SIMULATION ANIMIERTER MASSENSZENEN IN EINER PROZEDURAL ERSTELLTEN UMGEBUNG

Stefan Köpke

Thomas Brandstetter

Jörg Grubmüller

Fachhochschulstudiengang Telekommunikation und Medien

St. Pölten, Juni 2010

Prozedurale Modellierung von CG Film Sets und Character Design für Massensimulationen

Stefan Köpke

Erstbetreuer: Mag. Franz Schubert
Zweitbegutachter: Dipl.-Ing. Mario Zeller

Erfassung, Erstellung und Adaption von Bewegungsdaten für Massensimulationsanwendungen

Thomas Brandstetter

Erstbetreuer: Dipl.-Ing. Mario Zeller
Zweitbegutachter: Mag. Franz Schubert

Massensimulation mit Massive Prime

Jörg Grubmüller

Erstbetreuer: Dipl.-Ing. Mario Zeller
Zweitbegutachter: Mag. Franz Schubert

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.
- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter oder einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Diese Arbeit stimmt mit der von den Begutachtern beurteilten Arbeit überein.

Ort, Datum	Stefan Köpke
Ort, Datum	Thomas Brandstetter
Ort, Datum	Jörg Grubmüller

Zusammenfassung

Diese Diplomarbeit beschäftigt sich mit der filmischen Inszenierung einer computeranimierten Massensimulation in einer prozedural erstellten, urbanen Umgebung. Der Workflow und etwaige Probleme bei der Umsetzung, werden anhand einer Fallstudie, in Form eines animierten Kurzfilms, untersucht.

Die für dieses Projekt benötigten Arbeitsbereiche werden in drei Kerngebiete untergliedert, welche gleichzeitig die Themenschwerpunkte der Diplomanden in dieser kooperativen Facharbeit darstellen.

- Prozedurale Modelierung von CG Film Sets und Chracterdesign für die Anwendung in Massensimulationen
- Erfassung, Erstellung und Adaption von Bewegungsdaten für Massensimulationsanwendungen
- Massensimulation mit Massive Prime

Die Kommunikationfähigkeit zwischen verschiedenen Softwarepaketen in der Produktionspipeline, sowie eine detaillierte Analyse der benötigten Arbeitsschritte während der Umsetzungsphase des Projekts bilden die Kernelemente dieser Arbeit.

Abstract

This master thesis focuses on a cinematic presentation of a computer animated crowd simulation in a procedural, urban environment. The workflow and possible problems during the realization are researched on the basis of a case study in the form of an animated short. The different fields of work are subdivided into three core areas, which represent the main emphasis of the cooperative thesis.

- Procedural modeling of CG film sets and character design for crowd simulations
- Capture, creation and adaption of motion for crowd simulations
- Crowd simulation with Massive Prime

The ability of communication between different software packages within the production pipeline, as well as a detailed analysis of the required tasks during the realization process present the core elements of this thesis.

Danksagung

Hiermit möchten wir uns bei folgenden Personen für ihre Unterstützung bedanken:

Johannes Bernert, Ingrid und Gernot Brandstetter, Michi Brandstetter, Brigitte und Wolf-Dietrich Grubmüller, Ilse und Gerhard Köpke, Pascal Müller, Melanie Pichler, Bettina Polonay, Franz Schubert, Jiro Shimizu, Dominik Tarolli, Mario Zeller

Inhaltsverzeichnis

1 Einleitung

Seit Anbeginn der Filmgeschichte wird die gezielte Inszenierung von Massen als filmisches Stilelement genutzt um imposante Bilder zu vermitteln. Die Technologie hinter der Produktion von Massenszenen hat seit dem Filmklassiker „The birth of a Nation“ aus dem Jahr 1915 einige Stadien durchlaufen. Wurde in den Anfängen der Filmindustrie noch eine unüberschaubare Anzahl an Statisten eingesetzt, kommt bei neuen Produktionen vermehrt die Simulation von computergenerierten Akteuren zum Einsatz. Die Inszenierung von Massenszenen mit Hilfe von menschlichen Statisten erfordert einen immensen Aufwand an Zeit und finanziellen Mitteln für die Ausstattung, Verpflegung und Koordinierung der Akteure. Die computergestützte Massensimulation hingegen ermöglicht einen wesentlich flexibleren Handlungsspielraum und erleichtert die Koordinierung einzelner Akteure in der Masse. Weiters wird durch die computergenerierte Darstellung von Massen der logistische und finanzielle Aufwand reduziert.

Diese Technologie bildet das Fundament für folgende Arbeit und wird anhand einer Fallstudie durchleuchtet und analysiert. Im Zuge der projektspezifischen Umsetzung wird eine Produktionspipeline für die Erstellung von computeranimierten Massenszenen in einer prozedural erstellten urbanen Umgebung erarbeitet.

1.1 Ausgangssituation

Anhand der Umsetzung eines Kurzfilmes, basierend auf der Technologie der computergestützten Massensimulation, werden projektimmanente Arbeitsabläufe analysiert und dokumentiert. Im Hinblick auf die Erstellung der handlungstragenden Akteure, sowie der Umgebung wird besonderes Augenmerk auf die Integration der Massensimulationssoftware Massive Prime in die Produktionspipeline gelegt. Eine mittels künstlicher Intelligenz gesteuerte Masse stellt besondere Anforderungen an die Modellierung und Animation der virtuellen Akteure. Ziel der projektspezifischen Arbeit ist die Erarbeitung eines Workflows, sowie die Definition von softwareübergreifenden Schnittstellen und Dateiformaten um den reibungslosen Datenfluss in der Umsetzungsphase der Fallstudie zu gewährleisten. Etwaige Probleme und Komplikationen während der Realisierung, sowie spezifische Lösungsansätze bilden weitere Kernelemente der Arbeit.

1.2 Forschungsfragen

Im Zuge der Fallstudie sollen folgende Problemstellungen und Fragen analysiert und dokumentiert werden.

- Welche Anforderungen stellt die Realisierung von Massenszenen und die Produktion des prozeduralen Film Sets an Soft- und Hardware?

- Wie erfolgt der Datenfluss zwischen den einzelnen Softwarepaketen innerhalb der Produktionspipeline?

- Welche Anforderungen stellt die Simulation von virtuellen Akteuren in der Masse an die Produktion von Geometrien, Texturen und Animationsdaten?

- Welche Probleme treten während der Umsetzung der Fallstudie auf, und welche Lösungsansätze können dafür gefunden werden?

1.3 Aufbau

Die für die Realisierung der Fallstudie benötigten Arbeitsbereiche werden in drei Kerngebiete untergliedert, welche die Grundstuktur der Diplomarbeit definieren.

Kapitel 3 erklärt die stadtplanerischen Grundlagen bei der Konzeption des Stadtmodells und begründet die Designentscheidungen, die im Zuge der Gestaltung des urbanen Raumes getroffen werden. Dabei wird ein Bezug zwischen Form und Erscheinungsbild der Stadt und den Anforderungen der filmischen Handlung hergestellt.

In Kapitel 4 werden die Grundlagen des prozeduralen Modelings von urbanen Environments thematisiert und auf die Softwarelösung CityEngine von Procedural Inc. eingegangen - die grundlegende Arbeitsweise des Tools steht hierbei im Vordergrund. In Folge dessen werden die Charakteristika von L-Systemen erklärt, welche die Basis für die Arbeisweise der CityEngine darstellen. Weiters werden wesentliche Grundelemente der integralen, internen Abläufe in der CityEngine beschrieben. Ein weiterer Abschnitt gibt einen Einblick in die proprietäre Formbeschreibungssprache der CityEngine - die CGA Shape Grammar - mit der es möglich ist, immense Mengen an Gebäuden auf L-System Basis zu generieren.

Kapitel 5 befasst sich mit der praktischen Umsetzung der beschriebenen, theoretischen Grundlagen im Rahmen der Fallstudie. Dies umfasst die Generierung des Straßennetzes, der Grundstücksflächen sowie der Grundshapes der Straßen. Des Weiteren wird auf Probleme, die sich im Zuge der Fallstudie ergeben, eingegangen und es werden mögliche Lösungswege erklärt - dazu gehört die manuelle Fertigung von Teilen der Stadt, die sich nicht auf prozeduralem Weg entwickeln lassen. Ein weiterer Themenpunkt dieses Kapitels ist die Umsetzung der Gebäudeentwicklung mittels CGA-Shape-Grammar, wobei anhand eines Beispiels die Funktion des Regelwerks hinter der prozeduralen Gebäudegenerierung geschildert wird. Als letzten Punkt behandelt das Kapitel den Export und Transfer der erzeugten Geometrie aus der CityEngine nach Autodesk Softimage und beschreibt die Zusammenführung der verschiedenen Elemente des Stadtmodells.

Kapitel 6 behandelt die Integration von urbaner Vegetation in das Stadtmodell mit Hilfe der Software E-ON Vue. Im Zuge dessen wird die Generierung und von Ökosystemen erläutert und deren Eingliederung in eine bestehende 3D-Szene beschrieben.

In Kapitel 7 wird auf das Design der Charaktere des Kurzfilms eingegangen. Dies beinhaltet das Character Design, die Modellierung sowie die Texturierung der Charaktere in Bezug auf die Story des Kurzfilms. Des Weitern stehen die Anforderungen im Vordergrund, die von der Verwendung der Modelle als Crowd Agents innerhalb einer Massensimulation an das Character Design gestellt werden.

In Kapitel 8 werden die grundlegenden Elemente eines Character Setups beziehungsweise die erfordelrichen Voraussetzungen für eine flexible Character Animation erläutert - Rigging. Dies umfasst den hierarchischen Aufbau einer characterspezifischen Skelett Struktur, sowie ein Aufsetzen aller benötigter Relationen, die für eine zielgerichtete Keyframe Animation notwendig sind. Weiters muss das Rig bestimmten Anforderungen entsprechen, die essentiell für einen fehlerfreien Transfer von verschiedenen Motion Capture Daten sind.

Aufbauend auf das fertige Character Setup werden in Kapitel 9 verschiedene Möglichkeiten aufgezeigt, Animationsdaten zu erstellen beziehungsweise für den Einsatz in einer Massensimulation aufzubereiten. Keyframe Animation, Motion Capturing und ein prozeduraler Ansatz bilden die Kernelemente dieses Abschnitts der Arbeit. Eine Dokumentation komplexer Vorgänge im Rahmen der Erfassung, Erstellung und Adaption von Bewegungsdaten soll die für eine erfolgreiche Umsetzung nötigen Arbeitsprozesse verdeutlichen. Alle Erläuterungen der Arbeitsschritte werden in Anlehnung an die weitere Verwendung in der Massensimulationssoftware Massive Prime behandelt.

In Kapitel 10 wird anschließend die Realisierung der Massenszenen dokumentiert. Der programminterne Workflow von Massive Prime, der grundlegende Umgang mit der Simulationssoftware, sowie Schnittstellen und Integration von Animations- und Geometriedaten werden durchleuchtet. In weiterer Folge werden alle benötigten Arbeitsschritte zur Erstellung komplexer Crowd Agents erläutert.

Der Import von Rigs, die Integration von Geometrien, die Konzeption von Motion Trees und die Sinnesorgane der Agents werden in diesem Teil der Arbeit behandelt. Ein weiterer Abschnitt gibt einen Einblick in die Implementierung der Gruppendynamik durch die Erschaffung von künstlicher Intelligenz. Hier wird die Steuerung der Crowd Agents anhand von beispielhaften Brain Modulen im Zuge der Fallstudie dargelegt. Weiters werden Techniken zur Individualisierung einzelner Crowd Agents geschildert, um eine heterogene, realistische Masse zu erschaffen. Abschließend wird auf die Simulation, sowie das Rendern der Massensimulation eingegangen.

Für die cineastische Inszenierung müssen virtuelle Filmsets und Kamerafahrten in einer 3D-Software erstellt und kombiniert werden. Kapitel 11 verschafft einen groben Überblick über den während der Produktionsphase verfolgten Workflow.

1.4 Zielgruppe

Die Zielgruppe dieser Arbeit setzt sich aus filmschaffenden und 3D-affinen Lesern zusammen, die mit den Grundlagen der computergenerated Images (CGI) bewandert sind. Aus diesem Grund wird auf eine detaillierte Beschreibung vieler Grundfunktionen im 3D Bereich verzichtet. Weiters soll diese Arbeit als Leitfaden für die Erstellung einer cineastisch inszenierten Massensimulation mit Massive Prime dienen.

2 Versuchsaufbau

Bei der Fallstudie handelt es sich um einen dreiminütigen, animierten Kurzfilm, der alle Kriterien für die Implementierung einer computergenerierten Massensimulation erfüllt. Für diesen Zweck wird ein prozedurales Environment kreiert, welches den Handlungsspielraum der virtuellen Akteure definiert. Um den Eindruck einer Massensimulation auf den Zuseher zu verstärken, werden die entsprechenden Szenen in einen cineastischen Spannungsbogen eingebettet, der den Versuch der Änderung eines festgefahrenen Wertesystems in einer fiktiven Welt beschreibt.

2.1 Story

Autonomie und Individualismus werden im Keim erstickt. Effizienz und Produktivität stehen im Mittelpunkt einer neuen Gesellschaft. Der global agierende Konzern GLOBOTEX treibt die Unterjochung der Bevölkerung mit eiserner Faust voran. Das tägliche Leben ist geprägt von gesellschaftlicher Normierung durch bedingungslose Integration in das System.

Der aufgezwungene Konformismus führt zu der Existenz einer Subgesellschaft, die sich den Zwängen des Monopolisten nicht unterwerfen will. Als sich die Fronten verhärten, findet der Unmut der Gesellschaft Ausdruck in einer Demonstration vor dem Headquarter.

2.2 Anforderungen

Im Gegensatz zur Animation von einzelnen Charakteren fordert die Konzeption von Massensimulationen einen effizienten Arbeitsansatz zur Individualisierung der einzelnen Akteure auf verschiedensten Ebenen (vgl. Thalmann et al. 2007, S. 2). Die Variation des äußeren Erscheinungsbildes, sowie der Bewegungen und Bewegungsabläufe steht in diesem Zusammenhang im Vordergrund. Im Rahmen der Fallstudie soll dieser Ansatz verwendet werden, um zwei gegensätzliche, kontradiktorische Massen zu visualisieren. Eine aus Banker Bots bestehende homogene Masse, ohne jegliche Individualität einzelner Akteure, bildet den Gegensatz zur heterogenen Masse aus Bass Bot und Treble Bot Agenten. Diese Darstellung von Protagonisten und Antagonisten als Masse wird durch den fehlenden oder gezielten Einsatz von Variationen verstärkt.

Die Inszenierung von großen Massenszenen stellt in weiterer Folge hohe Ansprüche an Hard- und Software. Die Simulation und der Rendervorgang einer Vielzahl an Akteuren ist mit herkömmlichen 3D Softwarepaketen nur schwer zu bewerkstelligen. Die Massensimulationssoftware Massive Prime bietet auf diesem Gebiet Lösungsansätze für die Implementierung der Gruppendynamik und die recourcenschonende Berechnung der Simulation. Um realistische Massen zu erschaffen, bedient sich Massive Prime der Unscharfen Logik (Fuzzy Logic) zur Abbildung von menschenähnlichen Verhaltensmustern auf Basis der künstlichen Intelligenz. Die Integration von Massive Prime in eine zielgerichtete Produktionspipeline erfordert jedoch die Verwendung von Autodesk Maya, als Bindeglied zwischen Massive und weiteren 3D Softwarepaketen. Aus diesem Grund ist die Erfassung der Abhängigkeiten, sowie die Analyse der Schnittstellen und Dateiformate von Nöten, um einen reibungslosen Datenfluss zwischen allen benötigten Softwarelösungen sicherzustellen.

In Bezug auf die visuelle - und folglich auch funktionelle - Umsetzung der in dieser Arbeit thematisierten Fallstudie stellen sich verschiedene Anforderungen, die sich in zwei Hauptbereiche untergliedern lassen. Diese beiden Arbeitsbereiche werden in Folge der Arbeit nacheinander durchgearbeitet.

Einer der Hauptpunkte dieser Arbeit ist, jene Arbeitsweisen herauszufinden, mit deren Hilfe sich eine urbane Umgebung als virtuelles Bühnenbild beziehungsweise als digitaler Szenenaufbau auf prozeduralem Weg generieren und in einen filmischen Kontext stellen lässt. Dabei spielen sowohl formal gestalterische Grundlagen als auch die technische Umsetzbarkeit des Städtedesigns für eine Integration einer Massensimulation von hunderten 3D-Charakteren eine elementare Rolle. Das Ergebnis dieser Arbeit soll eine fertige, funktionsfähige Working Pipeline beschreiben, die diese gestalterischen und technischen Anforderungen erfüllt.

In weiterer Folge soll beschrieben werden, durch welche visuellen Gestaltungsmittel oder logistischen Arbeitsmethoden die gestellten Aufgaben bewältigt werden können, beziehungsweise mit Hilfe welcher Schnittstellen oder Datenformate es möglich ist, diese Anforderungen zu erfüllen.

Als zweiter Hauptpunkt dieser Arbeit soll eruiert werden, welche Anforderungen an das Design von 3D-Charakteren gestellt werden, die sowohl als eigenständige handlungstragende High-Quality-Modelle im Kontext der filmischen Handlung bestehen sollen, als durch ihre technischen und gestalterischen Voraussetzungen auch für die Anwendung innerhalb einer Massensimulation geeignet sein sollen. Im Zuge dessen sollen die Arbeitsschritte und Abfolgen im Designprozess beschrieben werden, durch die diese Problemstellungen zufriedenstellend gelöst werden können.

Aufbauend auf die im Designprozess kreierten handlungstragenden Charaktere werden Arbeitschritte definiert und analysiert, in welchen spezifische Bewegungsabläufe erstellt werden, die für eine Simulation von Massenszenen notwendig sind. Für die realistische Darstellung einer heterogen fortschreitenden Masse werden verschiedene variantenreiche Animationen benötigt, die jederzeit abrufbar, kombinierbar und zum Teil auch wiederholbar sein müssen.

Unterschiedliche Animationstechniken kommen zum Einsatz, um den quantitativen Anforderungen bezüglich der Generierung von Bewegungsdaten gerecht zu werden. Weiters ist es essentiell Übergangspunkte zwischen bestimmten Animationen zu

definieren und diese während der Produktionsphase zu berücksichtigen. Um den Einsatz unterschiedlicher Softwarelösungen zu ermöglichen muss die Aufbereitung des Bewegungsapparates für jeden Character so flexibel wie möglich gestaltet werden. Non-lineare Keyframe Animation, prozedurale Animation von Motion Loops und der Einsatz von Motion Capture Techniken bilden die Grundlage für die Generierung der benötigten Bewegungsdaten. Um die jeweiligen Vorteile unterschiedlicher Animationstechniken und der dafür vorgesehenen Programme zur Gänze ausnutzen zu können, müssen Schnittestellen und Datenformate definiert werden, die eine reibungslose und fehlerfreie Kommunikation unter einander gewährleisten.

Die Erstellung einer effektiven Pipeline für die Integration der benötigten Daten in die Massensimulationssoftware Massive Prime stellt eine weiter besondere Herausforderung in der Produktionsphase dar, da die Kommunikationsmöglichkeiten durch den forcierten Gebrauch von Autodesk Maya beschränkt sind.

2.3 Technologie

Die Umsetzung der Fallstudie setzte den Einsatz zahlreicher Softwarepakete voraus und stellte dabei hohe Anforderungen an die eingesetzte Hardware. Folgende Soft- und Hardwarelösungen kommen im Zuge der Realisierung des Projekts zum Einsatz:

2.3.1 Software

Hersteller	Software Paket	Version	Verwendungszweck
Procedural	CityEngine	v2008, v2009	Setbau, Generierung des urbanen Environments
E-ON	Vue	v8.1	Generierung urbaner Ökosysteme
Autodesk	Softimage	v7.5, v2010	Character Modeling, Erstellung von Objekten (Assets),Szenenaufbau, Character Setup und Animation, Camera-Setup, Rendering
Autodesk	Maya	v2009	Datentransfer-Platform für Massive Prime, Character Setup
Autodesk	3ds Max	v2010	Generierung von Objekten (Assets), prozedurale Gebäudemodellierung mit Greeble (Plugin), Erstellung prozeduraler Animationszyklen mit CAT (Plugin)
Adobe	Photoshop	vCS4	Erstellung von Texturen
Adobe	Illustrator	vCS4	Erstellung von Texturen
Apple	Final Cut	v6.0.4	Synchronisierung und Verwertung des Motion Capture Footage
Pixelfarm	PFTrack	v5.0	Optische Entzerrung des Motion Capture Footage
RealViz	Movimento	v1.0	Kalibrierung des Motion Capture Kamera Setups, Motion Tracking

Autodesk	MotionBuilder	v7.5	Keyframe Animation, Aufbereitung von Bewegungsdaten (.BVH), Verwertung optischer Markerdaten (.TRC)
Massive Software	Massive Prime	v3.5.5	Crowd Simulation
DNA Research	3Delight Render-man	v9.0	Crowd Rendering
Mental Images	Mental Ray	v3.7	Scene Rendering
Adobe	After Effects	vCS4	Erstellung animierter Texturen und Compositing

Tabelle 1: Verwendete Hardware

2.3.2 Hardware

Notebook LG R410
System: Intel Core2Duo, Windows Vista 32 Bit, 3GB RAM
Anwendungsgebiet: Crowd Simulation, 3ds Max, Asset Produktion.

Desktop PC
System: Intel Core2 Quad CPU, 2.40GHz
Windows Vista 64 Bit, 4 GB
Anwendungsgebiet: Motion Tracking, Character Setup, Animation, Asset Produktion, Scene Setup, Camera Setup.

Desktop PC
Apple Mac Pro 2008
System: 8 Core (2 x Intel XEON 2,8 GHz), Windows 7 64 Bit (Bootcamp) / Mac OSX 10.6, 16 GB RAM
Anwendungsgebiet: Prozedurales Modeling und Texturing der urbanen Umgebung, Modeling und Texturing der Character, Asset Produktion, Generierung von Ökosystemen, Aufbau der finalen Szenerie, Integration Animierter Charactere, Rendering mit Mental Ray.

2.4 Pipeline

Um den reibungslosen Ablauf der Umsetzungsphase des Prejekts zu gewährleisten, ist es essentiell, Datenflüsse und Schnittstellen zwischen allen Softwarepaketen vorab genau zu definieren. Während dieses Prozesses werden zahlreiche Testzyklen an Referenzobjekten durchgeführt um Dateiformate und Parameter der zu übertragenden Daten zwischen Softwarelösungen zu bestimmen.

Aufgrund der eingeschränkten Kommunikationsfähigkeit zwischen einzelnen Programmen ergibt sich in vielen Fällen die Notwendigkeit weitere Softwarepakete für die Konvertierung der Daten einzusetzen. Ein weiterer entscheidender Grund für die Integration bestimmter Softwarelösungen ergibt sich aus der Möglichkeit spezielle programmspezifische Tools einer bestimmten Softwarelösung verwenden zu können. Aus dieser empierischen Arbeit resultiert zuletzt die Produktionspipeline, welche die Basis für die Umsetzung des Projekts darstellt. Eine schematische Darstellung der Produktionspipeline ist auf der nächsten Seite in Abbildung 1 (S. 22) ersichtlich.

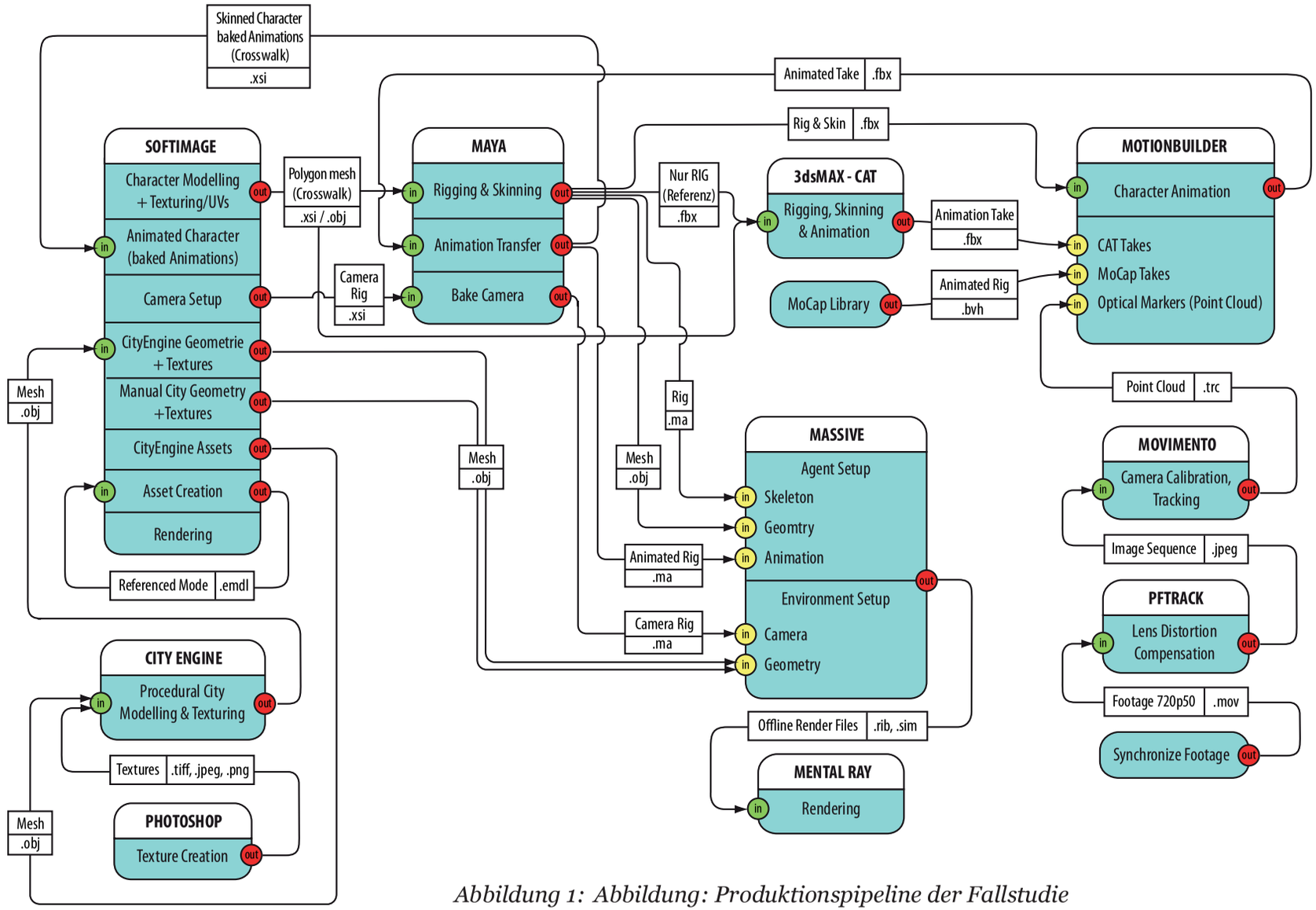


Abbildung 1: Abbildung: Produktionspipeline der Fallstudie

3 Szenerie

Seit Anbeginn des Filmschaffens sind Großstädte und Metropolen ein integrales filmisches Element und fungieren in vielen Produktionen als zentraler Ort der Handlung. Im gestalterischen Umgang mit einem urbanen Film Set haben sich über die Jahrzehnte zahlreiche gestalterische Konventionen etabliert, die sich in vielen Filmen wiederfinden; hierzu zählen beispielsweise bestimmte, charakteristische Bildausschnitte, der gezielte Einsatz räumlicher Tiefe insbesondere das Spiel mit perspektivischer Flucht und räumlicher Tiefenunschärfe, sowie Kamerafahrten und Luftaufnahmen. Das Auge des Zuschauers hat sich mittlerweile an diese bildgestalterischen Konventionen gewöhnt.

Bei der Produktion eines animierten Kurzfilmes ist es daher für die cineastische Wirkung von Vorteil, sich bewährter Techniken zu bedienen, auf die Sehgewohnheiten der Rezipienten Rücksicht zu nehmen und das städtische Ambiente auf eine Art und Weise in Szene zu setzen, die sich an klassischen Filmwerken orientiert.

Im Gegensatz zu konventionellen Filmaufnahmen die tatsächlich existierende Städte als Kulisse nutzen, bietet sich bei der Planung von urbanen 3D-Environments die Möglichkeit, eine Stadt von Anfang an so zu planen, dass sie einerseits den typischen, morphologischen sowie topologischen Eigenschaften einer Stadt entspricht, und andererseits durch zielgerichtetes Design den cineastischen Anforderungen des Films angepasst werden kann. Auf die Anforderungen und Designgrundlagen bei der Konzeption und Entwicklung eines fiktiven computergenerierten Stadtmodells für die Verwendung als 3D-Film-Set wird in den nächsten Abschnitten detailliert eingegangen.

3.1 Konzeption der Morphologie

Unter der Morphologie einer Stadt versteht man ihre zwei- und dreidimensionale Gestalt im Sinne ihrer baulichen Struktur. Sie bildet sich im Wesentlichen aus der Beschaffenheit und dem Aufbau des Strassennetzes, der generellen Bebauungsstruktur, d. h. der Größe und Anordnung von Parzellen, Häuserblocks und Gebäuden sowie der innerstädtischen Freiflächen (vgl. Curdes, 1993, S. VII). Eine weitere Rolle für die Morphologie einer Stadt spielt die örtliche Oberflächenbeschaffenheit der Erde, welche ebenfalls einen Teil des Planungsprozesses beim Design eines dreidimensionalen Stadtmodells ausmacht.

Am Anfang des Designprozesses einer computergenerierten Stadt steht somit die Auseinandersetzung mit der potentiellen Morphologie ihres Stadtraums, sowie die Definition des zu erzielenden Gesamteindrucks unter Beachtung der zu vermittelnden Emotion in der finalen filmischen Umsetzung. Durch die zielgerichtete Gestaltung der Stadtmorphologie wird einerseits gewährleistet, dass das Stadtbild den in Drehbuch und Storyboard anfänglich festgelegten Anforderungen entspricht, des weiteren bietet sich dadurch auch die Möglichkeit, anhand der Morphologie subtile Botschaften zu vermitteln, die filmische Handlung emotional zu unterstützen und ihre Aussage zu verdeutlichen.

Auch im Falle des in dieser Arbeit thematisierten 3D-Filmprojektes bestimmt die in Drehbuch und Storyboard festgehaltene Story die wichtigsten Parameter für die adäquate Planung des Stadtmodells. Das Science Fiction Genre des Animationskurzfilms ermöglicht zwar grundsätzlich eine etwas freiere Gestaltung der Stadtstruktur, es ist dennoch von großer Wichtigkeit, beim Design grundlegende, für den Charakter einer Stadt typische Proportionen zu beachten; andernfalls besteht die Gefahr, dass das Design dem Rezipienten als erzwungen und unglaubwürdig erscheint. Die für die Konzeption des dreidimensionalen Stadtmodells bedeutendsten Designgrundlagen werden in den folgenden Abschnitten erläutert.

3.2 Bebauungsstruktur und Straßennetz

Die Bebauungsstruktur einer Siedlung oder Stadt ist in den meisten Fällen das erste Charakteristikum, das bei schneller Betrachtung ins Auge sticht. Unter Bebauungsstruktur versteht man nach Huber das räumliche Zusammenwirken von Einzelobjekten und Baugruppen innerhalb eines Siedlungsraums, das wiederum zumeist stark von der örtlichen Topographie beeinflusst ist (vgl. Huber, 1989, S. 64). Aus der Bebauungsstruktur ergibt sich zeitgleich die Struktur des Strassennetzes, welches die bebauten Gebiete durchzieht.

Im Falle des behandelten 3D Projektes spielt die Geschichte in einer fernen virtuellen Zukunft. Der global agierende monopolistische Konzern mit dem bezeichnenden Konzernnamen GLOBOTEX versucht das Gesellschaftssystem umzukrempeln und durch gezielte Normierung unter seine Kontrolle zu bringen. Um den absoluten Herrschaftsanspruch und die Macht dieses Antagonisten visuell zu unterstreichen wurde eine Stadtmorphologie angestrebt, die beim Zuschauer Erinnerungen an jene stättische Strukturen weckt, die bereits vor Jahrhunderten von absolutistischen Herrschern und Diktatoren zur Demonstration ihrer Macht benutzt wurden.

Als Beispiel ist hier Paris zu nennen, dessen Zentrum sich seit den weitreichenden, von Eugène Haussmann im 19. Jahrhundert durchgeführten Umstrukturierungsanstrengungen, durch eine exakte radialkonzentrische Bebauungsstruktur auszeichnet. Im Zentrum befindet sich der Triumphbogen, der Arc de Triomphe, auf den zwölf mit Alleen besetzte Prachtstrassen sternförmig zulaufen, um sich im Zentrum in einem Grüngürtel zu vereinen. Diese Struktur aus Bauwerken und Strassen führt zu einer signifikanten Betonung des Zentrums und erhebt den Triumphbogen zu einem Bauwerk von immenser Bedeutung und Symbolik.

Abbildung 2: Stadtraum um den Arc de Triomphe in Paris

Um eine ähnliche Wirkung zu erzielen wurden die oben erwähnten städtebaulichen Grundsätze auch beim Entwurf des 3D Stadtmodells von Anfang an in den Designprozess miteinbezogen. Strassennetz und Bebauungsstruktur entsprechen einem streng radialkonzentrischen Aufbau. In der Mitte eines sternförmig zusammenlaufenden Strassennetzes befindet sich ein ausladender, rund angelegter Platz, in dessen Mitte sich das Zentralmonument des bereits beschriebenen monopolistischen Konzerns befindet – der GLOBOTEX Tower, welcher gleichsam das Hauptquartier des Unternehmens als auch das Zentrum der Macht symbolisiert. Das hunderte Meter hohe, monumentale Gebäude ist von einem großen mehrspurigen Kreisverkehr umgeben, der von zahlreichen sternförmig einmündenden Hauptstrassen und zwei übergeordneten Highways gespeist wird.

Desweitern befindet sich eine beidseitig befahrbare Außenringstrasse um den Kreisverkehr, die die Anbindung an sämtliche Zufahrtsstrassen ermöglichen soll. Die Freiflächen die von Kreisverkehr und ringförmiger Außenringanbindung ausgespart werden sind als Grünflächen angelegt, die sich durch den Bewuchs von Wiesengräsern und vereinzelten hohen Baumgruppen auszeichnet. Das Stadtgebiet außerhalb des Kreisverkehrs ist – abgesehen von den stark variirenden Gebäudehöhen - von einer sehr regelmäßigen Bebauung geprägt.

Außer den in Zentrumsnähe von Alleebäumen besetzen Hauptrouten wirkt die restliche Stadtstruktur – ihren Bewohnern entsprechend – durch die extreme Konzentrik und Radialität sehr maschinell und artifiziell. Die Wahl dieser streng konzentrischen Bebauungsstruktur erwies sich hierbei jedoch als die ideale Möglichkeit um die - im Großen und Ganzen nicht existente, und nur durch den Character des Banker Bot repräsentierte - Person des Monopolisten GLOBOTEX über den Weg des Bebauungskonzeptes zu charakterisieren.

3.3 Topographie der Stadt

Ein weiteres Charakteristikum einer Stadt ist die Topographie, die Beschaffenheit der Oberfläche auf der sie sich ausbreitet. Diese Oberflächengestalt, das sogenannte Relief, stellt einen Faktor dar, der die Bebauungsstuktur eines Siedlungsraumes maßgeblich mitbestimmt (vgl. Huber, 1989, S. 61). Da im Beispiel dieses Projektes die Morphologie der computergenerierten Stadt eine narrativ wichtigere Position einnimmt kommt dem topographischen Aspekt nur eine untergeordnete Rolle zu. Die Gestaltung des topographischen Profils der Stadt spielt für die finale Wirkung dennoch eine wichtige Rolle, da schon geringe Höhenunterschiede im Relief eine wesentlich natürlichere Erscheinung erzielen als es ein vollkommen flaches Terrain es jemals könnte.

Aus diesem Grund wurde auch im Designprozess des Stadtmodells eine subtile Varianz des Terrains miteinbezogen. Im Zentrum gestaltet sich der Untergrund nahezu völlig eben zum Stadtrand hin jedoch erhöht sich das Terrain in unterschiedlichem Maße so dass der Stadtkern in einer leichten Senke liegt die sich auf einer Seite öffnet.

3.4 Stadtbild

Das Stadtbild wird im Wesentlichen durch die Oberflächen und Bauformen sowie durch die Bauhöhe der Gebäude einer Stadt geprägt. Im aktuellen Fall dominieren Gebäude von mittlerer und sehr großer Bauhöhe das Stadtbild, deren Grundrisse (*Footprints*) sich nach amerikanischem Vorbild meist über einen ganzen Häuserblock erstrecken. Die verwendeten Fassadentexturen entsprechen dabei den für mehrgeschossige westliche Bauwerke unserer Zeit typischen Eigenschaften.

4 Prozedurale Modellierung von Städten

Die Kreation ansprechender 3D-Modelle ist eine entscheidende Aufgabe bei der Produktion erfolgreicher Filme und Computerspiele. Allerdings stellt die Modellierung und Texturierung von extrem großen und detaillierten 3D-Umgebungen wie beispielsweise Städten und Landschaften einen äußerst aufwändigen und kostenintensiven Prozess dar, der einer einzelnen Arbeitskraft bis zu einigen Jahren Arbeitszeit kosten kann. (vgl. Müller et al. 2006, S. 1)

Um die Produktion der in dieser Fallstudie thematisierten urbanen 3D-Umgebung mit Hilfe von nur einer Designkraft in einem akzeptablen Zeitraum durchführen zu können musste für die Modellierung und Texturierung der Stadt von klassischen, manuellen Modelingtechniken, deren man sich in bekannten 3D-Sortwarepaketen wie etwa Autodesk Maya, Softimage oder 3ds Max bedient, abgesehen werden. Um die logistischen Anforderungen dieser Modeling-Aufgabe bewältigen zu können bot sich die Anwendung eines prozeduralen Lösungsansatzes an.

Unter prozeduralen Techniken versteht man im Allgemeinen Code-Segmente oder Algorithmen welche bestimmte Charakteristiken eines computergenerierten Modells spezifizieren. Die Basis einer jeden prozeduralen Technik ist die Abstraktion; anstatt sämtliche Details einer komplexen Szenerie zu speichern, werden bei prozeduralem Vorgehen diese Details zu einem Regelwerk an Algorithmen abstrahiert, durch deren Ausführung sich letztlich Modelle mit flexiblem Detailgrad generieren lassen. Der zeitliche Aufwand des 3D-Künstlers bei der Erzeugung von Details wird demnach in Rechenaufwand für das Computersystem umgelegt. Ein weiterer Vorteil prozeduraler Techniken ergibt sich aus der Möglichkeit, die Algorithmen anhand von Parametern kontrollieren zu können. Anhand von Parametern lassen sich so beispielsweise der resultierende Detailgrad (*LOD...Level of Detail*) oder die Varianz der Ergebnisse des Algorithmus mittels verschiedener Zufallsvariablen (*Random Variables*) bestimmen. (vgl. Ebert et al. 2003, S. 1f)

Für die Realisierung des computergenerierten Stadtmodells dieser Fallstudie kam die neuartige, erstmals im Juli des Jahres 2008 veröffentlichte, prozedural arbeitende Modeling Software CityEngine zur Anwendung. Die Applikation, die sich ursprünglich aus der Foschung im Zuge der Diplomarbeit von Pascal Müller an der ETH Zürich entwickelte, wird seit 2008 von dem schweizer Unternehmen Procedural Inc. vertrieben, dem Pascal Müller als CEO vorsitzt. Die freundliche Bereitstellung einer kostenlosen voll funktionsfähigen Lizenz der CityEngine durch Pascal Müller machten letztlich diese Fallstudie möglich.

Der nächste Abschnitt soll einen generellen Einblick in die Funktionsweise und den generellen Workflow der CityEngine vermitteln, in den darauf folgenden Abschnitten wird auf den konkreten Einsatz der CityEngine bei der Realisierung des prozeduralen Stadtmodells der Fallstudie detailliert eingegangen.

4.1 Die CityEngine - Architektur und Workflow

Die CityEngine bedient sich eines durchgehend prozeduralen Modelingansatzes der es erlaubt, auf Basis von vergleichsweise wenigen statistischen und geographischen Informationen seitens des Benutzers, Kilometer weit reichende Straßennetze und hunderttausende von Gebäuden von Grund auf neu zu erschaffen und durch gezielte Veränderung von Parameterwerten interaktiv zu beeinflussen. Der Benutzer hat überdies die Möglichkeit, für das Design erforderliche Daten wie beispielsweise Daten zur Raumnutzung, auszusparende Wasserflächen oder andere Hindernisse in Form von zweidimensionalen Bilddateien in die CityEngine einzuspeisen um die Generierung des Straßennetzes zu steuern oder um einzelnen Arealen bestimmte Massenmodell-Algorithmen zur Gebäudegenerierung zuzuweisen. (vgl. Müller et al., 2001 S. 1f)

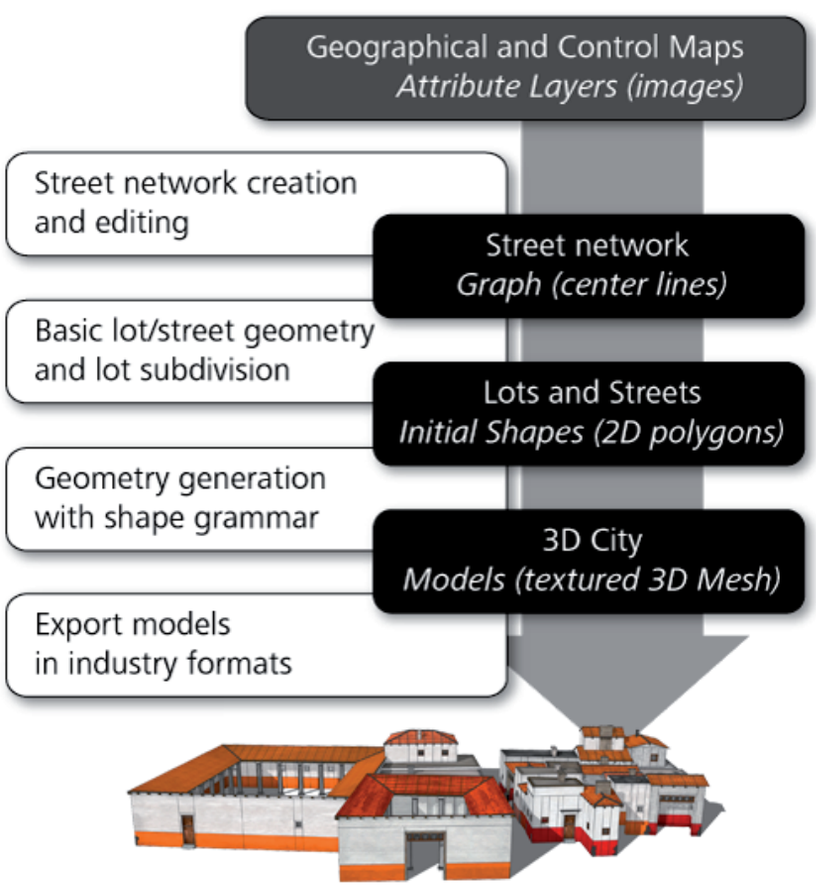


Abbildung 3: Workflowchart der CityEngie

Die CityEngine besteht aus mehreren Tools die sequentiell Ausgeführt werden und so den generellen Workflow bilden (siehe Abbildung 3, S. 30). Der Ausgangspunkt bei der Modellierung von Städten mit der CityEngine ist immer die Generierung des Straßennetzes. Da die Straßennetze realer Städte stets Muster darstellen, die gewissen Grundlagen folgen, macht sich die CityEngine bei der Strassengenerierung sogenannte *Extended L-Systems* zu Nutze (siehe Abschnitt 4.1.2 Strassengenerierung durch CityEngine L-Systems, S. 33).

Sobald die Eingabedaten und geographischen Control Maps in das *Road Generation System* gefüttert wurden, wird mit Hilfe der *extended L-System*-Technik das Strassennetz generiert. Die dadurch entstehenden Areale zwischen den Strassen ergeben die Grundstücke (*Lots*). Diese Lots können durch weitere Unterteilung nach einem bestimmten Algorithmus (*Subdividing*) zu kleineren Parzellen zerlegt werden, auf denen schließlich die Gebäude zu stehen kommen.

Im dritten Schritt kommt zur Generierung der Gebäude wieder ein anderes *L-System* zum Einsatz; die Gebäude entsprechen in diesem Stadium Zeichenfolgen (*Strings)*, die *boolean'sche Operationen* - angewandt auf simple geometrische Formen (*Shapes*) - repräsentieren. Die Resultate werden anschließend von einem *Parser* interpretiert und dem Visualisierungssystem zugänglich gemacht, welches polygonale Geometrie und *Texturemaps* berechnen kann. Die entstandene Geometrie kann anschließend in zahlreichen Formaten verschiedener Industriestandards exportiert werden. (vgl. Müller et al., 2001 S. 1ff)

4.1.1 L-Systeme

Das L-System ist ein Zeichenfolgen umschreibendes System, das erstmals im Jahr 1968 von Aristid Lindenmayer entwickelt wurde. Gegen 1990 wurde das System von Przemyslaw Prusinkiewicz erweitert um das Wachstumsverhalten von Pflanzen nachzuvollziehen, deren Aufbau und Struktur sich oftmals aus rekursiv wiederholenden Elementen zusammensetzt. Ein L-System ist eine Grammatik oder ein Alphabet von Symbolen wie etwa „F“, „+“ und „-“. Die Entwicklung eines L-System Modells vollzieht sich anhand eines Sets von Produktionsregeln welche die Ersetzung eines Symbols durch eine Zeichenfolge von 0 oder mehreren Symbolen beschreiben. Eine sehr einfache Produktion könnte zum Beispiel folgendermaßen lauten:

$$F \rightarrow F+F - - F+F$$

Dieser Ausdruck besagt, dass jedes F der Eingabezeichenfolge (*Input String*) durch die Folge F+F- -F+F ersetzt wird, was dazu führt, das die Größe der Zeichenfolge mit jeder rekursiven Durchführung der Prozedur zunimmt bzw. dass der Detailgrad nichtlinear ansteigt. Das Resultat wäre nach dem ersten Produktionsdurchlauf demnach:

F+F - - F+F + F + F - - F+F - - F+F - - F+F + F+F - - F+F

Abbildung 4 visualisiert diese Entwicklung unter der Annahme, dass F ein Längenvektor ist, und die Symbole „+“ bzw. „-“ Winkeln von +60° und -60° entsprechen. (vgl. Ebert et al. 2003, S. 307f)

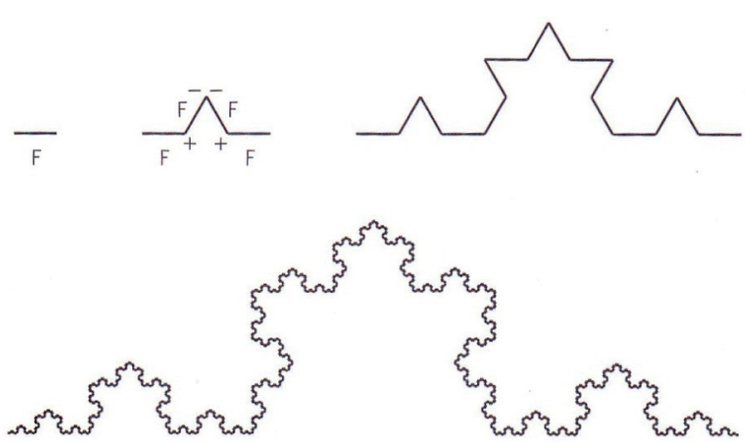


Abbildung 4: Abbildung: Koch‘sche Schneeflockenkurve - erstes Entwicklungsstadium und spätere Entwicklungsform nach zahlreichen Produktionen.

4.1.2 Strassengenerierung durch CityEngine L-Systems

Die CityEngine macht sich die bereits im vorigen Abschnitt grundlegend erläuterten L-Systeme zu Nutze um ein Strassennetzwerk zu generieren. Allerdings stellt die Generierung eines funktionierenden Strassennetzwerkes aufgrund der Notwendigkeit Hindernisse zu umfahren und funktionierende Kreuzungen zu bilden, weit höhere Ansprüche an ein L-System, als es bei der Modellierung natürlich vorkommender Formen wie Schneeflocken oder Baumstrukturen der Fall ist. Aus diesem Grund musste bei der Konzeption der CityEngine eine neue Art L-System entwickelt werden - das sogenannte *Extended L-System* (siehe Abbildung Abbildung 5, S. 34) (vgl. Müller et al., 2001 S. 2f).

Wie bereits am Anfang dieses Abschnittes erwähnt ist ein L-System ein Zeichenfolgen umschreibender, parallel arbeitender Mechanismus der auf einem Set an Produktionsregeln basiert. Jede der Zeichenfolgen (Strings) besteht aus unterschiedlichen Modulen die als Befehle interpretiert werden; die Parameter werden dabei in die Befehle integriert. (vgl. Müller et al., 2001 S. 3)

Die Produktion eines Regelsystems für komplexe Strassensysteme wie sie bei der CityEngine zu Anwendung kommen, macht es notwendig, eine extrem große Anzahl an Parametern und Bedingungen in das L-System einzuschreiben. Da die Komplexität jedoch bei jedem Produktionsdurchlauf rasant ansteigt, müssen bei jeder Implementierung einer neuen Zwangsbedingung sehr viele Regeln neu geschrieben werden, worunter die spätere Erweiterbarkeit des Systems leiden würde. Um dieses Problem bei der CityEngine zu umgehen legt das L-System bei jedem Schritt nur ein generelles *Template* (den *Ideal Successor*) an, anstatt zu versuchen, die Parameter der Module innerhalb des Produktionsdurchlaufes unterzubringen.

Die Bestimmung und Modifikation der Parameter des L-Systems wurde deshalb in externe Funktionen ausgelagert, die hierarchisch übergeordnete Tasks (*globalGoals*) und Bedingungen unterscheiden, welche sich aus der Umgebung ableiten (*localConstraints*). Jedes Mal, wenn die Regeln auf eine bestehende Zeichenfolge von Modulen angewandt wird werden folgende Schritte sequentiell durchgeführt:

- Der *Ideal Successor* wird vom L-System ausgegeben. Die Parameter der Module sind noch nicht zugewiesen.
- Aufruf der Funktion *globalGoals*, welche die dominanten globalen Parameterwerte bestimmt. Alle Parameterwerte sind jetzt gesetzt.
- Aufruf der Funktion *localConstraints*. Die Parameter werden mit den lokalen Bedingungen der Umgebung abgeglichen und so abgeändert dass sie diesen Bedingungen entsprechen. Werden keine passenden Parameter gefunden wird das Modul entfernt.

(vgl. Müller et al., 2001 S. 3)

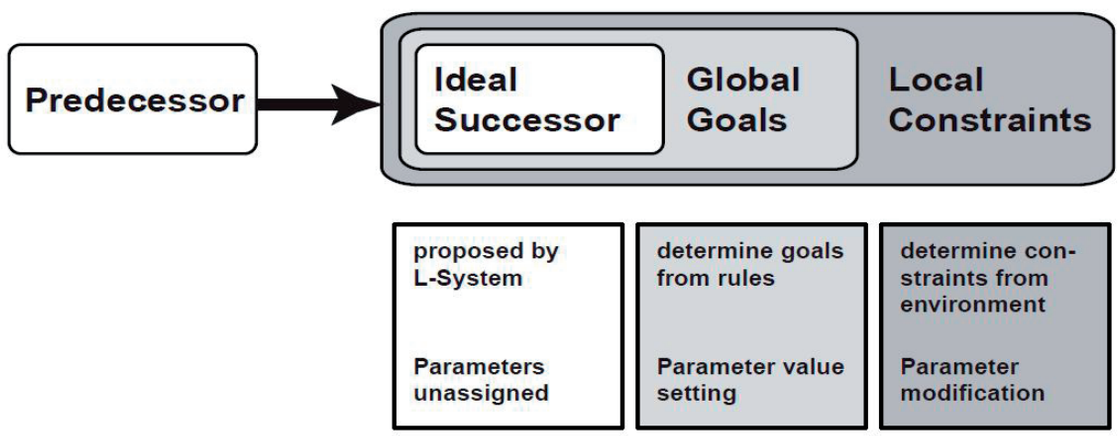


Abbildung 5: Extended L-System Funktionalität der CityEngine

Wie bereits oben erwähnt steuert die Function *globalGoals* die Anpassung der Strassensegmente an den übergeordneten Strassenraster beziehungsweise die Fortsetzung der hierarchisch höhergestellten Highways, von denen aus sich in weiterer Folge untergeordnete Nebenstrassen zu Straßennetzen der Siedlungsgebiete zusammenschließen. Die Anpassung an den vom Benutzer bestimmten Strassenraster wird durch verschiedene in der Applikation festgelegte Regeln gesteuert (siehe Abbildung 6).

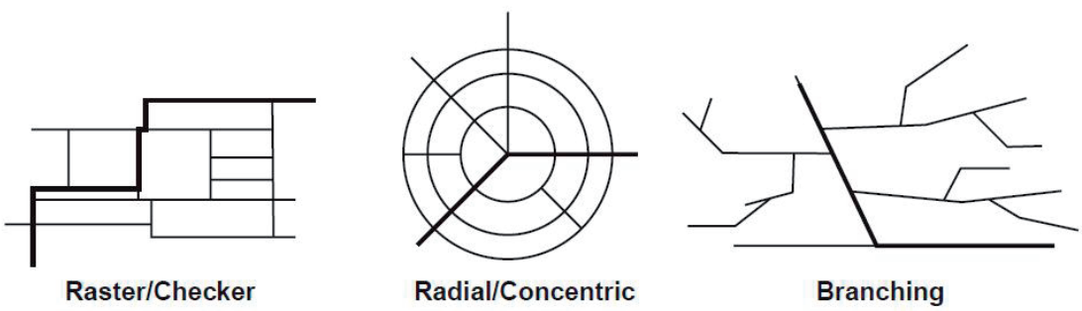


Abbildung 6: Schematische Veranschaulichung der wichtigsten von der CityEngine angewandten Straßennetzformen

Die Regel *Basic rule* ist die simpelst mögliche Regel, da sie eine Fortsetzung und Verzweigung der Straßenpfade veranlasst, ohne dabei Rücksicht auf einen übergeordneten Raster zu nehmen. Die sogenannte *New York rule* folgt einem gegebenen globalen oder lokalen Winkel unter Einbeziehung einer bestimmten Blocklänge und Blockbreite. Sie trifft somit auf urbane Gebiete zu in denen sowohl Haupt- als auch Nebenstrassen rechteckige Strukturen bilden. Die implementierte Regel namens *Paris* erzeugt eine radiale Struktur, die Highways und Nebenstrassen zentrieren sich hierbei sternförmig zu einem zentralen Punkt der Stadt. Eine weitere globale Regel ist die *San Francisco rule*. Der Strassenraster dieser Regel wird im Wesentlichen von der Form des Terrains der Stadt gesteuert. Die Hauptstraßen folgen hierbei dem Weg der geringsten Terrainerhöhung. Strassen unterschiedlicher Höhenlevels werden von Nebenstrassen verbunden die dem steilsten Terrainabfall folgen und demnach kurz sind. In vielen Fällen ist es notwendig mehrere dieser Ansätze bei der Generierung des Straßennetzes miteinzubeziehen. Für den Benutzer ist es in diesem Falle nicht erforderlich in den internen strassengenerierenden Code des L-Systems einzugreifen, da die Grundformen der Raster im Interface der CityEngine via Drop-Down Menu ausgewählt werden können. (vgl. Müller et al., 2001 S. 4)

Im nächsten Schritt werden mit Hilfe der Funktion *localCostraints* die durch die Funktion *gobalGoals* definierten Parameterwerte mit dem lokalen Umfeld abgeglichen. Kann kein zutreffendes Set an Regeln für die aktuelle Situation gefunden werden, wird der Status des jeweiligen Moduls auf FAILED (versagt) gesetzt und der entsprechende *String* durch eine neue Produktion des L-Systems entfernt; Straßen sowie Highways sind diesen Regeln unterworfen. Das Besondere an dieser erweiterten Variante eines L-Systems ist, dass bei der Produktion inkrementeller Fort- und Ersetzungen der Graphen sowohl geschlossene Kreisstrukturen als auch Kreuzungen von Graphen möglich sind, was zu der Bildung eines Netzwerks führt. Der erste Schritt der Funktion prüft ob das jeweilige Straßensegment innerhalb einer verbotenen Zone endet oder dieselbe durchquert.

In weiterer Folge wird nach Überlagerungen mit anderen Straßengraphen und nach Strassen und Kreuzungen gesucht, die sich innerhalb eines definierten Radius‘ des aktuellen Segmentendes befinden. Falls diese erste Überprüfung ein Enden eines Strassensegments innerhalb einer verbotenen Zone wie etwa Wasser diagnostiziert, versucht es die Parameter des Segments durch die Veränderung der Segmentlänge oder durch Rotation des Segments um einen bestimmten Winkel anzupassen, so dass sich das Segment innerhalb der erlaubten Zone befindet. Highways dürfen hingegen eine illegale Zone bis zu einer gewissen Länge überschreiten, um z. B. die spätere Bildung von Brücken über Wasserflächen zu ermöglichen. Wird eine Kreuzung gebildet, wird die weitere Ausbreitung der Strasse gestoppt. Sobald alle Parameterwerte gesetzt sind kann das generierte Strassennetz - der sogenannte Roadmap Graph - dem nächsten Schritt der CityEngine Pipeline weitergegeben werden. Abbildung 7: Anpassung der von globalGoals vorgeschlagenen Parameter und deren Korrektur durch localConstraints (S. 37) zeigt das korrigierende Eingreifen der Funktion *localConstraints* in das durch *globalGoals* vorgeschlagene erweiterte L-System. (vgl. Müller et al., 2001 S. 4f)

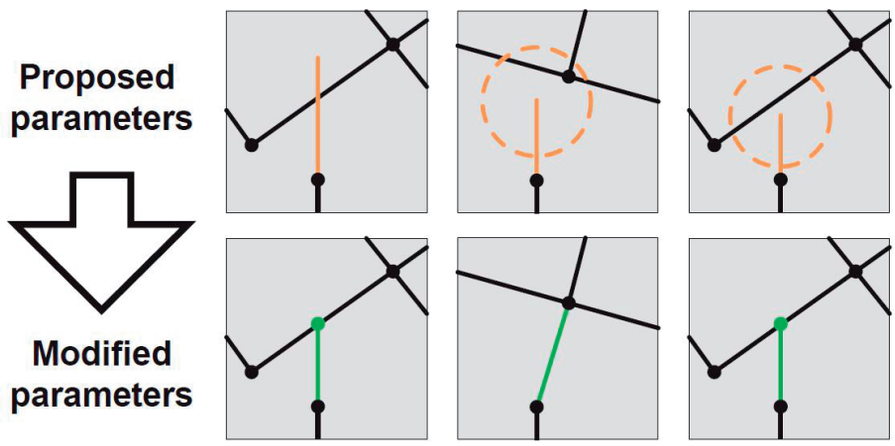


Abbildung 7: Anpassung der von globalGoals vorgeschlagenen Parameter und deren Korrektur durch localConstraints

Im darauf folgenden Schritt wird schließlich das zunächst noch auf Graphsegmenten basierende Straßennetz in *Street Shapes* umgewandelt, wobei die Straßenelemente erstmals grundlegende dreidimensionale Formen erhalten; Straßen, Gehwege und Kreuzungsstrukturen werden auf Basis der vom User festgelegten Vorgaben erzeugt und in einem eigenen *Layer* in der CityEngine *Szene* gesichert. Diese Basisformen dienen in Folge auch als Ausgangspunkt für die prozedurale Modellierung der Straßengeomtrie mittels der in Abschnitt 4.1.4 erläuterten CGA Shape Grammar. (vgl. CityEngine Manual, S.15)

4.1.3 Generierung von Gründstücksflächen

Aus den am Anfang der Prozedur erstellten Straßengraphen der Stadt lassen sich in weiterer Folge auch die Flächen der eingeschlossenen Häuserblocks und Gebäudeparzellen bilden. Um das zu erreichen werden zuerst in den Zwischenräumen des Strassennetzes Polygone erzeugt - die sogenannten *Alotments* oder kurz *Lots* - die den Grundstücksflächen der Häuserblöcke entsprechen. (vgl. CityEngine Manual, S.15f).

Um diese Grundstücksflächen weiter zu unterteilen kommt ein einfacher rekursiver Algorithmus zum Einsatz, der die längsten annähernd parallelen Seiten unterteilt, bis die entstehenden *subdivided lots* eine vom User bestimmte minimale Größe erreichen. Abbildung 8 verdeutlicht diesen Ablauf. (vgl. Müller et al., 2001 S. 4f)

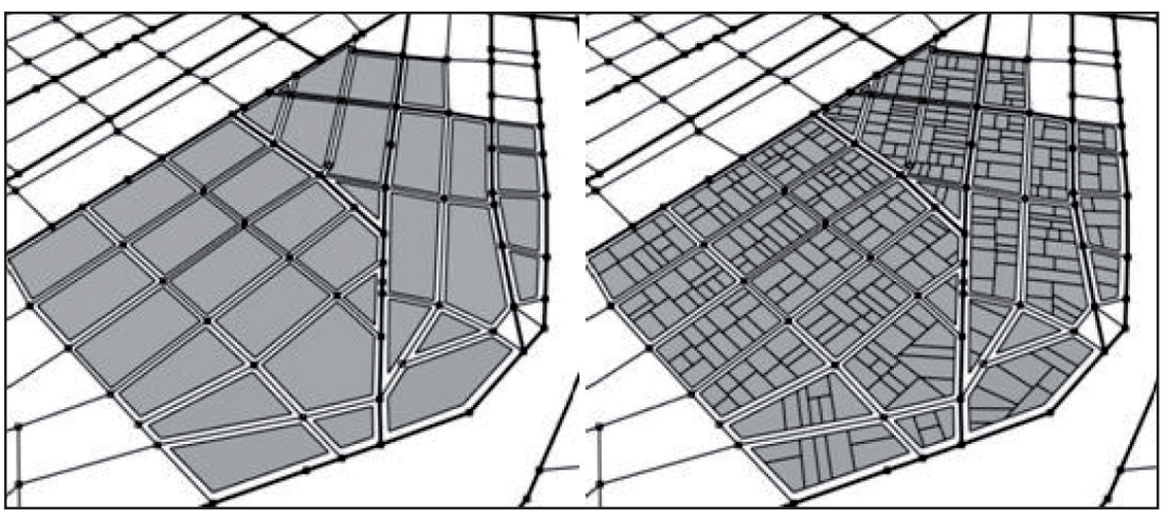


Abbildung 8: Bildung von Parzellen (subdivided Lots) auf Basis von Lots

Die bereits oben beschriebenen Prozeduren stellen somit den programminternen Algorithmus für die Entwicklung der Strassennetze und Grundstücksflächen dar. Im nächsten Abschnitt wird auf die Systematik bei der Programmierung von Regeln für die Generierung von Gebäuden und Straßengeometrie eingegangen, die ebenfalls auf dem Schema von L-Systemen aufbaut, der Regelsatz wird in diesem Fall jedoch durch den Benutzer mit Hilfe der CGA Shape Grammar selbst angelegt.

4.1.4 CGA Shape Grammar

Die letzten Kapitel dieser Arbeit befaßten sich ausschließlich mit auf L-Systemen aufbauenden Algorithmen, die innerhalb der Programmarchitektur der CityEngine laborieren, und durch den User lediglich durch die Bestimmung globaler Parameter, welche er selbst mittels Dialogsystem eingibt, kontrolliert werden. Um jedoch die finale dreidimensionale Form, beziehungsweise eine zur Weiterverarbeitung geeignete Geometrie von Gebäude- oder Strassenstrukturen generieren zu können, wurde es im Zuge der Entwicklung der Cityengine notwendig, eine gänzlich neuartige, formgebende Programmiersprache zu erschaffen, durch die geometrische Transformationen exakt beschrieben werden können.

Diese sogenannte *CGA Shape Grammar* ermöglicht es dem Benutzer, anhand von selbst erstellten, auf L-Systemen basierenden Regeln eine immens große Anzahl an Gebäuden mit hohem Detailgrad zu generieren. Die Architektur der Bauwerke wird hierbei durch die bewusste, händisch verfasste Definition von Attributen, Parametern, rekursiven Schleifen oder Bedingungen gesteuert. Der Kern dieses prozeduralen Systems besteht im Wesentlichen darin, dass eine bestimmte Regel durch die fortschreitende Entwicklung von Bestandteilen derselben Regel erweitert wird. Somit wird es möglich, durch die fortwährende Produktion des regelbasierten L-Systems, Geometrie immer weiter in kleinere Details aufzuteilen und zu überlagern, wodurch sich in Bezug auf die Architektur von Strassen und Häusern eine äußerst hohe Detailfülle erzielen lässt.

Die Struktur der CGA Shape Grammar wurde stark durch L-Systeme von Prusinkiewisz und Lindenmayer inspiriert, musste jedoch für die Produktion von Architektur erweitert werden. Während parallele L-Systeme geeignet sind um fotschreitendes Wachstum darzustellen, erlaubt eine sequentielle Anwendung von Produktionsregeln - wie es etwa bei der CGA Shape Grammar der Fall ist - die Formung von Strukturen sowie die räumliche Verteilung von Komponenten und Details. (vgl. Müller et al., 2006 S. 2f)

Laut der Syntax , die der CGA Formgrammatik zugrunde liegt, besteht eine Form (*Shape*) aus einem Symbol in der Form eines *Strings*, aus der eigentlichen Geometrie - repräsentiert durch geometrische Atrribute, und aus numerischen Attributen. Die wesentlichsten geometrischen Attribute sind die Position der Shape, die drei orthogonalen Vektoren X, Y und Z, welche ein Koordinatensystem beschreiben und der Größenvektor S. Diese Attribute definieren zugleich eine ausgerichtete, quaderförmige umhüllende Bounding Box, die in der CGA Shape Grammar als Scope bezeichnet wird. Der Produktionsprozess dieses Systems sieht es vor, aus einer beliebigen Anzahl von vorhandenen Shapes durch die Anwendung von Produktionsregeln neue Shapes zu erzeugen; in der Produktionsstufe mit der höchsten Priorität bilden die durch die Generierung der Straßen entstandenen Lots die erste Stufe des Produktionsprozesses. Die generierten Shapes werden darauf hin in die aktuelle Konfiguration eingebunden und die Ursprungsshape deaktiviert. Die allgemeine Form einer Produktionsregel lautet:

id: predecessor : cond -> successor prob

id steht für die Identifikation der Regel, *predecessor* ist ein Symbol das eine Shape definiert, die im Laufe der Produktion durch die *successor* -Shape ersetzt werden soll, wobei *cond* eine Leitlinie darstellt durch deren Verifizierung die jeweilige Regel angewendet werden kann. (vgl. Müller et al., 2006 S.2f)

Die grundlegendste und einfachste Regel stellt beispielsweise die Generierung der Urform eines Gebäudes dar. Durch die Anwendung einer Extrudierungsfunktion wird die noch zweidimensionale Grundstücksform *Lot* in Y-Richtung in einen volumetrischen Körper mit bestimmter Höhe (*height*) verwandelt, wodurch erstmals eine dreidimensionale Grundform des Gebäudes entsteht:

Lot --> extrude(height) Building

Durch diese Prozedur wird die der Regel *Lot* zu Grunde liegende Shape in die resultierende Shape überführt auf welche nun die Regel Building angewendet wird. Anhand des *Component-Split* Befehls ist es nun zum Beispiel möglich, diese quaderförmige Grundform in seine veschiedenen Seitenflächen aufzuteilen:

Building --> comp(f) { front: FrontFacade | side: SideFacade | top: Roof }

Als Folge entstehen hier drei neue Regeln, eine für die vordere Fassade, die seitlichen Fassaden sowie für das Dach. Jede dieser Regeln wird ab nun durch eine Vielzahl von Produktionsschritten einzeln weiterverarbeitet. Eine weitere wichtige Funktion ist hierbei die sogenannte Split-Funktion, die es möglich macht, eine Shape in einer definierten Richtung zu unterteilen, was beispielsweise bei der Definition von Stockwerken sowie Fensterreihen zum Tragen kommt (vgl. Schubiger-Banz, 2009 S. 21):

FrontFacade --> split(y) { 4: GroundFloor | {~3.5: Floor}* }

Die Vorderfassade wird hier in Y-Richtung in ein vier Meter hohes Erdgeschoss und in die Obergeschoße geteilt. Die Obergeschoße sind in dem Beispiel nicht exakt in der Höhe definiert, sondern folgen einem Richtwert von ungefähr 3,5 Metern pro Geschoß, das System versucht dann in die Anfangs definierte fixe Gebäudehöhe (height) die Anzahl an Floors einzuziehen, dass der Richtwert von 3,5 Metern Geschoßhöhe möglichst genau eingehalten wird.

