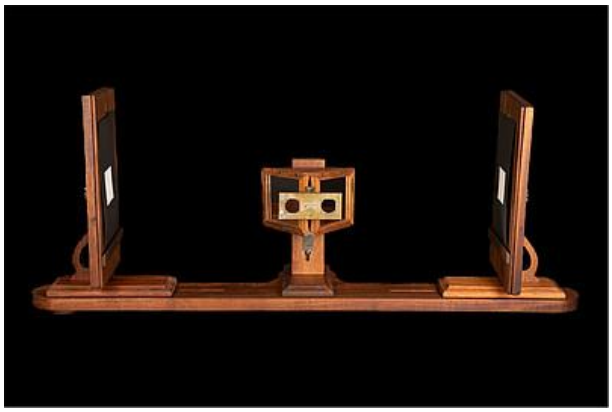


Abb. 2.7: Reflektierendes Stereoskop nach Charles Wheatstone. (Europa-Universität Flensburg, 2023)

Im Jahr 1849 machte der Physiker David Brewster einen weiteren bedeutenden Fortschritt, indem er die erste Zweiobjektiv-Kamera vorstellte. Diese ermöglichte es, stereoskopische Aufnahmen simultan anzufertigen. Bis dahin mussten Stereo-Halbbilder nacheinander aufgenommen werden, wobei die Kamera zwischen den Aufnahmen um den Augenabstand verschoben wurde. Zeitgleich präsentierte Brewster das Prismen-Stereoskop, ein Betrachtungsgerät, das es Nutzern erleichterte, die mit seiner Kamera erstellten Stereoaufnahmen zu betrachten. Die von Brewster entwickelte Technologie fand schnell Verbreitung und die Begeisterung für stereoskopische Bilder hielt bis zur Jahrhundertwende an.

Zu Beginn des 20. Jahrhunderts wurden auf der Pariser Weltausstellung zwei visionäre Attraktionen vorgestellt: das Maréorama und das Cinéorama. Der Künstler Hugo d'Alesi kombinierte beim Maréorama bewegliche Panoramagemälde mit einer großen Plattform, um den Eindruck einer Schiffsfahrt zu erzeugen. Raoul Grimoin-Sanson verwendete beim Cinéorama hingegen zehn synchronisierte 70-mm-Filmprojektoren, um die Illusion einer Heißluftballonfahrt zu schaffen. Beide Projekte verdeutlichten, wie weit die Technik der immersiven Visualisierung bis zu diesem Zeitpunkt bereits vorangeschritten war.[18, 19]

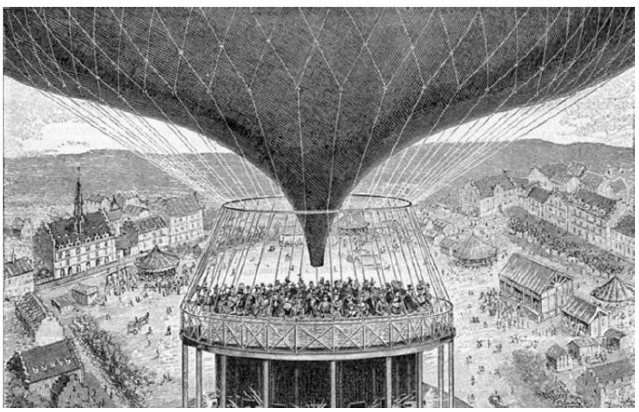


Abb. 2.8: Das Cinéorama auf der Pariser Weltausstellung im Jahr 1900. (Wölfel, 2023, S. 35)

[18] Wölfel (2023, S. 34)

[19] *Who's Who of Victorian Cinema*. (2025). Abgerufen am 23. April 2025, von https://www.victorian-cinema.net/grimoinsanson.php

2.2.3 VR-Technik in den 50er und 60er Jahren und „The Ultimate Display“

Einen bedeutenden Schritt hin zur virtuellen Realität machte der Kameramann Morton Heilig im Jahr 1956 mit seiner Erfindung des Sensorama. Diese Maschine, die er 1962 patentieren ließ, zielte darauf ab, alle menschlichen Sinne anzusprechen und den Betrachter vollkommen in das Erlebnis eines Films eintauchen zu lassen. Durch die Kombination eines stereoskopischen 3D-Bildschirms, Stereolautsprechern, einem vibrierenden Sitz, Lüftern sowie Geruchsdispersion konnte das Sensorama eine beeindruckende multisensorische Erfahrung bieten.

Abb. 2.9: Sensorama von Morton Heilig. (ResearchGate, 2018)

Neben dem Sensorama konzipierte Heilig auch die sogenannte Telesphere Mask, einen weiteren innovativen Schritt in Richtung VR-Technik. Diese Vorrichtung kann als erstes HMD betrachtet werden, da sie stereoskopische Bilder mit räumlicher Tiefe und dazu passenden Stereoklang lieferte. Allerdings fehlte der Telesphere Mask noch die Fähigkeit zur Bewegungsverfolgung, die für moderne VR-Systeme zentral ist.

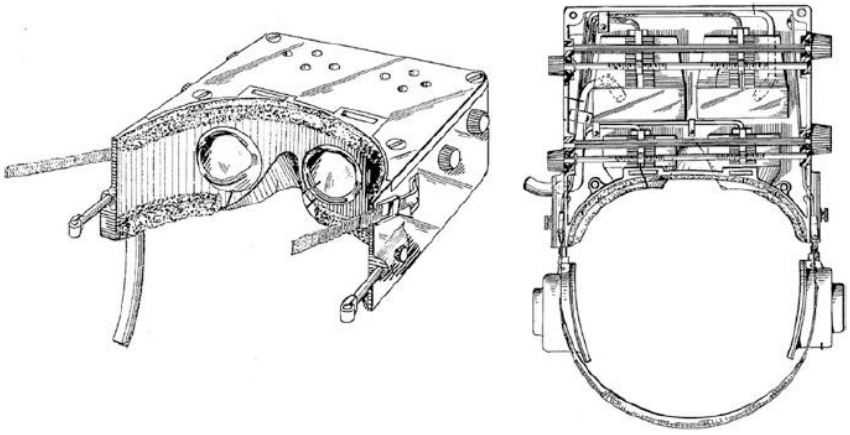


Abb. 2.10: Telesphere Mask von Morton Heilig. (Wölfel, 2023, S. 37)

Ein weiterer Meilenstein war die Entwicklung des Headsight durch Charles Comeau und James Bryan im Jahr 1961. Die beiden Ingenieure der Philco Corporation schufen das erste HMD, das über ein Bewegungserfassungssystem (Head Tracking) verfügte. Das Headsight war mit Videobildschirmen für jedes Auge ausgestattet und ermöglichte es, die Kopfbewegungen des Nutzers auf eine Kamera zu übertragen, die diese Bewegungen nachahmte. Im Gegensatz zu modernen VR-Systemen projizierte das Headsight keine virtuellen Inhalte, sondern erlaubte es lediglich, die Umgebung durch die ferngesteuerte Kamera aus unterschiedlichen Perspektiven zu betrachten. [20]

Abb. 2.11: Headsight. (Wölfel, 2023, S. 37)

Ein zukunftsweisender Beitrag zur VR-Entwicklung folgte 1965 durch Ivan Sutherland, der mit seinem Konzept des Ultimate Display die theoretischen Grundlagen für immersive virtuelle Umgebungen legte. Sutherland stellte sich ein Display vor, das den Betrachter in eine vollständig durch Computer generierte Welt versetzen konnte, die möglichst viele Sinne ansprach. Dabei erkannte er, dass vor allem visuelle, auditive und haptische Eingaben realisierbar waren, während der Einbezug von Geruch und Geschmack technisch noch schwierig umzusetzen war. Sutherlands Vision beschrieb ein System, das die manuelle Steuerung und kinästhetische Rückmeldungen kombinieren konnte. Ein anschauliches Beispiel lieferte er mit der Vorstellung eines Computermodells von Teilchen in einem elektrischen Feld. Ein Nutzer konnte hier die Position eines geladenen Teilchens steuern und dabei nicht nur die visuelle Darstellung, sondern auch die fühlbaren Kräfte erleben, die auf das Teilchen wirkten. Besonders bemerkenswert war Sutherlands Idee, dass der Computer Objekte unabhängig von physikalischen Regeln darstellen könnte. Sutherland skizzierte eine Welt, in der virtuelle Objekte real wirkten: Ein angezeigter Stuhl könnte stabil genug sein, um darauf zu sitzen, während Handschellen oder sogar Geschosse realistische physische

[20] Wölfel (2023, S. 36 ff.)

Eigenschaften hätten. Er verglich dieses hypothetische Display mit einem modernen „Wunderland“, ähnlich der fantastischen Welt aus Alice im Wunderland. [21]

2.2.4 VR-Technik in den 70er bis 90er Jahren

In den 1970er- bis 1990er-Jahren machte die VR-Technik erhebliche Fortschritte und legte die Grundlage für viele der heute bekannten Anwendungen. Im Jahr 1977 entwickelten Daniel Sandin und Thomas DeFanti am Electronic Visualization Laboratory der University of Illinois in Chicago, inspiriert von Richard Sayre, den ersten Datenhandschuh, den sogenannten Sayre Glove. Der Sayre Glove erfasste Fingerbewegungen, indem sich auf jedem Finger flexible Schläuche befanden, die auf der einen Seite Fotozellen und auf der anderen Seite Lichtquellen enthielten. Sobald der Nutzer seine Finger beugte, änderte sich die Lichtmenge, die auf die Fotozellen traf, wodurch die Fingerbewegungen in elektrische Signale umgewandelt und interpretiert werden konnten. [22]

Abb. 2.12: Der erste Datenhandschuh Sayre Glove. (Wölfel, 2023, S. 40)

Parallel zu dieser Entwicklung prägte Myron Krueger im Jahr 1976 mit VIDEOPLACE die Idee einer interaktiven virtuellen Umgebung, die ohne Brillen oder Handschuhe auskam. Kruegers Konzept zielte darauf ab, eine künstliche Realität zu schaffen, die den Nutzer umgibt und auf seine Bewegungen reagiert. Dabei kombinierte er Computergrafiken, Projektoren, Videokameras, Videodisplays und Positionssensoren. VIDEOPLACE ermöglichte es den Nutzern, ihre eigenen computergenerierten Silhouetten zu sehen, die in Echtzeit ihre Bewegungen nachahmten. Besonders innovativ war die Möglichkeit, dass Nutzer aus unterschiedlichen physischen Räumen miteinander im virtuellen Raum interagieren konnten. Kruegers Arbeiten förderten die Vision, dass virtuelle Realität als kollaboratives Medium fungieren kann, das Menschen unabhängig von ihrem physischen Aufenthaltsort miteinander verbindet.

In den 1980er Jahren folgten zahlreiche technische Entwicklungen, die die immersive Technologie weiter vorantrieben. Im Jahr 1980 brachte Vicon Motion Systems, damals noch als Oxford Medical Systems

[21] Sutherland, I. E., Information Processing Techniques & Office, ARPA, OSD. (1965). *The ultimate display*. https://worrydream.com/refs/Sutherland_1965_-_The_Ultimate_Display.pdf
[22] Wölfel (2023, S. 39)

bekannt, das erste kommerzielle Motion-Capture-System auf den Markt. Diese Technologie bildete nicht nur die Basis für Trickfilme und medizinische Bewegungsanalysen, sondern ermöglichte auch neue Anwendungsfelder für die virtuelle Realität. 1981 gründete Jim Clark, ein ehemaliger Student von Ivan Sutherland, gemeinsam mit sechs weiteren Studenten das Unternehmen Silicon Graphics Inc. (SGI). SGI spezialisierte sich auf High-End-Grafik-Workstations und trug mit der Entwicklung der OpenGL-Spezifikation entscheidend zur Weiterentwicklung der Echtzeit-Computergrafik bei. [23]

1985 gründeten Jaron Lanier und Thomas Zimmermann das Unternehmen VPL Research, das als erstes kommerziell VR-Brillen und Datenhandschuhe anbot. Zu den wichtigsten Produkten zählten der Data Glove, das EyePhone HMD und die AudioSphere, die neue Maßstäbe im Bereich der VR-Technologie setzten. Ebenfalls 1985 entwickelte das Ames Research Center der NASA innerhalb eines Jahres das Virtual Visual Environment Display (VIVED). Der Prototyp, der 1986 auf der Consumer Electronics Show (CES) vorgestellt wurde, enthielt ein stereoskopisches Weitwinkel-Anzeigesystem mit zwei monochromatischen 2,7-Zoll-LCD-Bildschirmen, welche ein effektives Sichtfeld von 120° für jedes Auge boten, wodurch eine besonders immersive Darstellung ermöglicht wurde.

Abb. 2.13: VIVED. (Wölfel, 2023, S. 42)

Ein weiterer wichtiger Meilenstein war der Power Glove von 1989. Basierend auf einer vereinfachten Technologie des DataGloves von VPL Research, wurde der Power Glove als Controller-Zubehör für die Nintendo Entertainment System (NES) Spielkonsole vertrieben. Er konnte mit Hilfe akustischer Sensoren seine Lage und Bewegung ermitteln. [24]

2.2.5 VR-Technik im 21. Jahrhundert

Im Jahr 2010 baute der damals erst 18-jährige Palmer Luckey den ersten Prototyp des Oculus Rift-Headsets, das mit einem Sichtfeld von 90° und der Rechenleistung eines externen Computers arbeitete,

[23] Wölfel (2023, S. 39 ff.)
[24] Wölfel (2023, S. 41 ff.)

um die Bildwiedergabe zu ermöglichen. Zwei Jahre später sorgte Luckey mit einer äußerst erfolgreichen Kickstarter-Kampagne, die 2,4 Millionen US-Dollar einbrachte, für großes Aufsehen. Die Oculus Rift läutete eine neue Ära der VR-Technologie ein. Ein entscheidender Meilenstein war der Kauf von Oculus VR durch Facebook (heute Meta) im Jahr 2014 für 2 Milliarden US-Dollar. Im Jahr 2016 veröffentlichte Oculus schließlich das erste verbraucherorientierte VR-Headset CV-1, das neben dem HMD auch integrierte Kopfhörer, eine Kamera für videobasierte Positionsverfolgung und einen Xbox-Controller ohne Positionsverortung enthielt (Wölfel 2023). Ebenfalls im Jahr 2016 brachte HTC in Zusammenarbeit mit Valve die HTC VIVE auf den Markt. Die HTC VIVE zeichnete sich durch ihr hochpräzises Tracking-System aus, das mithilfe von mindestens zwei Sensoren, sogenannte Lighthouse-Einheiten, die Position des Headsets und der Controller exakt erfasste.

Die weltweite Corona-Pandemie ab 2020 führte zu einer erheblichen Zunahme von digitalen Kommunikations- und Kollaborationsformaten, da Millionen Schüler, Studierende und Berufstätige auf Videokonferenzen angewiesen waren. Diese außergewöhnliche Situation förderte die Entwicklung von neuen VR-Anwendungen – von Hochschulen über Start-ups bis hin zu großen Unternehmen – die darauf abzielten, eine Zusammenarbeit in immersiven, virtuellen Räumen zu ermöglichen. Besonders Fitnessanwendungen und Anwendungen mit starkem Kommunikationsfokus erfreuten sich zunehmender Beliebtheit. Dennoch blieb immersive VR, verglichen mit herkömmlichen Videokonferenzlösungen, weiterhin ein Nischenmarkt, der vor allem durch spezifische Anforderungen und technische Hürden geprägt war. [25]

[25] Wölfel (2023, S. 48 ff.)

3 Stand der Forschung

Heutzutage herrscht ein ständiges Bestreben nach der Optimierung von Designprozessen und –workflows. Daher gibt es bereits einige Anwendungen, die 3D-Designer und VR-Entwickler bei der Erstellung ihrer Arbeiten oder während eines Feedback-Prozesses mit ihren Kunden unterstützen.

3.1 Unity

3.1.1 Pixyz Plugin

Das Unity Pixyz Plugin ist ein Unity Package, welches den Nutzern erweiterte Bearbeitungsmöglichkeiten für 3D-Objekte zur Verfügung stellt und nahezu auf jeden bekannten 3D-Dateityp anwendbar ist. Bereits beim Import der 3D-Assets werden diese automatisch optimiert und Informationen über Hierarchien, Materialzuordnungen und Metadaten bewahrt.

Nach dem erstmaligen Import eines Assets, kann dieses mithilfe einer durch das Plugin zur Verfügung gestellten Toolbox bearbeitet werden, ohne dass dafür das Programm gewechselt oder das Asset neu importiert werden muss. Zu den Bearbeitungsmöglichkeiten gehören u.a.:

- Die Korrektur von Normalen
- Die Erstellung von UVs oder Kollidern
- Das Wechseln des Materials
- Die Verwaltung der Pivots
- Die Bearbeitung der Meshes [26]

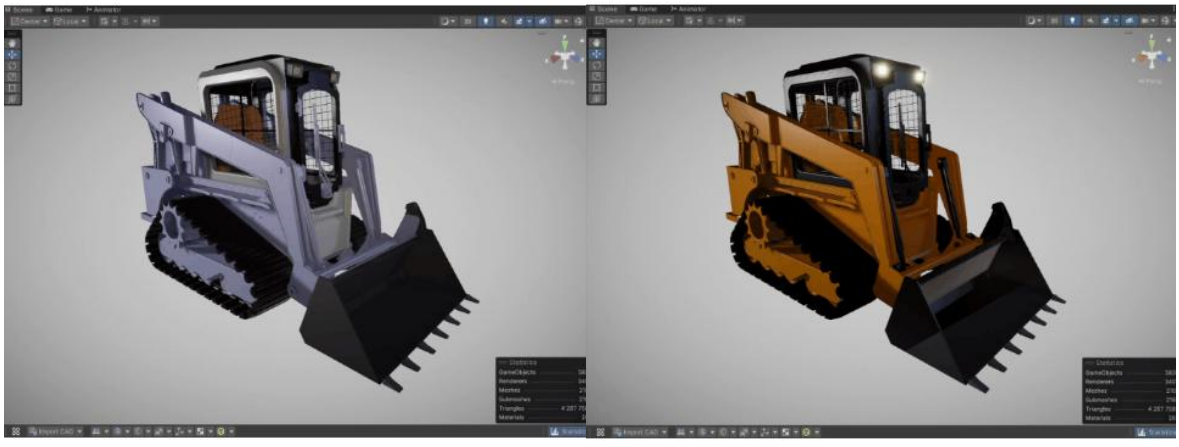


Abb. 3.1: Bearbeitung eines Modells mit Pixyz Plugin in Unity. (Unity Technologies, 2024a)

[26] Unity Technologies (Hrsg.). (2024e). *Transform your 3D Data with Pixyz Plugin | Unity*. Unity. Abgerufen am 19. März 2025, von https://unity.com/products/pixyz-plugin

Zudem ermöglicht das Pixyz Plugin die Aufstellung von Regeln mithilfe einer Rule Engine, in der Bearbeitungsabläufe gespeichert und später beliebig auf andere Assets übertragen werden können. Die Rule Engine bietet dabei sowohl die Nutzung vorgefertigter als auch die Erstellung benutzerdefinierter Regeln. Ein weiterer Mehrwert ist die automatische Generierung von verschiedenen LODs (Level of Details). Ein LOD oder dt. "Detaillierungsgrad" bestimmt die Informationstiefe und geometrische Genauigkeit eines 3D-Modells. Das LOD kann mithilfe des Plugins je nach Anforderung des Projekts angepasst werden. [27]

Trotz zahlreicher Vorteile gibt es hierbei jedoch einen wesentlichen Nachteil: den erschwerten Zugang für semi- oder nicht-professionelle VR-Entwickler. Die Nutzung des Pixyz-Plugins setzt den Besitz einer Unity-Industry-Lizenz voraus, die pro Nutzer monatlich 414 € kostet. Alternativ kann eine eigenständige Pixyz-Lizenz erworben werden, deren Preis pro Nutzer bei 2.254 € liegt. [28]

3.1.2 ProBuilder

ProBuilder ist ein vielseitiges Modellierungs- und Level-Design-Tool, das direkt in der Unity-Umgebung integriert ist. Es ermöglicht die Erstellung und Bearbeitung von 3D-Modellen sowie das Prototyping von Spielwelten, ohne dass externe 3D-Modellierungssoftware notwendig ist. ProBuilder wurde speziell entwickelt, um den Workflow von Spieleentwicklern zu optimieren und eine nahtlose Brücke zwischen Design und Entwicklung zu schaffen. Es bietet grundlegende und fortgeschrittene Modellierungswerkzeuge, um 3D-Modelle direkt innerhalb der Unity-Engine zu erstellen und zu bearbeiten.

Zu den Hauptfunktionen von ProBuilder gehören:

- **Erstellen von Grundgeometrien**: Mit ProBuilder lassen sich primitive Formen wie Würfel, Sphären oder Zylinder generieren, die als Ausgangspunkt für komplexere Modelle dienen.
- **Vertex-, Edge- und Face-Bearbeitung**: Modelle können direkt bearbeitet werden, indem Vertices (dt. „Scheitelpunkte"), Edges (dt. „Kanten") oder Faces (dt. „Flächen") angepasst und transformiert werden.
- **UV-Mapping und Texturierung**: ProBuilder bietet einfache UV-Mapping-Werkzeuge, mit denen Texturen auf die erstellten Modelle projiziert werden können.
- **Export und Import von Modellen**: Modelle können sowohl innerhalb von Unity genutzt als auch in externe Programme wie Blender exportiert werden.
- **Integriertes Level-Design**: Entwickler können Spielumgebungen direkt in Unity designen und iterativ anpassen, ohne die Engine verlassen zu müssen.

[27] Unity Technologies. (2024c). *Prepare and optimize models | Pixyz Plugin for Unity | 3.1.1*. Abgerufen am 25. April 2025, von https://docs.unity3d.com/Packages/com.unity.industry.toolkit@3.1/manual/prepare-optimize-model.html

[28] Unity Technologies. (2024d). *Real-time tools for 3D, AR, and VR development | Products*. Unity. Abgerufen am 25. April 2025, von https://unity.com/products?c=tools+marketplace

ProBuilder unterscheidet sich in einigen Punkten von umfangreicheren 3D-Tools wie z.B. Blender:

- *Funktionstiefe*: Blender bietet als vollwertige 3D-Software eine deutlich umfangreichere Palette an Werkzeugen für Modeling, Sculpting, Animation und Rendering. ProBuilder hingegen konzentriert sich auf grundlegendes Modeling und Level-Design.
- *Anwendungsbereich*: ProBuilder ist vor allem auf schnelles Prototyping und den Einsatz innerhalb der Unity-Engine optimiert. Blender eignet sich besser für detaillierte, hochauflösende 3D-Modelle, die z.B. für Animationen oder Filmproduktionen benötigt werden.
- *Workflow*: Mit ProBuilder entfällt der ständige Import/Export-Prozess, der notwendig ist, wenn externe Tools wie Blender verwendet werden. Dies spart Zeit und vereinfacht den Workflow erheblich. [29]

3.1.3 MeshSync

MeshSync ist ein Unity Package, das in Echtzeit die Synchronisierung von Meshes und Modellen zwischen DCC-Tools, wie z.B. Blender oder Autodesk 3ds Max, und Unity ermöglicht. Diese Integration erlaubt es Entwicklern, Änderungen an 3D-Modellen sofort in Unity zu sehen, ohne den üblichen Export- und Importprozess durchlaufen zu müssen.

Die Echtzeit-Synchronisierung bezieht sich dabei sowohl auf Änderungen des 3D-Modells als auch auf Material- und Texturänderungen und Animationsdaten. Letzteres sorgt dafür, dass Transformationen direkt in Unity überprüft werden können. [30]

3.2 Blender

3.2.1 VR-Szeneninspektion

Die Blender VR-Szeneninspektion ist ein Add-on, das es ermöglicht, in Blender 3D-Szenen direkt mit einem HMD zu betrachten. Mit diesem Tool können Nutzer ihre Szenen in einem immersiven VR-Erlebnis inspizieren, was besonders für Designprüfungen und Präsentationen nützlich ist. Zu den Hauptfunktionen gehören das Starten einer VR-Sitzung über die OpenXR-Plattform, verschiedene Tracking-Modi für die Navigation, sowie die Nutzung von Controllern zur Interaktion. Zusätzlich ermöglicht das Add-on die Erstellung von Landmarks, um feste Ansichten zu speichern. Die VR-Szeneninspektion erleichtert somit die Bewertung von 3D-Projekten in der virtuellen Realität. [31]

[29] Unity Technologies (Hrsg.). (2024b). *About ProBuilder*. ProBuilder 5.2.3. Abgerufen am 2. Januar 2025, von https://docs.unity3d.com/Packages/com.unity.probuilder@6.0/manual/index.html
[30] Unity Technologies (Hrsg.). (2023). *MeshSync*. Abgerufen am 28. Dezember 2024, von https://docs.unity3d.com/Packages/com.unity.meshsync@0.17/manual/index.html
[31] Blender (Hrsg.). (o. D.). *VR Scene Inspection — Blender Manual*. Blender 2.83 Manual. Abgerufen am 25. April 2025, von https://docs.blender.org/manual/en/2.83/addons/3d_view/vr_scene_inspection.html

3.2.2 Freebird XR

Freebird XR ist ein Plugin für Blender, das die Integration von Virtual Reality in den 3D-Designprozess ermöglicht. Es richtet sich an Nutzer, die ihre 3D-Modelle in einer immersiven Umgebung erstellen und bearbeiten möchten.

Hauptfunktionen von Freebird XR:

- **Erstellung und Bearbeitung von Linien:** Erstellte Linien können verschoben, gelöscht oder in ihrer Größe angepasst werden, um die gewünschte Form zu erreichen.
- **Skizzieren von Grundformen:** Mit dem Shape-Werkzeug können Nutzer grundlegende Formen wie Würfel, Kugeln und Zylinder direkt in der 3D-Umgebung erstellen. Diese Funktion erleichtert das schnelle Prototyping komplexerer Modelle.
- **Intuitive Charakterposen:** Das Plugin ermöglicht es, 3D-Modelle direkt in VR zu bearbeiten und Charaktere auf natürliche Weise zu posieren, was den Animationsprozess vereinfacht.
- **Nahtloser Wechsel zwischen Blender und VR:** Freebird XR funktioniert direkt innerhalb von Blender, sodass ein reibungsloser Wechsel zwischen der Desktop-Oberfläche und der VR-Umgebung möglich ist. Änderungen, die in der VR-Umgebung vorgenommen werden, erscheinen sofort in der Desktop-Oberfläche und umgekehrt.

Durch die Kombination von VR-Technologie mit den leistungsstarken Funktionen von Blender bietet Freebird XR eine innovative Lösung für die schnelle und effiziente Erstellung von 3D-Konzeptmodellen. [32]

3.3 Eigenständige Software

Nach der Betrachtung von Integrationen und Plugins für Unity und Blender, widmet sich der folgende Abschnitt eigenständigen Softwarelösungen und VR-Anwendungen. Diese Ansätze zeichnen sich insbesondere durch die Möglichkeit der Echtzeit-Bearbeitung von 3D-Modellen aus. Sie ermöglichen es, 3D-Objekte direkt in einer immersiven 3D-Umgebung zu erstellen, anzupassen und oft auch kollaborativ mit anderen Bearbeitern oder Kunden zu betrachten und zu bearbeiten. Der Fokus liegt hierbei auf einer interaktiven und flexiblen Arbeitsweise, die über traditionelle Desktop-Software hinausgeht und neue Potenziale für kreative Workflows eröffnet. Im Folgenden werden diese Lösungen detailliert vorgestellt.

[32] Freebird XR (Hrsg.). (o. D.-b). *Roadmap & Pricing*. Freebird XR. Abgerufen am 25. April 2025, von https://freebirdxr.com/roadmap/

3.3.1 Gravity Sketch

Gravity Sketch ist eine VR-Anwendung, die sowohl für das industrielle Design sowie die Gestaltung von Charakteren entwickelt wurde. Die Software ermöglicht es, in einer immersiven Umgebung zu arbeiten, wobei Modelle im Maßstab 1:1 direkt im virtuellen Raum erstellt werden können. Gravity Sketch bietet verschiedene Werkzeuge, darunter eine anpassbare Farbpalette und flexible Linienarten, deren Dicke und Form modifiziert werden können. Es ist möglich, Flächen und Körper zu erstellen und diese in Eigenschaften wie Größe, Form oder Kantenverlauf nachträglich zu bearbeiten. Zudem können Änderungen von Farben und Materialien vorgenommen werden. Für unerfahrene Nutzer bietet die Software eine integrierte Lernplattform, die über ein Menü in der VR-Umgebung zugänglich ist. Dort stehen thematisch geordnete Anleitungsvideos zur Verfügung, die die Hauptfunktionen erklären und direktes Ausprobieren ermöglichen.

