Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Eike Adler

Eine Software zur 3D-Simulation von Produktionssystemen – basierend auf einer Gameengine und KI

Fakultät Technik und Informatik
Department Maschinenbau und Produktion

Faculty of Engineering and Computer Science
Department of Mechanical Engineering and Production Management

Eike Adler

Eine Software zur 3D-Simulation von Produktionssystemen – basierend auf einer Gameengine und KI

Bachelorarbeit eingereicht im Rahmen Bachelorprüfung

im Studiengang Produktionstechnik und -management
am Department Maschinenbau und Produktion
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Markus Stallkamp
Zweitgutachter : Dipl.-Ing. Frank Peters

Abgegeben am 8.11.2013

Zusammenfassung

Eike Adler

Thema der Bachelorarbeit

Eine Software zur 3D-Simulation von Produktionssystemen – basierend auf einer Gameengine und KI

Stichworte

3D-Simulation, Logistik, Gameengine, Produktionssysteme, Künstliche Intelligenz, Digitale Fabrik, Virtuelle Realität

Kurzzusammenfassung

Die Bachelor-Thesis beschreibt den Entwicklungsprozess einer 3D Simulationssoftware, die ein vordefiniertes veränderbares Produktionssystem abbildet und simuliert.
Die Umsetzung erfolgt auf Basis einer Gameengine und KI. Die Anwendung einer modernen Gameengine, erlaubt eine Simulation in hoher grafischer Qualität und realisiert gleichzeitig künstliche Intelligenz zum Lösen von Wegfindungsproblemen. Dabei ist der Programmieraufwand gering, wodurch die Entwicklungskosten niedrig bleiben.

Eike Adler

Bachelor Thesis title

A software for 3D simulation of manufacturing systems - based on a game engine and AI

Keywords

3D simulation, logistics, game engine, manufacturing systems, Artificial Intelligence, Digital Factory, Virtual Reality

Abstract

The bachelor thesis describes the development process of a 3D simulation software that maps and simulate a predefined variable production system.
The implementation is based on a game engine and AI. The usage of a modern game engine allow a simulation with high graphic quality and realize an artificial intelligence for pathfinding solutions in the process. The programming effort is low, thus the development costs remain low.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

GUI Graphical User Interface

KI künstliche Intelligenz

PPS Produktionsplanung und -steuerung

SDK Software Development Kit

UML Unified Modeling Language

VR virtuelle Realität

WA Wareneingang

WE Warenausgang

1 Einleitung

Nach Kunicki [1] steigt die Nachfrage nach flexibleren und reaktionsschnelleren Produktionssystemen aufgrund einer sich stetig verändernden Unternehmensumwelt. Einflussfaktoren wie z.B. die Globalisierung führen zu einem erhöhten Konkurrenzdruck, steigender Produktkomplexität, einer steigenden Anzahl an Produktvarianten und einer Verlagerung von Wertschöpfungsaktivitäten zu Zulieferbetrieben. Eine Lösung für solch ein flexibles Produktionssystem, stell die Digitale Fabrik dar, die die VDI-Richtline 4499 [2] folgendermaßen definiert:

> "Die Digitale Fabrik ist der Oberbegriff für ein umfassendes Netzwerk von digitalen Modellen, Methoden und Werkzeugen – u. a. der Simulation und dreidimensionalen Visualisierung, die durch ein durchgängiges Datenmanagement integriert werden. Ihr Ziel ist die ganzheitliche Planung, Evaluierung und laufende Verbesserung aller wesentlichen Strukturen, Prozesse und Ressourcen der realen Fabrik in Verbindung mit dem Produkt."

Ein Simulationseinsatz eignet sich nach Eley [3] besonders, um vielfältige komplexe Planungsprobleme im Logistikbereich zu lösen. Die VDA-Empfehlung 4810 [4] unterteilt die Simulation in der Logistik in unterschiedliche Simulationsausprägungen. Hierbei wird unterschieden zwischen Werkssimulation, Belieferungssimulation, Supply-Chain-Simulation und Verkehrsflusssimulation.
Diese Arbeit erweitert eine Simulation mit einer dreidimensionalen Visualisierung. Dadurch ergeben sich Nutzeneffekte, wie z.B. die Verwendung als visuelles Kommunikations- und Präsentationsmittel. Auch in der Lehre kann ein solches System eine wichtige Rolle einnehmen um Produktionsstrukturen und verschiedene logistische Prozesse besser zu verdeutlichen. Die Verwendung geeigneter 3D-Hardware, zur Erzeugung einer virtuellen Realität, verstärkt den Präsentationseffekt.
Die Anwendung einer modernen Gameengine erlaubt eine Simulation in hoher grafischer

Qualität und realisiert gleichzeitig künstliche Intelligenz (KI) zum Lösen von Wegfindungsproblemen. Dabei ist der Programmieraufwand gering, wodurch die Entwicklungskosten niedrig bleiben.
Diese Arbeit beschreibt den Entwicklungsprozess einer 3D-Simulationssoftware basierend auf einer Gameengine, die eine Verkehrsflusssimulation innerhalb eines Produktionssystems abbildet. Kapitel 2 gibt einen Überblick über die Grundlagen, die zum Verständnis dieser Arbeit wichtig sind. Kapitel 3 betrachtet den Entwurfsprozess der 3D-Simulationssoftware. Diese beginnt zunächst bei einer klaren Definition der Ziele und Einsatzzwecke. Die Beschreibung der daraus resultierenden Systemdefinition und Anforderungen folgen im Anschluss. Ein wichtiger Teilschritt in der Softwareentwicklung ist die Softwarearchitektur, die durch standardisierte UML Diagramme dargestellt wird. Das letzte Unterkapitel definiert Zustandsgrößen und verhaltensbestimmende Größen, die das Simulationssystem messbar und veränderbar machen. Kapitel 4 stellt zunächst die Rahmenbedingungen und verwendeten Technologien vor. Der Hauptteil beschreibt die Implementierung der 3D-Simulationssoftware mit einer Gameengine. Hierzu erfolgt eine Systemaufteilung in einen Editormodus und einen Simulationsmodus. Der letzte Abschnitt stellt ein Anwendungsbeispiel vor. Kapitel 5 betrachtet rückblickend kritisch die Entwurfsentscheidungen der Arbeit. Dabei geht es auf Verbesserungsmöglichkeiten bzw. Erweiterungsmöglichkeiten der Software und offene Fragestellungen ein.

2 Grundlagen

Dieses Kapitel beschreibt die Grundlagen, die zum Verständnis dieser Arbeit notwendig sind. Abschnitt 2.1 definiert eine Gameengine und stellt verschiedene Untersysteme vor. Eine Gameengine ist eine Entwicklungsumgebung zur Programmierung von Computersystemen. Abschnitt 2.2 gibt eine Übersicht über verschiedene 3D-Hardwarekomponenten. Mit diesen lässt sich eine virtuelle Realität erzeugen. Das verwendete technische Verfahren wird erläutert. Abschnitt 2.3 gibt eine Einführung in die KI-Wegfindung. Dabei handelt es sich um ein Verfahren, das den Weg zwischen einem Startpunkt und einem Zielpunkt in einem definierten Suchraum findet. Häufig handelt es sich hierbei um den kürzesten Weg. Abschnitt 2.4 beschreibt ein Produktionssystem und stellt die Untersysteme, sowie Prinzipien und Methoden zur Organisation und Steuerung von Produktionssystemen vor. Schwerpunktmäßig wird auf die Produktionsplanung und -steuerung eingegangen. Abschnitt 2.5 erklärt die 3D-Simulation und stellt den Zusammenhang zur Digitalen Fabrik her. Hierbei spielt besonders der 3D-Aspekt eine wichtige Rolle.

2.1 Gameengine

Die Definition einer Gameengine ist nach Anderson und Engel [5] weit gefächert. Nach Sherrod [6] ist eine Gameengine eine Entwicklungsumgebung, die verschiedene Tools und Schnittstellen, deren Kombination ein Videospiel ausmachen, zur Verfügung stellt, aber die Hintergrundabläufe der Spielfunktionen ausblendet. Anderson und Engel bemängeln, dass keine klare Abgrenzung zwischen einer Gameengine und dem Videospiel erkennbar ist. Eine klarere Abgrenzung erfolgt in der Definition von Lewis und Jacobsen [7], die eine Gameengine als eine Sammlung von Modulen, die Simulationscode beinhalten, ansehen, die nicht direkt das Spielverhalten oder die Spieleumgebung bestimmen. Einigkeit herrscht darüber, dass eine Gameengine verschiedenen Tools, Module und Schnittstellen

zur Verfügung stellt. Nach Anderson und Engel, sowie Gregory [8], ist eine Gameengine auf ein bestimmtes Spielgenre, z.B. Strategiespiele ausgelegt und enthält neben den Standardmodulen zusätzlich spezialisierte genreabhängige Module. Tabelle 2.1 zeigt eine Übersicht von häufig genutzten, genreunabhängigen Standardmodulen, nach Thorn [9].

Grafikengine	Berechnet im Hintergrund der Anwendung die Computergrafik und bietet dem Programmierer verschiedene Funktionen und Effekte, wie z.B. Oberflächentexturierung, Spiegelungen etc.
Szenenmanager	Organisiert die Objekte, die in der Szene enthalten sind. Der Szenenmanager ermöglicht u.a. das Platzieren, Rotieren und Skalieren von Objekten in der Szene
Physiksystem	Berechnet das Verhalten von Objekten untereinander bzw. mit der Umgebung z.B. bei einer Kollision
Soundsystem	Erzeugt die Soundkulisse der Anwendung. Das Soundsystem bietet die Möglichkeit Soundeffekte und 5.1 / 7.1-Raumklang zu integrieren.
Scripting	Das Scripting dient dem Programmieren von Spielabläufen.
Datenverwaltung	Verwaltet Ressourcen, wie z.B. Texturen oder 3D-Objekte bzw. verfügt über Importfunktionen
Steuerung	Unterstützung von Joysticks, Gamepads, Bewegungssensoren zur Benutzereingabe
Netzwerk-Code	Integration von Multiplayermöglichkeiten über das Netzwerk bzw. Internet

Tabelle 2.1: Häufig genutzte Module einer Gameengine

Nach Meisinger [10] definiert die Architektur einer Gameengine, dass Zusammenwirken und die Kommunikation der Module untereinander. Häufig wird ein objektorientierter Ansatz genutzt. Herbei erbt ein Gameobjekt Eigenschaften von einer Superklasse, um dem Objekt weitere Funktionalitäten hinzuzufügen. Eine Gameobjekt besteht somit aus beliebig vielen Komponenten die verschiedene Funktionen implementieren. So ist z.B. eine Mesh-Renderer Komponente notwendig um das 3D-Objekt in der Szene darzustellen. Die Erzeugung von den benötigten 3D-Objekten erfolgt i.d.R. in 3D-Computergrafik- und Animationsprogrammen. Über eine entsprechende Schnittstelle z.B. .fbx-Format lassen sich die 3D-Objekte importieren.

Die Entwicklungsumgebung bzw. der Editor besteht aus mehreren Fenstern. Im Hauptfenster kann der Anwender die Objekte der 3d-Szene manipulieren. In weiteren Nebenfenstern lassen sich z.B. Eigenschaften von ausgewählten Komponenten verändern oder neue Komponenten hinzufügen. Das Programm bietet die Möglichkeit die Anordnung der Fenster zu ändern bzw. Fenster ein- und auszublenden.

2.2 3D-Hardware

3D-Hardware ermöglicht in Kombination mit entsprechender Software die Erzeugung einer virtuellen Realität. Brill [11] beschreibt die virtuelle Realität (VR) als eine computergestützte simulierte Realität, in der sich der Benutzer bewegen und mit Objekten interagieren kann. Ziel ist es dem Anwender das Gefühl zu vermitteln, sich in einer tatsächlichen Umgebung zu befinden.
Die visuelle Wahrnehmung der virtuellen Umgebung wird durch entsprechende Ausgabegeräte erreicht. Tabelle 2.2 zeigt zwei häufig genutzte Ausgabegeräte, die im Bereich der PC-Anwendungen eingesetzt werden.

Hardware	Erläuterung - Vor- / Nachteile	Preis
Occulus Rift	Die Occulus Rift ist eine VR-Brille. Sie zeichnet sich durch ein besonders weites Sichtfeld von 110 Grad, sowie einen schnellen Bewegungsensor aus. Nach Heise [12] füllt sie das Sichtfeld des Menschen nahezu komplett aus und erzeugt somit einen sehr beeindruckenden und überzeugenden räumlichen Eindruck. Bemängelt wird hingegen die geringe Auflösung, da eine Teilung des Displays, das sich im inneren der Brille befindet und eine Auflösung von 1280 x 800 Pixeln hat, erfolgt. Jedes Auge bekommt somit eine Hälfte des Displays zu sehen mit einer Auflösung von nur noch 640 x 800 Pixeln. Golem [13] kritisiert desweiteren, dass bei vielen Nutzern schon nach kurzer Zeit Übelkeit auftritt. Der Hersteller hat bereits reagiert und Tipps an Softwareentwickler und Kunden, um Übelkeit bei dem Tragen zu vermeiden, veröffentlicht. *(Quelle der Technischen Daten: oculus-rift.de [14])*	300 USD
nVidia 3D Vision 2	Das System basiert auf dem Shutterverfahren. Hierzu ist eine Shutterbrille, sowie ein von nVidia zertifiziertes Anzeigegerät, z.B. Beamer, mit mindestens 120Hz nötigt. Die Shutterbrille hat Gläser aus Flüssigkristallflächen, die elektronisch abgedunkelt werden können. Um einen stereoskopischen Effekt zu erzielen erfolgt nun eine abwechselnde Abdunklung eines Auges und synchron hierzu wird das jeweilige Bild für das rechte bzw. linke Auge auf dem Anzeigegerät dargestellt. Ein Nachteil dieser Technik ist, nach Gamestar [15], dass hohe Ansprüche an die Grafikkarte gestellt werden. Sie muss bei diesem Verfahren das Doppelte leisten im Vergleich zum normalen 2D-Betrieb. Ein weiterer Kritikpunkt ist, dass Ermüdungserscheinungen der Augen, aufgrund von vielen kleinen Unregelmäßigkeiten, wie z.B. Unschärfen, auftreten. *(Quelle der Technischen Daten: nvidia.de [16])*	140 Euro *(zusätzlich noch entsprechendes Anzeigegerät und Grafikkarte benötigt)*

Tabelle 2.2: Ausgabegeräte

Nach Gautzsch [17], sieht der Mensch stereoskopisch bzw. nimmt seine Umgebung mit einem räumlichen Eindruck von Tiefe wahr, da seine Augen horizontal leicht versetzt sind.

Er nimmt somit zwei leicht voneinander versetzt Bilder auf, die in dem Gehirn zu einem Bild mit Tiefeninformationen verarbeitet werden. Um den Effekt des räumlichen Sehens nachzuempfinden, nutzen daher die Ausgabegeräte ein Verfahren, bei dem jedem Auge ein zweidimensionales Bild aus einer leicht abweichenden Perspektive dargestellt wird. (s. Abbildung 2.1)

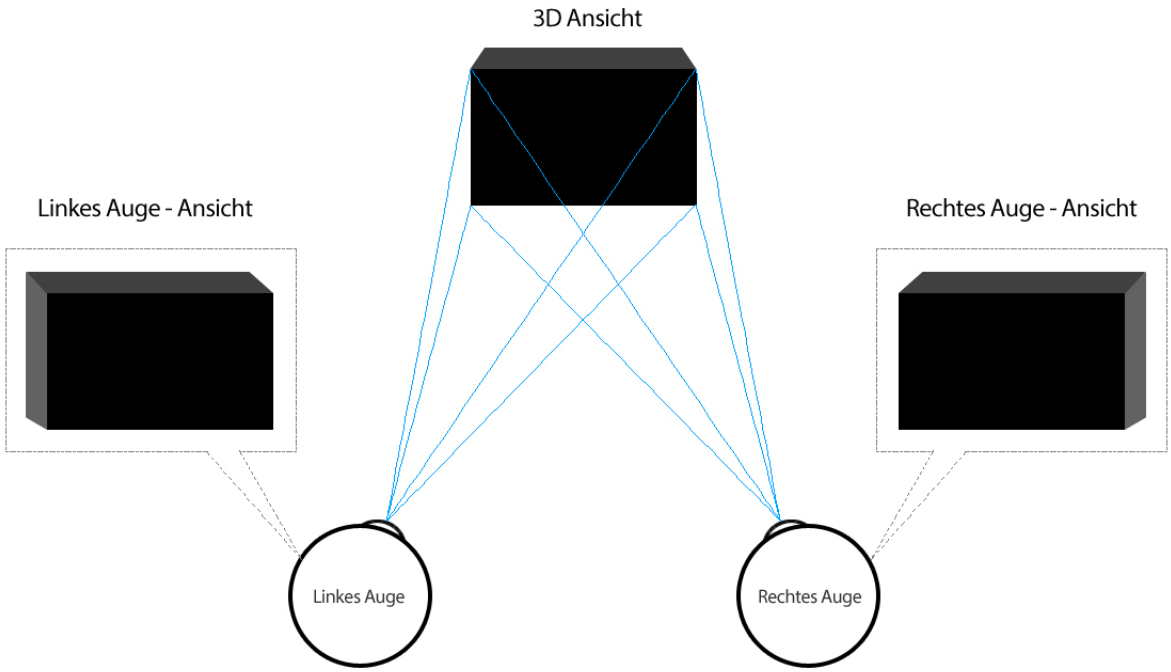


Abbildung 2.1: Stereoskopie

Das Interagieren mit der virtuellen Realität erfolgt, nach Izadi und Kim [18], mit 3D-Eingabegeräten. Popularität erreichte Microsoft mit der Kinect Tiefensensor-Kamera, bei der durch real-weltliche Gesten, z.B. das Greifen nach einem Objekt, mit der Szene interagiert wird. Hierbei ist es wichtig, dass die Eingabe und die entsprechende Reaktion des Systems in Echtzeit ablaufen. Um eine verzögerungsfreie visuelle Reaktion des Systems zu gewähren muss die Bildwiederholrate des Ausgabegerätes, nach Brill [11], mindestens 15-20Hz erreichen.