In weiterer Folge wird durch die Anwendung verschiedenster Regeln dieses Prozedere so lang wiederholt, bis sämtliche Einzelteile der Gebäudestruktur definiert sind, beziehungsweise bis der gewünschte Detailgrad erreicht wurde: (vgl. Schubiger-Banz, 2009 S. 21)

Floor --> split(x) { 1: Wall | {~3: Tile}* | 1: Wall }

Diese Regel teilt jedes Obergeschoß horizontal in ungefähr drei Meter breite Stücke (Tiles) die später die Fenster beinhalten werden, und in ein Randstück von einem Meter Breite auf jeder Seite der Fassade. Beim Erdgeschoss kommt die selbe Regel zum Tragen, allerdings wird hier noch ein Tile für die Eingangstür eingefügt (vgl. Schubiger-Banz, 2009 S. 21):

GroundFloor --> split(x) { 1: Wall | {~3: Tile}* | ~3: EntranceTile | 1: Wall }

Um die grundsätzliche Unterteilung des Gebäudes abzuschließen, müssen nun noch die Elemente mit dem Symbol *Tile* in sich gesplittet werden, um eine Fläche für die Implementierung der Fenster zu schaffen. Das lässt sich wiederum durch den *Split*-Befehl bewerkstelligen. Um Codezeilen zu sparen ist kann via *Nesting* in einen horizontalen *Split* ein weiterer vertikaler *Split* integriert werden (vgl. CityEngine Help, S. 394):

```
Tile -->
        split(x){ ~0.5 : Wall
        | 2 : split(y) { 1: Wall | 1.5: Window | ~1: Wall }
        | ~0.5 : Wall }

EntranceTile -->
        split(x){ ~0.5 : SolidWall
        | 2 : split(y) { 2.5: Door | ~1: Wall }
        | ~0.5 : SolidWall }
```

Um dieses simple Gebäude mit einem Material zu versehen werden nun den Elementen *Wall* und *Window* Texture zugewiesen, wobei Wall zusätzlich zur eigentlichen Textur auch eine *Dirt Map* erhält. Die Regel *Window* sieht zusätzlich vor, einen Fensterrahmen in Form eines importierten Polygon Objeks im Format .OBJ auf den Flächen der Fenstertextur zu positionieren um eine höhere Detailgenauigkeit zu erlangen (vgl. Schubiger-Banz, 2009 S. 22):

```
Wall -->
        set(material.colormap, „brickwall.tif“)
        set(material.dirtmap, „dirtmap.tif“)

Window -->
        set(material.colormap, „window01.tif“)
        i(„window.obj“)

Door -->    set(material.colormap, „door.tif“)
```

Während dieses Beispiel sehr einfache Regeln benutzt, ist es durch die Verwendung von Attributen und *Random* Funktionen möglich, Parameter und Texturen variieren zu lassen, um so ein weitaus heterogeneres Resultat zu erzielen. Beispielsweise können auf diese Weise Materialien aus einem großen Pool an verschiedenen Texturen ausgewählt und auf die Häuser verteil werden, oder die Form der Gebäude indivuduell beeinflusst werden. (vgl. Schubiger-Banz, 2009 S. 22)

In den nächsten Kapiteln wird auf die Vorgangsweise bei der Erstellung des Stadtmodells im Zuge der Fallstudie eingegangen. Hierbei werden dem Aufbau des Strassennetzes, der groben Struktur des zu Grunde liegenden Regelwerks und der Texturierung des Stadtmodells Aufmerksamkeit geschenkt. Weiters werden Hindernisse und Probleme, die sich auf dem Weg zum fertigen Modell ergeben haben, behandelt.

5 Städtebau mit der CityEngine

Dieser Abschnitt befasst sich mit der prozeduralen Generierung des Stadtmodells im Zuge der Durchführung der Fallstudie. Die in den Abschnitten 3.2, 3.3 und 3.4 erläuterten Anforderungen an den Aufbau der Stadt dienten hierbei als gestalterische Leitlinie. Dieses Kapitel schildert überdies auch die Hürden und Hindernisse, die bei der Erarbeitung der Fallstudie zu Tage getreten sind, sowie die Lösungswege die letztlich zur Lösung der Problemstellungen geführt haben.

5.1 Generierung des Street Network

Der erste Schritt beim Aufbau einer computergenerierten Stadt ist in den meisten Fällen die Erzeugung des Strassennetzes, da dieses die zu Grunde liegende hierarchische Ebene darstellt, auf der sämtlich weitere Bearbeitungsschritte aufbauen. Aufgrund der großen Bedeutung des Stadtzentrums für die Umsetzung der filmischen Handlung dieser Fallstudie, wurde bei der Planung des Straßennetzes von innen nach außen gearbeitet. An erster Stelle stand hierbei die Konzeption des zentralen Kreisverkehrs, der Grünflächen die von ihm eingeschlossen sind, sowie der zentrisch anschließenden Hauptstraßen.

Um die Ausmaße und Proportionen von Elementen wie Straßen, Randsteinen oder Grünflächen einschätzen zu können, wurde noch vor der tatsächlichen Produktion eine Vektorgrafik im Format .AI (Adobe Illustrator) angefertigt. Da die CityEngine erlaubt, Grafiken als Texture Maps auf Ebene des Grids - beziehungsweise an einer beliebigen Position - anzuzeigen, konnte diese Visualisierungsgrafik auch innerhalb der CityEngine als Orientierungshilfe benutzt werden. Hierzu wurde die Grafik mit dem Mittelpunkt des Kreisverkehrs exakt zentrisch auf einer quadratischen, 5000 mal 5000 Pixel großen Bitmap angeordnet und anschließend als Bilddatei im Format .PNG in die CityEngine als Texture Map eingebunden, so dass der Mittelpunkt der Grafik direkt im Ursprung des Raumkoordinatensystems der CityEngine Szene zu liegen kam. (siehe Abbildung 9).

Abbildung 9: Erster Plan des Stadtzentrums im Vektorformat .AI

Bei der Generierung des Strassenetzwerks ist das Design der sogenannten Street Graphs der erste Arbeitsschritt. Um die Straßengraphen zu erstellen, auf deren Basis später die endgültigen Street Shapes generiert werden, stehen dem User mehrere Möglichkeiten zur Verfügung; Der klassische Weg sieht vor, Strassennetzwerke durch die userseitige Konfiguration des Street Genereation Wizzard zu kreieren und durch fortwährende Produktion quantitativ zu erweitern, es ist jedoch auch möglich, die Straßengraphen mit Hilfe der Edit Tools der CityEngine manuell zu setzen und zu modifizieren.

Als weitere Methode muss an dieser Stelle auch der Import von Straßennetzwerken angeführt werden - die CityEngine erlaub den direkten Import von vektorbasierten Straßennetzwerken in den Formaten .DXF (AutoCAD), .SHP (GIS basiertes Shapefile) und .OSM (Open Street Map). Im Falle dieser Arbeit kamen die ersten beiden Varianten zu Einsatz. (vgl. CityEngine Manual, S. 122ff)

5.1.1 Erster Lösungsansatz und resultierende Probleme

Der einfachste und typischste Zugang zur Generierung der Stadtstruktur würde vorsehen, ein radiales Straßenraster mit Hilfe des *Street Generation Wizzard* vom Ursprung des Koordinatensystems aus anzugelegen und über viele weitere Produktionen zu einem radialen Straßennetz zu vergrößern. Im speziellen Fall dieses Projektes stellte der Aufbau des Straßennetzes jedoch eine besondere Herausforderung dar. Das Stadtzentrum besteht aus einem komplexen Verkehrssystem, das sich aus einem mehrspurigen, von den zentral zulaufenden Hauptverkehrsachsen gespeisten Kreisverkehr, einer umgebenden ringförmigen Umfahrungsstraße und mehreren Zuleitungen zu den Haupthighways zusamensetzt. Es stellte sich somit von Anfang an die Frage ob es möglich ist, diese komplexe Straßenstruktur durch prozedurale Algorithmen nachzubilden.

Um diesen Sachverhalt zu testen wurde der gesamte Kreisverkehr und seine Zubringerstraßen händisch mit *Street Graphs* nachgebildet (siehe Abbildung 10) und anschließend einige Testläufe für die Generierung der *Street Shapes* durchgeführt. Im Zuge dessen stellte sich heraus, dass Aufbau und Proportionen des Kreisverkehrsystems eine exakte prozedurale Straßenmodellierung unmöglich machten. Das offensichtliche Problem lag - wie in Abbildung 11 ersichtlich - in der fehlerhaften Interpretation der extrem komplexen Kreuzungsstrukturen.

Der prozedurale Ansatz der Modellierung des Kreisverkehrs ergab jedoch noch ein weiteres Problem: da die CityEngine aus speicherökonomischen Gründen keine einheitlichen Polygonobjekte ausgibt, sondern stattdessen sämtliche - zu einem bestimmten Material zugehörige - Geometrien in jeweils ein einzelnes Objekt zusammenlegt, würde der Kreisverkehr aus einer Vielzahl von einander separierten Objekte bestehen. Beispielsweise würden alle verschiedensten Arten von Fahrbahnbestandteilen, Randsteine, Kreuzungspunkte, Zebrastreifen, etc. als jeweils ein zusammengefügtes Objekt ausgegeben, was es praktisch unmöglich macht, die Kanten der prozedural erzeugten Straßenstrukturen mittels Subdivision Modeling in Autodesk Softimage in perfekte Rundungen zu verwandeln. Dieser Ansatz wurde deshalb aus besagten Gründen verworfen.

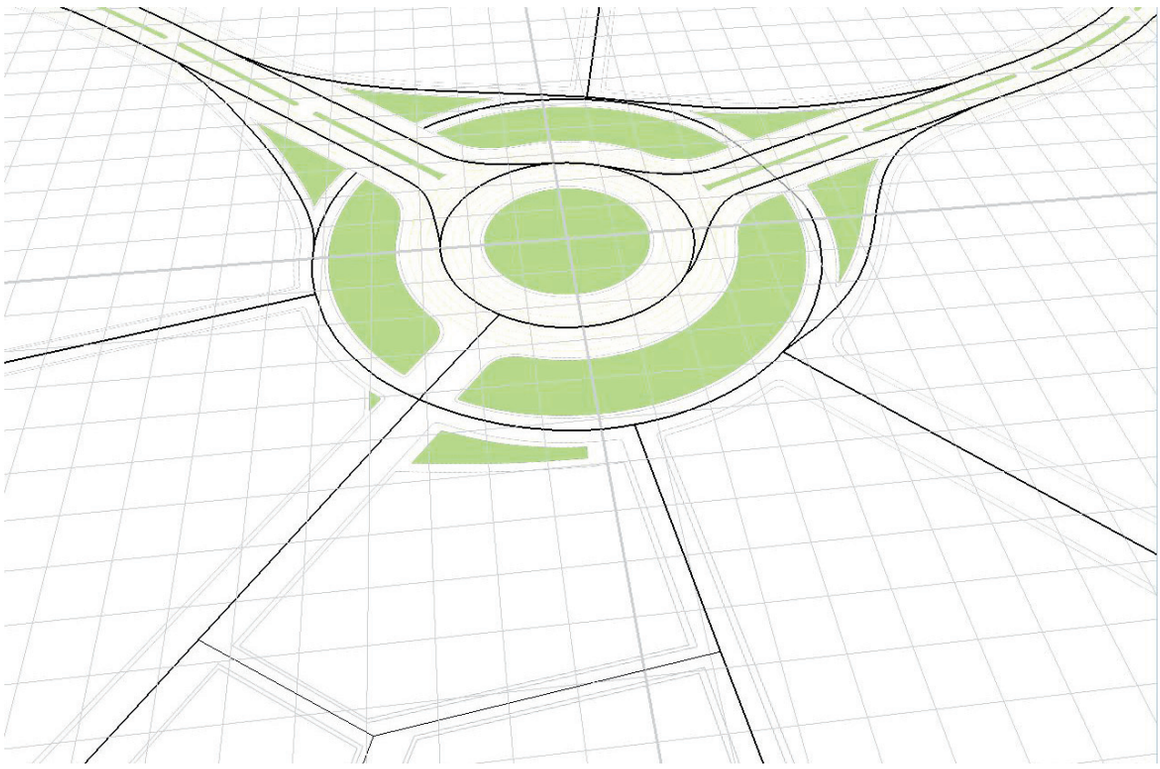


Abbildung 10: Kreisverkehrs mit theoretischem Street Graph Aufbau

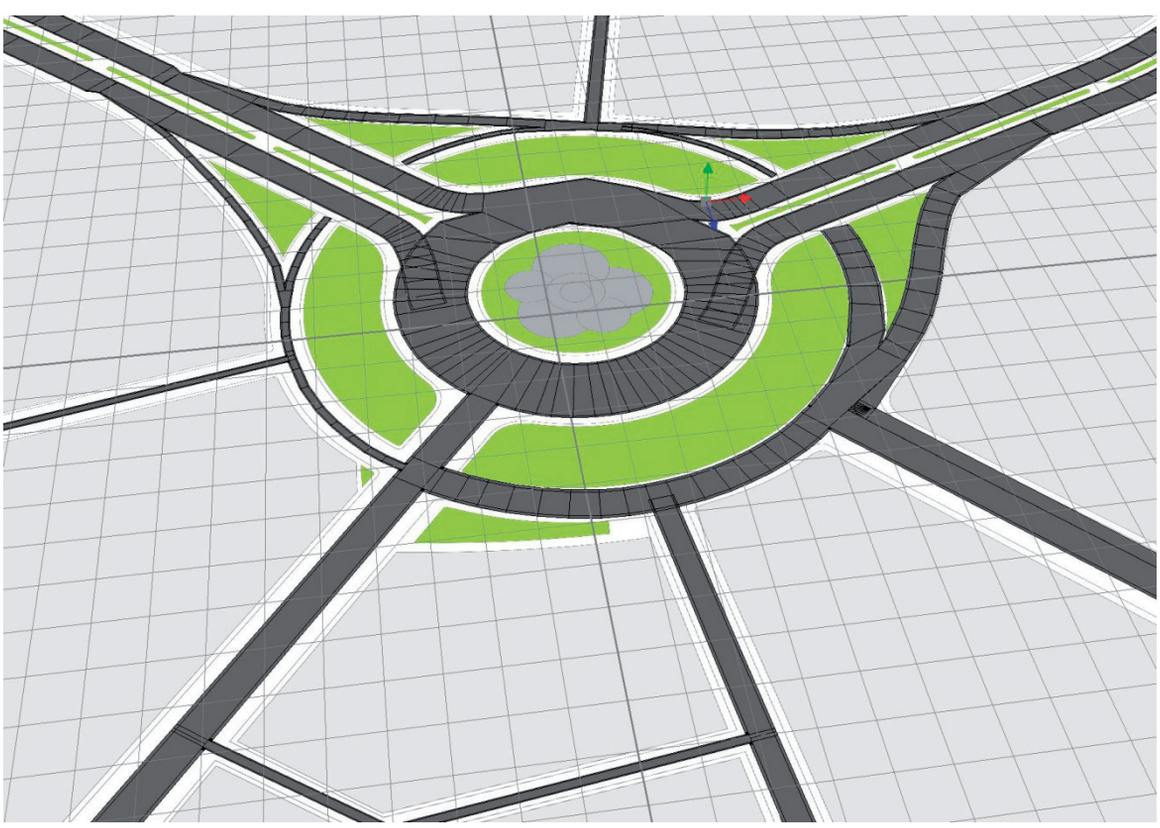


Abbildung 11: Unbrauchbares Resultat des Versuchs

5.1.2 Finaler Lösungsansatz

Um die in Abschnitt 4.2.1.1 beschriebenen Probleme zu umgehen fiel die Entscheidung auf die manuelle Ausmodellierung des Kreisverkehrsystems in Autodesk Softimage. Der darauf folgende Schritt sah vor, das in Softimage modellierte detailreiche Kreisverkehrmodell nachträglich mit den prozedural erzeugten Nebenstraßen der CityEngine zu kombinieren. Obwohl dieser Zugang die gravierenden Probleme des ersten Lösungsversuchs beseitigte, verursachte er dennoch neue Schwierigkeiten im Produktionsablauf, die in diesem Fall zwar eindeutig lösbar waren, jedoch mit einem extrem hohen Mehraufwand an Arbeitszeit verbunden waren. Auf die Modellierung und Texturierung des Kreisverkehrs wird in Kapitel 5.3 Manuelles Modeling des Hauptverkehrsknotens (S. 54) und Kapitel 5.7 Manuelles Texturing des Hauptverkehrsknotens (S. 70) detailliert eingegangen.

Das vordergründigste Problem des oben beschriebenen Ansatzes ergibt sich aus der Tatsache, dass sich die *Initial Shapes* der Grundstücke *(Lots)* (siehe Abschnitt 5.2. Initial Shape Creation und Parzellierung), auf denen in einem späteren Arbeitsschritt die prozeduralen Gebäude aufgebaut werden, nur in den Zwischenräumen eines Straßennetzes und nicht in offenen Straßenstrukturen generieren lassen. Es bedarf also eines Bereichs, der ringsum von Straßengraphen umgeben ist. Stünden die Straßengraphen direkt an den händisch modellierten Straßen an und endeten aus Sicht der CityEngine sozusagen als Sackgassen, könnten von dem Algorithmus im nächsten Arbeitsschritt - der sogenannten *Shape Creation Phase* - an diesen Stellen keine Grundstücksflächen erzeugt werden.

Aus diesem Grund war es zwingend notwendig eine Art Workaround anzuwenden um im prozeduralen Modelingprozess auch in denjenigen Gebieten *Lots* zu erhalten, die direkt an die manuell erzeugten Straßen anschließen. Um das zu erreichen wurden alle äußeren Konturen der nicht prozedural generierbaren Hauptverkehrsrouten sowie des Kreisverkehrs mit *Street Graphs* eingesäumt. So entstanden in sich abgeschlossene, von Straßengraphen umgebene Areale, die im Prinzip Stadtvierteln gleichen. Abbildung 12 verdeutlicht diesen Aufbau.

Abbildung 12: Umrandung der prozduralen Areale mit temporären Streets

Diese in Abbildung 12 dargestellten abgeschlossenen Graphnetzwerke konnten nun als Ausgangspunkt für die prozedurale Modellierung sämtlicher Nebenstraßen verwendet werden. Dazu wurden erst *Graph* Elemente markiert und anschließend mit Hilfe des Street Generation Wizard das Straßennetzwerk sukzessive in der gewünschten radialen Rasterform erweitert (siehe Abbildung 13).

Obwohl die CityEngine aufgrund ihres ausgefeilten Wizzards bei der Straßengenerieung sehr gute Ergebnisse lieferte, war es dennoch oft notwendig, die prozedural generierten Straßengraphen den eigenen Anforderungen von Hand anzupassen. Diese händischen Korrekturen beinhalteten größtenteils die Bereinigung diverser Straßenverläufe von ungewollten Kreuzungssegmenten, die Anpassung von Grundstücksgrößen sowie die Feinjustierung von Attributen wie Straßen- und Gehwegbreite, um dem städtebaulichen Anspruch des Storyboards gerecht zu werden. Zum Teil wurden auch ganze Stadtareale händisch nachgearbeitet, um das Straßennetz bis ins feinste Detail den Anforderungen anzupassen; dabei war die manuelle Einzeichnung von wichtigen größeren Straßen als Startpunkt für die prozedurale Weiterproduktion von Nebenstraßen der häufigste Bearbeitungsschritt.

Im Anfangsstadium der Fallstudie gestalteten sich diese Bearbeitungen noch als ein etwas schwerfälliges Unterfangen, da in der damals erhältlichen CityEngine Version (v2008.1) für alle Tools außer dem *CGA Shape Grammar Editor* keine Undo-Funktionalität zur Verfügung stand. Mit Version 2009.1 wurden jedoch eine Undo-Funktion sowie einige sehr hilfreiche Werkzeuge zur Bearbeitung der *Graph Networks* implementiert, was die Editierung der Graphen sehr erleichterte.

Nach dem das Straßennetzwerk im Grundriss fertiggestellt war, konnte der Topographie des Terrains durch die Anwendung einer sogenannten Heightmap etwas Abwechslung verliehen werden. Unter einer Heightmap versteht man einen Layer innerhalb der CityEngine Szene, der es erlaubt, mittels einer auf Graustufen basierenden Bitmap-Grafik den Höhenverlauf des Terrains zu gestalten. Heightmaps können den Höhenverlauf von Straßennetzwerken sowie in weiterer Folge auch von Initial Shapes - also Lots und Street Shapes - steuern. Die Helligkeit (Brightness) der Bitmap bewirkt Terrainanstiege, rein schwarze Areale bestimmen das Nulllevel (vgl. CityEngine Manual, S. 103).

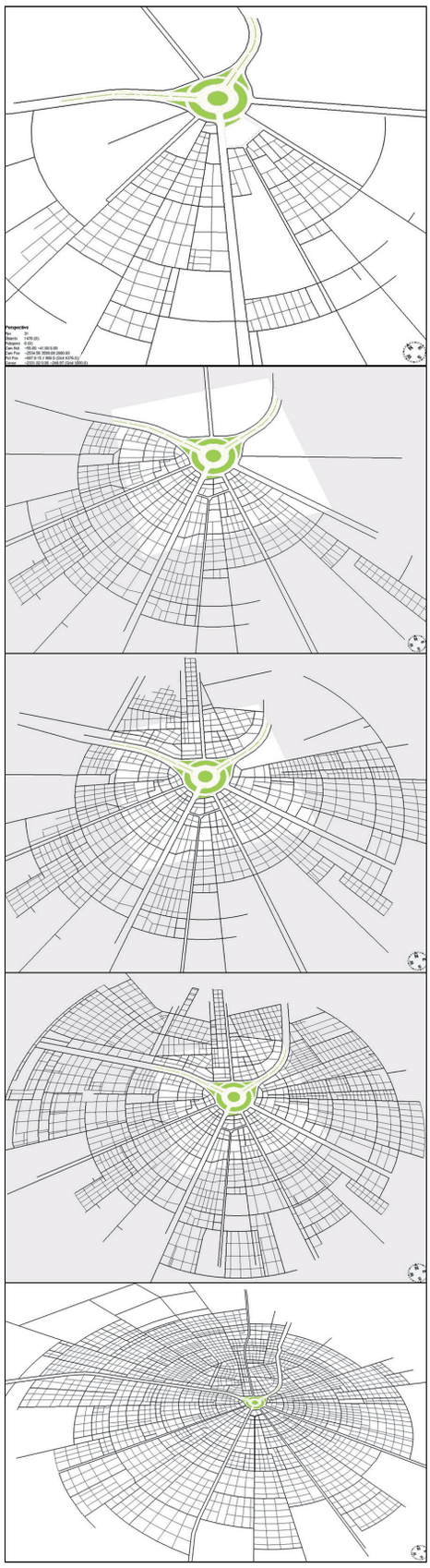


Abbildung 13: Evolution des Street Network

Im Falle dieses Projektes wurden die Terrainunterschiede sehr subtil gehalten, der Hauptzweck im Einsatz der Heightmap bestand darin, die urbane Umgebung auf unauffällige Weise heterogener wirken zu lassen, als sie es täte wenn sie auf einem exakt ebenen Untergrund platziert wäre. Abbildung 14 und Abbildung 15 zeigen einerseits die verwendete Heightmap und den resultierenden subtilen Effekt auf das Street Network. Der Höhenunterschied zwischen 100% schwarz und 100% weiß beträgt in dem Fall 200 Meter. Das zugehörige Attribut des Heightmap Layers lautet:

```
attr elevation = map_01(brightness, 0.0, 200.0)
```

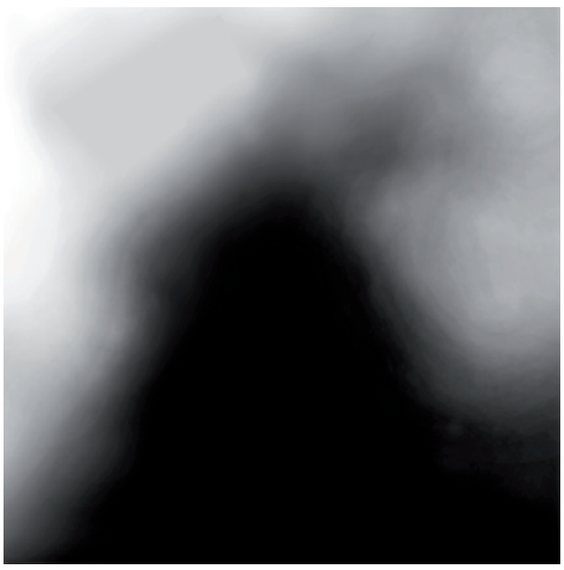


Abbildung 14: Die für die Gestaltung des Terrainanstiegs benutzte Heightmap

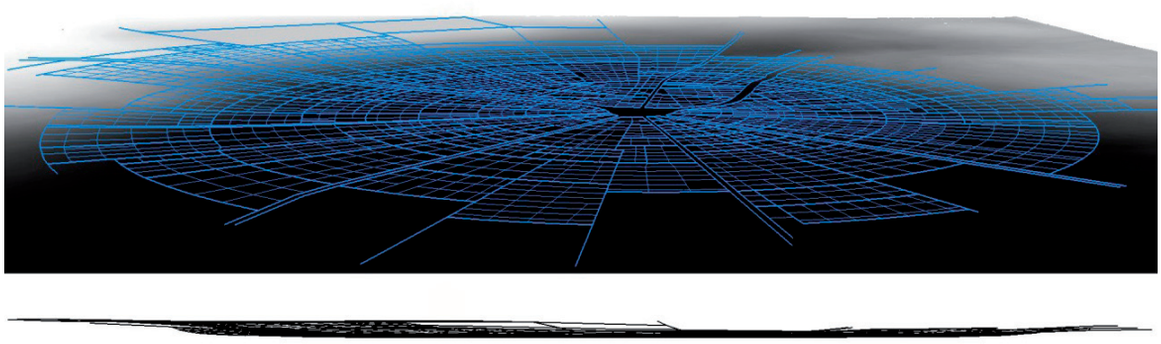


Abbildung 15: Die resultierende Höhenangleichung des Street Network in Frontriss und Perspektive

5.2 Initial Shape Creation und Parzellierung

Nach der Fertigstellung des *Graph Network* konnten nun die Initial Shapes der Strassen (*Street Shapes)* - und in Folge - die Initial Shapes der Gebäude (*Lots)* erzeugt werden. Unter einem Initial Shape versteht man die zu Grunde liegende polygonale *Shape* auf der eine Produktion durch die *CGA Shape Grammar* aufbaut, um durch fortwährende Modifikation der *Shapes* am Ende 3D-Geometrie auszugeben. Einem *Initial Shape* kann in der CityEngine ein CGA Rule File mit der Endung .CGA zugewiesen werden und kann desweiteren auf ein Initial Shape Symbol (die erste Regel die auf dem Initial Shape angewendet wird) und eine individuelle Attributkonfiguration referenzieren. Grundstücksflächen (Lots), Street Shapes sowie aus Softwarelösungen importierte Polygonobjekte können als *Initial Shape* herangezogen werden. (vgl. CityEngine Manual, S.135 f)

Abbildung 16 zeigt die generierten Street Shapes und die davon eingeschlossenen Lots. In der Abbildung ist gut zu erkennen wie die temporär angelegten Straßen den Kreisverkehr und seine Zubringerstraßen einsäumen. Dieser Workaround ermöglichte letzten Endes die Generierung aller benötigter Lots. In der Phase der Zusammenführung der prozedural sowie manuell erstellten Straßengeometrien in Autodesk Softimage mussten diese temporär angelegten Straßen nachträglich wieder entfernt werden um zwischen den beiden Modellen Kreuzungspunkte schaffen zu können. Auf dieses Prozedere wird in Kapitel 5.6 Zusammenführung der Geometrietypen in Softimage (S. 67) genauer eingegangen.

Die Generierten Lots wurden im Anschluss durch die CityEngine-interne Subdivide-Funktion Parzelliert. Dabei wurden die Einstellungen so getroffen, dass bei einem Großteil der Lots keine Unterteilung vorgenommen wird und nur extrem große Grundstücke unterteilt werden. Dieses Vorgehen war für die ästhetische Wirkung der Stadt von Bedeutung, da Gebäude ein eigenes von Straßen umgebenes Grundstück beanspruchen eine massivere Wirkung besitzen, was im Falle dieses Projektes für das Stadtbild einen Vorteil darstellte.

Abbildung 16: Generierte Street Shapes und Lots

Abbildung 17: Generierte Street Shapes und Lots

5.3 Manuelles Modeling des Hauptverkehrsknotens

Da es - wie in den Abschnitten 5.1.1 und 5.1.2 bereits beschrieben - nicht möglich war, die komplexe Kreisverkehrstruktur prozedural zu fertigen, musste ein händisch in Autodesk Softimage erstelltes Modell des Kreisverkehrs angefertigt werden, das zu einem späteren Zeitpunkt mit der exportierten prozedural erzeugten Geometrie aus der CityEngine verheiratet wurde.

Der ebenfalls in Abschnitt 5.1 vorgestellte Plan des Kreisverkehrs diente auch bei der manuellen Modellierung des Kreisverkehrs als Designvorlage. Ähnlich dem bereits bekannten Prozedere in der CityEngine wurde der Plan als Bitmapgrafik exakt mit dem Mittelpunkt des Kreisverkehrs im Ursprung des Raumkoordinatensystems angeordnet und auf einen zentral positionierten Grid projiziert. Mit Hilfe der Standard Modeling Tools von Softimage wurden anschließend die groben Züge des Kreisverkehrs in flacher Form nachmodelliert (siehe Abbildung 18).

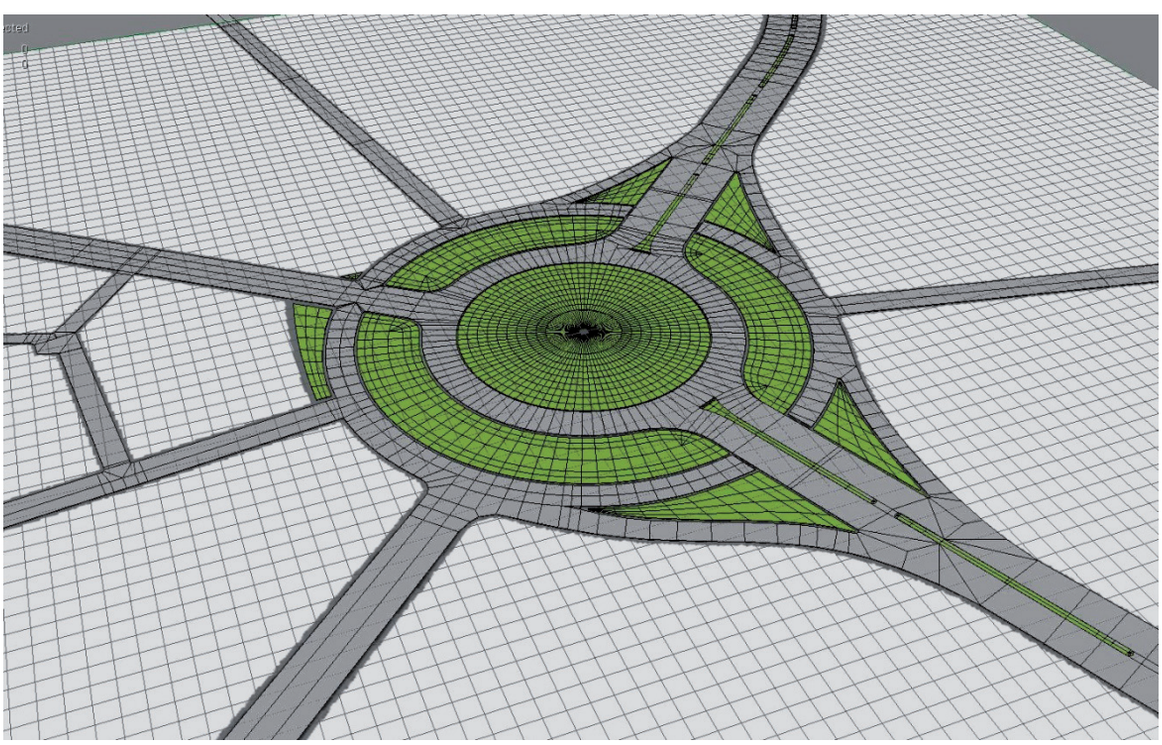


Abbildung 18: Erstes flaches Stadium des Kreisverkehrmodells ohne Unterführungen, Terraingestaltung und Randsteine

Im Anschluss wurden zunächst die Fahrbahnen um die Breite der Randsteine erweitert um die gesamte Straßenbreite an die Konturen der Grünflächen anzugleichen. Danach konnten die Reliefs sowohl der Fahrbahnelemente wie Randsteinen und Unterführungen als auch der Grünflächen gestaltet werden. Um dem Zentrum des Kreisverkehrs - dem Sitz des Hauptgebäude des Antagonisten GLOBOTEX - optisch zu betonen wurde das Terrain zu einem Hügel erhöht und die Umliegenden kreisförmigen Grünstreifen mit einem schwungvollen radialen Querschnitt versehen. Des Weiteren wurden die Bahnen der Außenringstraßen, die unter den Hauptverkehrsachsen durchführen zu Unterführungen verbunden und Rampen konstruiert, die den Banker Bot-Charakteren erlauben, in das GLOBOTEX Gebäude hinein- und wieder herauszufahren (siehe Abbildung 19).

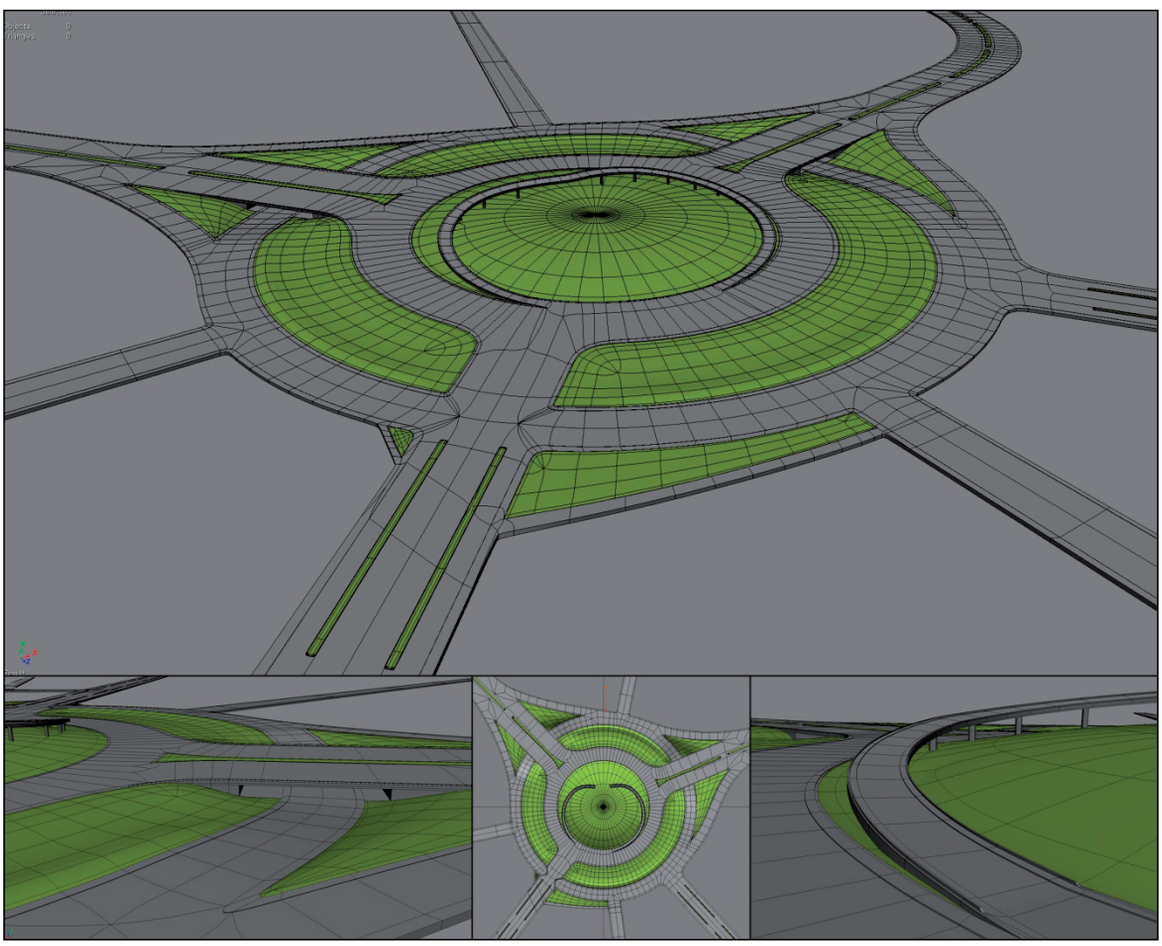


Abbildung 19: Modell des Kreisverkehrsystems mit finaler Topographie

Einen etwas unangenehmen Faktor bei der Gestaltung der Proportionen, insbesondere der Höhenlevels verschiedener Straßenelemente, stellte der fehlende Einheitenbezug von Autodesk Softimage dar, das in Softimage Units arbeitet und sich nicht global auf Einheiten wie etwa Meter oder Zentimeter einstellen lässt. Somit waren des öfteren Proportionsberechnungen erforderlich um einen Überblick über die Ausmaße des Modells zu erlangen.

In einem geschlossenen Softimage-Workflow wäre das *UV-Layout* und die Texturierung des fertig konstruierten Kreisverkehrmodells der nächste Bearbeitungsschritt. Im Falle dieses Projekt-Setup konnte das Texturing jedoch erst zu dem Zeitpunkt erfolgen, an dem die prozedural erzeugten Geometrien mit den händisch erstellten Modellen bereits zusammengefügt waren, da bis zu diesem Zeitpunkt keinerlei Referenz auf mögliche texturkritische Kreuzungs- und Einmündungspunkte der beiden Geometrien verfügbar war. Diese Informationen - resultierend aus den Polygonmustern der Kreuzungspunkte selbst, und ihrer durch die jeweilige Texturprojektion festgelegten *UVs* - waren jedoch unerlässlich um die Straßentexturen korrekt auszurichten. Zu diesem Zeitpunkt standen diese Kreuzungspunkte jedoch noch nicht fest und mussten erst im Zuge der Hochzeit der beiden Straßenmodelle per Hand in die manuell erzeugte Straßengeometrie eingearbeitet werden. In Rücksichtnahme auf den chronologischen Aufbau der Arbeit wird das Thema *Texturing des Hauptverkehrsknoten* erst nach dem Abschnitt 5.4 (Gebäude- und Straßengenerierung durch Rules) angereiht und behandelt.

Nach Fertigstellung des Kreisverkehrs wurden sämtliche Objekte des Kreisverkehrsystems in einem Softimage Model im Format .EMDL zusammengefasst und aus Softimage exportiert, um es im Anschluss in die finale Softimage Szene die alle Modelle vereint importieren zu können.

5.4 Gebäude- und Strassengenerierung durch Rules

In diesem Abschnitt wird die prozdurale Entwicklung des Stadtmodells thematisiert und der Weg von den ersten prozeduralen Modelingtests bishin zur finalen Lösung verdeutlicht. Hierbei wird der grundsätzliche systematische Aufbau des prozeduralen Codes erläutert, der hinter der Generierung des Modells steht. Da die Erläuterung des gesamten, für die Erstellung der Stadt verwendeten CGA Shape – Regelwerks, den Rahmen dieser Arbeit sprengen würde, befasst sich dieser Abschnitt im Wesentlichen mit dem generellen Aufbau des Rule-Systems und erklärt die Funktion des Codes anhand der bedeutendsten Regelzusammenhänge und Code-Phrasen der Rule *bot_buidlings.cga*. Der gesamte Code des Rule-File kann jedoch anhand der .CGA-Dateien auf der beigelegten Daten-CD nachgeschlagen werden.

Abbildung 20 zeigt das Interface der CityEngine mit geöffnetem Rule Editor. Der Rule Editor unterstützt Syntaxhervorhebung und deutet Fehler im Code durch farbige Kennzeichnungen in den entsprechenden Zeilen an, was die Übersicht über den Code erleichtert.

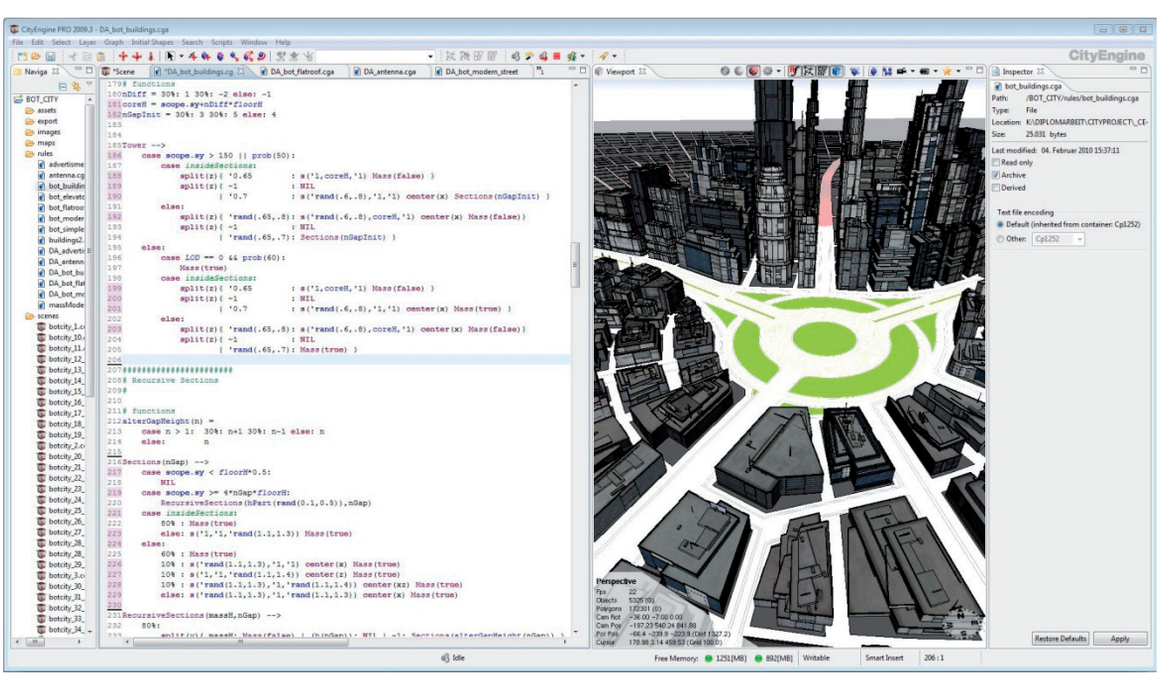


Abbildung 20: CityEngine Screenshot mit Rule Editor und Viewport

5.4.1 Erste Tests

Um einen ersten Einblick in die Funktion der CGA-Shape Grammar zu bekommen und um als 3D-Designer mit geringen Vorkenntnissen in Programmierung einen praktischen Zugang zur Programmierpsrache CGA-Shape zu erlangen, war es notwendig erste Tests durchzuführen. Hierzu wurde zu allererst ein simples Gebäudemodell programmiert und die Funktion von Massenmodellen erprobt. Abbildung 21 zeigt einen Screenshot der CityEngine in dieser Testphase.

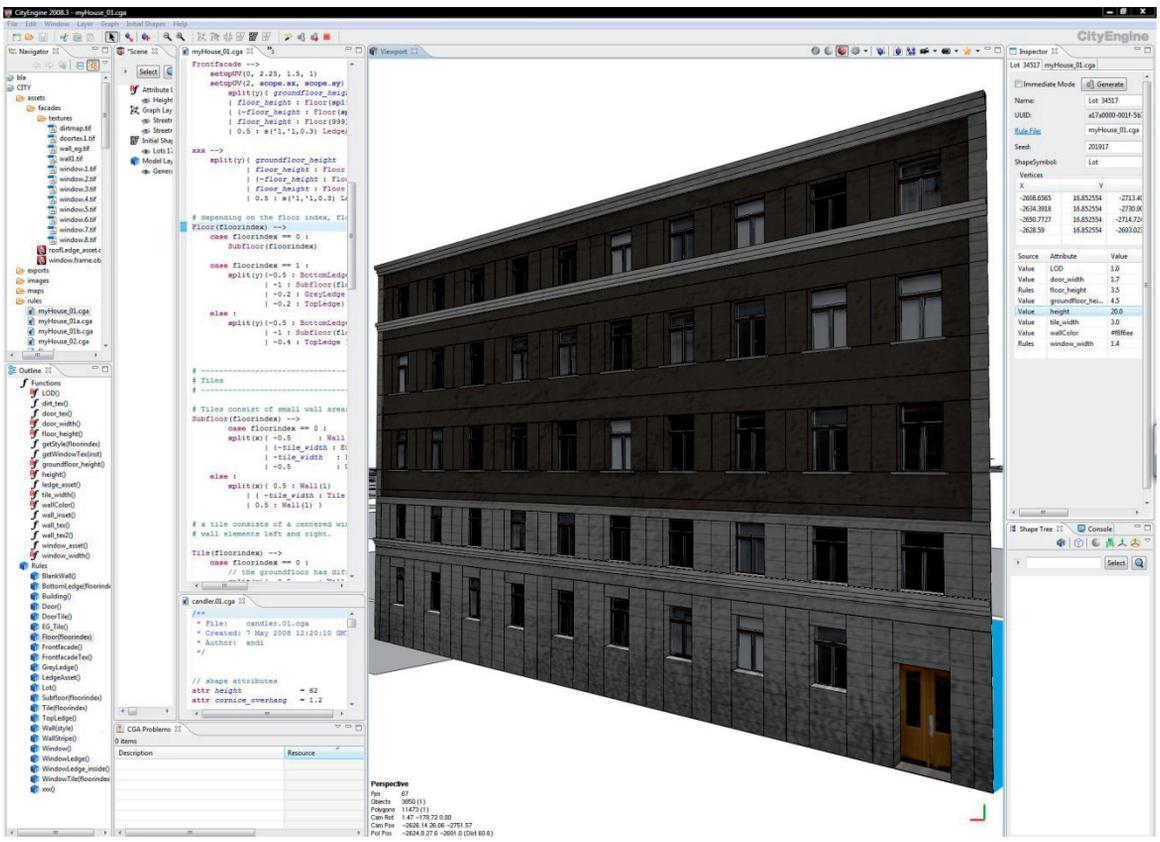


Abbildung 21: CityEngine Screenshot der ersten CGA-Tests

Zu diesem Zeitpunkt wurde noch mit der damals aktuellen CityEngine Version 2008.2 gearbeitet - mit der CityEngine Version 2009.1 wurde schließlich auch ein für diese Arbeit sehr praktisches Preset mitgeliefert, das für die Umsetzung des Stadtmodells übernommen und gemäß den Anforderungen dieses Projektes modifiziert wurde. Dieser Vorgang wird im nächsten Abschnitt genauer erläutert.

5.4.2 Aufbau des CGA-Rulesystem der finalen City

Wie bereits im letzten Abschnitt erwähnt wurde kam als Basis für das Regelwerk der Stadt ein Preset zur Anwendung, das mit der CityEngine Version 2009 mitgeliefert wurde. Da die durch das Preset NYC2159 produzierten Gebäudestrukturen in ihrer Form sehr gut zum Science Fiction Genre und zu den filmischen Anforderungen der Story des Kurzfilms passten, fiel die Entscheidung darauf, das Regelwerk dieses Presets in Grundzügen zu übernehmen und gemäß den Anforderungen der BotCity zu modifizieren, wodurch bei der Generierung des Stadtmodells viel Zeit gespart werden konnte.

Das Regelkonstrukt der Stadt setzt sich aus mehreren CGA Rule Files zusammen, die zum größten Teil miteinander verlinkt sind. Die Basis dieser Gebäudregeln bildet die Rule *bot_buildings.cga*. Sie kommt als erstes zum Einsatz und erzeugt die grundlegenden Formen der Gebäude. Weiters lädt die Regel am Anfang ihrer Ausführung die Regel *bot_flatroof.cga* für die Ausgestaltung der Hausdächer und die Regel *advertisments.cga* für das Anbringen von Werbeflächen. Die Regel *bot_flatroof.cga* lädt wiederum die Regeln *antenna.cga* (um Antennen auf den Dächern zu platzieren) und die Regel *advertisments.cga* (für Werbeflächen auf flachen Dächern). Tabelle 2 zeigt alle Rule Files und ihre Beziehung zueinander:

Rule File	Zweck	Import anderer Rules
bot_buidlings.cga	Basisregel für die Generierung der Gebäudeshapes	Importiert die Regeln bot_flatroof.cga und advertisments.cga
bot_flatroof.cga	Erzeugt Details wie Vorsprünge, Aufbauten oder Klimaanlagen auf den Dachflächen der Gebäude	Importiert die Regeln antenna.cga und advertisments.cga
antenna.cga	Setzt Antennen auf die Dächer von Häusern mit LOD=1	X
advertisments.cga	Setzt Werbeflächen auf Dächer von Gebäuden mit LOD=1	X
bot_modern_streets.cga	Regelwerk der Straßengeometrie	alleinstehende Regel

Tabelle 2: Liste der Rule Files

Die Produktion der Häuser beginnt wie bereits oben erwähnt mit der Regel bot_buildings.cga. Nach dem Produktionsstart dieses Rule Files werden (zuerst) weitere Rules Files - bot_flatroof.cga und advertisments.cga – geladen.

Anschließend werden die Attribute für die Regel festgelegt. Diese Attribute liegen als editierbare Parameter im Inspektor-Bereich der CityEngine-GUI an und beschreiben Parameter wie Gebäude- und Geschoßhöhe, Fassadentyp und Fassadentextur. Weiters kann bestimmt werden ob Gebäude Balkone oder Werbeschilder (Advertisements) besitzen. Das Attribut *finalBuildingH* beschreibt beispielsweise die Gebäudehöhe mittels unterschiedlicher Random-Funktionen in Abhängigkeit zur Grundstücksgröße – Gebäude auf extrem großen Grundstücken über 5000 m² werden große Gebäudehöhen zugewiesen, sehr kleine Lots (<600m²) haben dagegen kleine Gebäudehöhen zur Folge; für alle Grundstücksflächen innerhalb dieses Wertebereichs von 600 m² bis 5000 m² gilt das Attribut *buildingH,* das bei Verwendung einer Height map von selbiger kontrolliet wird:

```
attr buildingH = rand(40,400)

attr finalBuildingH =
        case geometry.area > 8000 :
                rand(buildingH*0.6,buildingH*1.2)
        case geometry.area > 5000 :
                rand(buildingH*0.4,buildingH)
        case geometry.area < 600 :
                rand(buildingH*0.15,buildingH*0.3)
        else:
                rand(30,buildingH*0.6)
```

Häuser mit einer annähernd rechteckigen Grundstücksform verwenden in diesem Fall Polygon-Primitives im Format .OBJ als Initial Shapes), anstatt nur auf die extrudierte Grundstücksfläche zurückzugreifen. Diese importierte polygonale Grundform bildet an dieser Stelle den Ausgangspunkt für die Ausführung der Rules. Das jeweils für ein Gebäude benutzte Volume wird durch das Attribut *volume* festgelegt:

```
attr volume =
        2%  : „volumes/cube1_bevel_round_sides.obj“
        1%  : „volumes/cube1_bevel_round_top.obj“
        2%  : „volumes/cube2_bevel_simple_sides.obj“
        5%  : „volumes/cube3_bevel_cut_sides.obj“
                [...]
        1%  : „volumes/cylinder1_top.obj“
        3%  : „volumes/cylinder2.obj“
        1%  : „volumes/cylinder2_mod.obj“
        1%  : „volumes/o_shape.obj“
        1%  : „volumes/prism1.obj“
        3%  : „volumes/u_shape.obj“
        3%  : „volumes/u_shape_mod1.obj“
        2%  : „volumes/u_shape_mod2.obj“
        else: „volumes/cube3_bevel_cut_sides.obj“
```

Die Liste der bereits im Preset vorhandenen .OBJ Initial Shapes wurde um etliche Objekte erweitert, um eine größere Hetrogenität der Gebäudeformen zu bewirken (siehe Abbildung 22). Weiters wurden zahlreiche Attribute und Parameter des Presets abgeändert um die Proportionen der Gebäude den Anforderungen der Story anzugleichen.

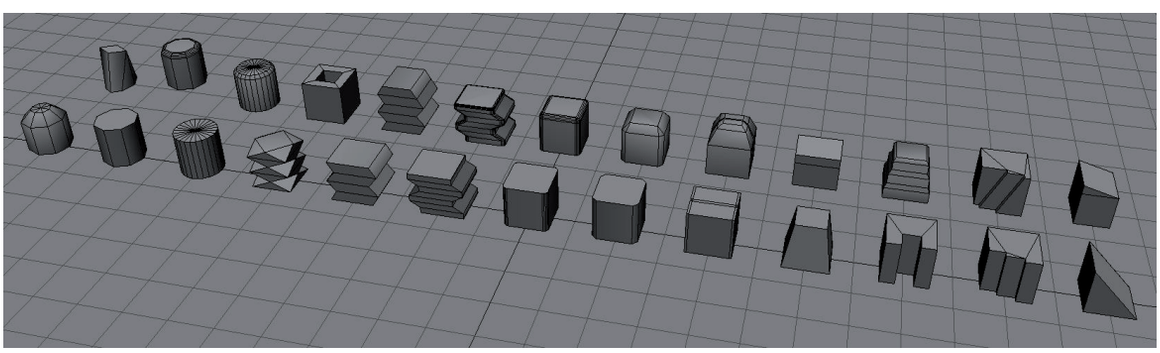


Abbildung 22: Initial Shape Objekte (in Softimage)

Nach der Etablierung der Attribute werden verschiedenste generelle Funktionen definiert, die beispielsweise Schwankungsbreiten von Random-Funktionen oder Wahrscheinlichkeiten bestimmen, oder Werte auf Integer-Zahlen runden. Folgende Funktion ermittelt beispielsweise die auf gerade Zahlen gerundete Höhe einer beliebigen Anzahl an Stockwerken:

```
h(n) = rint(n)*floorH
```

Auf die Definition der Funktionen folgt schließlich die eigentliche regelbasierte Produktion der Gebäudegeometrie, aufbauend auf verschiedensten Rules. Die erste Regel die zur Anwendung kommt ist die Rule *Lot*. Sie übernimmt die im Zuge der Shape Generation Phase erstellten Grundstücksflächen und erstellt auf der Fläche des jeweiligen *Lots* einen im Faktor 0,95 verkleinerten Grundriss des Gebäudes und übergibt ihn der Regel *Footprint*:

```
Lot -->
        s('0.95,‘1,‘0.95) center(xz) Footprint
```

Die Regel Footprint erzeugt anschließend die ersten dreidimensionalen Initial Shapes die Volumen besitzen. Footprints mit zu geringer Grundfläche oder falschen Proportionen werden durch die zuvor definierten Attribute *isVeryThin* und *isTooSmall* von der Produktion ausgeschlossen (*NIL*). Eine Fallunterscheidung bewirkt, dass bei annähernd rechteckigen Grundrissen, die aus den Footprints extrudierten Körper durch die oben gezeigten Volume ersetzt werden. Nicht rechteckige Grundrisse führen nur zu einer einfachen Extrusion:

```
Footprint -->
        case isVeryThin:        NIL
        case isTooSmall:        NIL
        case isRect && prob(90) && !isThin && !isVeryThin :
                extrude(finalBuildingH)
                i(volume)
                s('.85,h(nFloors),‘.85) center (xz)
                Eval(facadeType)
                Eval(getWallTexNr)
                Envelope
        else:
                extrude(finalBuildingH)
                s('1,h(nFloors),‘1)
                Eval(facadeType)
                Eval(getWallTexNr)
                Envelope
```

Die dadurch entstande Regel *Envelope* hat die Funktion, die vorliegenden 3D-Körper ab einer definierten minimalen Grundfläche von 5000 m² mittels Split-Funktionen in vertikaler Richtung in kleinere Teilezu zerlegen und diese Abschnitte unterschiedlich stark in X- und Z-Achse zu skalieren. Dadurch entsteht die Basis für die charakteristischen Gebäudeformen dieser Stadt. Das Ergebnis der Regel *Envelope* wird daraufhin der Regel *Tower* übergeben.

```
Envelope -->
        case scope.sx*scope.sz > 5000:
                13%:   split(y){'0.65: Tower
                                | ~4: s('0.85,‘1,‘0.85) center(xz) Tower
                                | ~1: s('0.65,‘1,‘0.65) center(xz) Tower }
                20%:   split(y){'rand68: Tower
                                | ~rand(0.4,1.6): s('0.9,‘1,‘0.9) center(xz) Tower
                                | ~1: s('0.7,‘1,‘0.7) center(xz) Tower }
                17%:   split(y){'rand48: Tower
                                | ~1: s('rand79,‘1,‘rand79) center(xz) Tower }
                15%:   split(y){'rand48: Tower
                                | ~1: t('rand03,0,‘rand(0.1,0.6)) s('0.7,‘1,‘0.7) Tower }
                else:  Tower
        else:  Tower
```

In weiterer Folge kommen eine Vielzahl weiterer Regeln und Split-Algorithmen zum Einsatz, um den Gebäuden ihre finale Form zu geben und den Grad an Varianz und Detail in die Höhe zu schrauben. Um zwischen einer Gebäudeversion mit vielen Details und einer abgespeckten Variante wählen zu können, werden manche Algorithmen nur aufgerufen wenn das Attribut *LOD* (*Level Of Detail*) auf 1 gesetzt ist, d.h. die Standardeinstellung *LOD* = O bewirkt eine Auslassung diverser Regeln dieses Rule Files.

Im Zuge der Anpassung des Presets wurden auch einige Regeln aus dem Code entfernt. Um das Stadtbild im Sinne des Drehbuchs anzugleichen wurden etwa anfangs im Preset implementierte Aufzugsanlagen an den Fassaden der Gebäude eliminiert.

Um die Oberflächen der Gebäude zu gestalten wurden 21 neue Texturen angefertigt und in das CityEngine-Projekt eingegliedert. Für die Fertigung der Texturen wurden Fotografien von Hochhäusern und Industrieanlagen herangezogen, die danach perspektivisch entzerrt und auf *Tiles* mit einem Raster von 16 Stockwerken zu 16 Fensterreihen angeordnet wurden. Pro Textur wurden zwei Varianten entwickelt - eine A-Version mit verstreuten Fenstern und größeren Fassadenflächen sowie eine B-Variante die rein aus aneinandergereihten Fensterelementen besteht. Bei der Ausführung der Regel bot_buildings.cga werden beide Texturvarianten nach einem festgelegten Schema auf einem Gebäude verteilt. Während Variante A über den Gebäudekorpus projiziert wird, ist Variante B den Vorsprüngen des Gebäudes vorbehalten. Abbildung 23 zeigt einige der erstellten Texuren.

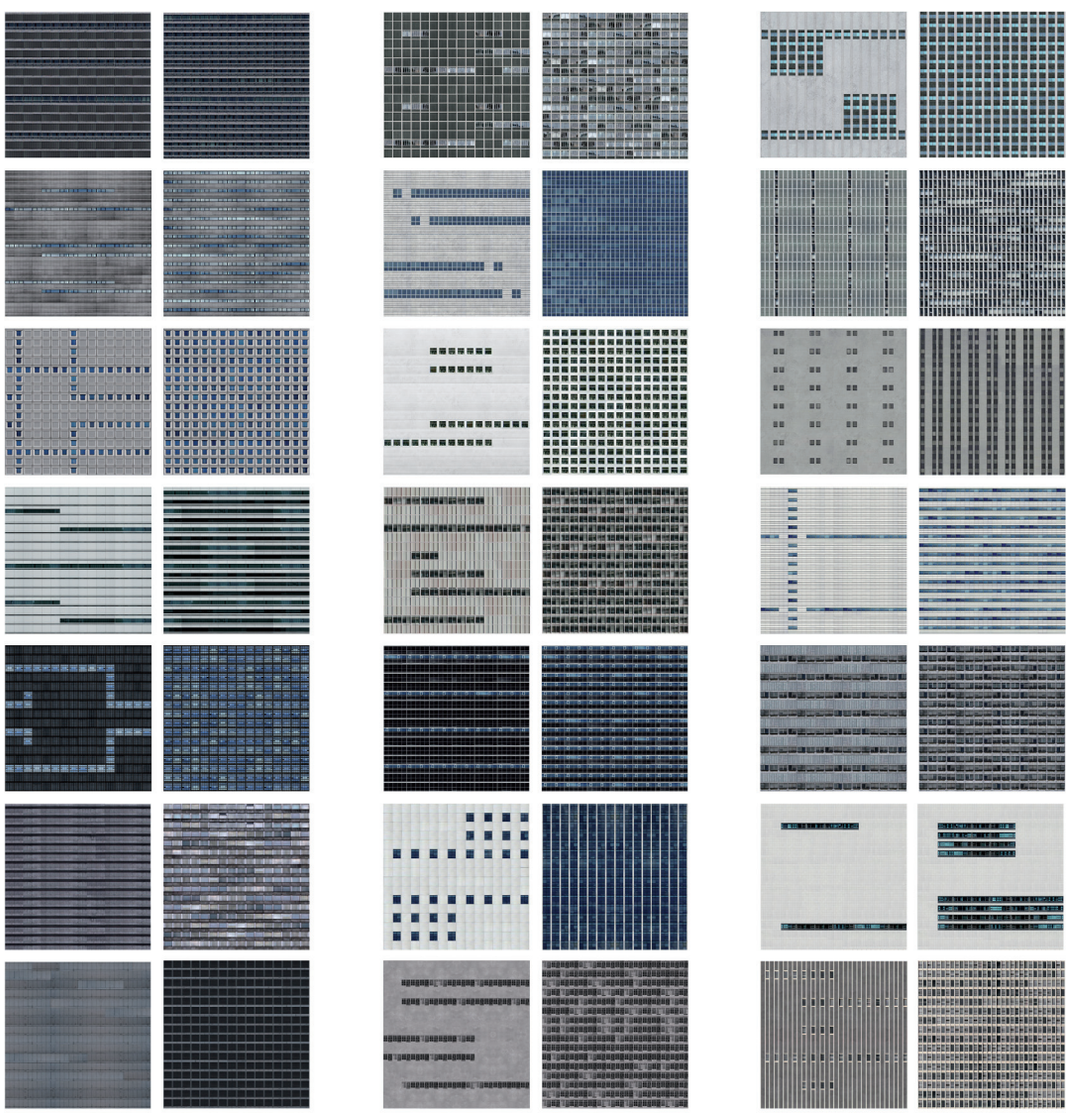


Abbildung 23: Gebäudetexturen - pro Reihe: links: A, rechts B

5.5 Export

Der Export der generierten Geometrie bildet in den meisten Fällen den letzten Schritt des CityEngine Workflows. Die aus der CityEngine exportierten 3D-Objekte können darauf hin in die 3D-Applikation importiert werden, in der sämtliche Komponenten aus diversen Applikationen zu einer finalen Szene zusammengesetzt werden; im Falle dieser Arbeit ist dies das Software Paket Softimage aus dem Hause Autodesk.

Die CityEngine unterstützt die Exportformate Wavefront .OBJ, Autodesk .FBX, Autodesk .3DS, Collada .DAE, Renderman .RIB, Mental Ray .MI und den Export ins Massive Dateiformat .MAS. Der Objekttransferstandard Wavefront .OBJ ist hierbei der Dateistandard mit der größten Kompatibilität zu nahezu allen 3D Applikationen. Abbildung 24 zeigt sämtliche Import- und Exportmöglichkeiten der CityEngine (vgl. CityEngine Manual, S. 210).

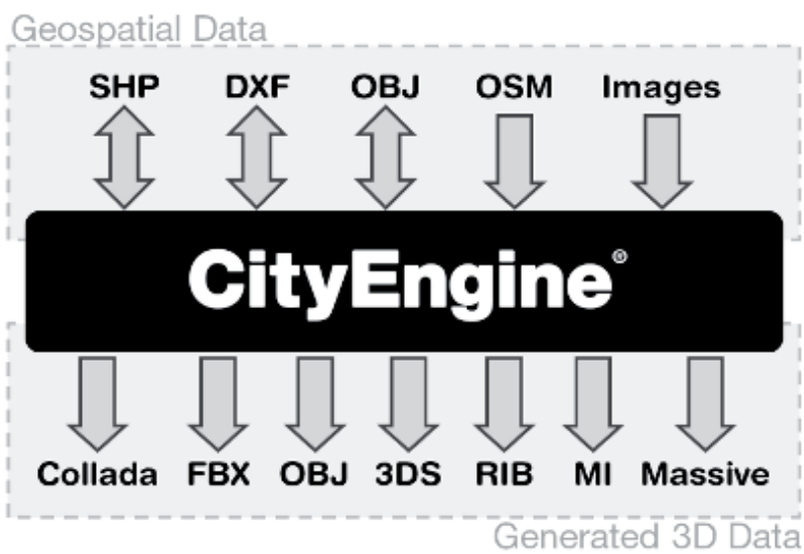


Abbildung 24: Import- und Exportformate der CityEngine

Für den Export der Geometrie im Rahmen dieser Fallstudie stellte sich das Format Wavefront .OBJ nach etlichen Export-Testläufen als das geeignetste Format heraus, da es einerseits beim Export der Geometrie die Dateien mit der geringsten Größe erzeugte und andererseits den Export vergleichsweise stabil und schnell abwickelte.

Um die Datenmengen von ungefähr zehn Millionen Polygonen der Gebäudemodelle und weiteren elf Millionen Polygonen der Strassengeometrie aus der CityEngine heraus- und in Autodesk Softimage hineinzubekommen, ohne Programmabstürze zu verursachen, wurde das prozedurale Stadtmodell in Etappen exportiert. Hierzu wurden die sieben Stadtgebiete, die voneinander durch die sternförmig zusammenlaufenden Hauptstraßen separiert werden, einzeln nacheinander exportiert und mit Buchstaben von A bis G gekennzeichnet. Dennoch ergaben sich beim Import von zwei der Stadtviertel in Autodesk Softimage Probleme; der Importvorgang brach in diesen Fällen stets bei den selben Stellen der Fortschrittsanzeige unvermittelt ab und verursachte einen Absturz von Autodesk Softimage. Dieses Problem konnte schließlich umgangen werden indem die .OBJ Dateien der beiden betroffenen Stadtgebiete zuerst in Autodesk 3ds Max importiert und anschließend aus 3ds Max wiederum als .OBJ Dateien exportiert wurden. Nach diesem Prozess ließen sich die beiden Dateien problemlos in Autodesk Softimage importieren. Die Geometrie der Straßen wurde ebenfalls in Teilstücken in Softimage eingefügt, was in diesem Fall aber ohne Probleme funktionierte.