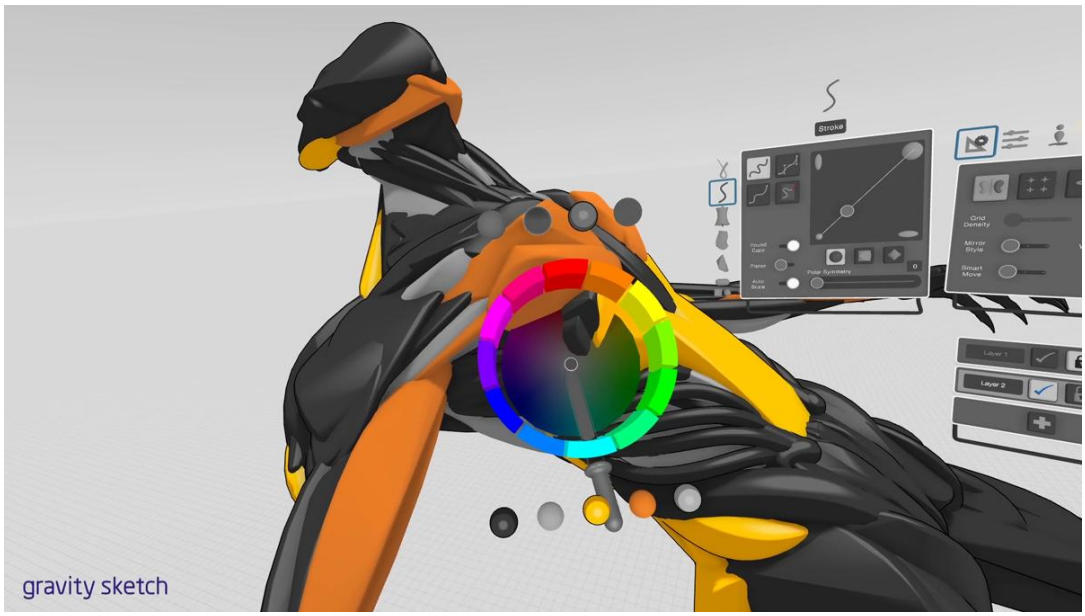


Abb. 3.2: Gravity Sketch. (Introduction To Gravity Sketch VR - PART 2, 2020)

Zentraler Bestandteil der Anwendung ist das sogenannte LandingPad, ein Design-Hub, das die Speicherung und Verwaltung von 3D-Kreationen ermöglicht. Über dieses Feature können Designs plattformübergreifend genutzt und Referenzmaterialien eingebunden werden. Zudem bietet eine integrierte Bibliothek Zugang zu vorgefertigten Modellen und virtuellen Räumen. Das LandingPad ermöglicht auf diese Weise auch die Zusammenarbeit mit Kunden oder Teamkollegen, da Entwürfe mit anderen geteilt werden können. Dabei ist die Nutzung eines Headsets optional, was die Anwendung auch für Personen ohne VR-Hardware zugänglich macht. Die Basisversion der Software steht Einzelanwendern, Freiberuflern und kleineren Teams von bis zu drei Personen pro Raum kostenfrei zur Verfügung. [33]

[33] Gravity Sketch. (2024, 12. Dezember). *Home - Gravity Sketch*. Abgerufen am 2. Januar 2025, von https://gravitysketch.com/

3.3.2 Autodesk Workshop XR

Autodesk Workshop XR ist eine immersive VR-Plattform, die speziell für kollaborative Designprüfungen entwickelt wurde. Im Gegensatz zu Tools, die sich auf die Erstellung von VR-Inhalten konzentrieren, ermöglicht Autodesk Workshop XR Teams, bestehende Projekte in einer realitätsnahen Umgebung und in Echtzeit zu begutachten und zu optimieren. Projektbeteiligte haben die Möglichkeit sich innerhalb der virtuellen Umgebung im Maßstab 1:1 zu bewegen und können zudem präzises Feedback geben.

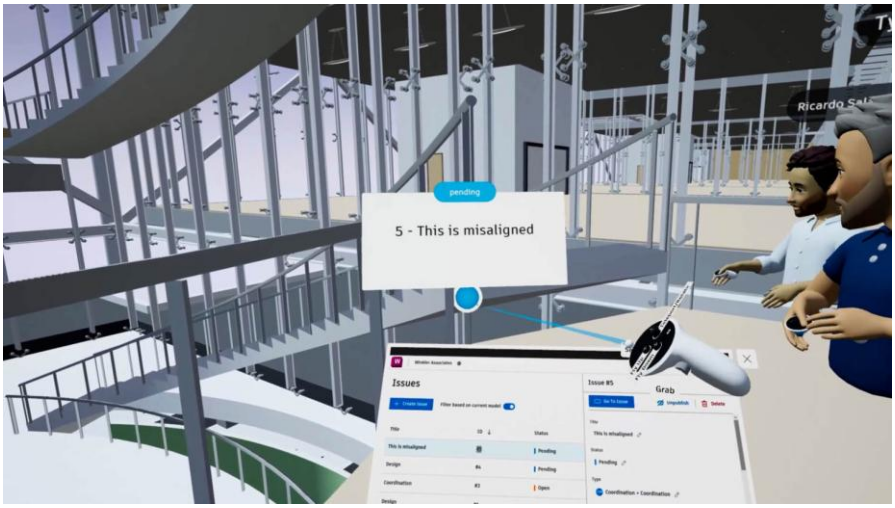


Abb. 3.3: Anmerkung von Fehlern in Autodesk Workshop XR. (Corke, 2024)

Die Plattform beinhaltet eine Integration mit der Autodesk Construction Cloud (ACC) und unterstützt so eine bidirektionale Synchronisation aller Daten. Dies bedeutet, dass alle Änderungen und Aktualisierungen in Autodesk Workshop XR, automatisch mit der Cloud synchronisiert werden. Diese Echtzeit-Updates sollen sicherstellen, dass alle Projektbeteiligten – unabhängig davon, ob sie in VR oder auf einem Desktop arbeiten – immer mit den aktuellsten Daten arbeiten. Zu den Funktionen gehören außerdem der Zugriff auf detaillierte BIM-Daten (Building Information Modeling) sowie Werkzeuge für die Versionskontrolle, die darauf abzielen, Arbeitsabläufe zu strukturieren und die Genauigkeit zu erhöhen.

Abb. 3.4: Zugang zu BIM-Daten und Eigenschaften. (Corke, 2024)

Insbesondere soll Autodesk Workshop XR dazu dienen, räumliche Probleme in großen, zusammengesetzten Modellen zu erkennen, die auf traditionellen 2D-Oberflächen ggf. übersehen werden. Über einfache Interaktionen, wie einen Klick oder eine Sprachanweisung, können Nutzer in der VR-Umgebung Probleme markieren und dokumentieren. Diese Informationen – einschließlich der entsprechenden Standortdetails und notwendigen Maßnahmen – werden dabei in Echtzeit zurück in die ACC übertragen. Ein Meta Quest VR-Headset ist Voraussetzung für die Nutzung von Autodesk Workshop XR. Die Möglichkeit, direkt in VR zu arbeiten, unterstützt die Zusammenarbeit und die Abstimmung mit externen Partnern oder Kunden. So bietet die Plattform zwar keine Werkzeuge zur Erstellung von VR-Umgebungen, jedoch dient sie als positives Beispiel für die Optimierung von Feedback-Prozessen innerhalb bestehender VR-Projekte. [34]

3.3.3 ShapeGuide

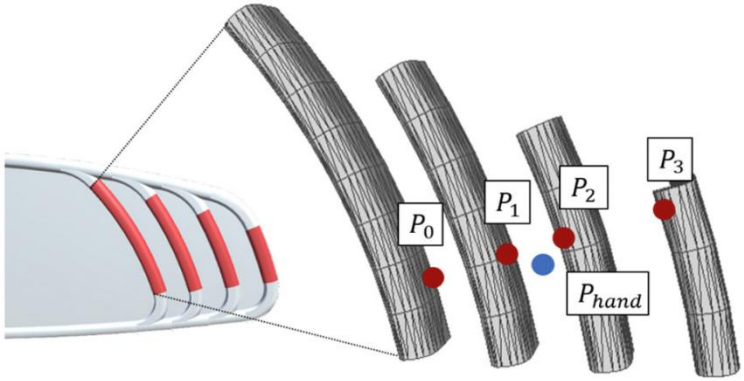
ShapeGuide, vorgestellt von Okuya et. Al (2018), ist eine innovative Anwendung, die es ermöglicht, native CAD-Daten in einer virtuellen Umgebung direkt anzupassen. Anders als Tools, die 3D-Modelle jeglichen Dateiformats bearbeiten, konzentriert sich ShapeGuide ausschließlich auf die Modifikation von CAD-Daten. Ziel ist es, die Arbeit mit CAD-Modellen durch immersive Interaktionen für Designer und Ingenieure effizienter und zugänglicher zu gestalten.

Die Plattform bietet Nutzern die Möglichkeit, die Oberfläche eines CAD-Objekts in VR zu greifen und durch Handbewegungen anzupassen. Zuvor berechnet das System verschiedene Formvariationen basierend auf den zugrunde liegenden CAD-Parametern, um unvorhersehbare Verformungen zu vermeiden. Die aktuell aktivierte Form wird direkt visuell dargestellt. Dabei soll ein haptisches Feedback die Handbewegungen des Nutzers stabilisieren und ihn gezielt zu den gewünschten Modifikationen führen. Die Anwendung verbindet sich dabei mit einem VR-CAD-Server, der die Berechnung und Visualisierung der Formvariationen steuert. Dank einer verteilten Systemarchitektur und der nahtlosen Synchronisation zwischen VR-Umgebung und CAD-Datenbank werden Änderungen in Echtzeit auf allen Plattformen aktualisiert. Auf diese Weise soll das Hin- und Herwechseln zwischen VR-Systemen und Workstations reduziert und dadurch die Anzahl der Iterationen im Entwurfsprozess signifikant gesenkt werden. Besonders für Nicht-CAD-Experten bietet ShapeGuide eine intuitive Möglichkeit, Formanpassungen vorzunehmen, ohne tiefgehendes Fachwissen über CAD-Strukturen zu benötigen. In Experimenten zeigte sich, dass ShapeGuide im Vergleich zu herkömmlichen Methoden nicht nur schneller ist, sondern auch präzisere Ergebnisse liefert. Nutzer empfanden das System als konsistenter und weniger frustrierend, da es eine bessere Übereinstimmung zwischen Handbewegungen und Formdeformationen ermöglicht.

Okuya et al. stellen ShapeGuide anhand eines praktischen Beispiels vor, welches die Modifikation eines Rückspiegels beinhaltet: Der Benutzer wählt einen bestimmten Teil des CAD-Modells aus, woraufhin

[34] Autodesk Inc. (Hrsg.). (2024, 12. November). *Immersive Design Review Workspace | Autodesk Workshop XR*. Workshop XR. Abgerufen am 19. März 2025, von https://workshopxr.autodesk.com/

ShapeGuide eine Reihe möglicher Formen berechnet. Das System zeigt dabei immer nur die Formvariante an, die der Hand des Nutzers am nächsten liegt. Änderungen werden direkt im VR-System validiert und zurück in die CAD-Datenbank übertragen, um den Workflow zu optimieren.



MeshId	$i = 0$	$i = 1$	$i = 2$	$i = 3$
PARAM	120	130	140	150

Abb. 3.5: Beispiel für berechnete Unterteilnetze des rechten Teils für 3D-Interaktion. Phand ist die Position der Hand des Benutzers. Pi ist der nächstgelegene Punkt auf jeder Fläche von Phand. (Okuya et al., 2018 S.6)

Herausforderungen wie die Übersteuerung bei kleinen Formvariationen und die Anpassung für komplexere Szenarien erfordern weitere Forschung. Die Entwickler erklären, dass perspektivisch Multi-User-Funktionen entwickelt werden könnten, die eine gleichzeitige Bearbeitung durch mehrere Experten ermöglichen. ShapeGuide zeigt, wie spezialisierte VR-Tools traditionelle Arbeitsweisen durch immersive Technologien transformieren können. Während die Plattform klar auf CAD-Daten fokussiert ist, setzt sie Maßstäbe in der effizienten und intuitiven Bearbeitung von 3D-Modellen in VR. [35]

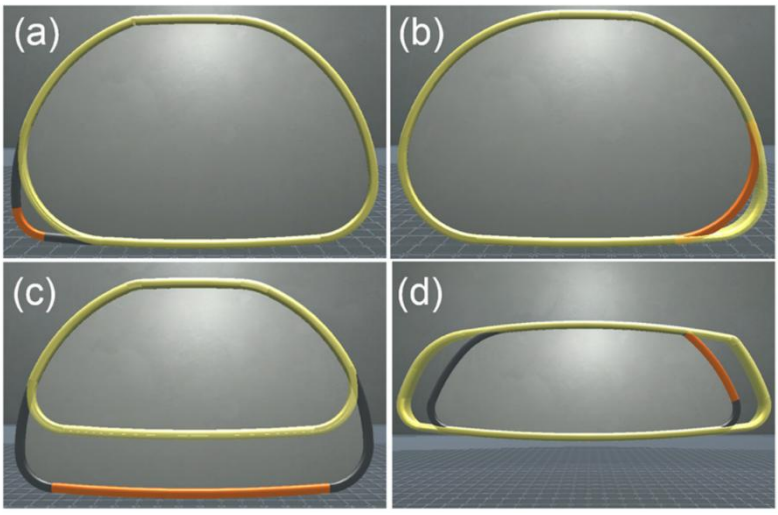


Abb. 3.6: Vier Beispiele für die experimentelle Aufgabe der Nutzer: orangefarbene Teile mussten verändert werden, um die gelben Ziele zu erreichen. (Okuya et al., 2018 S.9)

[35] Okuya, Y., Ladeveze, N., Fleury, C. & Bourdot, P. (2018). ShapeGuide: Shape-Based 3D Interaction for Parameter Modification of Native CAD Data. *Frontiers in Robotics And AI*, *5*. https://doi.org/10.3389/frobt.2018.00118

3.3.4 The Virtual Workshop

The Virtual Workshop ist eine VR-Anwendung, die speziell für die interaktive Erstellung und Bearbeitung von 3D-Modellen entwickelt wurde. Sie zielt darauf ab, die traditionellen Herausforderungen der 3D-Modellierung – wie die begrenzte Präzision und Benutzerfreundlichkeit bei der Arbeit auf zweidimensionalen Bildschirmen – zu überwinden, indem 3D-Modelle mit beiden Händen effizient und präzise gestaltet werden können. Das System unterstützt die Erstellung neuer 3D-Modelle aus vorgefertigten Basisobjekten und den Import bestehender Modelle aus anderen Anwendungen. Erstellte 3D-Modelle können im Standard-OBJ-Dateiformat gespeichert und in andere Softwarelösungen, wie Unity, exportiert werden. Dies soll die Integration in bestehende Workflows erleichtern und eine vielseitige Nutzung der Modelle in unterschiedlichen Kontexten ermöglichen.

Abb. 3.7: The Virtual Workshop: (a) Ein Benutzer modelliert ein 3D-Objekt, (b) Objektmodus, (c) Materialmodus. (Pan & Haberkorn, 2021, S. 280–2)

Ein zentrales Ziel von The Virtual Workshop ist die Verbesserung der Zusammenarbeit zwischen Designern und Kunden. Die offene Plattform erlaubt es, Ideen auszutauschen und virtuelle Modelle interaktiv in Echtzeit zu verfeinern. Die experimentellen Ergebnisse zeigen, dass das System eine hohe Effizienz bietet. Modelle mit normaler Komplexität können in weniger als 0,02 Sekunden gerendert werden. Allerdings erhöht sich die Rendering-Zeit bei steigender Modellkomplexität oder bei hochauflösenden Texturen. Trotz dieser Einschränkungen ermöglicht die Plattform eine schnelle und präzise Visualisierung und Bearbeitung von 3D-Objekten. [36]

[36] Pan, X. & Haberkorn, A. (2021). Interactive 3D Modeling with Virtual Reality. *Electronic Imaging*, *33*(8), 280–287. https://doi.org/10.2352/issn.2470-1173.2021.8.imawm-280

4 Forschungsfrage und Hypothesen

Die Erstellung von 3D-Modellen für virtuelle Welten erfordert in der Regel den Einsatz mehrerer spezialisierter Softwarelösungen. Während Modellierungsprogramme wie Blender oder 3ds Max für die Erstellung der Modelle genutzt werden, erfolgt die Integration und weitere Bearbeitung meist in einer Game Engine wie Unity. Dieser Wechsel zwischen verschiedenen Anwendungen kann den Designprozess sowie den Feedback-Austausch mit Kunden erschweren und verzögern.

Mit ProBuilder bietet Unity ein integriertes Modellierungswerkzeug, das potenziell eine Alternative zu herkömmlichen Workflows darstellen könnte. Allerdings ist bislang unklar, inwiefern sich ProBuilder für die Erstellung komplexer 3D-Modelle eignet und welche technischen oder gestalterischen Einschränkungen sich dabei ergeben. Zudem stellt sich die Frage, ob die Arbeit innerhalb einer einzigen Software die Effizienz des Design- und Feedbackprozesses beeinflusst.

Vor diesem Hintergrund werden in diesem Kapitel die zentralen Forschungsfragen formuliert, die dieser Arbeit zugrunde liegen. Zudem werden Hypothesen aufgestellt, die im weiteren Verlauf überprüft werden, um die Möglichkeiten und Grenzen der Nutzung von ProBuilder für die Entwicklung modularer VR-Welten zu analysieren.

4.1 Forschungsthema

Wie in den vorherigen Kapiteln dargelegt, stehen 3D-Künstlern zahlreiche Möglichkeiten zur Verfügung, um Modelle für virtuelle Welten zu erstellen. Diese 3D-Modelle können mit verschiedenen Softwarelösungen wie Blender oder 3ds Max modelliert und anschließend in eine Game Engine wie Unity importiert werden. Neuere Programme wie Gravity Sketch ermöglichen es zudem, 3D-Modelle direkt in einem dreidimensionalen Raum zu erstellen und sie anschließend in die virtuelle Realität innerhalb der Game Engine zu integrieren.

Ein gemeinsames Merkmal bisheriger Arbeitsabläufe ist jedoch die Nutzung mehrerer Programme. So erfolgt die Modellierung typischerweise in einer spezialisierten Software wie Blender, während im Anschluss die Integration in Unity und dort die Zusammensetzung der virtuellen Welt erfolgt. Insbesondere wenn Änderungen am Modell vorgenommen werden müssen, kann dies einen kontinuierlichen Wechsel zwischen den Anwendungen erfordern, da diese nach wie vor in der ursprünglichen Modellierungssoftware erfolgen müssen. Dieser iterative Prozess kann insbesondere in Abstimmungsphasen mit Kunden zeitaufwendig sein.

Während es in der Vergangenheit notwendig war, nach jeder Modifikation ein Modell erneut in Unity zu importieren, existieren inzwischen Plugins, die diesen Workflow optimieren. Beispielsweise ermöglicht das Pixyz-Plugin die Umsetzung von Anpassungen direkt in Unity, ist jedoch mit hohen Kosten

verbunden. Eine alternative Lösung bietet MeshSync, das eine Echtzeit-Synchronisation zwischen digitalen Content-Creation-Tools und Unity ermöglicht. Diese Entwicklungen vereinfachen den Feedbackprozess bereits erheblich. Dennoch bleibt die Frage offen, ob es eine Lösung gibt, die sämtliche Bearbeitungsschritte innerhalb einer einzigen Software realisierbar macht.

Eine mögliche Option stellt ProBuilder dar, ein in Unity integriertes Tool zur Erstellung von 3D-Modellen. Bisher wurde ProBuilder hauptsächlich für einfache Spiele oder Level-Designs mit grundlegenden geometrischen Formen genutzt. Daher soll in dieser Arbeit untersucht werden,

- ob mit ProBuilder die Erstellung eines komplexen, modifizierbaren und modularen 3D-Modells direkt in Unity realisierbar ist,
- inwiefern der Designprozess durch technische oder gestalterische Einschränkungen beeinflusst wird und
- welche Auswirkungen die Arbeit innerhalb einer einzigen Software auf den Feedbackprozess hat.

Diese Untersuchung soll Aufschluss darüber geben, ob die Nutzung von ProBuilder eine Alternative zu herkömmlichen Workflows darstellt und welche Potenziale sich daraus für den Entwicklungsprozess von 3D-Modellen ergeben.

4.2 Fragestellung

Die zentrale Fragestellung dieser Arbeit lautet:

F1: Welche technischen und gestalterischen Möglichkeiten und Grenzen bestehen bei der Entwicklung eines modifizierbaren und modularen 3D-Modells in Unity und wie wirkt sich dies auf die Effizienz des Feedbackprozesses aus?

Zur Beantwortung der zentralen Fragestellung gilt es, folgende Unterfragen zu klären:

- **F1.1:** Ist es mit ProBuilder möglich, ein komplexes und möglichst modifizierbares und modulares 3D-Modell zu erstellen?
- **F1.2:** Erweist sich der Designprozess mit ProBuilder als ebenso effizient wie mit einer etablierten DCC-Software?
    - Falls dies nicht der Fall ist, sollte untersucht werden, ob die geringere Effizienz auf mangelnde Vorerfahrung mit ProBuilder zurückzuführen ist, so dass sich die Effizienz ggf. durch eine längere Nutzung steigern lässt.
- **F1.3:** Gibt es technische und gestalterische Einschränkungen während des Designprozesses mit ProBuilder, und wenn ja, welche?
- **F1.4:** Welche Auswirkungen hat die ausschließliche Nutzung einer einzigen Software auf den Feedbackprozess?

4.3 Hypothesen

Folgende Hypothesen dienen dazu, die oben genannten Forschungsfragen zu untersuchen:

- **H1:** Mit ProBuilder lässt sich ein komplexes, möglichst modifizierbares und modulares 3D-Modell erstellen.
- **H2:** Der Designprozess eines komplexen, möglichst modifizierbaren und modularen 3D-Modells ist mit ProBuilder ebenso effizient wie mit einer etablierten Modellierungssoftware wie Blender.
- **H3:** Es gibt sowohl technische als auch gestalterische Herausforderungen bzw. Grenzen während des Designprozesses mit ProBuilder.
- **H4:** Die Nutzung einer einzigen Software führt dazu, dass der Feedbackprozess von Kunden als effizient wahrgenommen wird.

5 Methodik

Dieses Kapitel beschreibt die Vorgehensweise bei der Umsetzung des praktischen Teils dieser Arbeit. Die Methodik gliedert sich in drei aufeinander aufbauende Phasen.

Die erste Phase beinhaltete die Entwicklung eines modularen und modifizierbaren Prototyps. Es werden demnach zunächst grundlegende Überlegungen zur Modellauswahl beschrieben, die insbesondere im Hinblick darauf gemacht wurden, welche Kriterien ein Modell erfüllen muss, um als modular und modifizierbar zu gelten. Darauf folgt eine detaillierte Darstellung des technischen Aufbaus, einschließlich der verwendeten Hard- und Software. Anschließend wird die praktische Umsetzung des Prototyps erläutert – von der Modellierung bis zur Zusammenstellung einzelner Komponenten. Dabei werden sowohl die einzelnen Implementierungsschritte als auch auftretende Herausforderungen und entsprechende Lösungsansätze thematisiert.

Die zweite Phase umfasste die Konzeption und Durchführung einer Befragung mit dem Ziel, den entwickelten Prototyp hinsichtlich seiner Funktionalität und Flexibilität zu evaluieren. Im Fokus stand dabei der Feedbackprozess: Die erhobenen Rückmeldungen wurden systematisch auf potenzielle Anpassungs- und Verbesserungsvorschläge hin analysiert – insbesondere im Hinblick auf solche, die mit vertretbarem Aufwand zeitnah umgesetzt werden konnten.

Die dritte Phase widmete sich der Weiterentwicklung des Modells. Aufbauend auf den Ergebnissen der vorangegangenen Befragung wurde versucht, die identifizierten Verbesserungsvorschläge bestmöglich umzusetzen. Zudem wurde reflektiert, welche Aspekte in der verfügbaren Zeit nicht realisiert werden konnten.

5.1 Erste Phase: Entwicklung des Prototyps

Für die Entwicklung des Prototyps war es essenziell, ein 3D-Modell zu wählen, das sich durch eine hohe Variabilität auszeichnet. Ein solches Modell sollte in zahlreichen Ausführungen existieren und zugleich genügend Gestaltungsspielraum bieten, um Anpassungen in Struktur und Form flexibel umzusetzen.

Als zentrales Modell für den Prototypen wurde eine Pflanze gewählt. Die Entscheidung für eine Pflanze bot mehrere Vorteile, insbesondere im Hinblick auf die flexible Anpassbarkeit und Modularität. Pflanzen weisen von Natur aus komplexe Formen auf, die sich nicht nur zwischen verschiedenen Arten, sondern auch innerhalb einer einzigen Pflanzenart erheblich unterscheiden. Diese natürliche Variabilität erforderte ein hohes Maß an Flexibilität im Designprozess, da zahlreiche Modifikationen denkbar waren.

Damit das Modell als Grundlage für die prototypische Umsetzung geeignet war, wurden bestimmte strukturelle Anforderungen definiert. Der entwickelte Prototyp orientierte sich dabei nicht an einer spezifischen existierenden Pflanzenart, sondern setzte sich aus charakteristischen Merkmalen verschiedener

Pflanzen zusammen. Ziel war es, eine flexible, modular anpassbare Struktur zu schaffen, die nicht an eine feste botanische Vorlage gebunden ist. Das Modell sollte aus einem Topf bestehen, der mit Erde gefüllt war, sowie einer Pflanze, die sich aus unterschiedlich langen und geformten Stängeln zusammensetzte. An den Stängeln sollten Blätter in variierender Anordnung befestigt sein.

Modularität und Modifizierbarkeit

Die Begriffe Modifizierbarkeit und Modularität waren zentrale Konzepte für die Gestaltung des 3D-Modells. Modifizierbarkeit beschreibt die Möglichkeit, ein Modell nachträglich zu verändern, sei es durch Anpassung einzelner Parameter oder durch eine Umgestaltung von Strukturen.[37] Modularität hingegen bedeutet, dass ein Modell aus separaten, austauschbaren und miteinander kombinierbaren Komponenten besteht.[38] Ein modifizierbares und modulares Modell zeichnet sich demnach dadurch aus, dass möglichst viele seiner Eigenschaften flexibel verändert oder sogar vollständig ersetzt werden können, ohne die Gesamtstruktur zu beeinträchtigen.

In diesem Fall sollte das Modell aus einer Pflanze mit Stängeln bestehen, an deren Enden Blätter befestigt waren. Die Pflanze selbst sollte sich in einem mit Erde gefüllten Topf befinden. Um dieses Modell als modular und modifizierbar bezeichnen zu können, mussten verschiedene Anpassungsmöglichkeiten gegeben sein. Grundsätzlich sollten sowohl das gesamte Modell als auch dessen einzelne Komponenten – also der Topf mit Erde, die Stängel und die Blätter – veränderbar sein. Daher wurden im Vorwege die folgenden grundlegenden Anpassungsmöglichkeiten definiert:

- **Skalierung**: Das Modell sollte in alle Richtungen (x, y, z) vergrößert oder verkleinert werden können, sowohl als Ganzes als auch auf Ebene der einzelnen Komponenten.
- **Rotation**: Die Orientierung des gesamten Modells sowie der einzelnen Elemente sollte flexibel anpassbar sein.
- **Position**: Die Verschiebung des Modells oder einzelner Komponenten sollte unabhängig voneinander möglich sein.

Zusätzlich waren für die einzelnen Bestandteile des Modells spezifische Modifikationen vorgesehen:

- **Erde**:
    - Anpassung der Textur bzw. des Materials zur Variation des Oberflächencharakters.
    - Möglichkeit zur Verformung der Oberfläche, um Unebenheiten oder individuelle Strukturen darzustellen.

[37] DWDS (Hrsg.). (2025, Januar). *Modifizieren*. DWDS. Abgerufen am 25. April 2025, von https://www.dwds.de/wb/modifizieren
[38] DWDS (Hrsg.). (2024, 14. Mai). *Modular*. DWDS. Abgerufen am 25. April 2025, von https://www.dwds.de/wb/modular

- **Stängel**:
    - Veränderung der Textur bzw. des Materials zur Anpassung an unterschiedliche Pflanzentypen.
    - Anpassung der Form und des Verlaufs, um natürliche Wuchsvariationen darzustellen.
- **Blätter**:
    - Anpassung der Textur bzw. des Materials für unterschiedliche Oberflächenstrukturen.
    - Veränderung der Form der Blätter, um verschiedene Arten oder Entwicklungsstadien abzubilden.
    - Anpassung der Anzahl und Anordnung der Blätter an den Enden der Stängel.

Durch diese Vielzahl an Modifikationsmöglichkeiten sollte sichergestellt werden, dass das Modell nicht nur vielseitig nutzbar, sondern auch individuell anpassbar sein würde. Darüber hinaus sollte mit dieser Anpassungsfähigkeit der Feedbackprozess mit Kunden möglichst effizient gestaltet werden.