2.3 KI - Wegfindung

McCarthy [19] definiert die KI als eine Wissenschaft, die sich mit der Entwicklung von intelligenten Computerprogrammen beschäftigt. Zöller-Greer [20] teilt die KI zusätzlich in zwei zweckbezogene Bereiche ein:

KI als Kognitionswissenschaft Ziel ist die natürliche Intelligenz zu verstehen und nachzubauen. Der Computer dient hierbei als wichtigstes Werkzeug.

KI als Informatikdisziplin Ziel ist es Software mit Problemlösungsfähigkeit zu entwickeln. Hierbei spielt es keine Rolle, ob auf Mechanismen der natürlichen Intelligenz zurückgegriffen wird.

Kastenholz [21] sieht die Wegfindung als ein Teilgebiet der künstlichen Intelligenz (KI) an. Er erläutert, dass das Verfahren sich mit dem Finden eines Weges zwischen einem Ziel- und Startpunkt innerhalb eines vorgegebenen Suchraums befasst. Häufig wird dieses Verfahren im Bereich der Spiele-Entwicklung eingesetzt. Aber auch andere Bereiche, wie z.B. Navigations-Systeme, nutzen zunehmend die Wegfindung. Der Berechnungsaufwand ist in erster Linie abhängig von der Größe des Suchraums und der Dimension. Durch den Generationswechsel der vergangenen Jahre von 2D zum 3D-Standard liegt in der Regel ein 3D-Suchraum vor. Um diesen hohen rechnerischen Aufwand zu lösen, beschäftigt sich die 3D-Wegfindung mit der Frage nach Optimierungsmöglichkeiten. Als besonders effizient hat sich nach Kastenholz hierbei der zweiphasige Prozess herausgestellt, der aus dem Preprocessing und dem Realtime Processing besteht.
Das Preprocessing geschieht vor dem Ablauf des Spiels bzw. der Anwendung. Häufig wird hierbei der Suchraum in ein Raster unterteilt (s. Abbildung 2.2) und, anhand von Parametern, untersucht welche Bereiche begehbar sind.

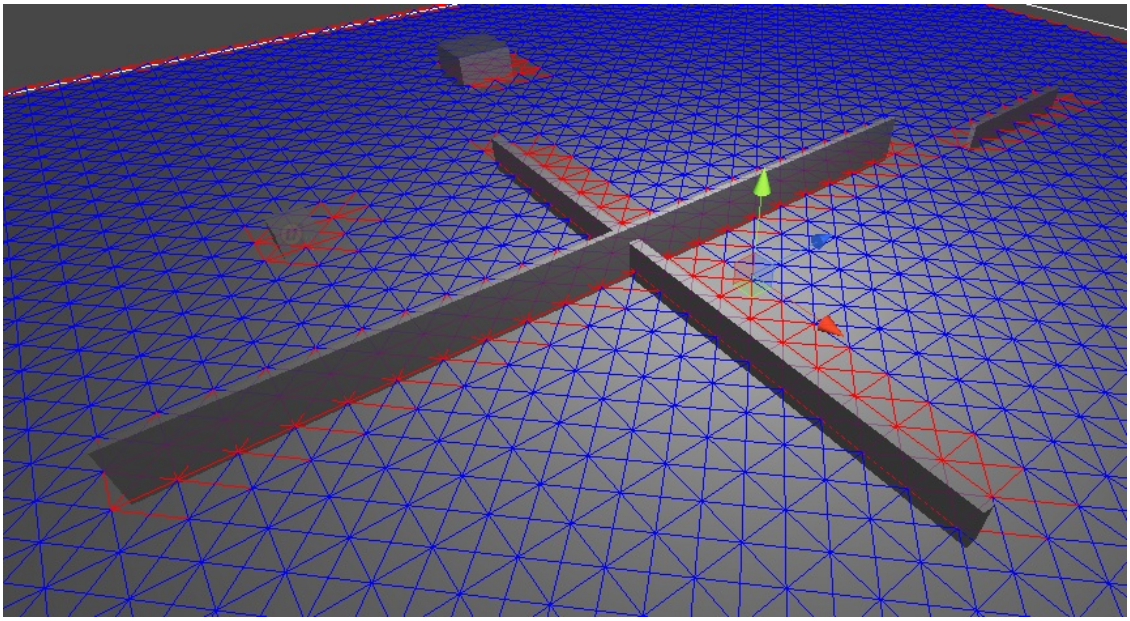


Abbildung 2.2: Preprocessing-Raster

Der begehbare Bereich ist durch das blaue Raster dargestellt. Der rote Bereich ist als nicht begehbar definiert, um eine Kollision mit den Objekten, die eine definierte Grenzhöhe überschreiten zu vermeiden.
Das Realtime Processing läuft parallel zum Spiel bzw. der Anwendung ab. Anhand der Parameter Startposition und Endposition wird, mithilfe des im Preprocessing berechneten Rasters, der kürzeste Weg gefunden. Für die Berechnung des kürzesten Weges gibt es verschiedene Algorithmen:

Dijkstra Hülsmann [22] benennt den Dijkstra-Algorithmus als die Standardlösung bei der Suche nach einem kürzesten Weg. Der Algorithmus beginnt am Startknoten und fügt alle umliegenden erreichbaren Knoten einer Prioritätenwarteschlange hinzu. Die Sortierung in der Warteschlange erfolgt anhand der Distanz der Knoten zum Startknoten. Somit beginnt der Vorgang bei dem Knoten mit minimaler Distanz, indem der Algorithmus seine Nachbarknoten erfasst und in die Warteschlange integriert. Dieses Verfahren wird bis zum Erreichen des Zielknotens wiederholt. Ein Beispiel zeigt Abbildung 2.3.

Der 1. Durchlauf beginnt bei dem Knoten Frankfurt und fügt seine Nachbarknoten, sortiert nach der Distanz, in die Warteschlange ein. Der 2.Durchlauf fügt die Nachbarknoten des ersten Knotens der Warteschlange hinzu und sortiert nun die

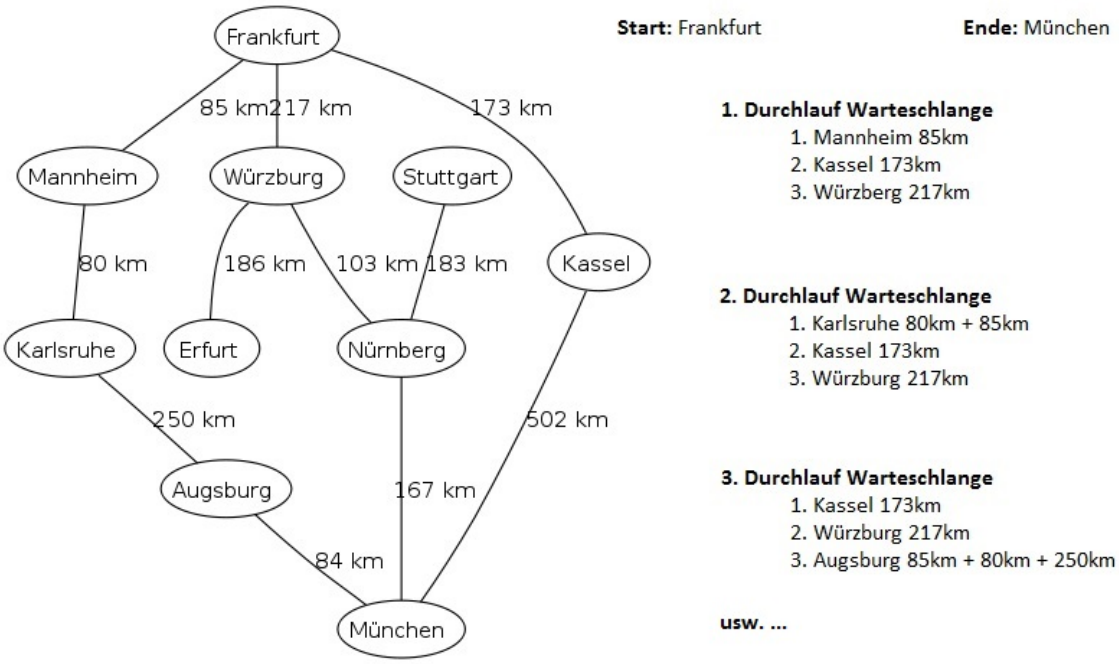


Abbildung 2.3: Dijkstra

Warteschlange erneut aufsteigend nach der Distanz zum Startknoten. Dieser Vorgang wird solange wiederholt bis der Algorithmus den Zielknoten München erreicht.

A-Star A* Dies ist ein ähnliches Vorgehen wie bei dem Dijkstra Algorithmus. Steinke [23] beschreibt, dass ständig eine Schätzung über die Restdistanz zum Zielknoten erfolgt. Dies geschieht durch die Ermittlung des geometrischen Abstands zwischen der aktuellen Position und dem Zielpunkt. Ebenso wird die Distanz der Nachbarknoten, bezogen auf die aktuelle Position, zu der Zielposition ermittelt und als sogenannte Kosten abgespeichert. In komplexeren Anwendungen werden zusätzliche Kosten berücksichtigt. Das sind z.B. Untergrundbeschaffenheiten, die ein Bewegen auf dem Untergrund erschweren oder erleichtern. Vom Starknoten werden nun alle Nachbarknoten untersucht. Er wählt hierbei den Knoten mit den geringsten Kosten (i.d.R. der geringsten Distanz) zum Ziel aus und untersucht von diesem Knoten ausgehend erneut seine Nachbarknoten. Dies geschieht solange, bis das Ziel erreicht ist.

Um ein realistisches Ergebnis des Wegpfades zu erhalten erfolgt, nach Kastenholz, eine Pfadglättung durch Splines. (s. Abbildung 2.4) Die Stärke der Glättung lässt sich über entsprechende Parameter einstellen bzw. korrigieren.

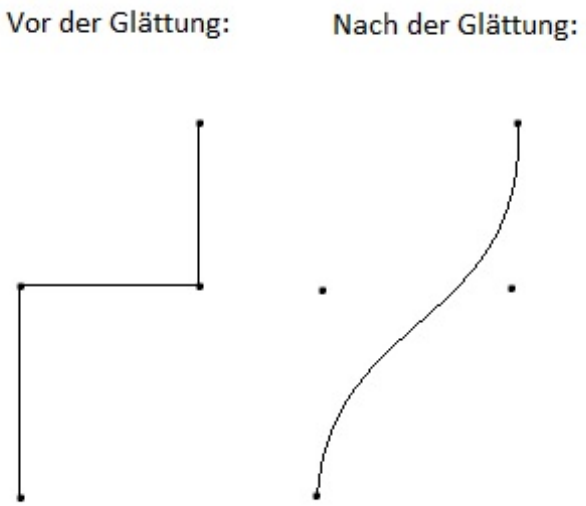
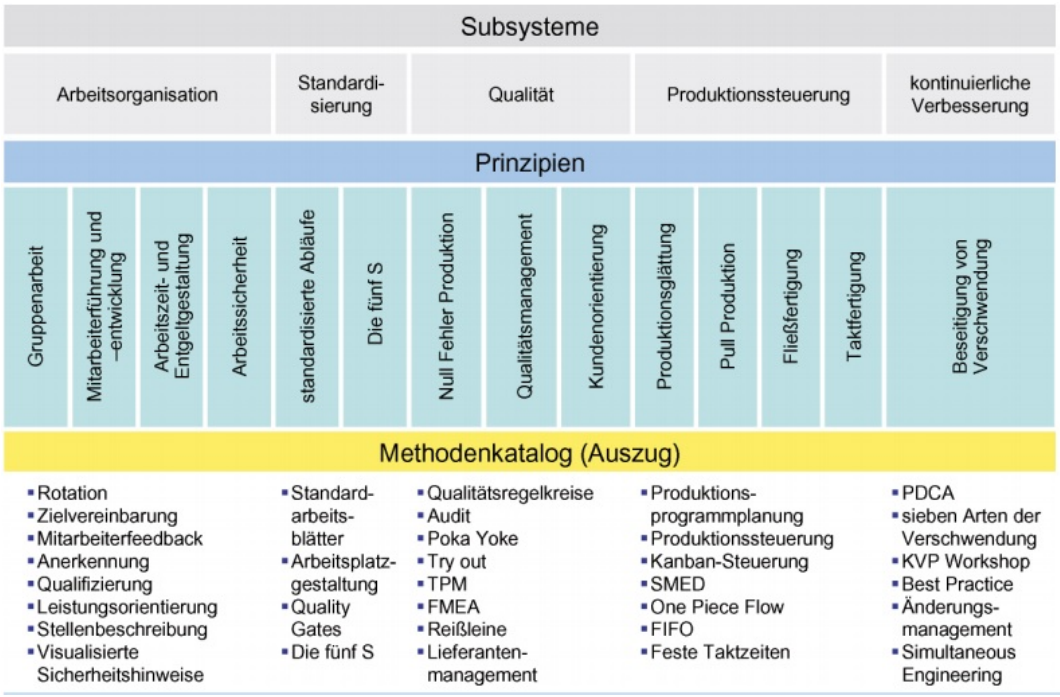


Abbildung 2.4: Pfadglättung

2.4 Produktionssystem

Barth [24] beschreibt, dass ein Produktionssystem alle Prozesse eines Fertigungsunternehmens umfasst. Grundsätzlich lässt es sich unterteilen in Subsysteme, Prinzipien und Methoden. (s Abbildung 2.5) Weiterhin erläutert Barth, dass die angewandten Prinzipien und die Methoden stark von dem Fertigungstyp, dem Produkt und der Unternehmenskultur abhängig sind. Daher entwickelt jedes Unternehmen sein eigenes Produktionssystem. Die Prinzipien und Methoden sind daher lediglich als Orientierungshilfe und Leitlinie aufzufassen. Sie basieren grundsätzlich auf dem Lean Management-Konzept (schlanke Unternehmensführung), das viele Unternehmen in der heutigen Zeit verfolgen, um ihr Produktionssytem zu optimieren bzw. zukunftssicher zu gestalten. Es ist letztendlich aufgrund eines Individualisierungstrends der Nachfrage entstanden. Die japanische Automobilindustrie erkannte zuerst diesen Trend und entwickelte das Lean-Management-Konzept. Die Kernidee besteht darin, Verschwendungen während der Wertschöpfungskette zu vermeiden, in dem Prozesse optimiert bzw. unnötige Prozessschritte vermieden werden. Ebenso wichtig ist hierbei, dass die eingeführten Prozesse nicht als starr gehandhabt werden, sondern sie so auszulegen, dass sie flexibel sind, um schnell auf Marktänderungen bzw. Kundenwünsche reagieren zu können.
Die Produktionsplanung und -steuerung (PPS) nimmt, nach Kurbel [25], eine wichtige Rolle in einem Produktionssystem ein. Sie befasst sich mit der Planung und Steuerung der



Quelle: Produktionssysteme im Fokus [24]

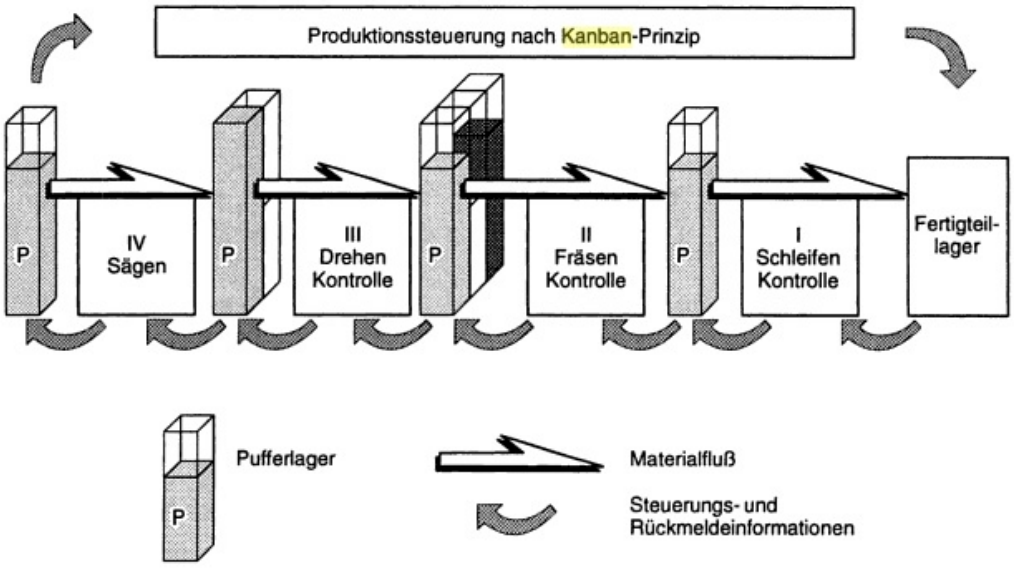
Abbildung 2.5: Produktionssystem

operativen Ebene. Unter vorgegebenen Rahmenbedingungen sind Entscheidungen über Produktionsmengen innerhalb eines definierten Zeitrahmens zu treffen. Günther [26] beschreibt, dass es zwei grundlegend unterschiedliche Konzepte gibt:

Push-Prinzip Hierbei werden die Produktionsaufträge in den Produktionsprozess „hineingedrückt“. Auf Basis aller Aufträge erstellt ein computergestütztes PPS-System, das regelmäßig einen Informationsaustausch mit den Daten aus der Produktion vornimmt, eine Durchlaufterminierung. Hierbei erfolgt eine Berechnung, mithilfe von durchschnittlichen Zeiten der einzelnen Bearbeitungsschritte von Auftragsterminen. Im Anschluss wird eine Kapazitätsauslastung für jede Ressource ermittelt, in dem der Kapazitätsbedarf dem Kapazitätsangebot gegenübergestellt wird. Das Ausgleichen von Kapazitätsüberlastungen erfolgt anschließend durch Terminverschiebungen oder Überstunden. Jede Ressource erhält dann einen Start- und Endtermin für die Auftrage aus der Auftragsliste.

Pull-Prinzip Abbildung 2.6 veranschaulicht das Pull-Prinzip. Die Produktion erfolgt erst, wenn ein Kunde bestellt oder die Bestände auf ein Minimum sinken. Die Grundidee ist, dass die Produktionseinheiten exakt die Mengen herstellen, die benötigt werden.