Nach dem vollständigen Import der CityEngine 3D-Geometrie beanspruchte die 3D-Szene von Autodesk Softimage, unter Einbeziehung der im Programm manuell erzeugten Straßenzüge, einen Speicherplatz von rund 350 Megabyte. Diese Menge an 3D-Information stellte sehr hohe Ansprüche an die Leistung des Computersystem, insbesondere die Echtzeitdarstellung und Navigation innerhalb der Softimage Szene erwies sich als äußerst träge. Durch den Wechsel zu einer NVIDIA Quadro FX 4800 Grafikkarte, und die dadurch erreichte Verdreifachung des GPU Speichers, konnte dieses Problem letztlich gelöst werden.

5.6 Zusammenführung der Geometrietypen in Softimage

Nachdem sowohl die prozedural generierte Geometrie als auch die in Autodesk Softimage erstellten Objekte fetiggestellt waren konnten die Modelle in Softimage zusammengelegt werden. Als erstes wurden die in der CityEngine generierten Gebäude in mehreren Etappen in eine neue Softimage Szene geladen. Im Abschnitt 5.5 (Export) wurde bereits erwähnt, dass der Importprozess in Softimage bei zwei Stadtteilen zu Abstürzen führte. Dieses Problem konnte jedoch umgangen werden, in dem die betreffenden .OBJ Dateien erst in Autodesk 3ds Max importiert und wieder exportiert wurden. Die Gebäudemodelle wurden in Softimage wiederum zu einem Softimage Modell im Format .EMDL umgewandelt und gesichert. Der nächste Schritt bestand im Import der Straßengeometrie sowie der *Lots*, die ebenfalls zu separaten Softimage Models umgewandelt wurden. An dieser Stelle konnte nun der in Softimage modellierte Kreisverkehr eingesetzt werden. Aufgrund des fehlenden Einheitenbezugs von Softimage besaß der Kreisverkehr einen falschen Maßstab und musste deshalb erst an die Größe der CityEngine Modelle angeglichen werden. Abbildung 25 zeigt eine Ansicht der fertig in Softimage importierten Objekte.

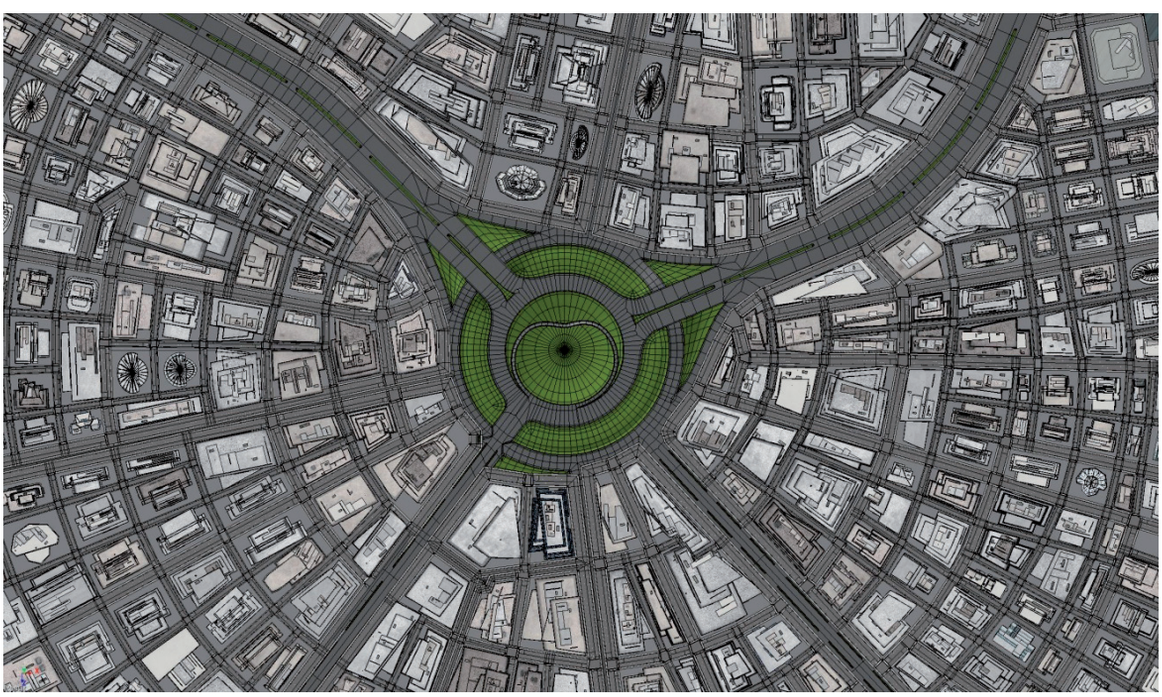


Abbildung 25: Abbildung X: Top-Ansicht der importierten Geometrie in Softimage

Wie in der Abbildung 25 (S. 67) ersichtlich ist, werden die Fahrbahnen des Kreisverkehrelements noch von den temporär angelegten Straßen überdeckt. In Kapitel 5.1.1 Erster Lösungsansatz und resultierende Probleme, (S. 46). und Kapitel 5.1.2 Finaler Lösungsansatz, (S. 48) wurde bereits auf die Problematik hingewiesen, dass die Bildung von Lots davon abhängt, ob der Bereich von einem geschlossenen Straßenverband umgeben ist. Diese als Workaround in der CityEngine angelegten temporären Straßen mussten nun händisch in Softimage entfernt werden, um die verbliebene Straßengeometrie mit dem händisch erzeugten Kreisverkehr zu verbinden.

Um nach diesem Arbeitsschritt die Objekte zusammenführen zu können, wurden zunächst die Hauptstraßen des Hauptverkehrsknoten bis an den Stadtrand verlängert und ihr Verlauf den prozeduralen Straßen in horizontaler und vertikaler Translation angepasst. Somit konnten nun die Enden der Nebenstraßen händisch an die Hauptstraßen herangezogen werde, bis die Geometriekanten deckungsgleich ausgerichtet waren (siehe Abbildung 26).

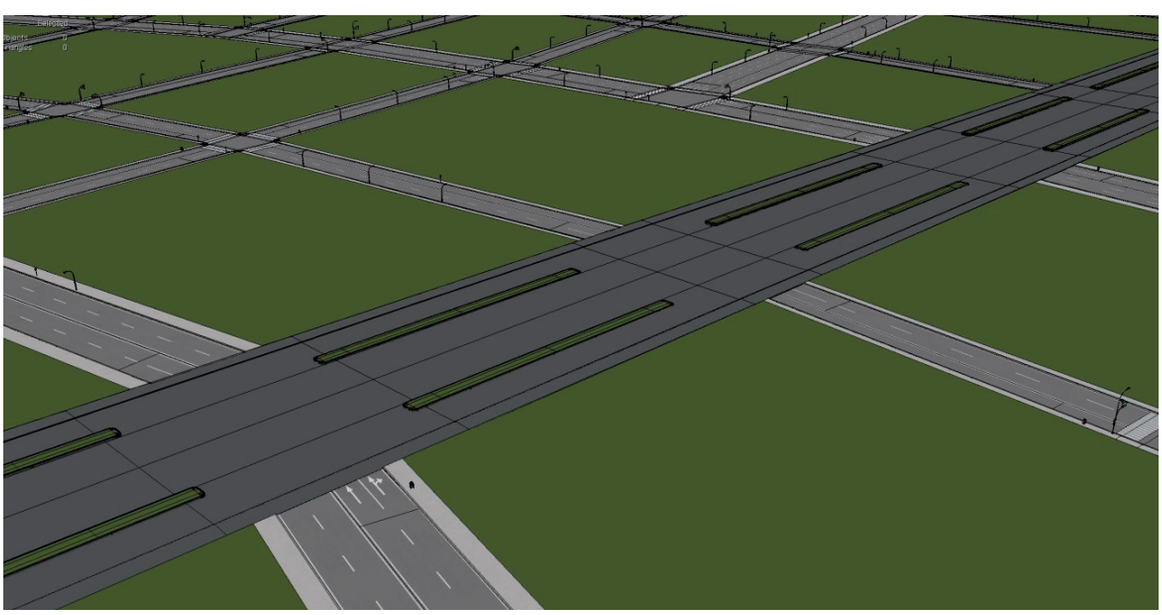


Abbildung 26: Deckungsgleiche Ausrichtung von Nebenstraßen

Da das Kreisverkehrmodell mangels Information über die Einmündungspunkte der Nebenstraßen bis zu diesem Stadium noch über keine Kreuzungen verfügte, mussten Kreuzungen per Hand in die Geometrie in tegriert werden.

Hierzu wurden die Hauptstraßen an den entsprechenden Stellen tesseliert und die Randsteine durchbrochen. Im Anschluss wurden alle Geometriepunkte der Nebenstraßen mittels Snapping an die Position der entsprechenden Punkte des tesselierten Kreuzungsbereiches auf der Hauptstraßengeometrie auf Deckung gebracht.

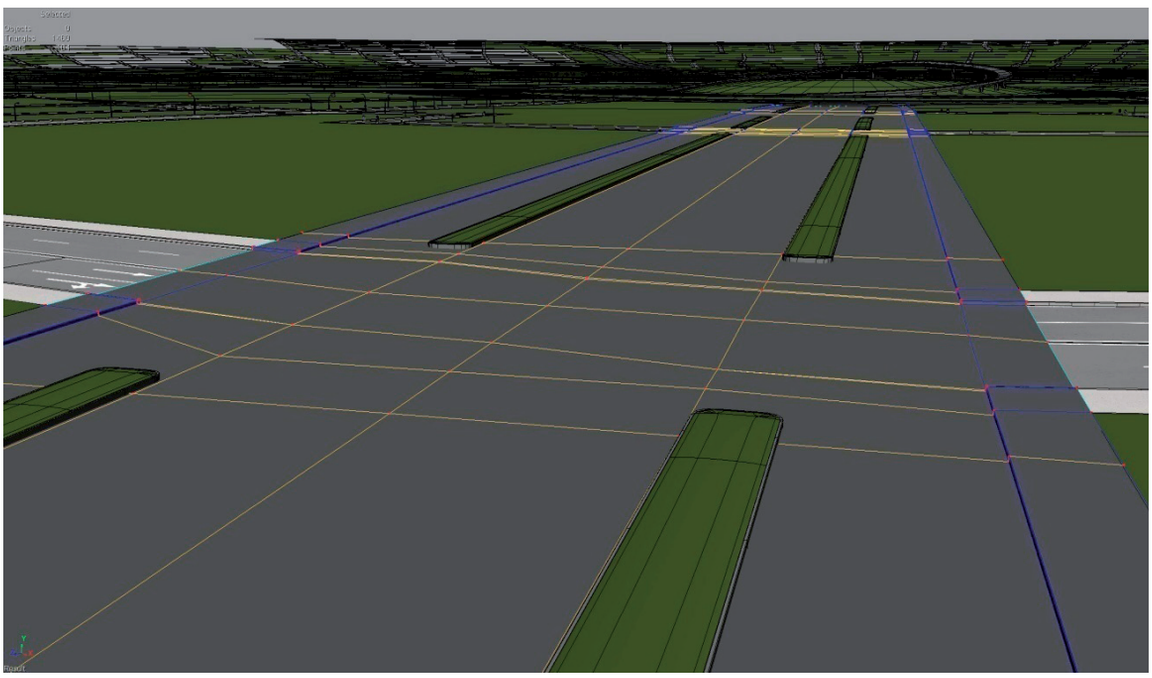


Abbildung 27: Aufbau einer typischen handgefertigten Kreuzung

Aufgrund des enormen Aufwandes der durch die manuelle Zusammenführung der Geometrien und das händische Modeling der Kreuzungspunkte verursacht wurde, fiel die Entscheidung diese Bearbeitung nur in zentrumsnahen Gebieten durchzuführen, die auch in Einstellungen des Films detaillierter zu sehen sind. Bei den übrigen, weiter außen liegenden und im Film nur von großer Entfernung sichtbaren Arealen, konnte in der Texturierungsphase durch die gezielte Einfärbung der kreuzungsnahen Randsteine mit Straßentextur der Eindruck einer voll ausmodellierten Kreuzung vermittelt werden, ohne tatsächlich jede Kreuzung manuell anzupassen zu müssen. Bei Kreuzungen am Stadtrand, die nur aus sehr großer Entfernung dargestellt werden mussten konnte auf alle Arbeitsschritte, außer der Angleichung der Nebenstraßen an die Hauptstraßen, verzichtet werden.

5.7 Manuelles Texturing des Hauptverkehrsknotens

Wie bereits im Abschnitt 5.3 einleitend beschrieben konnte das Texturing des Hauptverkehrsknotens erst nach der vollständigen Zusammenführung von prozedural generierter und händisch erstellter Geometrie erfolgen. Dieser Abschnitt schildert den Weg vom ungeshadeten Modell bis zur Implementierung einer approbaten Straßenttextur.

Der erste Arbeitsschritt bestand im Aufbau des *UV-Layout*, d.h. der komplexe dreidimensionale Polygonkörper des Kreisverkehrs musste im Texture Editor von Softimage so zu einer zweidimensionalen Form aufgefalten werden dass keine *Seams* entstehen, die später bei Einstellungen des Films sichtbar werden könnten. Vor der eigentlichen Texturierung wurden das Polygonmesh des Kreisverkehrs von den einmündenden Straßen getrennt. Dadurch war es möglich sowohl den Kreisverkehr selbst, als auch die Gesamtheit der anschließenden Hauptstraßen mit einer eigenen Projektion zu versehen, um in Folge Straßen und Kreisverkehr zwei individuelle UV-Layouts sowie separate Texturen zu verleihen.

Im Fall des Kreisverkehrs selbst bot sich für die Erzeugung der UVs die Anwendung einer Planaren XZ-Projektion an, die nachträglich um diejenigen Flächen erweitert wurde, die nicht auf einer waagrechten Ebene verlaufen, und eine visuelle Bedeutung für den finalen Film haben. Bei allen Polygonen, die in keinem Shot tatsächlich zu sehen sind wurde auf diesen Arbeitschritt verzichtet. Beim UV-Layout der Hauptstraßen konnte der seit der Version 7.5 in Softimage integrierte Unfold-Alghorithmus zur Erzeugung der UVs herangezogen werden, welcher sehr schnell gute Ergebnise lieferte. Die dadurch im Texture Editor vorliegenden *UV-Islands* konnten nun für jede der beiden Geometriegruppen zu UV-Layouts organisiert werden. Der wichtigste Faktor dabei war, so wenig Platz wie möglich auf der Texturfläche unbenutzt zu lassen, um einerseits die Qualität der Textur zu optimieren und andererseits den durch die gesamte Textur benutzten Hauptspeicher möglichst gut auszunutzen (siehe Abbildung 28).

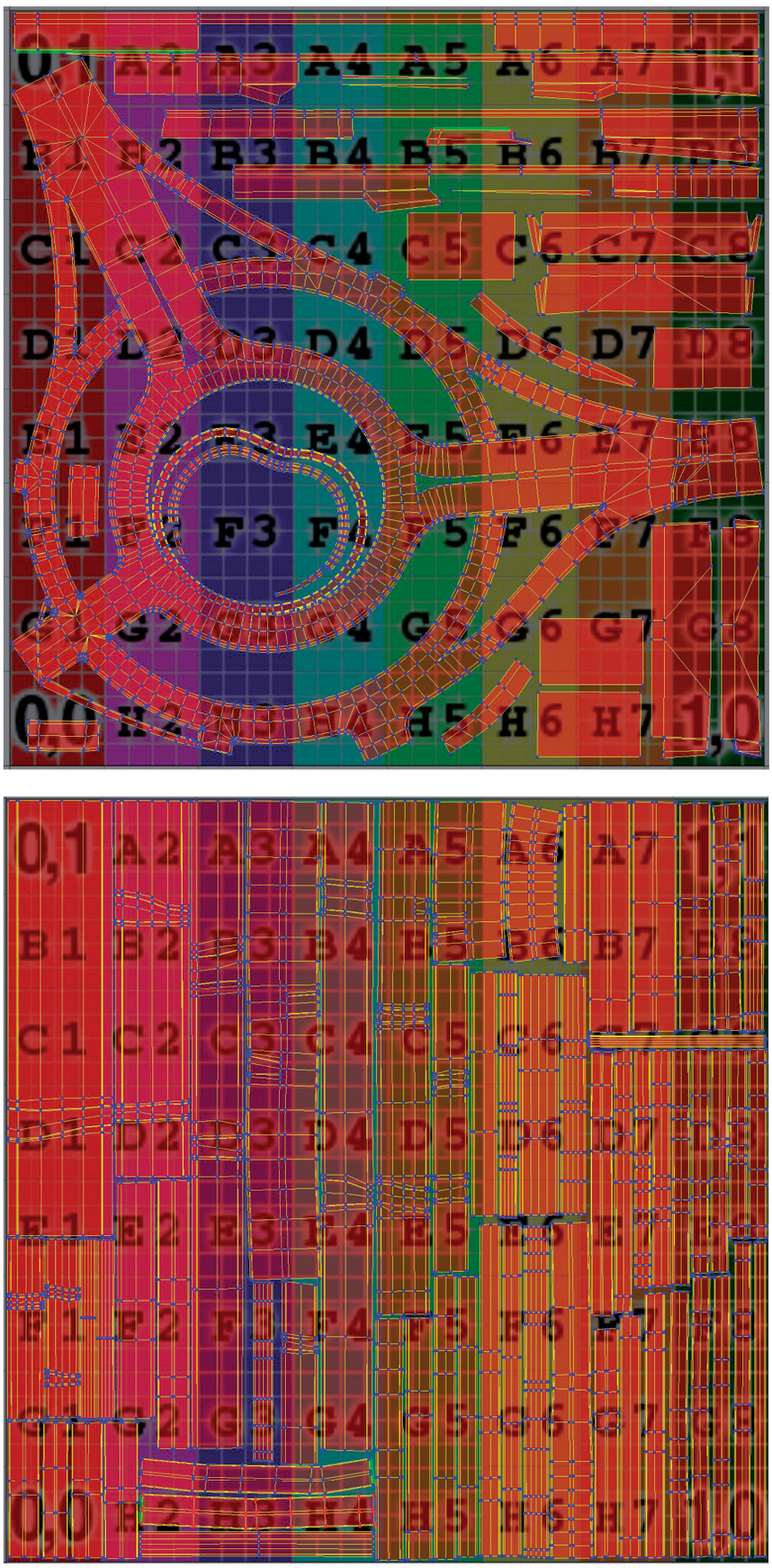


Abbildung 28: UV-Layout des Kreisverkehrs und der Hauptstraßen

Die UV-Meshes der fertigen UV-Layouts wurden anschließend - für den Export extra mit einem einfärbigen Hintergrund ausgestattet- vom Softimage Texture Editor aus in eine Grafikdatei im Format .JPEG exportiert (*Texture Stamping*), die daraufhin in Adobe Photoshop importiert werden konnte. Nachdem der Hintergrund der Grafik mittels Colorfiltering entfernt wurde, blieb die Gitterstruktur des UV-Mesh als Photoshop-Layer zur Orientierungshilfe für die Texturierung bestehen.

Um die Basistexturen des Hauptverkehrsknotens zu erzeugen wurden dieselben Texturelemente verwendet, die auch bei der prozeduralen Straßengenerierung in der CityEngine zur prozeduralen Texturierung herangezogen wurden. Somit konnte sichergestellt werden, dass die beiden Straßenmodelle von ihrem Erscheinungsbild her nahtlos zusammenpassen. Die entsprechenden Texturteile aus der Texture Library des CityEngine Projekts wurden in Adobe Photoshop so kombiniert und composed, dass sich verschiedene Teilstücke ergaben, mit denen sich verschiedenste Straßentexturen mit unterschiedlichen Breiten und Fahrbahnmarkierungen erzeugen ließen.

Bei der Texturierung der Fahrbahnmarkierungen des Kreisverkehrs gestaltete sich der Sachverhalt etwas komplizierter als bei den Hauptstraßen Um die runden und geschwungenen Markierungen des Kreisverkehrs glaubwürdig erscheinen zu lassen mussten diese Elemente in Adobe Illustrator auf Vektorbasis erstellt werden und nachträglich mit der Grundtextur in Adobe Photshop kombiniert werden. An dieser Stelle muss noch erwähnt werden, dass Texturen dazu neigen sich zu verziehen, wenn mit Subdivision Surfaces gearbeitet wird und die gebogenen Polygonformen nicht genug tesseliert sind. Da der gesamte Kreisverkehr seine stufenlose, runde Form durch den Einsatz von Subdivision Surface Modeling erhielt, kam es an den Kanten der Randsteine öfters zu Texturzerrungen. Diesem Effekt konnte nur entgegengewirkt werden, indem die betroffenen Stellen des Kreisverkehrs mit mehr Tesselierungen beziehungsweise Polygon Faces ausgestattet wurden.

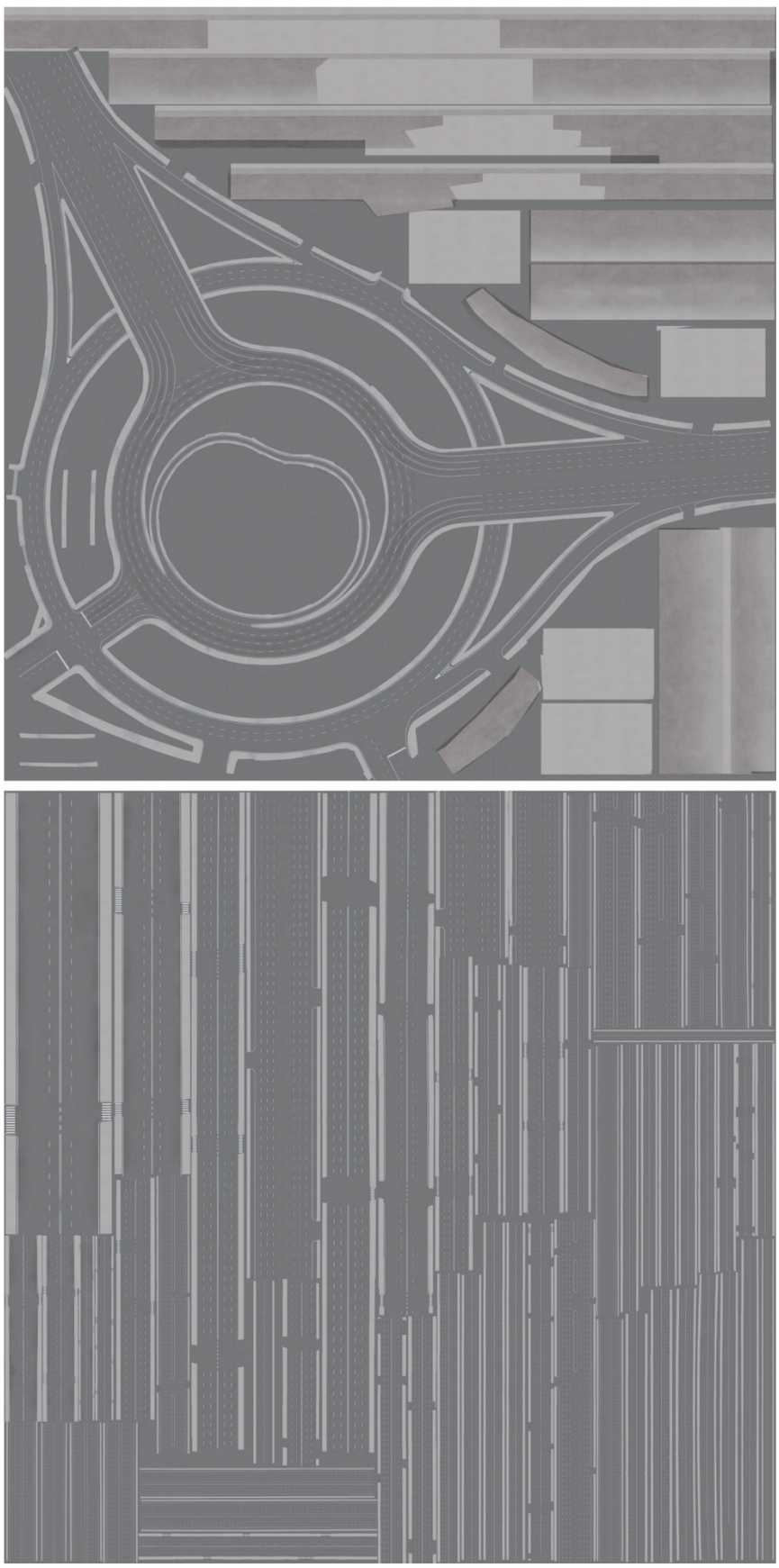


Abbildung 29: Basistextur des Kreisverkehrs und des Straßennetzes

Das Format der Texturen beläuft sich in beiden Fällen auf eine Bitmap von 8K, also 8192 mal 8192 Pixel. Diese Abmessungen stellten sich als der minimale Detailgrad heraus, mit dem feine Strukturen wie Fahrbahnmarkierungen noch in akzeptabler Qualität wiedergegeben werden können.

In weiterer Folge des Projektes ist angedacht in den Shots, die einen Bildausschnitt nahe des Bodens aufweisen, mit hochdetaillierten Texture-Overlays zu arbeiten, die auf einem transparenten Grid gelagert über der Basistextur liegen. Weiters ist geplant, durch dein Einsatz mehrerer prozeduraler Dirtmaps, welche über der gesamten Geometrie liegen, den Realitätsgrad der Straßenstrukturen durch die beimischung von übergelagertem Schmutz zu optimieren.

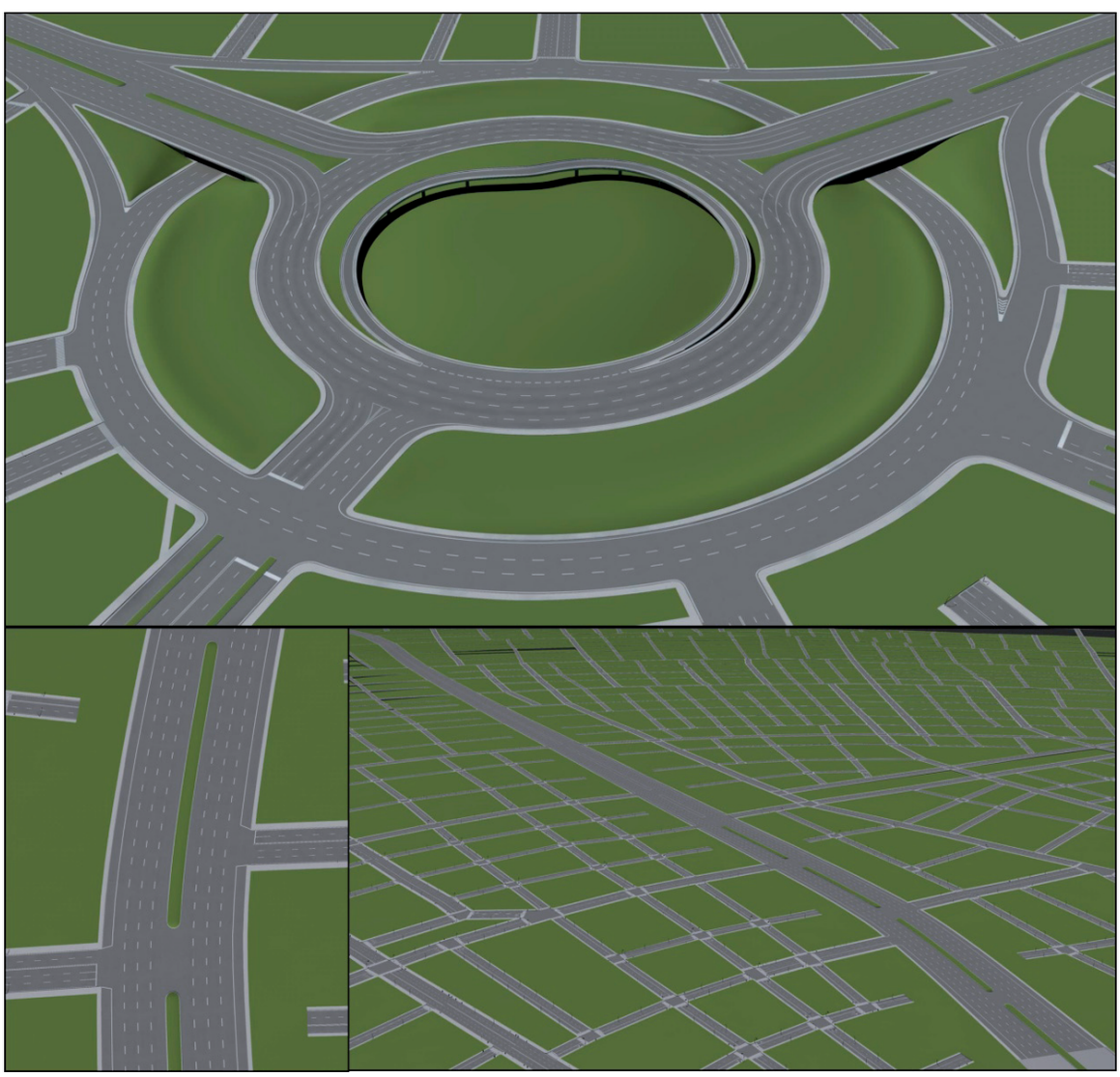


Abbildung 30: Texturen Straßennetz und Kreisverkehr im finalen Zustand

5.8 Resume und Resultate

Die Umsetzung des dreidimensionalen Stadtmodells mit Hilfe des prozeduralen Lösungsansatzes der CityEngine kann im Rückblick als erfolgreich bezeichnet werden. Durch die Adaptierung des Stadt-Presets konnte die Stadt in äußerst kurzer Zeit aufgebaut werden, was durch die Verwendung klassischer Modellierungstechniken, wie sie in Softwarelösungen wie Autodesk Maya, 3ds MAX oder Softimage zum Einsatz kommen, nicht annähernd so schnell möglich gewesen wäre. Als weiterer großer Vorteil erwies sich die Möglichkeit, einzelne Gebäude durch gezielte Beeinflussung von Attributen mit Hilfe des CityEngine Inspectors an die Anforderungen der filmischen Geschichte anpassen und beinahe in Echtzeit mit abgeänderten Eigenschaften neu generieren zu können.

Als besonders kritischer Punkt stellte sich dagegen die prozedurale Modellierung des Kreisverkehrs heraus. Die komplexe Grundform dieses Hauptverkehrsknotens konnte nicht mit Hilfe prozeduraler Techniken hergestellt werden und wurde daher händisch modelliert, texturiert und anschließend mit dem prozeduralen Modell vereinigt, was einen erheblichen Arbeitsaufwand nach sich zog. Auch der Export der generierten Objekte verlief nicht ganz reibungslos; ein Teil der Geometrie konnte erst nach dem Import der Daten in Autodesk 3ds MAX und anschließendem Export in die Zielsoftware Autodesk Softimage eingegliedert werden.

Das Erlernen der Programmiersprache CGA Shape Grammar stellte anfangs mangels Programmierkenntnissen ebenfalls eine Herausforderung dar, es ist jedoch auch Gestaltern ohne - oder mit nur wenig - Erfahrung im Programmieren nach einiger Zeit der Beschäftigung mit der Materie möglich, die Arbeitsweise des Systems und den Syntax des Codes zu verstehen.

Die Abbildungen auf den folgenden Seiten zeigen das in die finale Softimage Szene integrierte prozedurale Stadtmodell in verschiedenen Stadien. Diese Einzelbilder wurden zu Testzwecken in Autodesk Softimage mit Mental Ray v3.7 gerendert.

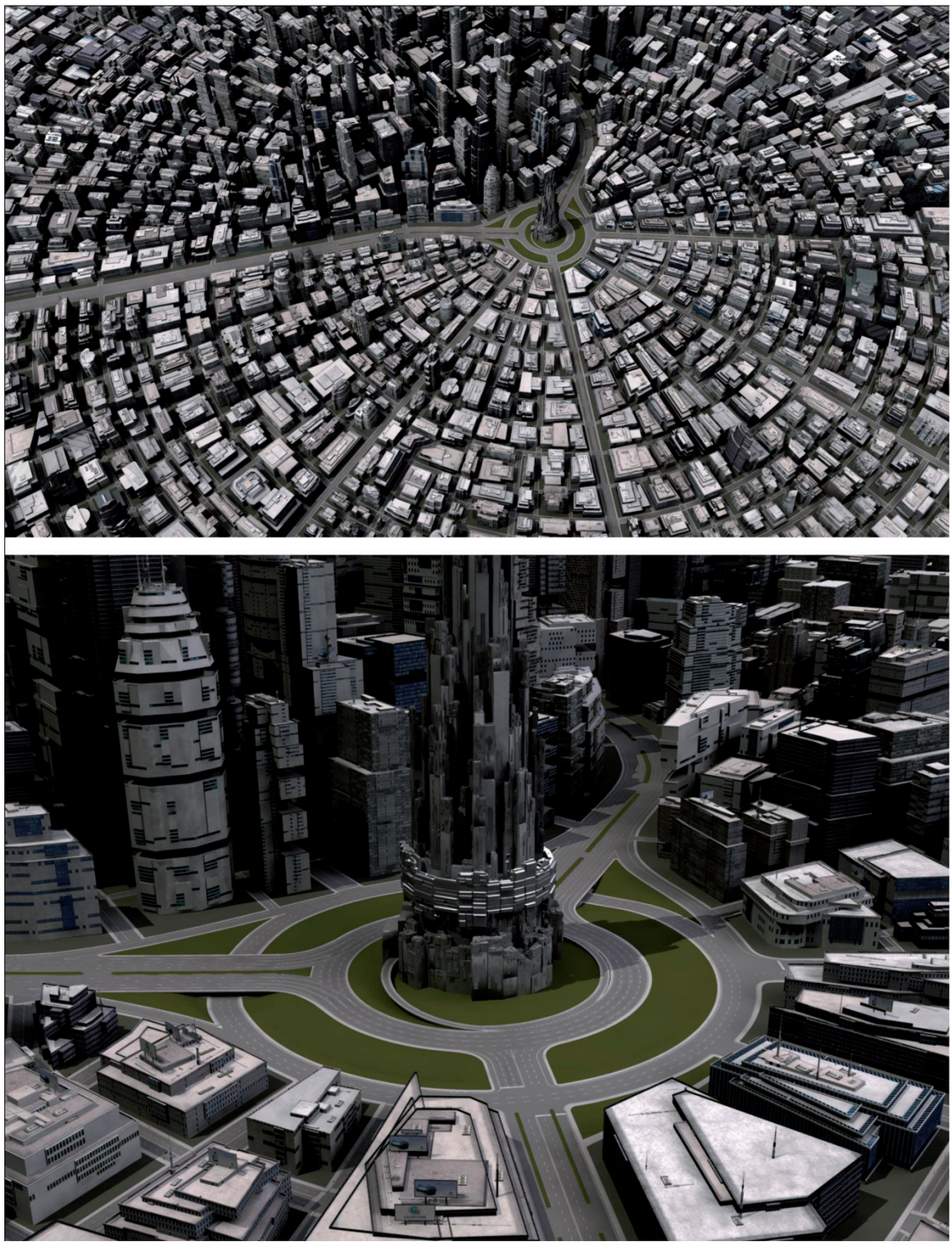


Abbildung 31: Mental Ray Testrenderings der City aus Softimage

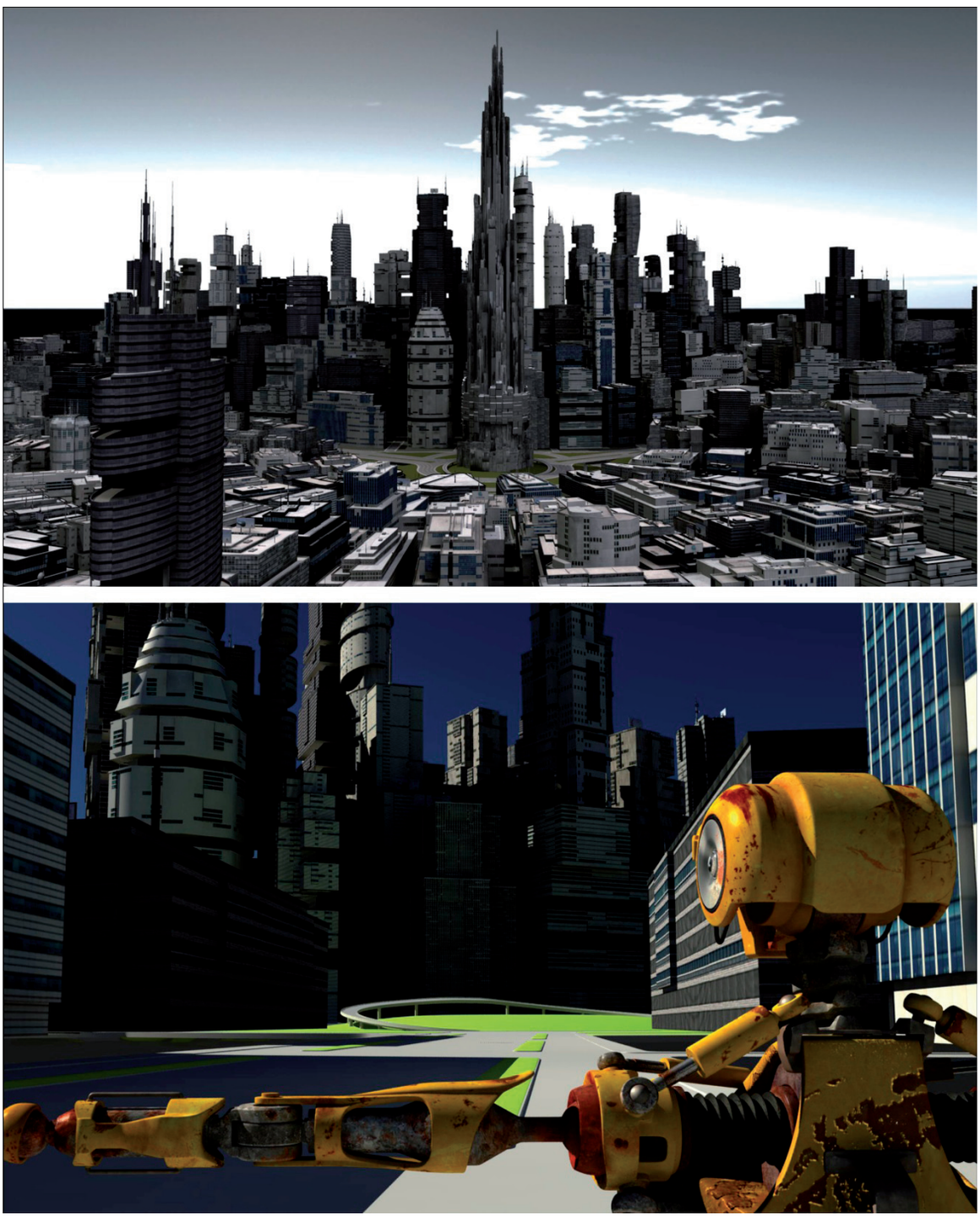


Abbildung 32: Mental Ray Testrenderings der City aus Softimage

6 Integration urbaner Ökosysteme mit Vue

Um die Grünflächen der bereits als fertiges 3D-Modell vorliegenden Stadt mit realistischer Vegetation auszustatten fiel die Entscheidung letztlich auf die Verwendung des Softwarepakets VUE v.8 xStream von E-on Software, mit dem es möglich ist, fotorealistische Landschaften, vielschichtige Ökosysteme sowie Atmosphären zu generieren und zu rendern.

Die folgenden Abschnitte verdeutlichen den angewandten Workflow und schildern den Integrationsprozess von urbanen Ökosystemen von der Planung bishin zum Rendering und zeigen erste Resultate in Form erster Testrenderings.

6.1 Workflow

Bei der Integration von VUE-Objekten in das Projekt wurden zwei unterschiedliche Workflows getestet. Die erste getestete Lösungsmethode bediente sich des mit VUE mitgelieferten Plug-Ins VUE xStream. Mit Hilfe von VUE xStream können VUE-Szenen direkt innerhalb von Autodesk Softimage erstellt, bearbeitet und gerendert werden. Somit können diejenigen Objekte, auf denen Vegetation benötigt wird, als Grundlage für die Bepflanzung definiert werden und mit Hilfe verschiedener Tools eine virtuelle Bepflanzung vorgenommen werden. Ökosysteme können beispielsweise flächendeckend auf die gesamte Geometrie angewendet werden oder punktuell mittels Pinselwerkzeug aufgebracht werden. Ein Ökosystem kann dabei aus beliebig vielen unterschiedlichen Vegetationsebenen bestehen, die im sogenannten Eco System Editor bearbeitet und kontrolliert werden.

Das Rendering erfolgte bei diesem Workflow direkt in Autodesk Softimage mit Mental Ray unter Zuhilfenahme des VUE Renderers für die Berechnung von Vegetation und Atmosphäre. Die auf diese Weise erzielten Ergebnisse konnten allerdings nicht mit der Qualität der in VUE intern berechneten Bilder mithalten.

Aufgrund der wenig zufriedenstellenden Ergebnisse beim Rendering der VUE-Objekte innerhalb von Softimage wurde ein anderer Workflow entwickelt, der auf dem Export der zu begrünenden Geometrie aus Softimage und anschließendem Import in VUE Standalone beruht.

Die Implementierung der Ökosysteme folgt hierbei dem selben Ablauf wie in der bereits oben beschriebenen Methode, für das Rendering innerhalb von VUE Standalone müssen allerdings auch die Softimage Kamera inklusive der animierten Kamerafahrten sowie die schattengebenden Lichtquellen nach VUE transferiert werden, um beim Rendering korrekte Kamerabewegungen beziehungsweise Licht- und Schattenverhältnisse zu erhalten.

Das Rendering erfolgt schließlich in unterschiedlichen Renderpasses, wobei Bildsequenzen mit Alphakanal exportiert werden. Dabei werden Vegetation, Atmosphäre, Z-Tiefeninformation als eigenständige Bildsequenzen gespeichert, um in der Post Production Phase eine möglichst Große Flexibilität bei der Integration des Materials zu ermöglichen. Dieser Workflow gestaltet sich zwar insgesamt etwas aufwändiger, ermöglicht aber eine größtmögliche Qualität bei der Erzeugung der Render Images.

Im Zuge der Finalisierung des Kurzfilmprojektes ist angedacht, die möglichen Ursachen für den Qualitätsverlust beim Rendering der Ökosysteme in Softimage aufzuspüren, um den Vorteil der schnelleren Navigation und Manipulation von Objekten in Softimage ausnutzen zu können.

6.2 Resultate

Dieser Abschnitt zeigt erste Resultate der Begrünung des Stadtmodells anhand von ersten gerenderten Probebildern (siehe Abbildung 33 und Abbildung 34, S. 81).

Abbildung 33: Erste Testrenderings der VUE Vegeation

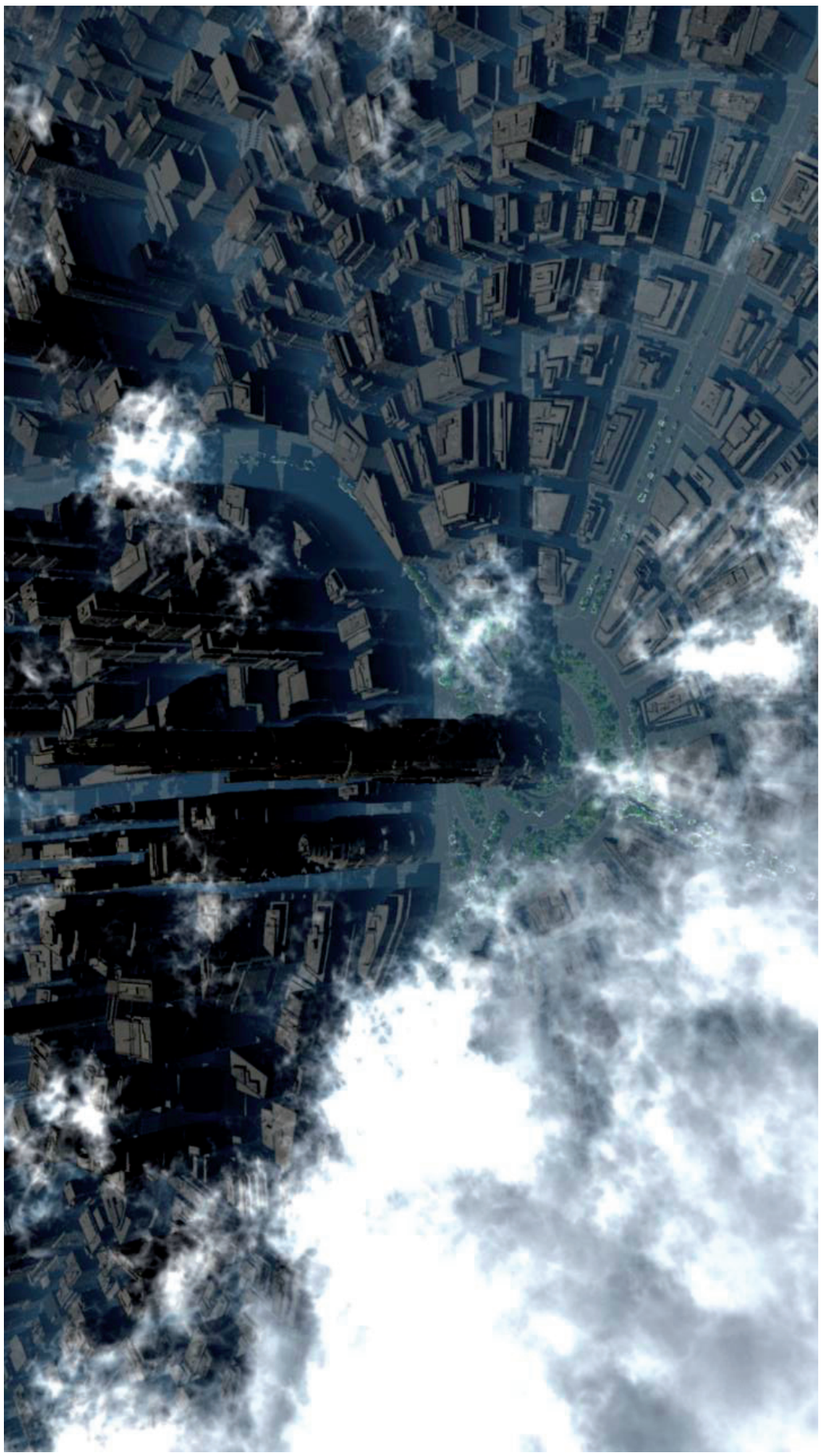


Abbildung 34: Erste Renderstudie des letzten Shots des Kurzfilmes

7 Character Design und Umsetzung

Der zweite wichtige Meilenstein bei der Erarbeitung dieser Fallstudie war das Design und die Erstellung der 3D-Charaktere, die für den resultierenden Kurzfilm als Handlungsträger fungieren.

Beim Design von 3D-Charakteren existieren nach Maestri zwei Zugänge: die zu visualisierende Story steht üblicher Weise entweder bereits vor der Realisierung der Charaktere fest, oder es existieren bereits Charaktere, die bloß nach einer Geschichte verlangen um in Aktion treten zu können. Beide Ansätze haben gemeinsam, dass Story und Characters gemeinsam eine narrative Einheit bilden und sich gegenseitig bei der Vermittlung einer Botschaft unterstützen. (vgl. Maestri, 2002 S. 28)

Im Falle dieses Projektes gestaltete sich der Zugang etwas komplizierter. Am Startpunkt des Projekts waren weder Charaktere noch eine Geschichte vorhanden, vielmehr existierte der Anspruch, einen Kurzfilm zu schaffen, der drei wesentliche Spezialgebiete aus der Welt des Animationsfilms vereint. Sowohl die filmische Geschichte des Kurzfilms, als auch die Charaktere sollten in ein Setup integriert werden, das die Themenfelder Crowd Simulation, Motion Capturing sowie das Design von prozeduralen Film Sets abdeckt. Die Entwicklung von Story (siehe Kapitel 2.1 Story, S. 15) und Charakteren erfolgte in diesem Fall zeitgleich und unter Einbeziehung der genannten konzeptionellen Anforderungen.

Die folgenden Abschnitte schildern den Design- und Produktionsprozess bei der Erstellung dieser 3D-Character Modelle. Weiters werden die Anforderungen thematisiert, die sich an das Modeling und Texturing stellen, wenn die Verwendbarkeit der Modelle als Crowd Agents in einer Massensimulation gewährleistet sein muss.

7.1 Anforderungen für die CG Characters

Bei der Entwicklung eines 3D-Characters ist es von besonderer Wichtigkeit, die Techniken im Auge zu behalten, durch die der Character zum Leben erweckt wird. Ein für die Realtime Engines von Computerspielen produzierter 3D Character ist an ganz andere technische Richtlinien gebunden, als ein Character der in einem Animationsfilm in Szene gesetzt werden soll. (vgl. Kerlov, 2009 S.58)

Die in dieser Arbeit thematisierte Fallstudie stellt an das Character Design ganz besondere Ansprüche. Die Modelle müssen einerseits einen hohen Detailgrad aufweisen, um für die Anwendung in nahen Kameraeinstellungen geeignet zu sein, andererseits müssen die Characters die Anforderungen erfüllen, die durch die Verwendung der Models als Crowd Agents innerhalb einer Massensimulation gestellt werden. Im speziellen ist es hierbei wichtig, die Anzahl der Polygone pro Character und den Speicherbedarf für dessen Texturen möglichst niedrig zu halten, um einerseits zu gewährleisten, dass die für die Massenszenen benötigte Menge an Characters die Rechenkapazität des eingesetzten Computersystems nicht überschreitet, und um andererseits die Geschwindigkeit von Rendering und Simulationsberechnungen zu steigern.

7.2 Character Design

Die Entwicklung des visuellen Looks der Character Modelle ist einer der wichtigsten Prozesse für die Vermittlung der filmischen Geschichte und ihrer generellen Atmosphäre. Konzept-Skizzen - sogenannte Concept Arts - helfen dabei, entscheidende Eigenschaften zu definieren, Ideen zu visualisieren und einen Überblick über die Ausmaße der geplanten Arbeiten zu vermitteln. Des Weiteren können zweidimensionale Concept Arts für die Erstellung der dreidimensionalen Charaktere zu Hilfe genommen werden. Auch im Falle dieser Arbeit bildete die Erstellung von Concept Arts den ersten Schritt in Richtung des gewünschten Looks. Um die Story dieses Kurzfilmprojektes optimal transportieren zu können, wurden von Anfang an mehrere Protagonisten eingeplant, aus produktionstechnischen und zeitlichen Gründen wurden zum Zeitpunkt dieser Arbeit vorerst drei der Charaktere vollständig umgesetzt. Die Umsetzung der restlichen Characters ist im Rahmen der Ausproduktion des Projekts geplant. Die Abbildung 35 und Abbildung 36 (S. 86) zeigen die ersten Concept Arts der Protagonisten und des Antagonisten. (vgl. Kerlow, 2009 S. 59)

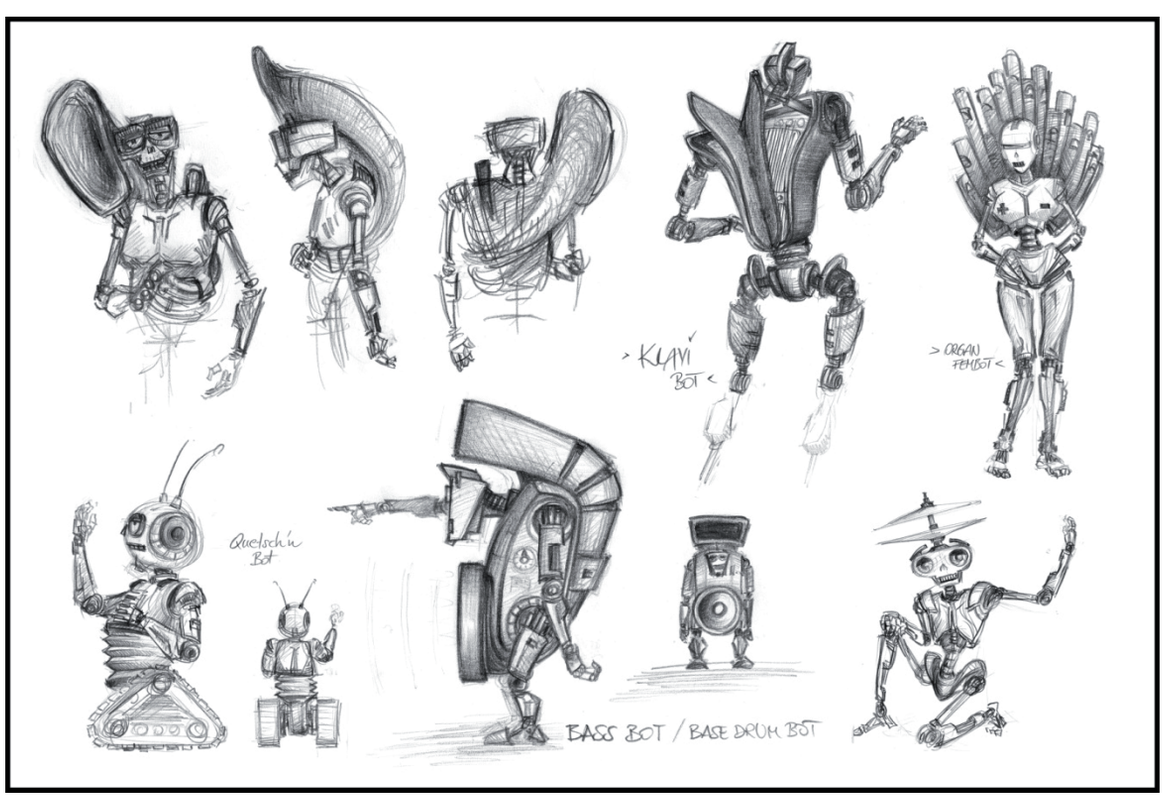


Abbildung 35: Abbildung X: Erste Concept Arts der Protagonisten

Für die Vermittlung der Handlung war es – nicht zuletzt durch das Komplette Fehlen von Dialogen - von besonderer Wichtigkeit, die charakterlichen Eigenschaften von Protagonisten und Antagonisten über deren äußerliche Erscheinung und die Art ihrer Körpersprache und Bewegung auszudrücken.

7.2.1 Protagonisten

Die Gruppe der Protagonisten zeichnet sich laut Storyboard generell durch eine größere Varianz der Individuen aus; dieser Effekt wird sich in Zukunft durch das Hinzufügen weiterer Charaktere noch wesentlich verstärken. In Bezug auf ihre äußere Form haben die Protagonisten einige gemeinsame Eigenschaften: Alle Individuen sind entweder eine Hybridform zwischen Roboter und Instrument beziehungsweise ein Roboter in Form eines Klangerzeugers. Weiters zeichnen sich alle Individuen durch eine stark abgenutzte, verwitterte äußere Hülle aus, die erahnen lässt, dass sie am unteren Ende der Gesellschaft und ohne Wohlstand existieren müssen.

Um den sonst recht technisch anmutenden Protagonisten eine Mimik zu verleihen, wurden ihre Augen so gestaltet, dass sie mit dem Augeninnenring Zoomen können, hinter Glaslinsen sitzen größenveränderliche Irisblenden zur Simulation echter Augenlinsen und herunterfahrbahre Deckklappen der Augen simulieren Augenlider. Da die Innenringe auch frei drehbar sind, lassen sich durch die Animation der Augenpartie Gefühle ausdrücken und den Protagonisten eine gewisse Liebenswürdigkeit einhauchen.

Auch im Sinne der Körpersprache weist die Protagonistengruppe eine größere Vielfalt auf, als die Gruppe der Antagonisten. Die Bewegungsmuster sind von behäbig und schwerfällig bishin zu leichtfüßig und hektisch gestreut. Während beispielsweise der Bass Bot Character sich wie ein tonnenschwerer Koloss bewegt, verfügt der Character des Treble Bot über eine leichte, weiblichere Körpersprache. Die Vielfalt der Bewegungen soll einerseits die Masse heterogener wirken lassen, andererseits drücken die stark unterschiedlichen Bewegungsmuster Individualität aus, was Protagonisten von Antagonisten grundlegend unterscheidet.

7.2.2 Antagonisten

Die Antagonisten treten als homogene Gruppe auf in der sich alle Akteure wie ein Ei dem anderen gleichen. Ihre äußere Hülle ist fast zur Gänze aus poliertem Autolack gefertigt, auf der Vorderseite formt sich aus dem Chassis ein angedeutetes Revers eines Sakkos mit darunterliegendem Hemd und Kravatte. Diese extreme Homogenität vermittelt die Assoziation von Identitätslosigkeit und erinnert stark an das Outfit eines typischen Bankers oder Managers. Anstatt einer Augenpartie besitzt der Antagonisten-Character ein flaches Visir, das gleichzeitig als Display fungiert und durch seine Formgebung einen bedrohlichen Eindruck vermittelt.

Als weiteres Charakteristikum – und als Kontrast zu den mit Klangerzeugern hybridisierten Protagonisten – besitzt der Banker Bot keine funktionierenden Beine, sondern ist gewisser Maßen mit einem Segway verschmolzen. Seine geradlinigen, unbeweglichen Beine enden in einer in Z-Richtung verbiegbaren Gummimuffe und einem darunterliegenden Wippgelenk in X-Richtung des Characters, welches den Körper des Roboters mit dem darunterliegenden Segway-Element verbindet. Durch die Implementierung dieses fahrbaren Untersatzes wurde eine weitere Charakterisierung des Antagonisten bewirkt. Da sich die Banker Bots aufgrund ihrer Anatomie souverän rollend und auf präzisen Bahnen bewegen, rufen sie weitere Assoziationen hervor, die sie als unnatürliche diabolische Maschinen ohne eigenen freien Willen darstellen und sie für den Zuseher bedrohlich erscheinen lassen.

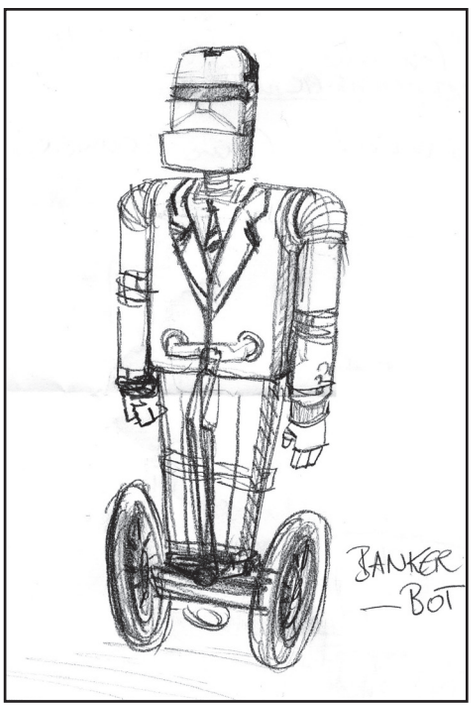


Abbildung 36: Erste Concept Arts des Antagonisten

7.3 Modeling

Um die in den Concept Artworks grob skizzierten Charaktere zu modellieren wurde Autodesk Softimage eingesetzt. Die Modelle wurden dabei mit Hinblick auf die Verwendbarkeit in Massive Prime ausschließlich als Polygonmodelle ausgeformt. Bei den Modellen des Bass Bot und des Banker Bot dienten die in der Designphase erstellten Skizzen als Modelingvorlage. Der Treble Bot Character entstand dagegen - basierend auf den bereits durch den Bass Bot definierten Designgrundlagen - ohne Verwendung einer Vorlage. Der Modeling Workflow setzte sich aus mehreren, aufeinander folgenden Arbeitsschritten zusammen, um sämtliche Anforderungen abzudecken. Die Modelle wurden aufgrund der guten Transferierbarkeit von Motion-Daten und der besseren Handhabung der Skeletons in der Riggingphase allesamt in T-Pose modelliert (siehe Kapitel 8.2.2 Kinematics, S. 108).

In der ersten Phase wurden die Charaktere als High Detail Modelle mit Subdivision Surfaces für die Anwendung in nahen Einstellungen modelliert. Danach sah der Workflow vor, mit den Modellen mittels Crowd Simulation Performancetests innerhalb von Massive Prime durchzuführen, um die Belastung des Simulationssystems zu eruieren.

Im nächsten Schritt wurden die Modelle für die Anwendung in der Massensimulation einem gründlichen Cleaning Prozess unterzogen, wobei die Geometrie von allen Edges und Faces bereinigt wurden, die für eine Darstellung der Modelle aus größerer Entfernung und innerhalb einer Masse von Crowd Agents keinen visuellen Nutzen trugen. Durch die Entscheidung, bei der Darstellung der Massen auf die Verwendung von Subdivision Surfaces zu verzichten, konnten auch die entsprechenden - für die Darstellung von Kanten bei Subdivision Surfaces benötigten - Edges entfernt werden. Nach Abschluss dieses Vorgangs wurden die Charactere als Softimage Models im Format .EMDL exportiert. Die Abbildung 37 (S. 88) zeigt Screenshots vom Modelingprozess der drei Characters.

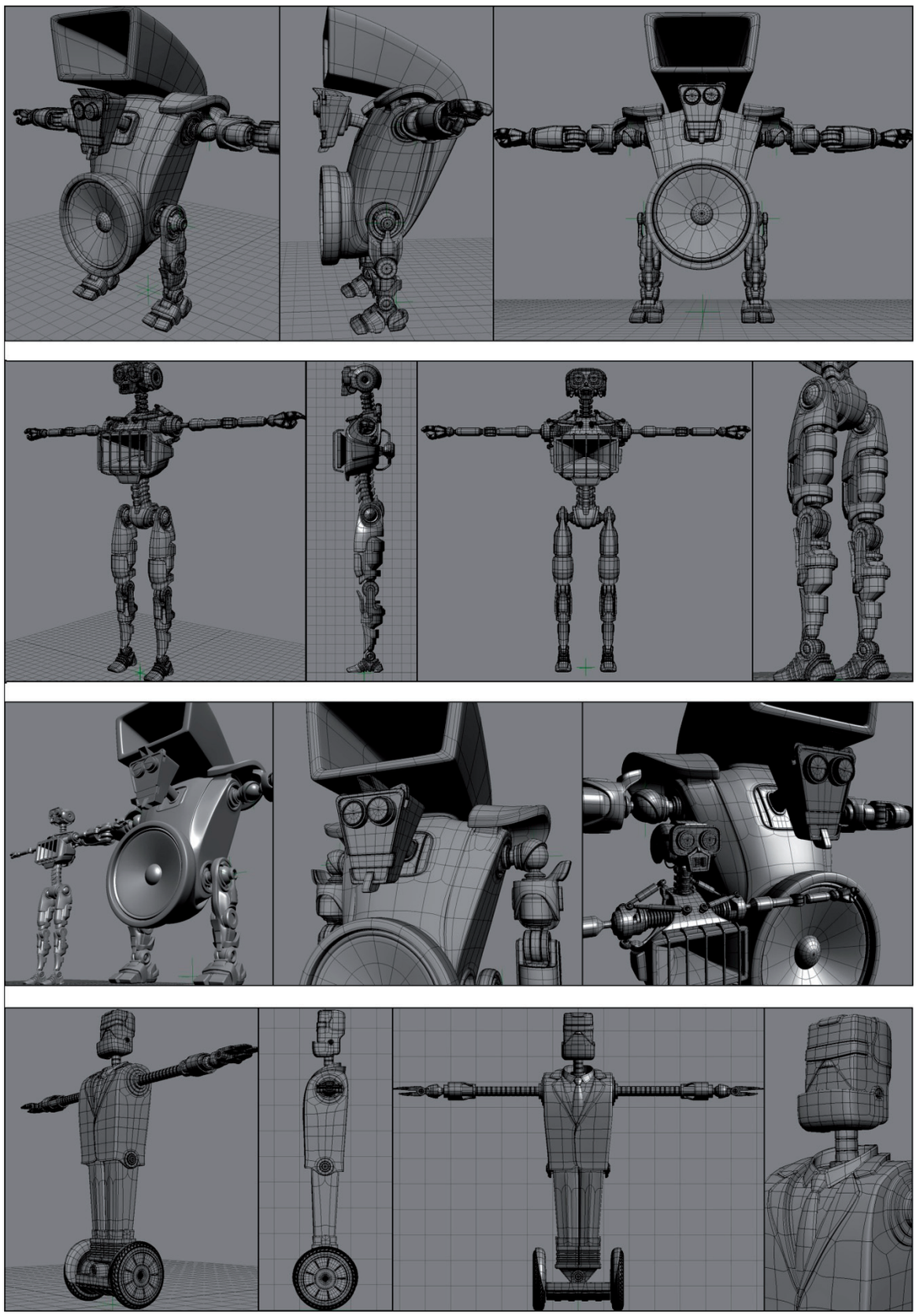


Abbildung 37: Die drei Charaktere in der Modelingphase

Um die Massenmodelle für die Anwendung in Massive Prime vorzubereiten wurde das jeweilige Low Quality Model erst im Format .OBJ oder via Crosswalk im Transferformat .XSI exportiert und in Autodesk Maya importiert. In Maya wurden anschließend sämtliche bewegliche Körperteile des Modells zu separaten Polygonobjekten konvertiert und als einzelne Objekte im Format .OBJ exportiert, um sie anschließend in Massive Prime importieren zu können. Hierbei war es von besonderer Wichtigkeit, keine *Normals* in den .OBJ Dateien mitzuspeichern, da es in Massive Prime sonst zu Problemen kam. In Massive Prime wurden schließlich die Einzelteile dieses sogenannten Segmented Character an die entsprechenden Skelettstrukturen des Massive Rig gebunden. Auf diese Thematik wird in den Kapiteln 9 und 10 detaillierter eingegangen.

Weiters wurden bereits in der Modelingphase erste Tests durchgeführt um mögliche Lösungswegen für das Rigging komplexer Modellstrukturen zu prüfen. Abbildung 38 zeigt beispielsweise den ersten Testdurchgang bei der Entwicklung der komplexen Schulteraufhängung des Treble Bot, auf dem in weiterer Folge die Umsetzung des finalen Schulter-Rig in Maya basiert.

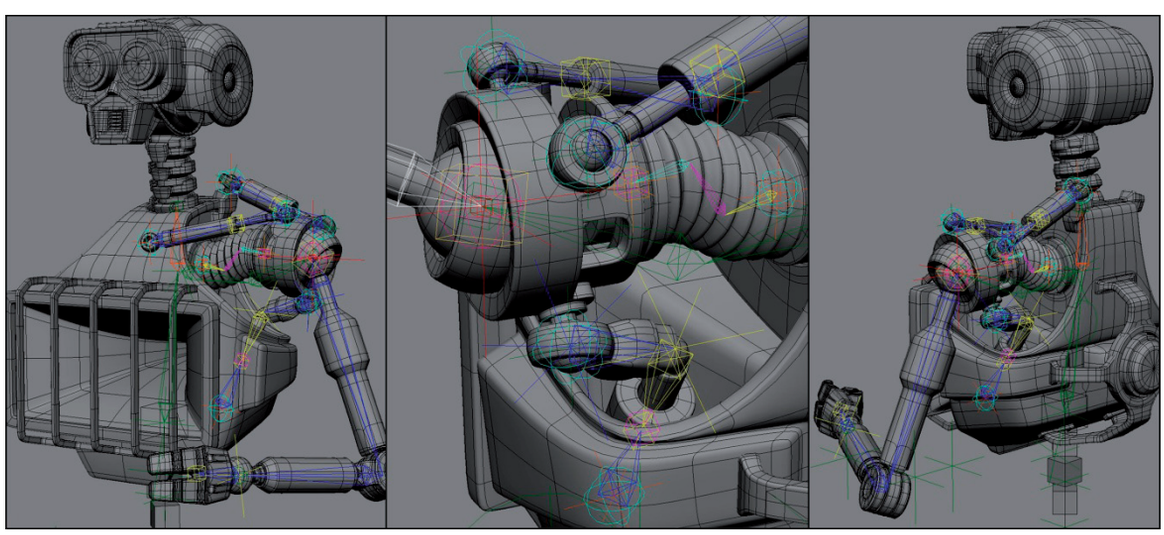


Abbildung 38: Prototyp des Treble Bot Schulter-Rig in Softimage

7.4 Texturing

Nach Abschluss der Modellierung der Characters konnten den Modellen in der Texturierungsphase Materialien und Texturen verliehen werden. Der Anspruch an das Design der Materialien und der Texturen war es hierbei ein stilisiertes Erscheinungsbild zu erreichen, das sich dadurch von Realismus und Hyperrealismus unterscheidet, dass die Oberflächen und Materialeigenschaften überzeichnet dargestellt werden und so einen leicht cartoonhaften Look bekommen. (vgl. Demers, 2002 S. 171)

Der erste Schritt der Texturierung war die Erzeugung der UV-Layouts. Dazu wurden allen Polygonobjekten der Modelle Texturprojektionen zugewiesen, wodurch allen Punkten der Polygonoberflächen entsprechende UVs zugeordnet wurden. Die meisten Objekte wurden hierbei mit Hilfe der seit Softimage 7.5 integrierten Unfold Texture Projection aufgefalten.