5.1.1 Hardware und Software

Hardware

Der Praxisteil dieser Arbeit wurde zunächst auf einem HP 255 G8 Laptop durchgeführt. Der Laptop war mit dem Betriebssystem Windows 11 Home ausgestattet und verfügte über einen AMD Ryzen 5 5500U Prozessor mit Radeon Graphics, der eine Taktfrequenz von 2,10 GHz bot. Der installierte Arbeitsspeicher betrug 8 GB. Eine dedizierte Grafikkarte war nicht vorhanden, was die Verarbeitung grafisch intensiver Anwendungen limitierte.

Um komplexere Anforderungen besser bewältigen zu können, wurde später auf ein leistungsfähigeres Modell gewechselt, nämlich das Microsoft Surface Book 3. Dieses Gerät lief ebenfalls unter dem Betriebssystem Windows 11 Home und war mit einem Intel Core i7-1065G7 Prozessor ausgestattet, der eine Basistaktfrequenz von 1,30 GHz und eine maximale Turbofrequenz von 1,50 GHz bot. Der installierte Arbeitsspeicher umfasste 32 GB, was eine erhebliche Steigerung der Verarbeitungsleistung darstellte. Zudem verfügte dieses Modell über eine dedizierte NVIDIA GeForce GTX 1650 Grafikkarte mit Max-Q Design, was bei der Arbeit mit grafikintensiven Anwendungen von Vorteil war.

Software

Für die Umsetzung des Projekts wurden verschiedene Softwaretools verwendet, die jeweils spezifische Aufgaben erfüllten. Nachfolgend sind die verwendeten Softwares und ihre jeweiligen Anwendungsbereiche aufgelistet:

- **Unity (Version 2022.3.52f1)**: Verwendung für die Erstellung bzw. Darstellung des Modells.
- **ProBuilder Package (Version 5.2.3)**: Einsatz für das Modeling innerhalb der Unity-Umgebung.

- **Visual Studio 2022**: Genutzt für das Schreiben von C#-Skripten, die dazu dienten, dem Modell bestimmte modulare bzw. modifizierbare Eigenschaften zu geben.
- **Adobe Illustrator**: Verwendung zur Erstellung eigener Texturen, die im Modell eingesetzt wurden.

5.1.2 Entwicklung des Prototyps

Die Entwicklung des Prototyps, einer modularen und modifizierbaren Pflanze in einem Topf mit Erde, erfolgte in mehreren aufeinander aufbauenden Schritten. Zunächst wurde die erforderliche Software eingerichtet und eine Einarbeitung in die Werkzeuge des ProBuilder Packages durchgeführt. Anschließend wurde das Modell schrittweise aufgebaut, wobei für jede Komponente geprüft wurde, wie ein möglichst hoher Grad an Modularität und Modifizierbarkeit erreicht werden kann. Dabei wurden verschiedene Ansätze getestet und durch iterative Anpassungen weiterentwickelt, so dass Trial-and-Error ein wesentlicher Bestandteil des Entwicklungsprozesses war. Die Umsetzung erfolgte unter Verwendung der Standardfunktionen von ProBuilder sowie durch die Implementierung zusätzlicher C#-Skripte, um spezifische Anpassungsmöglichkeiten zu realisieren.

Installation von ProBuilder und Einarbeitung

Die Installation des ProBuilder Packages erfolgte, wie auch bei anderen Unity-Erweiterungen üblich, über den Paketmanager. Mit dem erfolgreichen Abschluss der Installation, wurde der Scene View ein zusätzliches Auswahlmenü hinzugefügt (siehe Abb. 5.1). Dieses ermöglichte die gezielte Auswahl ganzer Objekte oder spezifischer Elemente, wie Vertices, Edges und Faces, zur präzisen und flexiblen Modellierung des Objekts.

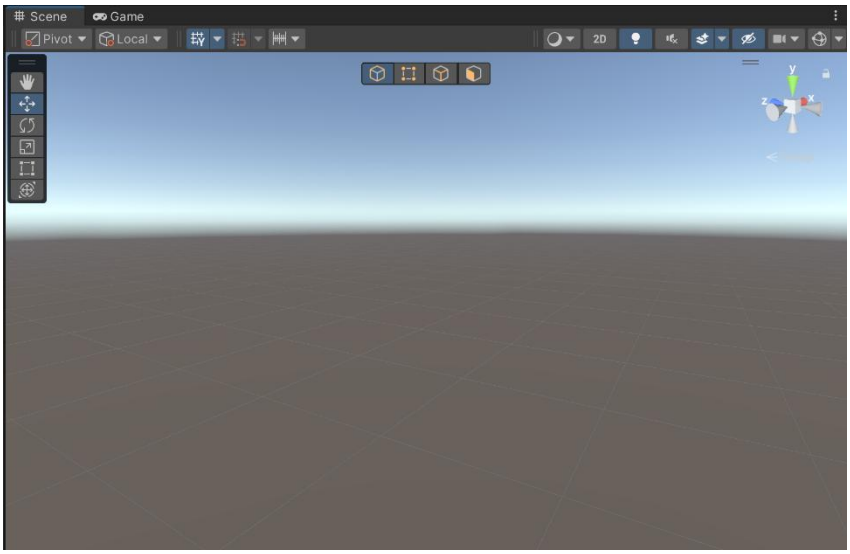


Abb. 5.1: Scene View mit zusätzlichem Auswahlmenü (oben Mitte). (Screenshot)

Das ProBuilder Menü brachte – ähnlich wie gängige 3D-Modellierungssoftware – verschiedene Werkzeuge zur Erstellung und Bearbeitung von Formen mit sich. Die Konstruktion grundlegender Geometrien wurde insbesondere durch drei zentrale Werkzeuge ermöglicht:

- **Shape-Werkzeug**: Erlaubt die Erstellung geometrischer Standardkörper, die als Ausgangsform für komplexere Modelle genutzt werden können (z.B. eine Kugel, ein Zylinder etc.).
- **Bezier Shape-Werkzeug**: Erzeugt röhrenartige Strukturen mit Bezier-Punkten an den Enden, die eine flexible Anpassung der Form ermöglichen.
- **Poly Shape-Werkzeug**: Ermöglicht die freie Erstellung individueller, zunächst eckiger Formen, die im weiteren Verlauf weiterbearbeitet und verfeinert werden können.

Nach der Erstellung einer Grundform standen verschiedene Bearbeitungswerkzeuge zur Verfügung, um die Modellgeometrie gezielt anzupassen. Mithilfe der Subdivide- und Merge-Funktionen ließen sich beispielsweise Faces und Edges unterteilen oder zusammenführen, während das Smoothing Tool dazu diente, Oberflächen zu glätten und Übergänge weicher zu gestalten.

Darüber hinaus beinhaltete das ProBuilder Package weitere essenzielle Funktionen zur Modellbearbeitung. Der Material Editor ermöglichte einen schnellen Zugriff auf zuvor erstellte Materials, so dass Texturen effizient zugewiesen werden konnten. Ergänzend dazu erlaubte der UV Editor eine präzise Anpassung der Texturplatzierung auf den Faces, um ein realistisches Erscheinungsbild des Modells zu gewährleisten.

Blumentopf

Zu Beginn der Modellerstellung wurde der Blumentopf als grundlegendes Objekt modelliert. Die Basisform setzte sich aus zwei Zylindern zusammen: Ein Zylinder bildete den eigentlichen Topfkörper, dessen Unterseite durch das manuelle Hinzufügen zusätzlicher Faces geschlossen wurde. Der zweite Zylinder wurde als oberer Rand aufgesetzt und entsprechend in der Breite angepasst, um die typische Silhouette eines Blumentopfs zu erzeugen.

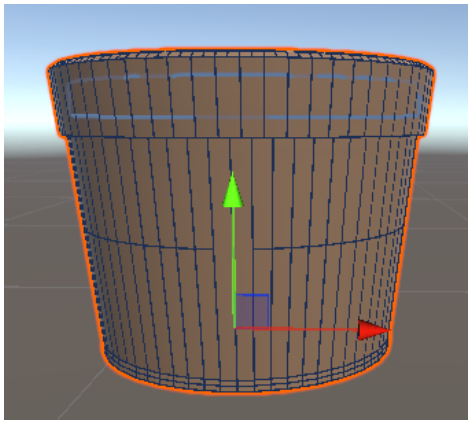


Abb. 5.2: Blumentopf. (Screenshot)

Durch gezielte Skalierung sowie den Einsatz des Smoothing Tools konnte die Form weiter verfeinert und die Oberfläche geglättet werden, um ein realitätsnahes Erscheinungsbild zu erzielen. Das verwendete Material wurde einfach gehalten und lediglich mit einer für Keramiktöpfe charakteristischen Farbgebung versehen. Der Pivot Point wurde mittig an der Unterseite des Objekts platziert. Diese Positionierung ermöglicht es, den Topf später bei einer Skalierung stets relativ zu seiner Standfläche zu verändern, ohne die Platzierung im Raum zu beeinflussen.

Erde

Für die Darstellung der Erde innerhalb des Blumentopfes wurde zunächst ein ProBuilder-Mesh in Form eines flachen Zylinders erstellt. Um die Oberfläche möglichst realistisch und unregelmäßig wirken zu lassen, wurde eine Verformung basierend auf dem Perlin-Noise implementiert. Dabei handelt es sich um ein algorithmisch erzeugtes Rauschmuster aus Float-Werten auf einer zweidimensionalen Ebene, das wellenförmige Strukturen erzeugt und daher häufig zur Erzeugung natürlicher Oberflächenformen verwendet wird.[39, 40]

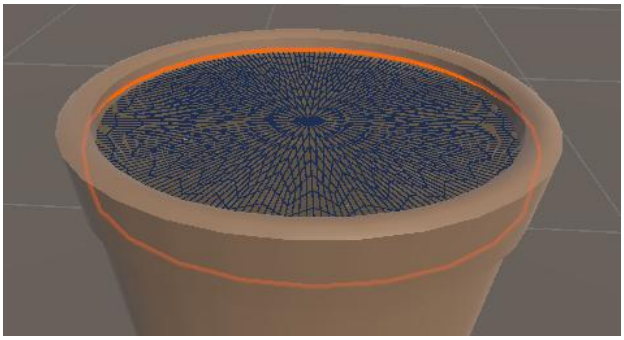


Abb. 5.3: Zylinder-Mesh der Erde. (Screenshot)

Zur Anwendung dieser Verformung wurde ein eigenes C#-Skript (siehe Anhang A.1: Erde_2.cs) erstellt und dem Zylinder-Mesh zugewiesen. Durch das Attribut `ExecuteAlways` wurde sichergestellt, dass alle Veränderungen unmittelbar im Szenenfenster sichtbar waren, ohne den Spielmodus starten zu müssen. Die Skalierung, Stärke und der Versatz der Verformung wurden als öffentliche Variablen festgelegt, so dass sie, falls notwendig oder gewünscht, im Unity Inspector angepasst werden konnten. Um jederzeit auf den ursprünglichen Zustand des Meshes zurückgreifen zu können, wurden in der `OnEnable`-Methode die ursprünglichen Positionsdaten des Meshes gespeichert (`originalPositions`).

Die Methode `OnValidate` wurde automatisch bei Änderungen im Inspector ausgeführt. Sie überprüfte, ob ein ProBuilder Mesh vorhanden ist, und wendete bei Bedarf die Verformung neu an.

[39] Mount, D. (2018). Procedural Generation: Perlin Noise. In *CMSC 425: Lecture 14*.
[40] Unity Technologies (Hrsg.). (o. D.-b). *Unity - Scripting API: MathF.PerlinNoise*. Unity Documentation. Abgerufen am 25. April 2025, von https://docs.unity3d.com/6000.0/Documentation/ScriptReference/Mathf.PerlinNoise.html

Die eigentliche Verformung erfolgte in der Methode `DeformWholeMesh`. Hier wurde zunächst eine Kopie der ursprünglichen Vertex-Positionen erzeugt (siehe Abb. 5.4).

```
Vector3[] verts = new Vector3[originalPositions.Length];
for (int i = 0; i < originalPositions.Length; i++)
{
    verts[i] = originalPositions[i];
}
```

Abb. 5.4: Kopiert die gespeicherten Originalpositionen. (Auszug aus Erde_2.cs)

Anschließend wurde die y-Komponente jedes einzelnen Punkts des Meshes auf Basis der Perlin-Noise-Werte modifiziert (siehe Abb. 5.5).

```
Vector3 v = verts[i];
float noiseValue = Mathf.PerlinNoise(
    (v.x + offsetX) * noiseScale,
    (v.z + offsetZ) * noiseScale
);
float delta = (noiseValue - 0.5f) * noiseStrength;
v.y += delta;
verts[i] = v;
```

Abb. 5.5: Wendet Perlin-Noise auf jede Vertex-Position an. (Auszug aus Erde_2.cs)

Zusätzlich wurde eine `ResetMesh`-Methode implementiert, die auf `originalPositions` zugriff, um das Mesh bei Bedarf auf seinen ursprünglichen Zustand zurückzusetzen. Hierbei wurde zur Verbesserung des Workflows im Editor ein zweites Skript (siehe Anhang A.2: Erde_2Editor.cs) erstellt. Es erweiterte die Inspector-Ansicht für Objekte, die das Erde_2-Skript enthielten, indem es die regulären Felder darstellte und zusätzlich einen „Reset Mesh“-Button einfügte. Über diesen Button konnte das Mesh direkt im Editor auf seinen Ursprungszustand zurückgesetzt werden, ohne den Code manuell auszuführen.

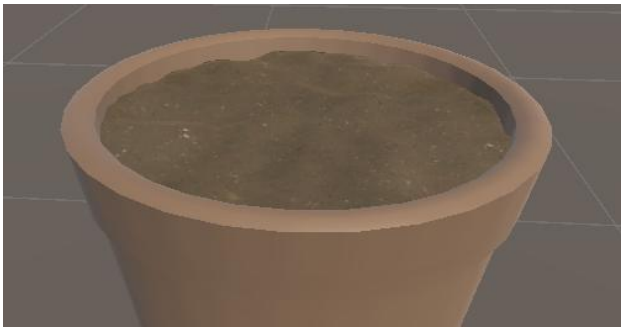


Abb. 5.6: Verformte Erde. (Screenshot)

Die Oberfläche der Erde konnte demnach flexibel angepasst und bei Bedarf in ihren Ausgangszustand zurückversetzt werden. Abschließend wurde dem Mesh ein Material mit einer erdtypischen Textur zugewiesen, um die visuelle Darstellung zu vervollständigen.

Pflanzenstängel

Die Pflanzenstängel bestanden aus länglichen, dünnen Zylindern, die mithilfe von ProBuilder erstellt wurden. Um eine gezielte Verformung zu ermöglichen, wurde das Zylinder-Mesh entlang der Längsachse in mehrere Segmente unterteilt. Die einzelnen Ringe des Zylinders konnten anschließend individuell verschoben und skaliert und auf diese Weise fünf unterschiedlich geformte Stängel modelliert werden. Die fünf Stängelvarianten dienten im weiteren Verlauf als Grundformen, aus denen je nach Bedarf gewählt werden konnte. Jede dieser Varianten ließ sich flexibel skalieren und bei Bedarf zusätzlich in verschiedene Richtungen verformen, um eine größere gestalterische Vielfalt zu ermöglichen.

Zur optischen Gestaltung erhielten alle Stängel ein einheitliches Material mit einer dunkelgrünen Farbgebung.

Blätter

Die Modellierung der Blätter stellte im Vergleich zu den Stängeln eine etwas größere gestalterische Herausforderung dar, da eine komplexere Formgebung erforderlich war. Als Ausgangsform diente ein Torus (siehe Abb. 5.7), der anschließend durch gezielte Verformungen, Skalierungen sowie das Entfernen und Wiederhinzufügen einzelner Vertices, Edges und Faces schrittweise in eine blattähnliche Form abgewandelt wurde.

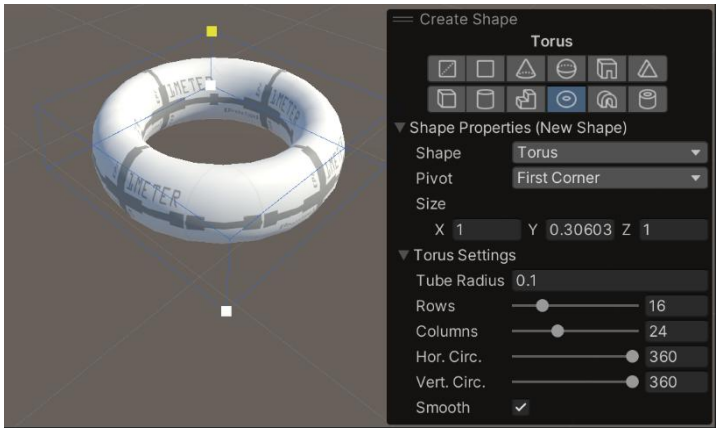


Abb. 5.7: Torus als Ausgangsform. (Screenshot)

Aus einer zunächst geraden und symmetrischen Blattform wurden anschließend weitere Varianten abgeleitet. Insgesamt entstanden zehn unterschiedliche Grundformen, die sich hinsichtlich Größe, Breite und Krümmung an den Enden unterschieden. Diese Grundformen konnten – analog zum Vorgehen bei den Stängeln – im späteren Verlauf weiter skaliert oder individuell angepasst werden, um so eine noch größere visuelle Vielfalt zu ermöglichen.

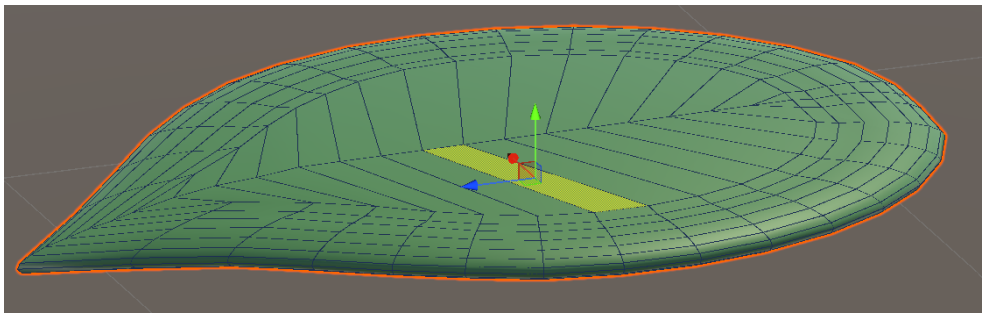


Abb. 5.8: Mesh eines Blattes ohne finale Textur. (Screenshot)

Für die Oberflächengestaltung wurde eine eigene Textur in Adobe Illustrator entworfen. Diese orientierte sich farblich an dem dunkelgrünen Material der Stängel, enthielt jedoch ein für Blätter typisches Muster, das eine längs über das Blatt verlaufende Linie abbildete. Als alternative Variante wurde eine zweite Textur erstellt, die zusätzlich zu der zentralen Längslinie feine Querlinien enthielt.

UV-Mapping

Wie im vorherigen Abschnitt beschrieben, wurde für die Blätter eine zweidimensionale Textur erstellt, die mittels UV-Mapping auf die 3D-Modelle projiziert werden musste. Für diesen Prozess stellte ProBuilder einen integrierten UV-Editor zur Verfügung, der eine präzise Zuweisung der Texturkoordinaten zu den jeweiligen Faces ermöglichte.

Da die Blätter aus einer Vielzahl einzelner Faces bestanden und eine geringe Anzahl an Faces das UV-Mapping erheblich erleichterte, war es zunächst erforderlich, diese so weit wie möglich zu größeren Flächen zusammenzufassen.

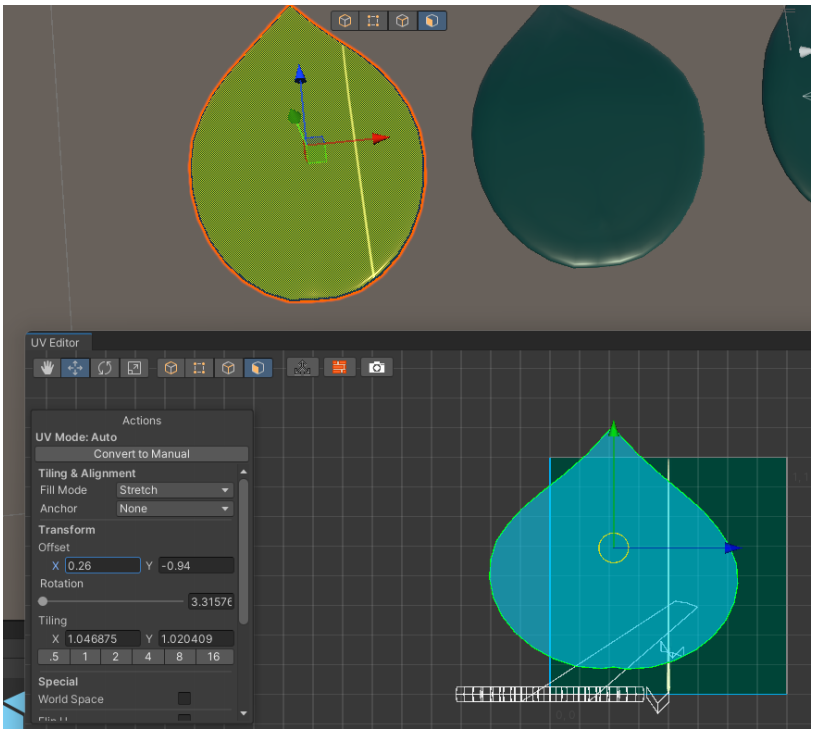


Abb. 5.9: Arbeit im UV-Editor; Faces wurden pro Seite zu einer Fläche zusammengefügt. (Screenshot)

Besondere Aufmerksamkeit erforderte das UV-Mapping bei Blättern mit unregelmäßiger Geometrie oder starken Biegungen. In diesen Fällen musste die Zusammenführung der Faces gezielt an die jeweilige Verformung des Modells angepasst werden, um eine möglichst verzerrungsfreie und stimmige Texturprojektion zu gewährleisten (siehe Abb. 5.10).

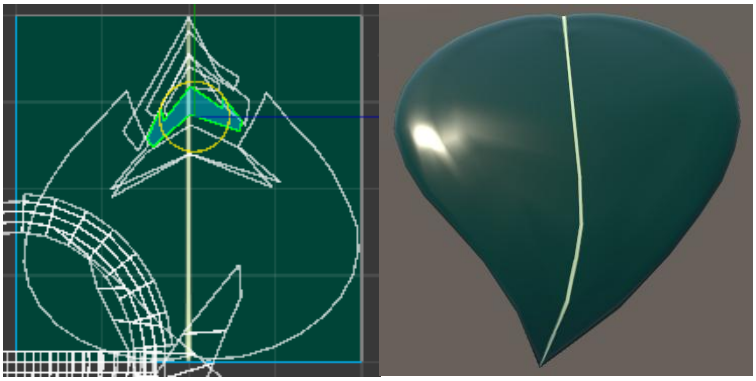


Abb. 5.10: Blatt mit einem krummen Verlauf. Links: UV-Editor, rechts: Scene View. (Screenshot)

Zusammensetzen der Pflanze

Ein zentrales Ziel bei der Modellierung der Pflanze war die Gewährleistung von Modularität. Die Pflanze sollte aus einzelnen, austauschbaren Bausteinen bestehen, um eine möglichst hohe Flexibilität in der Gestaltung und bei gewünschten Änderungen zu ermöglichen. Konkret wurde die Pflanze aus drei verschiedenen Pflanzenbausteinen zusammengesetzt: einem Stängel mit einem Blatt, mit zwei Blättern oder mit drei Blättern (siehe Abb. 5.11). Diese drei Bausteine wurden wiederholt in immer wieder abgewandelten Versionen eingesetzt.

Abb. 5.11: Drei Pflanzenbausteine, mit einem, zwei oder drei Blättern. (Screenshot)

Sowohl bei den Stängeln als auch bei den einzelnen Blättern konnte über den Unity-Inspector zwischen vordefinierten Varianten gewählt werden. Wie in den vorangegangenen Abschnitten beschrieben, standen hierfür fünf unterschiedliche Stängelformen sowie zehn verschiedene Blattvarianten zur Verfügung.

Zur technischen Umsetzung wurde ein C#-Skript mit dem Namen LeafSwitcher.cs entwickelt (siehe Anhang A.3). Dieses Skript definierte zunächst ein leeres Array, das über den Inspector mit den gewünschten Prefabs befüllt werden konnte. Jeder Eintrag i m Array erhielt dabei einen eigenen Index. Über die Variable `selectedLeafIndex` konnte im Inspector festgelegt werden, welche Variante ausgewählt und angezeigt werden sollte. Sobald eine Änderung erfolgte, reagierte das Skript mittels der Methode `OnValidate`. In einer verzögerten Ausführung wurde dabei geprüft, welche Variante aktuell ausgewählt war. Zunächst wurde das bisher angezeigte Element entfernt, bevor das neue, dem aktuellen Index entsprechende Prefab instanziiert wurde. Dadurch wurde sichergestellt, dass jeweils nur eine Variante aktiv dargestellt wurde.

```
private void UpdateLeaf()
{
    if (leafPrefabs == null || leafPrefabs.Length == 0)
        return;

    selectedLeafIndex = Mathf.Clamp(selectedLeafIndex, 0, leafPrefabs.Length - 1);

    ClearExistingLeaf();

    if (leafPrefabs[selectedLeafIndex] != null)
    {
     GameObject newLeaf = Instantiate(leafPrefabs[selectedLeafIndex], transform);
        newLeaf.transform.localPosition = Vector3.zero;
        newLeaf.transform.localRotation = Quaternion.identity;
        newLeaf.transform.localScale = Vector3.one;
    }
}
```

Abb. 5.12: Prüft, ob es Blätter gibt, korrigiert den Index, entfernt das alte Blatt, erzeugt das neue Blatt. (Auszug aus LeafSwitcher.cs)

Nachdem das LeafSwitcher-Skript in den Vorlagen für Stängel und Blätter integriert wurde, konnten daraus die beschriebenen drei Bausteine erstellt werden. Für die korrekte Positionierung wurden die Pivot Points der Einzelkomponenten jeweils an dem Punkt ausgerichtet, an dem Stängel und Blätter aufeinandertreffen. So blieb auch bei einem Wechsel unterschiedlich großer Varianten stets die Verbindung zwischen den Elementen erhalten. Die zusammengesetzten Bausteine selbst erhielten wiederum ihren Pivot Point am unteren Ende des Stängels, so dass sie sich problemlos und präzise in den Blumentopf einsetzen ließen und auch bei einer möglichen Skalierung weiterhin an dem für sie vorgesehenen Platz befanden.

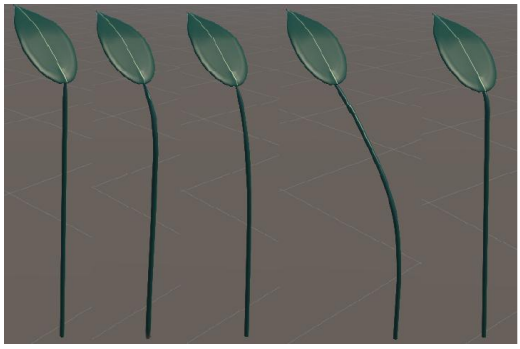


Abb. 5.13: Fünf Stängelvarianten. (Screenshot)

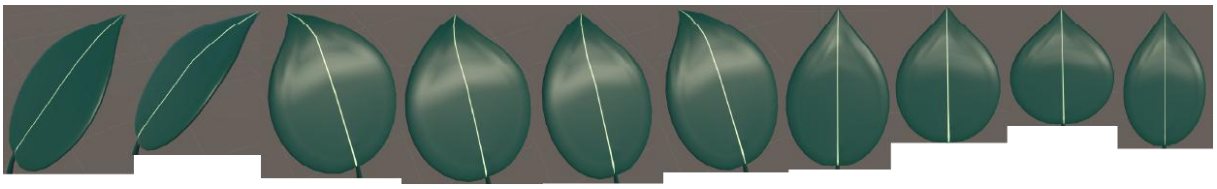


Abb. 5.15: Zehn Blattvarianten. (Screenshot)

Abschließend wurde die vollständige Pflanze durch Kombination und Anordnung der vorbereiteten Bausteine innerhalb des Topfes zusammengesetzt. Mithilfe von Skalierungs- und Rotations-Tools konnten zahlreiche visuell unterschiedliche Pflanzenteile erzeugt und flexibel zu einer variantenreichen Pflanze kombiniert werden (siehe Abb. 5.14).