In der Regel erfolgt die Steuerung von der letzten Fertigungsstufe. Sobald dort eine Unterschreitung des Lagerbestandes, eines benötigten Materials, unter einen definierten Mindestwert eintritt, löst dies ein Fertigungsauftrag der vorgelagerten Fertigungseinheit aus. Das Pull-Prinzip, auch Kanban genannt, funktioniert somit über eine Selbststeuerung durch die Mitarbeiter, die durch visuelle Anzeigen die nötigen Informationen erhalten, um entsprechende Materialanforderungen oder Fertigungen auszulösen.



Quelle: Handhabung der Fertigungstechnik [27]

Abbildung 2.6: Produktionssteuerung nach Kanban-Prinzip (Pull-Prinzip)

Wichtigste Voraussetzung für einen reibungslosen Ablauf nach dem Pull-Prinzip ist, nach Spur [27], die Harmonisierung der einzelnen Kapazitätseinheiten und möglichst kleine Lose fließen zu lassen. Dies ist durch die Stückelung eines Kundenauftrages zu erreichen. Eine Harmonisierung erfolgt durch das Angleichen von Prozesszeiten der Produktionseinheiten. Pufferbestände sind somit lediglich nur noch zum Ausgleich von Störungen nötig, aber nicht mehr zum Ausgleich unterschiedlicher Arbeitsgeschwindigkeiten und Arbeitstakte. Durch eine erfolgreiche Implementierung entfällt das Lagern von Teilprodukten und Fertigwaren und der damit verbundene logistische Aufwand.

2.5 3D-Simulation

Eine Simulation dient nach Bossel [28] der Analyse eines Systems, um hieraus Erkenntnisse für ein reales System zu gewinnen. In der Regel wird eine Simulation ausgeführt, wenn die rechnerische bzw. theoretische Behandlung zu komplex ist. März [29] beschreibt zusätzlich, dass dies häufig bei dynamischen Systemen gegeben ist. Optimierungen lassen sich mithilfe einer Simulation durch die Variation der Einflussgröße erreichen. Die Qualität der Optimierung ist durch ein Vergleichen der Ergebnisdaten mit den Zielvorgaben messbar. Die Computersimulation spielt heute eine wichtige Rolle. Sie hat eine Vielzahl an realen Versuchssystemen abgelöst. Bossel führt folgende Gründe für das Ablösen auf:

- Die Kosten der Modellerstellung und Simulation sind wesentlich geringer als bei realen Systemen.
- Der zeitliche Ablauf der Simulation kann verändert werden. Reale Systeme. die für Beobachtungen zu schnell arbeiten, lassen sich z.B. verlangsamen. Ebenso lassen sich langsame Systeme beschleunigen.
- Eine Dynamik, die evtl. zu einer Systemzerstörung führen würde, hat auf den Computer keine Auswirkungen

Weiterhin beschreibt Bossel, dass sich eine Simulation auf zwei verschiedene Weisen realisieren lässt. Bei der ersten Methode handelt es sich um ein Nachbilden der Wirkungsweise des realen Systems. Hierbei müssen die einzelnen Systemkomponenten bekannt sein und als Computermodell abgebildet werden. Eine Verknüpfung der Komponenten führt dann zu einem Simulationsmodell des realen Systems. Im zweiten Fall erfolgt, durch Beobachtungen des Verhaltens des realen Systems unter unterschiedlichen Bedingungen, eine Verhaltensbeschreibung. Die Beschreibung wird mathematisch abgebildet und in das Simulationsmodell übertragen.
Nach Mattern [30] wird die Simulation in verschiedenen Anwendungsbereichen eingesetzt:

- Industrielle Simulation, z.B. zur Materialflussplanung
- Ökologie, z.B. Modelle zur Schadstoffausbreitung
- Volkswirtschaft, z.B. Marktmodelle

- Architektur, z.B. graphisch animierte Planungsmodelle
- Forschung, z.B. physikalische Chemie

Gerade im Zusammenhang mit der Digitalen Fabrik spielt, nach Kühn [31], die 3D-Simulation eine wichtige Rolle. Hierbei wird eine virtuelle Fabrik, dass digitale Abbild einer realen oder geplanten Fabrik, erzeugt. Ein solches Abbild dient frühzeitig der Schwachstellenerkennung und weist sich, nach Bayer und Collisi [32], als ein besonders effizientes Kommunikationsmittel aus. Desweiteren definiert Kühn folgende Ziele, die durch eine Digitale Fabrik erreicht werden sollen:

- Verbesserung der Wirtschaftlichkeit
- Verbesserung der Planungsqualität
- Verkürzung der Produkteinführungszeit
- Transparente Kommunikation
- Standardisierung von Planungsprozessen
- Kompetentes Wissensmanagement

3 Entwurfsprozess der 3D-Simulationssoftware

Dieses Kapitel beschreibt die wichtigsten Schritte, die der Entwurfsprozess beinhaltet. Abschnitt 3.1 beschreibt die Ziele und Zwecke, die die Simulation schwerpunktmäßig erreichen soll. Hierauf basiert die Systemdefinition (Abschnitt 3.2), in der die Anforderungen an die Software erläutert sind. Ebenso beinhaltet das Kapitel eine Festlegung von Parametern bzw. Merkmalen der Simulation, wie z.B. Systemgrenzen oder die vorgesehenen zulässigen Systemänderungen.
Kern dieses Kapitels ist der Softwareentwurf bzw. das Systemkonzept der Simulation. (Abschnitt 3.3) Dieses beschreibt das Gesamtsystem, sowie die Hauptkomponenten der Simulation, mithilfe von UML-Diagrammen. Als Abschluss des Kapitels, definiert Abschnitt 3.4, die Zustandsgrößen und verhaltensbestimmenden Größen, die das Simulationssystem messbar und veränderbar machen.

3.1 Ziele und Einsatzzwecke

Das übergeordnete Ziel ist eine 3D-Simulationssoftware, die ein variables Produktionssystem simuliert und abbildet, zu realisieren. Das Produktionssystem besteht aus Produktionseinheiten, logistischen Einheiten und Lagereinheiten. Im Fokus steht hierbei die Verkehrsfluss- bzw. Materialflusssimulation.
Es sollen Veränderungsmöglichkeiten an dem Produktionssystem gegeben sein. Eine wichtige Veränderungsmöglichkeit ist hierbei das Verschieben von Fertigungseinheiten. Aber auch Änderungen von Produktionsfaktoren, wie z.B. das Definieren von Fertigungszeiten der Produktionseinheiten, sind vorgesehen. Die logistischen Einheiten reagieren während des Simulationsvorganges dynamisch auf die vorgenommen Änderungen und finden den optimalen Weg automatisch.

Die Simulation kann von verschiedenen Positionen, die der Benutzer definiert, betrachte werden. Um das 3D Erlebnis während der Simulation zu verstärken, besteht die Möglichkeit 3D-Hardware, wie z.B. ein Head-Mounted Display, zu nutzen. Es wird eine Aufzeichnung von wichtigen Kennzahlen des simulierten Produktionssytems, wie z.B. die Gesamtdurchlaufzeit, verlangt.
Es sind verschiedenen Einsatzzwecke der 3D-Simulationssoftware vorgesehen:

Optimierung: Testen von verschiedenen Szenarien, in denen ein Abbild einer realen Produktion erstellt und durch Änderungen von Komponenten eine mögliche Optimierung entsteht, z.B. eine Wegoptimierung nach einer Verschiebung einer Fertigungseinheit.

Schulung: Einsetzen der Simulation in der Lehre um z.B. verschiedene Produktionssteuerungsarten, Push-Systeme oder Pull-Systeme, kennenzulernen und diese, durch die optische Komponente, besser zu verstehen.

Präsentation: Durch die hohe optische Qualität, aufgrund der Nutzung einer Gameengine, bietet sich die Software ebenso als Präsentationstool an. Hiermit lässt sich z.B. eine neuentwickelte Produktion von verschiedenen Kamerapositionen begutachten. Dieser Effekt lässt sich durch 3D-Hardwarekomponenten, die integrierbar sind, verstärken.

Kommunikation: Durch das freie Bewegen im Raum ist ein Einsatz als Kommunikationstool denkbar. Durch ein Navigieren zu dem entsprechenden Bereich, der Teil der Diskussion ist, lässt sich die Kommunikation vereinfachen bzw. Missverständnisse vermeiden.

3.2 Systemdefinition und Anforderungen

Wie aus den Zielen zu entnehmen, bezieht sich die Simulation auf ein Produktionssystem. Als Systemgrenzen, die das System von der Umwelt abgrenzen, gelten die Kanten des Fabriklayouts. (s. Abbildung 3.1)
Das System kommuniziert über Schnittstellen mit der Umwelt. Es gibt eine Input und eine Output Schnittstelle, die als das Eingangs- bzw. Ausgangslager definiert sind. Über das Eingangslager gelangt neues Material in das System und über das Ausgangslager aus dem System hinaus.

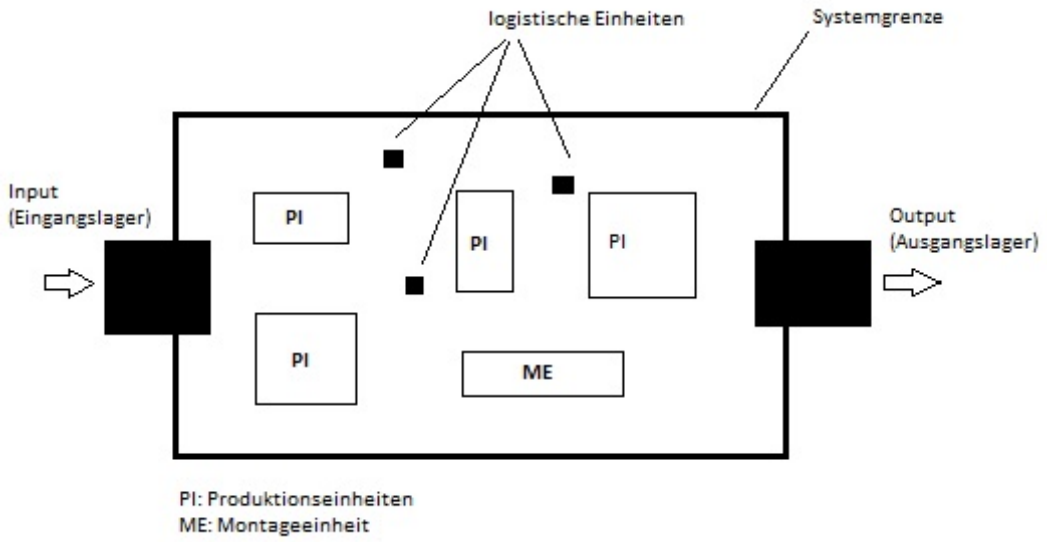


Abbildung 3.1: Systemgrenze

Durch die Ziele und die gewünschten Einsatzzwecke ergeben sich Anforderungen, die die 3D Simulationssoftware zu erfüllen hat:

3D Abbildung und Simulation eines Produktionssystem: Der Nutzer kann die Perspektive während eines Simulationsvorganges frei wählen. Die Fahrwege der logistischen Einheiten werden mithilfe der KI – Wegfindung (s. Abschnitt 2.3) berechnet und gesteuert.

Simulation jeder einzelnen Produktionseinheit: Jede Produktionseinheit hat intern eine eigenen Simulation, die die Fertigung der Produkte simuliert. Sie benötigt daher einen Warenausgang (WE) und einen Wareneingang (WA).

Systemänderung: Dies sind Veränderungen des Produktionssystems in einem Editiermodus. Es besteht die Möglichkeit, Kennzahlen der Produktionseinheiten zu verändern, wie z.B. die durchschnittliche Fertigungszeit. Ebenso lassen sich die Produktionseinheiten, sowie der Wareneingang und Warenausgang in dem voreingestellten Fabrikgrundriss verschieben. Die Anzahl der Gabelstapler sind einstellbar.

Wegfindung der logistischen Einheiten: Die logistischen Einheiten finden automatisch den optimalen Weg zu dem definierten Ziel.

Einbindung von 3D-Hardware möglich: Die Software bietet die Möglichkeit eine Videobrille oder ein 3D-Shuttersystem als visuelles Ausgabegerät zu nutzen.

Aufzeichnung wichtiger Kennzahlen: Produktionskennzahlen werden aufgezeichnet, um Simulationsergebnisse vergleichen zu können.

3.3 Softwareentwurf

Der Softwareentwurf ist ein wichtiger Bestandteil des Realisierungsprozesses. Er beschreibt, nach Arnold [33], die wesentlichen Systembausteine und deren Beziehung zueinander. Unterabschnitt 3.3.1 bis Unterabschnitt 3.3.3 erklärt die Funktionen und die Schnittstellen der einzelnen Softwarekomponenten.
Zur Erläuterung der Softwarearchitektur dient in diesem Fall ein Komponentendiagramm, das die Softwarekomponenten und deren Beziehungen zueinander zeigt. Es ist Teil der Unified Modeling Language (UML), die einen Standard für grafische Notation, für verschiedene Diagrammtypen, in der Softwaremodellierung definiert. [34]
Eine Softwarekomponente kann, nach Heide [34], aus mehreren Unterkomponenten bestehen und / oder aus verschiedenen (Software-)Artefakten, wie z.B. ein Quellcode oder eine Bibliotheksdatei. Sie zeichnet sich dadurch aus, dass sie wiederverwendbar ist und Schnittstellen für den Datenaustausch hat.
Eine komponentenbasierte Software bietet sich in diesem Fall aufgrund der Wiederverwendbarkeit von Komponenten an. Hierdurch lässt sich z.B. die Anzahl von logistischen Einheiten verändern ohne aufwendige Anpassungen vornehmen zu müssen. Auch eine Integration eines frei veränderbaren Produktionssystems, entsprechend den Anforderungen (s. Abschnitt 3.2), ist durch das Komponentensystem wesentlich einfacher.
Aufgrund der Übersichtlichkeit erfolgte eine Aufteilung des Gesamtsystems in zwei Systemzustände: Simulationsmodus und Editormodus. Abbildung 3.2 zeigt das erste Komponentendiagramm des Gesamtsystems während es sich im Simulationsmodus befindet. Der Simulationsmodus zeichnet sich dadurch aus, dass die Simulationslaufzeit fortschreitet und somit der Editormodus nicht aufrufbar ist.
In der Abbildung ist zu erkennen, dass das Gesamtsystem aus verschiedenen Komponenten besteht: Produktionseinheiten, logistischen Einheiten, Lagereinheiten, Graphical User Interface (GUI), Kamerasystem und einem Controller.

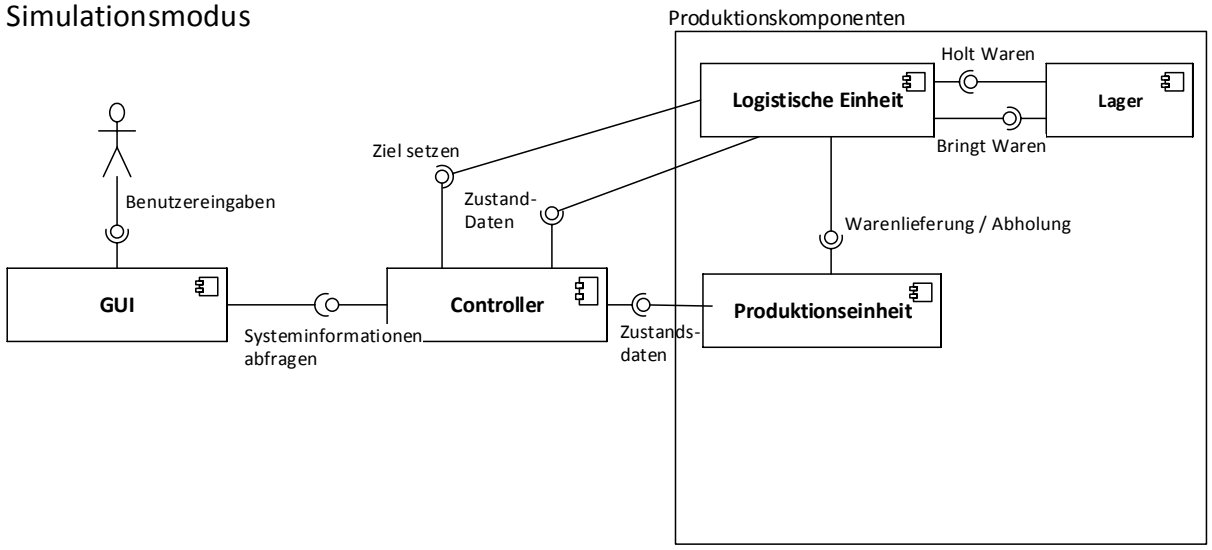


Abbildung 3.2: Komponentendiagramm Simulationsmodus

Der Controller steht in der hierarchischen Systemordnung ganz oben, da dieser Informationen bzw. Anweisungen an die Produktion verteilt. Hierzu wertet er ständig, in einem gewissen zeitlichen Abstand, Informationen der logistischen Einheiten und der Produktionseinheiten aus. Die genauere Funktion des Controllers wird in Unterabschnitt 3.3.1 erläutert.
Die Produktionseinheiten, logistischen Einheiten und Lagereinheiten bilden Zusammen die Produktion. Eine Beschreibung der einzelnen Produktionskomponenten erfolgt in Unterabschnitt 3.3.2. In dem GUI-System ist die Bedienoberfläche und die zugehörige Programmierung hinterlegt. Das System interagiert mit den Produktionskomponenten, um sich hieraus die jeweiligen Informationen zu beschaffen. Durch Benutzereingaben können Eigenschaften der Produktionskomponenten verändert werden. (s. Unterabschnitt 3.3.3)
Abbildung 3.3 zeigt das zweite Komponentendiagramm des Gesamtsystems während es sich im Editormodus befindet. Der Editormodus ist vor dem Simulationsstart geöffnet bzw. ist während einer Simulationspause aufrufbar und eingeschränkt nutzbar. Hierüber lassen sich Komponenteneigenschaften verändern oder neue Komponenten hinzufügen. Die Änderungswünsche erreicht die jeweilige Einheit über das GUI, in dem der Benutzer die gewünschte Änderung eingibt.
In Abbildung 3.3 ist der Controller nicht eingezeichnet, da dieser im Editormodus keine Funktionen hat. Es werden nur die Komponenten angezeigt, die auch tatsächlich vom

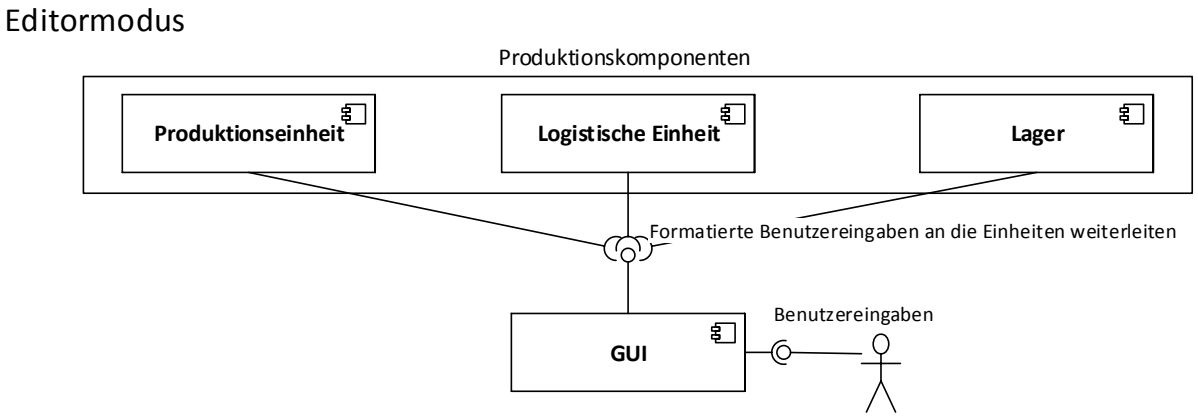


Abbildung 3.3: Komponentendiagramm Editormodus

Benutzer über das GUI veränderbar sind.

