Game-Engines

Jens Fehrenbacher, Philipp Künzel

Game-Engines sind Programme, in denen interaktive 2D- oder 3D-Anwendungen entwickelt werden. Das betrifft neben Games im engeren Sinne auch interaktive pädagogische, künstlerische oder forschende Software-Anwendungen sowie die meisten Formen von Augmented Reality, Virtual Reality oder *Digital Twinning* (vgl. Chia 2022). Damit haben Game-Engines einen großen Einfluss auf computergenerierte Bild- und Handlungsräume, also das, was gemeinhin unter Virtualität verstanden wird. Auch wenn dieser Band ein wesentlich weiteres Verständnis von Virtualität setzt, sind für eine Virtuelle Universität auch in diesem engeren Sinne des Begriffs Beschreibungs- und Umgangsweisen zu entwickeln, um zeitgenössische computationale Erzeugnisse wie Virtual Reality und Augmented Reality präzise in den Blick nehmen zu können. Game-Engines sind also genuine Medien der Welterzeugung, ohne die es keine Virtual Reality im engeren Sinn gäbe. Dabei fordert gerade die Vorstrukturierung und Standardisierung dieser Welterzeugung durch Game-Engines dazu auf, sie medienwissenschaftlich methodisch und analytisch ernst zu nehmen. In diesem Beitrag möchten wir unterschiedliche Weisen skizzieren, in denen die Game-Engines in der Virtuellen Universität eine Rolle spielen können.

1. Game-Engines als Untersuchungsgegenstand

Gerade weil sie für Endnutzer*innen nicht sichtbar sind und trotzdem einen großen Einfluss auf die Gestaltung von Softwareanwendungen aller Art haben, drängen sich Game-Engines als Untersuchungsgegenstand auf. Im Anschluss an Arbeiten der Software Studies (vgl. Manovich 2013; Soon/Cox 2020), Game Studies (vgl. Chia 2022; Nicoll/Keogh 2019) sowie Platform Studies (vgl. Bogost/Monfort 2009; Malazita 2024) lassen sich jeweils unterschiedliche Aspekte dieses Gegenstandes in den Blick nehmen. Dabei wird deutlich, dass Game-Engines keine neutralen Tools intentionalen Handelns sind, sondern sie sich je nach Perspektive als mediengeschichtlich situiert, in stetigen Wechselwirkungen mit komplexen Akteursnetzwerken oder als vorstrukturierte Interfaces virtueller Welterzeugung verstehen lassen.

Aus mediengeschichtlicher Perspektive lässt sich feststellen, dass es bis in die 1990er Jahre überwiegend üblich war, dass jedes Spiel eine eigene technologische Code-Grundlage hatte. In der Folge wurden viele Spiele allerdings zu komplex, um bei jedem neuen Programm erneut auf der grundlegendsten Ebene mit der Programmierarbeit zu beginnen. Das sich in diesen Jahren entwickelnde Konzept der Game-Engines trug somit dazu bei, Entwicklungsprozesse zu beschleunigen und spiel-spezifische Inhalte von grundlegenden, wiederholt einsetzbaren Funktionalitäten abzuspalten (vgl. Freedman 2020: 1). Die Game-Engines finden auch heute vorwiegend in spielerischen Kontexten Anwendung, haben sich aber über die Entwicklung von sogenannten *Serious Games* über Trainings-Anwendungen bis hin zu Simulationen für Militär und Medizin ausgebreitet (vgl. Berberich 2007; Malazita 2024).