Abb. 5.14: Fertiges Pflanzenmodell. (Screenshot)

5.1.3 Herausforderungen und Lösungen

Die Umsetzung eines komplexen, modularen und anpassbaren 3D-Modells innerhalb der Unity Engine stellte verschiedene technische und gestalterische Herausforderungen dar. Ziel war es, mit dem integrierten Unity Package ProBuilder ein vollständiges Pflanzenmodell zu erstellen, das sich durch eine hohe Flexibilität hinsichtlich Formvariationen, Austauschbarkeit einzelner Komponenten sowie einer realitätsnahen visuellen Darstellung auszeichnet.

ProBuilder bot eine direkt in Unity integrierte Umgebung zur Erstellung und Bearbeitung von 3D-Geometrien. Die Integration ermöglichte zwar einen nahtlosen Arbeitsablauf innerhalb der Engine, brachte jedoch auch gewisse Herausforderungen mit sich, insbesondere im Hinblick auf die Präzision und Bedienlogik im Vergleich zu herkömmlichen 3D-Modellierungstools wie bspw. Blender.

Die Konzeption eines modular aufgebauten Modells stellte darüber hinaus eine zusätzliche Herausforderung dar. Um die Austauschbarkeit einzelner Pflanzenteile zu gewährleisten, mussten sämtliche Komponenten so gestaltet und organisiert werden, dass sie sich flexibel kombinieren und individuell anpassen ließen.

Ein weiterer relevanter Aspekt betraf die Performance der Anwendung. Insbesondere bei einer zunehmenden Anzahl an Variationen und instanziierten Objekten stieg die Komplexität des Modells schnell an, was sich negativ auf die Rechenleistung und die Echtzeitdarstellung im Editor auswirkte.

Die folgenden Abschnitte gehen genauer auf die wichtigsten Herausforderungen ein, die im Verlauf des Projekts auftraten, sowie auf die jeweils entwickelten Lösungen zur erfolgreichen Umsetzung des Vorhabens.

Umgang mit ProBuilder

Die Nutzung von ProBuilder als Modellierungswerkzeug stellte zu Beginn des Projekts eine große Herausforderung dar. Im Vergleich zu etablierten 3D-Modellierungsprogrammen wie z.B. Blender war der vergleichsweise eingeschränkte Funktionsumfang ungewohnt, die Bedienlogik teilweise unintuitiv, und die verfügbaren Werkzeuge mussten zunächst durch eigenständiges Ausprobieren erschlossen werden. Die Einarbeitungsphase gestaltete sich dementsprechend langwierig, und der Arbeitsprozess verlief zunächst stockend. Viele Funktionen, die anfangs nicht umsetzbar erschienen, erwiesen sich im weiteren Verlauf doch als realisierbar, was dazu führte, dass sich die Arbeit mit zunehmender Erfahrung deutlich flüssiger gestalten ließ.

Ein Beispiel hierfür war die Verwendung von Shortcuts. Zu Beginn wurde vollständig ohne Tastenkombinationen gearbeitet, was den Modellierungsprozess erheblich verlangsamte. Erst später wurden eigene Shortcuts über die Unity Preferences definiert und die Effizienz so deutlich gesteigert.

Auch bei der Anwendung einzelner Tools traten anfangs wiederholt Schwierigkeiten auf. So wurde etwa das Subdivision Tool zunächst auf ganze Meshes angewendet, was zu einer unnötig hohen Anzahl an Faces und einer damit verbundenen Komplexitätssteigerung führte. Erst mit fortschreitender Erfahrung wurde erkannt, wie sich gezielt Subdivision auf ausgewählte Bereiche anwenden ließ.

Einige Funktionen von ProBuilder konnten hingegen bis zum Projektende nicht zufriedenstellend genutzt werden. Dazu zählte beispielsweise die Auswahl von Vertices, Edges und Faces als Loop oder Ring. Trotz intensiver Versuche blieb der Einsatz dieser Werkzeuge fehleranfällig oder unzuverlässig.

Während des gesamten Projekts bot in diesen Fällen die umfangreiche offizielle Unity-Dokumentation zu ProBuilder eine umfangreiche Unterstützung. Sie ermöglichte es, viele der auftretenden Probleme zu lösen oder zumindest besser zu verstehen, auch wenn nicht alle Werkzeuge erfolgreich in den Arbeitsprozess integriert werden konnten.

Komplexität, Modularität und Modifizierbarkeit des Modells

Eine zentrale Herausforderung bei der Entwicklung des Prototyps bestand in der Frage, wie das Modell so aufgebaut werden kann, dass es den Anforderungen an Modularität und Modifizierbarkeit gerecht wird. Die zugrunde liegende Idee war es, eine Struktur zu schaffen, bei der einzelne Bestandteile – insbesondere Blätter und Stängel – flexibel austauschbar, skalierbar und kombinierbar sind. Dabei wurde schnell deutlich, dass bestimmte Funktionalitäten ausschließlich durch den gezielten Einsatz von C#-Skripten realisierbar waren. Dies erforderte eine frühzeitige konzeptionelle Auseinandersetzung mit der Frage, welche Modellbestandteile mit welcher Art von Skript ausgestattet werden müssen.

Im Zuge des Modellaufbaus wurden verschiedene Strukturansätze getestet und teilweise wieder verworfen. Ein früher Ansatz sah vor, ein übergeordnetes GameObject („Pflanzenteil“) zu definieren, das sowohl Stängel- als auch Blatt-Vorlagen enthielt. Ein zentrales Skript sollte hier sämtliche Modifikationen steuern – darunter Listen zur Auswahl der Vorlagen, Einstellungen zum Skalieren, Rotieren und Positionieren sowie Funktionen zur zufälligen Auswahl von Blättern und zufälliger Rotation. Dieses Konzept erwies sich jedoch in der praktischen Anwendung als problematisch, da die zufälligen Rotationen häufig zu unnatürlichen und unrealistischen Darstellungen führten und die Gesamtstruktur des Skripts nicht zu einer verbesserten Kontrolle oder Übersichtlichkeit beitrug.

In der Folge wurde der Aufbau vereinfacht und stärker an den verfügbaren Werkzeugen von ProBuilder ausgerichtet. Die Standardfunktionen zur Skalierung, Rotation und Positionierung wurden manuell über die ProBuilder Oberfläche umgesetzt, wobei definierte Shortcuts die Bedienung effizient gestalteten.

Anstelle eines komplexen zentralen Skripts wurde schließlich ein Switcher-Skript auf Ebene der einzelnen Bausteine implementiert (Anhang A.3: LeafSwitcher.cs). Dieses wurde jeweils separat bei den Stängeln und Blättern angewendet und ermöglichte dort einen gezielten Wechsel zwischen verschiedenen Vorlagen. Die Struktur führte nicht nur zu einer besseren Modularität, sondern auch zu einer höheren Kontrolle über die Darstellung der jeweiligen Komponenten.

Auch weitere modelltechnische Herausforderungen, wie die korrekte Platzierung der Pivot Points zur Sicherstellung einer sauberen Skalierung und Kombination der Elemente, konnten im Verlauf des Projekts schrittweise gelöst werden. So entstand ein finaler Modellaufbau, der eine flexible und realistische Zusammenstellung variabler Pflanzenelemente ermöglichte.

Performance

Bereits zu Beginn des Projekts war absehbar, dass die Umsetzung eines detailreichen, modular aufgebauten 3D-Modells in Unity zu einem komplexen und kleinteiligen Mesh führen würde. Diese Komplexität stellte insbesondere in der Entwicklungsphase eine erhebliche Herausforderung für die Performance der Anwendung dar.

Die ersten Arbeitsschritte erfolgten an einem PC ohne Grafikkarte. Aufgrund der hohen Anzahl an Geometrieeinheiten und den damit verbundenen Systemanforderungen führte dies bereits bei grundlegenden Interaktionen – etwa beim Auswählen oder Verschieben von Objekten – zu erheblichen Verzögerungen. Oft kam es zu Latenzen im Sekunden- bis Minutenbereich oder gar zu Programmabstürzen, was einen kontinuierlichen Arbeitsfluss nahezu unmöglich machte. Ein Wechsel auf ein leistungsfähigeres System mit Grafikkarte führte zu einer deutlichen Verbesserung der Reaktionsgeschwindigkeit und der allgemeinen Arbeitsstabilität innerhalb der Unity-Umgebung.

Obwohl der Schwerpunkt des Projekts nicht auf der Entwicklung einer interaktiven Anwendung, sondern auf der Erstellung eines 3D-Modells für eine potenzielle Anwendung lag, war es dennoch wichtig, die Auswirkungen des Modells auf die Systemressourcen zu überprüfen. Um sicherzustellen, dass das Modell auch in einer realen Anwendungssituation effizient eingebunden werden kann, wurde beim Start der Anwendung gezielt die Anzahl der sogenannten Batches überprüft. Batches bezeichnen in Unity Mengen von Draw Calls, also von Zeichenbefehlen an die Grafik-API. Diese enthalten sämtliche Informationen, die benötigt werden, um ein Objekt auf dem Bildschirm darzustellen – darunter Texturen, Shader und Pufferdaten – und können somit je nach Komplexität der Szene sehr ressourcenintensiv sein. Ziel des Batching ist es, möglichst viele dieser Draw Calls zusammenzufassen, um die Zeichenoperationen effizienter durchzuführen. Je höher die Anzahl an Batches, desto mehr Ressourcen werden beim Rendern beansprucht, insbesondere auf Geräten mit begrenzter Rechenleistung.[41, 42, 43]

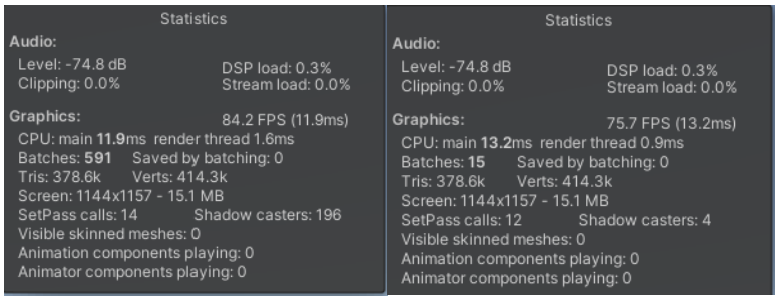


Abb. 5.16: Ohne Mesh Baker 591 Batches (links) und mit Mesh Baker 15 Batches (rechts). (Screenshot)

Die Überprüfung der Anwendung ergab, dass das Pflanzenmodell einer sehr hohen Batch-Anzahl bedurfte. Um auf diese Problematik zu reagieren, wurde die kostenlose Version des Unity Packages Mesh Baker eingesetzt. Durch die Zusammenführung von Geometrie und Texturen reduziert Mesh Baker effektiv die Anzahl der erforderlichen Draw Calls, was zu einer verbesserten Rendering-Leistung führt. Mit Hilfe des Mesh Bakers ließen sich nach Abschluss der Modellierung die separaten Meshes zu einem

[41] Mollah, S. (2022, 5. Januar). Unity3D Draw Call Batching and how it works? - Saharukh Mollah - Medium. *Medium*. https://saharukh.medium.com/unity3d-draw-call-batching-and-how-it-works-7624ce5bc255

[42] Unity Technologies (Hrsg.). (2024f). *Unity - Manual: Introduction to optimizing draw calls*. Abgerufen am 27. April 2025, von https://docs.unity3d.com/6000.0/Documentation/Manual/optimizing-draw-calls.html?utm_source=chatgpt.com

[43] Digital Opus (Hrsg.). (o. D.). *Mesh Baker – Digital Opus*. Abgerufen am 27. April 2025, von https://digitalopus.ca/site/mesh-baker/

vereinfachten Gesamt-Mesh zusammenführen. Durch diesen Schritt konnte die Anzahl der Batches im Projekt signifikant reduziert werden – von ursprünglich 591 auf lediglich 15 (siehe Abb. 5.16).

5.2 Zweite Phase: Studie

Da ein zentrales Anliegen dieser Arbeit die Bewertung eines Feedbackprozesses im Rahmen der Entwicklung eines 3D-Modells ist, war es notwendig, den Prozess unter praxisnahen Bedingungen zu testen. Zu diesem Zweck wurde ein Anwendungstest mit anschließender Befragung durchgeführt. An der Erhebung nahmen insgesamt zehn Personen teil. Es handelte sich dabei nicht um eine repräsentative Umfrage im statistischen Sinne, sondern vielmehr um eine qualitative Erhebung mit dem Ziel, ein Meinungsbild zu erzeugen sowie gezieltes Feedback zum Modell und Feedbackprozess zu erhalten.

Die Teilnehmenden setzten sich aus zwei Gruppen zusammen: Fünf Personen, die beruflich im Bereich der 3D-Modellierung oder der Entwicklung virtueller Anwendungen tätig waren und daher über fundierte Fachkenntnisse verfügten – nachfolgend „Experten“ genannt – sowie fünf Personen ohne jegliche Vorerfahrung in besagtem Bereich. Die Teilnehmenden waren zwischen 25 und 32 Jahre alt; dabei lag das Alter der Experten zwischen 28 und 32 Jahren und das der fachfremden Personen zwischen 25 und 29 Jahren. Hinsichtlich des Geschlechts ergab sich folgende Verteilung: Die Gruppe der Experten bestand ausschließlich aus männlichen Personen, während in der Gruppe der fachfremden Teilnehmenden vier Frauen und ein Mann vertreten waren. Bezüglich des Alters und Geschlechts erfolgte die Auswahl der Personen dabei nicht nach einem bestimmten Schema, sondern zufällig. Entscheidend für die Auswahl der Experten war in erster Linie ihre gute Verfügbarkeit sowie die Tatsache, dass ihre Fachkenntnisse im Bereich 3D-Modellierung bekannt waren. Die fachfremden Personen waren vorwiegend im Marketing- oder Medienbereich tätig oder hatten durch ihre Rolle als Kunden von Agenturen praktische Erfahrung mit Feedbackprozessen.

Allen Teilnehmenden wurde das entwickelte Modell vorgestellt, wobei insbesondere die Aspekte der Modularität und Modifizierbarkeit im Fokus standen. Darüber hinaus wurden sie gebeten, sich in die Rolle potenzieller Kunden hineinzuversetzen und den vorgesehenen Feedbackprozess unter den entsprechenden Gesichtspunkten zu beurteilen. Von den Experten wurde darüber hinaus eine technische Einschätzung zur Umsetzung in Unity erwartet – insbesondere im Vergleich zu herkömmlichen Modellierungs- und Entwicklungsprozessen. Ziel war es, potenzielle Verbesserungsvorschläge zu identifizieren, die – soweit zeitlich möglich – noch in die Weiterentwicklung des Modells einfließen konnten.

5.2.1 Vorbereitung

Die Vorbereitung der Befragung umfasste zwei zentrale Bestandteile: Zum einen wurden Überlegungen zum Ablauf des Praxistests angestellt, zum anderen wurde ein strukturierter Fragebogen entwickelt, der im Anschluss an den Test in Form einer Online-Umfrage ausgefüllt werden sollte.

Der Fragebogen war so konzipiert, dass er sowohl allgemeine als auch zielgruppenspezifische Informationen erfasste. Zu Beginn wurden grundlegende Personendaten, wie der Beruf der Teilnehmenden sowie deren etwaige Vorerfahrung im Bereich der 3D-Modellierung und VR-Anwendungsentwicklung erfragt. Im Anschluss folgten gezielte Fragen zur Bewertung des vorgestellten Modells sowie zum zugrunde liegenden Feedbackprozess.

Bei der Gestaltung der Bewertungsfragen wurde darauf geachtet, die Skalen entsprechend dem jeweiligen Fragentyp anzupassen. Bei der Bewertung des Modells und des Feedbackprozesses wurde eine Tendenz in die positive oder negative Richtung erwartet und daher eine Skala verwendet, die keine neutrale Mittelposition besaß. Ziel war es, eine schwache oder eindeutige Zustimmung oder Ablehnung zu bestimmten Aussagen zu ermöglichen. Anders verhielt es sich bei den Fragen zum Entwicklungsprozess, die ausschließlich von den Experten beantwortet werden sollten. Da diese den Prozess selbst nicht aktiv durchlaufen hatten, sondern lediglich auf Grundlage der Vorstellung und Erläuterung eine Einschätzung abgeben konnten, wurde hier bewusst eine neutrale Antwortmöglichkeit angeboten. So konnte vermieden werden, dass Teilnehmende zu einer Tendenz gezwungen wurden, obwohl ihnen dafür ggf. eine belastbare Grundlage fehlte.

Besonderer Wert wurde darauf gelegt, dass die Teilnehmenden nicht nur vorgegebene Antwortoptionen auswählten, sondern darüber hinaus die Möglichkeit hatten über Freitextfelder auch eigene Gedanken einzubringen und Bewertungen näher zu erläutern. Abschließend enthielt der Fragebogen einen zusätzlichen Abschnitt, der ausschließlich von den Experten beantwortet werden sollte. Dieser Teil bezog sich auf die Einschätzung des Herstellungsprozesses des 3D-Modells und bot Raum für technische Bewertungen sowie Verbesserungsvorschläge.

Auf Grundlage dieser Überlegungen wurde die Umfrage ausgearbeitet (siehe Anhang B.1).

5.2.2 Durchführung

Die Durchführung des Praxistests erfolgte im Rahmen eines persönlichen Gesprächs mit jeder teilnehmenden Person und bestand aus mehreren fest definierten Schritten. Ziel war es, den Teilnehmenden eine fundierte Einführung in das Projekt zu geben und anschließend strukturierte Rückmeldungen über eine Online-Umfrage zu erheben. Eine Durchführung in Virtual Reality war dabei nicht vorgesehen oder notwendig. Stattdessen bestand der Ansatz darin, dass sowohl die durchführende als auch die teilnehmende Person gemeinsam auf denselben Bildschirm bzw. dasselbe Softwarefenster blickten. Der Fokus lag somit nicht auf der immersiven Nutzung der Anwendung, sondern auf dem gemeinsamen Betrachten und Bewerten des Modells sowie des zugehörigen Feedbackprozesses. Dementsprechend konnte die Durchführung des Tests flexibel entweder online – per Bildschirmübertragung – oder in Präsenz stattfinden, wobei die Teilnehmenden nebeneinander vor dem gleichen Bildschirm saßen. Die Entscheidung hierüber richtete sich nach der individuellen Präferenz der teilnehmenden Person.

Der Ablauf gestaltete sich wie folgt:

1. Einführung in das Thema:
   Zunächst wurde das übergeordnete Thema der Arbeit vorgestellt und in den Kontext eingeordnet.

2. Vorstellung der Bewertungsziele der Studie:
   Die Teilnehmenden wurden darüber informiert, welche Aspekte im Fokus der Befragung stehen:

   - Ist das Modell modular und modifizierbar?
   - Wie effizient und verständlich ist der vorgesehene Feedback-Prozess aus Sicht potenzieller Kunden?
   - Für Experten: Wie wird der vorgestellte Entwicklungs-Prozess bewertet?

3. Präsentation des Modells:
   Im Anschluss wurde das entwickelte 3D-Modell vorgestellt und hierbei insbesondere der technische Aufbau sowie die verfügbaren Anpassungsmöglichkeiten erläutert. Die Teilnehmenden schauten dabei gemeinsam mit der ausführenden Person auf einen Monitor.

4. Rückfragen und individuelle Anpassungen:
   Die Teilnehmenden hatten die Möglichkeit, Fragen zu stellen und konkrete Änderungswünsche zu äußern. Diese wurden – soweit möglich – direkt umgesetzt und demonstriert, um eine realitätsnahe Einschätzung des Feedback-Prozesses zu ermöglichen.

5. Ausfüllen der Online-Umfrage:
   Abschließend füllten die Teilnehmenden den vorbereiteten Fragebogen in Form einer Online-Umfrage aus. Die Antworten bildeten die Grundlage für die anschließende Auswertung und die Weiterentwicklung des Modells (siehe Anhang B.2: Ausgefüllte Fragebögen).

5.2.3 Auswertung

Das Ziel der Online-Umfrage war es, gezielte Rückmeldungen zum entwickelten Modell, dem Feedbackprozess sowie zum gewählten Entwicklungsansatz mit ProBuilder in Unity zu erhalten. Die Auswertung erfolgte entlang dieser drei zentralen Themenbereiche der Befragung. Ergänzend dazu wird am Ende eine zusammenfassende Einschätzung des Gesamtprojekts dargestellt.

Bewertung des Modells

Im Rahmen der Befragung wurden die Teilnehmenden gebeten, zwei zentrale Aussagen zum entwickelten 3D-Modell hinsichtlich seiner Modifizierbarkeit und Modularität zu bewerten. Die Bewertung erfolgte anhand einer vierstufigen Skala mit den Antwortoptionen: *trifft zu*, *trifft teilweise zu*, *trifft eher nicht zu* und *trifft nicht zu*.

Aussage 1: „Das Modell ist modifizierbar."

Alle fünf fachfremden Teilnehmenden beurteilten diese Aussage mit *trifft zu* und sahen das Modell somit eindeutig als modifizierbar an. Unter den Experten stimmten zwei Personen der Aussage vollständig zu (*trifft zu*), während drei Personen eine eingeschränkte Zustimmung äußerten (*trifft teilweise zu*).

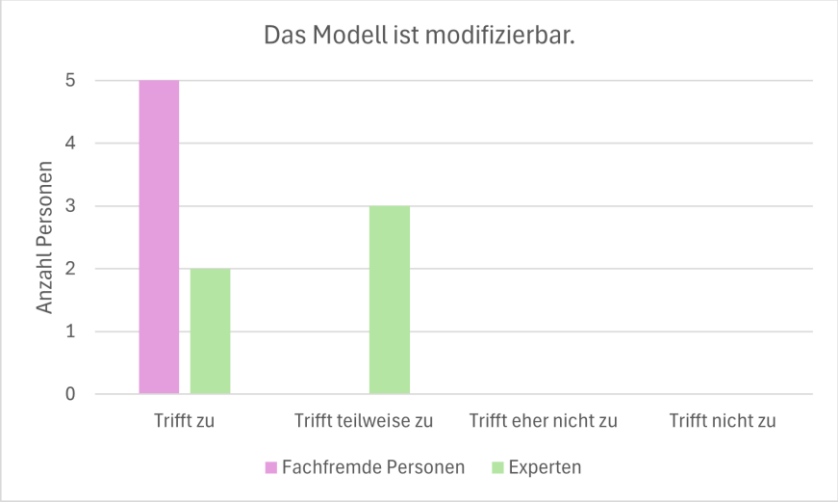


Abb. 5.17: Statistik zu Aussage 1. (Eigene Darstellung)

Aussage 2: „Das Modell ist modular."

Diese Aussage wurde von neun der zehn Teilnehmenden mit *trifft zu* bewertet. Lediglich ein Experte bewertete die Aussage mit *trifft teilweise zu*. Insgesamt wurde das Modell somit mehrheitlich als modular wahrgenommen.

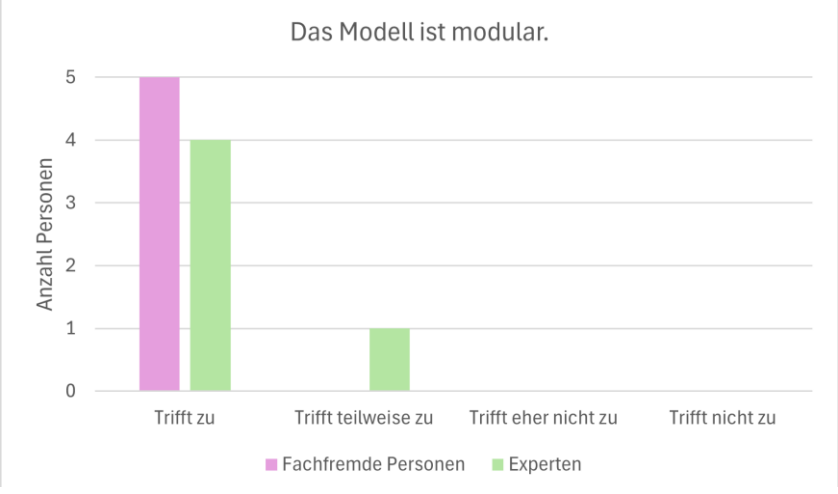


Abb. 5.18: Statistik zu Aussage 2. (Eigene Darstellung)

Zusätzlich wurden die Probanden gefragt, welche Anpassungsmöglichkeiten ihrer Meinung nach gefehlt hätten. Eine Antwort auf diese Frage war optional. Während bei den Experten jeder eine Anmerkung zu möglichen Verbesserungen hatte, äußerte sich von den fachfremden Personen lediglich eine. Aus den Aussagen ergaben sich die folgenden Optimierungsvorschläge:

- Prozedurale Logik ergänzen
    - Herstellung von Abhängigkeiten zwischen Parametern, z. B. trockene Erde bewirkt eine Veränderung der Blattfarbe.
    - Einbindung von einfachen Regelmechanismen zur Simulation realer Pflanzenverhalten
- Nutzeroberfläche (UI) zur Steuerung entwickeln
- Beschleunigung der Anpassung durch interaktive Steuerung, z.B. durch Verwendung von Schiebereglern zur:
    - Veränderung der Stängelanzahl
    - schnellen visuellen Variation des Modells
- Erweiterung der Modellvielfalt:
    - Auswahl unterschiedlicher Wachstumsformen und Pflanzentypen
- Schnellere Anpassung der geometrischen Form und Oberfläche der Blätter

Bewertung des Feedbackprozesses

Im Rahmen der Befragung wurde eine zentrale Aussage zur Effizienz des Feedbackprozesses bewertet. Die Teilnehmenden konnten dabei auf einer vierstufigen Skala angeben, inwieweit sie der folgenden Aussage zustimmen.

Aussage 3: „Anpassungen bzw. Änderungswünsche können schnell umgesetzt werden."

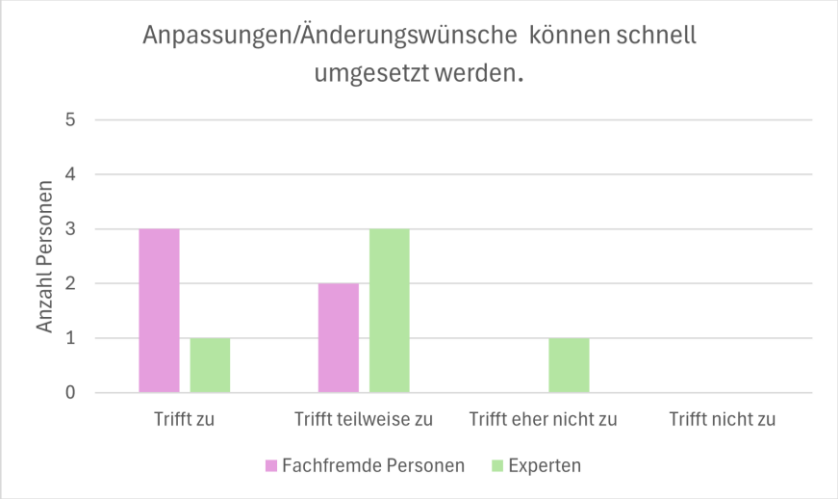


Abb. 5.19: Statistik zu Aussage 3. (Eigene Darstellung)

Die Rückmeldungen der fachfremden Personen fielen überwiegend positiv aus: Drei der fünf Befragten stimmten der Aussage vollständig zu (*trifft zu*), während zwei Personen eine eingeschränkte Zustimmung äußerten (*trifft teilweise zu*). Die Einschätzungen der Experten fielen differenzierter aus: Eine

Person stimmte der Aussage zu (*trifft zu*), drei bewerteten sie mit *trifft teilweise zu* und eine Person mit *trifft eher nicht zu*.

Zusätzlich wurde der Feedbackprozess anhand einer Punkteskala von 1 (sehr schlecht) bis 10 (sehr gut) bewertet. Die Ergebnisse dieser Einschätzung sind in Abbildung X dargestellt. Im Durchschnitt bewerteten die fachfremden Teilnehmenden den Feedbackprozess mit 9 Punkten, während die Experten eine durchschnittliche Bewertung von 7,2 Punkten abgaben.

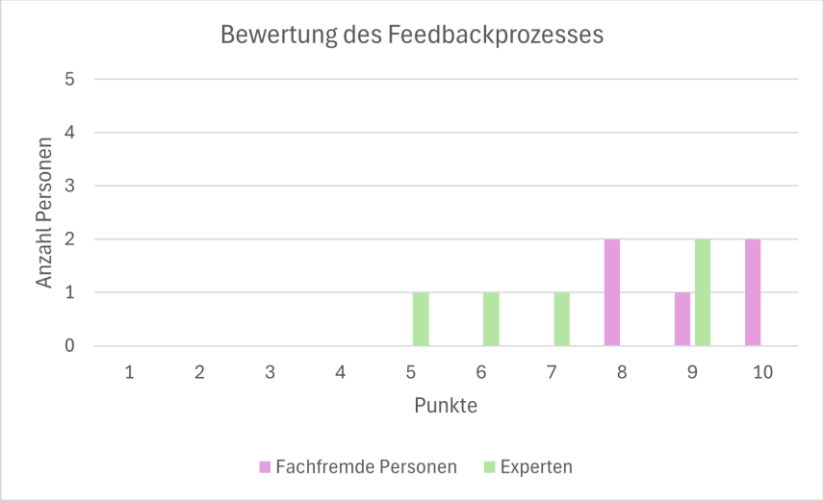


Abb. 5.20: Statistik zur Punktewertung des Feedbackprozesses. (Eigene Darstellung)

Die Rückmeldungen der fachfremden Teilnehmenden zum Feedback-Prozess des Modells fielen überwiegend positiv aus. Eine Person äußerte sich beispielsweise wie folgt: *„Grundsätzlich war ich sehr zufrieden mit dem Feedback-Prozess. Diverse Nachfragen […] konnten beantwortet werden.“* Das Feedback betonte insbesondere die Verständlichkeit des gezeigten Ablaufs und die Nachvollziehbarkeit der Umsetzungsschritte.