Den Körperteilen der Roboter Characters, die gleiche Materialeigenschaften aufweisen sollten, wurde in Folge jeweils ein einziges Softimage Material zugewiesen und anschließend sämtliche UVs der Geomtrie dieser Objekte im Texture Editor zu jeweils einem großen und möglichst dichten UV-Layout arrangiert. Dadurch entstanden beispielsweise ein UV-Layout für alle Objekte die eine verrostete Lacktextur aufweisen und ein separates UV-Layout für Objekte mit unlackierter Oberfläche. Um möglichst kleine Texturen verwenden zu können und die geladene Textur optimal auszunutzen war es wichtig, die UVs so eng wie möglich auf der Texturfläche auszurichten. Anschließend wurden die UV-Meshes als *gestamptes* Grafik-Bitmap exportiert und anschließend in Adobe Photoshop als Orientierungshilfe für die Erstellung der finalen Texturen benutzt, die fertigen Texturen wurden letztlich in Softimage den entsprechenden Materialien zugewiesen. In Abbildung 39 (S. 91) sind die UV-Layouts für die Materialien „lackiertes Metall“ und „rohes Metall“ des Treble Bot Characters abgebildet.

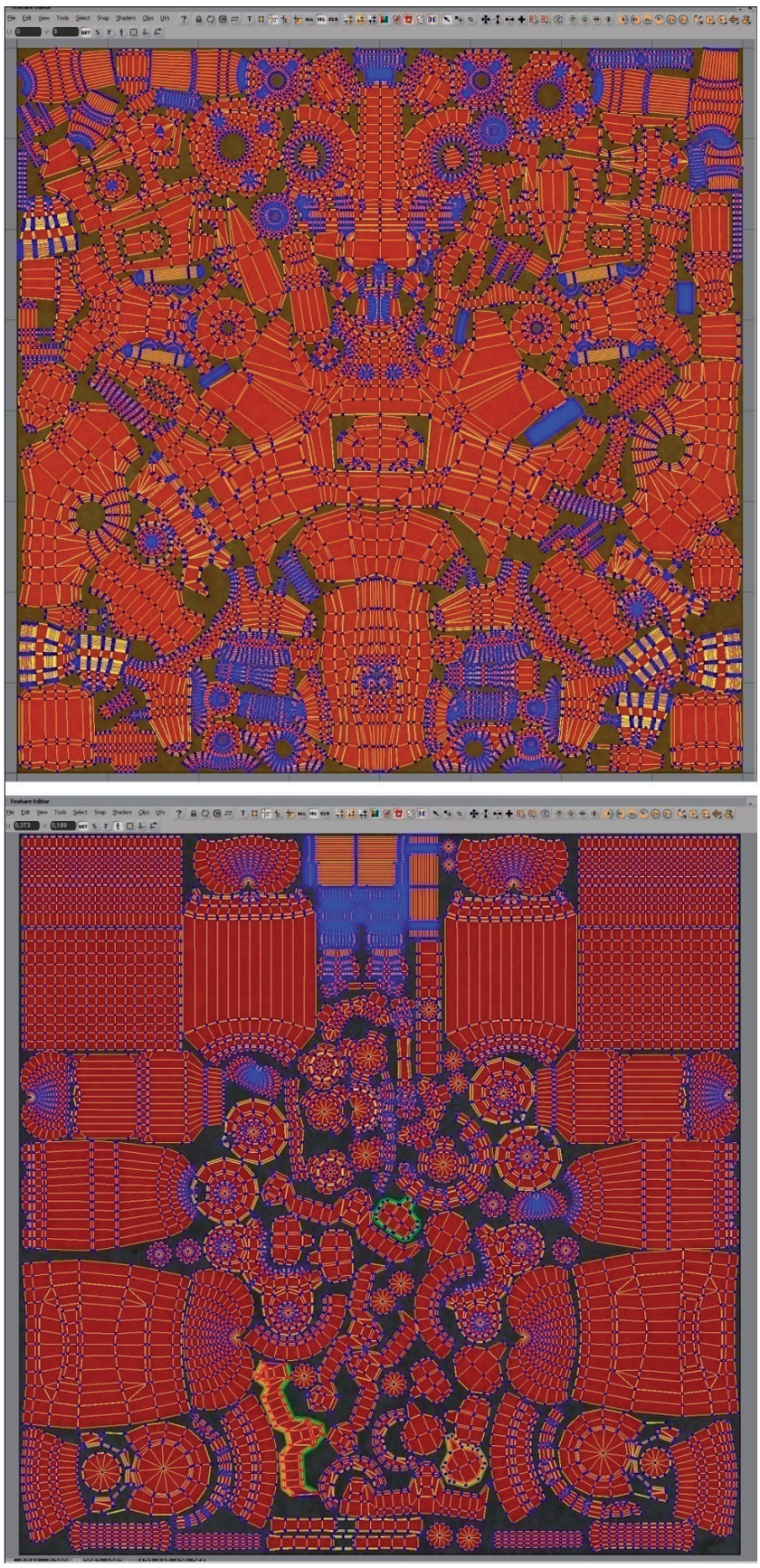


Abbildung 39: UV-Layouts von lackiertem Metall und rohem Metall des Treble Bot

Um die Texturen für den Transfer in Massive vorzubereiten mussten zusätzlich noch einige Anpassungen vorgenommen werden. Einige Materialien der Softimage Modelle waren auf *Polygon-Cluster* angewendet, die jedoch von Massive nicht erkannt werden. Beispielsweise waren Gummimuffen an Händen und Füßen von Treble Bot und Bass Bot als *Cluster* definiert, die über dem eigentlichen Material „lackiertes Metall“ sitzen – das Material „Gummimuffe“ bediente sich also der UV-Sets des zugrunde liegenden Polygonobjektes, hatte jedoch ein eigenes cluster-basiertes gummiartiges Material zugewiesen. Da Massive jedoch nicht mit Clustern arbeitet, wurde das Problem so gelöst, in dem nachträglich die Gummimuffen auf einer speziellen Massive-Variante der Lacktextur in Photoshop aufgemalt wurde. Die Gummimuffen erhielten so zwar die Materialeigenschaften von lackiertem Metall, doch dies konnte aufgrund der allgemeinen Komplexität der Masse und der großen Entfernung zu den einzelnen Modellen vernachlässigt werden.

Durch die Zusammenfassung sämtlicher Körperteile mit gleichen Materialien zu großen UV-Layouts, konnten die entsprechenden Texturen leicht zu gering aufgelösten Texturvarianten für die Anwendung in der Massensimulation heruntergerechnet werden. Dadurch wurde gewährleistet, dass Massive Prime ein Pool an Texturen zur Verfügung steht, um so die unterschiedlich großen Versionen der Texturen in Anpassung an die Entfernung des jeweiligen Models zur Kamera dynamisch auswählen zu können. Desweiteren ist im Zuge der Weiterführung des Projekts geplant, diesen Texturpool mit weiteren von Massive dynamisch zuweisbaren Texturen zu bestücken, die sich in Auswahl und Anordnung von Details, sowie in ihrer generellen Farbgebung unterscheiden, um so einen größeren Grad an Heterogenität in der Masse erzeugen zu können.

7.5 Resumee und Resultate

Auf dem Weg von der ersten Skizze, bis zum fertigen modellierten und texturierten Character konnten - auf das einzelne Modell bezogen - alle Anforderungen an das Charakter Design erfüllt werden. Allerdings stellte der zeitliche Aufwand bei der Produktion der Modelle einen limitierenden Faktor dar. Um eine Vielzahl unterschiedlicher Charaktere zu entwickeln, die für die Anwendung in einer Massensimulation wiederum mit individuellen, unterschiedlichen optischen Charakteristiken wie variierenden Körperformen oder Texturen ausgestattet sind, muss ein enormer Aufwand betrieben werden. Die Anforderung, die Aufgabe diese Heterogenisierung der Masse als einzelner Designer innerhalb des Produktionszeitraumes dieser Arbeit im Alleingang durchzuführen, konnte also nicht in der veranschlagten Zeit erfüllt werden; allerdings konnten im Zuge dieser Arbeit sämtliche Arbeitschritte und Funktionen des zu Grunde liegenden Workflows bestimmt werden, um nach Beendigung dieser Fallstudie die gewünschten visuellen Erweiterungen mit Hilfe weiterer Manpower in kürzerer Zeit problemlos umzusetzen.

Bezogen auf die Modellierung der Charaktere für die Anwendung als Massive Crowd Agents hätte der Ansatz, die Modelle vorerst mit niedrigem Detailgrad und ohne Subdivision Surface Modeling für den Einsatz in Massive zu fertigen und darauf zu High Detail Models mit Subdivision Surfaces auszubauen, einen Zeitvorteil bewirkt, da sich das nachträgliche Bereinigen der Modelle als sehr zeitaufwändig herausstellte.

Abgesehen von zeitlichen Problemen konnte die Working Pipeline für die Erstellung von 3D-Characters, die sowohl massentauglich als auch für nahe Einstellungen geeignet waren, erfolgreich eruiert werden. Während der Fertigstellung eines jeden Characters wurden zahlreiche Testbilder berechnet, um die Ergebnisse von Modellierung und Texturierung zu überprüfen; die Abbildungen auf den folgenden Seiten zeigen diese ersten für Texturtests gerenderten Bilder der Characters.

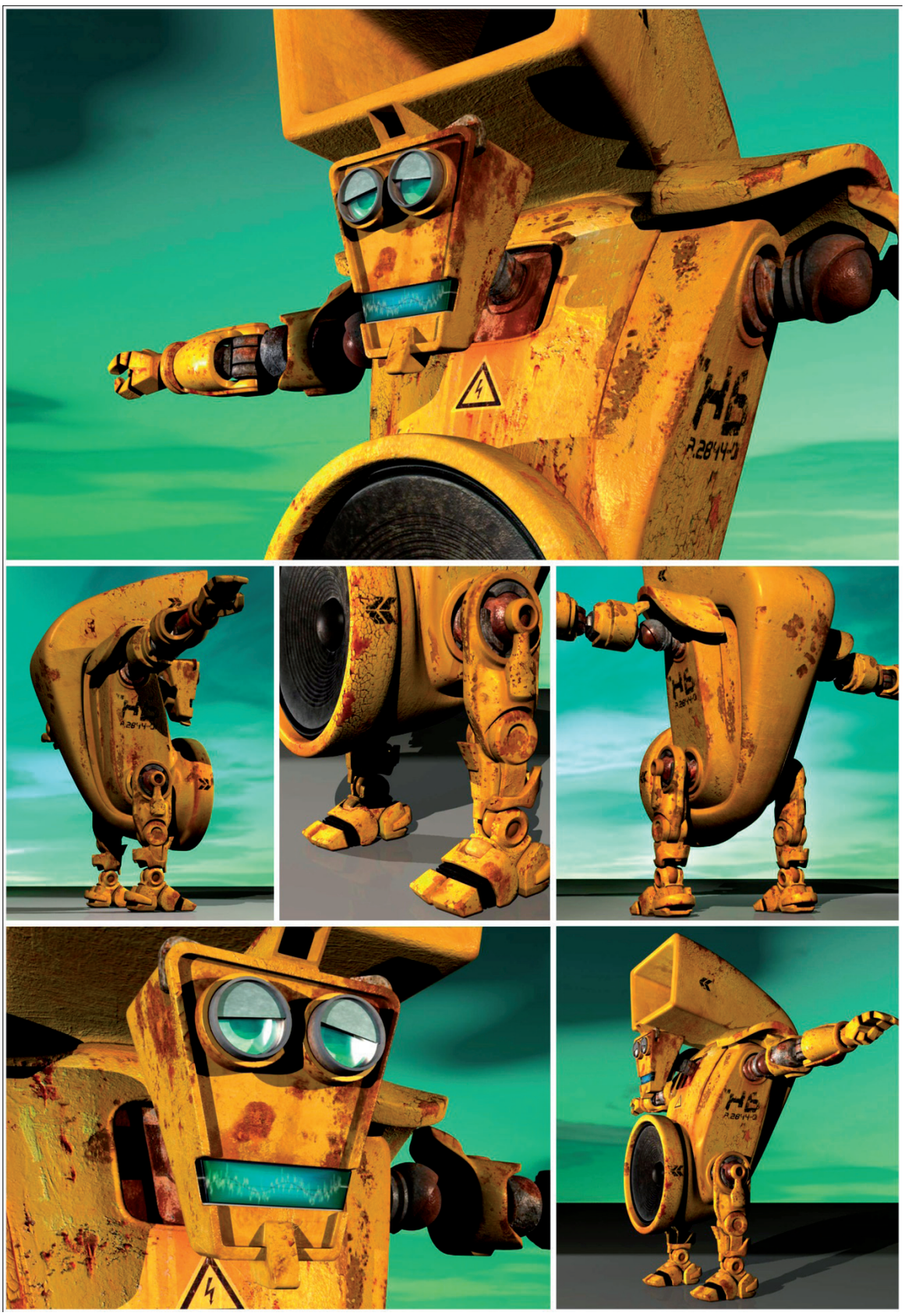


Abbildung 40: Erste Renderings des Bass Bot Character

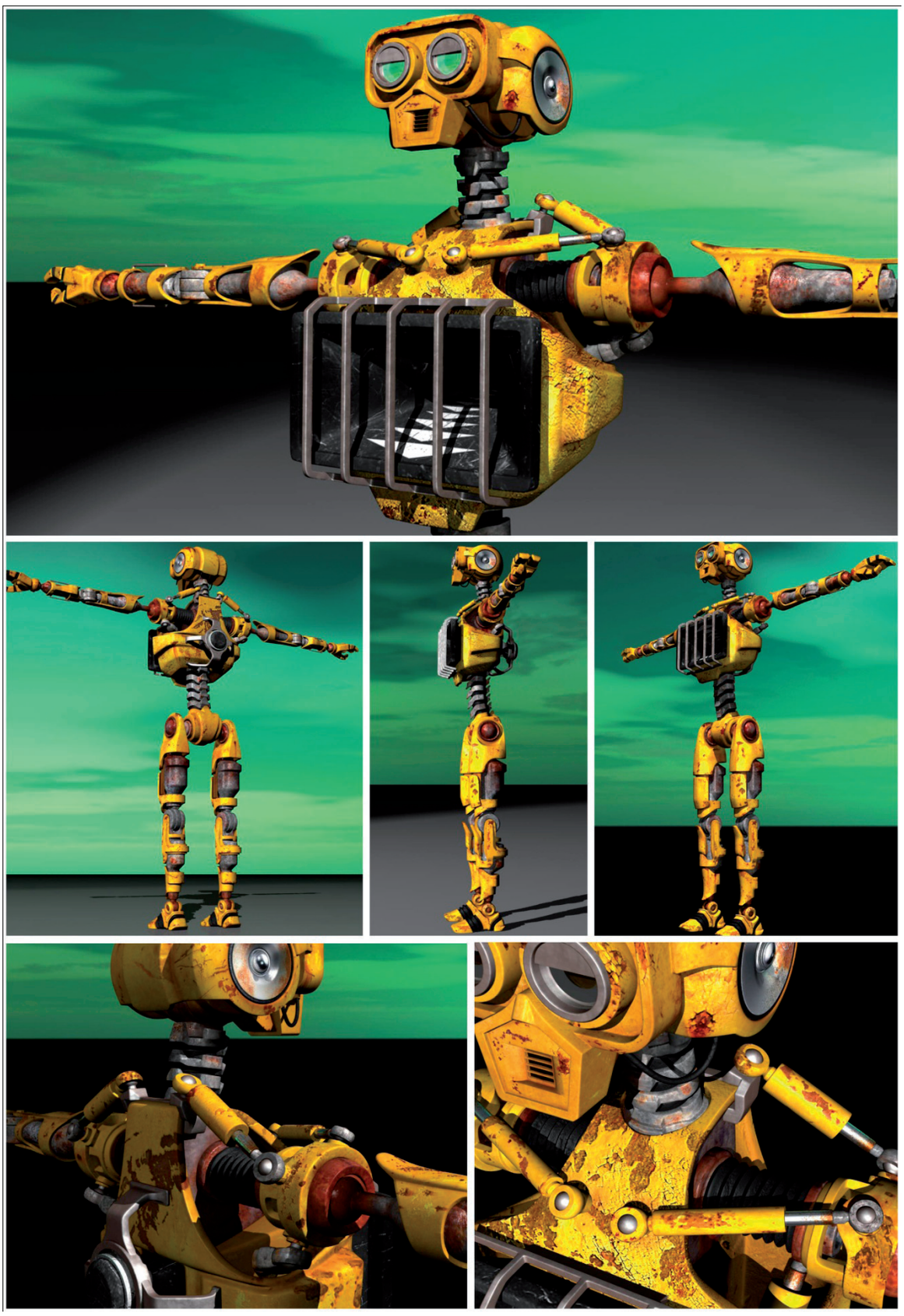


Abbildung 41: Erste Renderings des Treble Bot Character

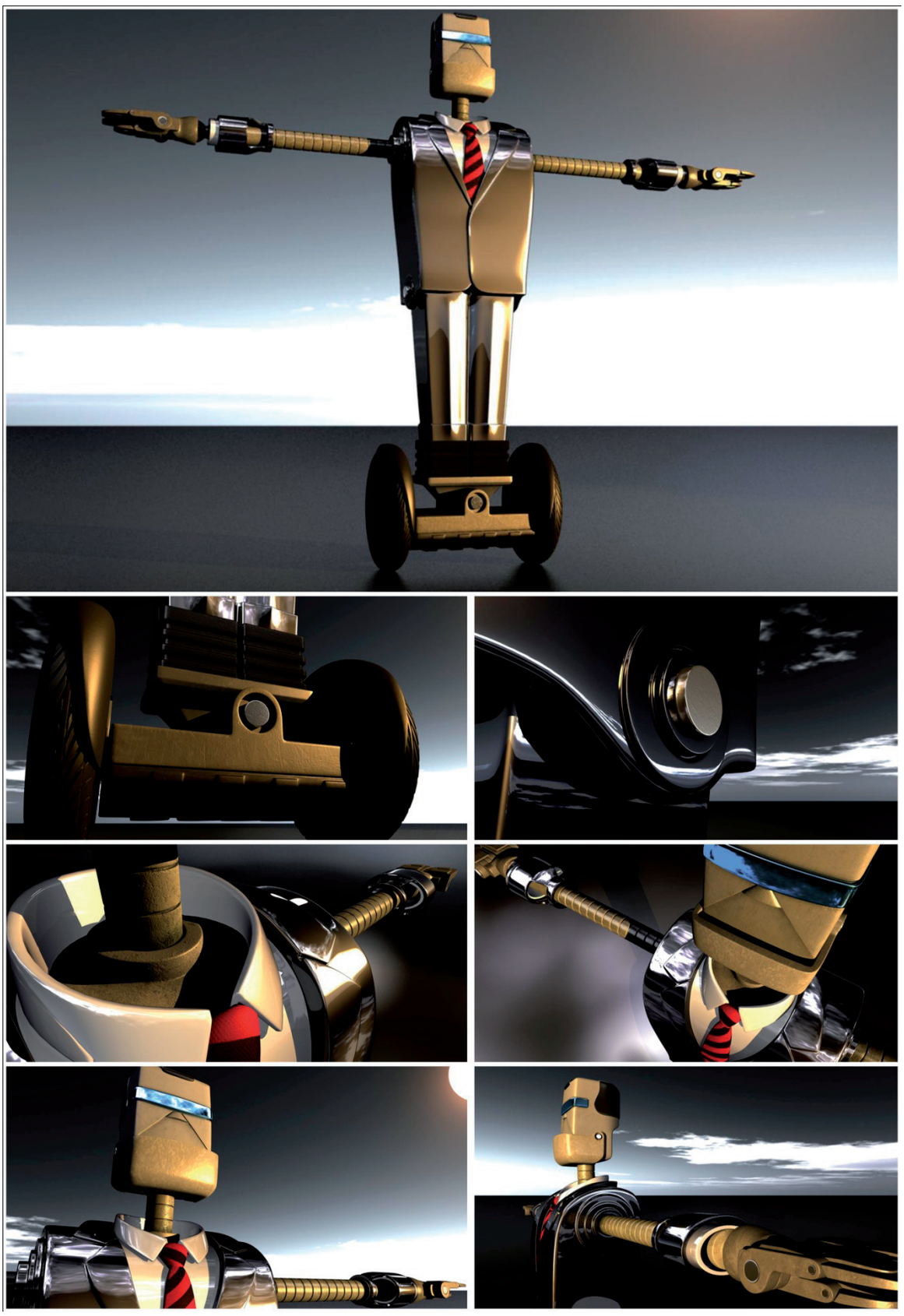


Abbildung 42: Erste Renderings des Banker Bot Character

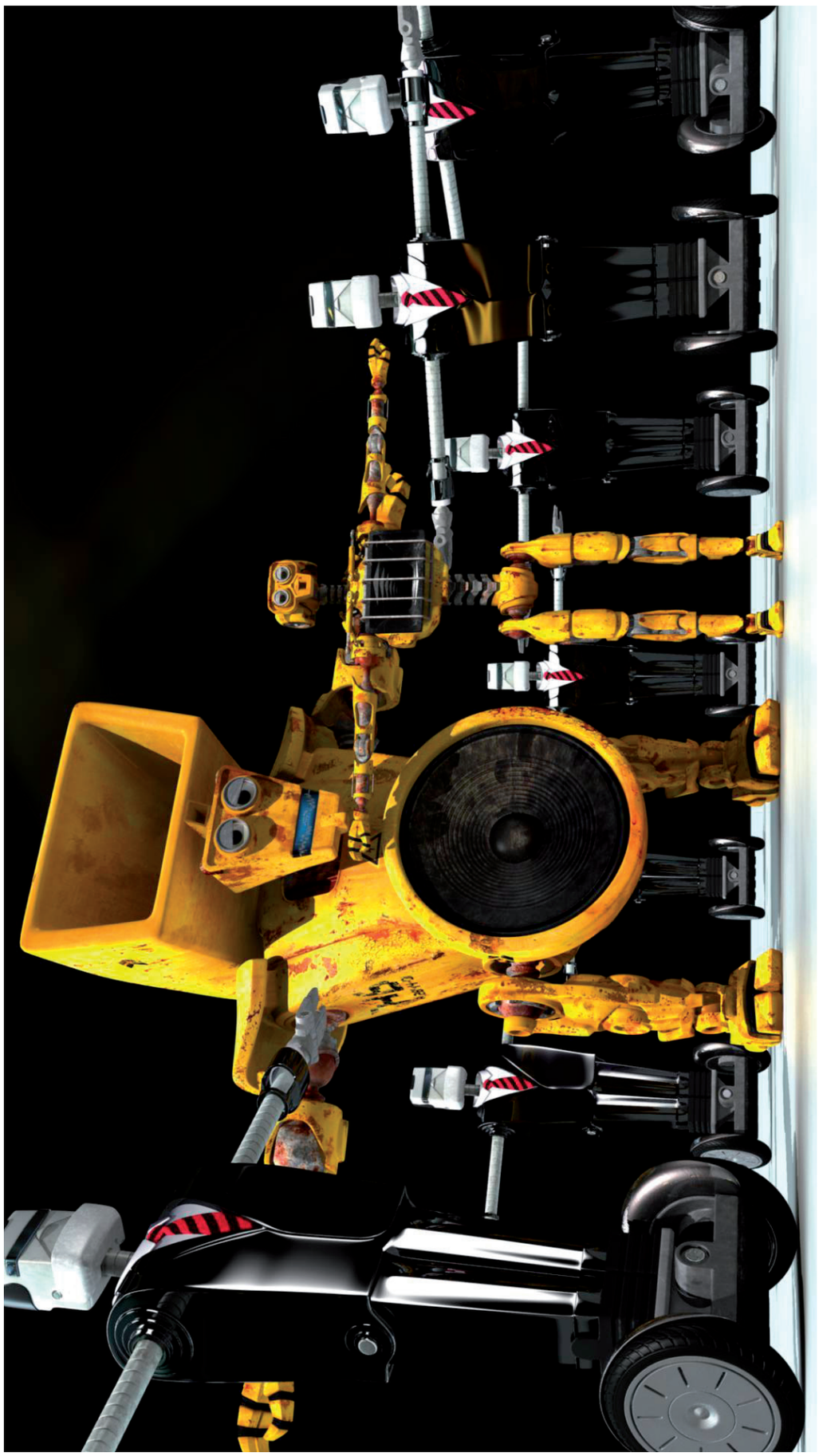


Abbildung 43: Test-Arrangement der drei Characters in T-Pose

8 Character Setup

Ein computergenerierter 3D Character zeichnet sich in seinen Wesenszügen nicht nur durch dessen Formgebung und dem daraus resultierenden äußeren Erscheinungsbild aus, sondern in gleichem Maße durch die Art der Bewegung sowie die Körperhaltung. Erst ein harmonischer Einklang zwischen Bewegung und Physiognomie ermöglicht die Illusion der haptischen Präsenz, die virtuellen Figuren ihre Magie verleiht.

„Charakteranimation hat zunächst nichts mit erlesenen 3D-Körpern und Köpfen zu tun, auch nichts mit feinporigen Shadern für digitale Menschenhaut oder anisotrop Licht reflektierendes Fell für Tiere. Charakteranimation hat mit Charakter zu tun, mit oft nur minimalen Bewegungen, welche Seele, Stimmung, Gefühl ausdrücken und die Handlung weitertreiben.“ (Schönherr 2000, S.157)

8.1 Anforderungen an das Character Setup

Um möglichst realistisch wirkende und auch grazil anmutende Bewegungen erzeugen zu können, ist ein durchdachtes und gut funktionierendes Character Setup zwingend notwendig. Eine genauere Einführung in die essentiellen Grundlagen der Aufbereitung eines virtuellen Characters für Animationszwecke, sowie eine Erläuterung der Grundvoraussetzungen für verschiedene Animationsvarianten werden in dieser Arbeit näher behandelt. Dazu zählt einerseits die non-lineare Keyframe Animation und das Arbeiten mit Animation Layers, andererseits die Erstellung, Verwertung und Adaption von Motion Capture Daten, sowie die Verfolgung eines prozeduralen Animationsansatzes.

Die non-lineare Keyframe Animation beschreibt eine Methode, die es dem Animator ermöglicht, eine oder mehrere Animationen von verschiedenen Quellen zu adaptieren und in weiterer Folge auch zu kombinieren, um auf diese Weise neue Animationen zu kreieren (vgl. Liverman 2004, S.11).

Layered Animation beschreibt einen Animationsansatz, der es erlaubt, auf unterschiedlichen Animationsebenen zu arbeiten, die unabhängig von einander editiert werden können. Durch die Überlagerung der Layer werden die verschiedenen Parameter mit einander verschmolzen, wodurch Animationen erweitert und verfeinert werden können. (vgl. Liverman 2004, S.10)

Prozedurale Animationstechniken beschreiben eine Methode, die auf den Gesetzen der Physik beziehungsweise der Mathematik beruht. Physikalische und mathematische Operationen übernehmen hierbei den Animationsprozess und setzten den Look der Animation fest. (vgl. Liverman 2004, S.11)

Unter Motion Capturing wird die Technologie verstanden, welche den Transfer einer bewegungsspezifischen Realaufnahme in den virtuellen Raum beziehungsweise auf eine Computergenerierte (CG) Figur ermöglicht. (siehe Kapitel 9.2 Motion Capturing, S. 127)

Nach vollendeter Umsetzung des Character Designs erfolgen die nötigen Schritte um virtuellen Figuren Leben einzuhauchen, beziehungsweise einen Bewegungsapparat zu verleihen. Für diesen Zweck wird eine, dem jeweiligen Körper entsprechende, Skelettstruktur angefertigt, die im Animationsprozess über Position und Haltung jedes einzelnen Körperteils entscheidet. Die Skelettstruktur setzt sich, wie beim menschlichen Körper, aus Knochen und Gelenken zusammen. Im Animationsprozess spricht man jedoch von Bones und Joints, die einer bestimmten hierarchischen Struktur entsprechend, miteinander verknüpft werden. Der Vorgang dient dazu, den Aufbau des gesamten Körpers sowie die Relationen einzelner Körperteile zueinander zu definieren. Die für die gewünschte Bewegung nötige Positionierung der Bones erfolgt im Normalfall über die Rotation von Joints.

Ein menschliches Skelett setzt sich aus über 200 unterschiedlichen Knochen zusammen. Jedoch sind nicht alle dieser Knochen für den Aufbau einer virtuellen Skelettstruktur zu beachten. „As an animator, you need to concern yourself with the major systems of the body and how those systems affect the shape and posture of the body as they move.“ (Maestri 1999, S.50)

Um jedoch die Natürlichkeit einer Bewegung zu rekonstruieren, werden eine genaue Beobachtungsgabe benötigt, sowie optimale technische Voraussetzungen, die Körperhaltung eines digitalen Characters zu kontrollieren.

„Animation is more than just moving things around. It involves proper construction of your characters, as well as a proper staging and the ability to pose your characters naturally.“ (Maestri 1999, S.165)

Zu einem fertigen Character Setup gehört demnach nicht nur ein hierarchischer Aufbau von Joint Strukturen, sondern auch die Erstellung sogenannter Control Objects, die dazu dienen, den eigentlichen Animationsprozess bestimmter Körperregionen zu erleichtern. (siehe Kapitel “8.2.2 Kinematics” on page 108)

„In projects with large volume of animations the setting up these character controls is usually the responsibility of a technical director while the animators devote themselves to animating after the basic controls are set up.“ (Kerlow 2004, S.65)

Das Setup der Joint Strukturen eines CG Characters stellt einen wichtigen Schritt im Character Animationsprozess dar (vgl. Vogel et al. 2000, S.323). Dieser beschreibt eine kritische Phase, da sie schwerwiegende Auswirkungen auf die Steueroptionen des jeweiligen Characters beziehungsweise die spätere Bewegungsfreiheit hat. Das Setup muss nicht nur alle Grundvoraussetzungen für eine flexible Keyframe Animation schaffen, sondern auch so aufbereitet werden, dass für den Import und Transfer von Motion Capture Daten bestmögliche Voraussetzungen geboten werden.

Die Erstellung computersimulierter Massenszenen stellt eine besondere Herausforderung an die Character Animation, da alle zu erstellenden Bewegungen nahtlos ineinander übergreifen müssen. Des Weiteren müssen für jeden Character verschiedene Walk-Cycles, Idle Positionen und weitere spezifischen Aktionen erstellt werden, welche die Darstellung einer heterogenen Masse ermöglichen.

Massive Prime setzt für den Import von Skelettstrukturen und entsprechender Animationen das Maya ASCII Fileformat (.MA) voraus. Aus diesem Grund wird das Grundsetup in Autodesk Maya erstellt, was in den folgenden Unterkapiteln näher

erläutert wird. Um jedoch den Animationsprozess zu erleichtern und so flexibel wie möglich zu gestalten, erfolgt die entsprechende Animationsaufbereitung mit Hilfe von Autodesk MotionBuilder beziehungsweise 3ds Max und dessen Animations-Plug-In CAT (Character Animation Toolkit). Auf diese Techniken wird im Kapitel 9 (siehe Kapitel 9 Character Animation, S. 119) näher eingegangen.

8.2 Rigging

Der Prozess der Animationsvorbereitung, sprich der Aufbau des entsprechenden digitalen Knochengerüsts, beziehungsweise die Erstellung aller nötigen Control Objects, wird im Fachjargon als Rigging bezeichnet, das fertige Produkt als Rig.

„A lot of technology is associated with the setup of skeletons for CG characters, and this can make the process somewhat confusing. Some of the confusion comes from the fact that many times the terminology varies depending on what software package you are using.“ (Vogel et al. 2000, S.325)

In dieser Arbeit werden die Terminologien verwendet, die das jeweilige Programm für den entsprechenden Arbeitsprozess vorgibt. Pipelinebedingt erfolgt der Aufbau des Rigs in Maya, weshalb auch die Grundlagen anhand dieses Softwarepaketes erläutert werden. Auf wesentliche Unterschiede zwischen verschiedenen Produkten wird jedoch im weiteren Verlauf der Arbeit hingewiesen.

Wie bereits erwähnt setzt sich ein Rig aus Bones und Joints zusammen. Bones sind untereinander über Joints verbunden und werden als lange pyramidenartige Objekte dargestellt, dessen Spitze immer in Richtung untergeordneter Hierarchieelemente deutet. Joints fungieren als rotierende, kugelförmige Verbindungselemente, beziehungsweise als Start- oder Endpunkte eines Bones. (siehe Abbildung 44, S. 102)

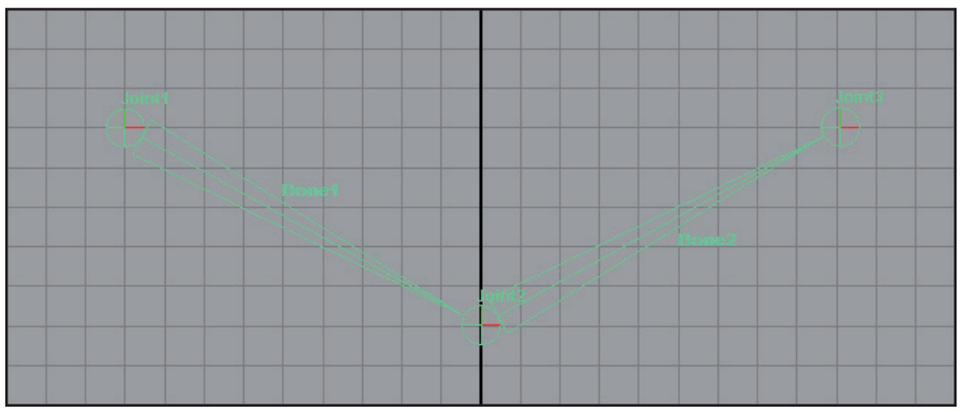


Abbildung 44: Maya - Bone-Joint Struktur

Das Character Setup erfolgt im Idealfall in der so genannten T-Pose. Die T-Pose beschreibt die Haltung, in der sich der Character während der Modeling-Phase befindet. Der Character steht dabei in aufrechter Haltung mit Blickrichtung in die positive Z-Achse. Die Arme sind dabei seitwärts ausgestreckt und die Handflächen zeigen nach unten.

Diese Pose erleichtert nicht nur das Modeling sondern auch den Aufbau und die Justierung des Rigs. Die symmetrische Haltung erlaubt eine Spiegelung bestimmter fertig gestellter Rig-Elemente. Auf diese Weise kann ein erneutes Erstellen von symmetrischen Elementen verhindert werden. Ein Testen aller über die Skelettstruktur erfolgenden Deformationen wird in dieser Haltung ebenfalls erheblich vereinfacht.

Die T-Pose ist auch erforderlich, um verschiedene Animationen auf ein fertiges Rig transferieren zu können. Über die T-Pose erfolgt eine Kalibrierung des Urzustandes für jeden einzelnen Bone. „It is a definition of the character's initial set of coordinate systems." (Menache 2000, S.116)

Die folgenden Abbildungen (siehe Abbildung 45 und Abbildung 46, S. 103) zeigen die fertigen Rigs der zwei Protagonisten, welche als .MA an die Massensimulationssoftware übergeben werden. Beim Treble Bot wurde jedoch für Massive aus Gründen der Performance auf das komplexe Schultersetup verzichtet. Der Aufbau des Rigs erlaubt es, die überflüssigen Bones für diesen Zweck einfach zu entfernen, ohne die Hierarchie negativ zu beeinflussen. Weiters

ist zu erkennen, dass bei diesen Rigs keinerlei Control Objects vorhanden sind. Die Pipeline über Autodesk MotionBuilder ermöglicht es, die Erstellung eben dieser automatisiert zu erzeugen. Auf diese Funktion wird später in der Arbeit näher eingegangen. (siehe Kapitel 9.3.2 Keyframe Animationen mit Control Rigs, S. 146)

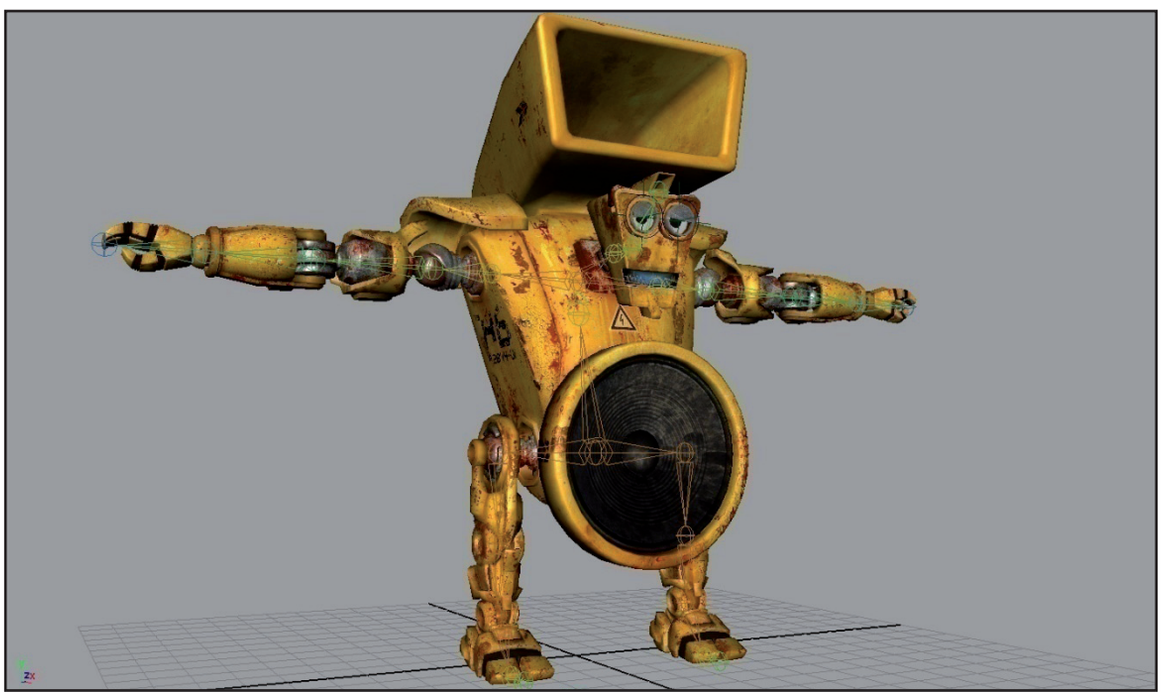


Abbildung 45: Bassbot - T-Pose mit FK-Rig

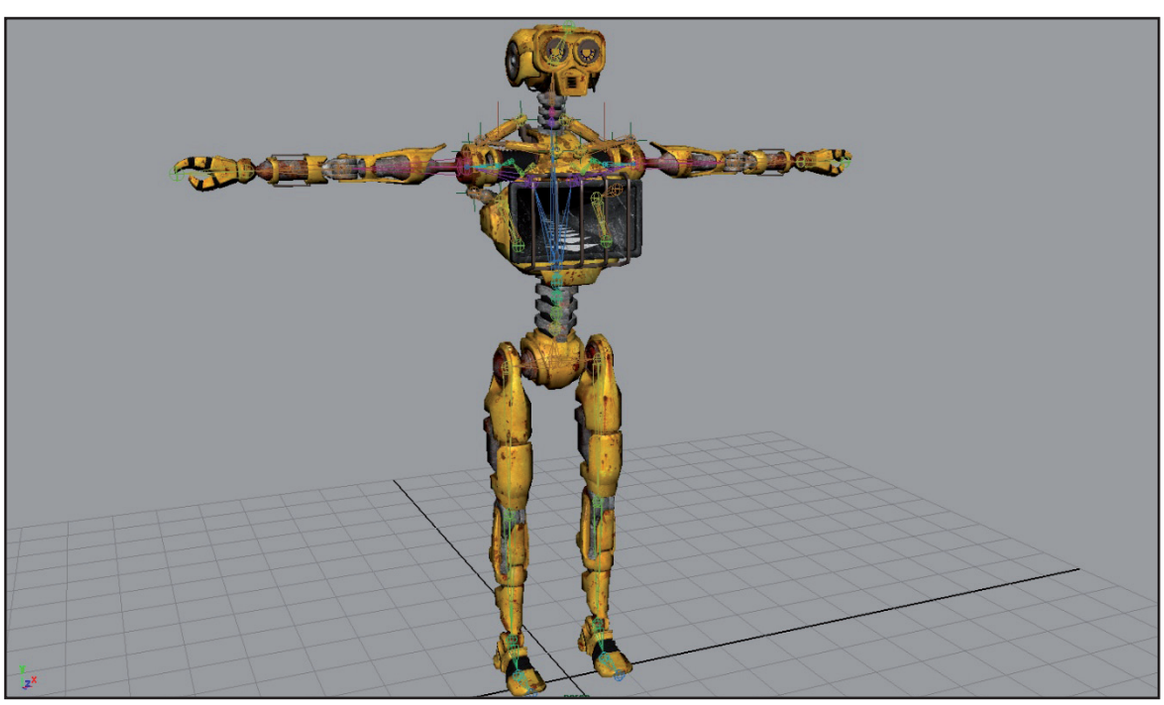


Abbildung 46: Treble Bot - T-Pose mit FK-Rig

Bei den Protagonisten handelt es sich um human anmutende Roboter, die weitestgehend dem Aufbau des menschlichen Körpers entsprechen. In diesem Fall spricht man von einem sogenannten Biped – „the most common skelton setup situation [...] (the character that walks on two legs)“ (Vogel et al. 2000, S.336).

Der Aufbau eines Bipes folgt immer einer bestimmten hierarchischen Anordnung, welche im nächsten Abschnitt genauer beschrieben wird.

8.2.1 Hierarchie

„A hierarchy is simply a way to tell the computer how the parts of your character are linked.“ (Maestri 1999, S.118) Über die hierarchische Struktur werden Zusammengehörigkeiten bestimmter Knochenelemente definiert. Bei einem klassischen Biped entspricht die Spitze der Hierarchie, auch Root genannt, immer der Hüfte. Ausgehend von der Hüfte entspringen weitere hierarchische Strukturen sowohl für die Beine, wie auch für die Wirbelsäule. Die Armstrukturen beginnen mit den jeweiligen Schulterknochen, welche, wie Hals und Kopf, ebenfalls Unterstrukturen der Wirbelsäule sind.

Abbildung 47 (S. 105) zeigt den klassischen, hierarchischen Aufbau eines Bipeds als Diagramm. Die Richtungspfeile geben dabei an, wo sich untergeordnete hierarchische Elemente befinden. Bei dieser Grafik handelt es sich jedoch um eine simplifizierte Darstellung. Jeder dargestellte Node kann aus mehreren Bone-Joint Strukturen bestehen.

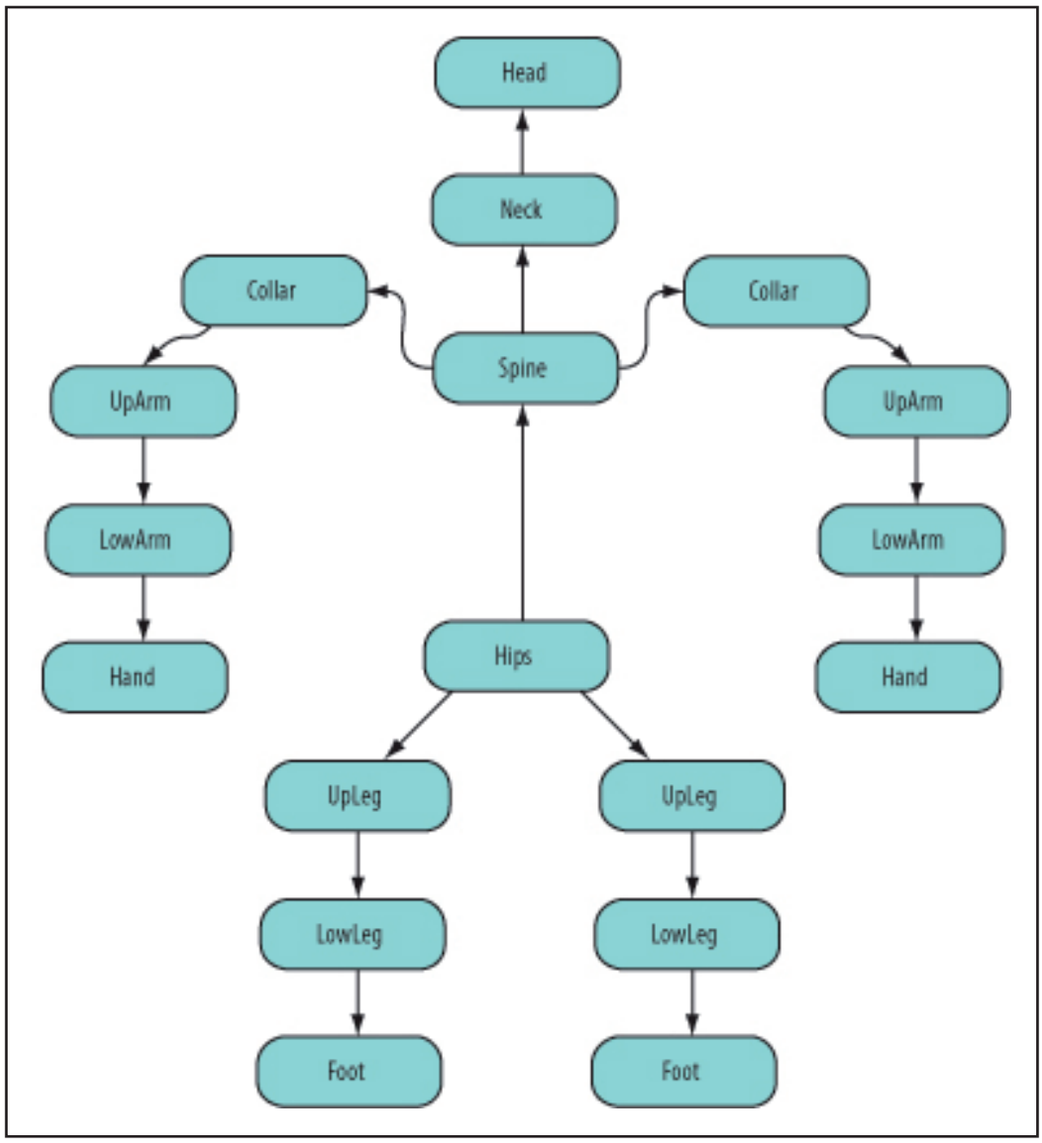


Abbildung 47: hierarchische Grundform eines Biped Rigs

Folgende Abbildung zeigt die Skeletthierarchie des Bassbots (siehe Abbildung 48, S. 106) . Dieser Character verfügt, seiner physischen Präsenz entsprechend, weder über eine Wirbelsäule noch über ein Schlüsselbein. Um jedoch einen fehlerfreien Transfer von Motion Capture Daten zu gewährleisten, sind diese Knochenelemente essentiell. (siehe Kapitel 9.3.3.2 MoCap Daten, S. 149)

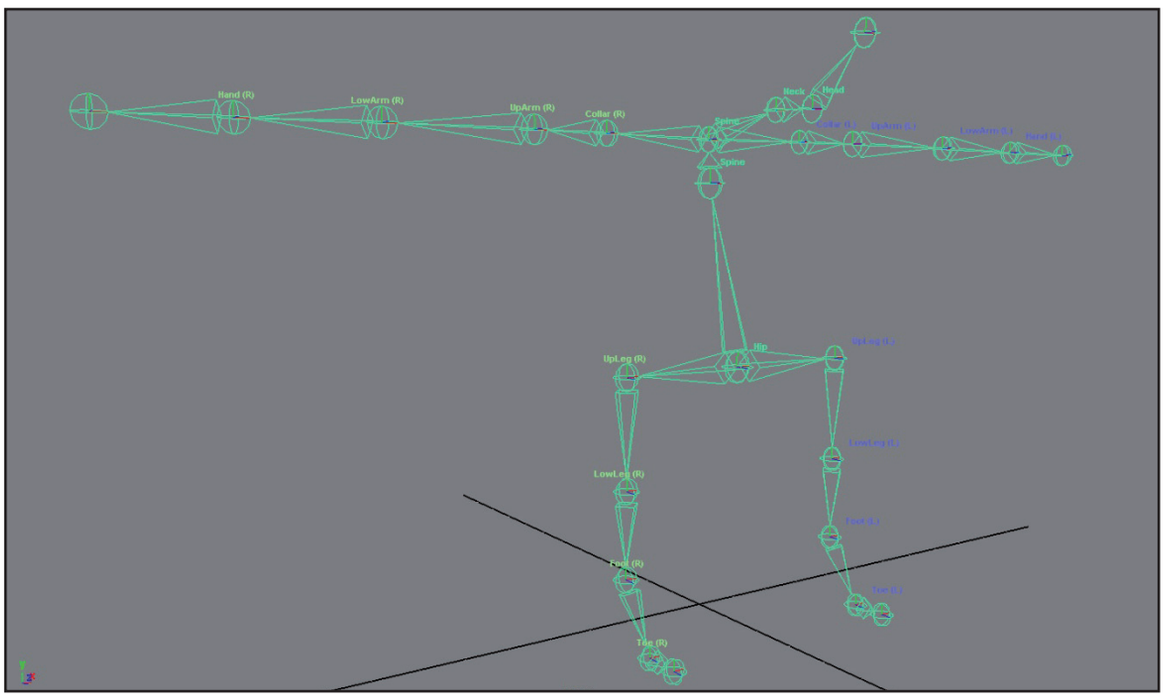


Abbildung 48: Bassbot - Maya IK-Rig

Wie in Abbildung 49 ersichtlich handelt es sich bei dem Rig um ein geradliniges Setup. In dieser Phase sind ausschließlich Forward Kinematics (FK) verwendet worden. (siehe Kapitel 8.2.2 Kinematics, S. 108) “When you rotate a joint that is the parent of other joints, all the children inherit the same rotation.“ (Vogel et al. 2000, S.326)

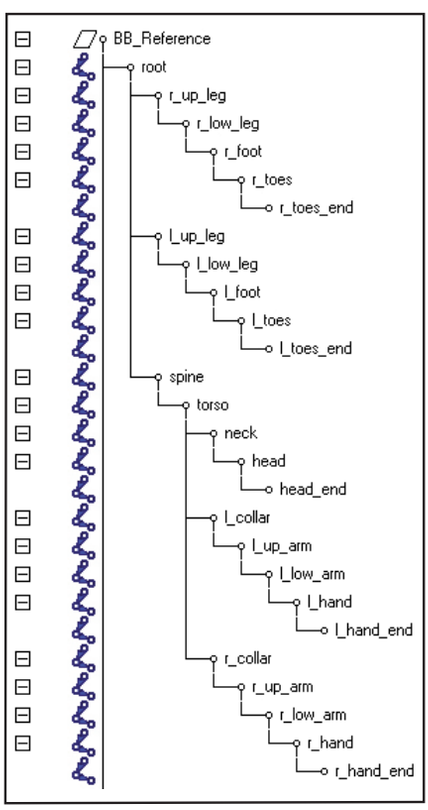


Abbildung 49: Bassbot - Outliner - Maya IK-Rig

Die Pipeline über Autodesk MotionBuilder ermöglicht es, ein einfaches und schnell erzeugtes FK Setup eines Biped Characters in ein komplexes und hervorragend steuerbares Character Setup zu transformieren, welches über alle nötigen Control Objects und IKs (siehe Kapitel 8.2.2 Kinematics, S.108) verfügt, um eine zielorientierte und effiziente Keyframe Animation zu ermöglichen. Die Funktionen und Vorteile Inverser Kinematics werden im folgenden Abschnitt näher behandelt.

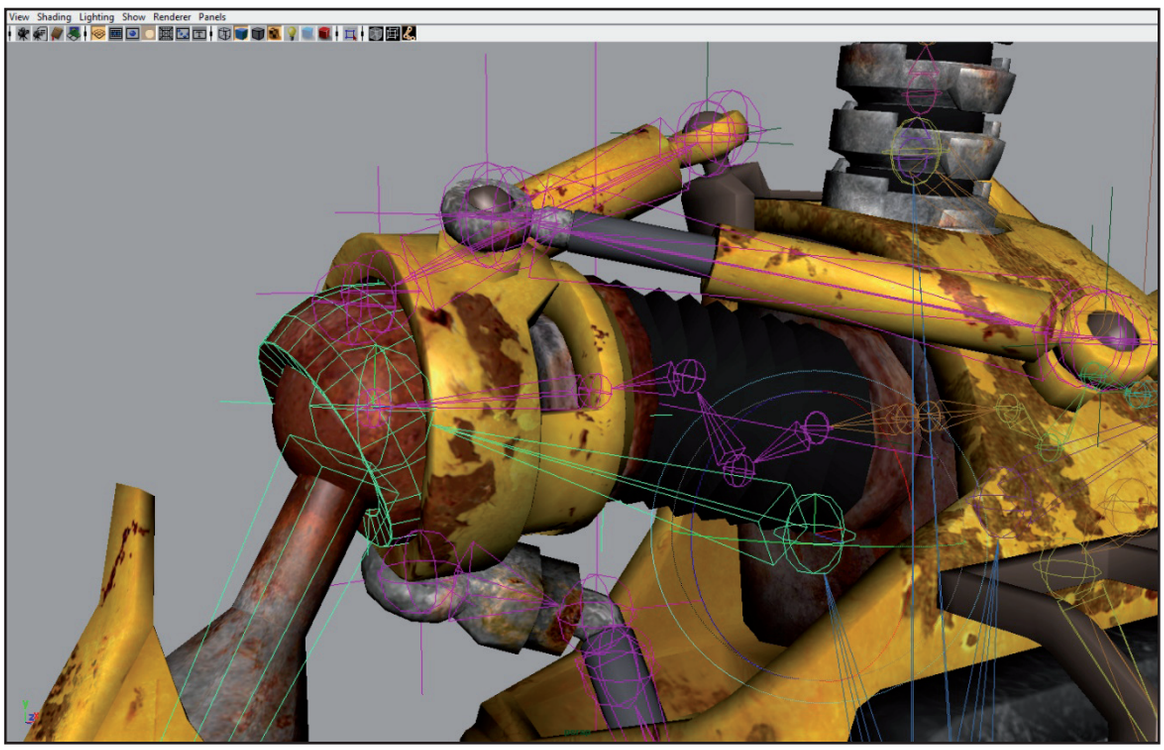


Abbildung 50: Treble Bot - Schulter Setup

Das Schulter Setup des Treble Bots ist ein gutes Beispiel um die Funktionsweise einer hierarchischen Struktur zu verdeutlichen (siehe Abbildung 50). Die Schulter selbst besteht aus vielen kleinen Einzelteilen. Drei hydraulische Elemente, bestehend aus Stempel und Zylinder, verbinden das Schultergelenk mit dem Oberkörper des Characters. Die Streben sollen sich entsprechend der Schulterbewegung ineinander schieben, wobei das Schlauchelement die Funktion einer Verbindungsmuffe einnimmt. Es ist jedoch für das weitere Verfahren unbedingt notwendig, dass das komplette Schulter Setup über die Rotation eines einzigen Joints zu steuern ist, dem Collar (in der Grafik ausgewählt). Nur auf diese Weise weicht das Setup nicht von der „Standard-Hierarchie“ eines Bipeds ab, wodurch sichergestellt wird, dass das Setup auch bei einem Transfer von Motion Capture Daten einwandfrei funktioniert.

Nullen, die mit einem Parent dem Oberarm beziehungsweise dem Torso untergeordnet sind, bilden die Grundlage für die Funktionalität der Hydraulikstreben. An diesen hängt jeweils ein Bone, der mit einem Aim Constraint auf die entsprechende Null der Gegenseite versehen ist. Dieses Konstrukt erfolgt beidseitig für jede der drei Streben eines Schultergelenks. Die Verbindungsmuffe

wird über eine IK-Chain (siehe Kapitel 8.2.2 Kinematics, S. 108) gesteuert, dessen Root mit dem Torso verbunden ist. Der IK-Handle befindet sich auf Höhe des vorletzten Gliedes der eingezogenen Bone-Chain und ist mit einem Offset Wert an den Oberarm mit einem Point Constraint versehen. Auf diese Weise folgt die IK-Chain der Position des Oberarm Joints, jedoch verbleibt das letzte Glied der Kette immer normal zum Torso.

8.2.2 Kinematics

In diesem Abschnitt der Arbeit werden unterschiedliche Methoden beleuchtet, welche die Manipulation von Skelettstrukturen und somit eine Animation beziehungsweise eine Erstellung von Bewegungsabläufen ermöglicht.

Forward Kinetamitcs (FK)
Bezüglich der Manipulation von Bone-Joint Strukturen beschreibt das Prinzip der Forward Kinematics einen so genannten “top-down-approach” (vgl. Maestri 2000, S.124). Auf diese Weise ist es dem Animator möglich, einzelne Bones von der Spitze der Hierarchie aus zu manipulieren. Bei einer Translation der Hüfte beziehungsweise der Root der Skeletthierarchie wird somit das komplette Rig bewegt. Will man zum Beispiel die Hand an eine bestimmte Stelle rücken, muss mit dem Oberarm begonnen werden, anschließend wird der Ellenbogen rotiert und abschließend erst die Hand. Diese Vorgehensweise kann bei komplexen Animationsprozessen schnell ausarten, da Animationskurven für jeden einzelnen Joint bearbeitet werden müssen.

“When only a few joints are available to select [...], the forward kinematics process is very simple and straightforward. However, when you must manipulate multiple joints, this process can get very complex and tedious.” (Vogel et al. 2000, S.328)

Mit Forward Kinematics ist es nicht möglich die Hand an eine bestimmte Stelle zu positionieren, ohne die übergeordneten Hierarchien ebenfalls manipulieren zu müssen. Um genau dies zu bewerkstelligen werden Inverse Kinematics verwendet.

Inverse Kinematics (IK)

Inverse Kinematic beschreibt eine alternative Art und Weise Knochenstrukturen zu steuern. Das IK System erlaubt eine Manipulation mehrerer Bones über die Translation eines sogenannten IK-Handels beziehungsweise End-Effectors. Eine Bewegung der Child-Nodes bewirkt also auch eine Bewegung der übergeordneten Parent-Nodes. Die Translation des Effectors steuert quasi die Rotationen aller in der IK-Chain-Hierarchie befindlichen Joints.

„IK Solvers can really streamline the animation of complex skeleton joint hierarchies. There will also be fewer animation curves in your scene; all the animation for the skeleton chain will be on the IK handle and not each individual joint." (Vogel et al. 2000, S.329)

Mit Hilfe von Up-Vektoren lässt sich die Richtung steuern, in welche eine Chain bei einer Beugung zeigen soll. Dieser ist zum Beispiel bei einem IK-Ellenbogen-Setup unumgänglich, um die Haltung des Armes bei einer Beugung des Ellenbogens zu kontrollieren.

Control Objects

Control Objects, auch Handles genannt, sind Hilfsobjekte, welche die Übersicht im Animationsprozess verbessern und die zu animierenden Parameter reduzieren. Meist handelt es sich dabei um Null- oder Curve-Objects, die entweder die Rotation bestimmter Joints, die Translation von Effectors oder auch bestimmte Transformationsparameter von ausgewählten Mesh-Objects steuern. Auf diese Weise kann verhindert werden, jeden einzelnen Joint mit Keyframes versehen zu müssen. Weiters sind Control Objects in einer Szene wesentlich leichter zu selektieren als unter dem Mesh verborgene Joint-Bone Strukturen.

Um die Funktion von Control Objects zu verdeutlichen wird an dieser Stelle der Arbeit das fertige Character Setup des Banker Bots näher erläutert, welches mit Autodesk Softimage umgesetzt worden ist. Dieses Setup wird ausschließlich für Vordergrundanimationen verwendet, wohingegen für Massive ein eigenständiges und simpler aufgebautes Rig in Maya erstellt worden ist (siehe Abbildung 51, S. 110). Die Animation des massentauglichen Banker Bots erfolgt in weiterer Folge in Massive Prime selbst über das Brain. (siehe Kapitel 10.8.4.1 Terrain Adaption, S. 206)

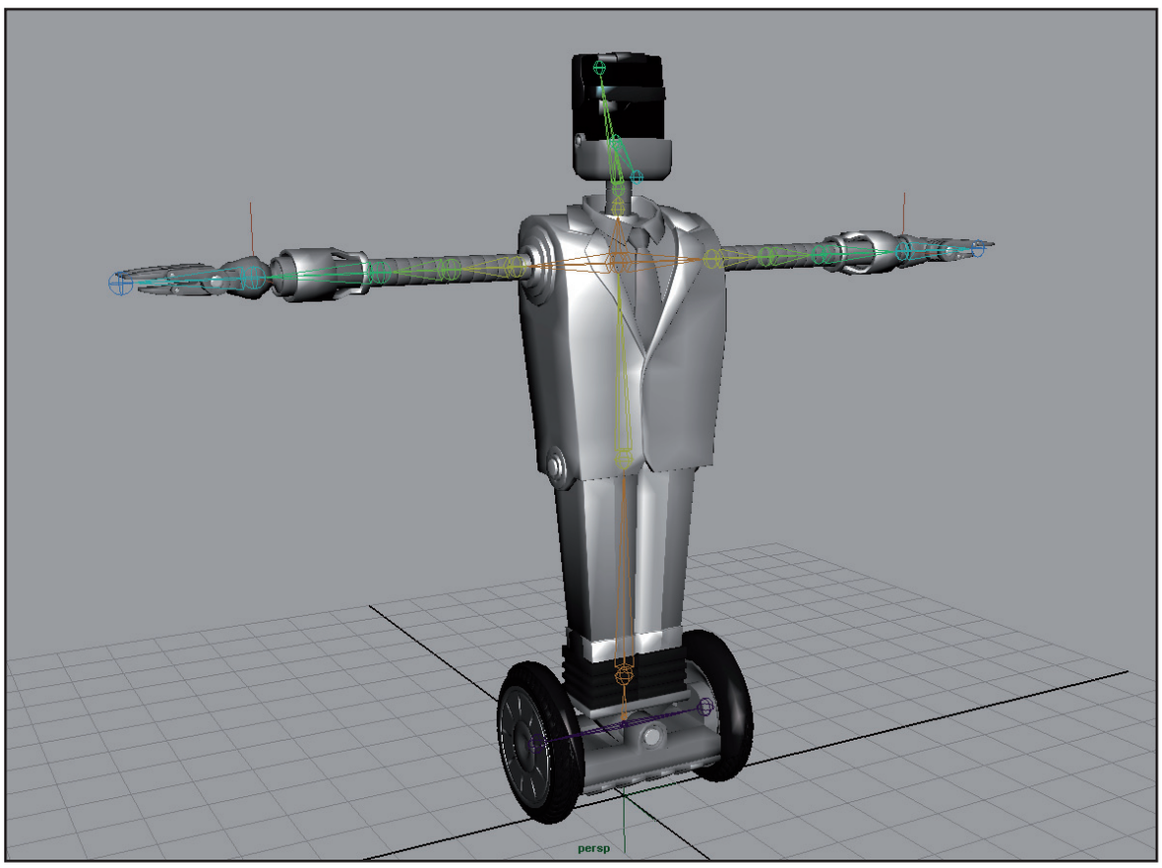


Abbildung 51: Banker Bot- Maya Rig

Aufgrund der Tatsache, dass es sich bei diesem Character nicht um einen klassischen Biped handelt, ist die Pipeline über MotionBuilder unpassend. Aus diesem Grund ist für diesen Character ein Full-Body-IK-Rig in Softimage erstellt worden, welches über alle nötigen Control Objects und IKs verfügt, um zielorientiertes und produktives Animieren innerhalb von Softimage zu ermöglichen. Für jede Bewegung, die die Physiognomie des Banker Bots erlaubt, steht ein Control Object zur Verfügung, welches entweder über Rotation oder Translation gesteuert wird. (siehe Abbildung 52, S. 111)

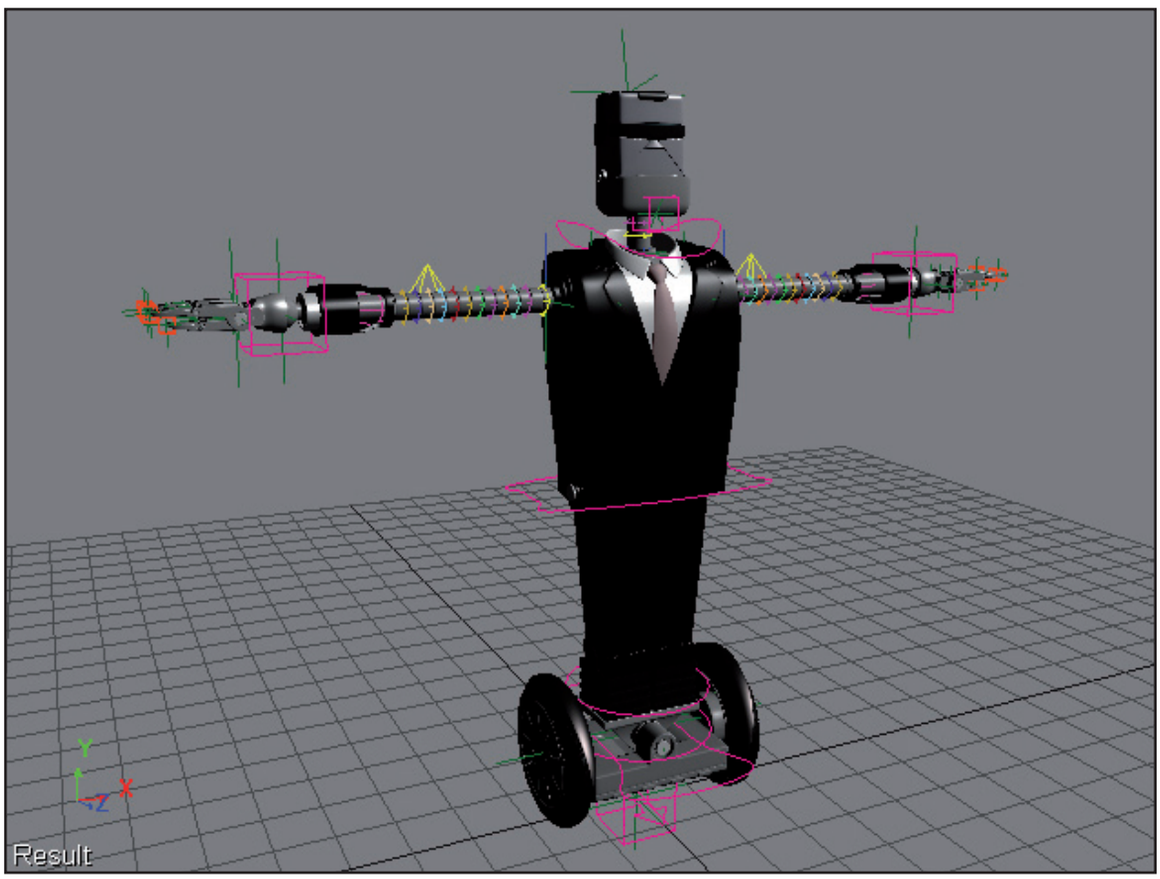


Abbildung 52: Banker Bot - Softimage Character Setup

Die Arme bestehen jeweils aus einer IK-Chain mit einem Up-Vector, die über eine Null gesteuert wird. Die Translation der Null gibt an, wie sich der Arm beugen soll und die Rotation der Null steuert die Rotation der Hand. Für Hüfte, Knöchel, Schwenkelement und Hals stehen Curve Objects zur Verfügung, welche die Rotation der jeweiligen Joints steuern. Die Reifen verfügen über eine Expression, welche die jeweilige Rotation in Bezug auf die entsprechende Translation des Rigs steuert. (siehe Kapitel 8.2.4 Skinning, S. 114)

Softimage und Maya unterscheiden sich weniger durch die Funktionsweise der einzelnen Skelettelemente, als durch deren Aufbau. In Softimage entsprechen 2D-Chains automatisch einer Inverse Kinematics Struktur. Eine Chain besteht immer aus einer Root, mindestens einem Bone und dem End-Effector. In Maya hingegen werden automatisch Forward Kinematic Bones erstellt. Über das IK Handle Tool können aber bestimmte Bone-Hierarchien als IK-Chains definiert werden. Diese verfügen dann ebenfalls über einen steuerbaren End-Effector, der in Maya IK-Handle genannt wird.