Grundlegend stellen Game-Engines eine in sich geschlossene Software dar, die eine allgemeine Funktionsgrundlage für mehrere Anwendungen bildet (vgl. Freedman 2020: 1). Sie sind die Rahmenbedingung für die Erzeugung interaktiver digitaler Inhalte (vgl. Nicoll/Keogh 2019: 9). In Game-Engines lassen sich also heterogene Elemente wie 3D-Modelle, zweidimensionale Grafiken, Text, Audio- und Videomaterial oder User-Interface-Elemente arrangieren und mit Skripten kombinieren. Die Skripte bestehen aus Abfolgen von Befehlen und Anweisungen, die einer Wenn-Dann-Logik folgen und teils interaktiv auf die Nutzung reagieren. Diese Funktionalität unterscheidet Game-Engines von reiner Grafiksoftware, in der zwar ganze Filme produziert werden können, aber keine Interaktivität angelegt ist. In der Vielzahl möglicher Elemente deutet sich bereits die Heterogenität der involvierten, notwendigen Kompetenzen an: von 2D-, 3D-, Audio-, Animations- und Interface-Design bis hin zu Programmierkenntnissen, die von einem Team oder Einzelpersonen abzudecken sind. Diese Heterogenität steht der zuvor vorherrschenden Hierarchisierung des Software-Entwicklungsprozesses entgegen, wodurch die Rolle der Programmierer*innen dezentralisiert und die Entwicklung zur »site of ›deep remixability‹ for the different software techniques« (ebd.: 8, nach Manovich 2013: 267ff.) wird.

Dabei wird bereits deutlich, dass Game-Engines in dieser Welterzeugung nicht isoliert eingesetzt werden, sondern in komplexen Akteursnetzwerken verortet sind, insbesondere zwischen anderen Hard- und Software Plattformen sowie Nutzer*innen verschiedener Professionen, deren Arbeitsabläufe von den involvierten Plattformen strukturiert werden. Die Game-Engines erlauben es, die »computational tasks that are central to many digital media production pipelines« (Freedman 2020: 2) von den gestalterischen Inhalten der Anwendungen abzuspalten. Die stark informatik- und mathematiklastigen Teile des Entwicklungsprozesses werden demnach abstrahiert und die Entwicklung von Anwendungen den Designer*innen selbst zugänglich gemacht, die somit selbst zu Editor*innen werden (vgl. Binay 2011: 83). Um Schwellen ab- und die Zahl der Nutzenden auszubauen, setzen die Unternehmen hinter den Game-Engines auf umfangreiche Rahmenangebote: kostenfreie Tutorials, herunterladbare Beispielprogramme sowie Asset-Stores, in denen vorgefertigte Elemente wie 3D-Modelle oder Skripte kostenfrei oder kostenpflichtig bezogen werden können. Dieses Angebot stellt eine umfangreiche Hilfestellung dar, präfiguriert aber die Entwicklung neuer Software (Fehrenbacher im Erscheinen). Auch abseits dieses rahmenden Ökosystems lassen sich in der Game-Engine

selbst multiple Voreinstellungen nachvollziehen, die den Entwicklungsprozess auf unauffällige Weise prägen.

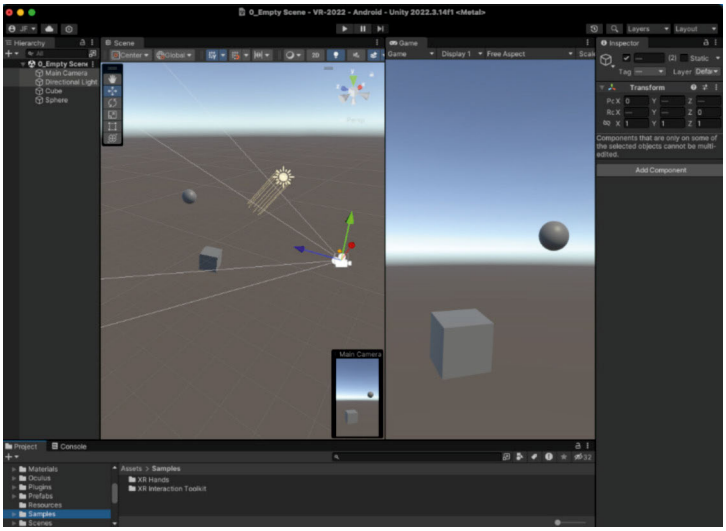

Abb. 1: Übersicht über eine 3D-Szene in der Game-Engine Unity. Linkes großes Fenster: Szenenansicht zur Bearbeitung. Rechtes großes Fenster: Game-Ansicht aus Perspektive der virtuellen Kamera. Bis auf den hinzugefügten Würfel und die Kugel wurde das 3D-Template nicht verändert. Die voreingestellten Kamera- und Lichtobjekte wurden angewählt, sodass deren Ausrichtung erkennbar ist. Eigener Screenshot