3.3.1 Controller

Der Controller regelt die Kommunikation der Hauptsysteme untereinander und verteilt entsprechend die Lieferungs- bzw. Abholaufträge an die logistischen Einheiten. Abbildung 3.4 zeigt das Aktivitätsdiagramm der Hauptaktivität des Controllers.
Er prüft in einem regelmäßigen Abstand ob die Produktionseinheiten Waren benötigen oder Waren zur Abholung bereit stehen. Falls ein Bedarf besteht, fügt er die Produktionseinheit einer Warteschlange hinzu. Die Sortierung der Warteliste erfolgt nach der Wartezeit der Produktionseinheiten. Gleichzeitig gliedert der Controller wartende logistische Einheiten in eine Warteschlange ein. Auch hier erfolgt die Sortierung nach der Wartezeit der logistischen Einheiten.
Der ersten Produktionseinheit aus der Warteliste der Produktionseinheiten wird nun die erste logistische Einheit aus der Warteliste der logistischen Einheiten zugewiesen. Durch das Löschen dieser Einheiten aus den Wartelisten rücken nun die Einheiten der zweiten Position an die erste Position und der nächste Auftrag wird vergeben.

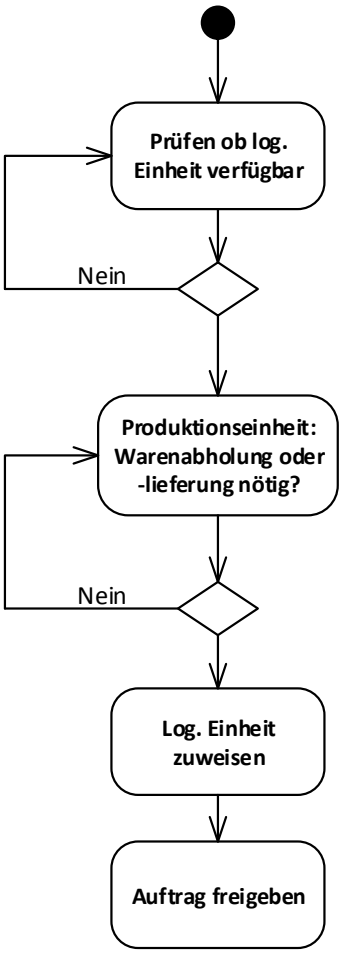


Abbildung 3.4: Aktivität des Controllers

3.3.2 Produktionskomponenten

Die Produktion besteht aus voneinander abhängigen Einheiten, die alle eine grafische 3D-Komponente besitzen. Sie spiegeln zusammen das grafische 3D-Abbild der Produktion wieder. Eine Simulationskomponente, die auf ein verknüpftes Script bzw. Programmcode zugreift, simuliert den individuellen Ablauf / Vorgang einer Einheit. Die Produktion besteht aus drei verschiedenen Einheiten: Produktionseinheiten, logistische Einheiten und Lagereinheiten.

Produktionseinheiten Sie bilden zusammen die Fertigung der Produktion. Jede Produktionseinheit verwendet, aufgrund der Wiederverwendung der Komponente, die

gleiche Simulationskomponente zur Simulation des internen Produktionsprozesses. Um Produktionseinheiten mit unterschiedlichen Produktionsparametern abzubilden, erfolgt eine Individualisierung der jeweiligen Produktionseinheit über Änderungen von Variablen der Simulationskomponente. Abbildung 3.5 zeigt vereinfacht den Ablauf bei einem Fertigungsauftrag.

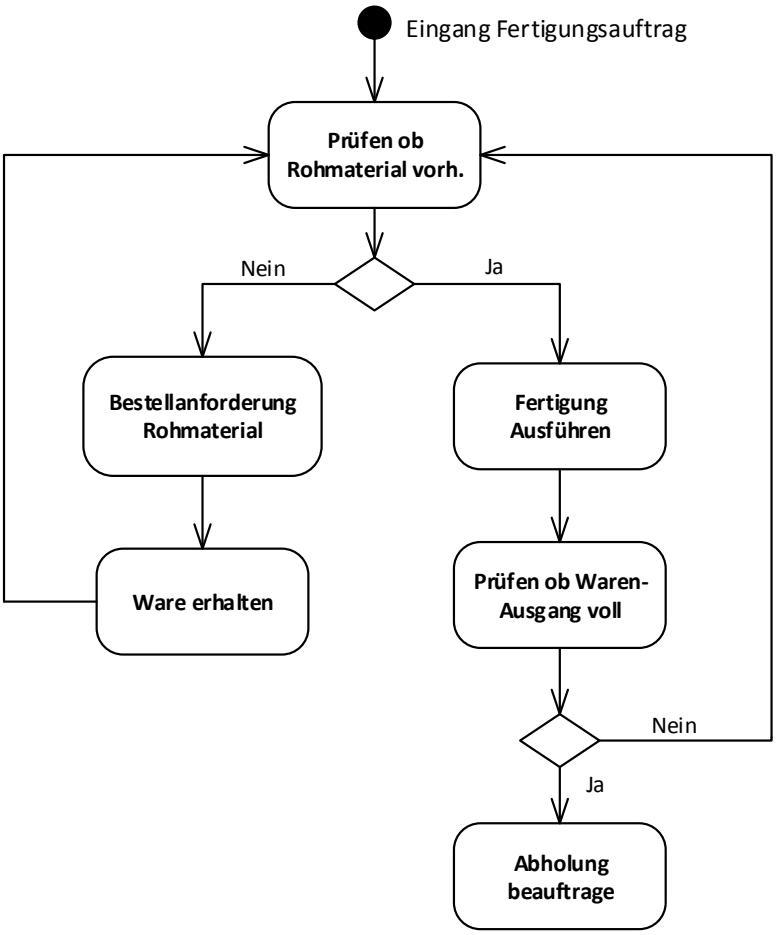


Abbildung 3.5: Aktivität einer Produktionseinheit

Eine Bestellanforderung für Rohmaterial oder einen Auftrag für eine Warenabholung einer Produktionseinheit erkennt der Controller und ordnet der Produktionseinheit einen freien Zulieferer, also eine logistische Einheit zu. Nach Aufruf der Aktivität „Fertigung ausführen“, startet der interne Simulationsprozess der Produktionseinheit. Hierbei wird mithilfe eines Zeitfaktors in bestimmten, schwankenden, Abständen ei-

ne Wareneinheit produziert und gleichzeitig eine definierbare Menge an Rohmaterial entfernt. Die schwankenden Abstände, innerhalb eines bestimmen Bereiches, werden mithilfe einer Zufallszahl generiert. Hierdurch soll eine Produktionsschwankung, die z.B. aufgrund eines kurzen Maschinenausfalls entstehen kann, abgebildet werden.

Logistische Einheiten Sie sorgen für den Warentransport zwischen den Produktionseinheiten und den Lagereinheiten. Abbildung 3.6 verdeutlicht den Ablauf bei einer Warenbelieferung einer Produktionseinheit. Der Ablauf beginnt mit der Prüfung,

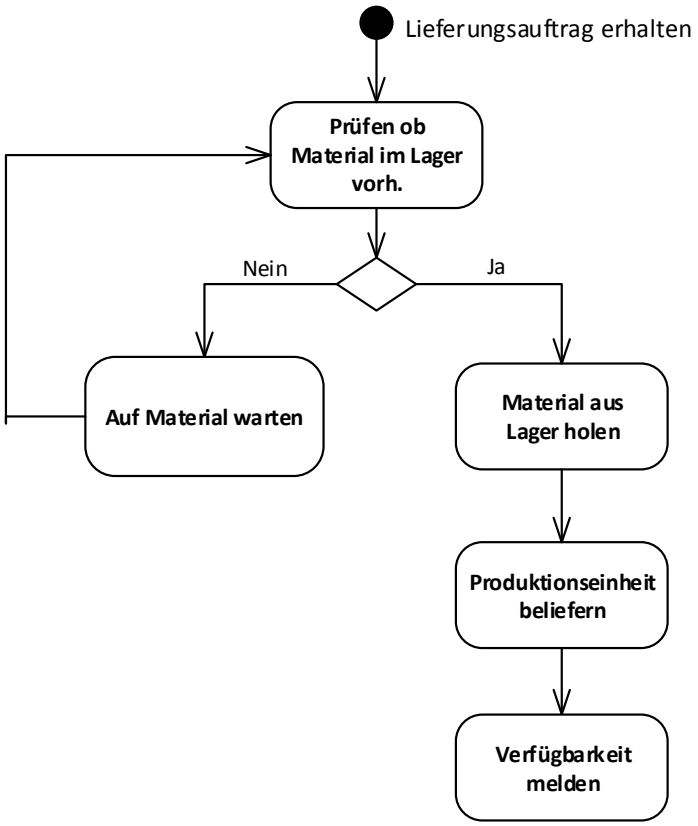


Abbildung 3.6: Aktivität einer logistischen Einheit

ob das gewünschte Material in ausreichender Menge im Lager vorhanden ist. Hierbei kann es sich um das Eingangslager – Rohmaterial oder dem Zwischenlager – Baugruppen etc. handeln. Falls nicht ausreichend Material vorhanden ist, wartet die logistische Einheit auf das Material und prüft regelmäßig, ob die gewünschte Menge vorhanden ist. Bei ausreichend vorhandener Menge wird das Material an den Gabelstapler übergeben. Dieser sucht sich nun den kürzesten Weg zu der Pro-

duktionseinheit, die das Material benötigt und übertragt der Produktionseinheit das Material nach Ankunft. Im Anschluss meldet die logistische Einheit, dass sie für Neuaufträge verfügbar ist. Bei einer Warenabholung von einer Produktionseinheit läuft ein sehr ähnlicher Prozess ab, wobei die Ablaufreihenfolge natürlich anders ist.

Lagereinheiten Sie sind unterteilt in Eingangslager und Ausgangslager. In dem Eingangslager sind die Bestellteile hinterlegt. Der Benutzer kann Bestellteile hinzufügen bzw. entfernen und die Mengen der Teile ändern. Als Alternative besteht die Möglichkeit den Wareneingang simulieren zu lassen. Hierbei basiert der Simulationsablauf auf historischen Daten mit denen eine Prognose berechnet wird. Eine Wareneingangssimulation hätte den Vorteil, dass Produktionskonsequenzen, die aufgrund von Lieferengpässen entstehen, abgebildet werden.
Die Erzeugung von Kundenbedarfen erfolgt im Ausgangslager, da die Fertigteile an dieser Stelle das Produktionssystem verlassen. Es kann als Schnittstelle zu dem Kunden angesehen werden. Die Ermittlung der Kundenbedarfe und ihre zeitliche Verteilung ist simulationsgesteuert. Es erfolgt, analog zum Eingangslager, eine Prognoseberechnung mithilfe von historischen Daten. Eine manuelle Eingabe von Kundenbedarfen ist auch möglich.
Das Ausgangslager enthält nun, entsprechend der Kundenbedarfe, von einer Montageinsel hergestellte Fertigteile. Hier kann der Benutzer keine Veränderungen vornehmen. Diese Teile werden während des Simulationsvorganges erzeugt und zu dem Lager geliefert.

3.3.3 GUI-System

Das GUI-System dient der Benutzereingabe, um Veränderungen an der Simulation vorzunehmen und Informationen zu erhalten. Dies geschieht, in dem es direkt auf die Produktionskomponenten zugreift, um Veränderungen vorzunehmen bzw. Informationen zu erhalten. Es ist unterteilt in ein Simulationsmenü und einem Editormenü.
Das Simulationsmenü enthält anwählbare Elemente zum Verändern von Simulationsparameter, wie z.B. die Simulationsgeschwindigkeit. In entsprechend vorgesehenen Bildschirmbereichen erhält der Benutzer verschiedene Informationen zu dem aktuell ausgewähltem Hauptsystem wie z.B. die aktuelle Produktionsmenge. Abbildung 3.7 zeigt den Entwurf für das Simulationsmenü.

In dem Editorsystem ist es möglich, über entsprechende Eingabefelder, Parameter einzel-

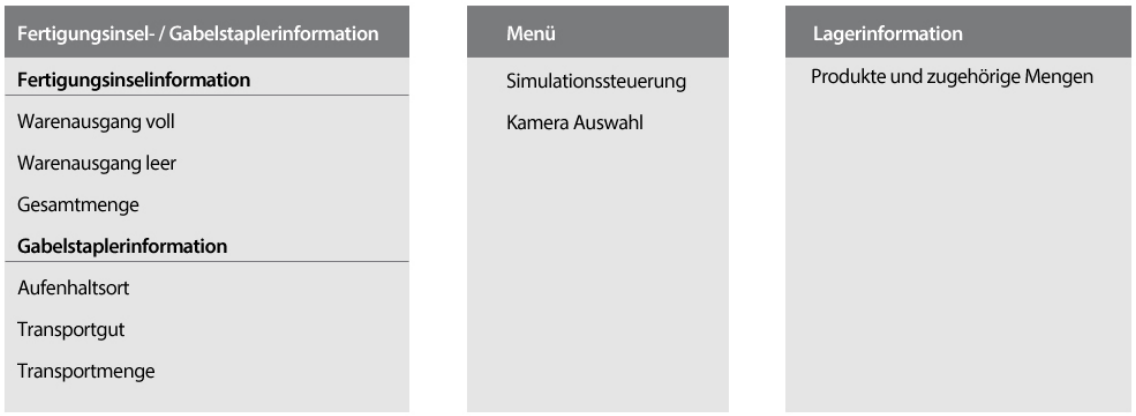


Abbildung 3.7: Entwurf für das Simulationsmenü

ner Hauptsysteme zu verändern, wie z.B. die durchschnittliche Fertigungszeit. Auch ein Umpositionieren von Produktions- bzw. Lagereinheiten ist möglich. Abbildung 3.8 zeigt den Entwurf für das Editormenü.

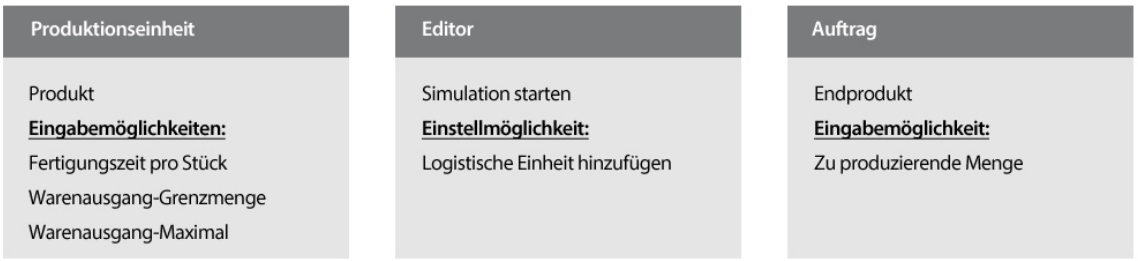


Abbildung 3.8: Entwurf für das Editormenü

3.4 Zustandsgrößen und verhaltensbestimmende Größen

Um einen Simulationsdurchgang auswerten zu können muss, eine Festlegung von geeignete Zustandsgrößen erfolgen. Diese speichern für jeden Zeitpunkt wichtige Systemkennzahlen. Durch eine Gegenüberstellung von Zustandsgrößen lässt sich ein Zusammenhang

zwischen Zustandsgrößen herstellen. Dies ist, nach Bossel [28], eine wichtige Erkenntnis, um z.B. systematisch Systemoptimierungen vornehmen zu können.
Über die verhaltensbestimmenden Größen lässt sich das Produktionssystem der Simulation beeinflussen. Eine Systemoptimierung kann nun mit dem hergestellten Zusammenhang der Zustandsgrößen, über eine Änderung einer verhaltensbestimmenden Größe, erreicht werden. Tabelle 3.1 zeigt die definierten Zustandsgrößen und verhaltensbestimmenden Größen, sortiert nach den Systemeinheiten.

	3D-Zustandsgrößen	Verhaltensbestimmende Größen
Produktionseinheiten	• Gesamtmenge an hergestellten Erzeugnissen • Stillstand / Produktionsausfall • aktuelle Menge im Warenausgang • aktuelle Menge im Wareneingang	• Puffergröße am Wareneingang und -ausgang • Grenzmenge für Bestell- bzw. Abholauslösung am Wareneingang und -ausgang • Produktionszeit pro Einheit • Position der Produktionseinheit
Logistische Einheiten	• Wartezeit • Zurückgelegter Weg • Transportiere Menge	• Anzahl an logistischen Einheiten • Geschwindigkeit • max. Transportmengen
Lagereinheiten	• Produkte mit Mengenangaben	• Position der Lagereinheit

Tabelle 3.1: Zustandsgrößen und verhaltensbestimmende Größen

4 Implementierung und Anwendungsbeispiel

Dieses Kapitel stellt den Implementierungsprozess und die Simulationssoftware an einem Beispiel vor. Abschnitt 4.1 erklärt die verwendeten Technologien und beschreibt die Auswahlkriterien, die für die verwendeten Technologie gesprochen haben. Abschnitt 4.2 und Abschnitt 4.3 stellen die Realisierung wichtiger Funktionen vor, wie z.B. die Wegfindung der logistischen Einheiten. Abschnitt 4.4 stellt ein Anwendungsbeispiel und wichtige Analysemöglichkeiten anhand des Beispiels vor.