„Using IK solvers also allows the animator to lock skeleton chains in place. This is great when you are trying to keep feet pinned to the ground or hands constrained to an object.“ (Vogel 2000, S.329)

Bei der Erstellung von Bipeds zeigen sich genau bei diesem Aspekt die Vorteile der Pipeline über Autodesk MotionBuilder. Mit diesem Softwarepaket ist es möglich, aus einem aus Maya exportierten simplen FK Rig ein sogenanntes Control-Rig zu erstellen, welches über alle nötigen IKs sowie die entsprechenden Control Objects verfügt, um einen Character schnell, bequem und vor allem effizient animieren zu können. (siehe Kapitel 9.3.2 Keyframe Animationen mit Control Rigs, S. 146)

Um die Pipeline von Bipeds jedoch möglichst effizient nutzen zu können, müssen alle Bones beziehungsweise Joints einer bestimmten vordefinierten Konvention entsprechend benannt werden.

8.2.3 Name Conventions

Da sich MotionBuilder und Massive bezüglich der bevorzugten Namensgebung der Bones unterscheiden, ist es sinnvoll, sich für eine Konvention zu entscheiden und diese auch strikt in allen Softwarepaketen einzuhalten. In diesem Fall fiel die Entscheidung auf Massive Prime.

Die folgende Abbildung zeigt den Character Definition Prozess in MotionBuilder, mit dessen Hilfe deklariert wird, welche Bones des importierten Rigs welchen Körperteilen eines Bipeds entsprechen (siehe Abbildung 53, S. 113). Dieser Vorgang ist unumgänglich um MotionBuilder mit all den gewünschten Features zur Gänze ausnutzen zu können. Falls mit den Name Conventions von MotionBuilder gearbeitet wird, kann dieser Prozess automatisiert erfolgen. Andernfalls müssen die einzelnen Bones den entsprechenden Körperteilen per Hand zugewiesen werden.

Um jedoch möglichst effizient mit nicht-konformen Name Conventions arbeiten zu können, ist es ratsam, innerhalb von MotionBuilder ein Template zu erstellen, welches die von MotionBuilder bevorzugte Namensgebung der gewünschten Namensgebung gegenüberstellt und miteinander verknüpft. Das erstellte

Template kann für jeden weiteren Import von Skeletthierarchien desselben Characters wiederverwendet werden. Dies trifft sowohl für Maya Rigs zu, als auch für Animationsdaten aus 3ds Max und dessen Character Animation Plug-In CAT. (siehe Kapitel 9.1 Prozedurale Animation von CG Characters, S. 120)

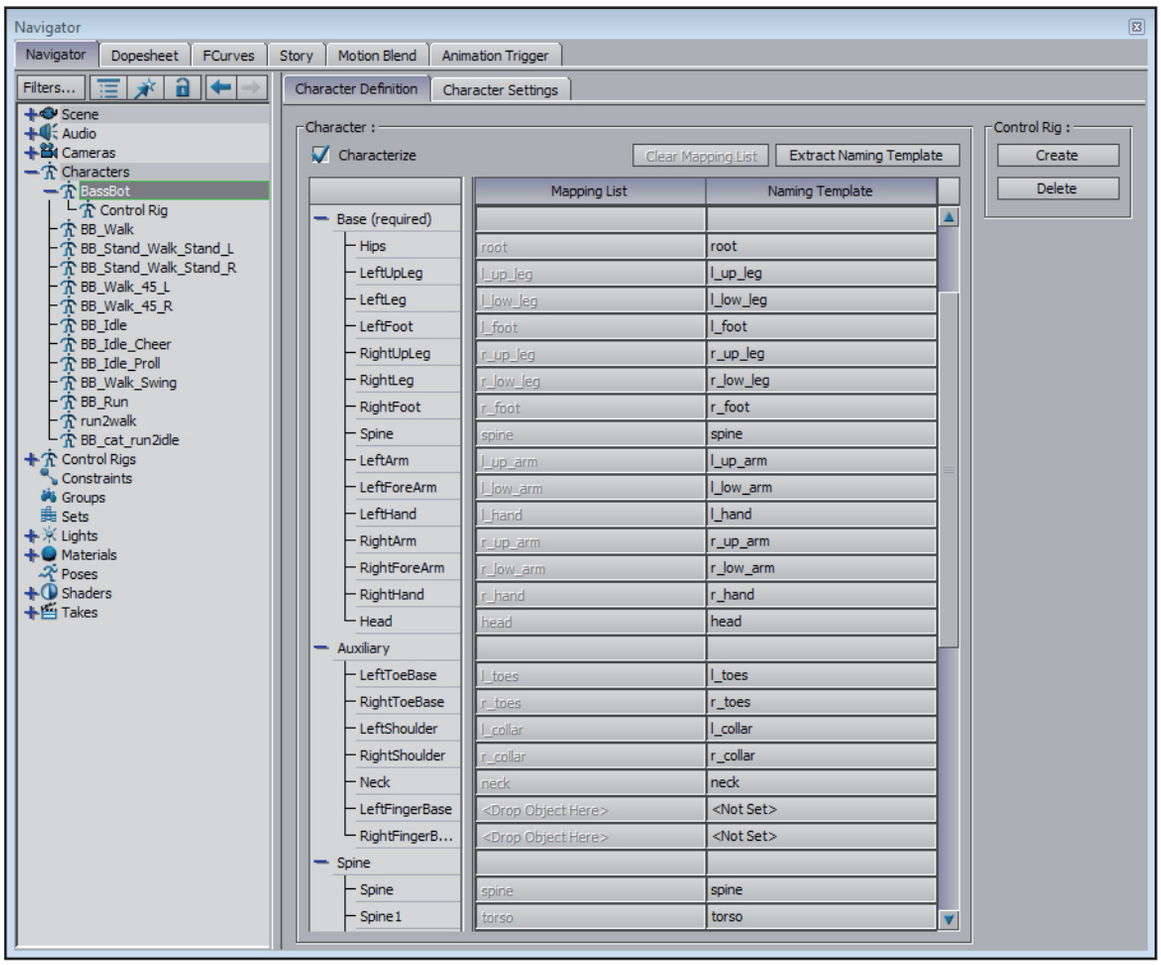


Abbildung 53: Gegenüberstellung der unterschiedlichen Name Conventions für die essentiellsten Elemente eines Biped Rigs

Falls mehrere Instanzen eines Characters in einer Szene verwendet werden und diese über mehr als nur einen Spine oder Neck Bone verfügen, ist es anzuraten, diese nicht mit fortlaufenden Nummern zu versehen, sondern über Buchstaben von a bis z zu differenzieren. In MotionBuilder können keine Namen für Bones doppelt existieren. Falls beim Import einer Skeletthierarchie ein Name bereits vorhanden ist, wird der entsprechende Bone mit einer fortlaufenden Nummer versehen. Ist in der anfänglichen Namensgebung bereits mit Nummern (z.B. spine3) gearbeitet worden, kann dies zu Komplikationen bei der Character Definition führen.

Dies hat zur Folge, dass im Definitionsprozess falsche Verknüpfungen geschlossen werden, die sich wiederum auf ein fehlerhaft funktionierendes Control-Rig auswirken (z.B. spine3 wird zu spine4).

Die Einhaltung einer einheitlichen Namensgebung in allen Softwarepaketen ist essentiell, um eine reibungslose Kommunikation zu gewährleisten. Aus diesem Grund entsprechen die Namen des aus Maya exportierten Rigs den Konventionen von Massive Prime, damit am Ende der Pipeline keine Komplikationen auftreten.

8.2.4 Skinning

Skinnig beschreibt im Allgemeinen den Prozess, in dem bestimmte geometrische Objekte den entsprechenden Elementen des Rigs zugewiesen werden. Für diesen Zweck existieren verschiedene Möglichkeiten, auf die in diesem Kapitel näher eingegangen werden soll. Primär wird zwischen Parents, Constraints und Weights unterschieden.

Parents (Direct Assignments)
Parenting beschreibt eine starre Verbindung zwischen Bone und geometrischem Objekt. Dabei handelt es sich ebenfalls um eine Eingliederung beziehungsweise Unterordnung in der hierarchischen Struktur. Der Nachteil dieser Verbindung besteht darin, dass sich Geometrien nicht verformen lassen (vgl. Schönherr 2000, S.157). Falls keine Deformationen des Meshes gewünscht sind, sollten Parents oder Constraints bevorzugt eingesetzt werden, da keine aufwendigen Verformungen in Echtzeit berechnet werden müssen (vgl. Maestri 1999, S.55). Dies ist wiederum von Vorteil um die Szene flüssig und ruckelfrei zu halten.

Weight-Assignments
Das so genannte Enveloping oder Binding wird verwendet um eine Geometrie an mehrere Bones zu binden. Über Weights kann der Einflussbereich einzelner Bones auf bestimmte Teile der Geometrie definiert werden.

„For a more organic deformation, you may need a vertex to be affected by more than one joint. [...] Many packages have a visual reference, typically known as an envelope. An envelope is basically a range of influence. Where the envelopes overlap, the vertices are affected accordingly.“ (Maestri 1999, S.135)

Bezüglich Skinning wird abgesehen von Cartoon Characters grob zwischen zwei verschiedenen Character Arten differenziert – dem „Segmented Character“ und dem „Single Mesh Character“ (vgl. Vogel et al. 2000, S.330).

Bei einem Segmented Character handelt es sich um ein Model, welches aus mehreren Einzelteilen besteht, wie es beim Bassbot oder Treble Bot der Fall ist. Im Gegensatz dazu steht der Single Mesh Character, bei dem eine Zuordnung zu dem jeweiligen Knochen im Gerüst rein über das Enveloping mit Weights gesteuert wird.

Da es sich bei dem Projekt ausschließlich um animierte Roboter handelt, werden die Funktionsweisen von Muskel Systemen und „stretchy Bones“ in dieser Arbeit nicht näher erläutert.

Constraints

Bei einem Segmented Character können die Zuordnungen sowohl über Parents als auch über Point- und Orient- Constraints erfolgen. Constraints beschreiben eine transformspezifische Verknüpfung zwischen zwei Objekten. „A [point] contstraint tells one object to stick to another object.“ (Maestri 1999, S.127) Dafür ist es aber unbedingt nötig, dass der entsprechende Pivot des Objekts – „das ist der Schwerpunkt, um den sich Objekte bewegen und drehen“ (Schönherr 2000, S.157) – an der richtigen Stelle sitzt und auch die entsprechende Achsausrichtung besitzt. Die Pivots einzelner Objekte müssen bezüglich Position und Rotation mit dem Joint, mit dem sie über einen Constraint verbunden sind, übereinstimmen, da der Pivot des Objektes den Center des Joints sowie dessen Rotationswerte einnimmt.

„Unfortunately, in character animation, arbitrary points are not acceptable“ (Maestri 1999, S.120)

Expressions

Expressions sind ein weiteres hilfreiches Mittel im Character Setup Prozess. Diese ermöglichen es, bestimmte Parameter verschiedener Objekte miteinander zu verknüpfen beziehungsweise in eine mathematische Relation zueinander zu setzen. Softimage bietet für genau diese Zwecke ein praktisches Tool an, den „Parameter Connection Editor“. In Maya werden Verknüpfungen dieser Art mit Hilfe des „Expression Editors“ vollzogen.

In der Fallstudie wurden Expressions einerseits eingesetzt um den Innenring des Schultergelenks des Treble Bots zu definieren, der die Armrotation zu fünfzig Prozent übernimmt. Andererseits wurde die Rotation der Reifen des Banker Bots über eine Expression gesteuert, welche die Relative Z-Translation sowie die Absolute Y-Rotation des kompletten Rigs in ein Verhältnis zu der X-Rotation des Reifens setzt. Folgende Formel zeigt die Banker Bot Wheel-Expression aus Maya, basierend auf der Formel der Länge eines Kreisbogens `Tx=2r(α/360)`.

```
Achsradius r = 1.552
Reifenradius l = 1.23
bankerBot_move = hierarchisch übergeordnete Null (Steuerelement)

l_wheel.rotateX=((((-bankerBot_move.translateZ+((bankerBot_move.
rotateY/360)*2*3.14*1.552))/(2*3.14*1.23))*360)*-1)

r_wheel.rotateX=((((bankerBot_move.translateZ+((bankerBot_move.
rotateY/360)*2*3.14*1.552))/(2*3.14*-1.23))*360)*-1)
```

$$\mathbf{wheel.rotateX}=\frac{\left(\mathbf{Bot\_move.translateZ}+\left(\left(\frac{\mathbf{Bot\_move.rotateY}}{\mathbf{360}}\right)*\mathbf{2l\pi}\right)\right)}{\mathbf{2r\pi}}*\mathbf{360}$$

8.2.5 Export und Transfer

Vor dem Export eines mit Rig und Skin versehenen Bipeds gibt es einiges zu beachten, damit das Rig in MotioBuilder einwandfrei funktioniert. Es ist hilfreich die komplette Rig-Hierarchie in einem leeren Referenz-Node zu gruppieren, mit dessen Hilfe der exportierte Character in MotionBuilder beliebig skaliert und positioniert werden kann.

Ein weiterer wichtiger Punkt, der nicht vergessen werden darf, beschäftigt sich mit den lokalen Rotationsachsen der einzelnen Joints. „Have your skeleton's rotation ‚zeroed' out in Maya so that there will be no rotational values left in the joint rotation channels when the skeleton is imported into MotionBuilder. [...] Orient all skeleton joints so that their local rotation axes become identical to the global space's axes." (Kitagawa et al. 2008, S.68)

Mit Hilfe der "Freeze all Transformations" Methode lassen sich in Maya alle Rotationswerte auf Null setzen. Die Option „joint orient" muss jedoch unbedingt aktiviert werden. Abbildung 54 zeigt das für den Massive Export simplifizierte Rig des Treble Bots, dessen Bones alle eine Orientierung eingenommen haben, die den globalen Achsen entspricht. Wird dieser Schritt nicht vollzogen, kommt es in MotionBuilder im Motion Blending Prozess zu Schwierigkeiten. (siehe Kapitel 9.3.4 Motion Blending, S. 158)

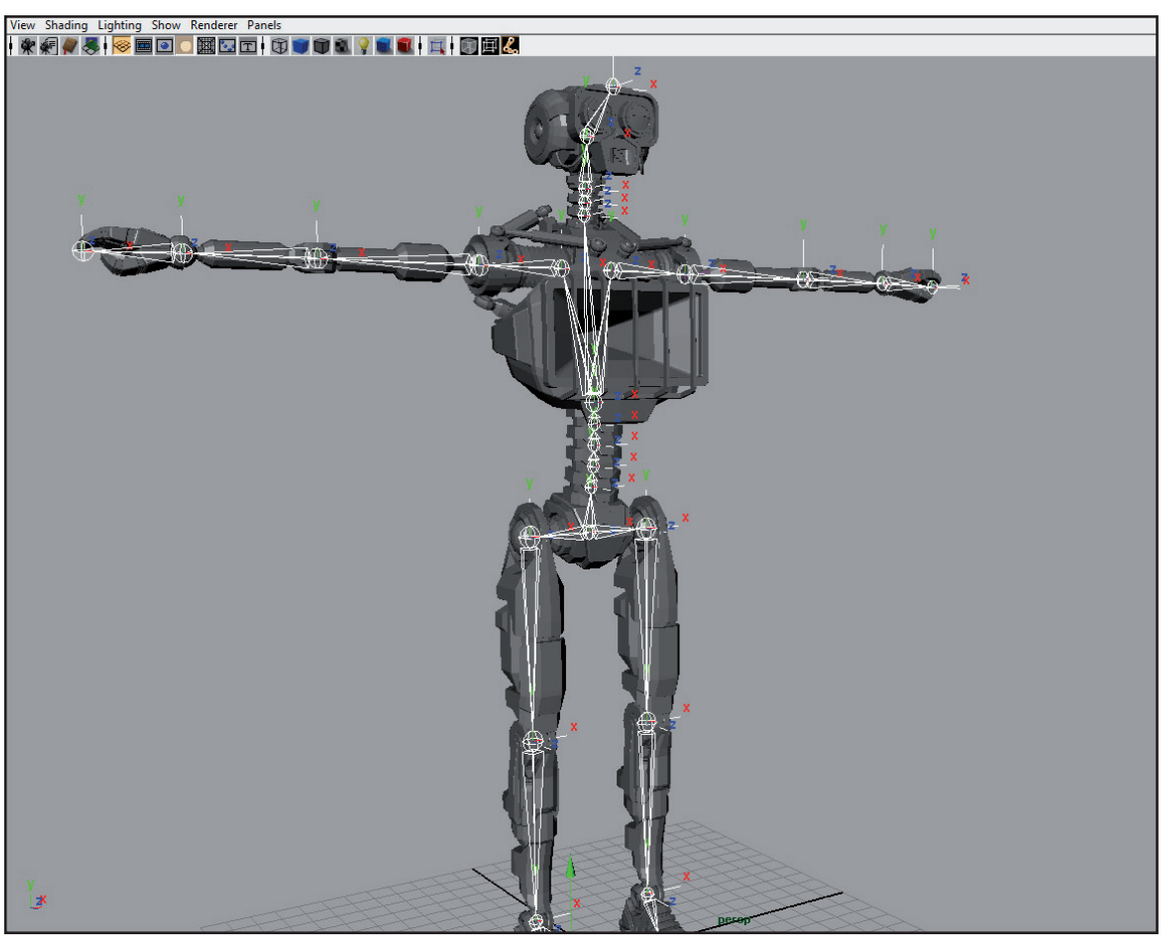


Abbildung 54: Treble Bot - Maya IK-Rig - lokale Rotationsachsen

„The end result after the blend may look fine, but when you are in the middle of using MotionBuilder's blend tool, the skeleton may look either skewed or as if it had been rotated 90 degrees about the hips. [...] it makes it almost impossible for you to see the result of the blend until the process is finished and rendered." (Kitagawa et al. 2008, S. 69)

Wenn alle Bones richtig benannt sind, alle Rotationen auf Null gesetzt sind und das Skinning des Characters abgeschlossen ist, erfolgt der Export als .FBX File für Autodesk MotionBuilder.

.FBX

Das .FBX Format wurde ursprünglich von Kydara für dessen 3D Animationssoftware Filmbox, dem Vorgänger von Autodesk MotionBuilder, entwickelt. Das .FBX Format ist ausgelegt Animationsszenen zu beschreiben, die von einer breiten Menge an 3D Softwarepaketen unterstützt wird. Geometrien, Texturen, Kameras, Lichter, Markerdaten, Skeletthierarchien und Animationsdaten können innerhalb eines .FBX Files gespeichert werden. (vgl. Kitagawa et al. 2008, S.183)

Für eine Animationsaufbereitung mit Hilfe der prozeduralen Animationsengine CAT (Character Animation Toolkit) wird lediglich das Rig ohne Skin ebenfalls als .FBX File exportiert, welches als Referenz für das in 3ds Max zu erstellende CAT-Rig dient. (siehe Kapitel 9.1 Prozedurale Animation von CG Characters, S. 120)

Das fertige Rig wird ebenfalls ohne Skin als Maya ASCII File (.MA) an Massive Prime übergeben. Dieses bildet die Grundlage für das Skeleton des Massive Agents, welches in der weiteren Folge mit Animationsdaten (ebenfalls .MA) versehen wird. (siehe Kapitel Abbildung 83: Lanes, S. 169)

Basierend auf den in diesem Kapitel erläuterten Character Rigs werden im folgenden Kapitel 9 Character Animation (S. 119) verschiedene Animations-techniken analysiert, Bewegungsdaten für die Verwendung in der Massen-simulationssoftware Massive Prime aufzubereiten.

9 Character Animation

Eine mittels künstlicher Intelligenz gesteuerte Masse stellt besondere Anforderungen an die Character Animation. Ähnlich wie bei Video-Spiel-Charakteren beschreibt die Konzeption der Bewegungsabläufe eine kritische Phase, da virtuelle Akteure benutzerdefiniert beziehungsweise über Fuzzy-Logik (siehe Kapitel 10.8.4 Brain-Module, S. 206) gesteuert agieren müssen. Um eine realistische Simulation einer fortschreitenden Masse generieren zu können, müssen ausreichend unterschiedliche Bewegungsabläufe abgespeichert, abrufbar und vor allem auch kombinierbar sein.

“Human motion is very complex. The way we walk, sit, run, jump, laugh, wave, fall, swim etc. are so ingrained in our memory, the smallest error jars the eye. [...] Thus when images are rendered with great realism, there is a natural expectation for the animation to be equally realistic. ” (Vince 2000, S.11)

Der Realismus in der Bewegung selbst stellt eine weitere Herausforderung dar. Aus diesem Grund kommen Motion Capture Techniken zum Einsatz, welche einen Transfer der Natürlichkeit einer Bewegung auf einen virtuellen Character ermöglicht. Motion Capture Libraries (.BVH Daten) sowie eigens erstellte Motion Capture Daten werden im Rahmen dieser Arbeit verarbeitet. Nahtlose Übergänge zwischen bestimmten Core-Animationen sowie die Generierung von Motion Loops sind für eine Massensimulation essentiell.

Um die quantitative Produktivität bezüglich der Erstellung von Motion Loops zu steigern, kommen auch prozedurale Animationstechniken zum Einsatz, die es ermöglichen, schnell und effizient Varianz in Bewegungszyklen zu generieren.

9.1 Prozedurale Animation von CG Characters

Das „Character Animaton Toolkit“ CAT ist ein Plugin für Autodesk 3ds Max 2010, welches sowohl Möglichkeiten für das Character Rigging bietet, als auch für unterschiedliche Animationserfahren. Dazu zählen non-lineare Keyframe Animation, Animation Layering, Motion Capture Import sowie Simulation von Muskelsystemen. Ein essentielles Feature, welchem im Rahmen dieses Projekts eine besondere Bedeutung zugeschrieben wurde, ist der CAT Motion Layer. Dieser ermöglicht die Generierung prozeduraler Motion Cycles (vgl. 3ds Max CAT Help 2010a). Gerade für die Erstellung von Bewegungsabläufen für Massensimulationen erweist sich der prozedurale Ansatz als günstige Alternative zu der klassischen Keyframe Animation beziehungsweise der Erstellung und Adaption von Motion Capture Daten.

Es gilt jedoch zu beachten, dass innerhalb von CAT eine eigene Rig Struktur verwendet wird, wodurch die Verwendung des in Maya erstellten Rigs nicht ermöglicht wird. Ein CAT Rig setzt sich aus Bones und sogenannten Hubs zusammensetzt. Ein Hub ist ein spezieller Typus von CAT Bone, welcher eine übergeordnete und organisierende Funktion im CAT Rig einnimmt. Normalerweise handelt es sich dabei um einen zentralen Punkt, von dem Wirbelsäule, Arme, Beine oder weitere Extremitäten entspringen. Ein Hub kann die Rolle einer Hüfte, eines Brustkorbes und auch eines Kopfes einnehmen.

Der erste erzeugte Hub entspricht wiederum der Root der kompletten Hierarchie. Bei Bipeds übernimmt auch diese Root im Normalfall die Funktion der Hüfte (vgl. 3ds Max CAT Help 2010b). Bei der Erstellung des Rigs existieren spezielle Funktionen für die Definition von Extremitäten, wie Add Arm oder Add Leg. Diese Parameter bilden die Grundlage für die Generierung von prozeduralen Motion Loops.

Mit Hilfe des CAT Motion Layers lassen sich nach erfolgtem Character Setup schnell und einfach verschiedenste Geh- und Laufzyklen mit variabler Länge des Loops erzeugen. Durch die Adaption der verschiedenen Parameter können mannigfaltige Varianten produziert werden. Diese Möglichkeit spiegelt auch den großen Vorteil wieder, den CAT für die Erstellung von Animationsdaten für Massenszenen bietet. Neben justierbaren Parametern für Extremitäten und Hubs stehen auch globale Variablen zur Verfügung, welche beispielsweise die Schrittgeschwindigkeit sowie die Schrittlänge definieren (siehe Abbildung 55).

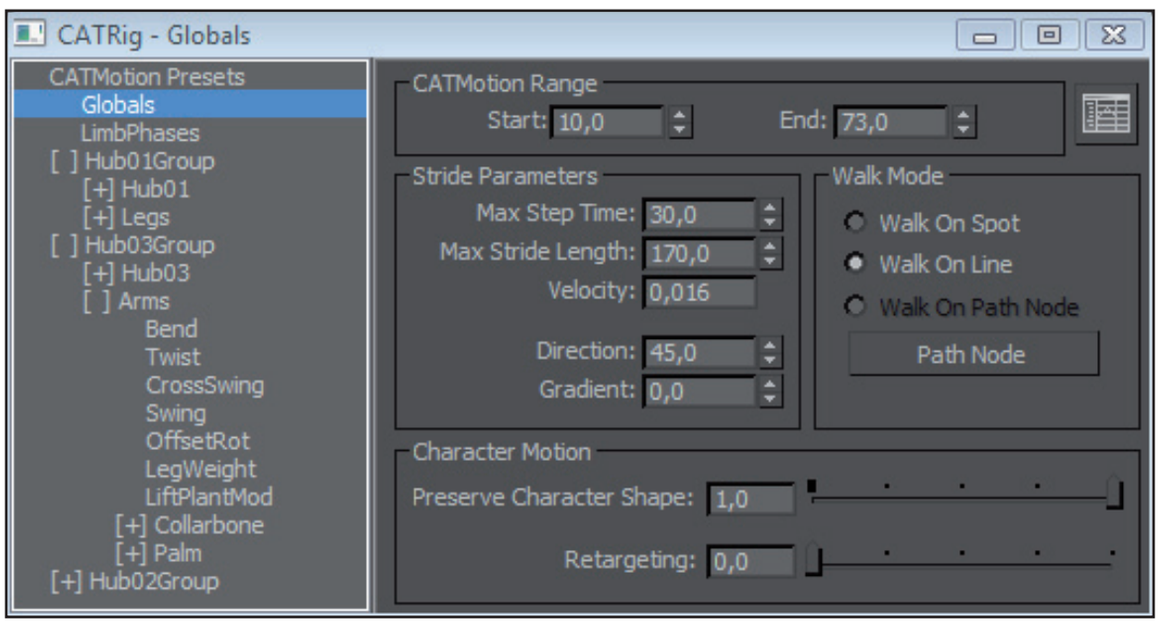


Abbildung 55: 3ds Max - CAT Rig - Globals

Für die realistische Darstellung von kollisionsvermeidenden Massen ist die Einbindung von Seitwärtsschritten unvermeidbar. Auch hierfür bietet CAT genau das richtige Feature, mit dessen Hilfe die Richtung definiert werden kann, in die sich der Character bewegen soll – 45 Grad links beziehungsweise rechts.

Ein weiterer Vorteil ergibt sich aus der Möglichkeit der Verknüpfung mit einem Path Node. Der animierte Character kann auf diese Weise einem erstellten Pfad folgen. In diesem Prozess werden automatisch Foot Step Controls erzeugt, welche die Auftrittsflächen des jeweiligen Fußes definieren. Durch die nachträgliche Adaption der Foot Steps lassen sich auch Animationen für das Losgehen beziehungsweise Stehenbleiben gut erzeugen (siehe Abbildung 56, S. 122).

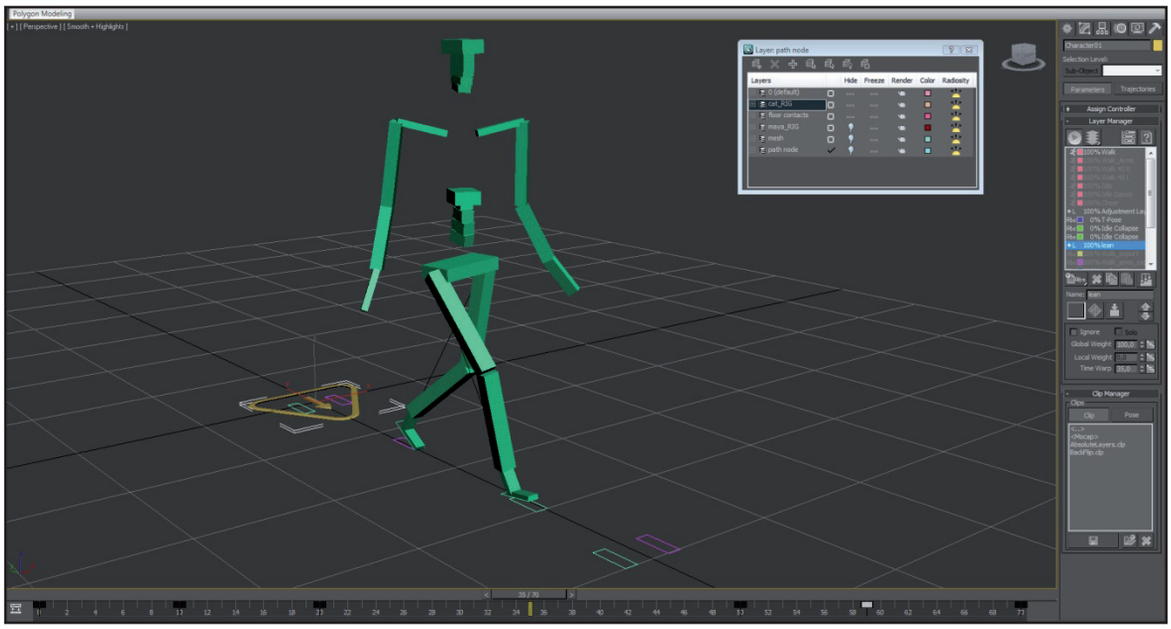


Abbildung 56: 3ds Max - CAT: Foot Step Controls

9.1.1 Workflow

Um die in CAT produzierte Animation wieder exakt auf das in Maya erstellte Rig transferieren zu können, muss das in CAT erstellte Rig über dieselben Proportionen und Hierarchien verfügen wie das Rig aus Maya. Für diesen Zweck wird das Maya Rig als .FBX importiert und dient als Referenz für das zu erstellende Rig in CAT. Die einzelnen CAT Bones und Hubs können nun exakt an die entsprechenden Joints des Maya Rigs mit Hilfe der Snap Funktion angeglichen werden. Auf diese Weise wird sichergestellt, dass sich die beiden Skelettstrukturen in keinster Weise unterscheiden (siehe Abbildung 57, S. 123). Natürlich muss auch innerhalb von CAT die entsprechende Name Convention eingehalten werden.

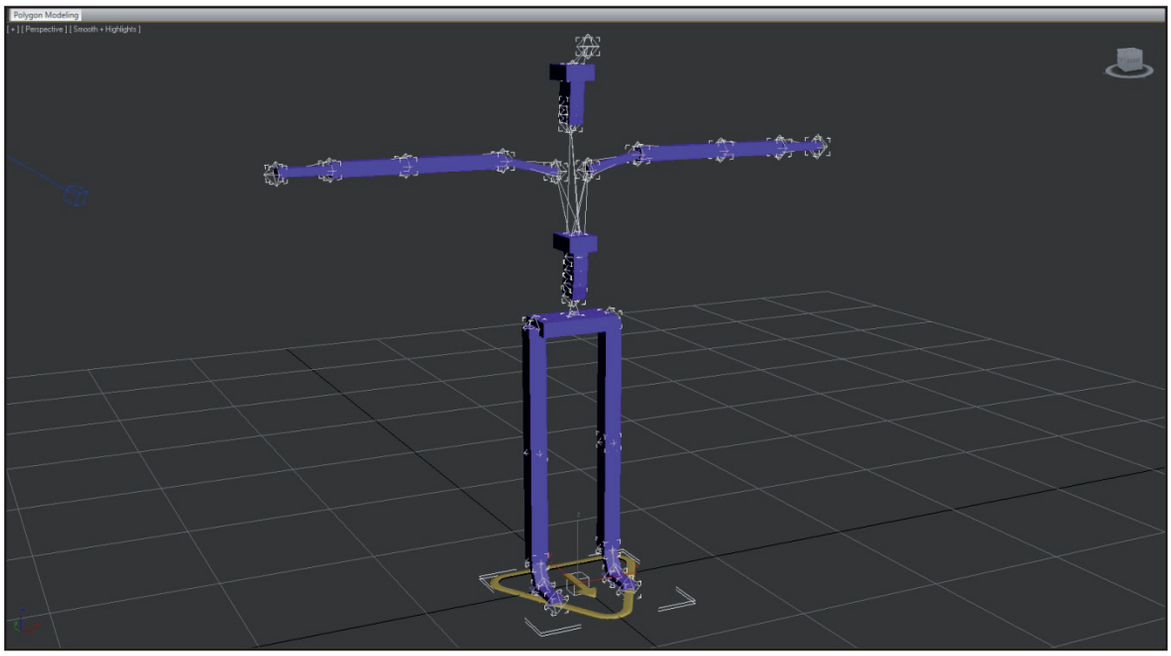


Abbildung 57: 3ds Max - CAT Rig über Maya Rig

Nach der Erstellung des CAT Rigs erfolgt ein erneutes Skinning des Meshes, welches in diesem Zusammenhang lediglich zur besseren Visualisierung der erstellten Bewegungsdaten dient. Ist das Character Setup innerhalb von CAT erfolgt, kann der Animation Layer für den prozeduralen Walk Cycle erzeugt und für den entsprechenden Walk adaptiert werden (siehe Abbildung 58).

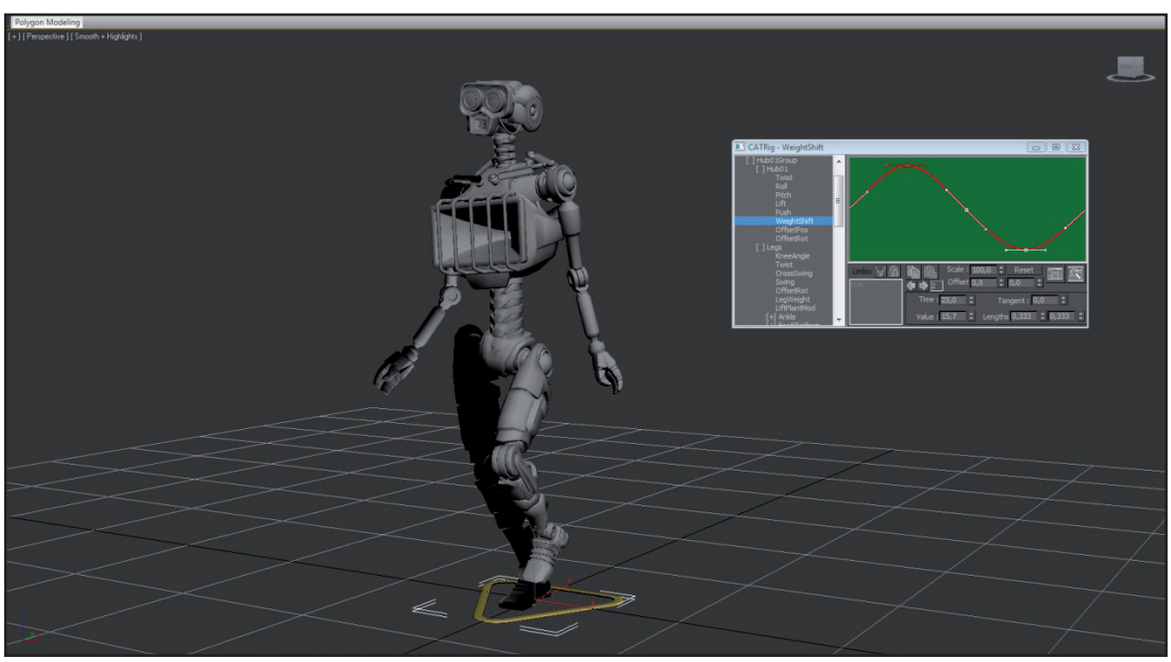


Abbildung 58: 3ds Max - CAT: Weight Shift

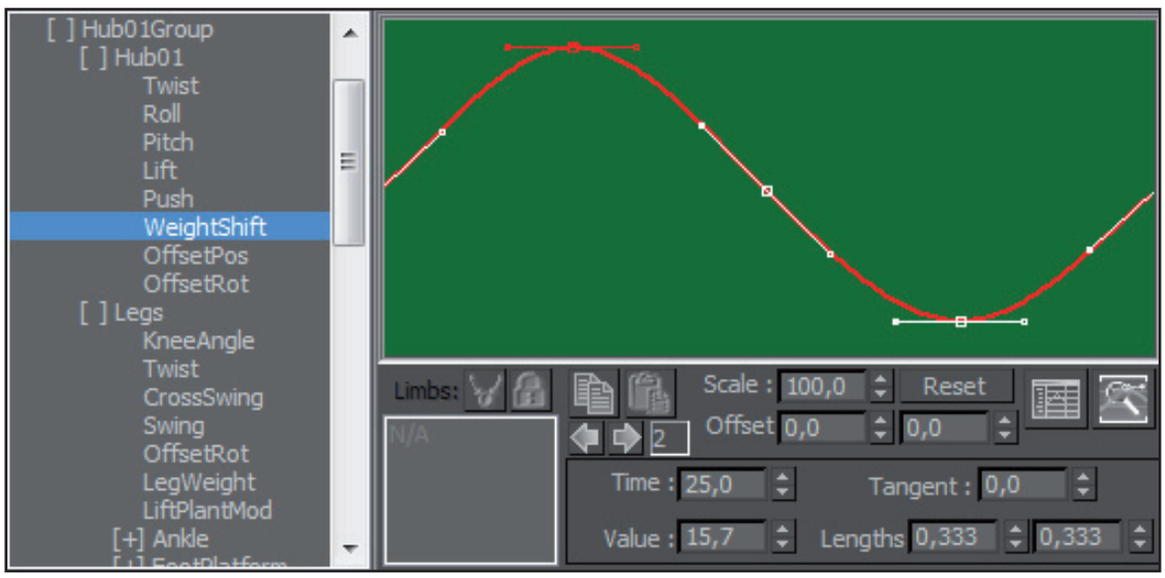


Abbildung 59: 3ds Max - CAT Motion Parameter

In der oberen Abbildung ist das Kontrollfeld für die verschiedenen Bewegungsparameter sichtbar (siehe Abbildung 59). Hub01 entspricht der Hüfte und beinhaltet die für diese Körperpartie möglichen Einstellungen. Weiters existieren Einstellungsmöglichkeiten für die Beine, die Knöchel, den Fuß, den Oberkörper, die Arme und den Kopf.

Jedoch stößt das CAT Motion Tool auch an seine Grenzen und ist für die Abbildung von komplexen, handlungsspezifischen Bewegungsabläufen ungeeignet. Der CAT Motion Layer ist primär für die Generierung von Basiszyklen geeignet (vgl. 3ds Max CAT Help 2010a). Jedoch können die einzelnen Zyklen auch innerhalb von CAT mit Hilfe zusätzlicher Animation Layers und den entsprechenden animierbaren Weight Werten gut erweitert werden.

Diese Technik wurde beispielsweise bei der „Stand to Walk“ und „Walk to Stand“ Animation verwendet. CAT differenziert dabei zwischen verschiedenen Layer Arten. Der CAT Motion Layer wurde in der Arbeit bereits behandelt. Weiters existieren globale sowie lokale und absolute Layer. Die folgende Abbildung zeigt den Layer Aufbau für die oben erwähnte Animation (siehe Abbildung 60, S. 125).

Absolute Layer verfügen über bestimmte abgeschlossene Bewegungsanimationen. Cat Motion Layer können beispielsweise auf einen absoluten Layer geplottet werden. Globale und lokale Layer beschreiben Komponenten, die ein nachträgliches Hinzufügen von weiteren Bewegungsparametern erlauben. Das jeweilige Attribut gibt Aufschluss über das in der Manipulation zu verwendende Koordinatensystem.

Ausgehend von der T-Pose (absoult Layer) geht die Animation über eine Idle Pose (absolut Layer) in den Walk (CAT Motion Layer). Nach dem Walk wird wieder eine Idle Pose (absolut Layer) ausgeführt. Der lokale Adjustment Layer, sowie der „lean“ Layer (local Layer) fügen Details in die Animation hinzu, nachdem die Basis Animation erstellt und zusammengefügt worden ist (vgl. Vogel et al. 2000, S405).

„When using the layering approach to animating a 3D character, the process is very organized and methodical. You start very generally and refine the motion with each layer and pass over the animation. [...] One of the drawbacks to using a layered approach is that you are working on only certain parts of the 3D character at a time. The layered approach is not an ideal method for animation where the character must achieve strong poses. [...] It is much better suited to motion cycles like a walk or run for instance.“ (Vogel et al. 2000, S. 417)

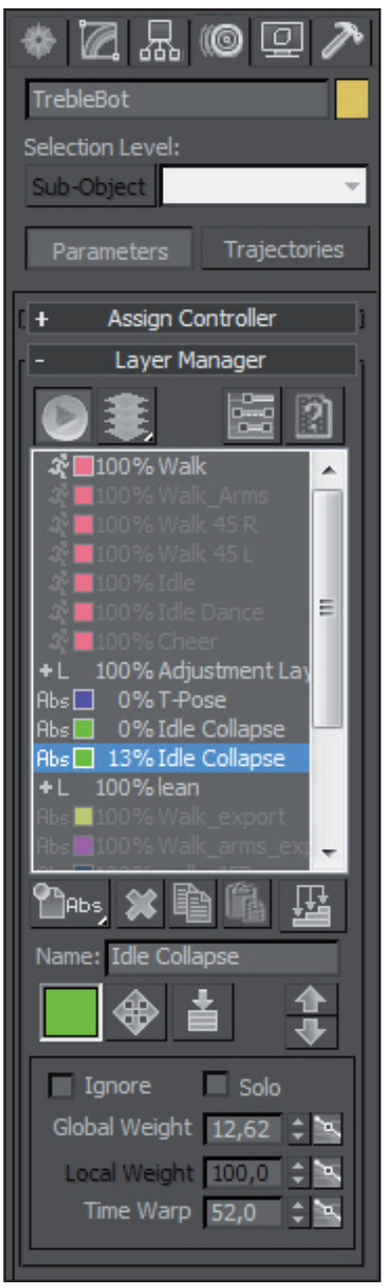


Abbildung 60: 3ds Max - Layered Animation

9.1.2 Export und Transfer

Für die Animation individueller Komponenten und die Erweiterung bestehender Animationen steht mit Autodesk MotionBuilder eine hilfreiche Softwarelösung zur Verfügung. Ist der jeweilige Animation Cycle zur Zufriedenheit vollendet, wird dieser wiederum mit vorgestellter T-Pose geplottet und als .FBX File exportiert. Aufgrund der Tatsache, dass das CAT Rig den gleichen Aufbau und dieselbe Namensgebung wie das Maya Rig hat, kann die CAT Animation in MotionBuilder problemlos und exakt, unter Verwendung des bereits erstellten Naming Templates, auf das MAYA Rig transferiert werden. In MotionBuilder stehen dann wieder alle Manipulationsmöglichkeiten offen, um die Animation zu verändern, zu erweitern und mit anderen zu kombinieren. Eine genauere Erläuterung des Workflows folgt in Kapitel 9.3.3.1 CAT Daten (S. 148).

9.2 Motion Capturing

„Motion Capture is the process of recording a live motion event and translating it into usable mathematical terms by tracking a number of key points in space over time and combining them to obtain a single three-dimensional representation of the performance. In brief, it is the technology that enables the process of translating a live performance into a digital performance.“ (Menache 2000, S.1)

Der Begriff „Motion Capture“ beschreibt einen Vorgang, bei welchem spezifische Bewegungen, bzw. gesamte Bewegungsabläufe einer realen Person auf einen virtuellen Character übertragen werden. Die Person, die im Motion Capture Prozess Bewegungen ausübt, wird als Talent bezeichnet. Diverse Bewegungsaktionen werden dabei erfasst, gespeichert und in weiterer Folge auch wiedergegeben. Live-Darbietungen können auf diese Weise in den virtuellen Raum transferiert und dort festgehalten werden.

„The development of modern day mocap technology has been led by the medical science, army and computer generated imagery (CGI) field where it is used for a wide variety of purposes.” (Kitagawa et al. 2008, S.2)

Das Einsatzgebiet dieser Technik ist sehr vielseitig. Sowohl in der Film- wie auch in der Computerspiele-Branche wird diesem speziellen Verfahren eine besondere Bedeutung zugeschrieben. Dabei handelt es sich jedoch weniger um eine Automation der Animation, als um das Erzielen eines bestimmten Effekts, der sich durch die Natürlichkeit der Bewegung ergibt. Mit Hilfe der erfassten Daten können dieselben Bewegungsabläufe auf verschiedene Charaktere oder Objekte ähnlicher Skelettstrukturen angewendet werden.

9.2.1 Grundlagen

Auf dem Markt existieren verschiedene Lösungen, die den Einsatz dieser Technik ermöglichen. Motion Capture Technologien werden primär in drei Gruppen unterteilt: optische, magnetische und mechanische Systeme. Eine kurze Auflistung der jeweiligen Vor- beziehungsweise Nachteile soll einen Einblick in diese Materie verschaffen.

9.2.1.1 Optische Systeme

Bei einem optischen Trackingverfahren werden die Positionen von (reflektierenden) Markern im dreidimensionalen Raum durch eine bestimmte Anzahl digitaler Kameras erfasst und in weiterer Folge mittels einer Softwarelösung analysiert. (vgl. Liverman 2004, S.8) Infrarot Systeme stellen in dieser Kategorie die am meisten ausgereifte Technik dar.

Vorteile	Nachteile
• Genauigkeit • Hohe Capture Rate • Simultanes Capturing von mehreren Subjekten • Verwendung hoher Anzahl an Marker möglich • Marker Konfiguration kann schnell geändert und angepasst werden • Capture Subjekte können sich frei bewegen • großes Capture Volumen	• aufwendiges Post-Processing notwendig • Rotationsdaten müssen von Positionsdaten errechnet werden • Marker können verdeckt werden (Occlusion) • Lichtsetup nötig • visuelles Feedback in Echtzeit sehr begrenzt möglich • teure Hardware

Tabelle 3: Vor- und Nachteile optischer Systeme (vgl. Kitagawa et al. 2008, S.10)

Verwendete Methode

Für die Fallstudie wurde eine auf synchronisierten Video Sequenzen basierende optische Motion Capture Technik verwendet. Über eine Softwarelösung ist es möglich dreidimensionale Daten anhand von Sets zweidimensionaler Bildern zu rekonstruieren (vgl. Kitagawa et al. 2008, S.18). Diese Technik repräsentiert den simpelsten Typus eines Motion Captue Systems (vgl. Kitagawa et al. 2008, S.47),

der jedoch ein aufwändiges Post Processing verlangt. Die Bewegung eines Körpers im dreidimensionalen Raum wird erfasst, indem Bewegungen anhand optischer Marker in verschiedenen Kamera-Perspektiven analysiert werden.

Marker, die durch den eigenen Körper oder durch andere Marker temporär überdeckt werden, stellen ein nicht zu unterschätzendes Hindernis im optischen Trackingverfahren dar (Occlusion). Hierfür ist die Positionierung im Capture Volumen ausschlaggebend. Das Talent sollte auf das Problem der Occlusion hingewiesen werden, jedoch darf dies in der Ausübung der gewünschten Aktion nicht hinderlich sein, da die Natürlichkeit der Bewegung darunter leiden würde. ”It is balancing good performances with avoiding occlusion.“ (Kitagawa et al. 2008, S. 38)

Resultat dieses optischen Trackingverfahrens ist eine so genannte Point Cloud, welche sich aus Punkten im Raum zusammensetzt, die Translationsdaten jedes einzelnen Markers beinhalten.

9.2.1.2 Magnetische Systeme

Die räumlichen Relationen von Sensoren zueinander werden anhand magnetischer Bemessungen ermittelt und an einen in der Nähe befindlichen Transmitter übergeben. Auch diese Sensoren werden an Gelenken befestigt, um die ausgeübte Bewegung bestmöglich wiederzugeben. (vgl. Liverman 2004, S.8)

Vorteile	Nachteile
• Position und Orientierung ohne Post Processing vorhanden • Echtzeit Playback ermöglicht Echtzeit Applikationen • Tracking Sensoren werden nicht von nicht metallischen Objekten verdeckt • Simultanes Capturing von mehreren Subjekten • billiger als optische Systeme	• Sensoren sind anfällig für magnetische und elektrische Interferenzen • Verkabelung und Batterien können Bewegungen einschränken • niedrigere Sampling Rate als optische Systeme • magnetische Daten neigen zu „Noise“ • Sensoren Konfigurationen sind schwer zu ändern • kleineres Capture Volumen

Tabelle 4: Vor- und Nachteile magnetischer Systeme (vgl. Kitagawa et al. 2008, S.11)

9.2.1.3 Mechanische Systeme

Diese Variante des Motion Capturing wird auch “Prosthetic Motion Capture” genannt, da mit Hilfe externer Strukturen gearbeitet wird, die an Extremitäten oder anderen Körperteilen angebracht werden. Verschiedene Encoder liefern Daten, die während einer Bewegung Aufschluss über Rotation und Position des jeweiligen Körperteils geben. Viele verschiedene Systeme und Typen von mechanischen Motion Capture Systeme existieren auf dem Markt. Waldos, spezielle Datenhandschuhe und elektromechanische Anzüge sind einige Beispiele, die in diese Gruppe fallen. (vgl. Liverman 2004, S.8)

Vorteile	Nachteile
• Echtzeit • relativ billig • keine Verdeckung • keine elektrischen oder magnetischen Interferenzen • portabel • großes Capture Volumen	• keine globale Translation • Bewegungseinschränkungen für das Subjekt • zerbrechlich • Fixe Konfiguration der Marker • niedrige Sampling Frequenz

Tabelle 5: Vor- und Nachteile mechanischer Systeme (vgl. Kitagawa et al. 2008, S.12)

9.2.2 Markersetup

"[A Marker is] a retro-reflective sphere or hemisphere that is attached to the object that you wish to capture and whose location is tracked by the cameras." (Liverman 2004, S.128)

Zunächst stellt sich die Frage, wie ein Marker für ein optisches System beschaffen sein muss. Verschiedene Kriterien wirken sich auf die Wahl der passenden Marker aus. Die Kameraspezifikationen sowie die Anzahl und Platzierung der Kameras entscheiden über den Einsatz bestimmter Markervarianten. (vgl. Liverman 2004, S.139)

Infrarotsysteme unterscheiden zwischen aktiven und passiven, bzw. Licht emittierenden und rein reflektierenden Markern. Für die Fallstudie wurde jedoch nicht mit Infrarot Kameras aufgezeichnet, sondern mit digitalen HDV Geräten. Die Größe der Marker wird über die Auflösung der jeweiligen Aufnahmesysteme, sowie über die Ausmaße des resultierenden Capture Volumens definiert (vgl. Kitagawa et al. 2008, S.8).

Bezüglich der Platzierung von Markern gibt es einige Punkte, die es zu beachten gilt. Für die richtige Platzierung der Marker ist es notwendig, die anatomischen Grundzüge des menschlichen Körpers zu kennen und zu beachten. Marker werden einerseits eingesetzt um Positionen von Gelenken zu definieren, andererseits liefern sie wertvolle Informationen, um Rotationen eben dieser Gelenke zu eruieren. „The basic goal of marker placement is to position them on the performer to best capture the motion desired and in such a way that allows the markers to be seen by the cameras without hindering the performer from executing the moves." (Liverman 2004, S.129)

Stellen, an denen sich beim Menschen Knochen direkt unter der Haut befinden, eignen sich besonders für die Platzierung von Markern, da diese durch wenige bis keine Muskelkontraktionen verfälscht werden. Dies trifft vor allem für Gelenke wie Ellenbogen und Knie zu. Bei anderen Stellen müssen Kompromisslösungen gefunden werden. Der Oberschenkel ist diesbezüglich ein gutes Beispiel.

Hier empfiehlt es sich eine Stelle am Muskelansatz zu wählen, die möglichst geringe Kontraktionsbewegungen aufweist. Aber auch lose sitzende Kleidung kann sich negativ auf das Resultat auswirken.

Bestimmte Marker werden in der Post Capture Phase gruppiert und trianguliert, um die Bewegung einer gesamten Körperregion zu erfassen. Dies erfolgt zum Beispiel beim Oberkörper oder beim Kopf. Durch die Gruppierung können auch temporär überdeckte Marker und dadurch resultierende fehlende Markerinformationen kompensiert werden, da die Position über die restlichen sichtbaren Marker rückberechnet werden kann.

Für die Handgelenke wurde ein spezielles Drahtgerüst erstellt, welches für eine größere Distanz zwischen den Markern der linken und rechten Seite sorgt. Dies ermöglicht eine überdeckungsfreiere und dadurch leichter zu trackende Handgelenksrotation (siehe Abbildung 61). Bei dieser Konstellation handelt es sich um einen speziellen Typ von Marker, einem so genannten virtuellen Marker. Ein virtueller Marker ist kein physisch vorhandenes Objekt, sondern entspricht dem Resultat einer mathematischen Berechnung, die Aufschluss über den wahren Rotationspunkt eines Gelenks gibt (vgl. Liverman 2004, S.134).

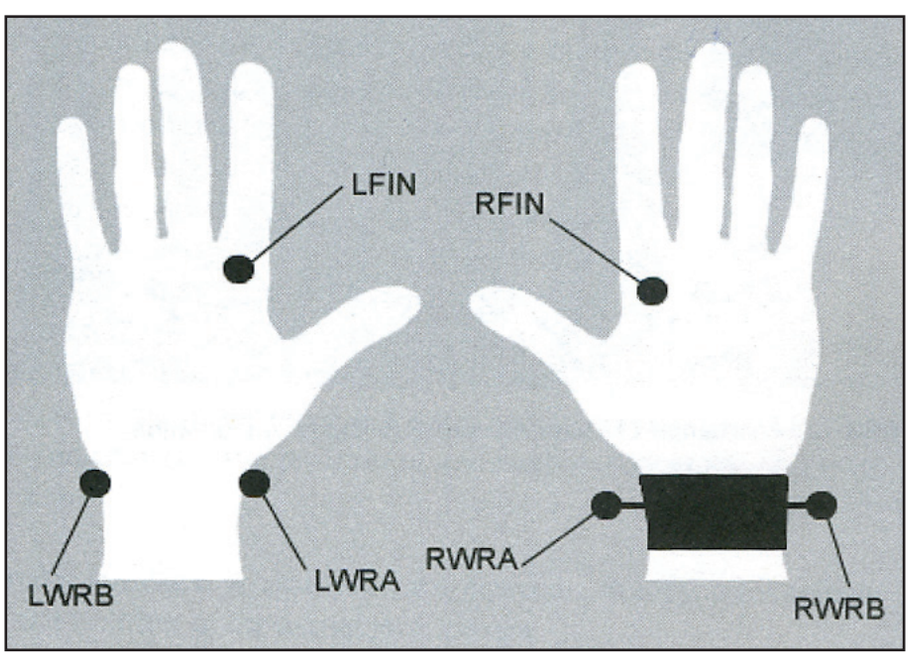


Abbildung 61: Marker Setup – Handgelenksmarker (zwei Varianten) (Liverman 2004, S.136)

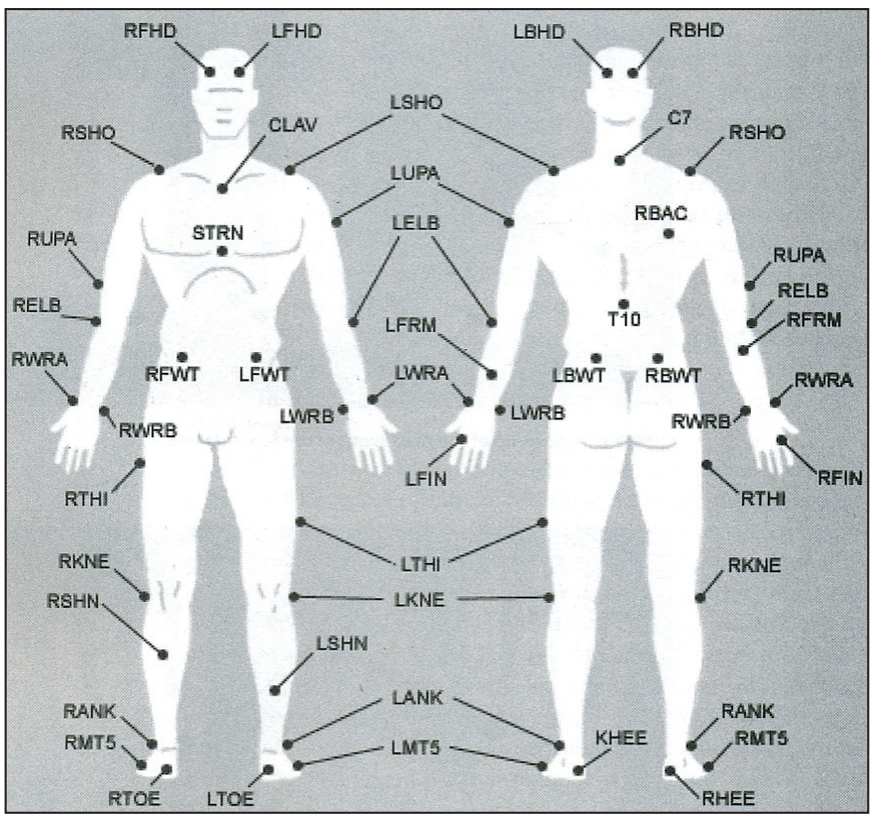


Abbildung 62: Marker Setup – Konfiguration für ganzen Körper (Liverman 2004, S.135)

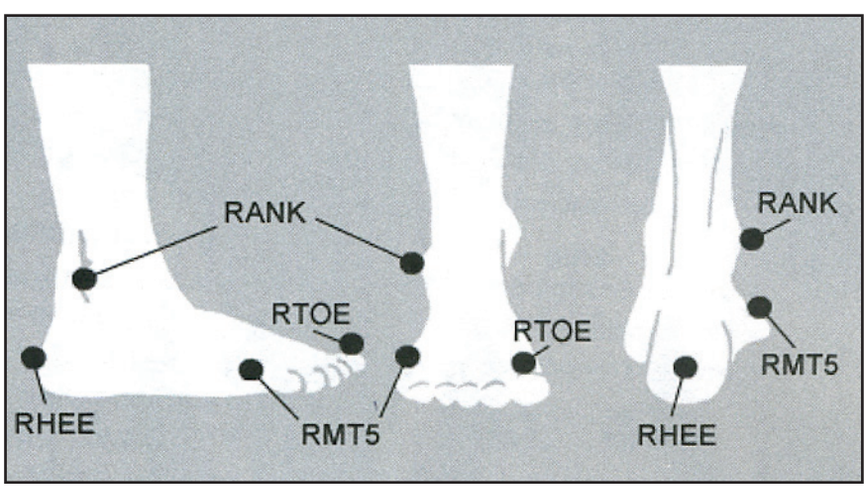


Abbildung 63: Marker Setup - Close-Up Platzierung der Fuß Marker (Liverman 2004, S.136)

Die Platzierung der Marker beschreibt einen kritischen Prozess, der ausschlaggebend ist, Daten zu erzeugen, welche die Bewegungen bestimmter Körperteile bestmöglich wiedergeben. Weiters müssen sie straff sitzen und dürfen während den Aktionen nicht verrutschen.

Abbildung 64: Talent in T-Pose

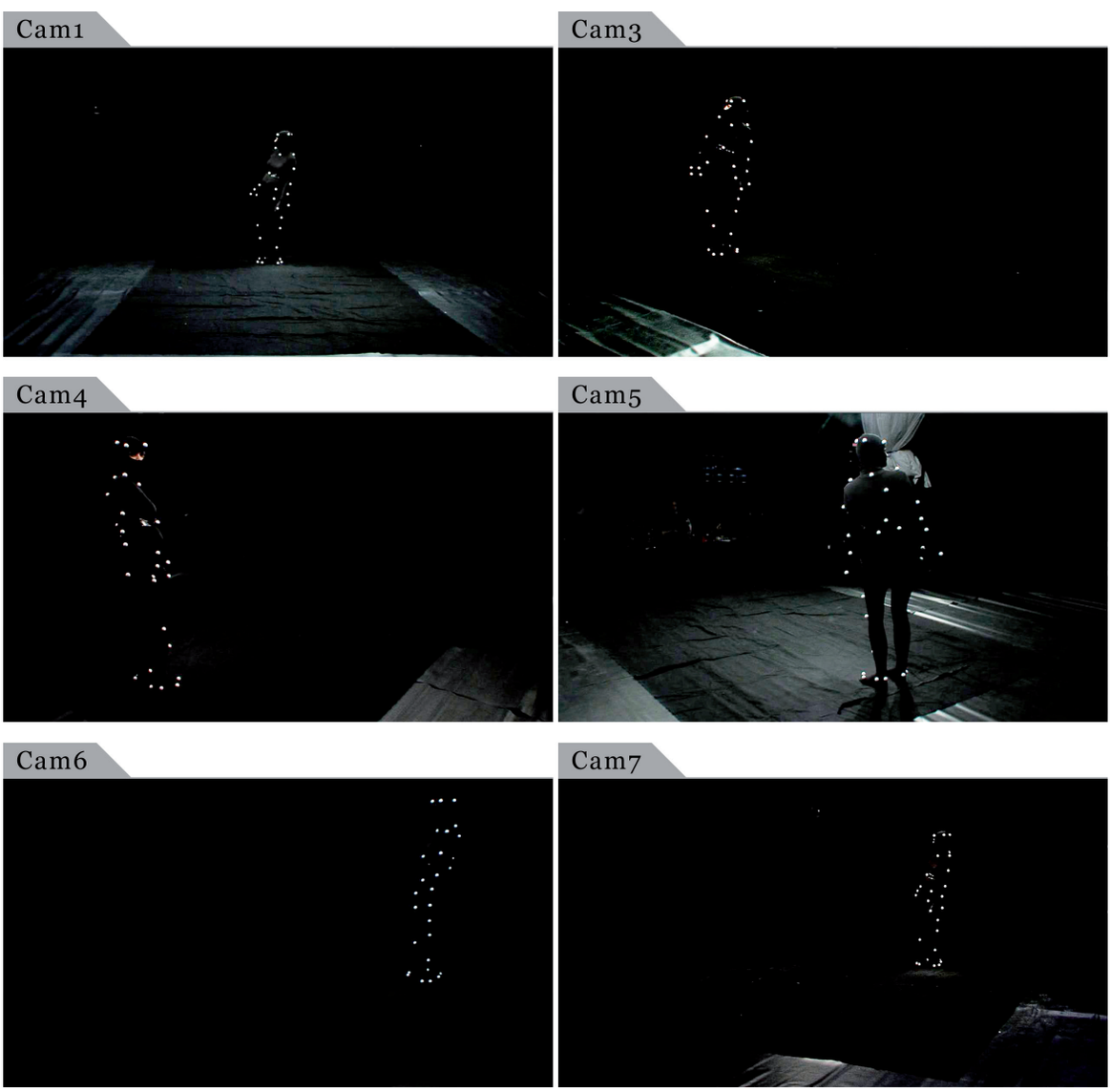


Abbildung 65: Talent in Capture Volume (6 Cameras)

9.2.3 Studioaufbau und Dreh

Die Aufnahmen erfolgten im Greenscreen-Studio der Fachhochschule St. Pölten, welches mit einer Grundfläche von 8,5 mal 6 Meter ideale Grundvoraussetzungen für ein Vorhaben dieser Art bietet. Um ein möglichst gutes Trackingergebnis zu erzielen, muss für ein gutes Kontrastverhältnis zwischen Markern, Performer und Hintergrund gesorgt werden. Alle Seitenwände sowie der Boden wurden aus diesem Grund mit schwarzem Molton beziehungsweise schwarzen Folien abgedeckt.

Gedreht wurde mit sechs JVC GY-HD200 Kameras in einer Auflösung von 1280 x 720 mit 50 Bildern in der Sekunde (720p50). Um den Kontrast noch weiter zu verstärken wurde der Schwarzwert jeder Kamera ebenfalls erhöht. Bei allen Kameras war die weitest mögliche Brennweite eingestellt, um ein für das Studio maximales Capture Volumen zu erzielen. Die vier Kameras, die sich am nächsten zum Talent befanden, wurden zusätzlich mit Weitwinkeladaptern versehen (Cam3 – Cam6). Die Formation der Aufstellung lässt sich der folgenden Abbildung entnehmen (siehe Abbildung 66, S. 136).

Bei der JVC GY-HD200 handelt es sich um eine 1/3“ 3 CCD Kamera. Für die spätere Kalibrierung der Software ist es unbedingt notwendig die richtige Bildsensorgröße anzugeben. Die Maße eines 1/3“ Chips betragen 3,6 x 4,8 Millimeter (vgl. Petrasch et al. 2003, S.101).

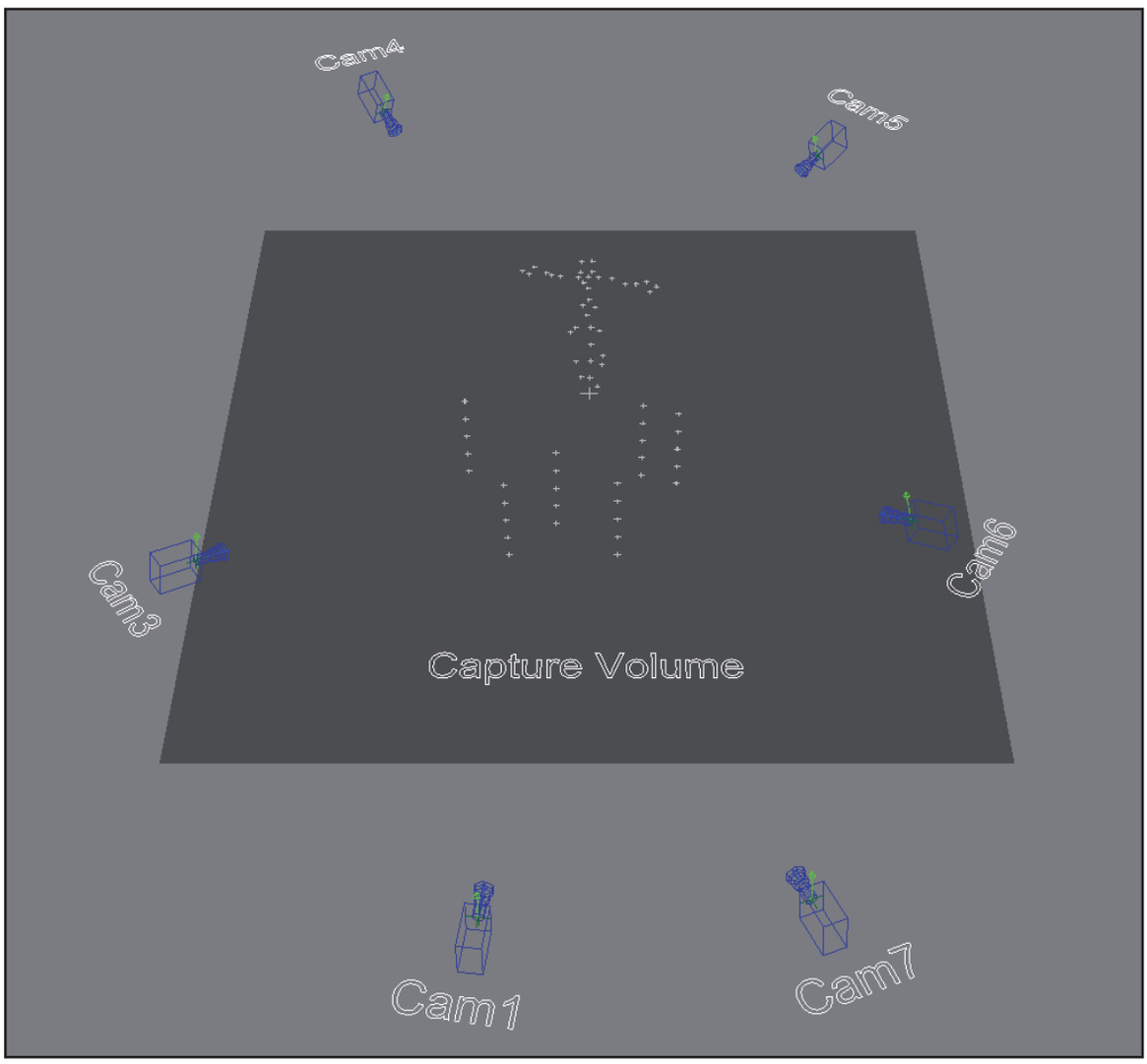


Abbildung 66: Studio Kamera Setup

Da kein eigener Timecode Generator vorhanden war, wurde die Synchronisierung mittels eines Blitzgerätes vorgenommen. Drei mal hintereinander wurde geblitzt, um sicherzustellen, dass in jeder Aufnahme mindestens ein Blitz zu sehen ist. Trotz fünfzig Bildern in der Sekunde wurden manche Blitze von einigen Kameras nicht erfasst. Anhand der vorhandenen Lichtimpulse war es in der Post Capture Phase jedoch möglich alle sechs Kameras miteinander zu synchronisieren.