Bereits der Blick auf ein leeres 3D-Projekt der Game-Engine Unity offenbart zahlreiche solcher Setzungen[1]: Ein Horizont, der blauen Himmel von graubrauner Erde trennt sowie ein Gitter, das als Bodenniveau zur Orientierung dient, sind voreingestellt. In der Szene sind standardmäßig eine virtuelle Kamera, die den Blick- und Aktions-Punkt der Nutzer*innen darstellt, und eine Lichtquelle vorhanden, auf die jedes 3D-Modell reagiert. Die Kamera, welche die Position und Blickrichtung der Nutzenden innerhalb der Endanwendung markiert, imitiert in ihrer Funktionsweise die physikalischen Eigenschaften analoger Kameras, wie etwa eine Zentralperspektive, obwohl auch eine Projektion selbst unmöglicher Bildlichkeiten denkbar wäre (vgl. Schröter 2003: 8,14). Das voreingestellte Licht scheint als globales Sonnenlicht (mit Sonnensymbol repräsentiert) diagonal von oben und leicht seitlich rotiert in Blickrichtung der Kamera. So sind eingefügte Objekte wie die Formen in Abb. 1 so beleuchtet, dass sie möglichst hell erscheinen, aber auch ihre Räumlichkeit durch Schattierung zur Geltung kommt. Gleichermaßen lässt sich von diesen Lichtstimmungen eine Linie zu spezifischen

1 Zum *empty file* als methodischem Ansatzpunkt, welches bei näherer Betrachtung alles andere als leer ist, vgl. Ojala 2024.

Gestaltungsprinzipien aus der euro-amerikanischen Malerei und ihren ästhetischen Vorentscheidungen ziehen (vgl. Scheler 2023). Doch Game-Engines sind grundsätzlich nicht bloße visuelle Medien, sondern operationale und computationale Medien (vgl. Livingston 2024: 182). Aufgabe der Game-Engine ist es, »[to] mediate between data and embodiment« (Freedman 2020: 170).

In dieser Operationalisierung und Verkörperung spielen wiederum die sogenannte Physik-Engines der Game-Engines eine Rolle, durch die physikalische Verhaltensweisen virtueller Objekte simuliert werden: Wird einem eingefügten Objekt die Eigenschaft *Festkörper* zugewiesen, fällt es beim Drücken des *Play*-Button, der das Programm ausführt, die irdische Schwerkraft affirmierend in Richtung eines willkürlich definierten Untens. Zahlreiche gestalterische Annahmen und physikalische Prozesse sind also in das Programm eingeschrieben, die so selbstverständlich und standardisiert sind, dass sie kaum auffallen. Selbst wenn sich viele dieser und weiterer Setzungen zumindest bis zu einem gewissen Grad modifizieren oder rückgängig machen lassen, prägen sie doch den Produktionsprozess und werden häufig unhinterfragt übernommen (Nicoll/Keogh 2019: 63f.).

Um einen derartigen Einblick in die Funktionsweisen dieser operativen Medien zu erhalten, reicht es jedoch nicht aus, sie lediglich distanziert zu betrachten, sondern es ist nötig, sie ebenfalls praktisch zu nutzen.

2. Game-Engines als Werkzeuge der Critical Technical Practice

Dementsprechend möchten wir vorschlagen, als Teil einer Virtuellen Universität nicht nur über, sondern auch mit Game-Engines zu forschen und zu lehren. Im praktischen Experiment mit diesen Softwareumgebungen lassen sich zum einen die vorstrukturierten Modi der Welterzeugung sowohl erkunden wie auch spekulativ über den Bereich des Gewohnten ausdehnen (vgl. Soon/Cox 2020). Zum anderen eröffnen Game-Engines die Möglichkeit, virtuelle Handlungsräume zu erzeugen, um mit exakt diesen Möglichkeiten der Welterzeugung in angrenzenden Themenbereichen, wie der Geschichtsvermittlung, der Kunstgeschichte oder den Science and Technology Studies zu experimentieren.