4.1 Verwendete Technologien und Rahmenbedingungen

4.1.1 Das Produktionssystem

In der ersten Softwareversion wird ein voreingestelltes Produktionssystem als Grundlage genutzt, dass eine eingeschränkte Flexibilisierung zulässt. Der Aufbau des Produktionssystems orientiert sich an einer Fahrradproduktion, da die Fahrradkomponenten bekannt sind und sich somit die Produktionsschritte nachvollziehen lassen. Die Erzeugnisstruktur ist Abbildung 4.1 zu entnehmen. Die Zahl in der Klammer hinter den Materialien gibt die erforderliche Menge für genau ein Fahrrad wieder. Wie die Abbildung zeigt ist die Fahrradproduktion vereinfacht. Teile mit geringem Transportaufwand und geringem Wert, wie z.B. Schrauben, Unterlegscheiben oder Muttern finden keine Beachtung in der Beispielproduktion. Diese hätte eine unnötige Komplexitätssteigerung zur Folge, die in der ersten Version für Testzwecke ungeeignet wäre.
Die hieraus abgeleitete Produktionslayout ist Abbildung 4.2 zu entnehmen. Es handelt sich hierbei um Produktionseinheiten bzw. Montageeinheiten , da diese noch nicht alle

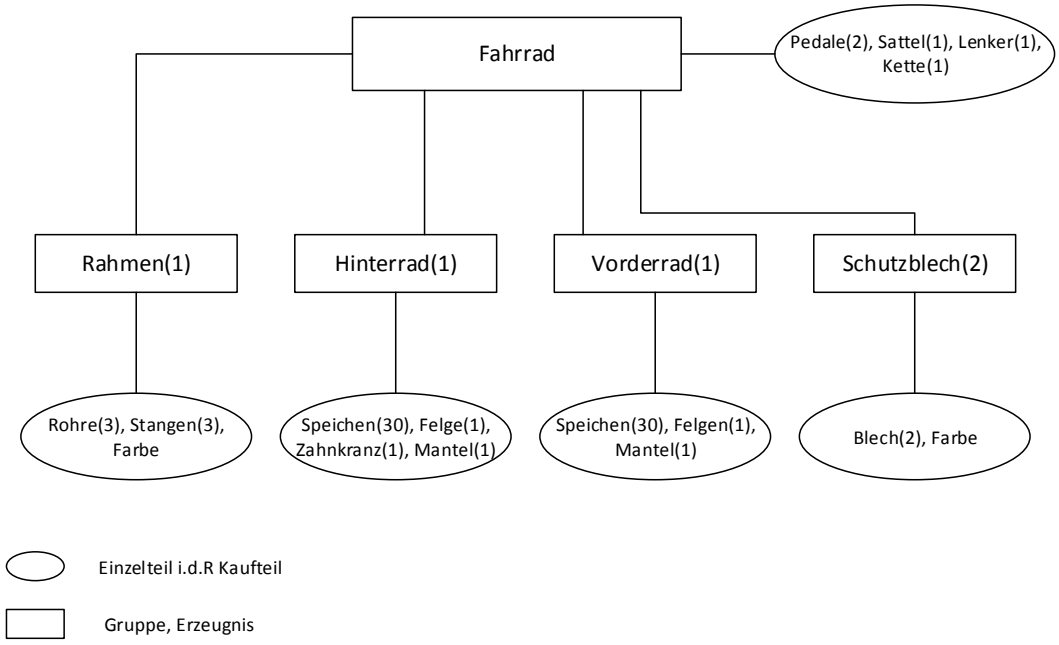


Abbildung 4.1: Erzeugnisstruktur

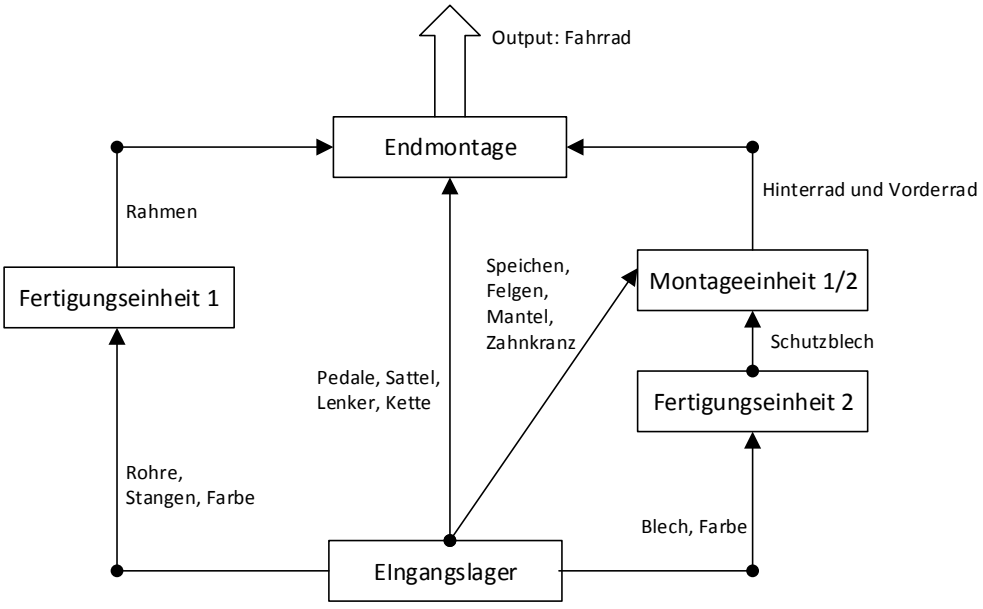


Abbildung 4.2: Produktionslayout

Eigenschaften einer Fertigungsinsel bzw. Montageinsel erfüllen. Sie erzeugen, wie in der Abbildung zu erkennen, jeweils nur ein Ausgangsprodukt. Ebenfalls ist zu erkennen, dass kein Zwischenlager vorgesehen ist. Die Montageeinheiten verfügen aber über Puffer am Wareneingang und Warenausgang.

4.1.2 Wahl der Gameengine

In diesem Abschnitt wird die ausgewählte Entwicklungsumgebung, die auf einer Gameengine basiert, vorgestellt. Aufgrund von fortschrittlichen Erfahrungen mit verschiedenen Gameengines lag der Gedanke nahe, diese Software mithilfe einer solchen Engine umzusetzen. Eine solche Umsetzung bietet, gerade im Zusammen mit der Realisierung einer 3D-Simulationssoftware, verschiedene Vorteile:

- hohe grafische Qualität erreichbar aufgrund der integrierten umfangreichen 3D-Engine, die jede Gameengine besitzt
- verschiedene Programmiersprachen / Programmierarten werden unterstützt
- eine Vielzahl von Modulen lassen sich integrieren, wie z.B. KI-Wegfindungssysteme

Eine Übersicht über verschiedene Gameeninges gibt Tabelle 4.1. Diese ist auf drei Gameengines beschränkt. Eine Aussortierung erfolgte in einer ersten Vorauswahl anhand von zu hohen Lizenzkosten.

	Unity3D Free [35]	Blender [36]	WebGL [37]
Unterstütze Zielplattformen	Windows, Mac OS X, Webbrowser, Android, iOS	Windows, Mac OS X, Linux	Plattformunabhängig da Webbasiert
Lizenzierungskosten	Kostenlos	Kostenlos	Kostenlos
Skriptsprachen	C++, JavaScript, Boo	Python	JavaScript
visueller Scripteditor vorhanden?	Ja	Ja	Ja - Alpha-Version daher nur eingeschränkt nutzbar
Editor vorhanden?	Ja	Ja	Ja

Tabelle 4.1: Übersicht Gameengines

Die Wahl fiel, in erster Linie, aufgrund von persönlichen fortgeschrittenen Grundkenntnissen, auf die Unity3D-Eingine. Hierbei ist zu erwähnen, dass auch die Blender-Engine sicherlich gut für eine 3D-Simulation geeignet wäre. Da sie sich, ähnlich wie Unity3D, durch eine einfache Bedienung auszeichnet und, aufgrund eines visuellen Scripteditors, wenig Programmierkenntnis voraussetzt.
WebGL ist eine relativ neue Engine (seit 2009), wobei es sich hierbei, nach [37], nicht um eine klassische 3D-Engine handelt, sondern um eine 3D-Grafik-Programmierschnittstelle

für den Webbrowser. Desweiteren kritisiert Steinberg [38] an WebGL den enormen Programmieraufwand. Dies ist auch auf das Fehlen eines Editors zurückzuführen, der z.B. das Importieren von Objekten oder das Erstellen von Objektanimationen deutlich vereinfacht. Im Folgenden sind zusammenfassend Vorteile der Unity3D-Engine aufgelistet:

Linzenzierungsmodell: geringe Entwicklungskosten aufgrund einer kostenlosen Basisversion. Die kostenpflichtige Version bietet zusätzliche Funktionen, die i.d.R. hauptsächliche professionelle Spielentwickler ansprechen wie z.B. Bild- und Soundeffekte.

Unterstützung vieler Zielplattformen: es ist eine breite Zielgruppe mit verschiedenen Endgeräten erreichbar

Entwicklungsumgebung übersichtlicher Editor, der sehr dem Aufbau gängiger Animationssoftware ähnelt Abbildung 4.3

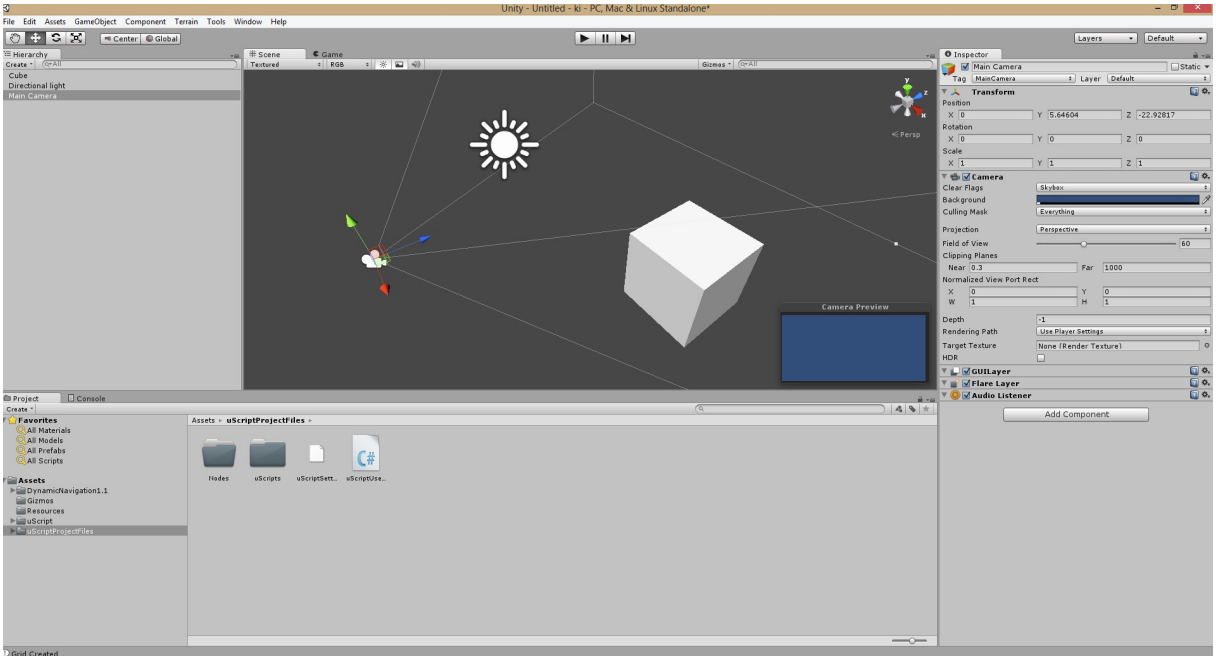


Abbildung 4.3: Unity3D Entwicklungsumgebung

Große Community: umfangreicher Support in verschiedenen Foren vorhanden

Umfangreicher Asset-Store: über diese Funktion lassen sich Plugins bzw. Erweiterungen von freien Entwicklern und Drittanbietern integrieren. Somit ist die Gameengine stark an die eigenen Bedürfnisse anpassbar. Es besteht z.B. die Möglichkeit visuelle Scripteditoren zu integrieren.

Komponentensystem: die Eigenschaften eines Objektes lassen sich, durch das Hinzufügen, Verändern oder Entfernen von einer Vielzahl an Komponenten, schnell verändern. Durch das Hinzufügen einer Materialkomponente lässt sich z.B. das Aussehen der Oberfläche des Objektes verändern.

Quelle: Unity3D

4.1.3 Wahl der Programmierart / Nutzung eines visuellen Scripteditors

Ein visueller Scripteditor bietet die Möglichkeit zu Programmieren ohne einen Quellcode zu schreiben. Durch logische Verknüpfungen der Ein- und Ausgänge von Blöcken, die als Linien dargestellt werden, lassen sich Logiken abbilden aus denen im Hintergrund ein Quellcode generiert wird. Ein Block, oder auch Node genannt, hat immer mindestens einen Ausgang in den meisten Fällen aber auch einen oder mehrere Eingänge. Die Logik beginnt immer bei einem Event-Block, der ein Signal nach Ausführung eines Events, z.B. einen Tastendruck sendet.
Die Entscheidung für einen visuellen Scripteditor fiel aufgrund der bisherigen Erfahrungen mit visuellen Prozessstrukturen während des Studiums. Abbildung 4.4 zeigt ein Beispiel, eine Umsetzung der Logik für einen Lichtschalter.

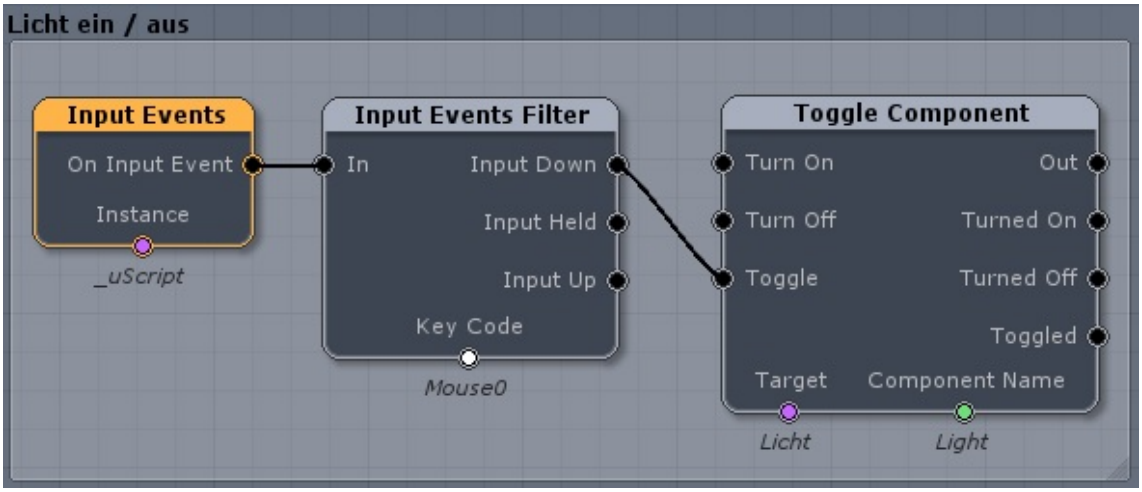


Abbildung 4.4: uScript Beispiel

Der Input Events Block sendet ein Signal sobald ein Eingabegerät, Maus oder Tastatur, betätigt wird. Das Signal wird in dem Input Events Filter Block nur weitergeleitet, falls die

gedrückte Taste dem definierten Key Code entspricht. In dem Beispiel ist der Key Code Mouse0, welches der linken Maustaste entspricht. In dem Toggle Component Block wird nun, bei Eingang eines Signals, der Lichtzustand zwischen ein und aus gewechselt.
Quelle: uScript [39]

4.1.4 Wahl der KI-Wegfindung

Die genutzte KI-Wegfindung basiert auf dem A-Star A*-Algorithmus (s. Abschnitt 2.3), da dieser im Gegensatz zum Dijkstra-Algorithmus, aufgrund einer Schätzfunktion, eine geringere Laufzeit aufweist und somit schneller den optimalen Weg zum Ziel findet. Des weiteren unterstützt die genutzte KI-Wegfindung eine dynamische Rasterbildung des Suchraums. Das heißt, dass die Möglichkeit besteht, dass Pre-Processing während der Laufzeit aufzurufen, um somit auf eine veränderbare Umgebung zu reagieren. Diese Funktion ist nützlich, um z.B. nach der räumlichen Verschiebung einer Produktionseinheit das Raster mit den Begehungsinformationen des 3D-Suchraumes zu aktualisieren.
Abbildung 4.5 zeigt einen Vergleich von einem Raster, in der eine Objektverschiebung, vor und nach einer Aktualisierung, erfolgte. In dem zweiten Abschnitt der Abbildung ist zu erkennen, dass sich ein Objekt in einem blauen Gitterbereich befindet. Diese hätte zur Folge, dass der Gabelstapler mit dem Objekt kollidiert, falls sich dieses auf dem berechneten Wegpfad befindet. Durch eine Gitteraktualisierung, siehe dritter Abschnitt der Abbildung, ließe sich eine solche Kollision vermeiden.