Um die im Trackingprozess erstellte Point Cloud im weiteren Verlauf auf eine virtuelle Skelettstruktur transferieren zu können, ist es erforderlich, dass jede Darbietung im Capture Volumen mit einer vorangestellten T-Pose beginnt.

9.2.4 Kalibrierung

Das optische Tracking wurde mit Realviz Movimento durchgeführt. Für die Kalibrierung des Systems ist es notwendig, einige Vorkehrungen vor dem eigentlichen Tracking Verfahren zu treffen. Die Kalibrierung entspricht einer Einstellung und Errechnung der wichtigsten Eckdaten und Parameter der im Studioaufbau verwendeten Kameras. Dazu zählen interne Parameter wie Chipgröße, Brennweite und Framerate, wie auch externe Parameter, sprich Kameraposition und Orientierung. Die Bildauflösung und die Aspect Ratio der Aufnahmen werden durch die importierten Clips selbst definiert.

Die Kalibrierung erfordert vier bis sechs Trackingpunkte (sogenannte Blobs) in einer Linie mit jeweils gleichen Abständen zueinander, sowie sieben verschiedene Positionen im Raum. Mit Hilfe eines Besenstiels, doppelseitigem Klebebands, einem Blatt Papier, Gaffer Band und einem Hantelgewicht als Standbein wurden die Grundvoraussetzungen für einen erfolgreichen Kalibrierungsprozess geschaffen. Abbildung 67 zeigt den an unterschiedlichen Stellen im Capture Volumen platzierten Kalibrierungsstab sowie einen erfolgreich abgeschlossenen Kalibrierungsprozess.

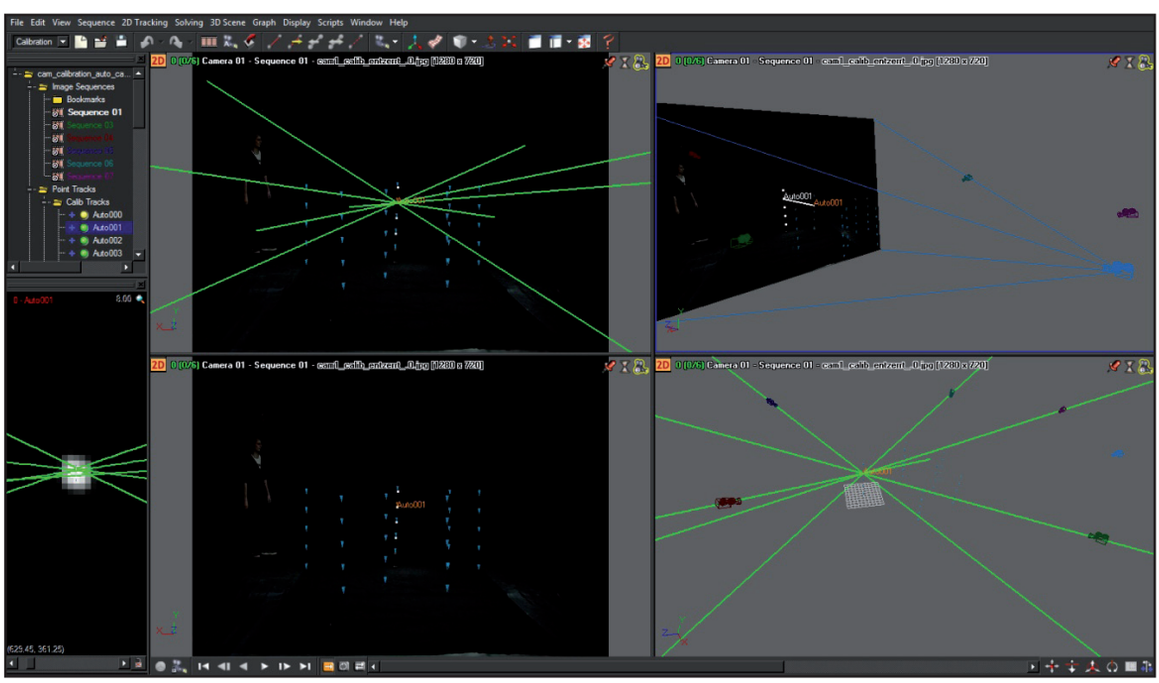


Abbildung 67: Movimento - Kalibrierungsprozess abgeschlossen

Um eine möglichst fehlerfreie Auto-Kalibrierung zu ermöglichen, gibt es einige Punkte, die es zu beachten gilt. Für jede der sechs Kameras wird eine Image Sequence mit den sieben Positionen des Kalibrierungsstabes erstellt. Eventuell hervorblitzende Bildanteile, die den Auto-Tracker irritieren und zu fehlerhaften Ergebnissen führen, können im Vorfeld wegretouchiert werden. Sind alle Vorkehrungen getroffen, kann eine erfolgreiche Auto-Kalibrierung durchgeführt werden. Anhand der erfassten Daten ist es der Software möglich, die genaue Position im Raum sowie die exakte Neigung jeder einzelnen Kamera zu eruieren. Die fertige Kalibrierung kann dann in einem weiteren Schritt als .RZML exportiert werden und für das Tracking von verschiedenen Motion Capture Aufnahmen, die in diesem Setup vollzogen worden sind, weiterverwendet werden.

Wie bereits erwähnt, ist es essentiell das Software Paket mit den wichtigsten Eckdaten der verwendeten Kameras zu versorgen. Bei der Initialisierung der Brennweite hat sich jedoch herausgestellt, dass es bei diesem Schritt besser ist, lediglich einen Richtwert anzugeben, als diese starr zu fixieren. Dies kann mittels der Funktion „constant initialized“ erfolgen. Obwohl bei allen Kameras die weiteste Brennweite eingestellt war, wurden doch Werte mit einer durchschnittlichen Differenz von 0,4065 mm errechnet. Diese Variante führte zu deutlich besseren Resultaten, als eine starre Fixierung der Brennweiten.

Camera	Eingestellte Brennweite	Errechnete Brennweite	Differenz
Cam 1	5,5 mm	5,827 mm	0,327 mm
Cam 3	4,5 mm	4,029 mm	0,269 mm
Cam 4	4,5 mm	4,769 mm	0,471 mm
Cam 5	4,5 mm	4,915 mm	0,415 mm
Cam 6	4,5 mm	4,958 mm	0,458 mm
Cam 7	5,5 mm	5.999 mm	0,499 mm

Tabelle 6: Kalibrierung - Brennweiten

9.2.5 Lensdistortion

Ein Punkt, der zu Problemen im optischen Trackingverfahren führen kann, ist die optische Verzerrung bedingt durch die Bauweise der verwendeten Linsen in Objektiven. „Linsen rufen Abbildungsfehler hervor, die um so stärker in Erscheinung treten, je weiter die Lichtstrahlen von der optischen Achse entfernt sind“ (Schmidt 2005, S. 334). Demnach treffen sich Randstrahlen, im Gegensatz zu achsnahen Strahlen, nicht genau im Brennpunkt, wodurch es zu Verzerrungen am Bildrand kommt.

Prime Optiken mit fixen Brennweiten weisen kaum Verzerrung auf, wobei viele Zoom Optiken vor allem in den weitwinkligen Bereichen sehr wohl zu Verzerrungen neigen. Die bekanntesten Formen durch Linsen verursachter optischer Verzerrungen sind bekannt als Pincushion Distortion, Barrel Distortion und Chromatic Aberration (vgl. Wheeler 2005, S.59).

Vor der Software Kalibrierung, wie auch vor dem eigentlichen Tracking Verfahren, ist es unbedingt notwendig, die Aufnahmen den Werten der Lens Distortion entsprechend zu entzerren. Mit Hilfe von PixelFarm PFTrack kann der Grad der Verzerrung eruiert werden und das Footage für die Kalibrierung sowie das Motion Capturing selbst entzerrt werden. Für die 1/3“ JVC Kameras ergaben sich folgende Werte:

- Ohne Weitwinkel: low order Rad: 0,0651172
- Mit Weitwinkel: low order Rad: 0,174805

Wird dieser Schritt nicht ausgeführt, kommt es zu sehr ungenauen Ergebnissen. Ein Fehler in der Kalibrierung kann alle Anstrengungen in dem darauffolgenden Trackingprozess zu Nichte machen.

9.2.6 Tracking

Ist die Kalibrierung erfolgt, kann mit dem Tracken der einzelnen Marker begonnen werden. Für diesen Zweck werden alle synchronisierten Aufnahmen der sechs Kameras mit Angabe der entsprechenden Framerate importiert, sowie auch das zuvor erstellte Kamera Kalibrierungsfile. Die in dem Footage sichtbare Bewegung muss immer mit einer T-Pose des Talents beginnen. (siehe Abbildung 68) Nur auf diese Weise kann sichergestellt werden, dass die aus dem Trackingprozess erstellte optische Markerinformation in weiterer Folge auf ein virtuelles Rig übertragen werden kann.

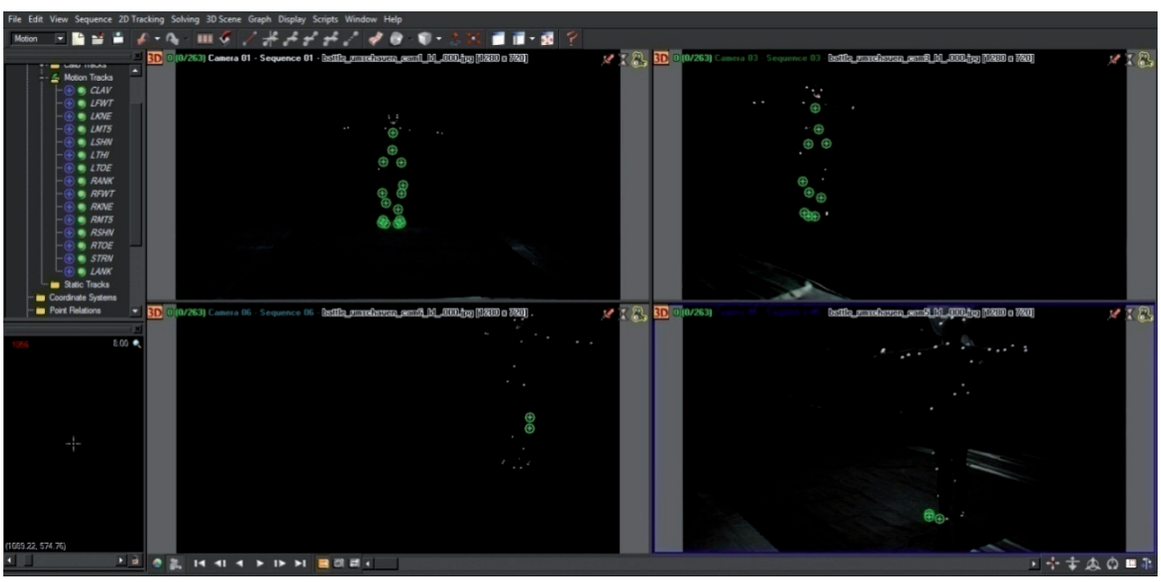


Abbildung 68: Movimento - Multi-Camera

Um die Bewegung zur Gänze in den digitalen Raum transferieren zu können, ist es essentiell jeden einzelnen Marker genau zu analysieren. Die exakte Position eines Markers im dreidimensionalen Raum kann jedoch nicht aus einem einzigen Viewport beziehungsweise aus einer einzigen Einstellung errechnet werden. Um dies zu bewerkstelligen, muss jeder Tracking Punkt von mindestens zwei Kameras erfasst werden (siehe Abbildung 69, S. 141). Die Grundvoraussetzungen eines Trackingverfahrens dieser Art sind dementsprechend zwei synchronisierte Kameras. Um das Capture Volumen zu maximieren und die Problematik der Occlusion zu minieren, ist jedoch eine höhere Anzahl an Kameras zu bevorzugen.

Das Trackingverfahren selbst ist ein langwieriger Prozess. Manche Marker sind jedoch leichter zu tracken als andere. Dies liegt hauptsächlich an der Problematik der Occlusion. Marker, die oft durch andere Marker oder Körperteile verdeckt werden, wirken sich negativ auf die Effizienz und Produktivität des Trackingverfahrens aus. Es ist wichtig, die Bewegung eines Markers vor dem Tracking genau zu analysieren und Bereiche zu definieren, in denen die Auto-Track-Funktion auf keinerlei Probleme stößt. Kurz vor oder nach einer Verdeckung ist jedoch aufgrund mangelnder Differenzierbarkeit zwischen Markern Tracking per Hand erforderlich. An diesen Stellen muss Frame für Frame die Position des Markers definiert werden.

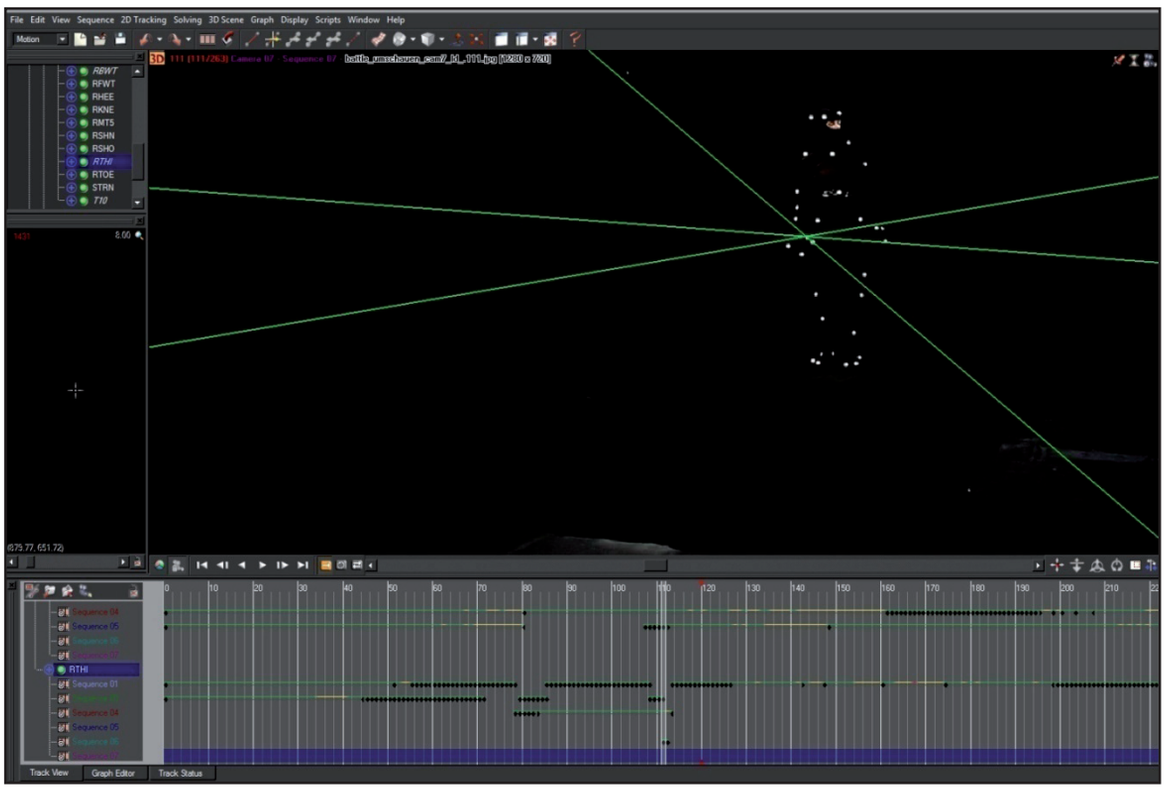


Abbildung 69: Movimento - Tracking (RTHI); epipolare Linien dreier Kameras

Ein simultanes Tracken von multiplen Markern ist prinzipiell möglich, jedoch erweist sich diese Vorgehensweise als unflexibel. Mit steigender Komplexität der Bewegungen wächst auch die Notwendigkeit der separaten Analyse jedes einzelnen Markers. Für jeden Trackingpunkt kann eine Pattern Zone und eine Search Zone definiert werden. Ein Track wird durch die Pixelmuster innerhalb der Pattern Zone definiert. Während dem Trackingprozess wird innerhalb der Search Zone nach dem definierten Pattern gesucht.

Wie in der folgenden Abbildung ersichtlich, ist der Marker des rechten Oberschenkels zu diesem Zeitpunkt bereits in der ersten Sequenz getrackt. In den weiteren Kameras ist nun die epipolare Linie ersichtlich, welche die Achse der Kamera wiederspiegelt, die den Marker bereits erfasst hat. Aufgrund der korrekten Kalibrierung des Systems schneidet die epipolare Linie den Marker des Oberschenkels exakt in der Mitte. Das Magnifier Window bietet eine praktische Hilfestellung in der Positionierung des Trackingpunktes. Durch die Eruierung der Position des Markers in einer zweiten Kamera kommt es zu einem Schnittpunkt zweier epipolarer Linien, wodurch die Position des Markers im Raum repräsentiert wird. (siehe Abbildung 70)

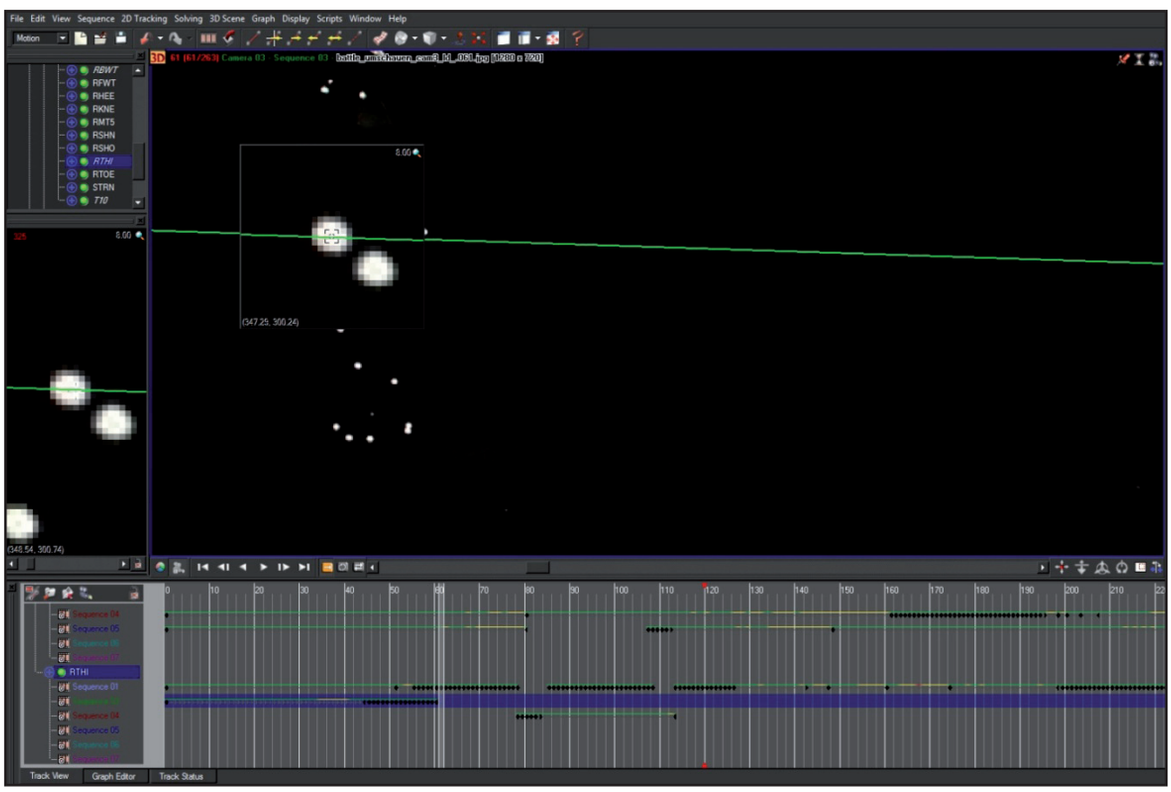


Abbildung 70: Movimento - Tracking (RTHI) CLOSE; Schnittpunkt mit epipolarer Linie

In der Timeline sind Keys für jeden einzelnen Trackingpunkt ersichtlich. Diese werden als schwarze Rauten beziehungsweise Richtungspfeile dargestellt. Mit Hilfe verschiedener Keyframearten können Start- und Endframes für einen Trackingprozess definiert werden. Intermediate Points oder Keys leiten wichtige Trackinginformationen sowohl für den Analyze Forward- als auch für den Analyze Backward Prozess weiter.

Wie sich gut an der Timeline erkennen lässt, gibt es Phasen der Bewegung, die ein automatisches Tracken über einen längeren Zeitraum gewähren (siehe Abbildung 70, S. 142). Die Farbe der über den Tracking Keys ersichtlichen Linie gibt Aufschluss über die Qualität der getrackten Information. Komplexere Stellen, die durch Marker oder Körperteile verursachte Occlusions aufweisen, erfordern jedoch einen höheren Zeitaufwand. Eine Durchsicht aller Kameraperspektiven ist erforderlich, um eine adäquate Einstellung zu finden, die das Tracking eines durch temporäre Occlusion verhinderten Markers ermöglicht. Trotz des Einsatzes von sechs Kameras ist dies nicht immer leicht zu bewerkstelligen. Vor allem die Rückenpartie des Talents, die lediglich von zwei Kameras erfasst wurde, stellte diesbezüglich eine besondere Herausforderung dar.

Resultat des Trackingvorgangs ist eine so genannte Point Cloud, die sich aus allen im dreidimensionalen Raum erfassten Markern zusammensetzt. Die Animation besteht somit aus reinen Translationsdaten, die im weiteren Verlauf in Rotationsdaten einer virtuellen Skelettstruktur umkonvertiert werden müssen.

9.2.7 Export und Transfer

Die Exportfunktion von Realviz Movimento unterstützt unterschiedliche Dateiformate, wie .FBX, .C3D und Motion Analysis .TRC, sowie programmspezifische Datentypen wie Maya .MA, 3ds Max .MS und Softimage .XSI.

Die Kommunikationsmöglichkeiten sind hierbei sehr vielseitig. Verschiedene Softwarepakete können angewendet werden, um Skeletthierarchien mit optischen Markerdaten zu verknüpfen. Um dies jedoch zu bewerkstelligen, muss ein spezielles Rig vorhanden sein, welches die nötigen Grundlagen für einen Transfer bietet. Das Rig muss mit bestimmten Control Objects versehen sein, welche die Translationsdaten der jeweiligen Markern übernehmen und an die entsprechenden Bones beziehungsweise Joints übergeben. Das spezielle Setup für einen Prozess dieser Art fällt jedoch sehr komplex aus und bietet wenige Möglichkeiten für ein notwendiges Cleaning der importierten Daten. (siehe Kapitel 9.3.3.4 Cleaning, S. 154)

Um diesen Vorgang zu erleichtern, bietet Autodesk MotionBuilder ein spezielles Feature, welches sich durch den Einsatz eines so genannten Actors auszeichnet. Eine genaue Erläuterung dieses Workflows wird in Kapitel 9.3.3.3 Optical Marker Daten (S. 152) genauer erläutert. Um jedoch alle Vorteile des Actors ausnutzen zu können, muss der Export als Motion Analysis .TRC erfolgen.

Motion Analysis .TRC
Das .TRC File wird von optischen Motion Capture Systemen generiert und enthält ausschließlich Translationsdaten (vgl. Menache 2000, S.132). Der Header des Files umfasst globale Informationen wie die Pfadangabe, die Data- beziehungsweise Framerate, die Anzahl aller Frames, sowie die verwendete Unit. Die Animationsdaten für jeden einzelnen Marker liegen framespezifisch als absolute Translationswerte für jede Achse des Raumes vor.

9.3 Verwertung der Animationsdaten

In diesem Kapitel wird näher auf den Animationsprozess als solchen eingegangen. Verschiedene Animationsvarianten sowie die Verwertung von Motion Capture Daten und optischen Markerdaten bilden die Kernelemente dieses Abschnitts der Arbeit.

Die in dem Kapitel Character Setup näher erläuterten Rigs sind die Grundlage für die folgende Erstellung und Aufbereitung von Animationsdaten, die mit Hilfe von Autodesk MotionBuilder vollzogen werden.

9.3.1 Einführung in MotionBuilder

Autodesk MotionBuilder hat sich als hilfreiches Softwarepaket bezüglich Character Animation beziehungsweise der Aufbereitung und Verwertung von optischen Marker Daten und Motion Capture Files erwiesen. Dabei handelt es sich um ein interaktives Environment, welches unterschiedliche Möglichkeiten für die Erstellung, Bearbeitung und Wiedergabe von komplexen Character Animationen bietet. Non Lineare Keyframe Animation, Cleaning und Transfer von optischen Marker Daten oder importierten Motion Capture Files (.BVH) und Arbeiten mit Animation Layers zählen zu den Kerngebieten von Autodesk MotionBuilder. Fertige, mit Rig und Skin versehene Characters, werden als .FBX geöffnet und können in weiterer Folge auf unterschiedliche Weise animiert werden. In den folgenden Kapiteln werden verschiedene Wege analysiert, Animationssequenzen für den Gebrauch in der Massensimulationssoftware Massive Prime zu erstellen. Damit mit Hilfe von MotionBuilder Motion Capture Daten sowie Animationen von andern Skeletthierarchien übernommen werden können, ist es erforderlich, die einzelnen Bones des importierten Characters zu definieren beziehungsweise bestimmten Körperteilen eines Bipeds zuzuweisen. In MotionBuilder wird für diesen Prozess die Terminologie „Characterize“ verwendet (siehe Kapitel 8.2.3 Name Conventions, S. 112). Auf diese Weise wird das Programm informiert, welche Teile des importierten Rigs bestimmten Joint-Hierarchien eines Bipeds entsprechen. Für den Characterazition Prozess ist es erforderlich, dass das Rig in T-Pose mit Blickrichtung in die positive Z-Achse ausgerichtet ist.

Nachdem dieser Schritt erfolgt ist, steht die importierte Figur als „Character“ zur Verfügung. Dies bietet wiederum weitere Möglichkeiten, um die importierte CG Figur mit weiteren Features zu versehen. Floor-Contacts können nun erstellt werden, welche die Auftrittsfläche des jeweiligen Fußes definieren (siehe Abbildung 71, S. 146). Dafür stehen Control Objects zur Verfügung, welche Aufschluss über die Ferse, den Ballen und auch die Zehen geben. Definierte Floor-Contacts verhindern, dass die Füße im Animationsprozess die Groundplane durchstoßen können, was eine erhebliche Erleichterung in der Keyframe Animation, wie auch im Cleaning-Prozess von Motion Capture Daten darstellt.

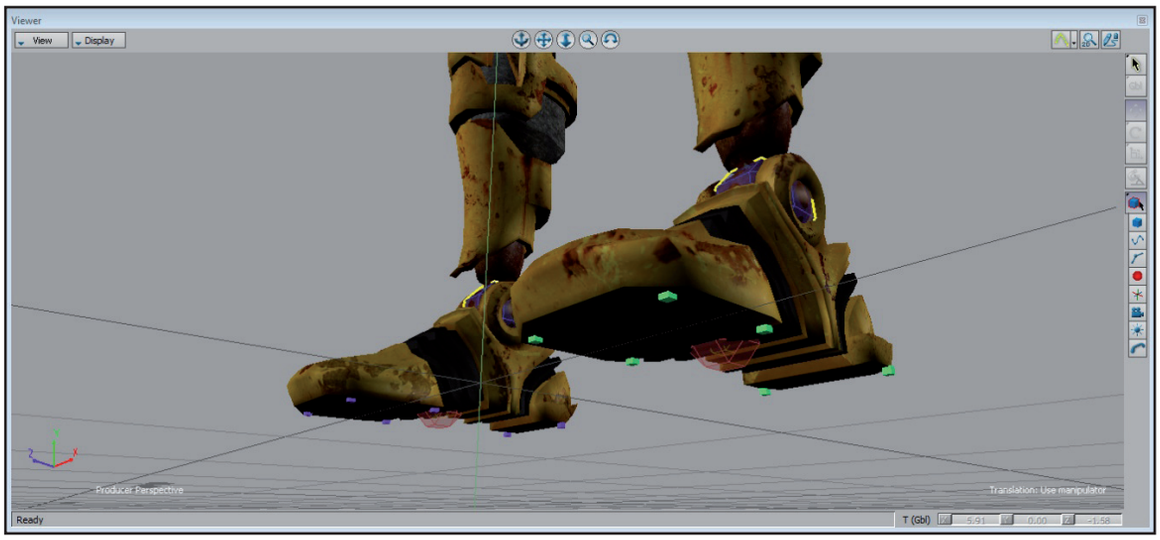


Abbildung 71: MotionBuilder - Floor Contacts

Ein Character in MotionBuilder kann Bewegungsanimationen von verschiedenen Quellen (Input Sources) annehmen. Die angeführten drei Varianten bilden die Grundelemente der folgenden Kapitel:

- Control-Rig (9.3.2 Keyframe Animationen mit Control Rigs, S. 146)
- Character (9.3.3.1 CAT Daten, S. 148, 9.3.3.2 MoCap Daten, S. 149)
- Actor (9.3.3.3 Optical Marker Daten, S. 152)

9.3.2 Keyframe Animationen mit Control Rigs

Nach erfolgtem Characterization Prozess ist es möglich mit Hilfe eines Klicks ein sogenanntes Control-Rig zu erstellen. Dabei handelt es sich um ein Full-Body-IK Rig, welches automatisiert erstellt wird und die wichtigsten Control Objects enthält, die für eine kontrollierte und effiziente Keyframe Animation nötig sind. (siehe Abbildung 72, S. 147)

Weitere nützliche Elemente stellen verschiedene Keyframe Modi dar. Full-Body, Body-Part und Selection stehen zur Verfügung. Diese definieren auf welche hierarchischen Über-Strukturen sich ein Effector in der Manipulation auswirken soll. Durch die Verwendung verschiedener Keyframe Typen fällt es dem Animator leichter, den in der Animation gewünschten Effekt zu erzielen.

Effector Pinning stellt ein weiteres nützliches Feature dar. Bestimmte Control Objects können sowohl bezüglich Translation wie auch Rotation fixiert werden. Diese Möglichkeit ist beispielsweise für die Erstellung von Walk-Cycles hilfreich. Hat ein Fuß Kontakt zum Boden aufgenommen, soll dieser während der Vorwärtsbewegung des restlichen Körpers nicht von der Stelle weichen. Lediglich ein Abrollen des Fußes über den Ballen soll dem Rig ermöglicht werden. Über das Pinning der Translation der jeweiligen Zehen kann genau dieser Effekt erzielt werden.

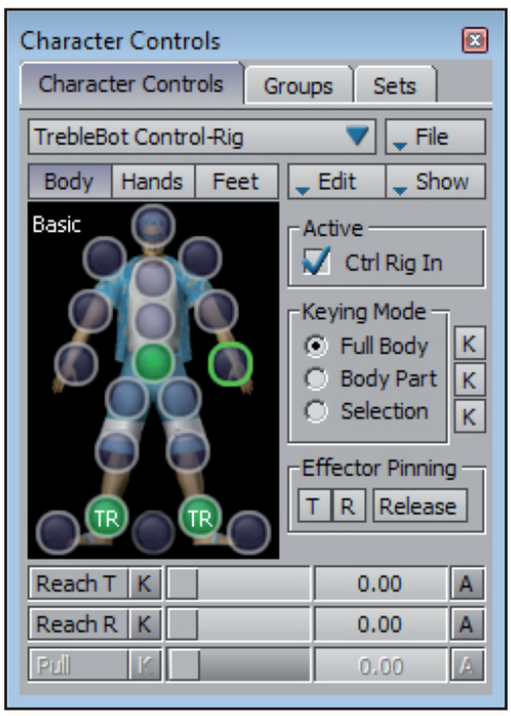


Abbildung 72: MotionBuilder - Character Controls des Control Rigs

Animation Layers stellen eine weitere Animationsmethode dar. Mit Hilfe von verschiedenen Ebenen können zusätzliche Parameter animiert werden, ohne die auf dem Base Layer zugrunde liegende Animation zu verändern. Diese Methode wird als „non destructive approach“ bezeichnet. Kopf- oder Armbewegungen können somit problemlos im Nachhinein hinzugefügt werden. Diese Methode ist ebenfalls in Bezug auf die Erstellung von Massenszenen hilfreich, um Varianz in den verschiedenen Bewegungsabläufen zu erzeugen.

In manchen Belangen ist es jedoch einfacher mit einem FK-Rig zu arbeiten. Die Möglichkeit zwischen FK-Skeleton und IK Control-Rig zu wechseln, stellt ein essentielles Feature von Autodesk MotionBuilder dar.

“Like FK, IK has its drawbacks. For many areas on a character, IK can make the simplest movements difficult to control. A character swinging his arm as he walks, for instance, can be difficult to animate with IK. Instead of just grabbing the shoulder and rotating it back and forth as you would with FK, IK would require you to translate the IK handle back and forth. Obtaining a nice arc can be a difficult task in this situation.” (Meade et al. 2007, S.292)

Ein Wechsel zwischen Control-Rig und Skeleton ermöglicht es, für unterschiedliche Problemstellungen im Animationsprozess einen adäquaten Manipulationszugang zu gewährleisten – „to get the best performance possible“ (Kitagawa et al. 2008, S.131)

9.3.3 Aufbereitung von importierten Animationsdaten

Der Transfer bereits erstellter Animationsdaten auf das in Maya erstellte Rig wird in diesem Abschnitt genauer behandelt. Dieser Schritt erfolgt in MotionBuilder und ist erforderlich, um eine spätere Übergabe der Bewegungsdaten an Massive zu gewährleisten, bei welcher Maya wieder als Verbindungselement und Kommunikationstool fungiert.

9.3.3.1 CAT Daten

Die mittels CAT erstellten Animationen werden jeweils als .FBX in MotionBuilder importiert. Dank der vorgestellten T-Pose und der eingehaltenen Name Convention kann dasselbe, bereits erstellte Name Template für den Characterization Prozess angewendet werden, welches auch bei dem Import des Maya Rigs selbst Verwendung findet. Dieser Prozess ermöglicht es, die CAT Animation auf den Maya Character zu transferieren. Um dies zu bewerkstelligen muss im Character Sheet des Maya Characters der CAT Motion Character als „driving source“ beziehungsweise als Input ausgewählt werden. Aufgrund der Tatsache, dass beide Rigs über dieselben Proportionen und Hierarchien verfügen, kann ein hundert Prozent exakter Transfer der Bewegungsdaten bewerkstelligt werden.

Anschließend kann die Animation auf das Skeleton des Maya Characters geplottet werden, wodurch die Aufbereitung für den Export grundlegend abgeschlossen wäre. Um jedoch die Animation für den Gebrauch in Massive weiter vorzubereiten, müssen innerhalb von MotionBuilder die Transferanimationen zwischen bestimmten Bewegungsstadien produziert werden, die einen flüssigen Übergang zwischen den einzelnen Animationszyklen in der Massensimulation ermöglichen. (siehe Kapitel 9.3.4 Motion Blending, S. 158)

9.3.3.2 MoCap Daten

Motion Capture Libraries bieten für die Erstellung von Bewegungsdaten eine gute Ergänzung zu den bereits erwähnten Methoden. Für die Fallstudie wurden ausschließlich .BVH Daten verwendet.

.BVH File

BVH steht für BioVision Hierarchical File Format und wurde ursprünglich von der Motion Capture Service Agentur BioVision entwickelt, die sich auf Sport Analysen und Animation spezialisiert hatte. Mittlerweile existiert die Firma nicht mehr, das File Format hat sich aber bis heute gehalten. (vgl. Kitagawa et al. 2008, S.182) Das File ist prinzipiell in zwei übergeordnete Sektionen unterteilt: Hierarchie und Motion. Der Hierarchie Part enthält alle nötigen Definitionen um eine Skelettstruktur in einem Animationsprogramm zu erstellen, wohingegen der Motion Part den eigentlichen Datenstrom der Animation enthält (vgl. Menache 2000, S.129).

Die Hierarchie der Gelenkstruktur setzt sich aus verschiedenen Blocks zusammen, die den jeweiligen Joints entsprechen und in der Darstellung durch geschwungene Klammern getrennt werden. Dabei werden Blocks wiederum in weitere Subblocks unterteilt. Die Position der Joints werden über relative Offset Translationswerte angegeben, die von den jeweils hierarchisch übergeordneten Joints ausgehen. Über den Offset Wert wird demnach auch die Länge des entsprechenden Bones definiert.

Der Motion Part ist in Spalten aufgebaut, die dem hierarchischen Aufbau der Skelettstruktur entsprechen. Im ersten Feld werden die Anzahl der Frames sowie die Dauer des einzelnen Frames in Sekunden angegeben. Für die Animation selbst werden jeweils frame- und jointspezifisch lediglich Rotationswerte gespeichert. Die Translation der kompletten Rig Hierarchie erfolgt ausschließlich über die Root beziehungsweise die Hips.

Motion Capture Daten, die gratis zur Verfügung stehen, sind jedoch selten für die direkte Anwendung zu gebrauchen. Ein aufwendiger Cleaning Prozess ist stets von Nöten, um Animationen in der gewünschten Qualität zu erhalten. MotionBuilder hat diesbezüglich eine Reihe interessanter Filter eingebaut, die diesen Prozess erleichtern. (siehe Kapitel 9.3.3.4 Cleaning, S. 154)

Die Verwendung von .BVH Libraries ist jedoch insofern von Vorteil, dass auf diese Weise Grundanimationen vorgegeben werden, die den gezielten Wünschen entsprechend adaptiert werden können. Vor allem für ungeübte Animatoren dienen diese Files als gute Hilfestellung in der Erstellung von realistischen, human wirkenden Bewegungsabläufen.

Bezüglich des Workflows werden .BVH Files ähnlich behandelt und verarbeitet wie die aus CAT exportierten .FBX Animationsfiles. Als erstes müssen auch .BVH Skelett Hierarchien den MotionBuilder Characterization Prozess durchlaufen, um einen Character zu kreieren, der als Input Source für den Maya Character dienen kann.

Für den Export erstellter Animationsdaten sowie für bestimmte Arbeiten im Cleaning Prozess müssen die gewünschten Bewegungsaktionen auf das Skeleton des Characters geplottet werden. Das bedeutet, dass jeder Joint des Rigs für jeden Frame der Animation entsprechende Rotationskeyframes erhält. Die Translationsparamter werden dabei ausschließlich an die Hüfte übergeben. Das Resultat ist ein exportfähiges, animiertes FK Rig. Natürlich kann eine, aus einer Motion Capture Aufnahme übernommene, Bewegung auch auf ein Control-Rig geplottet werden. Diese Möglichkeit ist in diesem Zusammenhang vor allem im Cleaning Prozess ein hilfreiches Feature. (siehe Kapitel 9.3.3.4 Cleaning, S. 154)

Um einen funktionierenden Transfer eines .BVH Motion Capture Files auf einen eigens erstellten Character durchführen zu können, muss dieser einer ähnlichen Physiognomie entsprechen. Diese Tatsache begründet auch den bereits erwähnten, zusätzlichen Spine-Bone des Bass Bots (siehe Abbildung 73). Dieser Bone übernimmt für den Character selbst keine spezielle Aufgabe, da dessen Torso einem starren Gebilde entspricht.

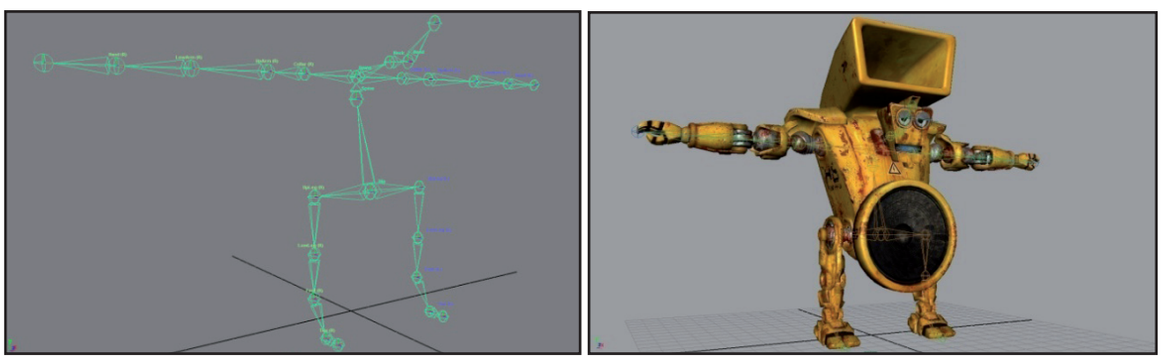


Abbildung 73: Maya - Bassbot: Rig und T-Pose

Die Präsenz einer „vorgetäuschten“ Wirbelsäule ermöglicht es, die Bewegung der Motion Capture Aufnahme korrekt zu übertragen. Jedoch übt der Character nun Aktionen aus, die seiner physischen Präsenz widersprechen. Aus diesem Grund müssen, nachdem der Plot Vorgang auf das Skeleton erfolgt ist, die überflüssigen Rotationskeys des Spine-Joints entfernt und auf Null gesetzt werden. Auf diesem Wege übernimmt der Körper des Bass Bots nun die gewünschte Bewegungsaktion, in der sich der Oberkörper entsprechend der Hüftbewegung verhält.

Wenn die Animation des .BVH Files erfolgreich auf den jeweiligen Maya Character übertagen ist, kann der Cleaning Prozess eingeleitet werden. (siehe Kapitel 9.3.3.4 Cleaning, S. 154)

9.3.3.3 Optical Marker Daten

Die durch das Tracking erstellte Point Cloud wird als .TRC in MotionBuilder importiert. Mehrere Formate stehen für diese Option zur Verfügung, jedoch bietet lediglich dieses Dateiformat die für den weiteren Verlauf nötigen Optionen, mit optischen Markern zu verfahren. Wie bereits erwähnt, erfolgt die Verwertung der optischen Marker Daten über das Actor Tool, welches die daraus gewonnene Information auf ein virtuelles Skelett überträgt. “The Actor is a collection of constraints and it is important to manipulate it to a degree that it matches as close to your marker data as possible. (Kitagawa et al. 2008, S.64)

Um einen Transfer zu gewährleisten, müssen sowohl der Actor als auch die Marker in T-Pose vorliegen. Die verschiedenen Körperteile des Actors müssen im Anschluss so adaptiert werden, dass sie der importierten Point Cloud bestmöglich entsprechen (siehe Abbildung 74). Dabei gilt es zu beachten, dass die jeweiligen Gelenke dem anatomischen Aufbau eines menschlichen Körpers entsprechend rotiert und angepasst werden müssen.

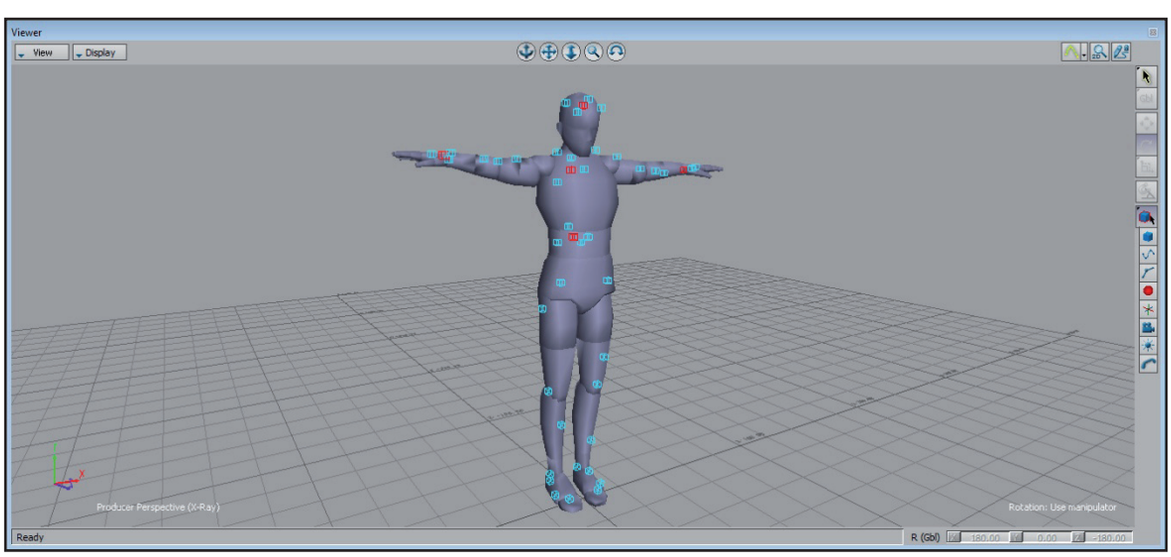


Abbildung 74: MotionBuilder - Abgleich Markerdaten und Actor

Nachdem die Proportionen des Actors an die importierten Marker angeglichen worden sind, erfolgt die Zuweisung der Marker zu den passenden Körperteilen. Dieser Schritt erfolgt in den Actor Settings (siehe Abbildung 75, S. 153).

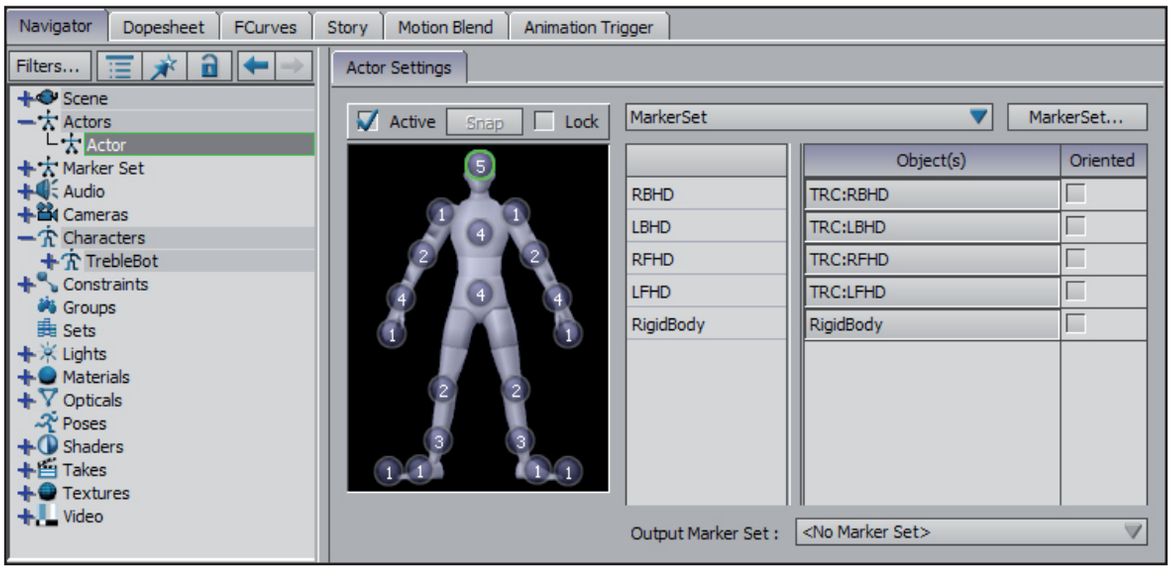


Abbildung 75: MotionBuilder - Zuweisung Markerdaten und Actor

Sobald alle Körperregionen des Actors den entsprechenden Markern zugewiesen sind, kann der Actor aktiviert werden. Dieser übernimmt dann die Bewegungen der im Motion Tracking Prozess generierten Point Cloud.

Der Actor kann dann in weiterer Folge die Bewegungen an andere Skeletthierarchien, die bereits einen Characterization Prozess durchlaufen haben, übertragen werden. „The Actor and the Character can be considered as a pair of mediators whose handshake has to occur in order for information exchange to happen. One mediator works on the marker data side and the other one on the skeletal data side. When the handshake between the two mediators is performed properly, the information is successfully transferred from the marker data to the skeleton.“ (Kitagawa et al. 2008, S.63)

Marker, die Scharniergelenke definieren, wie Knie und Ellenbogen, müssen beim Zuweisungsprozess mit dem Zusatzparameter „oriented“ versehen werden, um unwiderrufliche und unerwünschte Verdrehungen eben dieser Gelenke zu vermeiden. Für die Kompensation schlecht analysierter Marker stehen weitere Optionen zur Verfügung, die im folgenden Kapitel, Cleaning, näher behandelt werden.

9.3.3.4 Cleaning

Data Cleaning von Motion Capture Daten kann auf verschiedenen Ebenen erfolgen. Einerseits können die Translationsdaten von getrackten, optischen Markern selbst gesäubert werden, andererseits können auf das gewünschte Rig übertragene Rotations- und Translationsdaten von allen ungewünschten Nebenerscheinungen bereinigt werden. Bei dem Import und Transfer von .BVH Daten kann jedoch ausschließlich Letzteres erfolgen. Bei der Verwendung von optischen Markern ist es jedoch im Normalfall der beste Weg, die Daten anfangs so gut wie möglich auf dem Marker Data Level zu säubern (vgl. Kitagawa et al. 2008, S. 49).

Erfolg das Cleaning für eine spezifische Kameraeinstellung, kann der Fokus auf die in der jeweiligen Szene sichtbaren Elemente des Characters gelegt werden. Für Massensimulationsanwendungen jedoch muss jeder Körperteil beachtet werden, damit die Bewegung universell einsetzbar ist. „If you are using a dynamic close up shot, then there is no need to clean data for the character's legs [...] If your motion will be used in a video game or other interactive application where the player / user has full control over the camera (i.e., your character can be seen from any camera position), then you have to clean everything." (Kitagawa et al. 2008, S.48f) Für diese Cleaning Prozesse ist es erforderlich, die Bewegung des Characters aus allen Winkeln und Positionen genau zu betrachtet, um mögliche Fehler in der Animation zu entdecken und falls nötig auszubessern.

Aufgrund etwaiger Ungenauigkeiten sowohl in der Kalibrierung des Systems sowie im Trackingvorgang selbst, weisen Motion Capture Aufnahmen durchaus kleinere Fehler auf. Diese zeichnen sich zum Großteil durch eine leichte Zitterbewegung (Jitter) oder plötzlich auftauchende Spitzen (Spikes) in der Datenkurve aus. MotionBuilder hat genau für diesen Zweck bestimmte Filter eingebaut, die auf mathematischen Grundlagen basieren und Ungenauigkeiten dieser Art durch Reduzierung von Low- und High-Frequency Noise kompensieren. Die Filter erkennen unnatürliche Änderungen der Data-Curves und passen diese an. Jedoch ist der Filtervorgang mit Vorsicht zu behandeln, da dieser eine Bewegung schnell in seiner Natürlichkeit entstellen kann. Die wichtigsten Filter für das Cleaning von Motion Capture Daten sind der Smoothing-Filter, sowie der Peak-Removing Filter.

Cleaning Optical Marker Data

Die Genauigkeit der aus dem Motion Tracking resultierenden optischen Marker Daten ist hauptsächlich abhängig von der Kalibrierung des Systems, sowie der Positionierung des Talents innerhalb des Capture Volumens (vgl. Kitagawa et al. 2008, S.47).

Bezüglich des Cleaning Prozesses von optischen Marker Daten nimmt die Bereinigung von Data Gaps einen großen Platz ein. Data Gaps beschreiben kürzere Zeitstrecken, in denen ein Marker vom optischen System nicht erfasst werden konnte. Mittels linearer oder Bezièr Interpolation können die fehlenden Informationen überbrückt werden. Dabei werden die Daten vor beziehungsweise nach dem Verlust der Information miteinander verknüpft. Die Bezièr Interpolation ist jedoch aufgrund der verfügbaren Tangenten wesentlich flexibler in der Erstellung der gewünschten Verbindungsinformation. Auch die per Default eingestellte Auto-Rigid-Body Funktion liefert gerade für kurze Data Gaps gute Lösungen (siehe Abbildung 76). „Widening the gap by deleting small amounts of data before and after the gap and then applying spline interpolation will give you a smoother curve (i.e., a better solution).“ (Kitagawa et al. 2008, S.53)

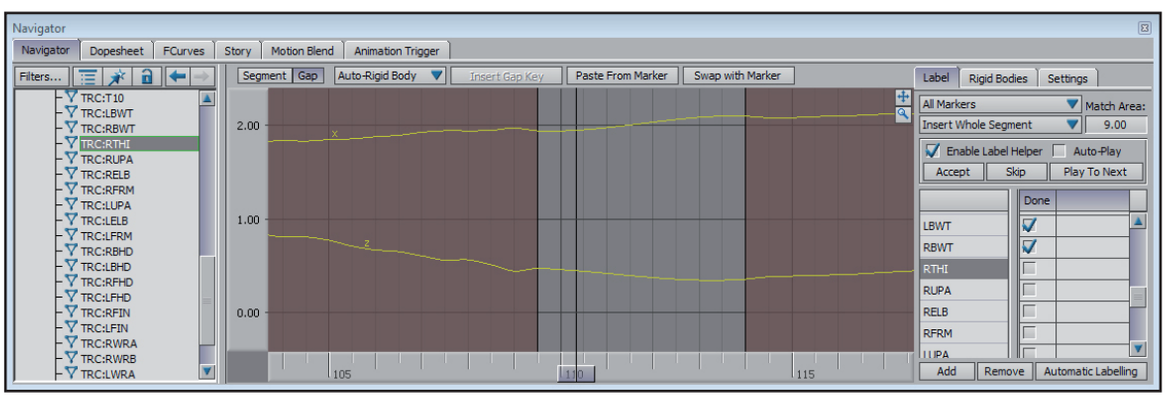


Abbildung 76: MotionBuilder - Data-Gaps; Auto-Rigid-Body

Eine weitere hilfreiche Methode Data Gaps zu überbrücken sind Rigid Bodys. Dieses Vorgehen ist vor allem bei fehlenden Informationen über einen längeren Zeitraum einzusetzen. Jedoch ist der Einsatz eines Rigid Bodies an bestimmte Bedingungen gebunden. Ein Rigid Body beschreibt ein Set aus mehreren Markern, die einen gesamten Körperteil definieren, der starr ist und in sich keinerlei Bewegung

aufweist. Wenn sich eine Rigid Body Sektion bewegt, bewegen sich die Marker innerhalb des Rigid Bodies mit, ohne die relative Position zueinander zu verlieren. Mindestens drei Marker werden benötigt, um einen Rigid Body zu definieren (siehe Abbildung 77). Zwei Marker alleine funktionieren weniger gut, da eine freie Rotation um die Verbindungsline der beiden Marker ermöglicht wird. Bei Data Gaps eines Markers in einem Rigid Body System werden die fehlenden Informationen durch die Position und Orientierung des Rigid Bodies kompensiert. (vgl. Kitagawa et al. 2008, S.56)

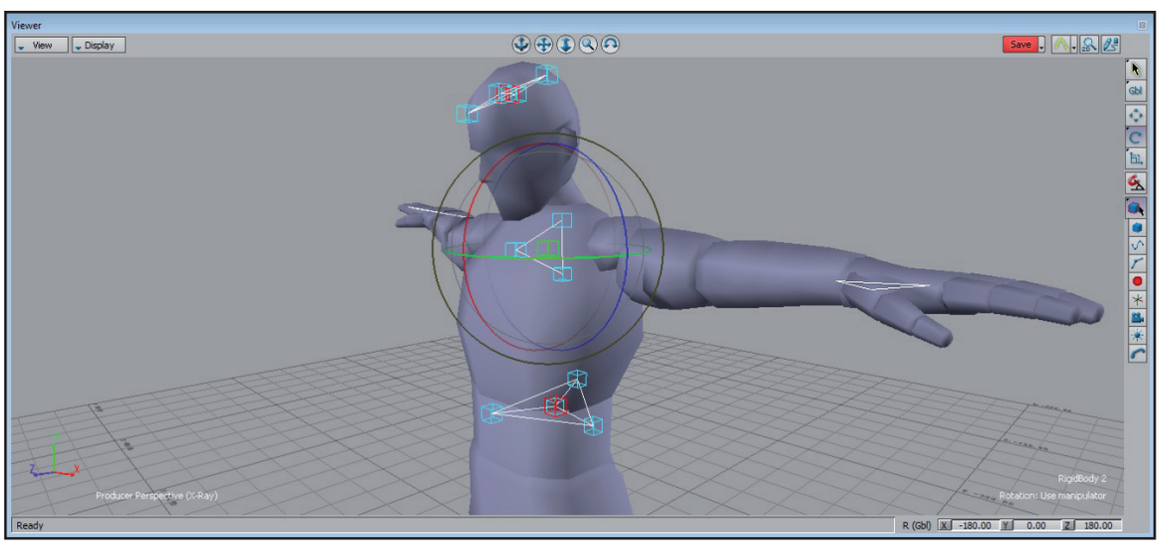


Abbildung 77: Motiobuilder - Definierung von Rigid Bodies

Cleaning Rig

Eines der Hauptprobleme bei der Verwendung von Motion Capture Daten stellen die Kontakte des Characters zum Boden dar. Vor allem Animationsdaten aus .BVH Libraries neigen dazu ein hohes Maß an „Sliding Foots“ aufzuweisen, oder den Boden nur spärlich zu berühren. Die Floor Contacts eines Control Rigs sind bezüglich dieser Problematik hilfreich. Über die Y-Translation der Hüfte kann der komplette Character näher gen Boden gerückt werden. Dank des Full-Body-IK-Systems in Zusammenhang mit den definierten Floor-Contacts, passen sich die Füße sowie der restliche Körper automatisch der Groundplane der Szene an. Durch diesen Vorgang verstärkt sich jedoch auch das Phänomen der über den Boden gleitenden Füße. Aufgrund der Tatsache, dass das Control-Rig nicht ausschließlich

über Rotationsparameter angesteuert wird, sondern auch über Translationen bestimmter Control Objects, ist es möglich, direkt auf die Translation der Füße Einfluss zu nehmen. Für die Zeiträume, in denen ein Fuß fix auf dem Boden bleiben soll, kann mit Hilfe des Control Objects allen Translationskeys ein fixer Wert zugewiesen werden, der ein Gleiten des Fußes verhindert.

Die zielgerichtete Verwendung des Control Rigs, wie auch des Skeleton sind für den Cleaning Prozess äußerst wichtig. Je nachdem welche Elemente des Rigs bearbeitet werden sollen, bieten beide Rig Varianten (FK / IK) verschiedene Vorteile. Sind zum Beispiel bei dem Ellenbogengelenk Verdrehungen vorhanden, die der Physiognomie des Characters widersprechen, können am Skeleton die Keyframes der entsprechenden Achsen entfernt werden, so dass lediglich die Achse übrig bleibt, über welche das Ellenbogengelenk rotieren kann. Auch Smoothing Filter lassen sich beim Oberkörper zum Beispiel besser auf das Skeleton anwenden, als auf das Control Rig. Das Control Rig wiederum zeigt seine Vorteile bei der Adaption der Beine und Füße. Ein reger Wechsel zwischen Skeleton und Control Rig ermöglicht es, die jeweiligen Vorteile von Forward- beziehungsweise Inverse Kinematics zu kombinieren und für den Cleaning Prozess auszunutzen.

Nachdem die importierten Motion Capture Aufnahmen erfolgreich auf das gewünschte Ziel Rig übertragen und gesäubert worden sind, müssen diese für die jeweilige weitere Verwendung in der Massensimulationsanwendung aufbereitet werden.

9.3.4 Motion Blending

Um animierte Characters massentauglich zu gestalten, müssen bestimmte Bewegungen miteinander kombinierbar und in vielen Fällen auch beliebig oft wiederholbar sein. Für die Verschmelzung von Bewegungen sowie für die Erstellung von Motion Loops wird Motion Blending verwendet. Das bedeutet, dass ein Übergang zwischen zwei Aktionen mittels Interpolation der entsprechenden Rotation von Joints geschaffen wird. Es gilt jedoch zu beachten, dass die Bewegungen zuvor auf dieselbe Skelettstruktur transferiert werden müssen, um eine Übereinstimmung bezüglich Hierachrie, Proportion, Name Conventions und Joint Orientation zu gewährleisten (vgl. Kitagawa et al. 2008, S.79).

Walk-Cycles, die prozedural mit Hilfe mit CAT erstellt werden, verfügen, aufgrund des Aufbaus und der Funktionsweise des CAT Motion Layers, über einen perfekten Loop. Auch die Übergangsanimationen können mit Hilfe von CAT erstellt werden. Jedoch eine Einbindung von weiteren Aktionen, wie zum Beispiel ein kurzes Stehenbleiben und Umschauen, kann mit Hilfe des Blend Tools vollzogen werden. Bei der Aufnahme von Motion Capture Daten ist darauf zu achten, dass das Talent bei der Ausübung eines Walks eine repetitive Bewegung ausführt. Dies ermöglicht nach erfolgtem Tracking und Transfer Prozess die Transformation in einen einwandfrei funktionierenden Loop.

Um aus einer .BVH Motion Capture Datei, die einen Walk enthält, einen Loop zu erstellen, muss schon etwas mehr Zeit aufgewendet werden. An erster Stelle muss die Animation einem Cleaning Prozess unterworfen werden, um beispielsweise die Bodenkontakte zu bereinigen und Jitter und Spikes zu beseitigen. Anschließend kann dieselbe Animation wiederverwendet werden, um den Walk zu verlängern. Der richtige Zeitpunkt, an dem der Verschmelzungsvorgang vorgenommen wird, ist für die Qualität des Loops ausschlaggebend. Eine genaue Analyse der Bewegung ist für diesen Prozess unausweichlich, denn nicht jeder Walk einer .BVH Library eignet sich für die Erstellung eines Motion Loops. Es müssen zwei entsprechende Stellen in der Bewegung gefunden werden, die über eine ähnliche Körperhaltung in Ausübung derselben Bewegungsaktion verfügen. Bei einem Walk-Cycle entspricht diese Stelle meistens dem Auftreten eines Fußes.

„A common blend point for walking or running is when a character has one foot on the floor and is shifting weight across the foot.“ (Kitagawa et al. 2008, S.82)

Ist der Blending Spot gefunden, können beide Bewegungsclips zeitlich nacheinander positioniert werden. Jedoch gilt zu beachten, dass sich der Character in der Zeit der ersten Bewegung bereits vorwärts bewegt hat. Im zweiten Bewegungsclip befindet sich der Character noch in der Anfangsposition. Um die Positionen der zwei zu verbindenden Rigs im Raum abzugleichen, existiert ein sogenanntes Ghost Rig. Andernfalls würde der Character nach Beendung des ersten Motion Clips zur Anfangsposition des zweiten Clips zurückgleiten. Abbildung 78 zeigt die Verschmelzung von vier Animationsclips.

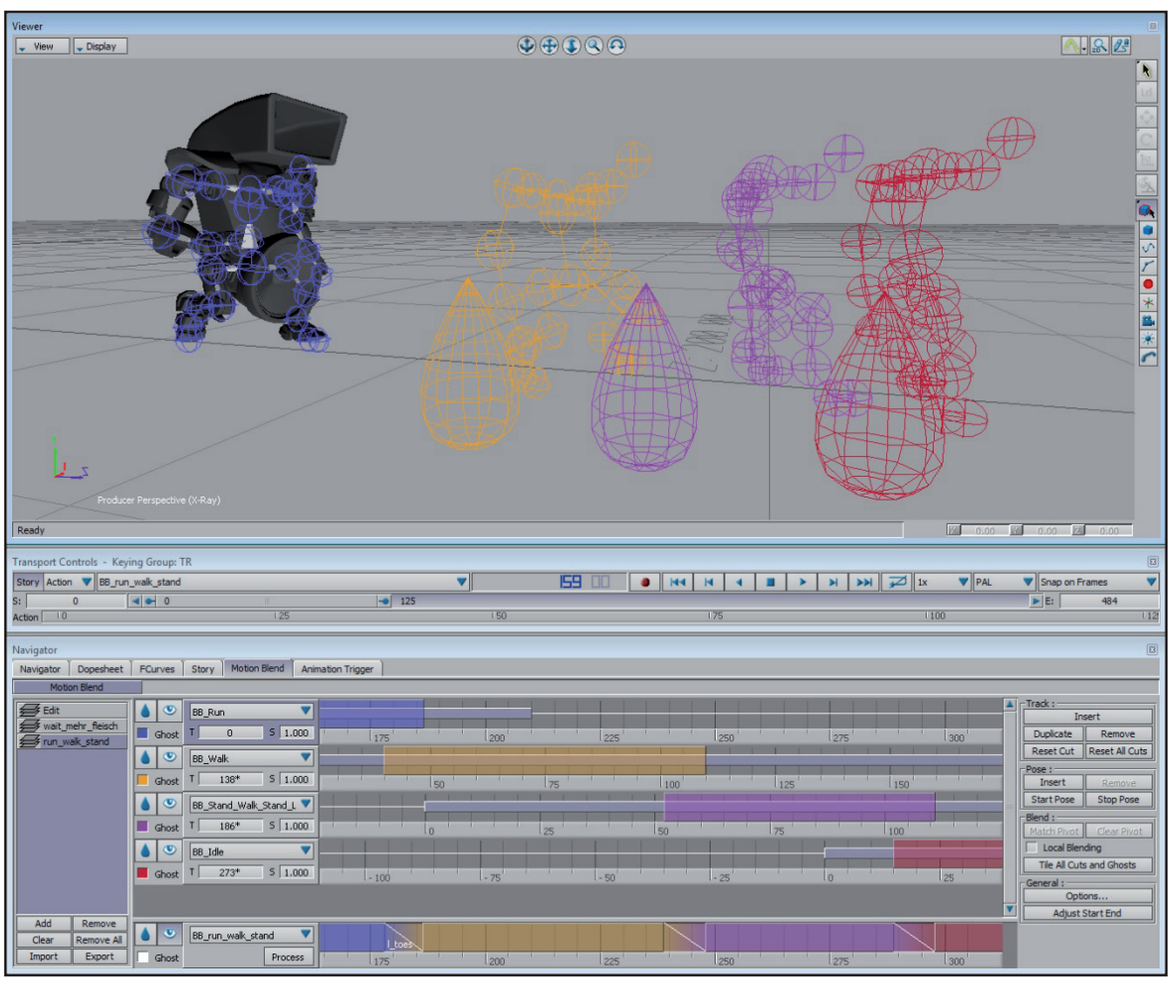


Abbildung 78: MotionBuilder - Motion Blend mit Ghost Rigs

Für den fehlerfreien Einsatz eines Ghost-Rigs ist die Anpassung der Joint Orientations im Setup Prozess von äußerster Wichtigkeit. Ist dieser Schritt nicht erfolgt, funktioniert der Import des Characters in MotionBuilder und die Erstellung eines Control Rigs einwandfrei, jedoch wird das Ghost Rig im Motion Blending Prozess nicht richtig dargestellt, was ein exaktes Justieren des Blending Points nahezu unmöglich macht (siehe Abbildung 79).