Die oben angedeutete Vorstrukturierung der Welterzeugung hinsichtlich der voreingestellten virtuellen Kamera und der Reaktion von Objekten auf Lichtquellen lässt sich so im Sinne der von Phil Agre beschriebenen »Critical Technical Practice« (vgl. Agre 1997) im praktischen Tun dekonstruieren, um Vorannahmen und Einseitigkeiten zu identifizieren (vgl. Bolinski und andere in diesem Band). Ein einstiegsfreundlicher Ansatz besteht etwa darin, die *Defaults* der Programme zu verstehen, indem voreingestellte Werte verändert werden, ohne zwingend im Vorhinein verstehen zu müssen, was der jeweilige Wert bewirkt. Was Lev Manovich bereits im Bezug auf Bildbearbeitung gezeigt hat, zählt in dreidimensionalen Handlungsräumen umso mehr: »What begins as a reference to a physical world outside of the computer if we use default settings can turn into something totally alien with a change in the value of a single parameter« (Manovich 2011: 20).

Beim Versuch, die Normalperspektive, den Horizont oder die Reaktion von Objekten auf Lichtquellen zu deaktivieren (Abb. 2), wird so deutlich, dass gerade letzteres computational ein durchaus aufwendiger Prozess ist, in der die relativen Positionen von

Flächen im Verhältnis zu Lichtquelle und Kamera in unterschiedliche Schattierungen umgerechnet werden. Wird diese Selbstverständlichkeit jedoch ausgeschaltet, präsentieren sich einfarbige Objekte nicht schattiert, sondern monochrom. Dadurch wird deutlich, inwiefern diese Lichtsimulation hilft, Räumlichkeit hervorzubringen. Auch die standardmäßige Festlegung auf eine Zentralperspektive offenbart ihre Folgenschwere gerade dann, wenn sie deaktiviert beziehungsweise durch eine orthogonale Perspektive ersetzt wird. Ohne die Setzung von Fluchtpunkten werden Objekte hier unabhängig von ihrer Distanz zur Kamera in gleicher Größe dargestellt. Die erzeugte Welt ist weiterhin dreidimensional, aber doch gänzlich abweichend von konventionalisierten Darstellungsweisen von Räumlichkeit. In diesem durchaus spielerischen Austesten wird offensichtlich: Game-Engines präsentieren sich als *tabula rasa* und verschleiern den Blick darauf, wie intensiv vorstrukturiert diese *tabula* ist.

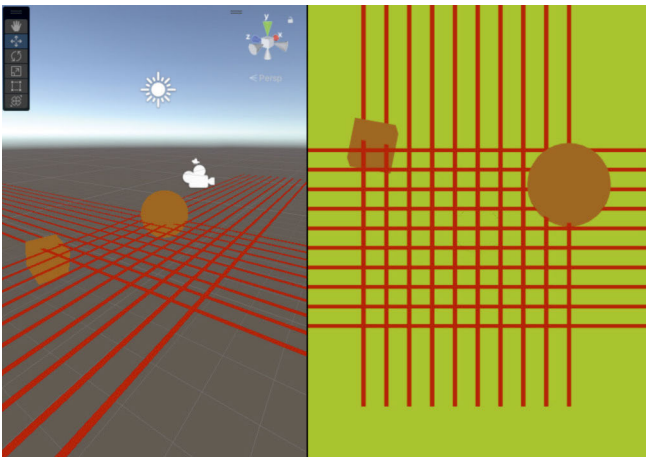

Abb. 2: Experiment mit der Revidierung von Voreinstellungen in der Game-Engine Unity. Links: Szenen-Ansicht, in der Objekte bearbeitet oder verschoben werden können. Rechts: Game-Ansicht, die das Ergebnis im Spiel anzeigt. Die sogenannten Shader-Einstellungen von Würfel und Kugel wurden so angepasst, dass sie nicht auf Licht reagieren und somit keine Schattierungen aufweisen. In den Kamera-Einstellungen wurde eine orthogonale Perspektive statt einer Zentralperspektive angewählt sowie der default Horizont mit einer monochronen Fläche ersetzt, was allerdings nur in der Game-Ansicht wirksam wird. Eigener Screenshot