Dennoch ergab sich aus den Kommentaren eine konkrete Optimierungsmöglichkeit:

- Verbesserung der Übersichtlichkeit und Auffindbarkeit:
    - Visuelle oder interaktive Hilfen (z. B. Markierungen, Tooltips, UI-Elemente) zur schnelleren Orientierung innerhalb des Modells
    - Klarere Strukturierung und Benennung der einzelnen Anpassungsmöglichkeiten im Interface oder bei der Präsentation

Die Rückmeldungen der Experten bezogen sich ausschließlich auf die Verwendung von ProBuilder als Tool zur Erstellung des Modells. Diese Anmerkungen werden im folgenden Kapitel aufgegriffen.

Bewertung des Entwicklungsprozesses mit ProBuilder in Unity

Die Einschätzung zum Entwicklungsprozess mit ProBuilder in Unity wurde ausschließlich von den Experten vorgenommen, da diese über das notwendige Fachwissen verfügten, um die gezeigte Arbeitsweise mit der herkömmlichen Modellierung in DCC-Tools, wie beispielsweise Blender, zu vergleichen. Die Bewertungen erfolgten auf Basis von drei Aussagen, zu denen jeweils eine von fünf Antwortoptionen gewählt werden konnte: *trifft zu*, *trifft eher zu*, *neutral / kann ich nicht sagen*, *trifft eher nicht zu* und *trifft nicht zu*.

Aussage 4: „Mit ProBuilder bzw. Unity lässt sich (mit ausreichender Bedienungssicherheit/Routine) ein komplexes, modulares und modifizierbares 3D-Modell erstellen."

Diese Aussage wurde insgesamt verhalten positiv eingeschätzt. Zwei Experten stimmten ihr vollumfänglich zu (*trifft zu*), eine Person wählte *trifft eher zu*, während zwei Teilnehmende eine neutrale Position einnahmen (*neutral / kann ich nicht sagen*).

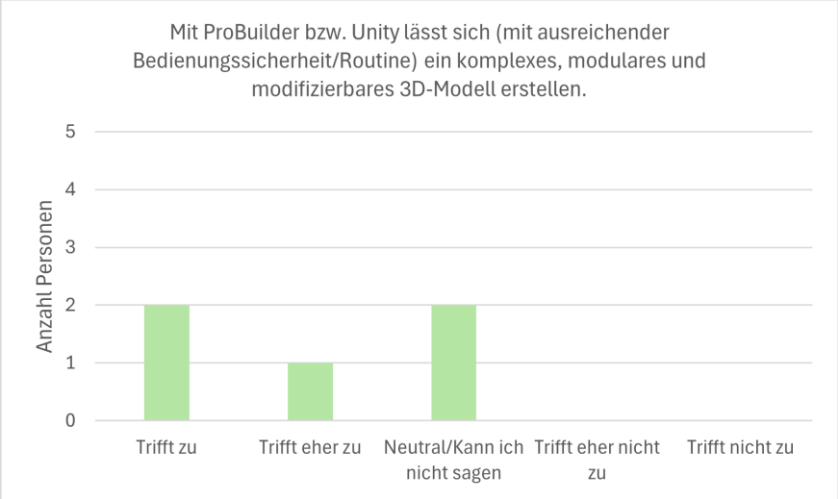


Abb. 5.21: Statistik zu Aussage 4 (Eigene Darstellung)

Aussage 5: „Die Erstellung eines 3D-Modells mit ProBuilder bzw. Unity wirkt wie eine gute Alternative zu der herkömmlichen Methode (z. B. mit Blender*). "*

Die Einschätzungen zu dieser Aussage fielen insgesamt kritisch aus. Während eine Person mit *trifft eher zu* antwortete, bewerteten drei Experten die Aussage mit *trifft eher nicht zu* und eine Person mit *trifft*

nicht zu. Damit wurde ProBuilder überwiegend nicht als gleichwertige Alternative zu klassischen Modellierungswerkzeugen wahrgenommen.

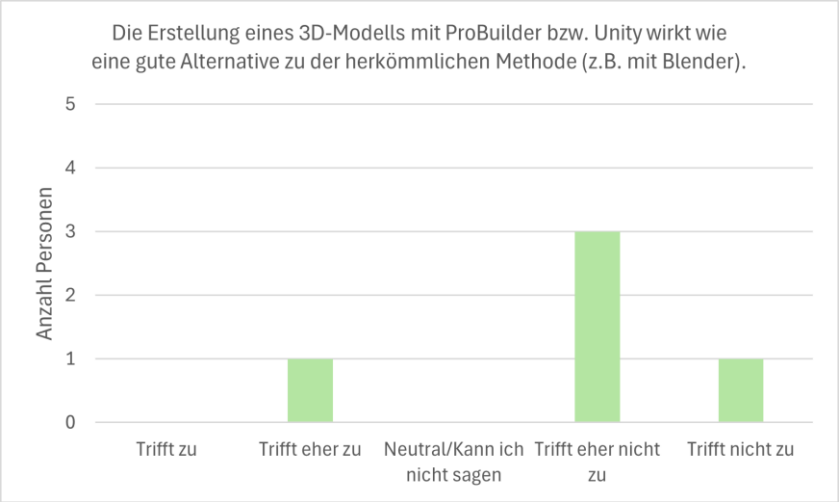


Abb. 5.22: Statistik zu Aussage 5. (Eigene Darstellung)

Aussage 6: „Der Feedbackprozess mit Kunden kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden (bei der Präsentation einer VR-Umgebung in Unity, ohne den Wechsel zu Blender bei Änderungen am Modell).“

Dieser Aussage wurde überwiegend zugestimmt. Zwei Experten stimmten vollständig zu (*trifft zu*), zwei weitere äußerten eingeschränkte Zustimmung (*trifft eher zu*), während eine Person die Aussage mit *trifft eher nicht zu* bewertete.

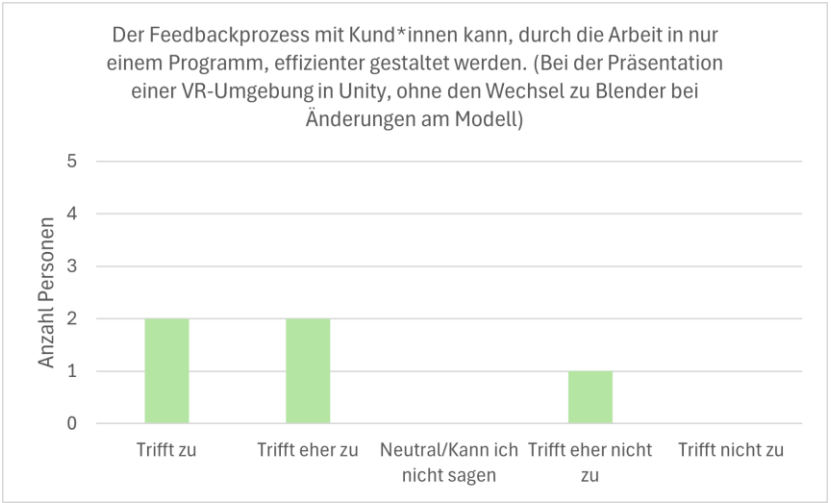


Abb. 5.23: Statistik zu Aussage 6. (Eigene Darstellung)

Abschließend wurden die Experten gebeten, eine Einschätzung darüber abzugeben, ob sie sich vorstellen können, 3D-Modelle zukünftig auf die im Projekt vorgestellte Weise zu erstellen. Die Antworten zeigten ein differenziertes Bild und reichten von grundsätzlicher Offenheit bis hin zu klaren Einschränkungen im Hinblick auf Anwendungsbereich und technische Rahmenbedingungen.

Zwei Experten äußerten grundsätzliches Interesse an der vorgestellten Arbeitsweise, etwas modular und modifizierbar aufzubauen, betonten jedoch, dass sie persönlich weiterhin mit bevorzugten Tools wie Blender arbeiten würden, da dort größere Erfahrung und ein erweiterter Funktionsumfang vorhanden seien. Ein weiterer Experte konnte sich die Nutzung des vorgestellten Konzepts in spezifischen Anwendungsfällen vorstellen – etwa dann, wenn die Performance eine untergeordnete Rolle spielt und die Konfigurierbarkeit direkt in Unity eine zentrale Anforderung des Workflows darstellt.

Zwei der Experten beurteilten die Nutzbarkeit der Methode als deutlich eingeschränkt. Sie sahen das Potenzial primär in frühen Projektphasen, etwa beim Block-Out von Objekten oder Szenen, nicht jedoch bei der Erstellung finaler Assets. Hier seien spezialisierte 3D-Programme unerlässlich, um qualitativ hochwertige Ergebnisse zu erzielen. In diesem Zusammenhang wurde auch darauf hingewiesen, dass in professionellen Projekten oft verschiedene Softwarelösungen kombiniert werden müssen, um die jeweils besten Werkzeuge für Modellierung, Texturierung und Animation nutzen zu können.

Insgesamt lässt sich festhalten, dass die vorgestellte Methode insbesondere für prototypische Anwendungen, schnelle Feedbackzyklen oder kleinere, individualisierbare Modellierungsaufgaben Potenzial bietet. Für den produktiven Einsatz in komplexeren oder qualitätskritischen Projekten wird dagegen weiterhin auf klassische DCC-Tools zurückgegriffen.

Bewertung des gesamten Projekts

Zum Abschluss der Befragung wurden die Teilnehmenden gebeten, eine zusammenfassende Einschätzung des Projekts abzugeben.

Die fachfremden Teilnehmenden äußerten sich insgesamt positiv zum Projekt und zeigten sich beeindruckt von der Möglichkeit, individuelle Anpassungen direkt innerhalb der Unity-Umgebung vornehmen zu können. Eine Person kommentierte: *„Ich frage mich, wie gut das funktioniert, wenn man ein noch komplexeres Modell hat, weil das jetzt schon recht unübersichtlich war (für mein ungeschultes Auge) […]“*. Damit wurde deutlich, dass die Bedienbarkeit des Systems, insbesondere wegen der mangelnden Übersicht, bei wachsender Modellkomplexität als potenzielles Problem wahrgenommen wurde – insbesondere aus Sicht von Nutzern ohne Vorerfahrung im 3D-Bereich.

Ein weiterer Aspekt, der in den Rückmeldungen mehrfach thematisiert wurde, war die grundsätzliche Kosten-Nutzen-Abwägung bei der Verwendung eines einzelnen Programms für Modellierung und Präsentation. Eine Person merkte an, dass es aus Laiensicht schwer zu beurteilen sei, ob der Gewinn durch den einheitlichen Softwareeinsatz mögliche Einschränkungen bei Leistung oder Flexibilität aufwiege.

Die Experten betonten in ihrer Rückmeldung die grundsätzliche Schwierigkeit, bestehende, etablierte Modellierungs- und Entwicklungsprozesse durch neue Workflows zu ersetzen. Eine Person formulierte dies wie folgt: *„Sehr schweres Thema, sehr schwer die etablierten Pipelines zu durchbrechen. Es muss einen wirklich guten Grund dafür geben.“*

Gleichzeitig wurden jedoch auch Potenziale erkannt, insbesondere in Bezug auf die Integration prozeduraler Prozesse. Als mögliches Anwendungsszenario wurde z.B. die Entwicklung eines Pflanzengenerators oder sogar eines Level-Editors angeführt, bei dem funktionale Prefabs mit In-Game-Logik erstellt und angepasst werden können. Dies wäre beim klassischen Import externer Modelle nur mit erheblichem Zusatzaufwand erreichbar.

Ein weiterer Aspekt war die Bewertung der Modellauswahl. Es wurde betont, dass Pflanzen aufgrund ihrer organischen Form und vielfältigen, komplexen Strukturen nur eingeschränktes Potenzial aufweisen, um den Workflow und dessen Vorteile vollständig abzubilden.

5.3 Dritte Phase: Weiterentwicklung des Modells

Im Anschluss an die Auswertung der Befragung wurde das erhaltene Feedback systematisch analysiert und dahingehend geprüft, welche der geäußerten Verbesserungsvorschläge im verbleibenden Projektzeitraum noch umgesetzt werden konnten. Da der Fokus dieser Arbeit auf der Entwicklung eines modularen und modifizierbaren Modells sowie auf einem effizienten Feedbackprozess lag, wurde entschieden, sich bei der Weiterentwicklung gezielt auf Optimierungsvorschläge zu konzentrieren, die unmittelbar diese beiden Aspekte betreffen.

In diesem Zusammenhang wurde zunächst zwischen zwei Arten von Rückmeldungen unterschieden: Einerseits solche, die sich auf grundlegende Eigenschaften wie Modularität und Modifizierbarkeit bezogen, andererseits jene, die primär auf eine inhaltlich oder visuell verbesserte Darstellung der gewählten Modellform – in diesem Fall einer Pflanze – abzielten. Ein Großteil der Kommentare bezog sich auf spezifische Eigenschaften pflanzlicher Strukturen, wie beispielsweise alternative Wachstumsformen oder die Integration prozeduraler Logik (z. B. Veränderung der Blattfarbe bei trockener Erde). Diese Vorschläge hätten zum Teil umfangreiche Anpassungen an Form, Textur oder Logik des Modells erfordert und konnten daher aus zeitlichen Gründen nicht umgesetzt werden.

Anders verhielt es sich bei jenen Rückmeldungen, die sich unabhängig vom konkreten Modell auf allgemeine Prinzipien der Modularität und Variantenvielfalt bezogen. Dazu zählte insbesondere der Wunsch, die Anzahl einzelner modularer Elemente – in diesem Fall Pflanzenteile aus Stängel und Blatt – flexibel verändern und anordnen zu können. Solche Anpassungen ließen sich in den verfügbaren Zeitrahmen integrieren und sind prinzipiell auch auf andere Modelle übertragbar.

Vor diesem Hintergrund wurde die Weiterentwicklung des Modells auf die Implementierung einer Funktionalität fokussiert, mit der sich die Anzahl der Pflanzenteile variabel steuern lässt. Ziel war es, auf diese Weise eine größere Variabilität im Erscheinungsbild zu erzeugen und die Möglichkeit zu schaffen, mit minimalem Aufwand neue Modellvarianten zu generieren.

5.3.1 Anpassungen

Ziel der Weiterentwicklung war es, eine möglichst effiziente Methode zu implementieren, um das bestehende Pflanzenmodell um zusätzliche Pflanzenteile zu erweitern oder bestehende zu entfernen. Dadurch sollte eine flexible Anpassung der Gesamtstruktur ermöglicht und die visuelle Variabilität des Modells deutlich erhöht werden. Anstatt neue Komponenten zu modellieren, wurde entschieden, mit den bereits vorhandenen Pflanzenteilen zu arbeiten und ein ergänzendes Skript (siehe Anhang A.4: PlantPotManagerWithSlider.cs) zu entwickeln, das auf Grundlage des Ursprungszustands der Pflanze dynamisch Pflanzenteile hinzufügt oder entfernt.

Als geeignetes Steuerungselement wurde ein Slider gewählt, mit dessen Hilfe die Anzahl der dargestellten Pflanzenteile in Echtzeit (`ExecuteAlways`) variiert werden konnte. Der Regler war so konfiguriert, dass der Ausgangswert 0 dem Ausgangszustand der Pflanze mit 42 Originalelementen entsprach.

Im Bereich von 0 bis –42 erfolgte eine sukzessive Reduktion der Pflanzenteile: Für jede Einheit, die der Slider in den negativen Bereich verschoben wurde, wurde ein Pflanzenteil entfernt – bis hin zur vollständigen Entfernung aller 42 Elemente. Um die visuelle Vielfalt zu erhöhen, erfolgte das Entfernen nicht in fester Reihenfolge, sondern zufällig.

```
if (elementSlider < 0)
{
    int desiredVisible = originalCount + elementSlider;
    int currentVisible = 0;
    for (int i = 0; i < originalParts.Count; i++)
    {
        if (originalParts[i] != null && originalParts[i].activeSelf)
                    currentVisible++;
    }

    if (currentVisible > desiredVisible)
    {
         List<int> candidates = new List<int>();
         for (int i = 0; i < originalParts.Count; i++)
         {
             if (originalParts[i] != null && originalParts[i].activeSelf
&& !hiddenIndices.Contains(i))
                 candidates.Add(i);
         }
         if (candidates.Count > 0)
         {
             int randomIndex = candidates[Random.Range(0,
             candidates.Count)];
             originalParts[randomIndex].SetActive(false);
             hiddenIndices.Add(randomIndex);
         }
    }

…
```

Abb. 5.24: Zufälliges Ausblenden eines Pflanzenteils beim Verschieben des Sliders in den negativen Bereich. (Auszug aus PlantPotManagerWithSlider.cs)

Im Bereich von 0 bis +42 konnten bis zu 42 zusätzliche Pflanzenteile ergänzt werden. Hierzu wurde bei jeder Änderung ein bestehendes Originalelement kopiert, zufällig in seiner Skalierung (bis zu ±10 %) angepasst und an einer neuen Position innerhalb der Pflanze eingefügt. Wurde der Slider wieder in Richtung des Ausgangswerts verschoben, wurden die jeweils zuletzt hinzugefügten Elemente wieder entfernt.

```
GameObject dup = Instantiate(original, transform);
dup.name = original.name + "_copy";
dup.transform.position = newPos;
Vector3 origEuler = original.transform.rotation.eulerAngles;
dup.transform.rotation = Quaternion.Euler(origEuler.x, origEuler.y +
180f, origEuler.z);
float scaleFactor = Random.Range(0.9f, 1.1f);
dup.transform.localScale = original.transform.localScale * scaleFactor;
```

Abb. 5.25: Erzeugen eines gespiegelten und zufällig skalierten Duplikats eines Pflanzenteils. (Auszug aus PlantPotManagerWithSlider.cs)

Diese Methode ermöglichte es, ausgehend vom ursprünglichen Modell zahlreiche Varianten mit unterschiedlichen Proportionen und Strukturen zu generieren. Neben der Gesamtanzahl der Pflanzenteile konnten auch deren Positionierung, Skalierung sowie das Verhältnis zwischen Pflanze, Blumentopf und Erde flexibel variiert werden. Im Anschluss an die Anpassung konnte zur Optimierung der Performance der Mesh Baker auf das finale Modell angewendet werden, um die Anzahl der Batches zu reduzieren und die Effizienz der Darstellung in der Unity-Anwendung zu verbessern.

5.3.2 Herausforderungen & Lösungen

Die Umsetzung der beschriebenen Anpassungsfunktion basierte in weiten Teilen auf einem iterativen Entwicklungsprozess mit zahlreichen Test- und Korrekturschleifen. Ziel war es, eine flexible Steuerung der Anzahl an Pflanzenteilen über einen Slider zu realisieren. Im Verlauf der Entwicklung traten dabei unterschiedliche Herausforderungen auf.

Der erste Ansatz bestand darin, einen Slider mit dem Wertebereich von –1 bis 1 zu verwenden. Da sich mit dieser Konfiguration jedoch häufig nicht zuverlässig eine bestimmte Anzahl an Elementen hinzufügen oder entfernen ließ, wurde der Bereich erweitert. Die Originalpflanze bestand aus 42 Pflanzenteilen, so dass ein Wertebereich von –42 bis +42 eine präzisere Steuerung ermöglichte, da so jede Einheit auf dem Slider nun exakt einer Veränderung (Hinzufügen oder Entfernen) eines Pflanzenteils entsprach.

Die Reduktion der Pflanzenteile im negativen Bereich funktionierte grundsätzlich stabil. Die Strategie, die Originalobjekte bei Bedarf zu deaktivieren und bei einer Umkehrung des Sliderwerts wieder zu aktivieren, erwies sich als zuverlässig. Die Herausforderung lag darin, dass die Pflanzenteile immer in der gleichen Reihenfolge ein- und ausgeblendet wurden und so die gewünschte Variabilität fehlte. Um dem entgegenzuwirken, wurde eine zufällige Auswahlmechanik implementiert. Dafür wurde eine Liste `hiddenIndices` eingeführt, in der die Indizes der ausgeblendeten Objekte gespeichert wurden. Bei

jeder Reduktion wurde ein sichtbares Objekt zufällig ausgewählt, deaktiviert und sein Index in die Liste eingetragen. Bei einer Erhöhung des Sliderwerts in Richtung 0 wurden die ausgeblendeten Objekte in der Reihenfolge ihrer Deaktivierung wieder sichtbar geschaltet.

Die Erweiterung des Modells durch das Hinzufügen zusätzlicher Pflanzenteile stellte sich als deutlich komplexer heraus. Zu Beginn wurden neue Duplikate immer an derselben Position wie das Ausgangselement erzeugt, was zu keiner sichtbaren Veränderung im Modell führte. Zur Lösung dieses Problems wurde der Mittelpunkt des übergeordneten GameObjects „Pflanzenteile“ als Bezugspunkt verwendet. Neue Elemente wurden um diesen Mittelpunkt rotiert und auf der gegenüberliegenden Seite des Originals eingefügt. Zudem wurde eine zufällige Skalierung von ±10 % angewendet, um die visuelle Ähnlichkeit zum Original zu reduzieren.

Ein weiteres Problem bestand in der Entfernung der hinzugefügten Duplikate. Diese ließen sich zunächst nicht korrekt löschen. Nachdem die gewünschten Abläufe grundsätzlich funktionierten, kam es jedoch zu einem neuen Problem: Beim Zurücksetzen des Sliderwerts wurden teils zu viele Objekte entfernt, sodass der ursprüngliche Zustand bei Sliderposition 0 nicht wiederhergestellt werden konnte.

Zur Problemlösung wurden zwei Ansätze getestet:

Ansatz 1:

Die Duplikate wurden mit der Endung _copy versehen. Bei Sliderwerten ≥ 0 sollten alle Originalelemente sichtbar bleiben, während die Duplikate entsprechend der gewählten Sliderzahl hinzugefügt oder entfernt wurden. Bei der Entfernung sollten ausschließlich Objekte mit der Endung _copy berücksichtigt werden. Im Verlauf der Tests zeigte sich jedoch, dass diese Trennung nicht zuverlässig funktionierte – teilweise waren am Ende ausschließlich Kopien im Modell enthalten.

Ansatz 2:

Die Einführung zweier getrennter Container für Originalelemente und Duplikate sollte die Trennung strukturieren. Bei der Zerstörung eines Elements sollte immer nur auf die Duplikate im entsprechenden Container zurückgegriffen werden. Die Aufteilung erwies sich jedoch als fehleranfällig, so dass weiterhin die falschen Elemente zerstört wurden. Zudem kam es beim Verschieben des Sliders im positiven Wertebereich zum Einfrieren der Anwendung, so dass mehrfach ein Programmabbruch erzwungen werden musste.

Schließlich wurde wieder auf Ansatz 1 zurückgegriffen, wobei jedoch keine finale, fehlerfreie Lösung gefunden werden konnte. In der aktuellen Version des Modells führt das Zurücksetzen des Sliders auf 0 dazu, dass nicht exakt der Ausgangszustand wiederhergestellt wird – einige Elemente bleiben dauerhaft entfernt oder doppelt vorhanden. Dennoch konnte mithilfe des Sliders eine Vielzahl unterschiedlicher und visuell zufriedenstellender Varianten des Modells generiert werden. Aufgrund der begrenzten verbleibenden Projektzeit wurde von einer weiteren Optimierung abgesehen.

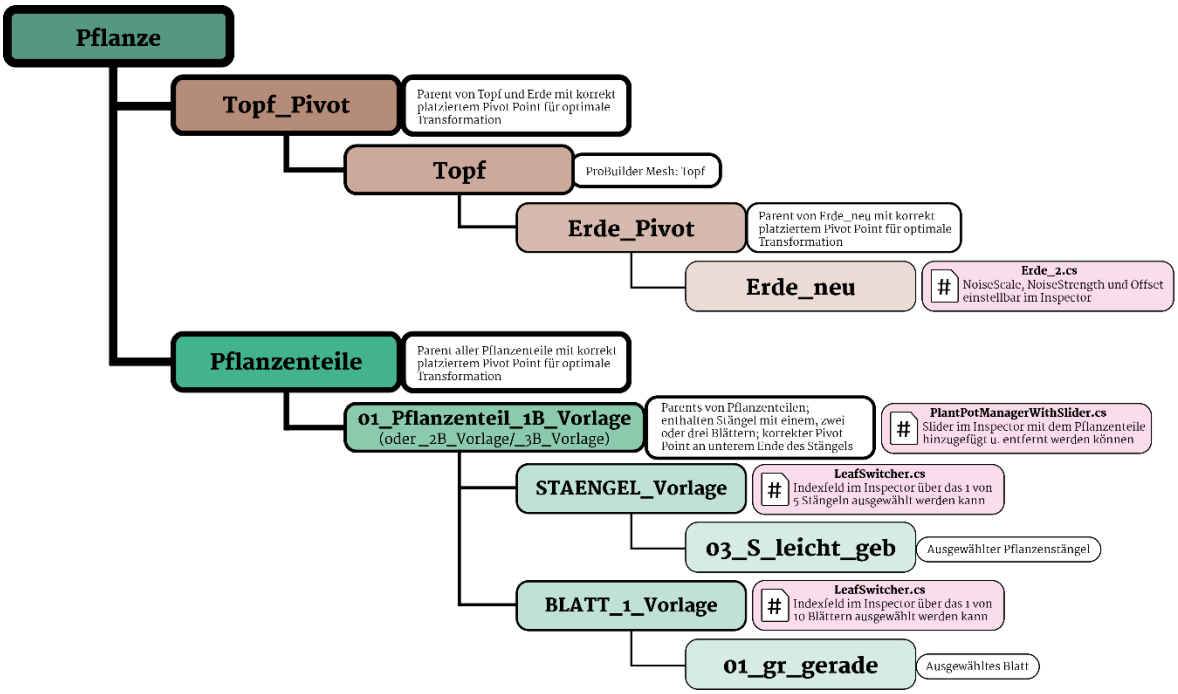


Abb. 5.27: Finaler Aufbau der Pflanze inkl. C#-Skripten. (Eigene Darstellung)

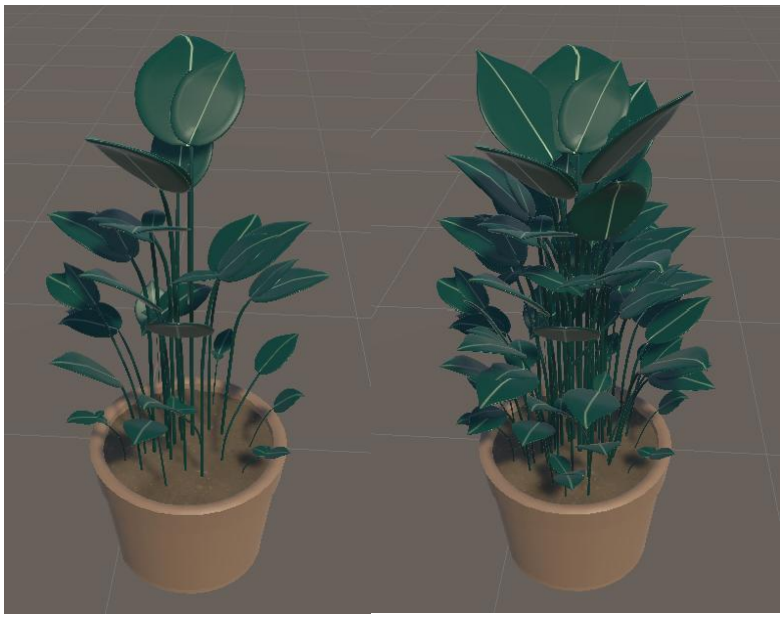


Abb. 5.26: Pflanze mit -19 Pflanzenteilen (links) und Pflanze mit +15 Pflanzenteilen (rechts). (Screenshot)

6 Auswertung und Diskussion

Die vorangegangenen Kapitel beschrieben den Entwicklungsprozess eines komplexen, modularen und modifizierbaren 3D-Modells sowie die Durchführung einer praxisorientierten Studie, die zur Bewertung dieses Modells sowie des zugehörigen Feedbackprozesses diente. In diesem Kapitel werden die dabei gewonnenen Ergebnisse systematisch ausgewertet und in den Kontext der eingangs formulierten Forschungsfragen und Hypothesen eingeordnet.