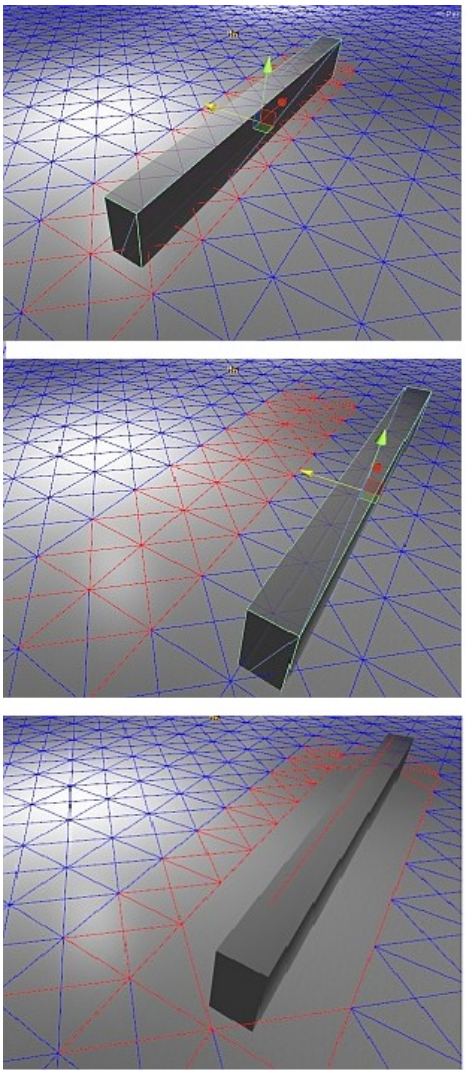


Abbildung 4.5: Vergleich Rasteraktualisierung

4.2 Simulationsmodus - Umsetzung und Funktionen

Die Simulation befindet sich nach dem Simulationsstart in dem Simulationsmodus, d.h. die Programmlaufzeit schreitet fort.

Umsetzung Die Umsetzung erfolgte auf Basis der in Abschnitt 3.3 beschriebenen Komponenten:

- Controller
- Produktionseinheiten
- Lagereinheiten
- Logistische Einheiten

Die Generierung von Erzeugnissen bzw. Materialbedarfen der Produktionseinheiten beschreibt Unterabschnitt 4.2.1. Während eines Simulationslaufes prüft der Controller in regelmäßigen Abständen, ob eine Produktionseinheit Ware benötigt oder Ware zur Abholung bereit steht.
Abbildung 4.6 zeigt die Umsetzung am Beispiel der Kaufteileabfrage. Kaufteile liegen im Eingangslager bereit. Produktionseinheiten gliedern alle benötigen Kaufteile in eine String

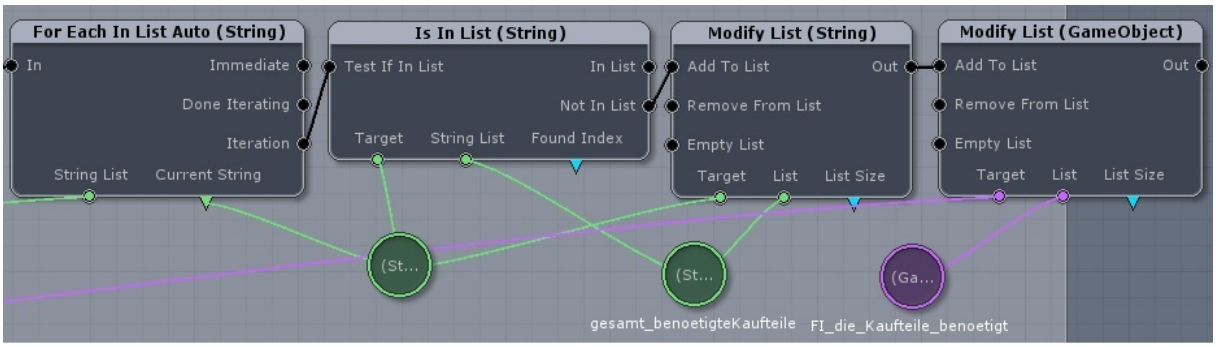


Abbildung 4.6: Controller - Kaufteile Abfrage

- Liste ein. Der Controller greift, bei allen Produktionseinheiten die Kaufteile benötigen, auf die zugehörige Liste zu. Dies geschieht mit dem *For Each In List Auto (String)* - Block. Anschließend gliedert der Controller diese Elemente, über den *Modify List (String)* - Block, in eine neue String-Liste ein. Diese Liste enthält nun also alle benötigten Kaufteile des

gesamten Produktionssystems. Parallel hierzu speichert er, an gleicher Position, die Produktionseinheiten in eine Gameobject-Liste, um eine Beziehung zwischen dem benötigtem Kaufteil und der Produktionseinheit, die dieses Kaufteil benötigt, herzustellen. Der gleiche Vorgang findet bei abzugebenden Fertigteilen, sowie bei benötigten und abzugebenden Erzeugnissen statt.
Es gibt nun also vier Auftragslisten mit vier dazugehörigen GameObject-Listen, die zusammen die Bedarfe sowie die zugehörigen Produktionseinheiten beinhalten. Der Controller weist nun nacheinander jedem ersten Element einer Auftragsliste eine freie logistische Einheit zu.
Freie logistische Einheiten erkennt er an einem sogenannten Trigger-Event. Ein Trigger-Event wird ausgelöst sobald eine logistische Einheit sich in dem Wartebereich aufhält.(s Abbildung 4.7) Hierzu erhält der *Trigger Event* - Block das Instance Element Wartebereich.

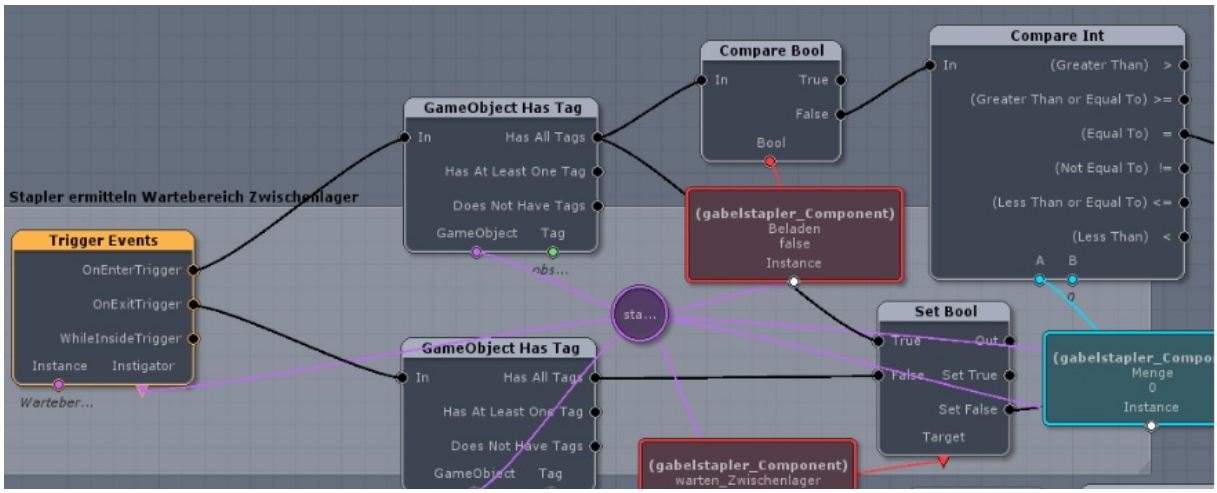


Abbildung 4.7: Controller - freie logistische Einheiten erkennen

Es findet nun jedes Mal eine Signalauslösung statt, sobald eine logistische Einheit den Wartebereich erreicht. Dieses Verhalten signalisiert dem Controller, dass die logistische Einheit verfügbar ist und gliedert die logistische Einheit in eine GameObject-Liste ein, die alle freien verfügbaren logistischen Einheiten enthalten.
Nach Erteilung eines Auftrages sucht sich die zugewiesene freie logistische Einheit automatisch den optimalen Weg und der Controller entfernt die Einheit aus der Liste, die alle freien logistischen Einheiten enthält. Eine genauere Erläuterung zu den logistischen Einheiten befindet sich in Unterabschnitt 4.2.3 .
Ein besonderer Fall tritt ein wenn eine Produktionseinheit Ware von einer vorgelagerten

Produktionseinheit benötigt. Abbildung 4.8) Hierzu prüft der Controller, mit dem *For Each in List (String)* - Block, ob das benötigte Erzeugnis der vorgelagerte Produktionseinheit in der entsprechenden Auftragsliste vorhanden ist und somit zur Abholung bereit steht. Erst dann erfolgt ein Lieferauftrag. Diese Verhalten ist einem Kanbansystem, also einem

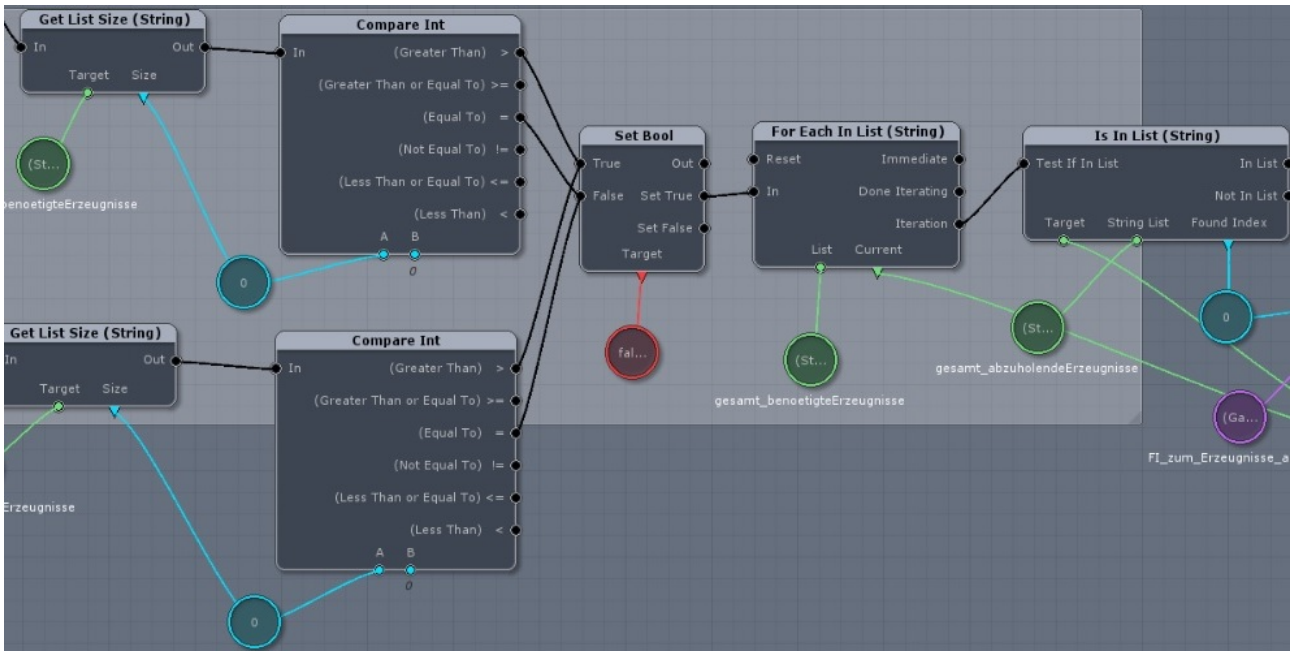


Abbildung 4.8: Controller - benötigte / abzuholende Erzeugnisse

Pull-System (s. Abschnitt 2.4), nachempfunden. Der Controller zieht die Ware durch die Produktion. Diesen Charakter verstärkt die Produktionsauslösung, die über das Definieren eines Kundenauftrages am Ausgangslager erfolgt. Dies geschieht über eine Eingabe des benötigten Endproduktes und die zugehörige Mengenangaben im Editormodus. (s Abschnitt 4.3) Sobald die gewünschte Menge erreicht ist stoppt die Produktion.

Die vollständige programmiertechnische Umsetzung, auf Basis des visuellen Scripteditors, ist im Anhang A1 auf der CD zu finden.

Funktionen In dem Simulationsmodus lässt sich, über das GUI (s. Abbildung 4.9) das Simulationssystem steuern. D.h. Die Simulationsgeschwindigkeit ist veränderbar. Ebenso lässt sich die Simulation anhalten. Über das GUI-Informationssystem, welches sich

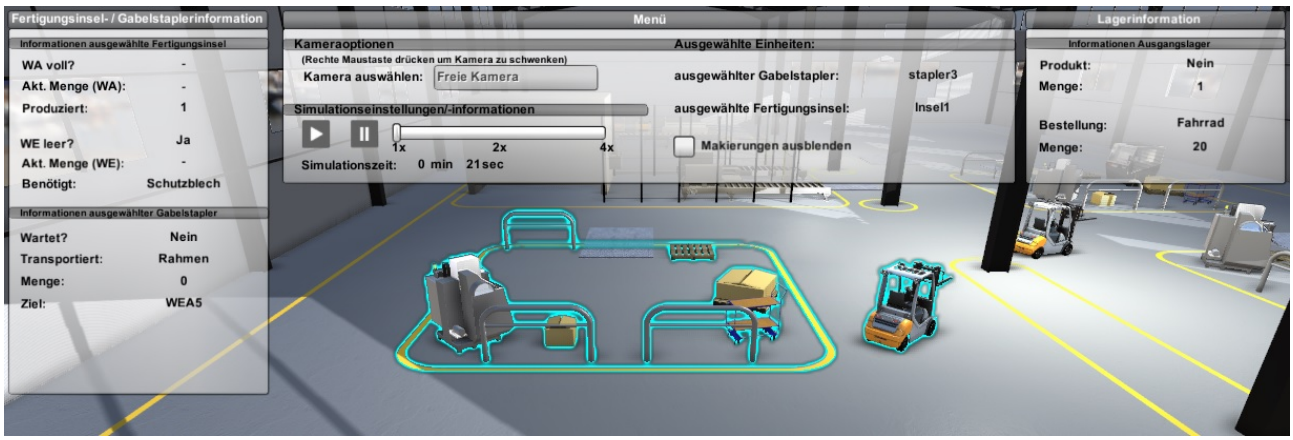


Abbildung 4.9: Simulationsmodus - GUI System

rechts und links von dem Menü befindet, lasse sich wichtige Kennzahlen der aktiven Einheiten und der Lagereinheiten anzeigen. Aktive Einheiten sind Einheiten die der Benutzer während der Simulation, per Mausklick auf die gewünschte Einheit, auswählt. Um eine visuelle Unterscheidung zwischen aktiven und nicht aktiven Einheiten zu gewähren erzeugt das System, wie ebenfalls in der Abbildung zu erkennen, einen hellblauen Markierungsrahmen um die aktiven Einheiten. Diese Markierungen lassen sich aber über eine Auswahlbox deaktivieren. Des Weiteren hat der Benutzer die Möglichkeit zwischen verschieden Kamerasystemen zu wählen:

Vogelperspektive Die Vogelperspektive betrachtet das Simulationssystem von oben. Es besteht keine Möglichkeit den Blickwindel zu verändern sondern es lässt sich lediglich über die Pfeiltasten die Kameraposition verändern.

Freie Kamera Die freie Kamera lässt sich frei im Raum bewegen. Dies geschieht ebenfalls mit den Pfeiltasten. Durch drücken der rechten Maustaste und bewegen der Maus in die gewünschte Richtung lässt sich die Kamera schwenken.

Folgende Kamera Die folgende Kamera folgt der aktiven logistischen Einheit bzw. nach Simulationsstart einer voreingestellten logistischen Einheit. Sie vermitteln das Gefühl auf dem Fahrersitz eines Gabelstaplers zu sitzen. Durch drücken der rechten Maustaste und bewegen der Maus lässt sich ebenfalls das Sichtfeld schwenken.

4.2.1 Produktionssimulation über verkettete Produktionseinheiten

Eine Produktionseinheit besteht aus mehreren 3D-Modellen, wie z.B. Werkeugmaschinen oder Laufbänder, sowie einer Simulationskomponente. (s. Abbildung 4.10) Die Produk-

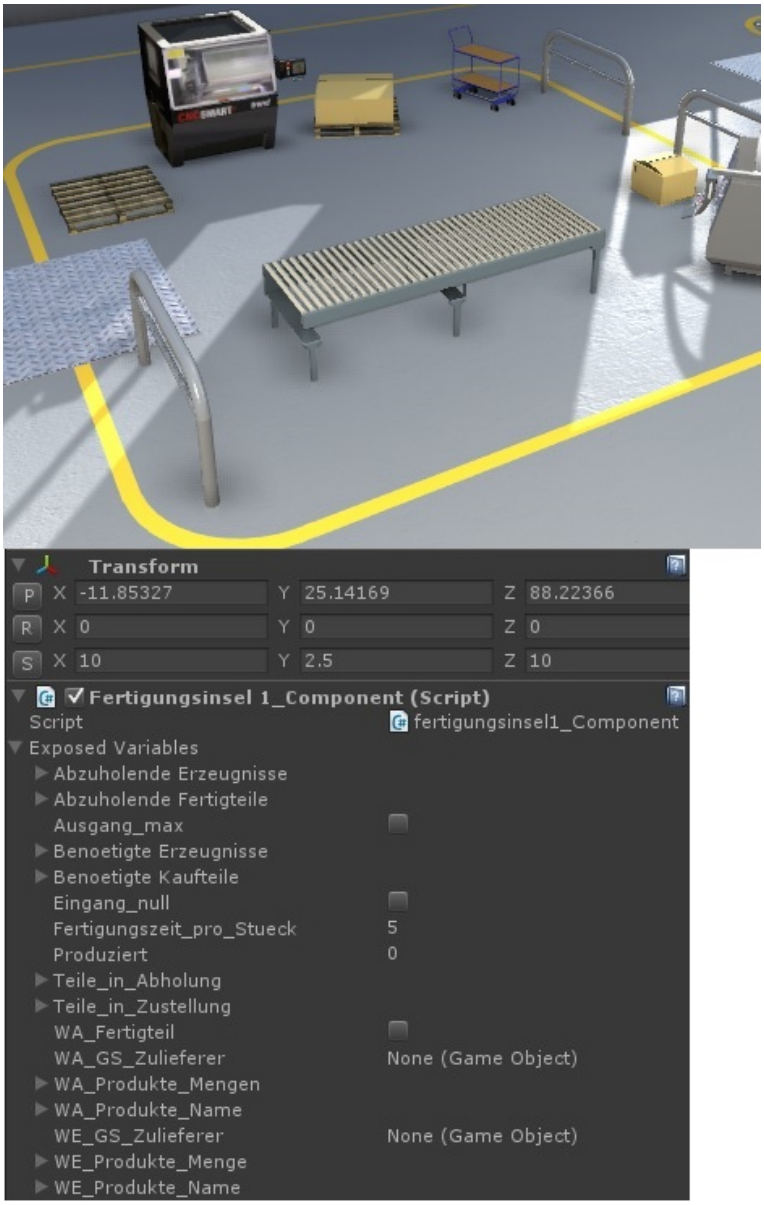


Abbildung 4.10: Produktionseinheit mit Komponente

tionseinheiten bzw. Montageeinheiten erzeugen die Materialbedarfe über eine interne Simulation, die über die Simulationskomponente gesteuert wird. Abhängig von einem

definierbaren Zeitfaktor *Fertigungszeit pro Stueck* (s. Abbildung 4.10) produzieren die Einheiten jeweils eine Wareneinheit und entnehmen hierzu eine entsprechende, ebenfalls definierbare Menge, der benötigten Ware(n) aus dem Wareneingang. Die Speicherung der produzierten Menge erfolgt in einer Vektor - Liste *(WA Produkte Mengen)* mit drei Komponenten, die noch zusätzliche Informationen enthalten:

1.Komponente: Speichert aktuelle Menge im Warenausgang

2.Komponente: Grenzmenge: Definiert ab welcher Menge ein Abholauftrag ausgelöst wird. Bei überschreiten der Grenzmenge wird das Erzeugnis in die Abzuholende Erzeugnisse - Liste eingetragen.