Neben solchen, ganz auf die Game-Engine selbst ausgerichteten Umgangsformen eröffnen sich für eine Virtuelle Universität noch weit heterogenere Möglichkeitsräume, wenn Game-Engines als Werkzeug der Welterzeugung in angrenzenden geisteswissenschaftlichen Themenfeldern Anwendung finden. Da dieses Potenzial in dem Beitrag von Bolinski und anderen in diesem Band detaillierter ausgeführt wird, sei an dieser Stelle

nur angedeutet, wie im Rahmen des *SFB Virtuelle Lebenswelten* und des *Virtual Humanities Lab* bereits damit experimentiert wurde, wie eigene Räume für soziale VR Anwendungen konstruiert, Schlachtengemälde geschichtsdidaktisch in VR aufgearbeitet, neuzeitliche Kupferstiche immersiv dekonstruiert, oder ethnografische Feldforschung in Augmented Reality umgewandelt werden können. Jenseits des rein gestalterischen Umgangs lässt sich anhand der operativen bzw. interaktiven Funktionalität der Game-Engines betonen, dass es sich hier nicht allein um visuelles Spekulieren handelt, sondern um ein Experiment mit körperlich-affektiven Verhältnissen zu den genannten Gegenständen und ihren virtuellen Übersetzungsformen.

3. Game-Engines als Orte hybrider Versammlung

Gerade in dieser praktischen Arbeit deutet sich bereits an, dass Game-Engines auch die Funktion hybrider Versammlungsorte annehmen können, in denen gemeinsam gearbeitet, erkundet und kommuniziert werden kann.

Im Rahmen des *SFB Virtuell Lebenswelten* wurde dementsprechend mit Formaten experimentiert, um die virtuellen Bild- und Handlungsräume von Game-Engines als Teil von Lehre und Forschung einzubinden. Im Xtended Room, einem Büroraum, der bis auf einen Tisch mit Computer und VR-Setup leer ist, wurden so nicht nur VR-Anwendungen zur Rezeption bereitgestellt, sondern auch Wissenschaftler*innen die Gelegenheit gegeben, die praktische Arbeit mit computergenerierter Virtualität in ihre Forschung einfließen zu lassen (Fehrenbacher/van der Veen 2024). So konnten 3D-Scans des eigenen Körpers in VR untersucht oder auch prototypische Entwicklungen von Kooperationspartner*innen erkundet und modifiziert werden.

Ebenso wurde mit dem *Virtual Artist Lab* ein Veranstaltungsformat ins Leben gerufen, in dem Künstler*innengespräche durch konkrete Einblicke in die Produktionsbedingungen der virtuellen Werke bereichert wurden. Die Künstlerin Sarah Rothberg legte dabei etwa offen, wie sie konkret in ihrer künstlerischen Praxis auf die Affordanzen der Game-Engine eingeht und gerade aus der Zweckentfremdung von Funktionen, die eigentlich der Vereinfachung dienen, eigenwillige Glitch-Poetiken entstehen können.

In Formaten wie diesen, die eher die Vorstrukturiertheit der Game-Engines in ihrer Dekonstruktion thematisieren, zeigt sich, dass der konventionalisierte, selektive Realismus, den die Programme als Default anbieten und der in vielen Fällen durch hintergründige Setzungen in den Game-Engines und komplexen computationalen Prozessen aufrechterhalten wird, alles andere als alternativlos ist. Während sich durch die Voreinstellungen, Hilfestellungen und umliegende Angebote in und rund um Game-Engines die Tendenz der Sedimentierung bestimmter Vorstellungen von virtuellen Welten andeutet, betonen solche spekulative Vorgehensweisen sowie Anwendungen die Präfiguration, die Kontingenz, aber auch die vielgestaltige Potenzialität computergenerierter virtueller Welten.