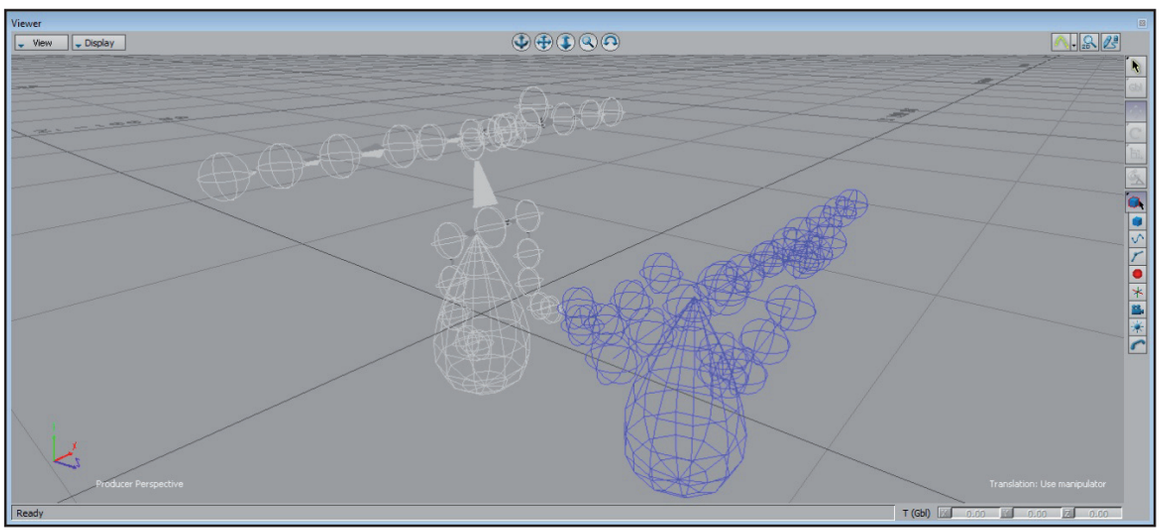


Abbildung 79: MotionBuilder - fehlerhaftes Ghost Rig

Um eine unnatürliche oder sprunghafte Bewegung des Körperteils zu vermeiden, bei welcher Bewegung A und Bewegung B exakt übereinander liegen, existiert die Funktion „Match Pivot“. Für diesen Zweck kann ein bestimmter Joint ausgewählt werden, der als eine starre Verbindungsbrücke dient, um die Bewegungen auf natürlichem Wege zu überblenden. In den meisten Fällen entspricht der auf dem Boden befindliche Fuß dem Pivot Point (siehe Abbildung 80).

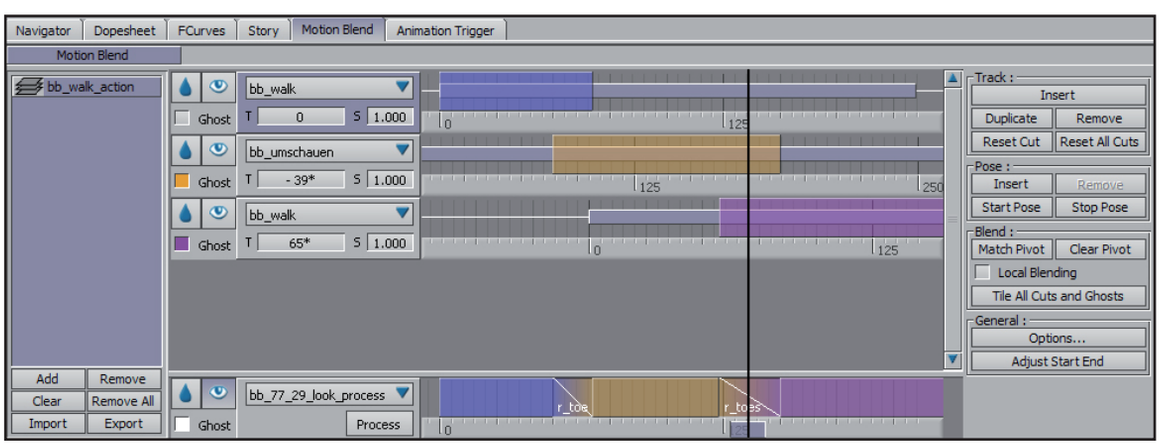


Abbildung 80: MotionBuilder - Match Pivot

Beliebig viele Bewegungen können auf diese Weise miteinander verknüpft werden. Kombinierte Aktionen können im Anschluss ebenfalls auf das Control Rig geplottet werden, um weitere Verfeinerungen im Bewegungsablauf vornehmen zu können. Dieser Prozess ist des Öfteren von Nöten, da es bei der Überblendung zu gleitenden Fußbewegungen oder anderen unnatürlichen Erscheinungen kommen kann. Diesem Problem kann aber Dank der Verwendung des Full-Body-IK-Systems Abhilfe geschafft werden.

„Having a way of animating on top of mocap data to add to the actor's performance will give you a perfect method for getting the timing and weight transfer of human motion. The ability to exaggerate motion will put much more life in the character." (Kitagawa et al. 2008, S.131)

9.3.5 Export und Transfer

Um die erstellten Bewegungen in Massive Prime und Autodesk Softimage verwenden zu können, müssen diese auf den anfangs in Maya mit Rig und Skin versehenen Character transferiert werden. Dieser Prozess kann nur erfolgen, wenn die Bewegung mit einer vorangestellten T-Pose versehen wird. Auch für diesen Zweck kann das Motion Blend Tool verwendet werden. Jedoch ist bei diesem Prozess die Einhaltung eines natürlichen Bewegungsablaufes außer Acht zu lassen, da die T-Pose lediglich einen referenziellen Nutzen verfolgt, um die Bewegungsaktion exakt übertragen zu können.

Die fertige Animation wird in MotionBuilder mit vorangestellter T-Pose auf das Skeleton des Characters geplottet und als .FBX exportiert. Das erstellte .FBX File wird in Maya in die Szene importiert, die den fertig mit Rig und Skin versehen Character enthält. Auf diese Weise führt der Maya Character die entsprechende Bewegung aus.

Anschließend kann der Export für Massive Prime erfolgen. Für diesen Zweck wird lediglich das Rig ausgewählt, welches als Maya ASCII .MA ausgegeben wird. Aufgrund des komplexeren Aufbaus des Treble Bots erfolgt für diesen Character ein weiterer Schritt. Um die Datenmenge für die Massensimulation möglichst

gering zu halten, wird für Massive auf das komplexe Schultersetup verzichtet. Alle Keyframes des Collar Joints werden entfernt und auf Null gesetzt. Auch alle weiteren Bones des Schulter Setups werden für den Massive Export gelöscht. Auf diese Weise entsteht ein vereinfachtes Rig, wodurch die Performance von Massive gesteigert werden kann.

Das Rig mit dem komplexen Schultersetup wird während des gesamten Animationsprozesses verwedet, damit jede Bewegung auch für Akteure im Vordergrund und Close-Up Einstellungen mit einer adäquaten Schulterbewegung eingesetzt werden kann.

Der Szenenaufbau für die filmische Inszenierung erfolgt mit Hilfe von Softimage. Der Export der handlungstragenden Figuren für die Vordergründe der Szenen erfolgt aus Maya über Autodesk Crosswalk im .XSI Format. Um einen reibungslosen Transfer des Treble Bots zu ermöglichen, ist jedoch ein Bake-Simulation Prozess für jeden Bone des kompletten Rigs unbedingt durchzuführen.

Ein weiteres Problem beim Export stellen die Expressions dar, welche die Rotation des Innenrings des Schultergelenks definieren. Die Expressions können entweder in Softimage mit Hilfe des Parameter Connection Editor neu gesetzt werden oder es muss in Maya for dem Export ein weiterer Bake-Simulation Prozess für den Innenring der Schulter selbst durchgeführt werden. Auch das Setup für Gesichtsausdrücke wird erst in Softimage nach erfolgtem Import aufgesetzt. Dieser Schritt ist jedoch unbedingt notwendig, um den Vordergrund-Akteuren mehr Charakter zu verleihen.

Die aus Autodesk Maya exportierten Rigs und die entsprechenden Animationsdaten bilden wesentliche Kernelemente für die Erstellung der Massensimulation, welche im folgenden Kapitel 10 Crowd Simulation (S. 163) detailliert analysiert werden.

10 Crowd Simulation

Dieser Abschnitt behandelt die Umsetzung der geplanten Massenszenen mit Hilfe der Massensimulationssoftware Massive Prime. Sämtliche Daten aus Arbeitsschritten, die in den vorigen Kapiteln beschrieben worden sind, fließen in diesem Teil der Arbeit zusammen.

10.1 Grundlagen der Crowd Simulation

Obwohl das Verhalten von Kollektiven schon seit dem neunzehnten Jahrhundert erforscht wird, ist die computergestützte Modellierung von komplexen Systemen ein sehr junges Gebiet der Forschung (vgl. Thalmann et al. 2007, S. 1).

Im Allgemeinen befasst sich die Technologie der Crowd Simulation mit der Simulation eines aus einzelnen Individuen zusammengesetzten Systems. Bei diesen Individuen, welche im Zusammenhang der Crowd Simulation als Agents bezeichnet werden, kann es sich um Menschen, Tiere, Fahrzeuge oder generalisierte Objekte handeln. Die Erschaffung von künstlicher Intelligenz der einzelnen Gruppenmitglieder stellt einen wesentlichen Bestandteil der Crowd Simulation Technologie dar. In den meisten Fällen reicht jedoch die Nachbildung künstlicher Intelligenz für eine realistische Abbildung von gruppendynamischen Abläufen nicht aus. Im Zuge der Crowd Simulation wird deshalb versucht das koordinierte Verhalten der einzelnen Individuen einer Gruppe in die Simulation mit einfließen zu lassen. Ziel ist eine möglichst realistische Reproduktion des Verhaltens von Akteuren in einer Masse.

Die Simulation von komplexen Systemen findet in vielen Sparten Anwendung. In der Architektur, der Verkehrsplanung und im Militärwesen hat die computergestützte Simulation von Massen Einzug gefunden. Seit einigen Jahren bedient man sich auch im Bereich von Film und Fernsehen dieser Technologie, um imposante und komplexe Massenszenerien zu veranschaulichen. Dieser sehr junge und zukunftsträchtige Bereich der Crowd Simulation wird in den folgenden Kapiteln der Arbeit aufgearbeitet und beleuchtet.

Die Filmindustrie stellt an die Umsetzung aufwendiger Massenszenen hohe ästhetische Anforderungen, welchen nur wenige Studios gerecht werden. Derzeit findet man am Markt etliche Softwarepakete zur computergestützten Crowd Simulation. Marktführer im professionellen Sektor ist die Software Massive Prime von Massive Software, welche von Stephen Regelous im Zuge der Produktion der „Herr der Ringe“-Trilogie entwickelt wurde.

10.2 Massive Prime

„Massive is the premier simulation and visualization solution system for generating and visualizing realistic crowd behaviors and autonomous agent driven animation for a variety of industries, including film, games, television, architecture, transportation, engineering, and robotics.“ (Massive Software 2010a)

10.2.1 Oberfläche

Um eine Grundlage für das Verständnis der nachfolgenden Kapitel zu schaffen, werden im folgenden Abschnitt die Basisfunktionen und die Oberfläche von Massive Prime kurz erklärt.

Die Oberfläche von Massive Prime gliedert sich im Wesentlichen in vier Arbeitsbereiche, die als Pages bezeichnet werden:

- **Body Page**
  Auf der Body Page wird die Grundstruktur eines Crowd Agents abgebildet. Die Geometrie und das Rig sowie Texturen und Shader eines Crowd Agents werden hier definiert.

- **Motion Page**
  Hier wird der Bewegungsablauf eines Crowd Agents in Form eines hierarchischen Motion Trees konzipiert. Der Motion Tree Workflow in Kombination mit dem integrierten Action Editor bildet die Basis für die Nutzung von Motion Capture Animationsdaten oder händisch erstellten Keyframe Animationen.

- **Brain Page**
  Der auf Nodes basierende Brain Editor bildet das Herz eines Crowd Agents. Hier wird die künstliche Intelligenz und die Logik für spezifische Verhaltensweisen der Agents erstellt.

- **Scene Page**
  Auf der Scene Page wird die Szene, bestehend aus Kameras, Lichtern, Terrain und Crowd Agents eingerichtet.

10.2.2 Workflow

Um einen vollständig funktionstüchtigen Crowd Agent in Massive zu konzipieren, bedarf es viel Aufwand und Zeit. In der Regel wird in der Konzeptionsphase nach einem Workflow gearbeitet, der schrittweise bis zur Komplettierung des Agents abgearbeitet wird (siehe Abbildung 81, S. 166). In der ersten Phase wird das Rig des Agents in Massive erstellt oder direkt aus Autodesk Maya importiert. Es folgt die Konzeption des Motion Trees. Als Resultat dieser Phase wird eine Liste aller benötigten Agent-Animationen exportiert und an den Animator weitergereicht. Im nächsten Schritt werden alle importierten Geometrien des Agents mit dem Rig verbunden und die Einstellungen für Rigid Body Dynamics justiert. Die vom Animator erstellten Animationen können in weiterer Folge importiert und in den Motion Tree integriert werden. Das Verhalten des Agents wird im folgenden Arbeitsschritt durch die Erstellung des Brain-Moduls definiert. Abschließend werden die Texturen des Agents importiert und Shader für die Renderphase konzipiert. Dieser Workflow wird für jeden geplanten Crowd Agent durchlaufen. Jeder Agent kann eine Reihe von Charaktervariationen abbilden. Geometrien, Texturen, Shader und die Skalierung von Charakteren sind bei der Platzierung von

Agents im Raum variabel und können durch einen einzigen Crowd Agent abgebildet werden. Unterscheiden sich die Agents jedoch in ihrer Grundstruktur, dem Rig, oder dem Verhalten, ist die Erstellung eines weiteren Massive Agents erforderlich.

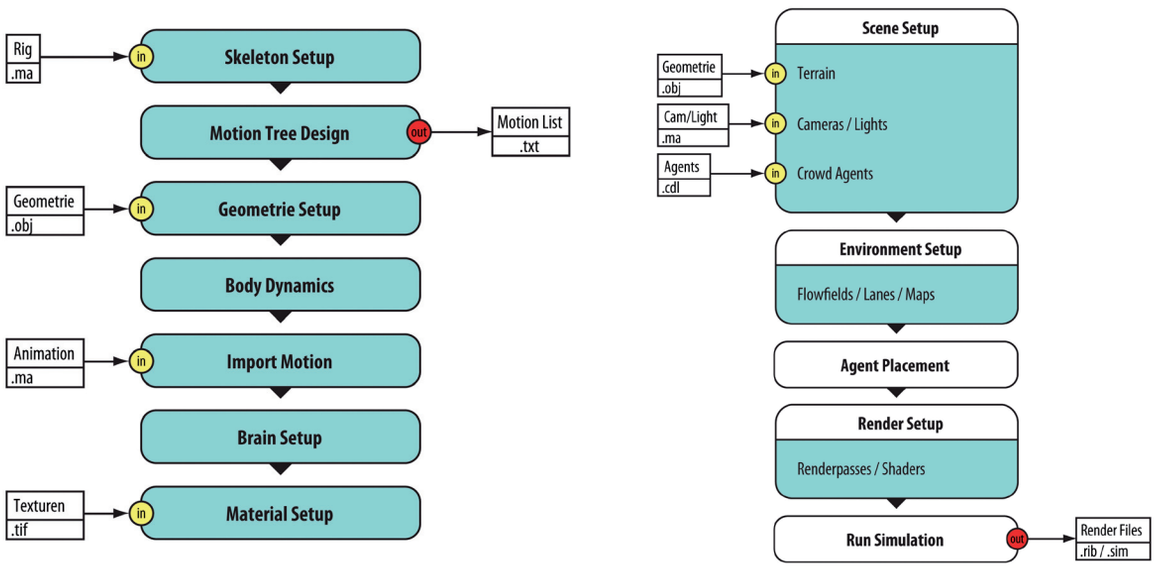


Abbildung 81: Agent Workflow / Massive Scene Workflow

Über den Import von Kameras, Lichtern und dem Terrain-Objekt wird in weiterer Folge eine Szenerie aufgebaut. Die erstellten Crowd Agents werden eingebunden, vervielfältigt und in der Szene platziert. Anschließend wird die Simulation gestartet und Parameter verfeinert, bis ein zufriedenstellendes Ergebnis erzielt wird. Die gesamte Simulation kann in weiterer Folge für die Renderphase exportiert werden.

10.3 Integration der Daten

Da Massive Prime in einer Maya Pipeline entwickelt wurde, ist für die Integration der Daten aus anderen 3D Softwarepaketen der Umweg über Maya in den meisten Fällen erforderlich. Der Import von Rigs und Animationsdaten als .XSI oder .FBX wird zwar von Massive unterstützt, führt aber oft zu mangelhaften Ergebnissen. Aus diesem Grund dient Maya als Zwischenglied für die Übertragung von Daten zwischen 3D Softwarelösungen und Massive. Rigs, Kameras, Lichter und Animationsdaten werden als Maya ASCII Dateien (.MA) importiert.

Animationsdaten sowie animierte Kameras und Lichter müssen vor dem Maya Export auf Keyframes mit absoluter Translation und Rotation reduziert werden. Geometrien sowohl für das Terrain als auch für Crowd Agents werden als polygonale Wavefront Objekte (.OBJ) geladen. Massive ist auch bezüglich der Interpretation der .OBJ Daten sehr stark an Maya orientiert. Ein direkter Import von Geometrien aus anderen Softwarepaketen wie Autodesk Softimage oder 3ds Max führt deshalb in den meisten Fällen zu Fehlermeldungen. Texturen werden als .TIF Daten (Taged Image File Format) geladen und in weiterer Folge einer Geometrie zugewiesen. Die vorhandenen Texturkoordinaten werden aus der zugehörigen .OBJ Datei extrahiert und auf die Textur angewendet.

Die Kombination der importierten Daten ermöglicht die detaillierte Rekonstruktion der Szenen aus Autodesk Softimage. Somit wird in der Compositing Phase des Projektes sichergestellt, dass sich das Endergebnis der Crowd Simulation problemlos mit Renderings aus den 3D Softwarepaketen verbinden lässt.

10.4 Terrain

Da sich die Crowd Agents in einer virtuellen Umgebung bewegen, müssen sie diese auch wahrnehmen und auf selbige reagieren können. In Massive wird diese Geometrie als Terrain bezeichnet. In dieser Fallstudie besteht das Terrain aus einer Stadt, die mit Hilfe der City Engine generiert wurde. Die Geometrie kann als .OBJ Datei in die Massive Szenerie eingebunden werden und verfügt über diverse Kanäle, die im Brain der Crowd Agents weiter verarbeitet werden können, um Bewegungsabläufe an das Terrain anzugleichen. Massive unterstützt bis zur Version 3.6 die Integration eines Terrain-Objekts. Ab der Version 4.0 sollen auch mehrere Geometrien in die Szene eingebunden werden können (Massive Software 2010b). Das Terrain verfügt über Kanäle, die bei der Programmierung der Crowd Agents herangezogen werden (siehe Kapitel 10.8.4.1 Terrain Adaption, S. 206).

10.4.1 Flow Fields

Mit Hilfe von Flow Fields werden vektorisierte Richtungs- und Flussinformationen für Crowd Agents bereitgestellt. Die Richtungsdaten werden in Form von Graustufenverläufen in den Alphakanal der Terrain Map gespeichert (vgl. Massive Manual 2008, flow_field.html). Diese Information kann anschließend im Brain der Agents verarbeitet werden, um den Bewegungsfluss der Massen zu kontrollieren. Besonders bei der Simulation von Gruppendynamiken humanoider Charaktere führt diese Methode zu realistischen Ergebnissen.

In Massive steht ein Tool für die Erstellung und Bearbeitung der Flow Fields bereit. Jedes Flow Field besteht aus einer Kurve mit beliebig vielen Ankerpunkten. Für jeden Punkt der Kurve kann die Breite des Feldes sowie der Krümmungswinkel und die Breite der Randbereiche bestimmt werden. Somit kann positionsabhängig erzielt werden, dass Agents in den Randbereichen des Flow Fields nach Innen oder Außen strömen und sich die Masse verdichtet respektive auflöst.

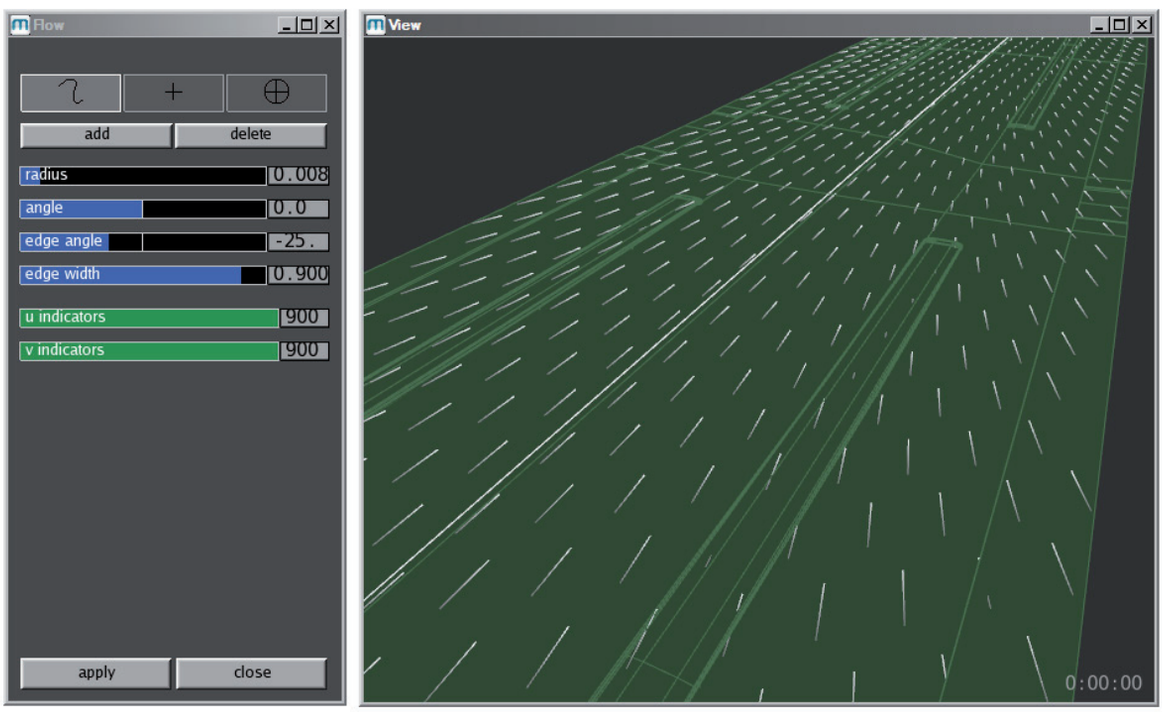


Abbildung 82: Flow Fields

10.4.2 Lanes

Mit Hilfe von Lanes werden Fahrbahnen für die Richtungsinformation von Agents definiert, welche in erster Linie der Kontrolle von motorisierten Agents dient. Komplexe Straßennetze und Kreuzungen können auf diese Weise abgebildet und simuliert werden. Im Fallbeispiel wurde diese Vorgehensweise herangezogen, um das Straßennetz im Zentrum der Stadt nachzubilden (siehe Abbildung 83). Die Anpassung der Lanes an die Geometrie des Terrains erfolgt hierbei automatisch und wird nach jedem Ladevorgang des Tarrains neu berechnet. Sogar für komplexe geometrische Anordnungen wie Unterführungen wird der Fluss der Lanes richtig berechnet. Die Erstellung der Lanes in Massive erfolgt, ähnlich wie bei den Flow Fields, über die Interpolation zwischen einzelnen Ankerpunkten und deren Tangenten. Der Import von Vektordaten oder Splines aus Maya wird derzeit noch nicht unterstützt und würde die Abbildung von komplexen Lane-Netzwerken deutlich vereinfachen.

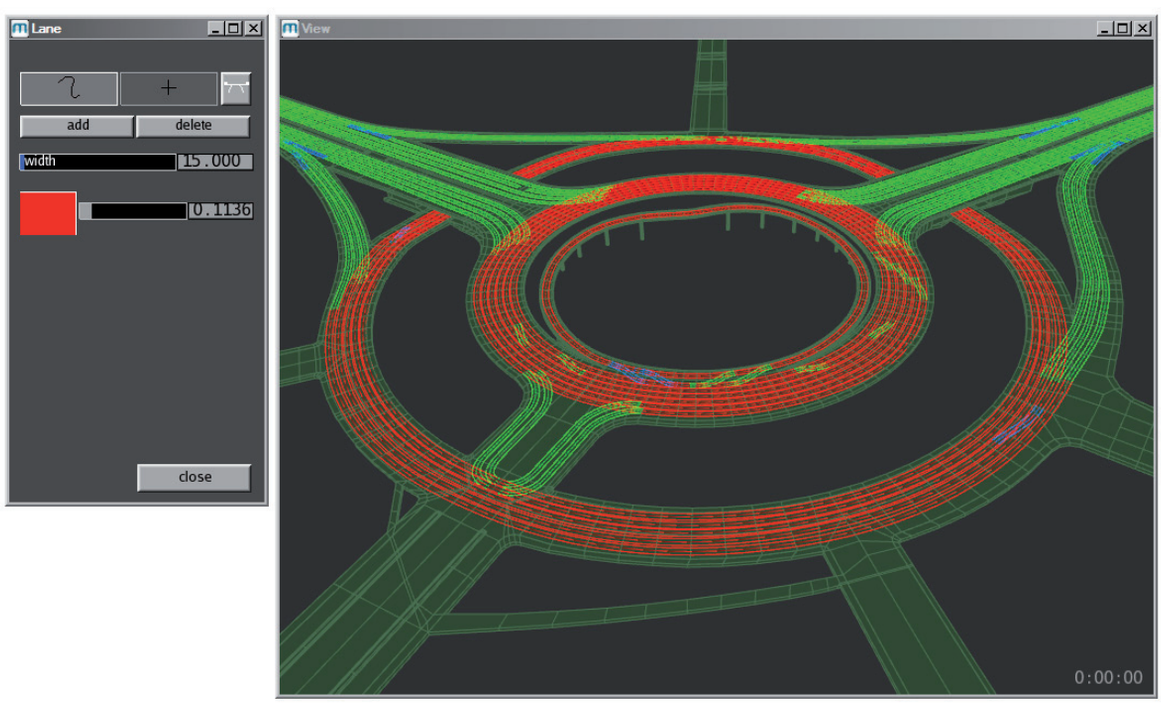


Abbildung 83: Lanes

10.4.3 Terrain-Textur

In Massive Prime kann die Textur der Terrain-Geometrie (siehe Kapitel 10.4 Terrain, S. 167) nicht nur für die Darstellung und den Render Output der Szenerie herangezogen werden. Die Farbinformation der Terrain-Textur dient auch zur Steuerung der Bewegungsabläufe und des Verhaltens der Crowd Agents in der Szene. Die Texture Map wird als .TIF Datei importiert und auf die Geometrie angewendet.

Jedem Crowd Agent, der sich auf dem Terrain bewegt steht in weiterer Folge die Farbinformation der Terrain-Textur zur Verfügung. Diese Variable wird in Rot-, Grün- und Blau-Werten getrennt an den Agent übergeben. Jedes Segment des Agents empfängt diese Farbinformation als Projektion des aktuellen Farbwertes der Terrain-Textur auf der y-Achse im Nullpunkt des Segments. Somit können die Farbwerte der Textur unter einzelnen Segmenten gezielt aufgegriffen, verglichen und interpretiert werden.

Zur gezielten Editierung der Textur bietet Massive die Möglichkeit Farbinformationen direkt auf die Geometrie zu zeichnen. Da das Terrain in den meisten Fällen nicht in das Rendering der Crowd Simulation mit einbezogen wird und nur als Input und Orientierung für die Massen dient, kann diese Map dazu verwendet werden, das Gehirn der Agents anzusprechen um auf bestimmte Farbinformationen zu reagieren. In der Fallstudie wird diese Methode angewendet, um Zonen zu markieren, welche von den Crowd Agents gemieden werden sollen. Grünflächen und Hindernisse wie beispielsweise Verkehrsinseln oder Schilder werden rot texturiert und nach Außen mit einem Verlauf von rot auf schwarz markiert. Bewegt sich ein Agent in einen dieser Bereiche, können im Gehirn Schritte eingeleitet werden, um diesem Verhalten gegenzusteuern.

10.5 Crowd Agents

Auf der Body Page in Massive Prime erfolgt die Erstellung und Bearbeitung der Grundstruktur von Crowd Agents. Der hierarchische Aufbau des Rigs, Geometrien, Materialien und Shader werden hier importiert oder konzipiert und als Agent-Datei (.CDL) gespeichert. Der Crowd Agent wird durch die hierarchische Vernetzung einzelner Nodes gebildet. In Massive stehen fünf Grundarten von Nodes zur Verfügung, um die Struktur eines Characters abzubilden.

- **Segment Nodes** dienen zur Abbildung des Rigs und stellen ein Pendant zu Bones in Autodesk Maya dar.

- **Spring Nodes** definieren das Verhalten von Segmenten bei den Berechnungen der Rigid Body Dynamics.

- **Geometry Nodes** enthalten die Geometrie des Agenten.

- **Option Nodes** dienen zur Individualisierung der Geometrie einzelner Instanzen des Crowd Agents (siehe Kapitel 10.9 Individualisierung von Crowd Agents, S. 211).

- **Material Nodes** werden verwendet um der Geometrie Texturen und Shader zuzuweisen.

Über den Einsatz und die Verkettung dieser Nodes kann in Massive ein komplexer Crowd Agent entstehen. In den folgenden Kapiteln wird die Funktionalität der einzelnen Segmente genau erläutert.

10.5.1 Skeleton

Der hierarchische Aufbau von Segment Nodes beschreibt in Massive die Struktur des Agent Rigs. Jedes einzelne Segment beschreibt einen Bone des Rigs und besteht aus einem Node, dem eine primitive Form zugrunde liegt. Rotation, Translation, Skalierung und Rotationspunkte der Segmente sind parametrisierbar. Das Rigging in Massive ist jedoch gewöhnungsbedürftig und erfordert einige Zeit zur Erstellung komplexer Rigs.

Einfacher ist es hingegen das Rig extern in Maya umzusetzen und zu importieren. Das gesamte Rig wird beim Import als Maya ASCII Datei (.MA) geladen und von Massive interpretiert. Der hierarchische Aufbau des geladenen Maya Rigs wird automatisch durch die Verknüpfung von Segment Nodes rekonstruiert. Somit ist sichergestellt, dass importierte Agent Rigs sowohl in Massive als auch in Autodesk Maya in Struktur, Dimension und Stellung ident sind. Die Abbildung 84 zeigt das Massive Rig des Treble Bot Agents nach dem Import aus Autodesk Maya.

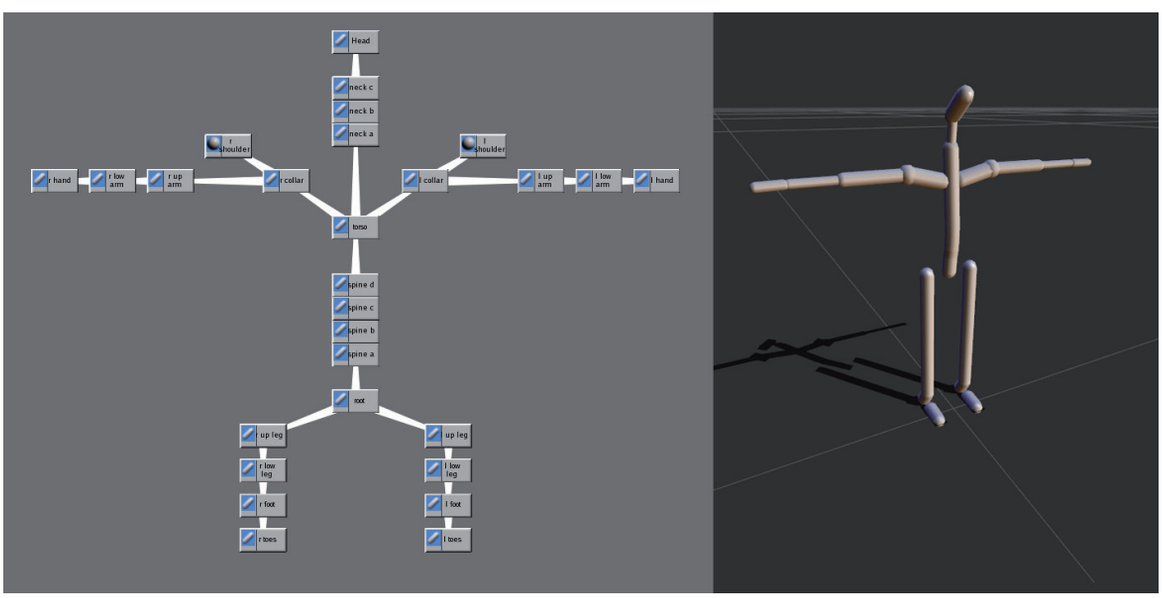


Abbildung 84: Agent Rig des Treble Bot Crowd Agents

Beim Import von Rigs gibt es jedoch einige Punkte, die zu beachten sind. Alle Bones in der Hierarchie eines Rigs müssen von einem Root Segment mit dem Titel [root] ausgehen. Dies hat zur Folge, dass sich auf der Root Ebene in der Hierarchie eines Crowd Agents lediglich ein einziger Bone befinden sollte.

Bone-Segmente und Segmentketten, die über Nullen oder Constraints an das Rig befestigt sind, werden von Massive nicht unterstützt. Driven Keys und Expressions auf Bones gehen beim Import ebenfalls verloren. Des Weiteren sollten schon vor dem Import des Rigs die Namenskonventionen von Massive eingehalten werden. Symmetrisch angeordnete Segmente werden hierbei mit den Endungen [Bonename]_l und [Bonename]_r versehen. Alle Parameter dieser symmetrischen Bones werden in Massive automatisch miteinander verknüpft. Vor allem bei humanoiden Agents stellt diese Verknüpfung eine enorme Erleichterung für die Parametrisierung von Extremitäten dar. Werden Änderungen an einem symmetrisch verknüpften Segment vollzogen, werden diese Parameter ebenfalls auf das gegenüberliegende Segment gespiegelt.

Im DOF (degrees of freedom) Tab des Segment Nodes können Rotationen und Translationen der einzelnen Bones beschränkt werden, um Scharnier-, Kugel- oder Radgelenke zu erzwingen. Diese Einstellungen kommen nur bei kinematischen und physikalischen Berechnungen zum Tragen und werden nicht auf Animationsdaten angewendet. Unterliegt eine Segementkette kinematischen Berechnungen, ist es empfehlenswert, die Rotationsfreiheit der betroffenen Bones genau zu definieren. Im Fall des Bass Bots führte beispielsweise die kinematische Repositionierung der Füße zu unerwünschten Kniestellungen. Dieses Fehlverhalten kann über die Justierung der Rotationsfreiheit der Knie-Segmente vermieden werden.

10.5.2 Geometrie

Ist das Rig eines Crowd Agents fertig konzipiert oder importiert wird im nächsten Schritt die Agent-Geometrie mit dem Rig verbunden. In dieser Phase werden Geometrie Nodes für jedes bestehende polygonale Objekt erstellt. Jedem dieser Nodes wird in weiterer Folge ein Wavefront-Objekt (.OBJ) zugewiesen. Die polygonale Information der Objekte wird beim Laden nicht in der Agent-Datei (.CDL) gespeichert. Für jeden Geometrie Node besteht jedoch eine relative Referenz auf das integrierte Objekt. Somit können Änderungen an der Geometrie des Agents durch die direkte Ersetzung der .OBJ Files jederzeit vollzogen werden. Als Konsequenz ist das erneute Laden der Daten in Massive nicht erforderlich. Dieser Workflow erspart Zeit und erleichtert die Arbeit zwischen Massive Operator und Modeler.

Die Integration der .OBJ Files kann jedoch zu Problemen führen. Massive Prime geht bei der Interpretation von Wavefront-Objekten sehr strikt vor und ist stark an die Datenstruktur von .OBJ Files orientiert, die mit Autodesk Maya exportiert wurden. Jedes 3D Softwarepaket interpretiert und exportiert das .OBJ Dateiformat auf unterschiedliche Art und Weise. Dieser Umstand birgt eine Fehlerquelle bei der Integration von Geometriedaten in Massive. Wavefront .OBJ Daten aus Autodesk Softimage beinhalten beispielsweise überflüssige Information an Vertex Normals. Diese Daten werden von Massive Prime falsch interpretiert und führen zu einer Fehlermeldung beim Import der Daten. Geometrien, die über Autodesk 3ds Max exportiert wurden, führen zu einem Absturz von Massive Prime. Aus diesem Grund ist der Export von Geometrien über Autodesk Maya erforderlich, um die fehlerfreie Integration der Daten in Massive sicherzustellen.

In folgendem Kapitel wird beschrieben, wie die importierten Geometrien mit dem Agent Rig verbunden werden.

10.5.3 Skinning

In der Skinning Phase werden alle Geometrien des Agents mit dem Rig verbunden (siehe Abbildung 85). Das Mesh des 3D Characters kann somit über die Steuerung des Rigs verändert und animiert werden.

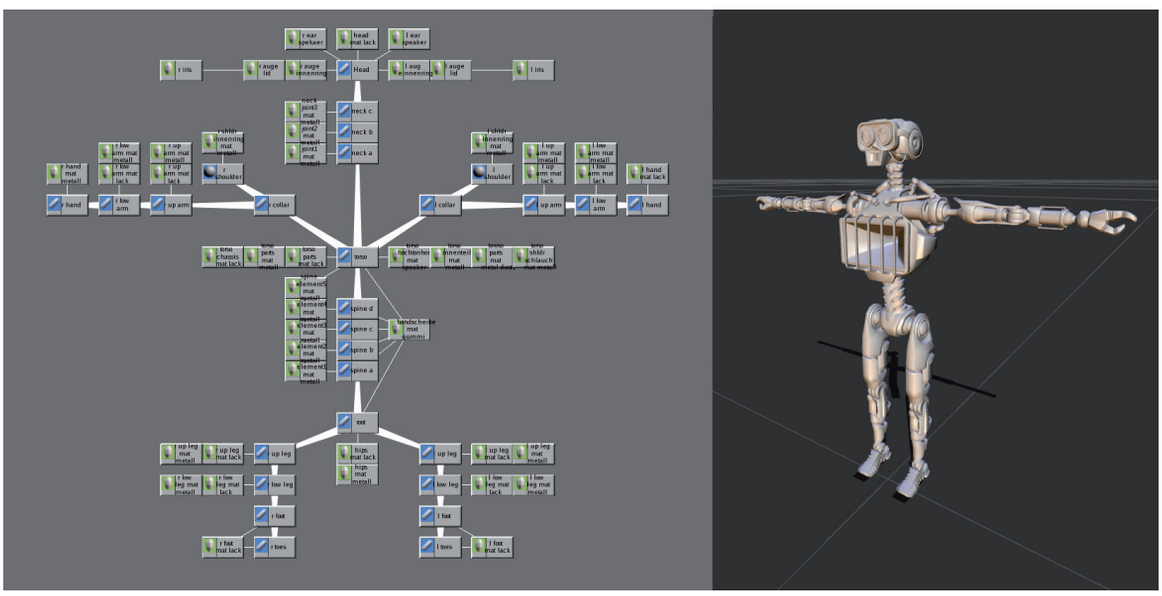


Abbildung 85: Rig des Treble Bot Agents mit Geometrie Nodes

In Massive stehen zwei Arten des Skinnings zur Verfügung:

- **Rigid Binds** werden durch die Verknüpfung eines Geometrie Nodes mit einem Segment Node in einer 1:1 Beziehung erzielt. Diese Verknüpfung hat zur Folge, dass dieser eine Bone des Agents die Position der angehefteten Geometrie steuert. Bei Rigid Binds handelt es sich um eine starre Verbindung, die keine Deformierung der Geometrie erlaubt. Da ein Großteil der Geometrie der Bot Agents aus starren Elementen besteht und eine Deformierung der Geometrie somit unerwünscht ist, empfiehlt sich das Skinning über Rigid Binds.

  Bei der Verknüpfung der Geometrie mit dem Segment Node wird der Nullpunkt des Objekts in den Rotationspunkt des Segmentes verschoben. Diese Translation kann im Geometrie Node mit der Option „world space“ kompensiert werden.

- **Soft Binds** erfolgen durch die Verknüpfung eines Geometrie Nodes mit einer beliebigen Anzahl von Segment Nodes in einer 1:n (n>1) Beziehung. Ist ein Geometrie Node mit keinem einzigen Segment verknüpft, wird dieser von allen Segmenten des Rigs beeinflusst. Der Einfluss eines Segments auf diese Geometrien wird als Envelope bezeichnet und kann in den Einstellungen des jeweiligen Segment Nodes im Bones Tab in Form von Weights gesteuert werden. Im Bones Window können die Envelopes und die daraus resultierenden Einflüsse der Segmente auf die Geometrie des Agents visualisiert werden (siehe Abbildung 86). Bei der durch Animationsdaten resultierenden Rotation, Translation und Skalierung der Segmente des Rigs werden Geometrien im Einflussbereich der Bones deformiert. Soft Binds werden ausschließlich auf Geometrien angewendet, deren Form und Verhalten von mehreren Segmenten beeinflusst wird und erfordern ein höheres Ausmaß an Rechenleistung als Rigid Binds.

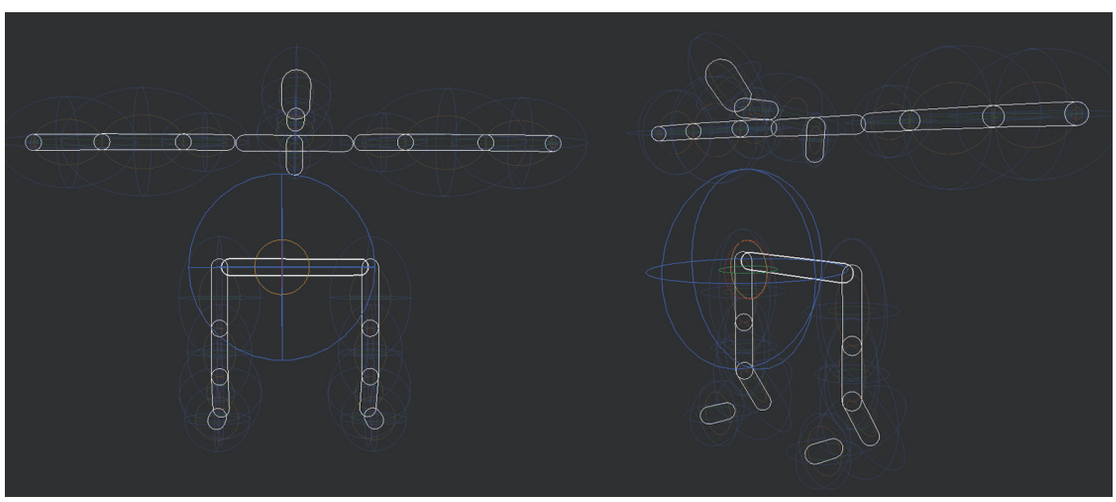


Abbildung 86: Envelopes zur Steuerung der Soft Binds

10.5.4 Blend Shapes

Blend Shapes stellen, neben den auf Bones basierenden Animationen, eine weitere Möglichkeit dar, die Geometrie der Agents zu verformen und zu animieren. „A blend shape deformer enables you to morph the shape of any object into the shape of another object.“ (Meade et al. 2007, S. 230) Blend Shapes werden oft verwendet, um die Gesichtszüge eines 3D Characters zu animieren. In Massive Prime wird das Blenden zwischen geometrischen Formen durch die Definition von Blend Targets in Form von .OBJ Dateien im Blend Tab eines Geometrie Nodes ermöglicht (siehe Abbildung 87). Diese Blend Targets sollten in der Anzahl der Polygone und der Ordnung der Punkte der originalen Geometrie entsprechen. Für jedes Blend Target wird ein Variablenname deklariert. Der Wert dieser Blend-Variable bewegt sich zwischen 0 und 1 und kann im Brain des Crowd Agents gesteuert und gesetzt werden. Auch die Beeinflussung mehrerer Blend-Variablen eines Geometrie Nodes zur selben Zeit, und somit die Überlagerung mehrerer Blends Shapes, ist möglich.

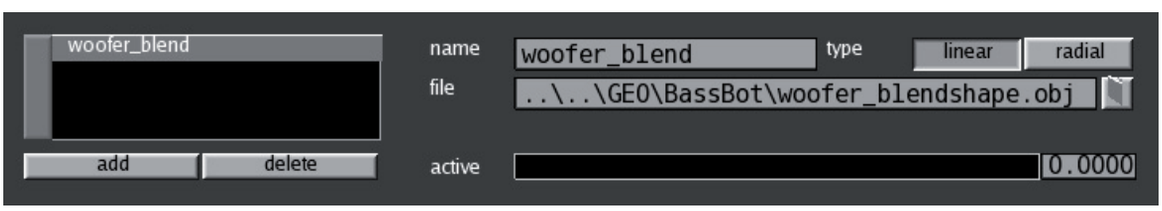


Abbildung 87: Blend Shape

Bei der Berechnung der Blends werden die Differenzen der Positionsdaten der Punkte zwischen ursprünglicher Geometrie und Blend Target errechnet. Die errechnete neue Position eines Punktes ergibt sich aus dem Produkt der errechneten Differenz mit dem aktuellen Wert der Blend-Variable. Ein Blend-Wert von 0 ergibt somit die ursprüngliche Geometrie, während ein Blend Wert von 1 vollständig der Geometrie des Blend Targets entspricht.

Bei der Berechnung der Blend Shapes wird in Massive zwischen zwei Interpolationsverfahren unterschieden.

- **Lineares Blending**
  Hier werden die Punkte geradlinig auf dem Vektor zwischen ursprünglicher Geometrie und Blend Target bewegt. In dieser Fallstudie wurde lineares Blending eingesetzt, um die Bewegungen des kalottenförmigen Lautsprechers des Bass Bot Agents zu animieren.

- **Radiales Blending**
  Hier wird jeder einzelne Punkt der Geometrie auf einer Kreisbahn bewegt, deren Ursprung im Zentrum des Objektes liegt. Dieses Verfahren eignet sich vor allem zum Blenden von runden Objekten wie z.B. dem Augenlid eines Agenten. Somit wird verhindert, dass sich einzelne Polygone beim Öffnen und Schließen des Auges durch den Augapfel hindurch bewegen.

10.5.5 Materialien und Texturen

Die Zuweisung von Materialien erfolgt in Massive über die Verknüpfung von Material Nodes mit den erstellten Geometrie Nodes des Agents (siehe Abbildung 88). Jeder Material Node hält Informationen über die Textur und den Shader des Materials. Texture Maps werden in Form von .TIF Daten als Referenzen in den Material Nodes geladen und an die Geometrie des Agents übergeben.

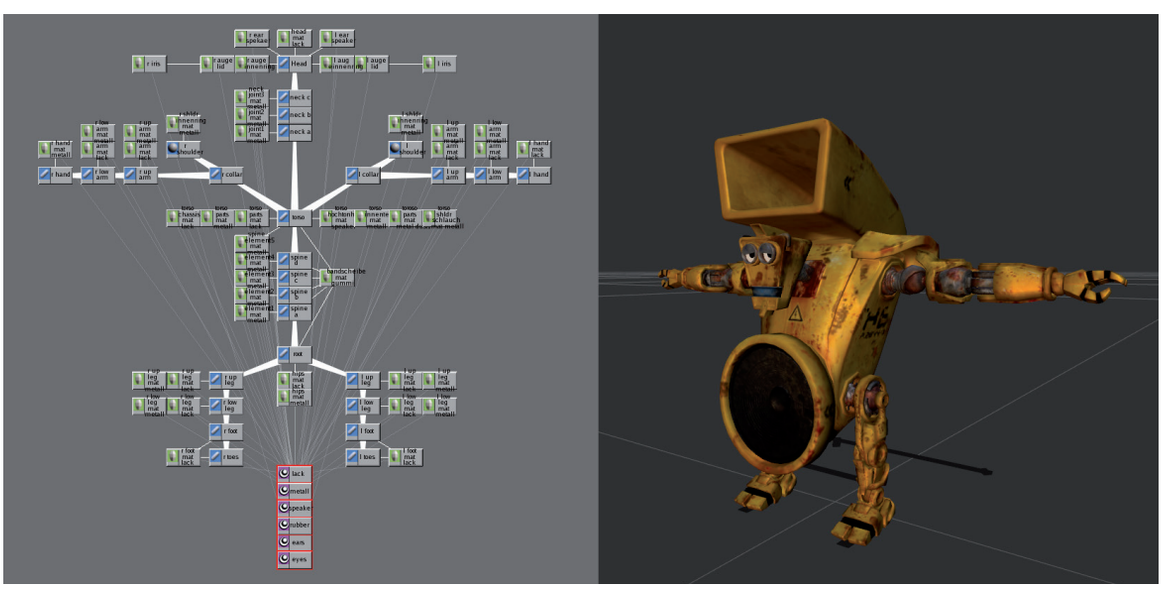


Abbildung 88: Agent Rig des Bass Bot Agents mit texturierten Geometrien

10.5.6 Rigid Body Dynamics

Der Einsatz von Rigid Body Dynamics ermöglicht die physikalische Berechnung der Bewegung von Segmenten, basierend auf Masse und Gravitation. Bei der Berechnung von Rigid Body Dynamics werden Spring Nodes verwendet, um Segmentketten an das Rig zu binden. Diese Springs wirken der Gravitation entgegen und versuchen die Segmente an ihre ursprüngliche Position zu bringen. Die Berechnungen basieren auf dem Hookeschen Gesetz der Federkraft (vgl. Massive Manual 2008, spring_node.html). „Bei der Dehnung oder beim Zusammendrücken einer Feder ist die Längenänderung x der auf die Feder wirkenden Kraft F proportional." (Wenisch 2005, S. 40) Die Steifigkeit der Feder wird über die Federkonstante D kontrolliert.

$$F = D * x$$

Spring Nodes verfügen über fünf individuelle Berechnungsarten.

- **Normal**
  Die Federkraft dieses Springs wird über das Hookesche Gesetz berechnet. Zusätzlich zur Federkonstante kann die Dämpfung der Federkraft reguliert werden. Eine Expansion (Stretch) oder Kontraktion (Squash) der Segmente wird nicht unterstützt.

- **Stretch**
  Dieser Spring Node verhält sich ähnlich wie der Normale Spring. Bei der Berechnung der Stretch Springs wird jedoch die Expansion (Stretch) der Segmente unterstützt.

- **Squash**
  Dieser Spring Node verhält sich ähnlich wie der Normale Spring. Bei der Berechnung der Stretch Springs wird jedoch die Kontraktion (Squash) der Segmente unterstützt.

- **Pin**
  Dieser Spring Node entspricht einem einfachen Position Constraint.

- **Parent**
  Der Parent Constraint bindet das betroffene Segment an die Position des übergeordneten Segments.

Die Kalkulation der Dynamik kann individuell für einzelne Segmenthierarchien aktiviert werden. Im Agent Dynamics Tab auf der Body Page werden die grundlegenden Parameter wie Erdanziehungs- und Reibunskraft für die Berechnung der Rigid Body Dynamics definiert. Die Masse und Dichte der einzelnen Segmente können in den Segmenteigenschaften gesetzt werden.

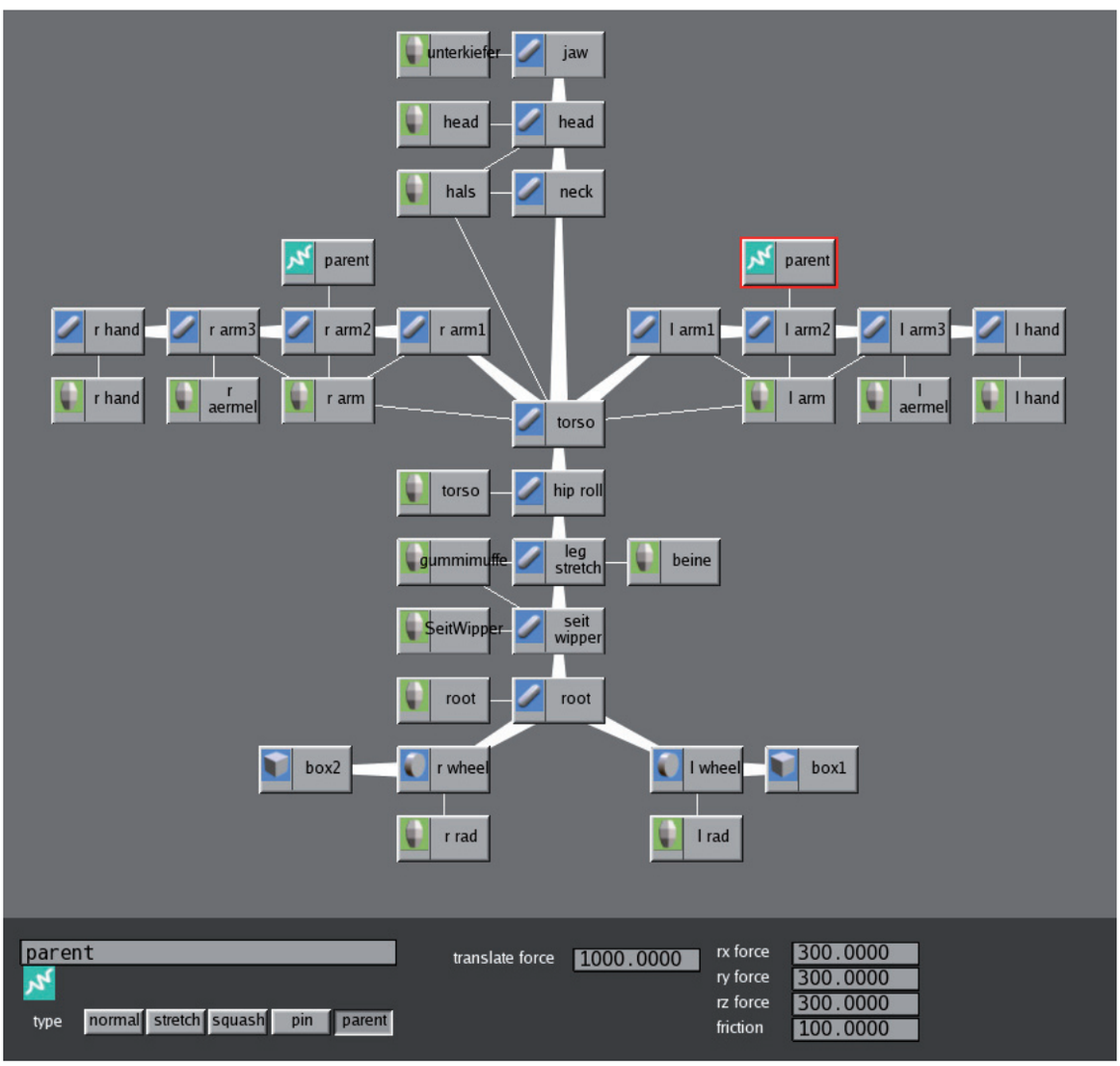


Abbildung 89: Spring Nodes am Rig des Banker Bot Agents

Im Fall des Banker Bots werden die beiden Unterarmsegmente über Parent Springs an die Oberarmsegmente gebunden (siehe Abbildung 89, S. 180). Durch die Aktivierung der Dynamics für die beiden Unterarmsegmente wird die gesamte Segmenthierarchie der Arme in die Kalkulationen mit einbezogen. Somit sind die Bewegungen der schlauchartigen Arme des Banker Bots abhängig von der Bewegung des gesamten Körpers (siehe Abbildung 90).

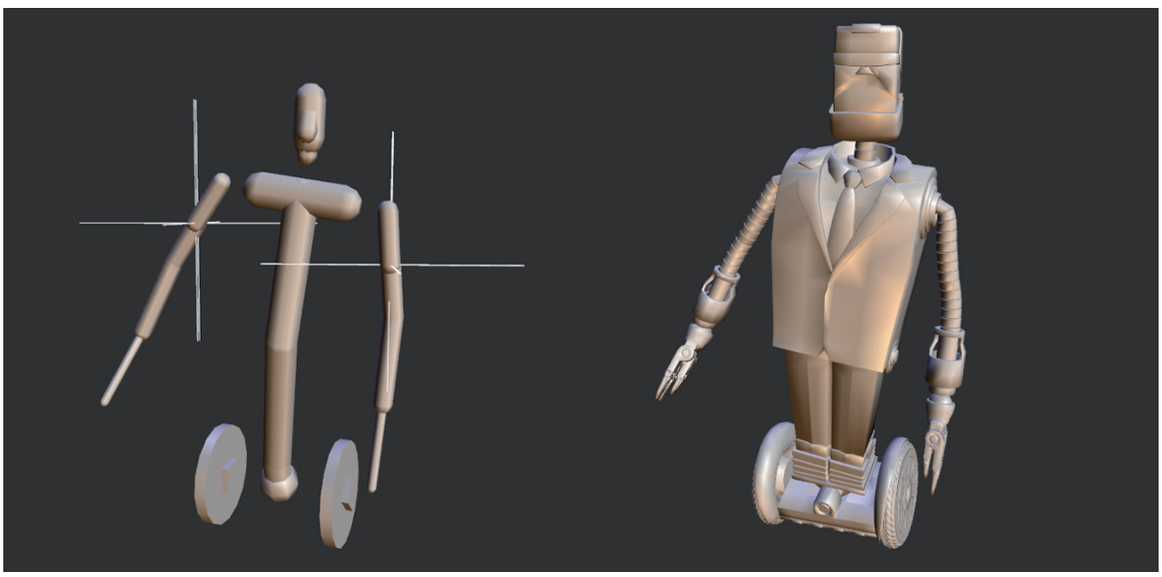


Abbildung 90: Rigid Body Dynamics

10.6 Motion Tree

In Massive Prime bestehen zwei unterschiedliche Systeme zur Animierung von Crowd Agents. Einerseits die programmierte Animierung des Rigs über Brain-Module, andererseits die Animierung des Agent Rigs über importierte Bewegungsdaten. Die Programmierung der Bewegungen jedes einzelnen Segments eines Agents ist für einfache Crowd Agents, wie z.B. Autos oder andere Vehikel, gut geeignet und wird in der Fallstudie eingesetzt, um die Bewegungen des Banker Bot Agents zu steuern (siehe Kapitel 10.8.4 Brain-Module , S. 206). Für komplexe Bewegungsabläufe, wie sie bei menschlichen Charakteren vorkommen, ist dieser Ansatz jedoch unbrauchbar. An dieser Stelle kommen importierte Animationsdaten zum Tragen, die auf das Agent Rig angewendet werden. Um einen Agent auf diese Art und Weise zu animieren, werden mehrere Aktionen

benötigt. Diese Aktionen können in weiterer Folge aneinandergereiht werden, um einen vollständigen Bewegungsablauf abzubilden.

Auf der Motion Page können diese Aktionen definiert und verknüpft werden. In dieser Phase werden alle Optionen festgelegt, die für einen Crowd Agent zu einem beliebigen Zeitpunkt bereitstehen (vgl. Aitken et al. 2004, S. 21) beziehungsweise bereitstehen müssen, um die geplanten Aktionen durchzuführen. Das Resultat dieser verknüpften Aktionen beschreibt den Bewegungsfluss des Agents und wird als Motion Tree bezeichnet.

10.6.1 Motion Design

Für die Umsetzung des Motion Trees wird in erster Instanz für jede Szene festgelegt, welche Bewegungen die einzelnen Crowd Agents durchführen müssen, um die erwünschten gruppendynamischen Abläufe abzubilden. In weiterer Folge werden diese Bewegungsabläufe in die Grundbewegungsarten des Agents unterteilt. Die geplanten Bewegungsabläufe des Bass Bot Agents werden beispielsweise auf zwei Grundformen der Bewegung reduziert: s*tand* und w*alk* (siehe Abbildung 91). Diese beiden Aktionsformen bilden die Basis des Motion Trees und werden in Massive als Transition Nodes bezeichnet. Die Transition Nodes an sich beinhalten keine Animationsdaten, sondern dienen lediglich als zusammenfassendes Element aller untergeordneten Aktionen und als Übergang zwischen Aktionen.

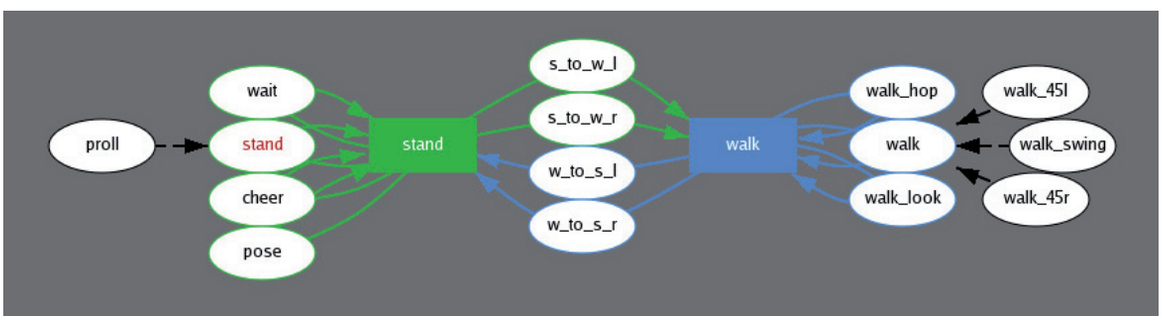


Abbildung 91: Motion Tree des Bass Bot Agents

In weiterer Folge werden alle relevanten Bewegungen und Aktionen für Transition Nodes definiert und verknüpft. Diese Aktion Nodes dienen als Platzhalter für die eigentlichen Animationsdaten aus Autodesk MotionBuilder. Jeder Aktion Node wird mit Daten aus importierten Animationsdateien (.MA) gespeist. Die Verknüpfung von Aktion Nodes im Motion Tree kann auf drei Arten erfolgen.

- Die gerichtete Verknüpfung einer Aktion zwischen zwei Transition Nodes dient als Übergangsaktion zwischen den durch die Nodes definierten Grundbewegungsformen. Ein Übergang zwischen den Transition Nodes *stand* und *walk* erfordert beispielsweise eine gerichtete Aktion von *walk* nach *stand* und eine weitere Aktion von *stand* nach *walk*.

- Eine intratransitionale Aktion ist innerhalb eines Transition Nodes verknüpft. Die Aktion geht von einem bestimmten Node aus und leitet auf selbigen zurück. Es entsteht eine Art Schleife. Ein einfaches Beispiel stellt die Aktion *walk* dar. Diese beinhaltet den Walk Cycle eines Agenten und ist mit dem gleichnamigen Transition Node verknüpft. Somit kann der Walk Cycle innerhalb des Transition Nodes geloopt werden.

- Die einfache Verknüpfung zweier Aktionen initiiert einen Motion Blend. Die Bewegungsdaten beider Aktionen werden hierbei mit einem Faktor überblendet.

Der gesamte Motion Tree kann als Textdokument (.TXT) exportiert werden. Das Dokument enthält die textuelle Beschreibung aller benötigten Bewegungsdaten für den konzipierten Crowd Agent und fungiert als Referenz für die Erstellung der benötigten Animationen (siehe Kapitel 9.2 Motion Capturing, S. 127). Der Animator kann somit alle benötigten Bewegungen sequentiell nach dieser Liste abarbeiten und an den Massive Operator weitergeben. Auf den Import von Animationsdaten und die Verknüpfung der Bewegungen mit dem Motion Tree wird im folgenden Kapitel näher eingegangen.

10.6.2 Aufbereitung und Integration von Animationsdaten

In dieser Phase werden die aus Autodesk Maya exportierten Animationsdaten (siehe Kapitel 9 Character Animation, S. 119) mit dem Motion Tree eines Crowd Agents verknüpft. Der Import der Bewegungsdaten erfolgt über die Integration der Animationsdaten in Form von Maya ASCII (.MA) Dateien.

Die importierten Daten werden auf das Agent Rig übertragen und können im Action Editor von Massive bearbeitet und verwaltet werden. Die Verknüpfung der Bewegungsdaten mit Aktionen im Motion Tree erfolgt über die Namensgebung. So wird beispielsweise die Bewegung mit dem Namen *walk* im Action Editor dem gleichnamigen Action Node im Motion Tree des Agents zugewiesen. Die Kurven und Werte jedes in der Animation vorhandenen Bones können editiert werden, um etwaige Fehler oder unerwünschte Keyframes in den Animationskurven zu entfernen. Um die importierten Bewegungsdaten für Massive aufzubereiten sind jedoch weitere Schritte von Nöten, die in folgenden Kapiteln näher beschrieben werden.

10.6.2.1 Loops

Ist eine Animation fertig gesäubert, wird die gesamte Animation oder ein bestimmter zeitlicher Ausschnitt daraus bestimmt, der die tatsächliche Bewegung des Agenten definiert. Besonders bei geloopten Bewegungen, wie z.B. einem Walk Cycle, ist die exakte Definition des Ausschnitts wichtig, um einen reibungslosen Übergang zwischen Ende und Anfang der Bewegung sicherzustellen. Massive bietet für diese Zwecke die Option, bei der Definition eines Loops zwischen Ende und Anfang des Aktionsausschnitts zu blenden. Diese, in Massive als Crossfade bezeichnete, Überblendung würde die Arbeit mit Bewegungsdaten erheblich beschleunigen und vereinfachen, ist jedoch sehr fehleranfällig. Die Durchführung eines Crossfades führte bei Tests mit den Massive Prime Versionen 3.0.3 und 3.5.5 sowohl unter Linux als auch unter Windows ausnahmslos zu einem Absturz der Software. Die Überblendung und genaue Definition eines Animationsloops muss deshalb bereits bei der Bearbeitung der Bewegungsdaten in Autodesk MotionBuilder erfolgen (siehe Kapitel 9.3.2 Keyframe Animationen mit Control Rigs, S. 146).

10.6.2.2 Agent Curves

Um die Bewegungsinformation im Brain des Crowd Agents für die Berechnung von Animationsabläufen nutzen zu können, werden aus den importierten Bewegungsdaten Agent Curves errechnet. Bei der Berechnung der Agent Curves werden aus der globalen Translation und Rotation des Root Segments der Animationsdaten lokale Velocity Curves generiert. Ein fixer Wert von 1 in der *tz* Kurve eines Agents beschreibt beispielsweise die kontinuierliche Translation des Agents auf der z-Achse mit einer Geschwindigkeit von einer Einheit pro Sekunde (vgl. Massive Manual 2008, ae_agent.html). Um eine positionsunabhängige Aneinanderreihung von Aktionen zu ermöglichen, wird die aktuelle Position der Agents in der Simulation anhand dieser Velocity Curves errechnet. Somit können Aktionen beliebig ohne Sprünge in den Rotations- und Positionskurven des Root Segments kombiniert werden.

Bei der Generierung der Agent Curves können Translation und Rotation der Achsen bewegungsabhängig eingeschränkt werden, um ein zufriedenstellendes Ergebnis zu erreichen. Werden die Velocity Curves mit falschen Parametern berechnet, kann es bei der Aneinanderreihung von Aktionen zu unerwünschten Positionssprüngen der Agents kommen. Ein einfacher Walk Cycle benötigt beispielsweise ausschließlich die Berechnung der Translationen auf der z- und x-Achse. Die resultierenden Agent Curves für diesen Walk Cycle beinhalten demnach Kurven für tz und tx. Erfolgt eine Drehung des Agents in der Bewegung, ist die Berechnung der Rotationskurve ry von Nöten. Um die Erstellung der Agent Curves zu vereinfachen, bietet Massive Prime ein Set an bereits definierten bewegungsspezifischen Berechnungseinstellungen (siehe Abbildung 92).