3.Komponente: Maximale Menge: Wenn aktuelle Menge = maximale Menge, dann folgt ein Produktionsstop

Parallel zu der Vektor - Liste wird der Erzeugnisname in die*WA Produkte Name* - Liste an gleicher Listeposition eingetragen, um einen Zusammenhang zwischen Menge und Erzeugnisname herzustellen. (s. Abbildung 4.11) Es handelt sich hierbei um Listen, damit

Vektorliste

Listenposition	Vektor
1	$\begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$
2	$\begin{bmatrix} 0 \\ 2 \\ 4 \end{bmatrix}$
usw.	

Erzeugnisliste

Listenposition	Erzeugnisname
1	Erzeugnis 1
2	Erzeugnis 2
usw.	

Abbildung 4.11: Vektor-Liste und Erzeugnisname-Liste

eine nachträgliche Implementierung einer Simulation mit Fertigungsinselverhalten, d.h. es können mehrere ähnliche Erzeugnisse von der gleichen Produktionseinheit hergestellt werden, einfacher umzusetzen ist. In dieser Simulationsversion handelt es sich um Produktionseinheiten die genau ein Erzeugnis herstellen.

Die Speicherung der benötigten Menge im Wareneingang erfolgt nachdem selben Prinzip. Allerdings handelt es sich hierbei um eine Vektor - Liste mit vier Komponenten. Die vierte Komponente definiert zusätzlich ob es sich um ein Kaufteil oder ein Erzeugnis einer vorgelagerten Produktionseinheit handelt. Dies ist eine wichtige Information für den Controller, da dieser den Auftrag dann entsprechend anders bearbeitet. (s. Abschnitt 4.2) Ein Sonderfall nimmt die letzte Produktionseinheit in der Produktionskette ein, da diese ein Fertigteil produziert. Diese wird über die boolsche Variable WA Fertigteil definiert. Hierdurch erkennt der Controller, ob es sich um Fertigteile handelt, um diese dann von den logistischen Einheiten in das Ausgangslager liefern zu lassen.

Die vollständige programmiertechnische Umsetzung, auf Basis des visuellen Scripteditors, ist im Anhang A1 auf der CD zu finden.

4.2.2 Wegfindung und Animation der logistischen Einheit

Eine logistische Einheiten besteht aus einem 3D-Gabelstaplermodel und sorgt für den Warentransport zwischen den Produktionseinheiten und den Lagereinheiten. Verschiedene Komponenten sind die für die Funktionen der Einheit verantwortlich sind. Abbildung 4.12 zeigt, aus welchen verschiedenen Komponenten sich die logistische Einheit zusammensetzt. Die Komponenten geben der Einheit das gewünschten Verhalten bzw. die gewünschten Eigenschaften.
Die interne Simulation, also den Be- und Entladevorgang, übernimmt die *GabelstaplerComponent* Komponente. Diese ist mit einem Script verknüpft, dass den Status, z.B. „Entladevorgang am Zwischenlager“, des Gabelstaplers von dem Controller erhält und hieraus das Verhalten, z.B. Entladeanimation für 10 Sekunden abspielen, bestimmt.
Um die KI-Wegfindung zu integrieren sind weitere Komponenten nötig. Die Komponente Navigator steuert den Gabelstapler zu dem Ziel. Hierzu ist eine sogenannte Grid Position Komponente nötig, die die momentane Position der Einheit auf dem Grid bestimmt um den A*-Alogorithmus die aktuelle Knoteninformationen zu übermitteln (s. Abschnitt 2.3). Das Grid ist ein durch das Preproccesing erzeugtes Gitterobjekt, dass Informationen zu begehbaren bzw. nicht begehbaren Bereichen enthält. In der AIMove Komponente ist das festgelegte Ziel definiert, sowie die Geschwindigkeit mit der sich die Einheit bewegt.

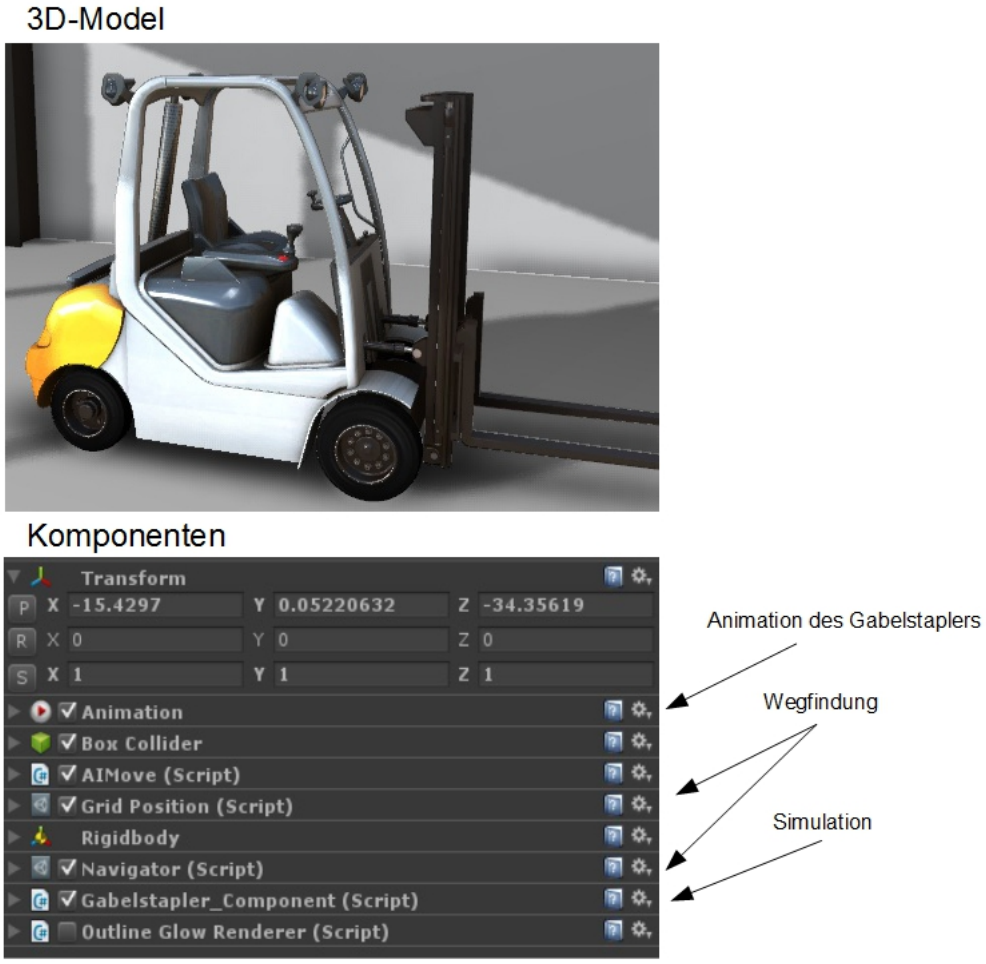


Abbildung 4.12: Logistische Einheit mit Komponenten

4.2.3 Einbinden Testen von 3D-Hardware

Dieser Abschnitt beschreibt die Einbindung von 3D-Hardwaresysstemen, die in den Grundlagen beschrieben sind (s. Abschnitt 2.2), in das Simulationssystem.

nVidia Vision 2 Ein nVidia Vision 2 System stand zu Testzwecken im Produktionslabor der HAW zur Verfügung. Ein Test hat gezeigt, dass sich dieses System durch eine sehr einfache Integration auszeichnet und problemlos mit dem in Unity3D erzeugten Simulationssystem arbeitet. Die einfache Integration ist möglich, da das nVidia Vision 2 System das Stereobild berechnet und dieses nicht auf die Software ausgelagert. Es sind also keine besonderen Anforderungen in der Softwareprogrammierung zu berücksichtigen. Der Kritikpunkt aus den Grundlagen bzgl. einigen Unschärfen ist zu bestätigen, wobei dieses stark vom Blickwinkel abhängig ist.

Oculus Rift Leider stand hierzu kein Testsystem zur Verfügung. Eine Einbindung des Oculus Rift Software Development Kit (SDK) hat jedoch gezeigt, dass dieses System ebenfalls mit Unity3D problemlos läuft. Nach Download des Oculus Rift SDK für Unity3D wird das SDK per Doppelklick in Unity3D integriert. Eine voreingestellte Kamera befindet sich in dem Oculus Rift Ordner, die durch das Platzieren in der Szene integriert wird.

Nach Einbindung ist auf dem Monitior das entsprechende Bild (s. Abbildung 4.13), dass normalerweise zu der Brille übertragen wird, zu erkennen. Es zeigt zwei Bilder der Szene aus einer leicht abweichenden Perspektive für das rechte bzw. linke Auge. Es fällt auf, dass die Darstellung des GUI-Systems nur auf dem Bild für das rechte Auge erfolgt. Entsprechende Lösungen hierzu bietet die Oculus Rift Community [40] an.

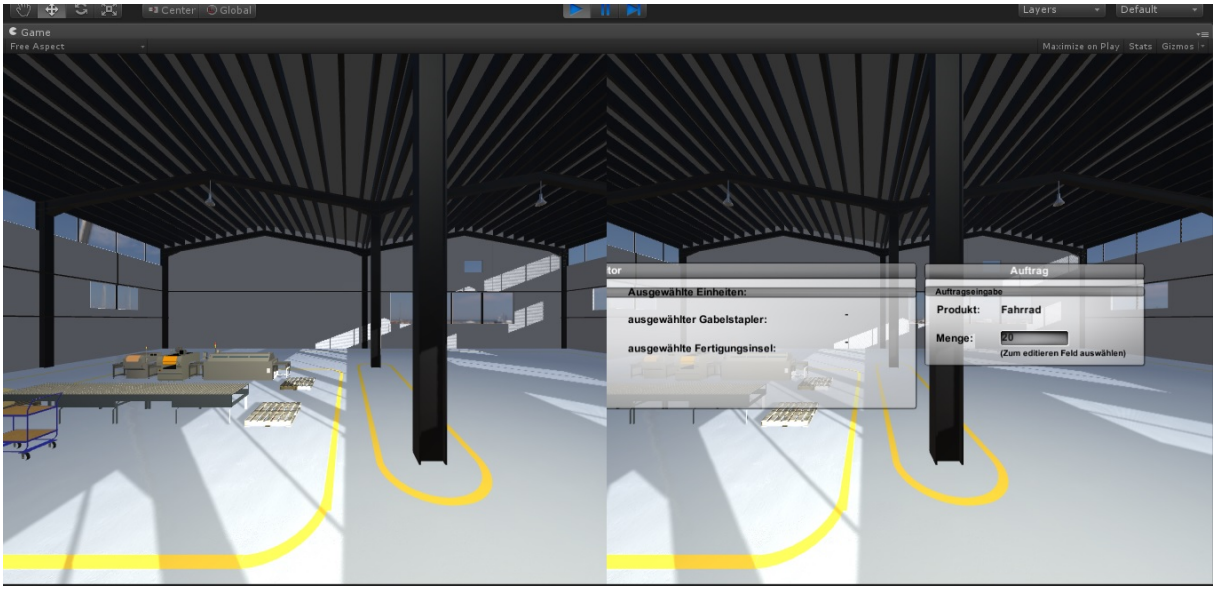


Abbildung 4.13: Screenshot Oculus Rift

4.3 Editormodus- Umsetzung und Funktionen

Die Simulation befindet sich nach dem Programmstart in dem Editormodus. In dem Editormodus lässt sich, über ein entsprechendes GUI, das Simulationssystem, welches sich im Stillstand befindet, verändern. (s. Abbildung 4.14) Folgende Änderungen am

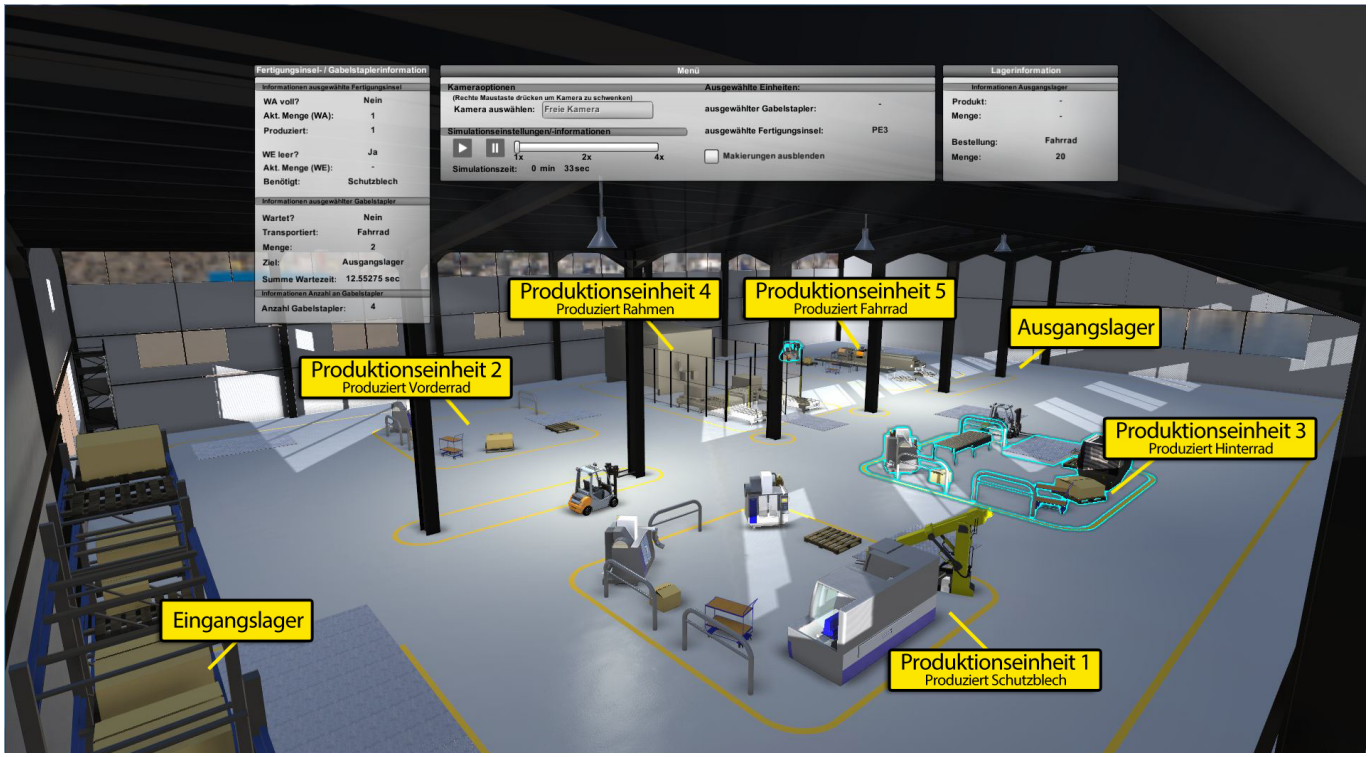


Abbildung 4.14: Editorsystem

Simulationssystem sind möglich:

Parameter der Produktionseinheiten Die gewünscht Einheit wird per Maus ausgewählt und ein Fenster zeigt die zugehörigen Parameter an. Veränderbare Parameter sind entsprechend gekennzeichnet und können direkt in dem Fenster neu definiert werden. Der Benutzer hat nun die Möglichkeit diese Variablen über das GUI-System zu editieren. Die entsprechenden GUI Eingabefelder sind mit den Scriptvariablen der ausgewählten Produktionseinheiten verknüpft.

Anzahl logistischer Einheiten Ebenso lässt die Anzahl an verfügbaren logistischen Einheiten verändern. Das Hinzufügen einer logistischen Einheit ist über den entspre-

chenden Button im Editormenü möglich. Das Programm stellt hierbei im Hintergrund eine Verbindung zu einem Prefab einer logistischen Einheit, die in einem Ressource-Ordner liegt, her. Ein Prefab ist in Unity3D ein vorkonfiguriertes Gameobjekt, dass bereits alle Komponenten enthält und beliebig oft in einer Szene duplizierbar ist. Bei diesem Vorgang wird eine sogenannte Instanz erzeugt, die mit dem Original Prefab verlinkt ist und hieraus alle benötigten Informationen erhält. Bei Veränderung des Prefabs werden somit alle erzeugten Instanzen ebenfalls verändert. (s. Unity3D Documentation [30])

Verschieben von logistischen Einheiten und Produktionseinheiten Beim Anwählen einer logistischen Einheit oder eine Produktionseinheit erscheinen Verschiebungspfeile in der Mitte des 3D-Modells. (s. Abbildung 4.15) Durch Anwählen eines Verschie-

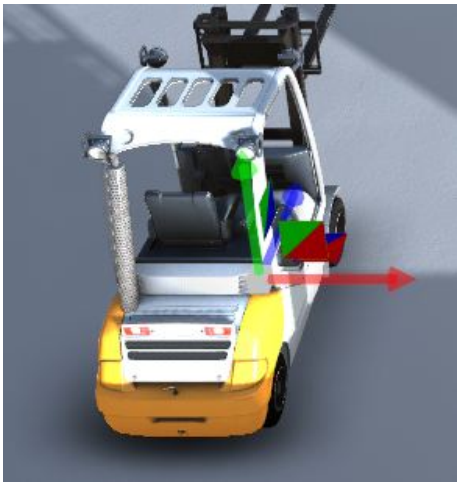


Abbildung 4.15: Verschiebungspfeile

bungspfeils ist es nun möglich die Einheit in die entsprechen Richtung, innerhalb des Produktionsbereiches, zu verschieben. Ebenfalls besteht die Möglichkeit das Objekt, um seine vertikale Achse, zu rotieren.