Literatur

Agre, Phil E. (1997): »Toward a critical technical practice: lessons learned in trying to reform AI«, in: Geoffrey Bowker et al. (Hg.): Social Science, Technical Systems, and Cooperative Work: Beyond the Great Divide, Mahwah, NJ: Lawrence Erlbaum Associates, S. 131–157.

Berberich, Steve (2007): »Video Games Starting to Get Serious«, in: gazette.net (08.2007). Online unter: https://web.archive.org/web/20081203174009/www.gazette.net/stories/083107/businew11739_32356.shtml (letzter Zugriff: 17.03.25).

Binay, Berk (2011): »Game development tools«, in: Navigationen – Zeitschrift für Medien- und Kulturwissenschaften 11 (2), S. 81–84.

Bogost, Ian/Montfort, Nick (2009): »Platform Studies: Frequently Questioned Answers«, in: UC Irvine (Hg.), Proceedings of the Digital Arts and Culture Conference 2009. Online unter: https://escholarship.org/uc/item/01r0k9br#main (letzter Zugriff: 27.03.2025).

Chia, Aleena (2022): »The metaverse, but not the way you think: game engines and automation beyond game development«, in: Critical Studies in Media Communication 39 (3), S. 191–200. https://doi.org/10.1080/15295036.2022.2080850.

Fehrenbacher, Jens (im Erscheinen): »Default Engines. Normative Dynamiken in der Diffusion von 4D- Entwicklungsplattformen in Forschung, Kunst und Digital Twinning«, in: Kathrin Friedrich et al. (Hg.): 4D, München: Open Publishing LMU.

Fehrenbacher, Jens/van der Veen, Manuel (2024): »Xtended Room«, in: Early Career Forum des SFB 1567 (Hg.), Vokabular des Virtuellen. Ein situiertes Lexikon, Bielefeld: transcript, S. 192–198. https://doi.org/DOI: %2010.14361/9783839472071-056.

Freedman, Eric (2020): The persistence of Code in Game Engine Culture, New York: Routledge. https://doi.org/10.4324/9780429434242.

Livingston, Tom (2024): »Game Engines: Optimising VFX, Reshaping Visual Media«, in: NECSUS – European Journal of Media Studies 13 (1), S. 180–201. http://dx.doi.org/10.25969/mediarep/22809.

Malazita, James (2024): Enacting Platforms. Feminist Technoscience and the Unreal Engine, Cambridge/London: MIT Press. https://doi.org/10.7551/mitpress/14978.001.0001.

Manovich, Lev (2011): »Inside Photoshop«, in: Computational Culture 1 (11.2011). Online unter: http://computationalculture.net/inside-photoshop (letzter Zugriff: 27.03.2025).

Manovich, Lev (2013): Software Takes Command, New York: Bloomsbury. https://doi.org/10.5040/9781472544988.

Nicoll, Benjamin/Keogh, Brendan (2019): The Unity Game Engine and the Circuits of Cultural Software, Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-25012-6.

Ojala, Mace (2024): »File, empty«, in: Early Career Forum des SFB 1567 (Hg.), Vokabular des Virtuellen: Ein situiertes Lexikon, Bielefeld: transcript, S. 79–82. https://doi.org/10.14361/9783839472071-021.

Scheler, Carolin (2023): Computergrafik – Zur Geschichte und Produktionsästhetik synthetischer Bilder, Bielefeld: transcript.

Schröter, Jens (2003): »Virtuelle Kamera. Zum Fortbestand fotografischer Medien in computergenerierten Bildern«, in: Fotogeschichte 88, S. 3–16.
Soon, Winnie/Cox, Geoff (2020): Aesthetic Programming: A Handbook of Software Studies, London: Open Humanities Press.

Abbildungsverzeichnis