Darüber hinaus erfolgt eine kritische Reflexion der eingesetzten Methodik. Es wird analysiert, inwieweit sich die ursprünglich formulierten Annahmen bestätigt haben, welche Aspekte unerwartet oder kritisch zu bewerten sind, und welche methodischen oder inhaltlichen Einschränkungen sich im Verlauf des Projekts gezeigt haben. Abschließend werden Empfehlungen für weiterführende Forschung formuliert, um auf Grundlage der gewonnenen Erkenntnisse potenzielle Weiterentwicklungen oder alternative methodische Herangehensweisen anzuregen.

6.1 Entwicklung eines komplexen, modularen und modifizierbaren 3D-Modells

6.1.1 Zusammenfassung der quantitativen und qualitativen Ergebnisse

Im Rahmen der Studie wurde das entwickelte 3D-Modell von den Teilnehmenden hinsichtlich seiner Modularität und Modifizierbarkeit bewertet. Die Ergebnisse zeigen ein überwiegend positives Bild:

Modularität

Neun von zehn Teilnehmenden bewerteten das Modell als „modular“. Lediglich ein Experte stimmte teilweise zu. Damit wurde der modulare Aufbau mehrheitlich – auch unter den fachkundigen Teilnehmenden – anerkannt.

Modifizierbarkeit

Alle fünf fachfremden Teilnehmenden stimmten der Aussage, dass das Modell modifizierbar sei, uneingeschränkt zu. Zwei der fünf Experten stimmten ebenfalls voll zu, drei bewerteten die Aussage mit „trifft teilweise zu“.

Weiterentwicklung des Modells

In einem weiteren Schritt wurde das Modell funktional erweitert, unter anderem durch die Integration eines Sliders zur Steuerung der Anzahl der Pflanzenteile. Damit konnten zusätzliche Varianten erzeugt werden, was die Flexibilität des Modells erhöhte – trotz vereinzelter technischer Einschränkungen.

Qualitative Rückmeldungen

Da qualitatives Feedback optional war, gaben die fachfremden Teilnehmenden nur vereinzelt Rückmeldungen und äußerten dabei sich insgesamt zufrieden mit dem Modell. Die Experten hingegen machten

zahlreiche Vorschläge zur Verbesserung, wie unter anderem:

- Ergänzung prozeduraler Logik (z. B. automatische Reaktionen des Modells auf Veränderungen)
- Erweiterung der Modellvielfalt (z. B. unterschiedliche Wachstumsformen).

6.1.2 Interpretation der Ergebnisse

Die zusammengefassten Ergebnisse lassen mehrere Schlussfolgerungen zur Wahrnehmung und Qualität des entwickelten Modells zu.

Die vollständige Zustimmung der fachfremden Teilnehmenden zur Modifizierbarkeit deutet darauf hin, dass aus der Perspektive potenzieller Kunden keine relevanten Funktionen vermisst wurden. Die vorgestellten Anpassungsmöglichkeiten erschienen nachvollziehbar und es entstand der Eindruck, dass gewünschte Änderungen bei einer Beauftragung realistisch und zuverlässig umgesetzt werden könnten. Die Rückmeldungen legen nahe, dass das Modell in seiner aktuellen Form den Erwartungen entspricht, die potenzielle Kunden an ein modulares und flexibel anpassbares Ergebnis stellen würden.

Im Gegensatz dazu zeigen die differenzierteren Rückmeldungen der Experten, dass sie über weitere technische Aspekte und mögliche Erweiterungspotenziale nachdachten. Ihre zurückhaltendere Bewertung verweist auf wahrgenommene Limitationen – etwa bei der praktischen Umsetzung komplexerer Workflows.

Auch im Hinblick auf die Modularität ist das Urteil überwiegend positiv. Die breite Zustimmung, insbesondere auch von fachkundiger Seite, spricht dafür, dass die Struktur des Modells als klar und nachvollziehbar wahrgenommen wurde. Die vereinzelte Einschränkung eines Experten lässt sich als Hinweis auf weiterführende Erwartungen an Modularität interpretieren.

Die qualitativen Rückmeldungen ergänzen dieses Bild und verdeutlichen die unterschiedlichen Perspektiven der beiden Gruppen: Während die fachfremden Teilnehmenden, also potenzielle Kunden, das Modell in seiner derzeitigen Umsetzung als überzeugend und funktional beschrieben, waren die Rückmeldungen der Experten kritischer. Sie äußerten zahlreiche Optimierungsmöglichkeiten, von denen viele nicht als Mangel, sondern vielmehr als Weiterentwicklung des Modells interpretiert werden konnten. Dies legt nahe, dass die Experten berufsbedingt dazu neigen, automatisch in Weiterentwicklungs- und Optimierungsmöglichkeiten zu denken, weswegen einige ihrer Anmerkungen eher als konstruktive Vorschläge und nicht als zwingend erforderliche Änderungsbedarfe zu verstehen waren.

Potenzielle Kunden hingegen beurteilen das Modell primär danach, ob es ihren Erwartungen entspricht und ob Änderungswünsche grundsätzlich umsetzbar erscheinen. Wenn das Modell in seiner Funktionalität und im äußeren Erscheinungsbild ihren Vorstellungen entspricht, sehen sie wenig Anlass, weiterführende Verbesserungen einzufordern.

Ein weiterer wichtiger Aspekt betraf den Modelltyp: Die Kritik und die Verbesserungsvorschläge der Experten bezogen sich inhaltlich häufig sehr konkret auf die Pflanze als gewähltes Modellobjekt. Diese

Rückmeldungen sind somit teilweise weniger als grundsätzliche Kritik am modularen Ansatz oder an der Methodik zu verstehen, sondern als spezifische Anforderungen an ein realistisches Pflanzenmodell.

6.2 Entwicklungsprozess mit ProBuilder in Unity

6.2.1 Zusammenfassung der quantitativen und qualitativen Ergebnisse

Im Rahmen der Befragung wurde der Entwicklungsprozess mit Unity und dem integrierten Tool ProBuilder sowohl durch die durchführende Person und als auch durch die Experten quantitativ und qualitativ bewertet und eingeschätzt.

Umsetzbarkeit eines komplexen, modularen und modifizierbaren Modells

Der Aussage *„Mit ProBuilder bzw. Unity lässt sich ein komplexes, modulares und modifizierbares 3D-Modell erstellen“* stimmten 40 % der Experten vollständig zu, 20 % wählten *„trifft eher zu“*, während 40 % sich neutral äußerten. Negative Bewertungen wurden nicht abgegeben. Die Ergebnisse zeigen, dass die grundsätzliche Umsetzbarkeit anerkannt wird, jedoch mit Einschränkungen.

Bewertung als Alternative zu DCC-Tools

Deutlich kritischer fiel die Bewertung der Aussage aus, dass ProBuilder eine gute Alternative zu herkömmlichen Modellierungsprogrammen wie Blender darstellt: 80 % der Experten wählten hier *„trifft eher nicht zu“* oder *„trifft nicht zu“*. Nur 20 % entschieden sich für *„trifft eher zu“*. Damit wird ProBuilder mehrheitlich nicht als gleichwertiger Ersatz zu klassischen DCC-Tools eingeschätzt.

Qualitative Einschätzungen

Die Entwicklung des 3D-Modells mit ProBuilder in Unity gestaltete sich aus Sicht der durchführenden Person als anspruchsvoll. Insbesondere zu Beginn war eine längere Einarbeitungszeit erforderlich, da die Bedienlogik des Tools im Vergleich zu klassischen DCC-Programmen wie Blender weniger intuitiv erschien. Rückblickend zeigte sich jedoch, dass sich der Umgang mit ProBuilder durch praktische Erfahrung zunehmend vereinfachte und effizienter gestaltbar wurde. Die Erstellung zusätzlicher Funktionalitäten erforderte die Arbeit mit C#-Skripten, was entweder bestehende Programmierkenntnisse oder intensivere Recherche voraussetzte. Ein weiterer kritischer Punkt betraf die Performance: Komplexe Modelle führten in Unity zu einer hohen Anzahl an Batches und damit zu einer spürbaren Belastung des Systems. Zwar konnte dies mithilfe des Tools Mesh Baker deutlich verbessert werden, allerdings ging diese Lösung mit Einschränkungen hinsichtlich der nachträglichen Bearbeitbarkeit einher, so dass die Arbeit mit Backup-Modellen notwendig war.

Die Experten erwähnten sowohl Potenziale als auch Einschränkungen des Entwicklungsprozesses mit ProBuilder. Während drei Teilnehmende die Methode zwar als geeignet für spezifische Szenarien einstuften, bevorzugten sie in der eigenen Praxis jedoch weiterhin etablierte Tools wie Blender, u. a. wegen

des erweiterten Funktionsumfangs und vertrauter Workflows. Zwei Experten äußerten sich deutlich kritischer: Sie sahen ProBuilder lediglich als Werkzeug für frühe Projektphasen wie beispielsweise Block-Outs, nicht jedoch für die Erstellung finaler Assets.

Insgesamt wurden ProBuilder somit primär als Werkzeuge für prototypisches Arbeiten, Projekte mit schnellen Änderungszyklen und Präsentationen innerhalb Unitys eingeordnet. Für die Produktion qualitativ hochwertiger Assets oder komplexer Modelle mit hohen Anforderungen bleibt laut Experteneinschätzung der Einsatz klassischer DCC-Tools wie Blender weiterhin unerlässlich.

6.2.2 Interpretation der Ergebnisse

Die Ergebnisse der Expertenbefragung zeigen, dass Unity in Kombination mit ProBuilder grundsätzlich zur Erstellung komplexer, modularer und modifizierbarer 3D-Modelle geeignet sein kann – allerdings nur unter bestimmten Voraussetzungen. Die mehrheitlich neutral bis positive Einschätzung zur generellen Umsetzbarkeit lässt erkennen, dass das Tool prinzipiell als funktionstüchtig wahrgenommen wurde, der erfolgreiche Einsatz jedoch stark von individuellen Vorkenntnissen, technischer Routine und klarer Planung abhängt.

Deutlich kritischer wurde hingegen die Frage bewertet, ob ProBuilder eine echte Alternative zu herkömmlichen DCC-Programmen wie Blender darstellen kann. Die fast ausschließlich ablehnenden Rückmeldungen sprechen dafür, dass ProBuilder nicht als vollwertiger Ersatz angesehen wird. Die Gründe hierfür liegen vor allem im reduzierten Funktionsumfang und in begrenzten Workflow-Möglichkeiten.

Die qualitativen Aussagen vertiefen dieses Bild: Aus Perspektive der durchführenden Person stellte der Umgang mit ProBuilder anfangs eine deutliche Herausforderung dar, insbesondere aufgrund der weniger intuitiven Bedienung im Vergleich zu DCC-Tools. Gleichzeitig wurde jedoch erkannt, dass sich mit wachsender Erfahrung effizientere Arbeitsabläufe entwickeln lassen.

Die Experten betonten in ihren Rückmeldungen, dass ProBuilder durchaus Potenzial für spezifische Einsatzbereiche bietet, insbesondere z.B. für frühe Projektphasen eines VR-Projekts. Gleichzeitig wurde jedoch deutlich, dass spezialisierte 3D-Tools wie Blender weiterhin als Standard angesehen werden. Diese Einschätzung wird durch die Performanceprobleme bei komplexen Modellen und die damit verbundene Notwendigkeit nachträglicher Optimierungen (z. B. durch Mesh Baker) weiter bestärkt.

Unity selbst bewirbt ProBuilder als integriertes 3D-Modellierungswerkzeug, das speziell für das schnelle Erstellen und Bearbeiten von Geometrien innerhalb des Unity Editors entwickelt wurde. Ziel ist es, vor allem in frühen Produktionsphasen wie Blockouts oder Prototypings zügige und flexible Modellierungsprozesse zu ermöglichen oder simple Level-Designs zu erstellen. Beworben werden insbesondere die einfache Erstellung parametrischer Grundformen, die direkte Mesh-Bearbeitung in der Szene, eine unkomplizierte UV-Zuweisung sowie eine API zur Automatisierung.

Im Abgleich mit den praktischen Erfahrungen zeigt sich, dass die angestrebten Einsatzbereiche grundsätzlich erfüllt werden. Die Erstellung einfacher Formen gelingt problemlos, darüber hinaus ist jedoch auch die Umsetzung komplexerer Modelle möglich, wenn auch mit steigendem Aufwand. Einschränkungen zeigten sich vor allem in der Bedienbarkeit einzelner Werkzeuge, die im Vergleich zu etablierten Programmen wie Blender oft als weniger intuitiv empfunden wurde. Die Grundidee von ProBuilder, ein einfaches, direkt im Editor integriertes Modellierungswerkzeug für spezifische Szenarien zu bieten, konnte durch den Praxistest insgesamt bestätigt werden, wobei aber auch deutliche Grenzen im Funktionsumfang und in der Nutzerfreundlichkeit sichtbar wurden.

Insgesamt legen die Ergebnisse nahe, dass ProBuilder in der Regel nicht als eigenständige Modellierungslösung verstanden werden sollte, sondern primär als ergänzendes Werkzeug innerhalb spezifischer Anwendungsrahmen dient. In bestimmten Einsatzszenarien kann jedoch auch der ausschließliche Einsatz von ProBuilder zweckmäßig sein, beispielsweise bei prototypischen Anwendungen die weniger komplexe Modelle beinhalten.

6.3 Feedbackprozess

6.3.1 Zusammenfassung der quantitativen und qualitativen Ergebnisse

Ein zentrales Ziel der Untersuchung bestand darin, die Effizienz des Feedbackprozesses bei der Präsentation eines 3D-Modells in Unity zu erproben und zu bewerten. Die Rückmeldungen zeigen insgesamt eine positive Tendenz, wobei deutliche Unterschiede zwischen fachfremden Teilnehmenden und Experten erkennbar wurden.

Meinung zur Umsetzungsgeschwindigkeit von Änderungswünschen

Bezüglich der Aussage *„Anpassungen bzw. Änderungswünsche können schnell umgesetzt werden“* stimmten drei von fünf fachfremden Personen uneingeschränkt zu, zwei äußerten eine teilweise Zustimmung. Die Experten bewerteten zurückhaltender: Eine Person stimmte zu, drei gaben *„trifft teilweise zu“* an, eine Person *„trifft eher nicht zu“*. Insgesamt wurde die Umsetzbarkeit von Änderungen also tendenziell positiv, aber mit Vorbehalten aufseiten der Experten eingeschätzt.

Bewertung des integrierten Workflows

Die Aussage *„Der Feedbackprozess mit Kunden kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden.“* wurde ausschließlich von den Experten beurteilt, die einen Vergleich zu der alternativen Methode ziehen konnten. Zwei Personen stimmten vollständig zu, zwei weitere eher zu. Eine Person bewertete die Aussage mit *„trifft eher nicht zu“*. Damit wurde ein integrierter Workflow mehrheitlich positiv gesehen.

Punktebewertung des Feedbackprozesses

Auch die Punktevergabe verdeutlicht die Differenz in der Einschätzung: Fachfremde Teilnehmende bewerteten den Feedbackprozess im Durchschnitt mit 9,0 Punkten, die Experten vergaben durchschnittlich 7,2 Punkte.

Qualitative Rückmeldungen

Die fachfremden Personen lobten insbesondere die Nachvollziehbarkeit und Verständlichkeit des gezeigten Feedbackprozesses. Die Experten machten hingegen nur wenige direkte Aussagen zum Prozess selbst. Stattdessen bezog sich ein Großteil ihrer Kritik – wie bereits im Kontext des Entwicklungsprozesses dargestellt – auf den Einsatz von ProBuilder als Modellierungswerkzeug. Aus dem qualitativen Feedback ergaben sich dennoch zwei konkrete Optimierungsvorschläge:

- die Einführung visueller oder interaktiver Orientierungshilfen (z. B. UI-Elemente),
- eine klarere Strukturierung und Benennung der Anpassungsmöglichkeiten im Interface oder bei der Präsentation.

6.3.2 Interpretation der Ergebnisse

Die quantitativen Ergebnisse deuten darauf hin, dass der gezeigte Feedbackprozess grundsätzlich als effizient wahrgenommen wurde – insbesondere von den fachfremden Teilnehmenden. Diese beurteilten die Änderbarkeit des Modells als durchweg positiv und gaben dem Prozess im Durchschnitt eine sehr hohe Punktzahl. Hier zeigt sich, dass insbesondere für Personen ohne Vorwissen die direkte Umsetzbarkeit ihrer Rückmeldungen als überzeugend empfunden wurden.

Die zurückhaltendere Einschätzung durch die Experten kann als Ausdruck einer differenzierteren Perspektive verstanden werden. Sie bewerteten die Anpassbarkeit grundsätzlich positiv, wiesen jedoch implizit auf potenzielle Grenzen hin – etwa in Bezug auf technische Komplexität, Performance oder Skalierbarkeit bei größeren Projekten. Dies spiegelte sich auch in der vergleichsweise moderaten Punktevergabe (7,2) wider, die zwar auf ein anerkanntes Potenzial, aber auch auf Optimierungsbedarf hindeutet.

Die Aussage zur Effizienz eines integrierten Workflows wurde mehrheitlich positiv bewertet. Vier von fünf Experten sahen Vorteile in der ausschließlichen Arbeit innerhalb Unitys. Allerdings wurde auch hier deutlich, dass die Effizienzgewinne kontextabhängig sind: Bei komplexen Projekten mit hohen Anforderungen an Modellqualität und technische Präzision könnten klassische Workflows mit spezialisierten Tools weiterhin im Vorteil sein.

In den qualitativen Rückmeldungen wurde der Feedbackprozess von fachfremden Teilnehmenden als verständlich, nachvollziehbar und gut strukturiert beschrieben. Ihre Anregungen zur Verbesserung –

etwa durch interaktive UI-Elemente oder klarere Beschriftungen – zeigen dennoch, dass die Nutzererfahrung weiter optimiert werden könnte, insbesondere mit Blick auf Orientierung und Bedienbarkeit.

Insgesamt lässt sich festhalten, dass der Feedbackprozess in seiner Grundstruktur positiv bewertet wurde, insbesondere hinsichtlich der Transparenz und Umsetzbarkeit von Rückmeldungen. Unterschiede in der Einschätzung zwischen den Gruppen unterstreichen jedoch, dass die wahrgenommene Effizienz stark von der Vorerfahrung mit 3D-Modellierung abhängt.

6.4 Bezug auf Fragestellungen und Hypothesen

In diesem Kapitel werden die zentralen Ergebnisse der Studie zusammengeführt und im Hinblick auf die zu Beginn der Arbeit formulierten Forschungsfragen und Hypothesen eingeordnet. Ziel ist es zu überprüfen, inwiefern die ursprünglichen Annahmen bestätigt oder widerlegt werden konnten. Darüber hinaus werden unerwartete Erkenntnisse sowie kritisch zu bewertende Aspekte identifiziert und reflektiert.

H1: Komplexes, modulares und modifizierbares 3D-Modell

Diese Hypothese kann auf Basis der quantitativen und qualitativen Ergebnisse bestätigt werden. Sowohl die Bewertung der Teilnehmenden als auch der Entwicklungsprozess selbst zeigten, dass mit ProBuilder grundsätzlich die Erstellung eines komplexen 3D-Modells möglich ist. Der modulare Aufbau und auch die Modifizierbarkeit wurden – insbesondere aus Sicht der fachfremden Personen, also potenziellen Kunden – durchgehend positiv eingeschätzt. Die Weiterentwicklung des Modells zeigte zudem, dass zusätzliche Funktionalitäten mit überschaubarem Aufwand realisierbar sind.

Die Forschungsfrage **F1.1** kann demnach bejaht werden.

H2: Effizienter Designprozess

Diese Hypothese kann nicht bestätigt werden. Zwar war der Designprozess aus Sicht der durchführenden Person mit zunehmender Erfahrung effizienter gestaltbar, dennoch zeigte sich, dass die Arbeit mit spezialisierten DCC-Tools für komplexe Modellierungsvorhaben weiterhin überlegen ist. Bekräftigt wird dies durch die Einschätzung der Experten, deren Mehrheit (80 %) die Aussage, dass ProBuilder eine gute Alternative zu DCC-Tools wie Blender darstellt, kritisch oder ablehnend bewertete.

Die Fragestellung **F1.2** muss demnach verneint werden.

H3: Technische und gestalterische Grenzen

Diese Hypothese kann bestätigt werden. Im praktischen Entwicklungsprozess zeigten sich klare technische Grenzen, etwa bei der Performance, der Bedienbarkeit komplexer Werkzeuge und der Notwendigkeit zusätzlicher C#-Programmierung zur Realisierung bestimmter Funktionen. Auch auf gestalterischer Ebene traten im Arbeitsprozess Herausforderungen auf. Bestimmte Modellierungsaufgaben z.B. orga-

nischer Formen oder detailreicher Oberflächen erwiesen sich aufgrund fehlender spezialisierter Werkzeuge (z. B. Sculpting-Tool) als deutlich schwerer umsetzbar, jedoch nicht grundsätzlich unmöglich. Die Limitierungen lagen weniger in der prinzipiellen Umsetzbarkeit, sondern vielmehr im erhöhten manuellen Aufwand und der eingeschränkten Werkzeugunterstützung. Die Fragestellung **F1.3** kann demnach bejahrt werden. Es handelt sich jedoch nicht um unüberwindbare Einschränkungen, sondern um Aspekte, deren Umsetzung mit mehr oder weniger hohem Aufwand verbunden ist.

H4: Effizienter Feedbackprozess

Diese Hypothese kann vorwiegend bestätigt werden. Die fachfremden Teilnehmenden bewerteten den Feedbackprozess insgesamt als sehr positiv, was sich in den hohen Punktwerten und den positiven Kommentaren widerspiegelte. Auch die Experten beurteilten den einheitlichen Workflow innerhalb von Unity überwiegend als effizient. Dennoch zeigte sich, dass diese Einschätzung nicht automatisch zu einer Akzeptanz des neuen Workflows führte. Die Fragestellung **F1.4** kann demnach wie folgt beantwortet werden: Die Nutzung einer einzigen Software wird von potenziellen Kunden als effizient wahrgenommen, insbesondere weil der Feedbackprozess für sie nachvollziehbar und schnell war. Aus professioneller Sicht wird die Effizienz des integrierten Workflows anerkannt, jedoch nicht als hinreichender Grund gesehen, um bestehende Methoden und Tools vollständig zu ersetzen.

F1: Welche technischen und gestalterischen Möglichkeiten und Grenzen bestehen bei der Entwicklung eines modifizierbaren und modularen 3D-Modells in Unity und wie wirkt sich dies auf die Effizienz des Feedbackprozesses aus?

Die technischen und gestalterischen Möglichkeiten und Grenzen wurden bereits erläutert, wobei verdeutlicht wurde, dass es keine klaren Grenzen bei der Umsetzung eines modularen und modifizierbaren 3D-Modells mit ProBuilder in Unity gibt, sondern lediglich Herausforderungen. Der Feedbackprozess wurde insbesondere von fachfremden Teilnehmenden als transparent, schnell und effizient wahrgenommen. Die ausschließliche Nutzung eines einzigen Programms wirkte sich dabei nachweislich positiv auf die Wahrnehmung des Prozesses aus.

Gleichzeitig wurde deutlich, dass dieser Effizienzgewinn nicht automatisch zu einer Veränderung etablierter Arbeitsprozesse führt. Trotz positiver Einschätzung des integrierten Feedbackprozesses halten die meisten Experten weiterhin an klassischen, bewährten Workflows fest. Dies lässt sich auf Vorteile professioneller DCC-Tools, aber auch auf Routine, Tool-Präferenzen und bestehende Produktionspipelines zurückführen. Der potenzielle Effizienzgewinn beim Feedback allein reicht demnach nicht aus, um den Wechsel zu einem vollständig Unity-basierten Entwicklungsprozess zu rechtfertigen.

6.5 Beschränkungen der Methodik

Trotz sorgfältiger Planung und Durchführung weist die vorliegende Untersuchung methodische Begrenzungen und Makel auf. Im Folgenden werden zentrale Aspekte benannt, die den Erkenntnisgewinn potenziell eingeschränkt haben oder an denen sich künftige Untersuchungen orientieren könnten.

Stichprobengröße und Repräsentativität

Die Befragung wurde mit insgesamt zehn Personen durchgeführt – fünf fachfremde und fünf fachkundige Teilnehmende. Zwar ermöglichte diese Zusammensetzung einen gezielten Vergleich unterschiedlicher Perspektiven und lieferte wertvollen Input für Optimierungsmöglichkeiten, doch stellte sie keine repräsentative Stichprobe im statistischen Sinne dar. Die Ergebnisse bieten daher eher ein qualitatives Meinungsbild als verallgemeinerbare Aussagen. Dies entsprach zwar dem Ziel der Studie, sollte aber im Hinblick auf die Aussagekraft der Ergebnisse berücksichtigt werden. Trotz der begrenzten Stichprobengröße zeigte sich jedoch eine klare Tendenz in den Rückmeldungen der Teilnehmenden. Aufgrund der Eindeutigkeit dieser Einschätzungen ist davon auszugehen, dass sich diese grundlegenden Tendenzen auch bei einer größeren Teilnehmerzahl nicht grundlegend verändern würden.

Fehlender direkter Vergleich mit herkömmlichen Arbeitsprozessen

Ein vollständiger Vergleich zwischen dem gezeigten Arbeitsprozess innerhalb von Unity und einem klassischen Vorgehen mit Modellierung in Blender oder einem anderen DCC-Tool und anschließender Präsentation in Unity war im Rahmen der Untersuchung nicht möglich. Fachfremde Teilnehmende wurden ausschließlich mit dem Unity-internen Workflow konfrontiert, wodurch kein direkter Vergleich zur herkömmlichen Methode gezogen werden konnte. Eine solche Gegenüberstellung hätte vermutlich zu einer differenzierteren Bewertung der Effizienz geführt, war jedoch aus zeitlichen Gründen nicht realisierbar.

Speziellere Nachfrage

Während die Experten gezielt befragt wurden, ob die Nutzung eines einzigen Programms den Feedbackprozess effizienter macht, wurde den fachfremden Personen diese Frage nicht in derselben Form gestellt. Zwar fehlte ihnen der direkte Vergleich zu einem herkömmlichen Workflow mit mehreren Programmen, dennoch hätte eine gezielte Nachfrage dahingehend sinnvoll sein können, ob die ausschließliche Arbeit in einem Programm zur positiven Bewertung des Prozesses beigetragen hat. Dadurch hätten zusätzliche Rückschlüsse über die Wahrnehmung der Nutzerfreundlichkeit und Klarheit des Feedbackprozesses gewonnen werden können.

Subjektivität bei der Durchführung der Studie

Da die Einführung in das Modell und der Ablauf des Feedbackprozesses durch die gleiche Person erfolgte, die auch für die Entwicklung verantwortlich war, lässt sich eine gewisse Subjektivität nicht ausschließen. Unbewusste Einflussnahmen, wie z.B. durch den Tonfall, Formulierungen oder Reaktionen,

könnten die Antworten der Teilnehmenden beeinflusst haben. Eine standardisierte, neutrale Präsentationsform, z. B. durch ein Video, wäre gegebenenfalls sinnvoll gewesen.

Wahl des Modells

Im Verlauf der Untersuchung zeigte sich, dass die Entscheidung, eine Pflanze als Beispielmodell zu wählen, methodische Einschränkungen mit sich brachte. Fachliche Rückmeldungen sowie eigene Erfahrungen verdeutlichten, dass Pflanzen im Bereich der 3D-Modellierung ein besonders komplexes Thema darstellen – unter anderem wegen der Erwartung vielfältiger Wachstumsformen, organischer Strukturen und prozeduraler Prozesse. Zudem bestehen in der Branche etablierte Workflows zur Darstellung von Pflanzen, die mit optimierten Meshes und Shadern arbeiten. Diese Ausgangslage erschwerte die Beurteilung, inwiefern die vorgestellte Methode auch für andere Modelltypen übertragbar ist.

Zeitlicher Rahmen als begrenzender Faktor

Nach Abschluss der Studie konnten lediglich ausgewählte Optimierungsvorschläge aus dem Feedback der Teilnehmenden umgesetzt werden. Aufgrund des engen zeitlichen Rahmens war es nicht möglich, alle Hinweise vollständig zu berücksichtigen oder weiterführende Iterationen des Modells zu erstellen. Entsprechend bildet die finale Version des Modells nur einen Zwischenstand im Entwicklungsprozess ab und nicht das volle Potenzial, das durch umfassendere Weiterentwicklung möglich gewesen wäre.

6.6 Empfehlungen für weiterführende Forschung

Aus den zuvor benannten methodischen Beschränkungen sowie den Rückmeldungen der Teilnehmenden lassen sich mehrere Empfehlungen für zukünftige Forschungsvorhaben ableiten. Diese betreffen sowohl die methodische Herangehensweise als auch die Wahl und Gestaltung des zugrunde liegenden Modells.