Anlegen eines Kundenauftrages In der rechten Menübox befindet sich die Auftragsansicht. Hier kann die gewünschte Menge angegeben werden. Die Produktion stoppt, nachdem Simulationsstart, sobald die angegebene Menge erreicht ist.

Durch Drücken des Startbuttons startet die Simulation, indem die Programmlaufzeit vom Wert 0 auf den Wert 1 gesetzt wird. Parallel hierzu erfolgt im Hintergrund ein Gitterupdate, denn durch das Verschieben der Produktionseinheiten muss erneut geprüft

werden, welche Bereiche für die logistischen Einheiten befahrbar sind um eine Kollision mit Produktionseinheiten zu vermeiden.

4.4 Anwendungsbeispiel

Dieses Anwendungsbeispiel verdeutlicht den Umgang mit der 3D-Simulationssoftware. Nach Starten der Simulation kann der Benutzer über vorgegebene Auswahlfelder die voreingestellten Werte für die jeweiligen Produktionseinheiten und den Auftrag verändern. Für die Produktionseinheiten sind die Werte für die Fertigungszeit/Stk., WA-Grenzmenge und die WA-Maxmenge einstellbar. Beim Auftrag ist die zu produzierende Menge veränderbar. Standardmäßig sind beim Start der Simulation zwei Gabelstapler aktiv. Dem Benutzer wird die Möglichkeit geboten, weitere Gabelstapler hinzuzufügen.

4.4.1 Durchführung

Der erste Simulationsdurchlauf begann mit den voreingestellten zwei Gabelstaplern. Nach jedem Durchlauf wurde die Anzahl der Gabelstapler jeweils um eine Einheit erhöht. Damit die Durchläufe vergleichbar sind, blieben die Einstellungen für Produktionseinheiten und des Auftrags konstant. Nach Simulationsende wurden die Kennzahlen notiert, wobei besonderes Augenmerk auf die Gesamtwartezeit je Gabelstapler und auf die gesamte Simulationszeit bis Auftragsende gelegt wurde.

4.4.2 Ergebnis und Auswertung

Tabelle 4.2 zeigt die Auswertung der Durchläufe.

Es zeigt sich, dass sich die Simulationszeit bei Steigerung der Anzahl an Gabelstapler verkürzt. Abbildung 4.16 verdeutlicht, dass sich der Effekt mit jedem weiteren Gabelstapler reduziert. Bei einer Anzahl von sechs Gabelstaplern tritt, im Vergleich zu fünf Gabelstaplern im System, keine Simulationszeitverkürzung mehr auf. Dies ist damit zu erklären, dass die Lieferbedarfe (Anzahl an Logistikaufträge), die das Simulationssystem

Durchlauf	Simulationszeit	Anzahl Gabelstapler (GS)	durchschnittl. Wartezeit pro GS [Sek]
1	00:18:08	2	120
2	00:13:56	3	159
3	00:11:27	4	173
4	00:10:07	5	202
5	00:10:07	6	252

Tabelle 4.2: Auswertung Anwendungsbeispiel

automatisch generiert und verteilt, im Vergleich zu dem Lieferangebot (Anzahl der Gabelstapler) bei einer Anzahl von fünf Gabelstaplern gesättigt ist. Dies bestätigt auch die durchschnittliche Wartezeit pro Gabelstapler. Denn bei einer Anzahl von sechs Gabelstaplern im System, ist ein deutlicher Anstieg der durchschnittlichen Wartezeit zu erkennen.

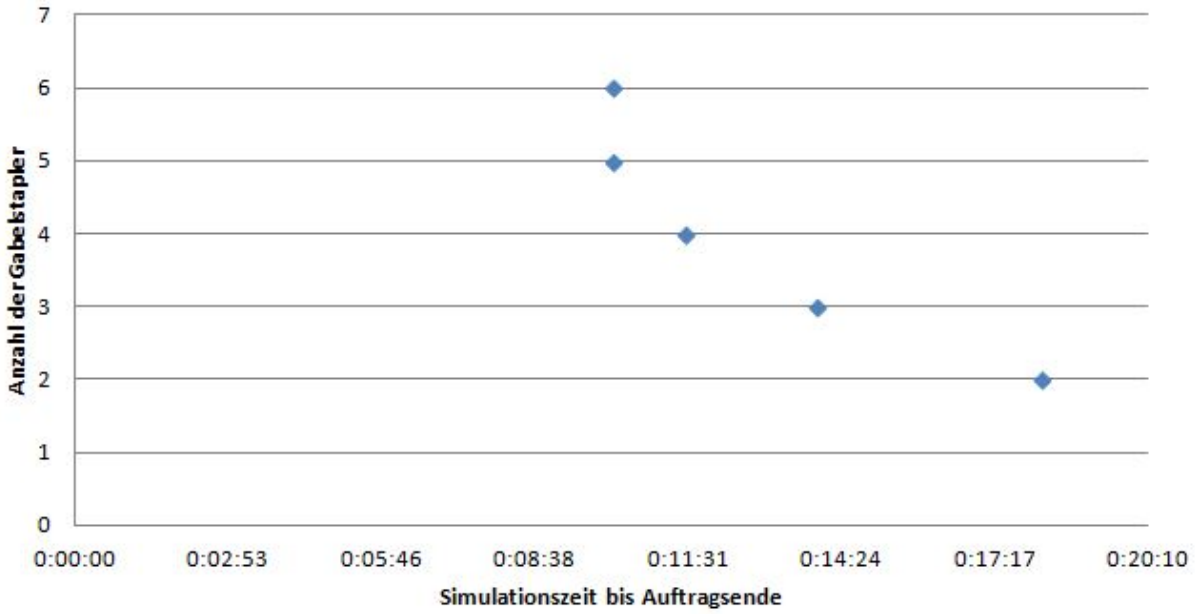


Abbildung 4.16: Simulationszeit abhängig von Anzahl der Gabelstapler

5 Zusammenfassung und Ausblick

In dieser Arbeit soll eine 3D-Simulationssoftware eines Produktionssystems mithilfe einer Gameengine und KI realisiert werden. Dieses letzte Kapitel fasst die Ergebnisse dieser Arbeit zusammen (Abschnitt 5.1) und gibt einen Ausblick auf weiterführende Arbeiten an dem Simulationssystem (siehe Abschnitt 5.2).

5.1 Zusammenfassung der Ergebnisse

Die in Abschnitt 3.2 gestellten Anforderungen können als erfüllt betrachtet werden. In der Simulation ist es möglich Parameter von Produktionseinheiten und logistischen Einheiten zu verändern, sowie diese im Raum zu verschieben. Während des Betriebes versorgen die logistischen Einheiten automatisch, mithilfe der KI-Komponente, die Produktionseinheiten. Das GUI-System gibt wichtige Kennzahlen einer ausgewählten Produktionseinheit oder logistischen Einheit wieder und bietet Änderungsmöglichkeiten des Simulationssystems an.
Es hat sich im Laufe dieser Arbeit gezeigt, dass ein 3D-Simulationssystem mithilfe einer Gameengine schnell zu realisieren ist und dadurch die Entwicklungskosten gering sind. Ein Anwendungsbeispiel verdeutlicht die benutzerfreundliche Anwendung und hohe grafische Qualität. Aus diesen Gründen bietet sich ein solches System gerade für kleinere und mittlere Unternehmen an. Diese beklagen nach [41], dass die Softwarelösungen im Bereich der digitalen Fabrik zu teuer und nicht auf ihre Bedürfnisse abgestimmt sind.

5.2 Ausblick

Im Rahmen der Arbeit wurde die grundlegende Funktionsweise der 3D-Simulationssoftware implementiert. Ein wichtiger Teilschritt, die Validierung und Verifikation muss noch erfolgen. Hierbei geht es darum, dass System zu testen um dieses als vertrauenswürdiges Simulationssystem zu deklarieren.
Das Simulationssystem bietet, neben den bereits erfüllten Anforderungen, weitere Potentiale bzw. Erweiterungen, die im Folgenden dargestellt sind:

Freie Systemgestaltung: Es ist denkbar, dass der Anwender über eine Importfunktion eigene 3D-Komponenten einfügt und platziert. Somit wäre die Möglichkeit gegeben ein eigenes Fabrikmodell einzufügen. Durch das Hinzufügen von vorkonfigurierten Produktionskomponenten über das Menü, entsteht anschließend das Produktionssystem. Auch hier besteht die Möglichkeit, dass optische Erscheinungsbild durch eigene 3D-Komponenten zu verändern. Durch Individualisierungsmöglichkeiten über Variablen der Produktionskomponenten ist ein völlig freies und eigenes Simulationsmodell erzeugbar.

Animationen erweitern: Am Lagerausgang bzw. -eingang der Produktionseinheit wird, mithilfe einer Animation, Material auf- bzw. abgebaut. Die logistische Einheit bewegt dieses Material, in dem es ihr als 3D-Komponente hinzugefügt wird, nun tatsächlich zur nachgelagerten Produktionseinheit oder Lagereinheit. Animationen von Arbeitern verstärken den Effekt eines realistischen Erscheinungsbilds der 3D-Simulationssoftware.

Fertigungsinsel implementieren: Eine wichtige Funktion ist es, den Produktionseinheiten die Eigenschaften einer Fertigungsinsel, wie z.B. die Fertigung von Varianten, zu implementieren um realistische Produktionssystem abbilden zu können.

Lagersimulation implementieren: Es besteht die Möglichkeit, die Ware im Eingangslager manuell einzugeben oder, auf Basis von Prognosedaten, simulieren zu lassen. Ebenso kann der Bedarf an Fertigprodukten eingegeben oder, ebenfalls auf Basis von Prognosedaten, über eine Simulation gesteuert werden.

Abbildung verschiedener Produktionssteuerungsarten: Umschalten zwischen Pull-System und Push-System. Dieses wäre besonders in der Lehre hilfreich, um die Verfahren besser zu verdeutlichen.

Speichern und Laden von Szenarien: Durch Speichern und Laden von Szenarien ist ein Optimierungsprozess einfacher. Hierzu muss die Möglichkeit gegeben sein, dass veränderte Produktionssystem als Szenario zu speichern, bzw. ein bereits gespeichertes Szenario zu laden.

Exportfuntkionen implementieren: Durch eine Exportfunktion sind wichtige Kennzahlen in ein Tabellenkalkulationsprogramm übertragbar. Dieses erleichtert die Auswertung der Simulationsergebnisse.

A Anhang

A.1 CD mit Simulationssoftware

Literaturverzeichnis

[1] KUNICKI, Matthias: *Ziele und Nutzen der Digitalen Fabrik - aus Sicht der Wissenschaft*. `http://wiki.zimt.uni-siegen.de/fertigungsautomatisierung/index.php/Ziele_und_Nutzen_der_Digitalen_Fabrik_-_aus_Sicht_der_Wissenschaft`. Version: Dez 2010, Abruf: 3.11.2013

[2] Verein Deutscher Ingenieure (VDI): *Digitale Fabrik – Grundlagen 4499 Blatt 1*. Feb 2008

[3] ELEY, Michael: *Simulation in der Logistik*. 2012

[4] Verband der Automobilindustrie (VDA): *Ausprägung der Logistiksimulationen - 4810 T x1*. März 2011

[5] ANDERSON, Eike F. ; ENGEL, Steffen ; COMNINOS, Peter ; MCLOUGHLIN, Leigh: The case for research in game engine architecture. In: *Proceedings of the 2008 Conference on Future Play: Research, Play, Share* ACM, 2008

[6] SHERROD, Allen: *Ultimate 3D Game Engine Design & Architecture*. Charles River Media, Inc., 2007

[7] LEWIS, Michael ; JACOBSON, Jeffrey: Game Engines. In: *Communications of the ACM* 45 (2002), Nr. 1, S. 27–31

[8] GREGORY, Jason ; LANDER, Jeff ; WHITING, Matt: *Game engine architecture*. AK Peters, 2009

[9] THORN, Alan: *Game engine design and implementation*. Jones & Bartlett Publishers, 2011

[10] MEISINGER, Gerold: Game-Engine-Architektur mit funktional-reaktiver Programmierung in Haskell/Yampa. (2010)

[11] BRILL, Manfred: *Virtuelle Realität*. Springer DE, 2009

[12] HEISE: *Videobrille Oculus Rift im Test*. http://www.heise.de/newsticker/meldung/Uebelkeit-mit-Begeisterung-Videobrille-Oculus-Rift-im-Test-1834647.html. Version: März 2013, Abruf: 30.10.2013

[13] MICHAEL, Wieczorek ; IHLENFELD, Jens: *Seekrank in der Golem.de-Redaktion*. http://www.golem.de/news/test-oculus-rift-dev-glotz-wuerg-freu-1304-98885-4.html. Version: April 2013, Abruf: 30.10.2013

[14] RIFT.DE oculus: *Oculus Rift*. http://oculus-rift.de/, Abruf: 30.10.2013

[15] GAMESTAR.DE: *nVidia 3D Vision 2 im Test*. http://www.gamestar.de/hardware/grafikkarten/nvidia-3d-vision-2/test/nvidia_3d_vision_2,379,2563994.html, Abruf: 30.10.2013

[16] NVIDIA.DE: *3D-Vision*. http://www.nvidia.de/object/3d-vision-technology-de.html, Abruf: 30.10.2013

[17] GAUTZSCH, Steffen: *3D-Kino im Wohnzimmer: Analyse einer Unterhaltungs-Revolution durch Stereoskopie*. Diplomica Verlag, 2012

[18] IZADI, Shahram ; KIM, David ; HILLIGES, Otmar ; MOLYNEAUX, David ; NEWCOMBE, Richard ; KOHLI, Pushmeet ; SHOTTON, Jamie ; HODGES, Steve ; FREEMAN, Dustin ; DAVISON, Andrew u. a.: KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: *Proceedings of the 24th annual ACM symposium on User interface software and technology* ACM, 2011, S. 559–568

[19] MCCARTHY, John: What is artificial intelligence. In: *URL: http://www.formal.stanford.edu/jmc/* (2007)

[20] ZÖLLER-GREER, Peter: *Künstliche Intelligenz: Grundlagen und Anwendungen*. BoD–Books on Demand, 2010

[21] KASTENHOLZ, Daniel: *3D Pathfinding*. Juni 2006

[22] HÜLSMANN, Michael: *Entwurf und Implementierung eines Routing-Algorithmus und eines zugrundeliegenden Datenmodells am Beispiel des Strassennetzes*. GRIN Verlag, 2008

[23] STEINKE, Lennart: *Spieleprogrammierung*. Hüthig Jehle Rehm, 2008

[24] BARTH, Heiko: Produktionssysteme im Fokus. In: *wt Werkstattstechnik online* 95 (2005), Nr. 4, S. 269–274

[25] KURBEL, Karl: *Produktionsplanung und -steuerung im Enterprise Resource Planning und Supply Chain Management*. Oldenbourg Verlag, 2005

[26] GÜNTHER, Hans-Otto ; TEMPELMEIER, Horst: *Produktion und Logistik*. Springer DE, 2011

[27] SPUR, Günter ; STÖFERLE, Theodor: *Handbuch der Fertigungstechnik: Fügen, Handhaben und Montieren*. Hanser Verlag, 1994

[28] BOSSEL, Hartmut: *Systeme, Dynamik, Simulation*. BoD–Books on Demand, 2004

[29] MÄRZ, Lothar: *Simulation und Optimierung in Produktion und Logistik: praxisorientierter Leitfaden mit Fallbeispielen*. Springer DE, 2011

[30] MATTERN, Friedemann: Modellbildung und Simulation. In: *Perspektiven der Informatik*, 1993, S. 56–64

[31] KÜHN, Wolfgang: *Digitale Fabrik: Fabriksimulation für Produktionsplaner*. Hanser Verlag, 2006

[32] BAYER, Johann ; COLLISI, Thomas ; WENZEL, Sigrid: *Simulation in der Automobilproduktion*. Springer DE, 2003

[33] ARNOLD, Ingo ; CHUGHTAI, Arif ; IHLER, Edmund ; KEHRER, Timo ; MEHLIG, Uwe ; ZDUN, Uwe: *Software-Architektur*. Springer DE, 2008

[34] HEIDE, Balzert: UML 2 in 5 Tagen, Der schnelle Einstieg in die Objektorientierung, 2. In: *Auflage, W3L-Verlag Bochum* (2009)

[35] UNITY3D: *Unity3D Documentation*. `http://docs.unity3d.com/Documentation/Manual/index.html`. Version: November 2012, Abruf: 30.10.2013

[36] BLENDER.ORG: *Introduction to Game Engine*. `http://wiki.blender.org/index.php/Doc:2.6/Manual/Game_Engine`, Abruf: 31.10.2013

[37] KHRONOS.ORG: *OpenGL ES 2.0 for the Web*. `http://www.khronos.org/webgl/`, Abruf: 25.10.2013

[38] STEINBERG, Monika: *3D Web: Standards, Plugins und Frameworks auf einen Blickt*. `http://t3n.de/news/3d-web-standards-plugins-356877/`, Abruf: 31.10.2013

[39] USCRIPT: *uScript Documentation*. `http://uscript.net/docs/index.php?title=Main_Page`. Version: May 2012, Abruf: 30.10.2013

[40] OCULUSVR: *Oculus Developer Forums*. https://developer.oculusvr.com/forums/viewforum.php?f=27, Abruf: 31.10.2013

[41] CORBAN, Michael: *Die Digitale Fabrik ist keine IT-Spinnerei mehr*. http://www.industrieanzeiger.de/home/-/article/12503/28797917/Die-Digitale-Fabrik-ist-keine-IT-Spinnerei-mehr/art_co_INSTANCE_0000/maximized/, Abruf: 6.11.2013

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Adler

Vorname: Eike

dass ich die vorliegende Bachelorarbeit bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Eine Software zur 3D-Simulation von Produktionssystemen - basierend auf einer Gameengine und KI

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der -bitte auswählen- ist erfolgt durch:

Ort	Datum	Unterschrift im Original