Erweiterung und Anpassung der Methodik

Künftige Studien sollten mit einem größeren Stichprobenumfang durchgeführt werden, um die Aussagekraft der Ergebnisse zu erhöhen und gegebenenfalls repräsentativere Rückschlüsse zu ermöglichen. Besonders sinnvoll wäre zudem ein direkter Vergleich zwischen zwei Arbeitsweisen: Einer Modellierung mit durchgängigem Workflow in Unity und einer konventionellen Vorgehensweise mit einem externen DCC-Tool. Die Integration solcher Vergleichsszenarien könnte differenziertere Bewertungen zur Effizienz und Praxistauglichkeit ermöglichen.

Auch die Art der Befragung sollte angepasst werden. So könnten gezieltere Fragen – insbesondere an fachfremde Teilnehmende – Aufschluss darüber geben, welche konkreten Faktoren zur positiven Bewertung des Feedbackprozesses beigetragen haben. Die Erhebung könnte darüber hinaus neutraler gestaltet werden, um eine unbewusste, subjektive Beeinflussung zu vermeiden.

Überarbeitung oder Neuwahl des Modells

Die Wahl eines Pflanzenmodells hat sich, wie zuvor beschrieben, im Nachhinein als herausfordernd erwiesen. Daher erscheint es für zukünftige Studien sinnvoll, das zugrunde liegende Modell zu überarbeiten oder durch eine Alternative zu ersetzen. Ein geeignetes Beispiel wären Anwendungsfälle im Bereich der (Innen-)Architektur: Hier lassen sich klar strukturierte, nicht-organische Modelle mit modularen Elementen (z. B. Raumaufteilungen oder Möbelkonfigurationen) abbilden, bei denen Änderungswünsche von Kunden realistisch und nachvollziehbar umgesetzt werden können. Gleichzeitig entfällt der Bedarf nach komplexer Logik oder hochdetaillierter, prozeduraler Modellierung, wodurch sich der Fokus stärker auf die Effizienz des Feedbackprozesses richten lässt.

7 Fazit

Die vorliegende Arbeit untersuchte, ob sich in Unity, unter Nutzung des integrierten Tools ProBuilder, ein komplexes, modular aufgebautes und modifizierbares 3D-Modell erstellen lässt, das zur Effizienzsteigerung von Feedbackprozessen mit Kunden beitragen kann. Im Mittelpunkt stand dabei die Entwicklung eines prototypischen Pflanzenmodells, dessen Modularität und Modifizierbarkeit durch Nutzerfeedback überprüft und weiterentwickelt wurde.

Die Untersuchung ergab, dass die Umsetzung eines technisch und gestalterisch anspruchsvollen 3D-Modells in Unity mithilfe des ProBuilder Packages prinzipiell möglich ist. Das entwickelte Modell wurde sowohl von fachfremden als auch fachkundigen Teilnehmenden überwiegend als modular und modifizierbar bewertet. Die Integration zusätzlicher Funktionen, wie beispielsweise eines interaktiven Sliders zur Steuerung der Modellvarianten, verdeutlichte das Potenzial für dynamische Anpassungen und iterative Modellveränderungen in Echtzeit.

Der Entwicklungsprozess selbst stellte sich – insbesondere aus Sicht der durchführenden Person – als herausfordernd dar. Die Einarbeitung in ProBuilder war zeitintensiv, da die Funktionstiefe und Bedienlogik hinter der von professionellen DCC-Tools wie Blender zurückbleibt. Auch die Experten beurteilten ProBuilder nicht als vollwertige Alternative zu spezialisierten Modellierungsprogrammen, erkannten jedoch dessen Potenzial für frühe Projektphasen oder prototypische Anwendungen an.

Die Evaluation des Feedbackprozesses ergab, dass dieser von fachfremden Teilnehmenden als besonders effizient und nachvollziehbar wahrgenommen wurde. Auch die Experten sahen Vorteile in einem integrierten Workflow, bei dem die Arbeit ausschließlich in einer Software stattfindet. Dennoch zeigte sich, dass etablierte Arbeitsprozesse und Toolpräferenzen in der Praxis häufig überwiegen – selbst wenn alternative Workflows theoretisch effizienter sein könnten.

Insgesamt zeigt die Arbeit, dass Unity mit ProBuilder unter bestimmten Voraussetzungen für die Entwicklung modularer, modifizierbarer Modelle geeignet ist. Für zukünftige Anwendungen und Forschungsarbeiten bietet die vorgestellte Methode eine interessante Grundlage – insbesondere dann, wenn schnelle Iterationen, Echtzeit-Präsentationen und eine enge Zusammenarbeit mit Kunden im Fokus stehen. Eine Weiterentwicklung des gezeigten Ansatzes könnte darin bestehen, das Modell gezielt funktional auszubauen, z.B. durch den Einsatz optimierter Meshes oder prozeduraler Logik. Ebenso denkbar wäre es, alternative Modelltypen zu untersuchen, die sich aufgrund ihrer Struktur besser für modulare und kundennahe Anpassungen eignen. Besonders Szenarien aus der Architektur oder Produktkonfiguration erscheinen vielversprechend, da sie sich durch klar definierte Strukturen und wiederkehrende Elemente auszeichnen und damit ideale Anwendungsfelder für ein integriertes Modellierungs- und Feedbacksystem in Unity darstellen.

Quellenverzeichnis

Autodesk Inc. (Hrsg.). (2024, 12. November). *Immersive Design Review Workspace | Autodesk Workshop XR*. Workshop XR. Abgerufen am 19. März 2025, von https://workshopxr.autodesk.com/

Blender Foundation. (o. D.-a). *History — Blender.org*. blender.org. Abgerufen am 1. Januar 2025, von https://www.blender.org/about/history/

Blender Foundation (Hrsg.). (o. D.-b). *VR Scene Inspection — Blender Manual*. Blender 2.83 Manual. Abgerufen am 25. April 2025, von https://docs.blender.org/manual/en/2.83/addons/3d_view/vr_scene_inspection.html

Carlson, W. E. (2017). *Computer Graphics and Computer Animation: A Retrospective Overview*.

Carlson, W., Hackathorn, R. & Parent, R. (2021). Computer Graphics and Animation at The Ohio State University. *IEEE Computer Graphics And Applications*, *41*(3), 8–17. https://doi.org/10.1109/mcg.2021.3070624

Corke, G. (2024, 31. Januar). *Autodesk Workshop XR launches for AEC collaboration*. AEC Magazine. Abgerufen am 25. April 2025, von https://aecmag.com/vr-mr/autodesk-workshop-xr-launches-for-aec-collaboration/

Di Marco, G. (2019). A Reasoned Approach to the Integration of Design and Fabrication Technologies in Architecture Education. *KnE Social Sciences*. https://doi.org/10.18502/kss.v3i27.5553

Digital Opus (Hrsg.). (o. D.). *Mesh Baker – Digital Opus*. Abgerufen am 27. April 2025, von https://digitalopus.ca/site/mesh-baker/

DWDS (Hrsg.). (2024, 14. Mai). *Modular*. DWDS. Abgerufen am 25. April 2025, von https://www.dwds.de/wb/modular

DWDS (Hrsg.). (2025, Januar). *Modifizieren*. DWDS. Abgerufen am 25. April 2025, von https://www.dwds.de/wb/modifizieren

Europa-Universität Flensburg (Hrsg.). (2023, 25. April). *Spiegelstereoskop nach Wheatstone - Institut Physik - Europa-Universität Flensburg (EUF)*. Abgerufen am 23. April 2025, von https://www.uni-flensburg.de/physik/histolab/thematische-sammlung/optik/spiegelstereoskop-nach-wheatstone

Freebird XR (Hrsg.). (o. D.). *Roadmap & Pricing*. Freebird XR. Abgerufen am 25. April 2025, von https://freebirdxr.com/roadmap/

Government Art Collection (Hrsg.). (2025, 15. Januar). *Panorama: London from the Roof of Albion Mills - Government Art Collection*. Government Art Collection. https://artcollection.dcms.gov.uk/artwork/1222/

Gravity Sketch. (2024, 12. Dezember). *Home - Gravity Sketch*. Abgerufen am 2. Januar 2025, von https://gravitysketch.com/

Introduction to Gravity Sketch VR - PART 2. (2020, 17. Februar). Abgerufen am 25. April 2025, von https://www.3dconceptartist.com/blog/introduction-to-gravity-sketch-in-vr-part-2-of-3

Jackèl, D., Neunreither, S. & Wagner, F. (2006). Methoden der Computeranimation. In *eXamen.press*. https://doi.org/10.1007/3-540-33407-6

Lenk, S. (o. D.). *Zoetrope [Das Lexikon der Filmbegriffe]*. Abgerufen am 7. Dezember 2024, von https://filmlexikon.uni-kiel.de/doku.php/z:zoetrope-852

media storehouse. (Hrsg.). (2025). *Phanakistoscope*. Media Storehouse. Abgerufen am 22. April 2025, von https://www.mediastorehouse.com/mary-evans-prints-online/phanakistoscope-580023.html

Mollah, S. (2022, 5. Januar). Unity3D Draw Call Batching and how it works? - Saharukh Mollah - Medium. *Medium*. https://saharukh.medium.com/unity3d-draw-call-batching-and-how-it-works-7624ce5bc255

Mount, D. (2018). Procedural Generation: Perlin Noise. In *CMSC 425: Lecture 14*.

Okuya, Y., Ladeveze, N., Fleury, C. & Bourdot, P. (2018). ShapeGuide: Shape-Based 3D Interaction for Parameter Modification of Native CAD Data. *Frontiers in Robotics And AI*, *5*. https://doi.org/10.3389/frobt.2018.00118

Pan, X. & Haberkorn, A. (2021). Interactive 3D Modeling with Virtual Reality. *Electronic Imaging*, *33*(8), 280–287. https://doi.org/10.2352/issn.2470-1173.2021.8.imawm-280

Pyfer & James. (2014, 10. Juni). *SketchPad | Interactive Drawing, Vector Graphics & CAD*. Encyclopedia Britannica. Abgerufen am 19. März 2025, von https://www.britannica.com/technology/Sketchpad

Schmidt, U. (2013). *Professionelle Videotechnik: Grundlagen, Filmtechnik, Fernsehtechnik, Geräte- und Studiotechnik in SD, HD, DI, 3D*. Springer Vieweg.

Sensorama Morton Heilig. (2018, Januar). ResearchGate. Abgerufen am 23. April 2025, von https://www.researchgate.net/figure/Sensorama-Morton-Heilig-8_fig3_340144636

Sharma, M. (2023, 14. November). Evolution of Computer Graphics - Mitesh Sharma - Medium. *Medium*. https://medium.com/@miteshryp/evolution-of-computer-graphics-b970be31d2c9

Sutherland, I. E. (1964). Sketchpad a Man-Machine Graphical Communication System. *SIMULATION*, *2*(5), R-20. https://doi.org/10.1177/003754976400200514

Sutherland, I. E., Information Processing Techniques & Office, ARPA, OSD. (1965). *The ultimate display*. https://worrydream.com/refs/Sutherland_1965_-_The_Ultimate_Display.pdf

Trickbüro Basil Vogt (Hrsg.). (o. D.). *Objekt*. Abgerufen am 22. April 2025, von https://trickbuero.ch/objekt.html

Unity Technologies (Hrsg.). (o. D.). *Unity - Scripting API: MathF.PerlinNoise*. Unity Documentation. Abgerufen am 25. April 2025, von https://docs.unity3d.com/6000.0/Documentation/ScriptReference/Mathf.PerlinNoise.html

Unity Technologies (Hrsg.). (2023). *MeshSync*. Abgerufen am 28. Dezember 2024, von https://docs.unity3d.com/Packages/com.unity.meshsync@0.17/manual/index.html

Unity Technologies (Hrsg.). (2024a). *About Pixyz Plugin*. Pixyz Plugin For Unity 3.0.6. Abgerufen am 2. Januar 2025, von https://docs.unity3d.com/Packages/com.unity.industry.toolkit@3.0/manual/index.html

Unity Technologies (Hrsg.). (2024b). *About ProBuilder*. ProBuilder 5.2.3. Abgerufen am 2. Januar 2025, von https://docs.unity3d.com/Packages/com.unity.probuilder@6.0/manual/index.html

Unity Technologies. (2024c). *Prepare and optimize models | Pixyz Plugin for Unity | 3.1.1*. Abgerufen am 25. April 2025, von https://docs.unity3d.com/Packages/com.unity.industry.toolkit@3.1/manual/prepare-optimize-model.html

Unity Technologies. (2024d). *Real-time tools for 3D, AR, and VR development | Products*. Unity. Abgerufen am 25. April 2025, von https://unity.com/products?c=tools+marketplace

Unity Technologies (Hrsg.). (2024e). *Transform your 3D Data with Pixyz Plugin | Unity*. Unity. Abgerufen am 19. März 2025, von https://unity.com/products/pixyz-plugin

Unity Technologies (Hrsg.). (2024f). *Unity - Manual: Introduction to optimizing draw calls*. Abgerufen am 27. April 2025, von https://docs.unity3d.com/6000.0/Documentation/Manual/optimizing-draw-calls.html?utm_source=chatgpt.com

Weston, Z. (2023, 2. Oktober). Title: The Evolution of Adobe: A Journey Through History. *Medium*. https://medium.com/@zacharywestonintech/title-the-evolution-of-adobe-a-journey-through-history-21b1c029174

Who's Who of Victorian Cinema. (2025). Abgerufen am 23. April 2025, von https://www.victorian-cinema.net/grimoinsanson.php

Williams, R. (2002). *The animator's survival kit: A Manual of Methods, Principles and Formulas for Classical, Computer, Games, Stop Motion and Internet Animators*. Faber & Faber.

Wölfel, M. (2023). *Immersive virtuelle Realität*. Springer Vieweg.

Anhang

Anhang A.1: C#-Skript „Erde_2.cs“

```
using UnityEngine;
using UnityEngine.ProBuilder;
using UnityEngine.ProBuilder.MeshOperations;
using System.Linq;

[ExecuteAlways]
[RequireComponent(typeof(ProBuilderMesh))]
public class Erde_2 : MonoBehaviour
{
    [Header("Noise-Einstellungen")]
    public float noiseScale = 5f;
    public float noiseStrength = 0.01f;
    public float offsetX = 100f;
    public float offsetZ = 200f;

    private ProBuilderMesh pbMesh;
    private Vector3[] originalPositions;

    private void OnEnable()
    {
        pbMesh = GetComponent<ProBuilderMesh>();
        if (pbMesh != null)
        {
            if (originalPositions == null || originalPositions.Length !=
pbMesh.positions.Count)
            {
                originalPositions = pbMesh.positions.ToArray();
            }
            DeformWholeMesh();
        }
    }

    private void OnValidate()
    {
        if (pbMesh == null)
            pbMesh = GetComponent<ProBuilderMesh>();

        if (!Application.isPlaying && pbMesh != null && originalPositions
!= null)
        {
            DeformWholeMesh();
        }
    }

    private void DeformWholeMesh()
    {
        if (pbMesh == null || originalPositions == null)
            return;

        Vector3[] verts = new Vector3[originalPositions.Length];
        for (int i = 0; i < originalPositions.Length; i++)
        {
            verts[i] = originalPositions[i];
        }
```

```
        for (int i = 0; i < verts.Length; i++)
        {
            Vector3 v = verts[i];
            float noiseValue = Mathf.PerlinNoise(
                (v.x + offsetX) * noiseScale,
                (v.z + offsetZ) * noiseScale
            );
            float delta = (noiseValue - 0.5f) * noiseStrength;
            v.y += delta;
            verts[i] = v;
        }

        pbMesh.positions = verts;
        pbMesh.ToMesh();
        pbMesh.Refresh();
    }

    public void ResetMesh()
    {
        if (pbMesh != null && originalPositions != null)
        {
            pbMesh.positions = originalPositions.ToArray();
            pbMesh.ToMesh();
            pbMesh.Refresh();
        }
    }
}
```

Anhang A.2: C#-Skript „Erde2_Editor.cs“

```
using UnityEngine;
using UnityEditor;

[CustomEditor(typeof(Erde_2))]
public class Erde_2Editor : Editor
{
    public override void OnInspectorGUI()
    {
        DrawDefaultInspector();

        if (target == null)
        {
            Debug.LogWarning("Target object is null. Skipping custom inspector logic.");
            return;
        }

        Erde_2 script = (Erde_2)target;

        if (script == null)
        {
            Debug.LogWarning("Failed to cast target to Erde_2. Skipping custom inspector logic.");
            return;
        }

        if (GUILayout.Button("Reset Mesh"))
        {
            script.noiseScale = 0f;
            script.noiseStrength = 0f;
            script.offsetX = 100f;
            script.offsetZ = 200f;

            script.ResetMesh();

            EditorUtility.SetDirty(script);
        }
    }
}
```

Anhang A.3: C#-Skript „LeafSwitcher.cs“

```
using UnityEngine;
#if UNITY_EDITOR
using UnityEditor;
#endif

[ExecuteAlways]
public class LeafSwitcher : MonoBehaviour
{
    public GameObject[] leafPrefabs;

    [SerializeField]
    private int selectedLeafIndex = 0;
    private bool isUpdateScheduled = false;
    private void OnValidate()
    {
        if (!isUpdateScheduled)
        {
            isUpdateScheduled = true;
#if UNITY_EDITOR
            EditorApplication.delayCall += DelayedUpdateLeaf;
#else
            DelayedUpdateLeaf();
#endif
        }
    }

    private void DelayedUpdateLeaf()
    {
        if (this == null) return;
        isUpdateScheduled = false;
        UpdateLeaf();
    }

    private void ClearExistingLeaf()
    {
        while (transform.childCount > 0)
        {
            Transform child = transform.GetChild(0);
#if UNITY_EDITOR
            DestroyImmediate(child.gameObject);
#else
            Destroy(child.gameObject);
#endif
        }
    }

    private void UpdateLeaf()
    {
        if (leafPrefabs == null || leafPrefabs.Length == 0)
            return;

        selectedLeafIndex = Mathf.Clamp(selectedLeafIndex, 0, leafPre-
fabs.Length - 1);

        ClearExistingLeaf();
```

```
        if (leafPrefabs[selectedLeafIndex] != null)
        {
            GameObject newLeaf = Instantiate(leafPrefabs[selected-
LeafIndex], transform);
            newLeaf.transform.localPosition = Vector3.zero;
            newLeaf.transform.localRotation = Quaternion.identity;
            newLeaf.transform.localScale = Vector3.one;
        }
    }
}
```

Anhang A.4: C#-Skript „PlantPotManagerWithSlider.cs“

```
#if UNITY_EDITOR
using UnityEditor;
#endif
using System.Collections.Generic;
using UnityEngine;

[ExecuteAlways]
public class PlantPotManagerWithSlider : MonoBehaviour
{
    [Header("Slider: -42 bis 0: Originalteile ein-/ausblenden; 0 bis 42:
Duplikate hinzufügen")]
    [Range(-42, 42)]
    [SerializeField]
    private int elementSlider = 0;
    private List<GameObject> originalParts = new List<GameObject>();
    private List<GameObject> duplicateParts = new List<GameObject>();
    private List<int> hiddenIndices = new List<int>();
    private int originalCount = -1;
    private int lastSliderValue = 0;

    private void Awake()
    {
        InitializeOriginalParts();
        lastSliderValue = elementSlider;
    }

    private void InitializeOriginalParts()
    {
        originalParts.Clear();
        foreach (Transform child in transform)
        {
            if (!child.gameObject.name.EndsWith("_copy"))
                originalParts.Add(child.gameObject);
        }
        originalCount = originalParts.Count;
        foreach (GameObject orig in originalParts)
        {
            if (orig) orig.SetActive(true);
        }
        hiddenIndices.Clear();
        RemoveAllDuplicates();
    }

    private void Update()
    {
        if (originalCount < 0 || originalParts.Count == 0)
            InitializeOriginalParts();

        if (elementSlider != lastSliderValue)
        {
            ApplyChanges();
            lastSliderValue = elementSlider;
        }
    }
```

```
    private void ApplyChanges()
    {
        if (elementSlider < 0)
        {
            int desiredVisible = originalCount + elementSlider;
            int currentVisible = 0;
            for (int i = 0; i < originalParts.Count; i++)
            {
                if (originalParts[i] != null && originalParts[i].ac-
tiveSelf)
                    currentVisible++;
            }

            if (currentVisible > desiredVisible)
            {
                List<int> candidates = new List<int>();
                for (int i = 0; i < originalParts.Count; i++)
                {
                    if (originalParts[i] != null && originalParts[i].ac-
tiveSelf && !hiddenIndices.Contains(i))
                        candidates.Add(i);
                }
                if (candidates.Count > 0)
                {
                    int randomIndex = candidates[Random.Range(0, candi-
dates.Count)];
                    originalParts[randomIndex].SetActive(false);
                    hiddenIndices.Add(randomIndex);
                }
            }
            else if (currentVisible < desiredVisible && hiddenIndices.Count
> 0)
            {
                int indexToUnhide = hiddenIndices[0];
                if (originalParts[indexToUnhide] != null)
                    originalParts[indexToUnhide].SetActive(true);
                hiddenIndices.RemoveAt(0);
            }
            RemoveAllDuplicates();
        }
        else
        {
            foreach (GameObject orig in originalParts)
            {
                if (orig != null)
                    orig.SetActive(true);
            }
            hiddenIndices.Clear();

            int targetDuplicates = elementSlider;
            AdjustDuplicates(targetDuplicates);
        }
    }
```

```
    private void AdjustDuplicates(int targetDuplicates)
    {
        duplicateParts.Clear();
        foreach (Transform child in transform)
        {
            if (child.gameObject.name.EndsWith("_copy"))
                duplicateParts.Add(child.gameObject);
        }

        if (duplicateParts.Count < targetDuplicates)
        {
            while (duplicateParts.Count < targetDuplicates)
            {
                AddDuplicate();
                duplicateParts.Clear();
                foreach (Transform child in transform)
                {
                    if (child.gameObject.name.EndsWith("_copy"))
                        duplicateParts.Add(child.gameObject);
                }
            }
        }
        else if (duplicateParts.Count > targetDuplicates)
        {
            RemoveLastDuplicate();
        }
    }

    private void AddDuplicate()
    {
        if (originalParts.Count == 0)
            return;

        GameObject original = originalParts[Random.Range(0, original-
Parts.Count)];
        if (original == null)
            return;

        Vector3 pivot = transform.position;
        Vector3 origRel = original.transform.position - pivot;
        Vector3 newPos = pivot - origRel;
        newPos.y = original.transform.position.y;

        GameObject dup = Instantiate(original, transform);
        dup.name = original.name + "_copy";
        dup.transform.position = newPos;
        Vector3 origEuler = original.transform.rotation.eulerAngles;
        dup.transform.rotation = Quaternion.Euler(origEuler.x, origEuler.y
+ 180f, origEuler.z);
        float scaleFactor = Random.Range(0.9f, 1.1f);
        dup.transform.localScale = original.transform.localScale * scale-
Factor;

        duplicateParts.Add(dup);
    }
```

```
    private void RemoveLastDuplicate()
    {
        duplicateParts.Clear();
        foreach (Transform child in transform)
        {
            if (child.gameObject.name.EndsWith("_copy"))
                duplicateParts.Add(child.gameObject);
        }
        if (duplicateParts.Count == 0)
            return;
        GameObject dup = duplicateParts[duplicateParts.Count - 1];
        duplicateParts.RemoveAt(duplicateParts.Count - 1);
        DestroyImmediate(dup);
    }

    private void RemoveAllDuplicates()
    {
        List<GameObject> toRemove = new List<GameObject>();
        foreach (Transform child in transform)
        {
            if (child.gameObject.name.EndsWith("_copy"))
                toRemove.Add(child.gameObject);
        }
        foreach (GameObject dup in toRemove)
        {
            DestroyImmediate(dup);
        }
        duplicateParts.Clear();
    }
}
```

Anhang B.1: Blanko Umfragebogen

Auswertung – Befragung

Person Nr.:

Alter:

Berufsbezeichnung:

Erfahrung mit 3D/VR-Tools:

Falls ja, mit welchen:

Bitte bewerte die folgenden Aussagen:

Das Modell ist modifizierbar.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Das Modell ist modular. (modular = aus austauschbaren und miteinander kombinierbaren Komponenten bestehend)			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Anpassungen/Änderungswünsche können schnell umgesetzt werden.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Optional: Welche Anpassungsmöglichkeiten haben deiner Meinung nach gefehlt oder können/konnten nicht schnell genug umgesetzt werden?

Wie viele Punkte würdest du dem Feedback-Prozess geben? (1=schlecht, 10=ausgezeichnet)

1 2 3 4 5 6 7 8 9 10

Falls du mit dem Feedback-Prozess nicht zufrieden warst: Woran hat dies gelegen? (Eingeschränkte Möglichkeiten des Tools, fehlende Erfahrung der Nutzerin etc.)

NUR FÜR PERSONEN MIT ERFAHRUNG IN 3D-MODELLIERUNG:

Bitte gib deine Einschätzung über die Arbeit mit ProBuilder/Unity zu Erstellung eines 3D-Modells ab, indem du die folgenden Aussagen bewertest. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Mit ProBuilder bzw. Unity lässt sich (mit ausreichender Bedienungssicherheit/Routine) ein komplexes, modulares und modifizierbares 3D-Modell erstellen.				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Die Erstellung eines 3D-Modells mit ProBuilder bzw. Unity wirkt wie eine gute Alternative zu der herkömmlichen Methode (z.B. mit Blender).				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Der Feedback-Prozess mit Kund*innen kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden. (Bei der Präsentation einer VR-Umgebung in Unity, ohne den Wechsel zu Blender bei Änderungen am Modell)				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Kannst du dir vorstellen in Zukunft auf die vorgestellte Weise 3D-Modelle zu erstellen? Bitte begründe deine Antwort. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Die letzten Fragen dürfen gerne wieder von allen beantwortet werden. :)

Hast du noch letzte Anmerkungen oder Gedanken zu dem vorgestellten Projekt?

Anhang B.2: Ausgefüllte Umfragebögen

Auswertung – Befragung

Person Nr.:	1
Alter:	25
Berufsbezeichnung:	Managerin Mitarbeiter- und Medienkommunikation
Erfahrung mit 3D/VR-Tools:	Nein
Falls ja, mit welchen:	-

Bitte bewerte die folgenden Aussagen:

Das Modell ist modifizierbar.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Das Modell ist modular. (modular = aus austauschbaren und miteinander kombinierbaren Komponenten bestehend)			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Anpassungen/Änderungswünsche können schnell umgesetzt werden.			
Trifft zu	**Trifft teilweise zu**	Trifft eher nicht zu	Trifft nicht zu

Optional: Welche Anpassungsmöglichkeiten haben deiner Meinung nach gefehlt oder können/konnten nicht schnell genug umgesetzt werden?

-

Wie viele Punkte würdest du dem Feedback-Prozess geben? (1=schlecht, 10=ausgezeichnet)

1	2	3	4	5	6	7	**8**	9	10

Falls du mit dem Feedback-Prozess nicht zufrieden warst: Woran hat dies gelegen? (Eingeschränkte Möglichkeiten des Tools, fehlende Erfahrung der Nutzerin etc.)

-

NUR FÜR PERSONEN MIT ERFAHRUNG IN 3D-MODELLIERUNG:

Bitte gib deine Einschätzung über die Arbeit mit ProBuilder/Unity zu Erstellung eines 3D-Modells ab, indem du die folgenden Aussagen bewertest. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Mit ProBuilder bzw. Unity lässt sich (mit ausreichender Bedienungssicherheit/Routine) ein komplexes, modulares und modifizierbares 3D-Modell erstellen.				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Die Erstellung eines 3D-Modells mit ProBuilder bzw. Unity wirkt wie eine gute Alternative zu der herkömmlichen Methode (z.B. mit Blender).				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Der Feedback-Prozess mit Kund*innen kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden. (Bei der Präsentation einer VR-Umgebung in Unity, ohne den Wechsel zu Blender bei Änderungen am Modell)				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Kannst du dir vorstellen in Zukunft auf die vorgestellte Weise 3D-Modelle zu erstellen? Bitte begründe deine Antwort. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

-

Die letzten Fragen dürfen gerne wieder von allen beantwortet werden. :)

Hast du noch letzte Anmerkungen oder Gedanken zu dem vorgestellten Projekt?

-

Auswertung – Befragung

Person Nr.:	2
Alter:	30
Berufsbezeichnung:	DOP/Editor für immersive Medien
Erfahrung mit 3D/VR-Tools:	Ja
Falls ja, mit welchen:	Die meiste Erfahrung habe ich mit Blender. Unity nur ganz rudimentär.

Bitte bewerte die folgenden Aussagen:

Das Modell ist modifizierbar.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Das Modell ist modular. (modular = aus austauschbaren und miteinander kombinierbaren Komponenten bestehend)			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Anpassungen/Änderungswünsche können schnell umgesetzt werden.			
Trifft zu	**Trifft teilweise zu**	Trifft eher nicht zu	Trifft nicht zu

Optional: Welche Anpassungsmöglichkeiten haben deiner Meinung nach gefehlt oder können/konnten nicht schnell genug umgesetzt werden?

Das Modell ist zwar modular, aber nicht prozedural. Änderungen haben keinen Bezug zueinander. ZB.: Ist die Erde trocken, so ändert sich automatisch die Blattfarbe oder die maximale Anzahl der Stiele ist ändert sich mit dem Durchmesser des Topfes. Es gibt für den Kunden keine Möglichkeit einen Überblick über die modularen Option zu bekommen. Ein UI würde helfen alles zentral zu verwalten und zu bedienen. So könnte evtl. auch der Kunde selbst Änderungen vornehmen.

Wie viele Punkte würdest du dem Feedback-Prozess geben? (1=schlecht, 10=ausgezeichnet)

1	2	3	4	5	6	7	8	**9**	10

Falls du mit dem Feedback-Prozess nicht zufrieden warst: Woran hat dies gelegen? (Eingeschränkte Möglichkeiten des Tools, fehlende Erfahrung der Nutzerin etc.)

-

NUR FÜR PERSONEN MIT ERFAHRUNG IN 3D-MODELLIERUNG:

Bitte gib deine Einschätzung über die Arbeit mit ProBuilder/Unity zu Erstellung eines 3D-Modells ab, indem du die folgenden Aussagen bewertest. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Mit ProBuilder bzw. Unity lässt sich (mit ausreichender Bedienungssicherheit/Routine) ein komplexes, modulares und modifizierbares 3D-Modell erstellen.				
Trifft zu	Trifft eher zu	**Neutral/kann ich nicht sagen**	Trifft eher nicht zu	Trifft nicht zu

Die Erstellung eines 3D-Modells mit ProBuilder bzw. Unity wirkt wie eine gute Alternative zu der herkömmlichen Methode (z.B. mit Blender).				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	**Trifft eher nicht zu**	Trifft nicht zu

Der Feedback-Prozess mit Kund*innen kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden. (Bei der Präsentation einer VR-Umgebung in Unity, ohne den Wechsel zu Blender bei Änderungen am Modell)				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Kannst du dir vorstellen in Zukunft auf die vorgestellte Weise 3D-Modelle zu erstellen? Bitte begründe deine Antwort. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Ja, aber nicht mit Unity. Da ich mich mit Blender viel besser auskenne würde wahrscheinlich eher zu Blender greifen. Ein gutes modulares Konzept kann die Entscheidungsfindung beim Kunden erheblich vereinfachen. Der Kunde kann durch kurze, schnelle Iterationen Ideen oder Vorschläge kostengünstig ausprobieren. Der Workflowprozess ist dadurch effizienter, skalierbar und damit auf lange Sicht kostengünstiger.

Die letzten Fragen dürfen gerne wieder von allen beantwortet werden. :)

Hast du noch letzte Anmerkungen oder Gedanken zu dem vorgestellten Projekt?

-

Auswertung – Befragung

Person Nr.:	3
Alter:	32
Berufsbezeichnung:	3D Artist
Erfahrung mit 3D/VR-Tools:	Ja
Falls ja, mit welchen:	Blender

Bitte bewerte die folgenden Aussagen:

Das Modell ist modifizierbar.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Das Modell ist modular. (modular = aus austauschbaren und miteinander kombinierbaren Komponenten bestehend)			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Anpassungen/Änderungswünsche können schnell umgesetzt werden.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Optional: Welche Anpassungsmöglichkeiten haben deiner Meinung nach gefehlt oder können/konnten nicht schnell genug umgesetzt werden?

Die Menge an Anpassungsmöglichkeiten ist schon sehr gut. Die Anpassungen könnten wahrscheinlich noch schneller umgesetzt werden, wenn sie nicht alle einzeln und händisch durchgeführt werden müssten. Das könnte z.B. durch ein UI mit Schiebereglern gelöst werden, die eine schnelle Änderung beispielsweise der Anzahl der Stengel, ihrer Neigung oder Rotation oder anderer Faktoren ermöglichen.

Wie viele Punkte würdest du dem Feedback-Prozess geben? (1=schlecht, 10=ausgezeichnet)

1	2	3	4	5	6	7	8	**9**	10

Falls du mit dem Feedback-Prozess nicht zufrieden warst: Woran hat dies gelegen? (Eingeschränkte Möglichkeiten des Tools, fehlende Erfahrung der Nutzerin etc.)

-

NUR FÜR PERSONEN MIT ERFAHRUNG IN 3D-MODELLIERUNG:

Bitte gib deine Einschätzung über die Arbeit mit ProBuilder/Unity zu Erstellung eines 3D-Modells ab, indem du die folgenden Aussagen bewertest. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Mit ProBuilder bzw. Unity lässt sich (mit ausreichender Bedienungssicherheit/Routine) ein komplexes, modulares und modifizierbares 3D-Modell erstellen.				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Die Erstellung eines 3D-Modells mit ProBuilder bzw. Unity wirkt wie eine gute Alternative zu der herkömmlichen Methode (z.B. mit Blender).				
Trifft zu	**Trifft eher zu**	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Der Feedback-Prozess mit Kund*innen kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden. (Bei der Präsentation einer VR-Umgebung in Unity, ohne den Wechsel zu Blender bei Änderungen am Modell)				
Trifft zu	**Trifft eher zu**	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Kannst du dir vorstellen in Zukunft auf die vorgestellte Weise 3D-Modelle zu erstellen? Bitte begründe deine Antwort. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Ja, von der grundsätzlichen Idee des modularen Aufbaus her kann ich mir das vorstellen. Für kurze Feedbackschleifen kann ich mir die modulare Bauweise als sinnvoll vorstellen, da ziemlich genaue Anpassungen entsprechend des Feedbacks möglich sind. Für die personalisierte Modellierung von einer überschaubaren Menge an Objekten funktioniert das Prinzip so wie es ist schon ganz gut. Das könnte z.B. für die Gestaltung von Objekten für den 3D-Druck nützlich sein, wenn man Kunden auf ihre Wünsche angepasste Variationen anbieten möchte.

Die letzten Fragen dürfen gerne wieder von allen beantwortet werden. :)

Hast du noch letzte Anmerkungen oder Gedanken zu dem vorgestellten Projekt?

-

Auswertung – Befragung

Person Nr.:	4
Alter:	32
Berufsbezeichnung:	3D Artist
Erfahrung mit 3D/VR-Tools:	Ja
Falls ja, mit welchen:	Blender, Unity, Unreal, Godot. Viele weitere.

Bitte bewerte die folgenden Aussagen:

Das Modell ist modifizierbar.			
Trifft zu	**Trifft teilweise zu**	Trifft eher nicht zu	Trifft nicht zu

Das Modell ist modular. (modular = aus austauschbaren und miteinander kombinierbaren Komponenten bestehend)			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Anpassungen/Änderungswünsche können schnell umgesetzt werden.			
Trifft zu	**Trifft teilweise zu**	Trifft eher nicht zu	Trifft nicht zu

Optional: Welche Anpassungsmöglichkeiten haben deiner Meinung nach gefehlt oder können/konnten nicht schnell genug umgesetzt werden?

Pflanzen, die andere Grundstrukturen haben. Im Moment gibt das Template der Pflanze vor, dass alle Blätter direkt aus der Erde wachsen. In wirklichkeit sind Pflanzen viel diverser. Eine typische Beispielpflanze wäre ein Drachenbaum. Dort wachsen die Blätter aus einem zentralen Stamm heraus.

Wie viele Punkte würdest du dem Feedback-Prozess geben? (1=schlecht, 10=ausgezeichnet)

1	2	3	4	5	**6**	7	8	9	10

Falls du mit dem Feedback-Prozess nicht zufrieden warst: Woran hat dies gelegen? (Eingeschränkte Möglichkeiten des Tools, fehlende Erfahrung der Nutzerin etc.)

Eingeschränkte Möglichkeiten des Tools. Ich denke, dass gerade bei Pflanzen so viele verschiedene Arten möglich sind, dass es schwer ist jedem Wunsch gerecht zu werden.

NUR FÜR PERSONEN MIT ERFAHRUNG IN 3D-MODELLIERUNG:

Bitte gib deine Einschätzung über die Arbeit mit ProBuilder/Unity zu Erstellung eines 3D-Modells ab, indem du die folgenden Aussagen bewertest. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Mit ProBuilder bzw. Unity lässt sich (mit ausreichender Bedienungssicherheit/Routine) ein komplexes, modulares und modifizierbares 3D-Modell erstellen.				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Die Erstellung eines 3D-Modells mit ProBuilder bzw. Unity wirkt wie eine gute Alternative zu der herkömmlichen Methode (z.B. mit Blender).				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	**Trifft nicht zu**

Der Feedback-Prozess mit Kund*innen kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden. (Bei der Präsentation einer VR-Umgebung in Unity, ohne den Wechsel zu Blender bei Änderungen am Modell)				
Trifft zu	**Trifft eher zu**	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Kannst du dir vorstellen in Zukunft auf die vorgestellte Weise 3D-Modelle zu erstellen? Bitte begründe deine Antwort. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Nur in sehr speziellen Usecases. Ich könnte es mir dann vorstellen, wenn Performance vernachlässigt werden kann und die Konfigurierbarkeit direkt in Unity ein sehr wichtiger Faktor im Workflow ist. In der Regel kann man mit einer guten 3D Asset Pipeline bessere Pflanzen produzieren, wenn man die Pflanzen im externen 3D Tool gut optimiert und modelliert. Danach gibt es in der Regel einen sehr schnellen und reibungslosen Exportprozess um ein Asset von Blender nach Unity zu bekommen.

Die letzten Fragen dürfen gerne wieder von allen beantwortet werden. :)

Hast du noch letzte Anmerkungen oder Gedanken zu dem vorgestellten Projekt?

Sehr schweres Thema, sehr schwer die etablierten Pipelines zu durchbrechen. Es muss einen wirklich guten Grund dafür geben.

Auswertung – Befragung

Person Nr.:	5
Alter:	28
Berufsbezeichnung:	Unity Developer
Erfahrung mit 3D/VR-Tools:	Ja
Falls ja, mit welchen:	Unity 3D, Unreal Engine, Blender, Shapes XR

Bitte bewerte die folgenden Aussagen:

Das Modell ist modifizierbar.			
Trifft zu	**Trifft teilweise zu**	Trifft eher nicht zu	Trifft nicht zu

Das Modell ist modular. (modular = aus austauschbaren und miteinander kombinierbaren Komponenten bestehend)			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Anpassungen/Änderungswünsche können schnell umgesetzt werden.			
Trifft zu	**Trifft teilweise zu**	Trifft eher nicht zu	Trifft nicht zu

Optional: Welche Anpassungsmöglichkeiten haben deiner Meinung nach gefehlt oder können/konnten nicht schnell genug umgesetzt werden?

Komplexere Strukturen sind etwas schwierig mit dem Tool. Generell eignet es sich in der aktuellen Form eher für BLock-Outs von Objekten, als kleinere Details zu ändern.

Wie viele Punkte würdest du dem Feedback-Prozess geben? (1=schlecht, 10=ausgezeichnet)

1	2	3	4	**5**	6	7	8	9	10

Falls du mit dem Feedback-Prozess nicht zufrieden warst: Woran hat dies gelegen? (Eingeschränkte Möglichkeiten des Tools, fehlende Erfahrung der Nutzerin etc.)

Das Tool eignet sich eher für rudimentäre Block-Outs, weil vieles entweder auf Primitives basiert oder vordefiniert sein muss. Die Nutzung sollte am besten immer in Verbindung mit einer Person geschenen, die mit dem Tool geschult ist. Ich denke es würde insbesondere Leute, die nichts mit 3D und VR zu tun haben etwas überfordern, auch wenn der Funktionsumfang in Vergleich zu dezidierten 3D Tool natürlich schon leichter verständlich ist.

NUR FÜR PERSONEN MIT ERFAHRUNG IN 3D-MODELLIERUNG:

Bitte gib deine Einschätzung über die Arbeit mit ProBuilder/Unity zu Erstellung eines 3D-Modells ab, indem du die folgenden Aussagen bewertest. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Mit ProBuilder bzw. Unity lässt sich (mit ausreichender Bedienungssicherheit/Routine) ein komplexes, modulares und modifizierbares 3D-Modell erstellen.				
Trifft zu	Trifft eher zu	**Neutral/kann ich nicht sagen**	Trifft eher nicht zu	Trifft nicht zu

Die Erstellung eines 3D-Modells mit ProBuilder bzw. Unity wirkt wie eine gute Alternative zu der herkömmlichen Methode (z.B. mit Blender).				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	**Trifft eher nicht zu**	Trifft nicht zu

Der Feedback-Prozess mit Kund*innen kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden. (Bei der Präsentation einer VR-Umgebung in Unity, ohne den Wechsel zu Blender bei Änderungen am Modell)				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	**Trifft eher nicht zu**	Trifft nicht zu

Kannst du dir vorstellen in Zukunft auf die vorgestellte Weise 3D-Modelle zu erstellen? Bitte begründe deine Antwort. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Ich kann es mir nicht für finale Assets vorstellen, eher für Block-Outs von 3D Objekten und ganzen Szenen, sollte das Tool weiter entwickelt werden. Für richtige Assets brauch man denke ich einfach den ganzen Funktionsumfang einer 3D Suite und einen guten gestreamlineten Importprozess in die Engine.

Die letzten Fragen dürfen gerne wieder von allen beantwortet werden. :)

Hast du noch letzte Anmerkungen oder Gedanken zu dem vorgestellten Projekt?

Ich denke, dass es sich eher zum Erstellen von Block-Outs eignet und für prozedurale Generatoren mit eingeschränkten Themen zum Beispiel ein Pflanzen- oder Straßen-Generator oder eben sogar ein Level Editor. So kann auch auf eine Stärke der Engine zurückgegriffen werden. Es können nicht nur einfache 3D Objekte platziert werden, sondern auch Prefabs mit Skripten und echter in Game Logik. Sowas wäre bei einem Import aus der 3D Software nicht ohne mühsame Einzelarbeit möglich.

Auswertung – Befragung

Person Nr.:	6
Alter:	26
Berufsbezeichnung:	Cutterin
Erfahrung mit 3D/VR-Tools:	Nein
Falls ja, mit welchen:	-

Bitte bewerte die folgenden Aussagen:

Das Modell ist modifizierbar.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Das Modell ist modular. (modular = aus austauschbaren und miteinander kombinierbaren Komponenten bestehend)			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Anpassungen/Änderungswünsche können schnell umgesetzt werden.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Optional: Welche Anpassungsmöglichkeiten haben deiner Meinung nach gefehlt oder können/konnten nicht schnell genug umgesetzt werden?

-

Wie viele Punkte würdest du dem Feedback-Prozess geben? (1=schlecht, 10=ausgezeichnet)

1	2	3	4	5	6	7	**8**	9	10

Falls du mit dem Feedback-Prozess nicht zufrieden warst: Woran hat dies gelegen? (Eingeschränkte Möglichkeiten des Tools, fehlende Erfahrung der Nutzerin etc.)

Hat manchmal etwas gedauert, bis die jeweilige Funktion/Objekt etc. gefunden wurde

---

NUR FÜR PERSONEN MIT ERFAHRUNG IN 3D-MODELLIERUNG:

Bitte gib deine Einschätzung über die Arbeit mit ProBuilder/Unity zu Erstellung eines 3D-Modells ab, indem du die folgenden Aussagen bewertest. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Mit ProBuilder bzw. Unity lässt sich (mit ausreichender Bedienungssicherheit/Routine) ein komplexes, modulares und modifizierbares 3D-Modell erstellen.				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Die Erstellung eines 3D-Modells mit ProBuilder bzw. Unity wirkt wie eine gute Alternative zu der herkömmlichen Methode (z.B. mit Blender).				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Der Feedback-Prozess mit Kund*innen kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden. (Bei der Präsentation einer VR-Umgebung in Unity, ohne den Wechsel zu Blender bei Änderungen am Modell)				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Kannst du dir vorstellen in Zukunft auf die vorgestellte Weise 3D-Modelle zu erstellen? Bitte begründe deine Antwort. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

-

Die letzten Fragen dürfen gerne wieder von allen beantwortet werden. :)

Hast du noch letzte Anmerkungen oder Gedanken zu dem vorgestellten Projekt?

War sehr cool! Ich frage mich wie gut das funktioniert, wenn man ein noch komplexeres Modell hat, weil das jetzt schon recht unübersichtlich war (für mein ungeschultes Auge) und ob man dann noch gut den Überblick behalten kann und schnell die richtigen Funktionen findet und anpassen kann.

Auswertung – Befragung

Person Nr.:	7
Alter:	29
Berufsbezeichnung:	Producer
Erfahrung mit 3D/VR-Tools:	Nein
Falls ja, mit welchen:	-

Bitte bewerte die folgenden Aussagen:

Das Modell ist modifizierbar.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Das Modell ist modular. (modular = aus austauschbaren und miteinander kombinierbaren Komponenten bestehend)			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Anpassungen/Änderungswünsche können schnell umgesetzt werden.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Optional: Welche Anpassungsmöglichkeiten haben deiner Meinung nach gefehlt oder können/konnten nicht schnell genug umgesetzt werden?

-

Wie viele Punkte würdest du dem Feedback-Prozess geben? (1=schlecht, 10=ausgezeichnet)

1	2	3	4	5	6	7	8	9	**10**

Falls du mit dem Feedback-Prozess nicht zufrieden warst: Woran hat dies gelegen? (Eingeschränkte Möglichkeiten des Tools, fehlende Erfahrung der Nutzerin etc.)

-

NUR FÜR PERSONEN MIT ERFAHRUNG IN 3D-MODELLIERUNG:

Bitte gib deine Einschätzung über die Arbeit mit ProBuilder/Unity zu Erstellung eines 3D-Modells ab, indem du die folgenden Aussagen bewertest. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Mit ProBuilder bzw. Unity lässt sich (mit ausreichender Bedienungssicherheit/Routine) ein komplexes, modulares und modifizierbares 3D-Modell erstellen.				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Die Erstellung eines 3D-Modells mit ProBuilder bzw. Unity wirkt wie eine gute Alternative zu der herkömmlichen Methode (z.B. mit Blender).				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Der Feedback-Prozess mit Kund*innen kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden. (Bei der Präsentation einer VR-Umgebung in Unity, ohne den Wechsel zu Blender bei Änderungen am Modell)				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Kannst du dir vorstellen in Zukunft auf die vorgestellte Weise 3D-Modelle zu erstellen? Bitte begründe deine Antwort. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

-

Die letzten Fragen dürfen gerne wieder von allen beantwortet werden. :)

Hast du noch letzte Anmerkungen oder Gedanken zu dem vorgestellten Projekt?

-

Auswertung – Befragung

Person Nr.:	8
Alter:	30
Berufsbezeichnung:	VR Software Entwickler
Erfahrung mit 3D/VR-Tools:	Ja
Falls ja, mit welchen:	Unity, Blender, Unreal, Maja

Bitte bewerte die folgenden Aussagen:

Das Modell ist modifizierbar.			
Trifft zu	**Trifft teilweise zu**	Trifft eher nicht zu	Trifft nicht zu

Das Modell ist modular. (modular = aus austauschbaren und miteinander kombinierbaren Komponenten bestehend)			
Trifft zu	**Trifft teilweise zu**	Trifft eher nicht zu	Trifft nicht zu

Anpassungen/Änderungswünsche können schnell umgesetzt werden.			
Trifft zu	Trifft teilweise zu	**Trifft eher nicht zu**	Trifft nicht zu

Optional: Welche Anpassungsmöglichkeiten haben deiner Meinung nach gefehlt oder können/konnten nicht schnell genug umgesetzt werden?

Änderung der Blattformen, Änderung am Blumentopf, Erstellung unterschiedlicher Pflanzenformen (Wachstum)

Wie viele Punkte würdest du dem Feedback-Prozess geben? (1=schlecht, 10=ausgezeichnet)

1	2	3	4	5	6	**7**	8	9	10

Falls du mit dem Feedback-Prozess nicht zufrieden warst: Woran hat dies gelegen? (Eingeschränkte Möglichkeiten des Tools, fehlende Erfahrung der Nutzerin etc.)

Es hätten detailliertere Fragen gestellt werden können.

NUR FÜR PERSONEN MIT ERFAHRUNG IN 3D-MODELLIERUNG:

Bitte gib deine Einschätzung über die Arbeit mit ProBuilder/Unity zu Erstellung eines 3D-Modells ab, indem du die folgenden Aussagen bewertest. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Mit ProBuilder bzw. Unity lässt sich (mit ausreichender Bedienungssicherheit/Routine) ein komplexes, modulares und modifizierbares 3D-Modell erstellen.				
Trifft zu	**Trifft eher zu**	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Die Erstellung eines 3D-Modells mit ProBuilder bzw. Unity wirkt wie eine gute Alternative zu der herkömmlichen Methode (z.B. mit Blender).				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	**Trifft eher nicht zu**	Trifft nicht zu

Der Feedback-Prozess mit Kund*innen kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden. (Bei der Präsentation einer VR-Umgebung in Unity, ohne den Wechsel zu Blender bei Änderungen am Modell)				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Kannst du dir vorstellen in Zukunft auf die vorgestellte Weise 3D-Modelle zu erstellen? Bitte begründe deine Antwort. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Es kommt immer auf das Ziel/Zielgruppe an, wenn es um Prototypen geht ist diese vorgestellte weise durchaus Praxisnah. Wenn es jedoch um Vollendete Projekte/Produkte geht, werden die Vorteile von verschiedener Softwaren benötigt um eine hohe Qualität zu gewährleisten.

Die letzten Fragen dürfen gerne wieder von allen beantwortet werden. :)

Hast du noch letzte Anmerkungen oder Gedanken zu dem vorgestellten Projekt?

Eine Pflanze ist für diese Art des Projekte, in erster Betrachtung sinnvoll, jedoch bei genauerem Verständnis der Zielsetzung/Umfangs nicht der perfekte Gegenstand zum erproben des Workflows innerhalb einer Software. Eine mehr abstrakter Gegenstand hätte eine bessere Darstellung des Workflows aufzeigen können und die Vor- und Nachteile der Idee veranschaulichen können.

Auswertung – Befragung

Person Nr.:	9
Alter:	25
Berufsbezeichnung:	Student
Erfahrung mit 3D/VR-Tools:	Nein
Falls ja, mit welchen:	-

Bitte bewerte die folgenden Aussagen:

Das Modell ist modifizierbar.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Das Modell ist modular. (modular = aus austauschbaren und miteinander kombinierbaren Komponenten bestehend)			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Anpassungen/Änderungswünsche können schnell umgesetzt werden.			
Trifft zu	**Trifft teilweise zu**	Trifft eher nicht zu	Trifft nicht zu

Optional: Welche Anpassungsmöglichkeiten haben deiner Meinung nach gefehlt oder können/konnten nicht schnell genug umgesetzt werden?

Weitere potentielle Anpassungsmöglichkeiten die bisher noch nicht im Modell implementiert waren, wären die Anpassung von geometrischer Form und Oberfläche der Blätter.

Wie viele Punkte würdest du dem Feedback-Prozess geben? (1=schlecht, 10=ausgezeichnet)

1	2	3	4	5	6	7	8	**9**	10

Falls du mit dem Feedback-Prozess nicht zufrieden warst: Woran hat dies gelegen? (Eingeschränkte Möglichkeiten des Tools, fehlende Erfahrung der Nutzerin etc.)

Grundsätzlich war ich sehr zufrieden mit dem Feedback-Prozess. Auf diverse Nachfragen zum Tool und wie es sich potentiell von anderen unterscheidet, soweit das USP des Programms konnten beantwortet werden.

NUR FÜR PERSONEN MIT ERFAHRUNG IN 3D-MODELLIERUNG:

Bitte gib deine Einschätzung über die Arbeit mit ProBuilder/Unity zu Erstellung eines 3D-Modells ab, indem du die folgenden Aussagen bewertest. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Mit ProBuilder bzw. Unity lässt sich (mit ausreichender Bedienungssicherheit/Routine) ein komplexes, modulares und modifizierbares 3D-Modell erstellen.				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Die Erstellung eines 3D-Modells mit ProBuilder bzw. Unity wirkt wie eine gute Alternative zu der herkömmlichen Methode (z.B. mit Blender).				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Der Feedback-Prozess mit Kund*innen kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden. (Bei der Präsentation einer VR-Umgebung in Unity, ohne den Wechsel zu Blender bei Änderungen am Modell)				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Kannst du dir vorstellen in Zukunft auf die vorgestellte Weise 3D-Modelle zu erstellen? Bitte begründe deine Antwort. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

-

Die letzten Fragen dürfen gerne wieder von allen beantwortet werden. :)

Hast du noch letzte Anmerkungen oder Gedanken zu dem vorgestellten Projekt?

In meinen Augen recht zentral für den Nutzen ist die Abwägung zwischen Aufwand und Ertrag. Als Laie fällt es mir schwer zu beurteilen ob der Ertrag aus der uniformen / universellen Verwendung nur einer Software, die mögliche Kosten im Bereich Computing Power und ggf. eingeschränkter Flexibilität rechtfertigt. In meinen Augen ist die Erforschung dieser Frage entsprechend sinnvoll.

Auswertung – Befragung

Person Nr.:	10
Alter:	29
Berufsbezeichnung:	Marketingmanager
Erfahrung mit 3D/VR-Tools:	Nein
Falls ja, mit welchen:	-

Bitte bewerte die folgenden Aussagen:

Das Modell ist modifizierbar.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Das Modell ist modular. (modular = aus austauschbaren und miteinander kombinierbaren Komponenten bestehend)			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Anpassungen/Änderungswünsche können schnell umgesetzt werden.			
Trifft zu	Trifft teilweise zu	Trifft eher nicht zu	Trifft nicht zu

Optional: Welche Anpassungsmöglichkeiten haben deiner Meinung nach gefehlt oder können/konnten nicht schnell genug umgesetzt werden?

-

Wie viele Punkte würdest du dem Feedback-Prozess geben? (1=schlecht, 10=ausgezeichnet)

1	2	3	4	5	6	7	8	9	**10**

Falls du mit dem Feedback-Prozess nicht zufrieden warst: Woran hat dies gelegen? (Eingeschränkte Möglichkeiten des Tools, fehlende Erfahrung der Nutzerin etc.)

-

NUR FÜR PERSONEN MIT ERFAHRUNG IN 3D-MODELLIERUNG:

Bitte gib deine Einschätzung über die Arbeit mit ProBuilder/Unity zu Erstellung eines 3D-Modells ab, indem du die folgenden Aussagen bewertest. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

Mit ProBuilder bzw. Unity lässt sich (mit ausreichender Bedienungssicherheit/Routine) ein komplexes, modulares und modifizierbares 3D-Modell erstellen.				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Die Erstellung eines 3D-Modells mit ProBuilder bzw. Unity wirkt wie eine gute Alternative zu der herkömmlichen Methode (z.B. mit Blender).				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Der Feedback-Prozess mit Kund*innen kann, durch die Arbeit in nur einem Programm, effizienter gestaltet werden. (Bei der Präsentation einer VR-Umgebung in Unity, ohne den Wechsel zu Blender bei Änderungen am Modell)				
Trifft zu	Trifft eher zu	Neutral/kann ich nicht sagen	Trifft eher nicht zu	Trifft nicht zu

Kannst du dir vorstellen in Zukunft auf die vorgestellte Weise 3D-Modelle zu erstellen? Bitte begründe deine Antwort. (Diese Frage überspringen, wenn keine Erfahrung mit 3D/VR-Tools)

-

Die letzten Fragen dürfen gerne wieder von allen beantwortet werden. :)

Hast du noch letzte Anmerkungen oder Gedanken zu dem vorgestellten Projekt?

-

Anhang C: Inhalt des beigefügten USB-Sticks

Dateiname	**Datei-typ**	**Beschreibung**
Readme.docx	DOCX	Textdokument mit Erklärungen zu Dateien
MA_Chiara_K_Pflanze_UNITY	Ordner	Unity-Projekt
MA_Chiara_K_Screencast_Unity.mp4	MP4	Screencast der Anwendung
MA_Chiara_K_Bewertung_Praxisteil_Empirio.xlsx	XLSX	Ergebnisse der Befragung
MA_Chiara_K_Masterarbeit.pdf	PDF	Endversion der Masterarbeit

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit mit dem Titel:

Entwicklung eines modularen und modifizierbaren 3D-Modells in Unity zur Echtzeit-Kollaboration mit Kunden

Welche technischen und gestalterischen Möglichkeiten und Grenzen bestehen bei der Entwicklung eines modularen und modifizierbaren 3D-Modells in Unity und wie wirkt sich dies auf die Effizienz des Feedbackprozesses aus?

_____

selbständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

Datum	Unterschrift