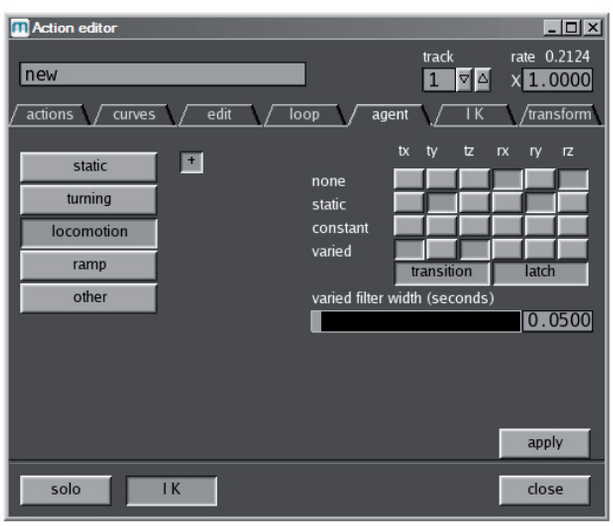


Abbildung 92: Generierung der Agent Curves im Action Editor

- **Static**
  Allen Agent Curves wird sowohl für die Rotation als auch für die Translation ein statischer Wert zugewiesen. Der Ursprung des Agents befindet sich demnach in einem fixen Punkt. Diese Parameter eigenen sich für statische Bewegungen, wie beispielsweise Stehen oder Sitzen.

- **Turning** (tx, tz, ry)
  Diese Einstellung wird für Bewegungen eingesetzt, welche eine Drehung des Agents auf der y-Achse beinhalten.

- **Locomotion** (tx,tz)
  Hier werden ausschließlich Agent Curves für die Translationen auf der x und z-Achse errechnet. Diese Einstellung wird in erster Linie für geradlinige Bewegungen, wie z.B. einem Walk Cycle, verwendet.

- **Ramp** (tx,ty,tz)
  Diese Einstellung erweitert die der Locomotion um die Berechnung der Translation auf der y-Achse und eignet sich für aufwärts oder abwärts gerichtete Bewegungen.

10.6.2.3 Kinematics

Für die Anpassung und Veränderung von Bewegungen im Brain des Agents reichen die generierten Agent Curves nicht aus. Um einen Bewegungsablauf zu modifizieren, ohne die physikalischen Grundstrukturen eines Agents zu verletzen, sind kinematische Berechnungen erforderlich. Die Definition der IK Chains erfolgt nicht wie in den meisten 3D Softwarepaketen über das Agent Rig, sondern wird direkt in den Bewegungsdaten im Action Editor bestimmt.

Zur Berechnung der Kinematik werden folgende Kurven definiert.

- **RC Curves**

  Für die ausgewählten Segmente des Rigs werden Kurven für die Einschränkungen der Rotation (Rotation Constraint) anhand der Animationsdaten errechnet. Diese Daten fließen in die Berechnung der IK Curves mit ein und stellen sicher, dass die Rotation der betroffenen Segmente bei einer Repositionierung durch kinematische Berechnungen nicht beeinflusst wird. RC Curves werden in erster Linie für die Berechnung der Füße herangezogen, um sicherzustellen, dass diese bei Bodenkontakt immer flach auf dem Untergrund liegen.

- **IK Curves**

  Das ausgewählte Segment, als Endpunkt einer Segmentkette, wird als Effektor für die Berechnung der inversen Kinematik herangezogen. Segmentspezifische Constraints (siehe Kapitel 10.5.1 Skeleton, S. 172) werden bei der Berechnung der Kurven berücksichtigt. IK Curves werden häufig auf die Extremitäten menschlicher Agents angewendet und ermöglichen die lokale Translation, Rotation und Skalierung der Effektoren. Somit kann eine Anpassung der Schritte eines Agents an das Terrain erfolgen, ohne die Animation in ihren Grundzügen zu verändern.

- **Hold Curves**

  Anhand von parametrisierbaren Schwellenwerten werden Hold Curves für Effektoren einer IK Chain errechnet (siehe Abbildung 93). Ist der Hold-Wert der Kurve 1 wird das ausgewählte IK Segment an die derzeitige statische Position gebunden. Ist der Wert der Hold-Kurve 0 wird die ursprüngliche Animation des Segmentes nicht beeinflusst. Durch die Definition von Hold Curves kann beispielsweise das Gleiten der Fußflächen auf dem Untergrund verhindert werden.

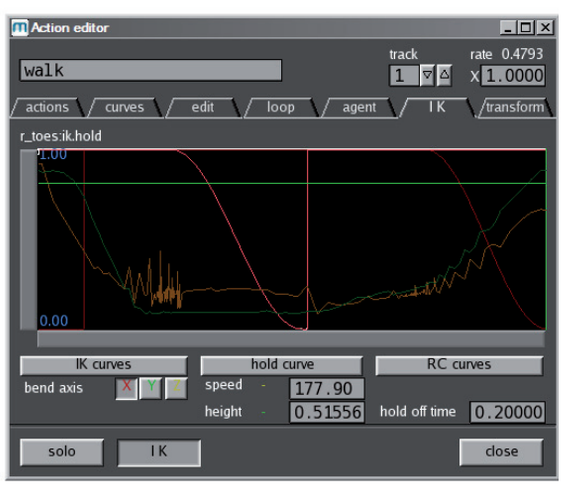


Abbildung 93: Hold Curves

Anhand des folgenden Abschnitts wird der typische Workflow zur Definition der Kinematik anhand des Bass Bot Agents erläutert. Diese Vorgehensweise beschreibt die Generierung von IK, RC und Hold Curves für den primären Fortbewegungsapparat von humanoiden Agents.

In der ersten Phase werden RC Kurven für beide *Foot* Segmente generiert. Diese Constraints stellen sicher, dass sich die Füße des Agents, auch nach der Berechnung der Kinematik, flach auf dem Boden befinden. In weiterer Folge werden die *Toe* Segmente des Agents als IK Effektoren definiert. Für diese Effektoren werden IK Curves und Hold Curves generiert. Das Ergebnis der Berechnungen wird anschließend über die Justierung des Hold Curve Schwellenwerts optimiert. Ist das Ergebnis noch nicht zufriedenstellend, können die aus dem Schwellenwert resultierenden Hold Curves im Curve Editor einer speziellen Behandlung unterzogen werden. Besonders bei komplexen Schrittbewegungen ist die Anpassung der einzelnen Kurven oft erforderlich, um ein zufriedenstellendes Ergebnis zu erzielen.

Zur Erleichterung im Umgang mit Hold Curves können IK Effektoren in der Bewegung am Rig im Viewport visualisiert werden (siehe Abbildung 94).

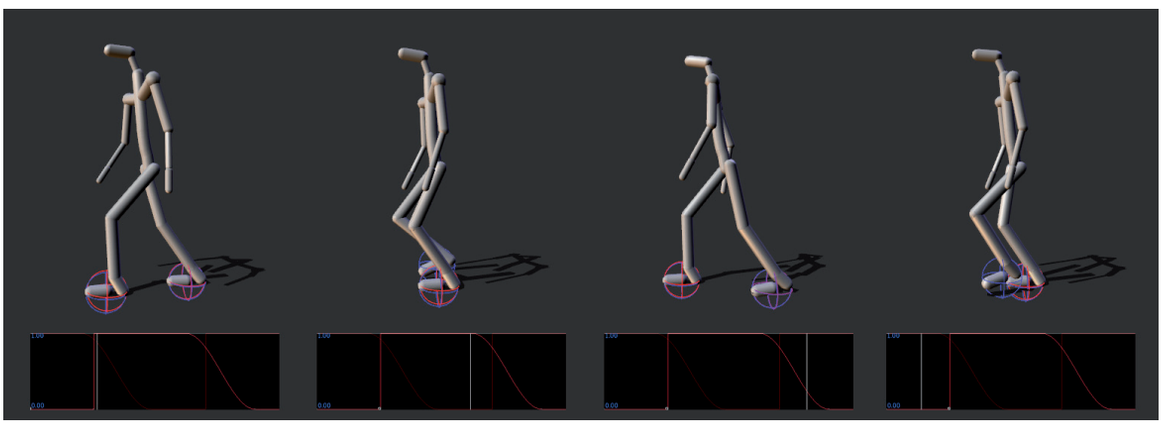


Abbildung 94: IK Hold Curve Visualisierung

10.6.2.4 Transition Curves

Bei einer sequentiellen Folge von mehreren Bewegungen werden die erstellten Aktionen nacheinander auf das Agent Rig projiziert. Beim Übergang von zwei Aktionen kann es jedoch zu Sprüngen in den Bewegungskurven kommen, die ungewollte Jittereffekte in der resultierenden Animation des Agents zur Folge haben. Nicht alle Bewegungskurven des Agents fügen sich nahtlos ineinander und müssen aus diesem Grund geblendet werden. Die Transition Curve bestimmt das Verhalten dieses Blendfaktors über die Zeitachse beim Übergang auf die betroffene Aktion. Diese Kurve kann individuell für jede Aktion definiert werden. Hierbei werden Start und Endpunkt sowie die Krümmung der Kurve auf der Zeitachse der Aktion gesetzt (siehe Abbildung 95).

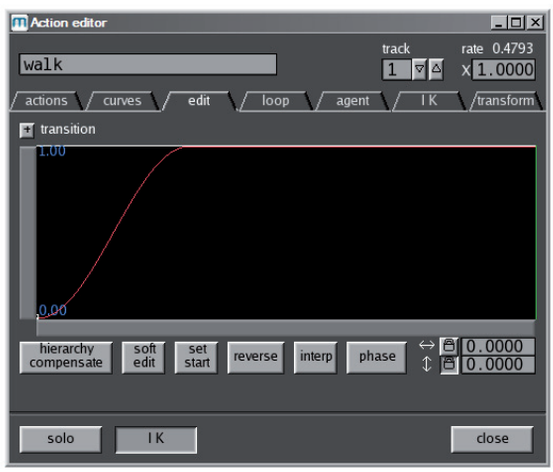


Abbildung 95: Transition Curve

Die Berechnung des Transition Blends tritt nur beim Übergang von zwei verschiedenen Aktionen in Kraft. In sich geloopte Bewegungsdaten werden demnach nur beim Einstieg geblendet. Die Transition Curve beeinflusst jedoch nicht den Loop der Bewegung an sich. Bei einer Aktion, die aufgrund der einfachen Verknüpfung im Motion Tree (siehe Kapitel 10.6.1 Motion Design, S. 182) bereits als Blend fungiert, ist die Transition Curve obsolet.

10.6.2.5 Latch Curves

Durch die Latch Curve werden die möglichen Ausstiegspunkte einer Aktion definiert. Unabhängig von der gesamten Länge einer Aktion wird mit Hilfe der Latch-Kurve festgelegt, zu welchen Zeitpunkten der Übergang auf eine andere Aktion möglich ist. Der Graph dieser Latch-Kurve beschreibt eine Rechteckfunktion, deren Werte entweder 0 oder 1 betragen. Der Wert 0 wird als Latch Low bezeichnet und bestimmt eine Zeitspanne, die den Übergang zu einer anderen Aktion ermöglicht (siehe Abbildung 96).

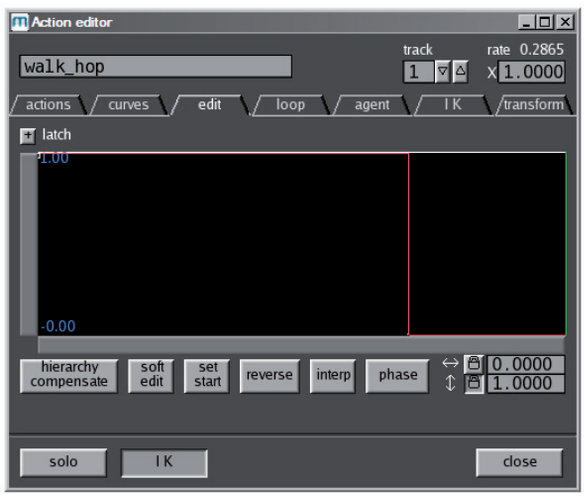


Abbildung 96: Latch Curve

Anhand folgender Beispiele soll die Funktionalität und Anwendung der Latch Curves genauer erklärt werden.

Die Aktion *stand* beinhaltet beispielsweise eine dreisekündige, geloopte Animation des Crowd Agents während der Stehphase. Wird der Ausstiegspunkt dieser Animation am Ende der Aktion gesetzt, kann ein Übergang zur nächsten Animation erst nach einem vollständigen Abspielvorgang der *stand*-Aktion erfolgen. Im ungünstigsten Fall tritt also bei einem erwünschten Übergang von stand auf eine andere Aktion die Folgeanimation mit einer dreisekündigen Verzögerung ein. Um diesen unerwünschten zeitlichen Versatz zu eliminieren, wird die Latch-Kurve der Aktion angepasst. Ein Latch Low nach etwa einer Sekunde gewährleistet somit, dass die maximale Zeitverzögerung bei einem erwünschten Aktionsübergang gering gehalten wird.

In den meisten Fällen verfügen Aktionen über einen einzigen Ausstiegspunkt. Bestimmte Animationen erfordern jedoch die Definition mehrerer Latches. Beim Übergang von der Aktion Gehen (walk) zu der Aktion Stehen (stand) ist eine geringe Latenz von Nöten, um den Agent vor Kollisionsobjekten so schnell

als möglich in einer natürlichen Bewegung zum Stillstand zu bringen. Ist also in der Gehbewegung der rechte Fuß in Bodenkontakt, kann der nächste Schritt des linken Fußes den Stillstand des Agents einleiten. Befindet sich zu diesem Zeitpunkt der linke Fuß des Crowd Agents am Boden, kann die Stehbewegung mit dem rechten Fuß erfolgen. Dieser Umstand erfordert die Bereitstellung von zwei Animationen, welche den Bewegungsablauf von der Gehbewegung in den Stand, sowohl mit dem rechten als auch mit dem linken Fuß, abbilden. Die Latches der Gehbewegung müssen in weiterer Folge so gesetzt werden, dass der unmittelbare Einstieg in die Stehbewegung an zwei Punkten des Walk Cycles erfolgen kann. Somit wird eine der beiden *walk-to-stand* Animationen in Abhängigkeit von der derzeitigen Beinstellung des Agents eingeleitet. In diesem speziellen Fall tritt jedoch ein Problem auf. Die beiden Übergangsanimationen von der Geh- in die Stehbewegung stehen in einer direkten Abhängigkeit zum Ausstiegspunkt der Gehbewegung. Das bedeutet, dass jedes der beiden Zeitfenster der Latch-Kurve die dementsprechende Folgeanimation vorgibt. Die beiden *walk-to-stand* Aktionen müssen demnach jeweils einem spezifischen Ausstiegspunkt im Walk Cycle zugewiesen werden. Dies geschieht über die Erstellung von Latches in Form von Tree-Variablen. Auf dieses Verhalten wird im folgenden Kapitel näher eingegangen.

Um den reibungslosen Ablauf von Aktionssequenzen zu überprüfen und etwaige Fehler im Zusammenspiel zwischen Latch Curves und Transition Curves zu erkennen, verfügt Massive Prime über ein sehr nützliches Tool. Im Sequenzer können alle erstellten Aktionen beliebig hintereinander gereiht, geloopt und wiedergegeben werden. In der Testphase empfiehlt es sich, jede mögliche Kombination aus Aktionen zu überprüfen und gegebenenfalls Einstellungen zu korrigieren.

10.6.2.6 Tree-Variablen

Um den Ablauf der Bewegungsabfolgen eines Crowd Agents anhand des Motion Tree-Modells zu steuern, ist die Deklaration von Tree-Variablen von Nöten. Diese Variablen stellen das Bindeglied zwischen dem Motion Tree und dem Brain des Crowd Agents dar. Durch die Zuweisung verschiedenster Tree-Variablen an einzelne Aktionen im Motion Tree werden somit Bewegungsabfolgen im Brain kontrollierbar.

Folgende Typen von Variablen können deklariert und den Aktionen zugewiesen werden.

- **Latch-Variablen**
  Im Gegensatz zu den Latch Curves im Action Editor werden hier keine Ausstiegspunkte definiert. Hier kann, über eine Hüllkurve, für jeden Ausstiegspunkt einer Aktion eine Variable initialisiert werden. In unserem Fall werden für die beiden Ausstiegspunkte aus der *walk* Bewegung die Variablen *leftfoot* und *rightfoot* deklariert (siehe Abbildung 97). Den *walk-to-stand* Bewegungen kann in weiterer Folge eine der beiden Variablen zugewiesen werden. Somit ist sichergestellt, dass beim Übergang von der Geh- in die Stehbewegung die richtige Aktion ausgeführt wird.

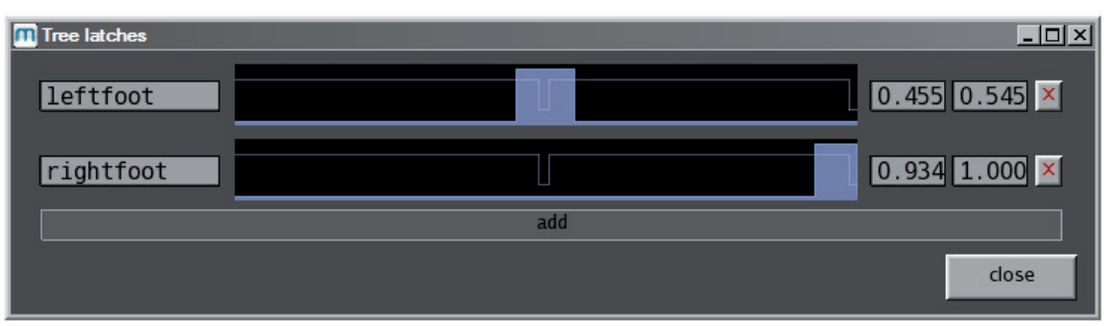


Abbildung 97: Latch Variablen

- **Trigger-Variablen**
  Über den Einsatz von Trigger-Variablen wird der Aktionsfluss im Motion Tree gesteuert. Für jede Aktion auf der untersten Ebene des Motion Trees wird eine Variable mit eindeutigem Namen definiert und der spezifischen Aktion zugewiesen. Wird eine dieser Variablen im Brain des Agents aktiviert, wird der Aktionsfluss im Motion Tree auf die betreffende Zielaktion geleitet. Hierbei kalkuliert Massive den kürzesten Weg im Motion Tree von der aktuell ausgeführten Aktion zur Zielaktion. Alle Action Nodes, die sich zwischen der ausgehenden Aktion und der Zielaktion befinden, werden automatisch sequentiell ausgeführt.

  Im Motion Tree des Bass Bot Crowd Agents wurden beispielsweise unter anderem Trigger-Variablen für die Aktionen *stand* und *walk* angelegt. Befindet sich der Crowd Agent in der *stand*-Bewegung während der *walk*-Trigger aktiviert wird, vollführt der Crowd Agent automatisch die *stand-to-walk*-Animation, die als Bindeglied zwischen dem *stand* und dem *walk* Transition Node fungiert, bevor der Agent schließlich in die *walk*-Bewegung übergeht. Wird in weiterer Folge der *stand*-Trigger ausgelöst, wird je nach der aktuellen Beinstellung des Agents die zugehörige, durch den Latch definierte (siehe Kapitel 10.6.2.5 Latch Curves, S. 190), *walk-to-stand*-Animation eingeleitet.

- **Modifier-Variablen**
  Modifier-Variablen sind in ihrer Funktionalität mit Trigger-Variablen zu vergleichen. Ein Modifier kann als Erweiterung der Trigger verstanden werden. Aus der Kombination der aktiven Trigger- und Modifier-Variable, ergibt sich die Möglichkeit komplexere Bewegungsmuster abzubilden. So kann beispielsweise der Trigger *block* eine Abwehrbewegung einleiten. Wird der Modifier *high* gesetzt und ausgelöst, werden Trigger und Modifier kombiniert und lösen die Aktion *block_high* aus, um eine nach oben gerichtete Abwehrbewegung einzuleiten. Nach selbigem Prinzip können ebenfalls Modifier für *middle* und *low* gesetzt werden um, die *block*-Aktion zu erweitern.

- **Blend-Variablen**
  Eine Blend-Variable definiert den Blendfaktor für Action Nodes, die als Blends fungieren. Eine deklarierte Blend-Variable kann mehreren Action Nodes zugewiesen werden, um das Bewegungsverhalten aktionsübergreifend zu beeinflussen. So können beispielsweise Gemütszustände wie *happy* oder *sad* als Blend-Variablen eingesetzt werden. Im Fall des Treble Bots wurden Blend Actions auf den Walk Cycle angewendet, um jedem Agent durch das Blenden zweier unterschiedlicher Gehbewegungen einen individuellen Charakter zu verleihen.

- **Output-Variablen**
  Im Gegensatz zu anderen Tree-Variablen werden Outputs verwendet, um den aktuellen Bewegungsstatus im Brain des Agents abzufragen. Output-Variablen können verschiedenen Action Nodes zugewiesen werden. Der Wert dieser Variablen wird durch die Aktivität der entsprechenden Nodes gesetzt.

  Im Fall des Bass Bot Agents wurden beispielsweise die Output-Variablen *stationary* und *moving* deklariert, um die aktuelle Bewegungsart des Agents abzufragen. Diese Output-Variablen werden im Brain des Agents verwendet und weiter verknüpft, um die Aktivierung von Trigger-Variablen in Abhängigkeit des aktuellen Zustands zu steuern.

10.7 Sinnesorgane

Um ihre Umgebung wahrzunehmen und auf selbige zu reagieren, verfügen Crowd Agents über eine Reihe von menschenähnlichen Sinnesorganen. Die Wahrnehmung des Umfeldes ermöglicht die Interaktion zwischen einzelnen Agents. Kollisionsvermeidung, Gruppenbildung oder Follow-the-Leader-Verhalten können über die Verarbeitung dieser Sinne im Brain des Agents verwirklicht werden. In den meisten Fällen reichen ein bis zwei Sinnesorgane aus, um ein menschenähnliches Verhalten zu erzielen.

- **Vision**

  Über die Vision verfügt der Agent über ein Blickfeld, welches analysiert und interpretiert werden kann. Bei aktivierter Vision wird das Blickfeld jedes einzelnen Agents gerendert und im Brain des Agents über Kanäle zur Verfügung gestellt (siehe Abbildung 98). Der Sehbereich der Agenten sowie die Auflösung der Renderings ist parametrisierbar. Über die Interpretation der Vision Information ist die genaue Bestimmung von Position, Geschwindigkeit und Farbe möglicher Hindernisse im Umkreis des Agents möglich. Die Kalkulation der Vision ist jedoch bei der Simulation von großen Szenen rechenaufwändig, da das Blickfeld jedes Agents für jeden einzelnen Frame der Simulation berechnet wird.

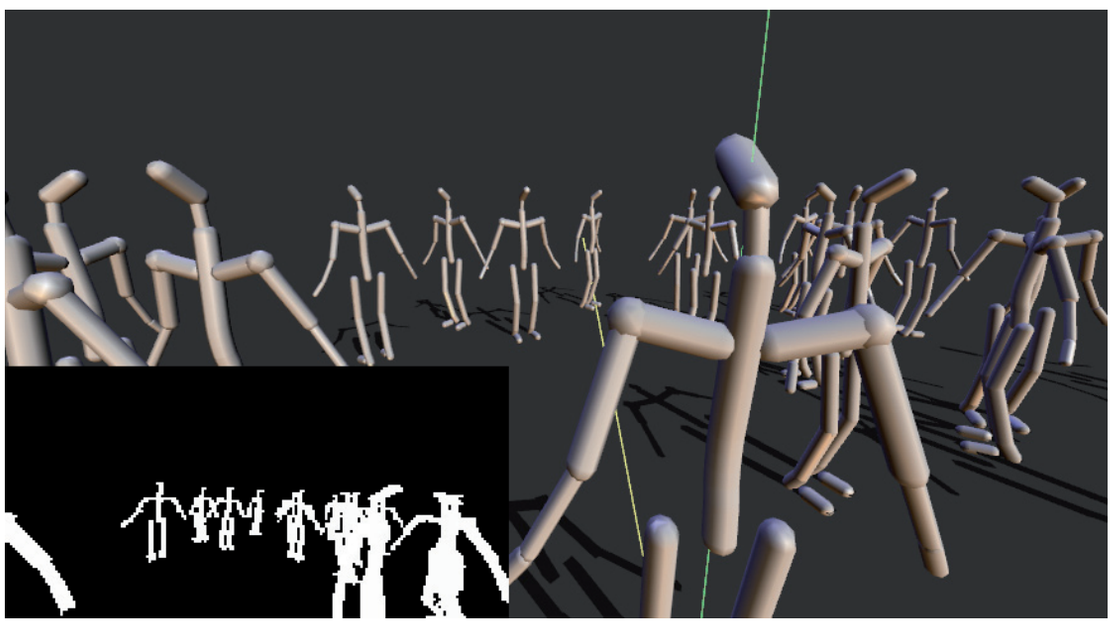


Abbildung 98: Vision

- **Sound**

  Soundkanäle entsprechen dem Gehör der Crowd Agents. Agents können sowohl Signale emittieren als auch empfangen. Die Reichweite der emittierten Sounds wird durch die Amplitude *a* in dB SPL einen Meter vom Ursprung des Agents angegeben. Die Frequenz des ausgestrahlten Signals kann ebenfalls gezielt gesetzt werden. Somit können unterschiedlichen Agents in der Szene verschiedene Frequenzen zugewiesen werden, auf welche individuell reagiert werden kann (siehe Abbildung 99, S. 196).

Im Brain des Agents können die emittierten Signale empfangen und interpretiert werden. Ähnlich wie beim Menschen erfolgt die Tiefenstaffelung durch die wahrgenommene Amplitude des Signals. Des Weiteren kann der relative Ursprung des empfangenen Sounds im Raum berechnet werden. Diese Parameter können in weiterer Folge verknüpft werden, um beispielsweise eine Kollisionsvermeidung zwischen Crowd Agents zu implementieren.

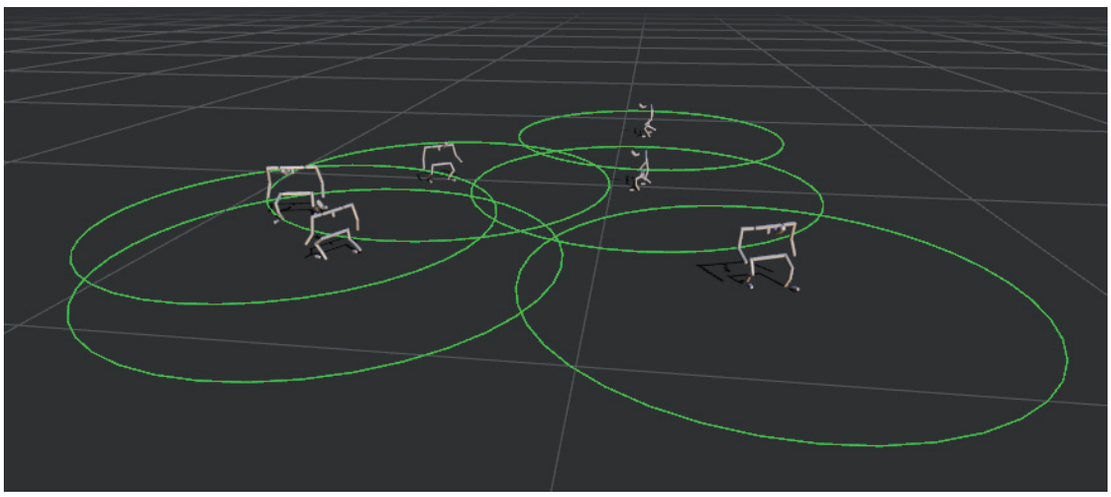


Abbildung 99: Sound Emission

- **Collision**
  Kollidieren die Segmente zweier Agents, kann diese Information im Brain der Agents herangezogen werden, um auf diese Interaktion entsprechend zu reagieren. Die genaue Position und Geschwindigkeit der Kollision stehen über Kanäle im Brain des Agents zur Weiterverarbeitung bereit. Diese Informationen können beispielsweise in einer Kampfszenerie in Massive verwendet werden, um die Reaktion der Agents auf Schwerthiebe oder andere Einflüsse zu steuern.

- **Agent Fields**
  Das grundlegende Konzept der Agent Fields ist vergleichbar mit dem Sound System. Im Gegensatz zur Emission von Signalen basieren Agent Fields jedoch auf Vektoren. Agent Fields gliedern sich in drei verschiedene Typen: Repel, Atract und Align.

- Repel Fields können genutzt werden, um Agents auseinander zu bewegen.
- Attract Fields generieren einen Vektor, der benutzt werden kann, um Crowd Agents näher zusammenzuführen.
- Align Fields werden verwendet, um Crowd Agents aneinander auszurichten.

Im Zuge der Fallstudie wurden die verschiedenen Sinnesorgane und deren Funktionsweisen getestet und analysiert. Zur Realisierung der Massenszenen wurde aufgrund der einfachen Handhabung und der, durch diese Methode erzielten, realistischen Ergebnisse schließlich eine auf Sound basierende Navigation und Kollisionsvermeidung implementiert (siehe Kapitel 10.8.4.2 Kollisionsvermeidung über Soundemission, S. 208).

10.8 Brain

Im Brain des Crowd Agents werden sämtliche Daten erfasst und verarbeitet. Hier werden das Verhalten der einzelnen Agents und die gruppendynamischen Abläufe der Agents zueinander definiert, um eine menschenähnliche Intelligenz der Akteure nachzubilden. In diesem Zusammenhang wird der Begriff der Künstlichen Intelligenz (A.I. Artificial Intelligence) verwendet.

10.8.1 Künstliche Intelligenz

„Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better.“ (Rich 1983, S. 2)

Um das menschliche Verhalten der Crowd Agents abzubilden, werden äußere Einflüsse und Charaktermerkmale definiert, die den Entscheidungsprozess der einzelnen Agents beeinflussen. Bei diesem Entscheidungsprozess steht jedoch nicht die Genauigkeit der Berechnungen im Vordergrund. Vielmehr ist es wichtig diesen Prozess auf einer menschenähnlichen Logik anzulegen. Im Bereich der künstlichen Intelligenz gibt es bereits eine Vielzahl an Modellen, die sich mit diesem Problem auseinandersetzen und Lösungen anbieten. Massive Prime bedient sich der Unscharfen Logik (Fuzzy Logic), um diese menschenähnlichen Entscheidungsprozesse zu simulieren.

10.8.2 Fuzzy Logic

Die Theorie der Unscharfen Logik (Fuzzy Logic) wurde durch die 1965 veröffentlichte Arbeit „Fuzzy Sets“ von L. A. Zadeh begründet. Anders als die Klassische Logik versucht dieses Modell durch die Unschärfe von Variablendeklarationen subjektive, menschliche Entscheidungsprozesse zu simulieren. „Sie knüpft an das Konzept der mehrwertigen Logik, insbesondere der Wahrscheinlichkeitslogik, an und macht anstelle des Begriffs ‘Aussage’ den Begriff der ‘unscharfen (fuzzy) Aussage’ zum Grundbegriff. Eine unscharfe Aussage ist nicht entweder wahr oder falsch. Eine unscharfe Aussage ist ein Satz, dem ein reeller Zahlenwert zwischen 0 und 1 zugeordnet ist.“ (Winter 2001, S. 174) Dieser Zahlenwert wird als Unschärfewert bezeichnet.

Anhand eines einfachen Beispiels lässt sich die Funktionalität der Fuzzy Logik am besten erklären. Basierend auf der Steigung des Untergrunds soll der Neigungswinkel eines Agents angepasst werden. Als Input-Variable dient demnach die Steigung des Untergrunds in Grad. Diese Variable trägt zu jedem Zeitpunkt einen absoluten Wert der sich zwischen -180 und +180 Grad bewegt. Um diesen Wert in verschiedene Unschärfewerte zu verwandeln, werden Unschärfevariablen mit den Variablennamen *up*, *down* und *straight* eingeführt. Als Outputs werden in diesem Fall die Variablen *upright*, *lean_forward*, und *lean_backward* deklariert.

Input: Steigung in Grad
Fuzzy Input: *straight, up*, *down*
Fuzzy Output: *upright*, *lean_forward*, *lean_backward*

In weiterer Folge können aufgrund der deklarierten Variablen Regeln für den Neigungswinkel des Agents definiert werden.

If straigt then upright
if up then lean_forward
if down then lean_backward

Das Ergebnis jeder dieser Regeln ist ein Wahrscheinlichkeitswert, der sich fließend zwischen 0 und 1 bewegt und den Neigungswinkel des Agents beeinflusst.

Die Unschärfewerte werden durch eine Membership-Funktion definiert, die in der Regel aus einer einfachen Kurve besteht. In den meisten Fällen werden Z-, Lambda-, Pi- oder S-Kurven verwendet um die Abhängigkeit des Unschärfewerts zum Zahlenwert des Inputs zu definieren (siehe Abbildung 100). Über den Einsatz dieser Membership-Kurven wird aus einem absoluten Wert ein Wahrscheinlichkeitswert gebildet, der sich zwischen 0 und 1 bewegt. Durch diesen Vorgang können im Brain des Agents absolute Input-Werte auf eine unscharfe Ebene transportiert werden.

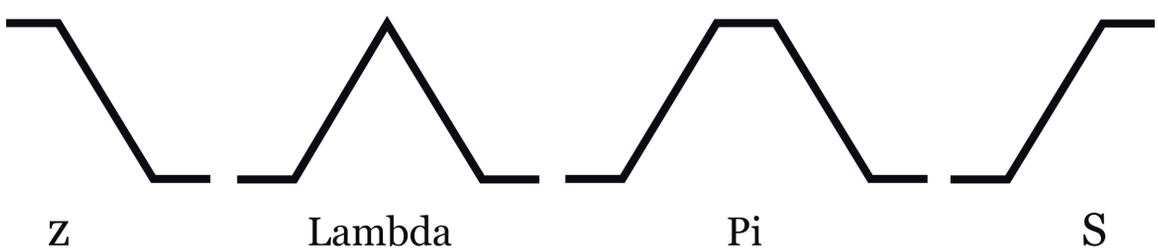


Abbildung 100: Membership Funktionen (vgl. Massive Manual 2008, fuzzy_logic.html)

10.8.3 Brain Nodes

Die Modellierung des Verhaltens der Agents erfolgt gänzlich auf der Basis von Nodes. Durch die hierarchische Verkettung von einzelnen Nodes können komplexe „Gehirne“ entstehen. Folgende Node Arten stehen im Brain Editor in Massive Prime zur Verfügung.

- **Input Node**

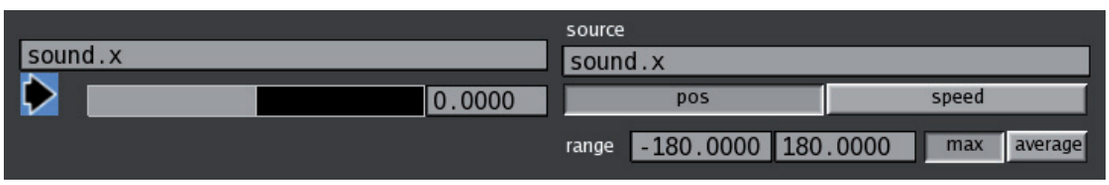


Abbildung 101: Input Node

Input Nodes beinhalten absolute Zahlenwerte in Form von Floating Point-Werten. Der Wert dieser Nodes kann einen händisch definierten Zahlenwert enthalten oder über das Source-Feld auf einen Animationskanal des Agents referenzieren. Diese Animationskanäle bilden das Bindeglied zwischen dem Brain und dem Body eines Agents und beinhalten Werte zur aktuellen Rotation und Position der Segmente sowie Informationen zu den Sinneswahrnehmungen und Aktionen der Agents. Eine komplette Liste der verfügbaren Kanäle ist der Diplomarbeit in Anhang beigelegt.

Durch die Definition von Position oder Speed in Kombination mit Animationskanälen beinhaltet der Input Node den aktuellen Wert des Animationskanals oder die Geschwindigkeit, mit der sich der Wert des Kanals ändert. Über die Range können des Weiteren Minimum und Maximum der erwarteten Input-Zahlenwerte definiert werden. Diese Werte beeinflussen in weiterer Folge auch das Spektrum der verknüpften Fuzz Nodes.

- **Timer Node**

Abbildung 102: Timer Node

Der Timer Node ist eine spezielle Art des Input Nodes. Über diesen Node kann ein Zahlenwert zur Verfügung gestellt werden, der sich linear über die Zeit ändert. Wird ein Timer Node ausgelöst, erhöht sich der Wert des Nodes in jedem Frame um die angegebene Rate bis der Range-Wert erreicht ist.

Über einen Input Node kann der Timer nach dem Erreichen des Maximalwertes erneut aktiviert werden. Steigt der Wert des Input Nodes über 0.5 wird der inkrementelle Vorgang des Timers von Neuem ausgelöst.

- **Noise Node**

Abbildung 103: Noise Node

Über den Noise Node werden über eine Random-Funktion zufällige Werte generiert, die sich fließend zwischen 0 und 1 bewegen. Diese Werte können in weiterer Folge als Input herangezogen und im Gehirn weiter verarbeitet werden um das Verhalten der Agents zu streuen. Die Frequenz, mit der sich die Werte eines Noise Nodes ändern, kann dabei über den Rate Parameter angepasst werden.

- **Fuzz Node**

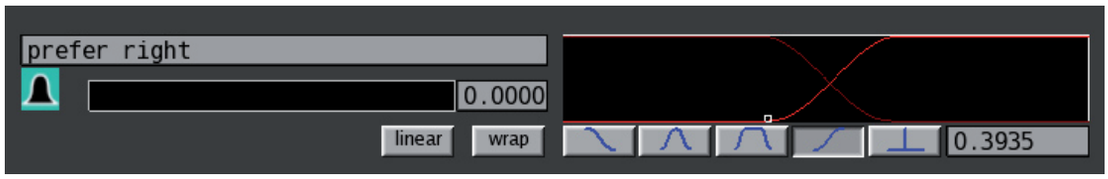


Abbildung 104: Fuzz Node

Über Fuzz Nodes werden absolute Zahlenwerte in Unschärfewerte umgewandelt. Durch die Verknüpfung von beliebig vielen Fuzz Nodes an einen Input Node können mehrere Unschärfewerte aus einem Input generiert werden. Jeder Fuzz Node beinhaltet eine parametrisierbare Membership-Kurve, welche die Abhängigkeit des Unschärfewerts zum Input definiert. Das Spektrum wird über den Minimal- und Maximalwert im Input Node gesetzt.

- **Rule Nodes**

AND und OR Nodes zählen zu den Rule Nodes und definieren Regeln auf der Basis der Unscharfen Logik. Durch die Verkettung von Rule Nodes und Fuzz Nodes werden Regeln in Bezug auf Input-Werte aufgestellt, um auf bestimmte Aktionen oder Sinne zu reagieren. Die resultierenden Unschärfewerte der Operation werden aus allen eingehenden Werten gebildet.

AND Node

Abbildung 105: AND Node

Für die Berechnung der AND Operation stehen zwei unterschiedliche Methoden zur Verfügung. Über den Einsatz der *Min* Operation wird der Output des AND Nodes aus dem geringsten Wert aller eingehenden Input-Werte gebildet. Die *Prod* Operation hingegen errechnet den resulierenden Wert aus dem Produkt aller eingehenden Node-Werte.

OR Node

Abbildung 106: OR Node

Die OR Operation verfügt über zwei verschiedene Berechnungsarten. Über den Einsatz der *Max* Operation wird aus allen eingehenden Inputs der höchste Wert übernommen und an den Output des Nodes weitergegeben. Das Ergebnis der *Sum* Operation wird hingegen aus der Summe aller eingehenden Werte errechnet.

Der berechnete Unschärfewert kann in weiterer Folge über den Weight Parameter gewichtet werden. NAND (negative AND) und NOR (negative OR) Operationen können mit Hilfe von negativen Verknüpfungen realisiert werden. Bei einer negativen Verknüpfung von Nodes wird der übergebene Wert vor der Weiterverarbeitung mit -1 multipliziert. Dies entspricht in der Fuzzy Logic einem NOT Operator.

- **Defuzz Node**

Abbildung 107: Defuzz Node

Defuzz Nodes werden verwendet um unscharfe Werte zurück in absolute Zahlenwerte zu wandeln. Wird als Input ein unscharfer Wert übergeben, wird dieser im Defuzz Node als Gewichtung des gesetzten maximalen Wertes eingesetzt. Der absolute Maximalwert im Defuzz Node entspricht demnach einem Unschärfewert mit der Gewichtung von 1. Fließen mehrere Defuzz Nodes in einem Output Node zusammen, wird der Absolutwert aus allen Defuzz Nodes entsprechend ihrer Gewichtung errechnet. Über die *Else*-Funktion kann einem Defuzz Node eine statische Gewichtung von 1 aufgezwungen werden.

- **Output Node**

Abbildung 108: Output Node

Output Nodes werden in der Regel in Kombination mit Defuzz Nodes verwendet, um das Verhalten der Agents zu beeinflussen. In diesem Fall wird der Wert des Output Nodes einem Animationskanal zugewiesen. So kann beispielsweise die Rotation eines Segments des Agent Rigs über einen Output Node im Brain gesteuert werden.

Der Wert des Output Nodes kann einerseits manuell gesetzt werden oder wird aus den Werten aller eingehenden Defuzz Nodes gebildet. Die Berechnung des Output-Wertes aus den Defuzz Nodes kann auf drei unterschiedliche Arten erfolgen. Mit Hilfe der *Average*-Funktion wird der absolute Zahlenwert aus dem gewichteten Durchschnitt aller eingehenden Defuzz Nodes gebildet. Die *Max*-Funktion übernimmt den Wert des eingehenden Defuzz Nodes mit der höchsten Gewichtung. Dem Output wird in weiterer Folge der Maximalwert des Defuzz Nodes mit der höchsten Gewichtung zugewiesen. Die Berechnung des Output-Werts mit der *Blend*-Methode verhält sich ähnlich wie die *Max*-Funktion. Auch hier wird der Output-Wert aus dem Defuzz Node mit der höchsten Gewichtung übernommen. Anstatt jedoch den Maximalwert direkt zu übernehmen, wird dieser vor der Zuweisung anhand des aktuellen Defuzz Nodes gewichtet.

Der errechnete Wert des Output Nodes an Kanäle oder Segmente des Agents kann als absoluter Wert oder als Geschwindigkeit übergeben werden. Über die Methoden *Pos* und *Speed* wird festgelegt, wie der Wert des Output Nodes interpretiert wird. Handelt es sich beispielsweise um die Rotation ry eines Segments wird der Wert 60 mit der Output-Methode *Pos* als absoluter Wert auf die Rotation des Segments übertragen. Das betroffene Segment wird demnach um 60 Grad um die y-Achse gedreht. Wird der Wert jedoch mit der

Speed-Methode weitergegeben, wird dieser als Geschwindigkeit interpretiert. Das Segment wird folglich in jedem Frame um 60 Grad, ausgehend von der aktuellen Rotation, weiter gedreht.

Durch die Verknüpfung dieser Nodes entsteht ein hierarchisches Brain-Netzwerk, welches das Verhalten der Crowd Agents abbildet. Dieses System erreicht oft eine Komplexität, die es erschwert den Überblick über alle Funktionen und Abhängigkeiten zu behalten. Einzelne Teile des Netzwerks können aus diesem Grund in Module zusammengefasst werden. Jedes Modul beinhaltet einen Teil des Node-Netzwerks mit definierbaren Input und Output Nodes. Diese Gliederung erleichtert die Übersicht und die Arbeit im Brain Editor.

10.8.4 Brain-Module

Im folgenden Abschnitt der Arbeit wird gezielt auf die Funktionen der Brain-Module eingegangen. Die genaue Beschreibung der Brain-Netzwerke in ihrer Gesamtheit würde den Rahmen dieser Arbeit bei weitem sprengen. Aus diesem Grund werden in den nächsten Kapiteln ausgewählte, beispielhafte Module aufgegriffen und erklärt. Diese Module sind Teile der im Zuge der Fallstudie erstellten Brain-Netzwerke zur Steuerung der Crowd Agents.

Im Fall der Bass Bot und Treble Bot Agents werden Brain-Module zur Steuerung des Motion Trees verwendet, um Aktionen und Animationen einzuleiten. Die Bewegungen des Banker Bots werden hingegen direkt über Output Nodes gesteuert, da dieser Agent über keinerlei importierte Animationsdaten verfügt.

10.8.4.1 Terrain Adaption

Das Modul in der Abbildung 109 (S. 207) ist Teil des Brain Setups des Bass Bot Agents. Dieses Modul steuert die Fuß- und Beinstellung des Agents und gleicht diese an die Höhe und den Winkel des Terrains unter dem Agent an. Die Repositionierung der *Toe*-Segmente in der Bewegung der Agents setzt Berechnungen der Inversen Kinematik anhand der erstellten RC und IK Curves voraus (siehe Kapitel 10.6.2.3 Kinematics, S. 186).

Im ersten Teil des Moduls wird der gesamte Körper des Agents durch die Translation des Root Segments auf der y-Achse an die Höhe des Terrains angeglichen. Als Ausgangspunkt dient ein Input Node mit dem Agent-Kanal *ground* als Eingang. Dieser Kanal beliefert den Input Node mit dem Abstand des Agents auf der y-Achse zum nächsten Polygon der Terrain-Geometrie. Dieser absolute Wert wird in weiterer Folge durch zwei Fuzz Nodes in die Unschärfevariablen *high* und *low* umgewandelt. Die Membership Curve dieser Fuzz Nodes beschreibt einen linearen Anstieg beziehungsweise Abfall der Werte von 0 bis 4 beziehungsweise von 0 bis -4. Befindet sich das Terrain also mindestens vier Einheiten unter dem Crowd Agent erhält die Unschärfevariable *high* den Wert 1, während die Variable

low einen Unschärfewert von 0 enthält. Die beiden Unschärfevariablen werden im weiteren Verlauf des Moduls über den Einsatz von Defuzz Nodes erneut in absolute Werte zurückverwandelt. Der Defuzz Node *up* trägt einen Maximalwert von 100 während der Node *down* einen Wert von -100 enthält. Diese Werte fließen in einem Output Node zusammen, welcher über die Zuweisung des Agent-Kanals *[ty]:offset* die Translation des Agents auf der y-Achse initiiert. Der Output Node muss jedoch in diesem Fall auch über den alternativen Defuzz Node *straight* verfügen, welcher einen Wert von 0 trägt und als Else Node deklariert ist. Sind also *up* und *down* nicht aktiv, was einen Input-Wert von 0 bedeutet, wird der Wert des *straight* Nodes übernommen und der Agent bleibt unverändert. Ohne dem Einsatz des Else Nodes würde lediglich die positive oder negative Translation des Agents ermöglicht, eine Translation von 0, also keine Translation, jedoch nicht berücksichtigt.

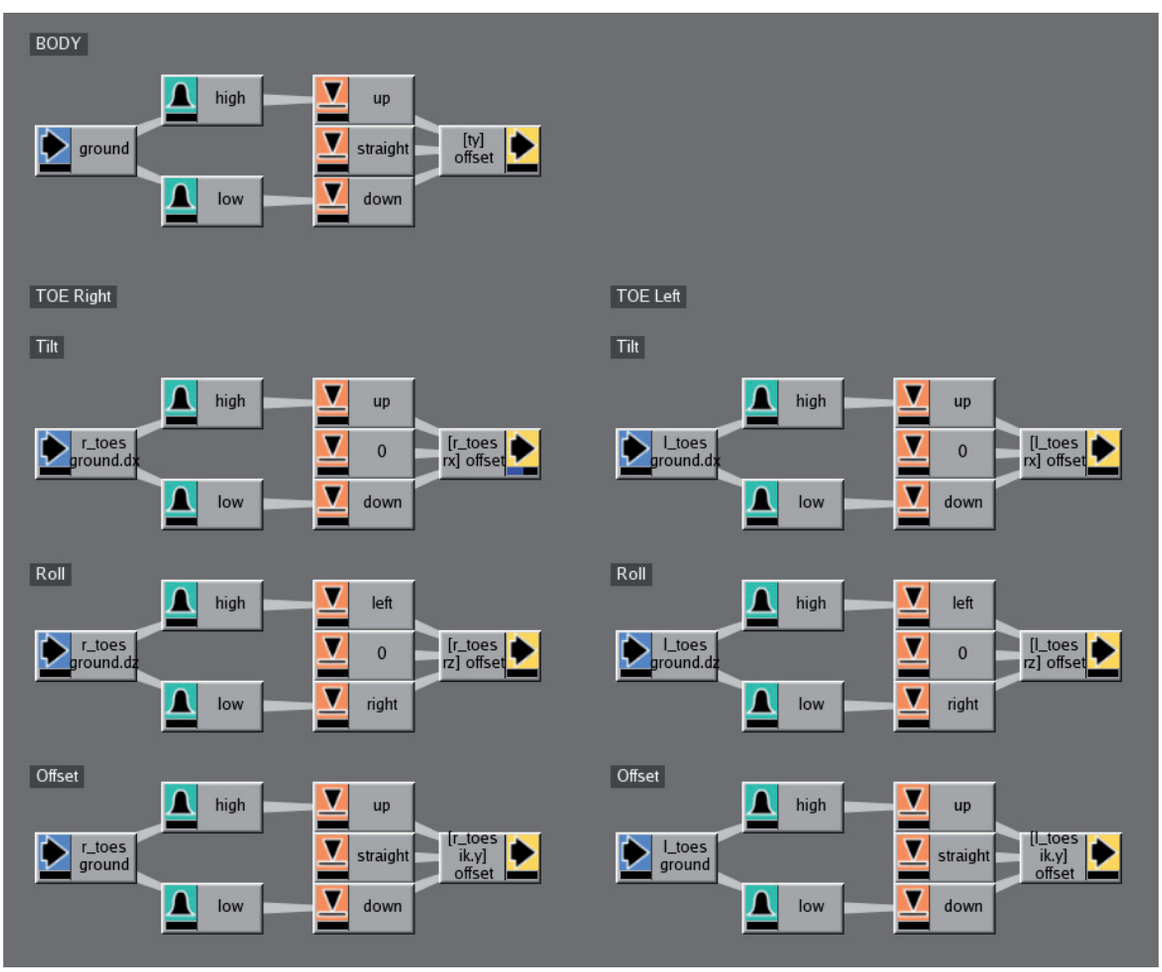


Abbildung 109: Brain-Modul zur Terrain Adaption

Dieses Modul beschreibt eine grundlegende Node-Hierarchie, wie sie häufig im Brain der Crowd Agents verwendet wird. Zur Anpassung der Beine werden ähnliche Module verwendet, um die Translation der *Toe*-Segmente auf der y-Achse an die Höhe des Terrains anzugleichen. Gleichermaßen kommen diese Module zum Einsatz um die Rotation der *Toe*-Segmente an den Untergrund anzupassen.

10.8.4.2 Kollisionsvermeidung über Soundemission

In dieser Fallstudie wurde die Kollisionsvermeidung der Crowd Agents über die Emission von Sound Signalen realisiert (siehe Kapitel 10.7 Sinnesorgane, S. 194). Die Amplitude sowie die Frequenz des Signals werden im Brain des Agents über Output Nodes kontrolliert. Werden diese Signale von einem Crowd Agent empfangen, können Maßnahmen zur Kollisionsvermeidung im Brain des betroffenen Agents eingeleitet werden (siehe Abbildung 110).

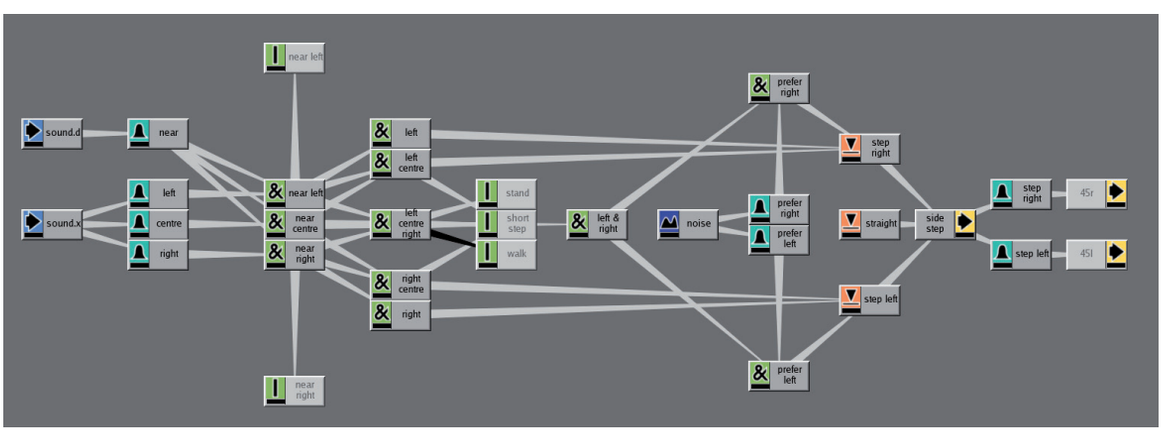


Abbildung 110: Brain-Modul zur Kollisionsvermeidung der Bass Bot Agents

Als Input dient der Agent-Kanal *sound.d*, welcher die Amplitude der empfangenen Signale beinhaltet, sowie der Kanal *sound.x*, der Informationen über die Koordinaten des empfangenen Signals bereitstellt. Die Richungsinformation wird über Fuzz Nodes in die Unschärfevariablen *left*, *center* und *right* untergliedert und mit der Amplitude des Signals gekoppelt. Somit erfolgt eine Unterscheidung zwischen dem Ursprung und der Amplitude des empfangenen Signals. Wird beispielsweise ein „lautes“ Signal von Links wahrgenommen, leitet das Modul eine Seitwärtsbewegung nach Rechts ein. Des Weiteren kontrolliert das Modul die

Schrittlänge des Agents durch die Variable *short step*. Der Wert dieser Variablen definiert den Skalierungsfaktor der IK Kurven und somit die Skalierung der Schrittlänge des Agents. Wird ein Signal mit hoher Amplitude direkt vor dem Agent wahrgenommen, wird die Schrittlänge des Agents verringert, um eine Kollision zu vermeiden. In diesem Fall muss über einen Entscheidungsprozess entschieden werden, in welche Richtung dem Objekt ausgewichen wird. Diese Entscheidung wird über einen Noise Node gesteuert.

10.8.4.3 Agent Navigation mit Lanes

Das Brain-Modul in der Abbildung 111 ist zuständig für die Navigation der Banker Bot Agents anhand des Lane-Netzwerks im Zentrum der Stadt.

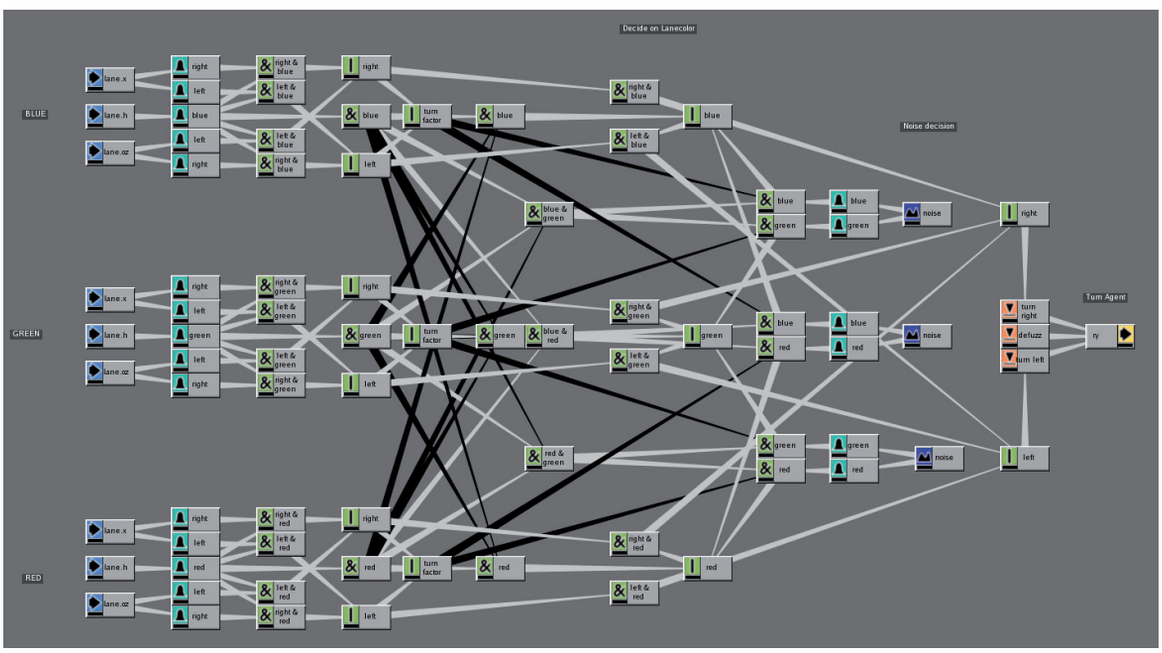


Abbildung 111: Brain-Modul zur Navigation der Banker Bot Agents auf Basis von Lanes

Der Output dieses Moduls definiert die Drehung *ry* des Agents. Als Input dienen diverse Lane-Kanäle für die Position des Agents auf der Lane, sowie die Farbinformation der Lane, auf welcher sich der Agent gerade bewegt. Der Agent-Kanal *lane.x* beschreibt die aktuelle Position des Agents auf der Lane, der Kanal *lane.ox* enthält den relativen Winkel zwischen Agent und Lane in Grad

und der Kanal *lane.h* enthält Informationen über den Farbwert der aktuellen Lane. Durch die Interpretation der Informationen über Position und Winkel des Agents relativ zur Lane kann die Drehung des Agents berechnet werden, die erforderlich ist, um der aktuellen Lane zu folgen. Überschneiden sich mehrere Fahrbahnen der selben Farbe, kann jedoch keine Differenzierung der einzelnen Richtungsinformationen erfolgen. Bei der Kreuzung einzelner Lanes werden aus diesem Grund drei unterschiedliche Farbcodes - rot, grün und blau - verwendet, die eine Unterscheidung der Fahrbahnen im Brain des Agents zu ermöglichen. Werden auf diese Art und Weise zwei oder mehr mögliche Lanes als richtungsgebende Fahrbahnen wahrgenommen, muss sich der Agent für eine dieser Fahrbahnen entscheiden. Dieser Entscheidungsprozess wird über Noise Nodes gesteuert. Befinden sich also beispielsweise eine rote und eine grüne Lane unter dem Agent, entscheidet der Noise Node zwischen rot und grün. Die Richtungsinformationen der präferierten Lane werden in weiterer Folge an den Output Node weitergegeben, um den Agent in die richtige Richtung zu lenken.

Bei großen Kreuzungen oder Unterführungen kommt es jedoch zwangsweise zu unerwünschten Überschneidungen der Lanes. Um den unerwünschten Spurwechsel bei der rechtwinkeligen Überschneidung von Lanes abzufangen, wird im Brain des Agents der Differenzwinkel errechnet, der bei einem möglichen Spurwechsel erfolgt. Dieser Winkel – im Modul als *turning factor* bezeichnet – wird in den Entscheidungsprozess zur Wahl der präferierten Fahrbahn mit eingebunden. Somit kann der Spurwechsel auf scharfen Kreuzungen verhindert werden.

10.8.4.4 Car Wheel Expressions

Das Verhalten der Räder des Banker Bot Agents wurde in der Entwicklungsphase des Agents über den Einsatz von Rigid Body Dynamics getestet. Da die Kalkulation der Dynamics jedoch mit hohem Rechenaufwand verbunden ist, wurde als Alternative die Berechnung der Rotation mit Hilfe von Expressions verwendet. Somit lässt sich die Rotation eines Rades in einer einfachen, direkten Verkettung eines Inputs mit einem Output Node realisieren. Die Rotation rx des rechten Rades des Banker Bot Agents wird mit Hilfe folgender Formel errechnet.

$$rx = \left(\frac{\left(\left(\frac{ry}{360} \right) * 2 * L * \pi \right) - tz}{2 * r * \pi} \right) * 360$$

Die Rotation rx des Rades resultiert demnach aus der lokalen Position des Agents ry, dem Achsabstand L, der lokalen Drehung des Agents tz und dem Radius des Rades r. Diese Formel wird im Input Node angewendet und ergibt die lokale Rotation des Rades, welche direkt an den Output Node übergeben wird und das betreffende Segment über den Kanal *l_wheel:rx* entsprechend rotiert. Da der vom Input Node übergebe Wert die lokale Rotation in Grad pro Frame, also eine zeitbezogene Rotationsänderung beschreibt, muss die Output Methode des Output Nodes auf *Speed* gesetzt werden.

10.9 Individualisierung von Crowd Agents

Die möglichst realistische Darstellung von Massen ist in den meisten Fällen das oberste Ziel einer Crowd Simulation. Die Grundlage einer realistischen Massensimulation stellt die Individualisierung der einzelnen Akteure einer Gruppe dar. Diese Individualisierung kann durch Variationen an Geometrien, Texturen oder Bewegungen der Crowd Agents erzielt werden. In Massive Prime wird die Diversifikation der Agents über den Einsatz von Agent-Variablen realisiert.

Bei der Deklaration von Agent-Variablen wird der Variablenname, der Wertebereich und der Defaultwert der Variable definiert (siehe Abbildung 112). Bei der Platzierung der Agents in der Szene wird jeder Instanz des Crowd Agents ein zufälliger Wert im Wertebereich dieser Agent-Variable zugewiesen. Eine beliebige Anzahl von Agent-Variablen kann auf diese Weise deklariert werden, um die Darstellung oder das Verhalten der Agents zu beeinflussen.

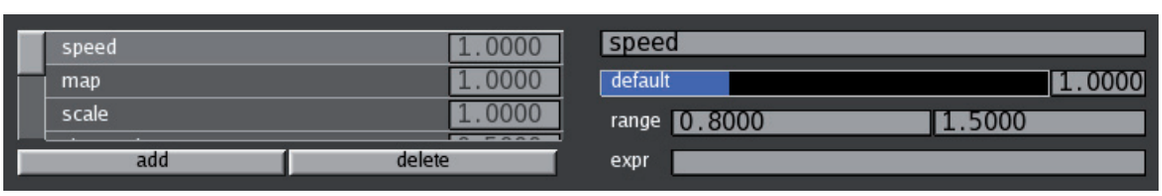


Abbildung 112: Agent-Variablen

- **Geometrie**
  Die Individualisierung der Agent Geometrie wird über den Einsatz von Option Nodes auf der Body Page des Agents realisiert. Der Option Node fungiert in diesem Fall als Platzhalter für mögliche Geometrien und kann wie ein Geometrie Node mit Segmenten des Rigs verknüpft werden. Als Input dienen dem Option Node eine beliebige Anzahl von möglichen Geometrien in Form von Geometrie Nodes. Über eine Agent-Variable wird bei der Instanzierung des Agents eine dieser untergeordneten Geometrie Nodes adressiert und dem Option Node zugewiesen. Auf diese Weise können verschiedenste Geometrien ein und demselben Agent Rig zugewiesen werden, um einzelnen Agents in der Masse ein individuelles Erscheinungsbild zu verleihen.

- **Material**
  Die Anpassung von Shadern und Texturen über Agent-Variablen ist in der Regel einfach zu bewerkstelligen und liefert gute Ergebnisse bei der Individualisierung einzelner Crowd Agents. Ambient-, Diffuse- und Specular-Kanäle verfügen über die direkte Zuweisung von Agent-Variablen, um die Farbe des Materials zu beeinflussen. Des Weiteren kann jeder einzelne Parameter eines Shaders mit den Werten einer Agent-Variable gespeist werden.

  Die Individualisierung von Texturen wird über den Dateinamen der Textur realisiert. Eine Agent-Variable kann im Verweis auf die Texturdatei in einfachen Anführungszeichen in den Dateinamen integriert werden, um verschiedene Texturen anzusprechen. Somit kann beispielsweise die Agent-Variable *lack* mit einem Wertebereich von 1 bis 2 über den Verweis auf die Datei *Lacktextur_'lack'.tif* die Integration der Textur unter dem Dateinamen *Lacktextur_1.tif* oder *Lacktextur_2.tif* zur Folge haben.

- **Motion**
  Die Differenzierung einzelner Crowd Agents über die Individualisierung von Bewegungen und Bewegungsabfolgen ist ein Kernpunkt dieser Fallstudie. Da jede Instanz eines Crowd Agents über dieselben Animationsdaten verfügt, ist eine Differenzierung einzelner Agents anhand der Nutzung dieser Animationen essentiell für die Darstellung realistischer Massensimulationen.

In der Fallstudie wird dies durch den Einsatz von Motion Blending realisiert. Treble Bot sowie Bass Bot Agents verfügen jeweils über zwei individuelle Walk Cycle-Animationen, welche durch Motion Blending zu einer individuellen Gehbewegung gemischt werden. Das Mischverhältnis dieser Blends wird über den Wert einer Agent-Variable im Action Node gesteuert. Somit erhält jeder Crowd Agents in der Szene eine individuelle Gehbewegung.

Eine weitere Möglichkeit Instanzen eines Agents über Bewegung zu differenzieren, stellt die Variation der Abspielgeschwindigkeit der Animationsdaten dar. Die Steuerung der Abspielgeschwindigkeit erfolgt im Brain des Agents über den Agent-Kanal *rate*. So kann die Geschwindigkeit einzelner Aktionen im Motion Tree über einen Output Node mit dem Kanal *action:rate* angepasst werden. Soll die Abspielgeschwindigkeit aller Bewegungen eines Agents verändert werden wird die Geschwindigkeit über den Wildcard Operator **:rate* gesetzt. Als Input dient erneut eine Agent-Variable, deren Wert über den Variablennamen an den Input Node übergeben und an den Output Node weitergeleitet wird.

Des Weiteren besteht die Möglichkeit, Animationskurven auf einzelnen Segmenten zu skalieren, um Bewegungsabläufe einzelner Agents zu individualisieren. In der Fallstudie wird die Rotation des Root Segments des Bass Bot Agents über einen Noise Node gesteuert, um die Auslenkung des Körpers bei der Gewichtsverlagerung während der Gehphase des Agents über die Zeit zu verändern.

Zudem wird eine Vielzahl an Segmenten direkt über zufällig generierte Werte aus Noise Nodes gesteuert und beeinflusst. Bewegungen von Kopf, Armen und dem Oberkörper der Agents werden auf diese Weise gesteuert und individualisiert.

- **Skalierung**

  Die Skalierung von Agents ist eine weitere Möglichkeit Individuen in einer Masse zu differenzieren. Die Definition der Skalierung erfolgt auf der Body Page im Vary Tab des Agents (siehe Abbildung 113). Der gleichförmigen Skalierung des Agents kann hier eine Agent-Variable zugewiesen werden.

Abbildung 113: Agent Skalierung im Vary Tab

Des Weiteren ist die Angabe einer Breite möglich, um Instanzen des Agents durch horizontale Skalierung dünner oder dicker erscheinen zu lassen. Diese ungleichförmige Skalierung ist jedoch auf Geometrien beschränkt, welche mit Hilfe von Soft Skinning an das Agent Rig gebunden werden. Da die Geometrien der Bot Agents in der Fallstudie zum Großteil über Rigid Binds mit dem Rig verbunden sind, kann diese Funktion nicht genutzt werden.

10.10 Setup der Szene

Nach der Fertigstellung aller benötigter Crowd Agents erfolgt die Initialisierung der geplanten Massenszenen. Dies beinhaltet die szenenspezifische Erstellung und Platzierung von Massen, bestehend aus den umgesetzten Crowd Agents, die Durchführung von Simulationsdurchläufen und das Rendering der Simulation.

10.10.1 Platzierung der Agents

Wird ein Crowd Agent in Form einer Agent-Datei (.CDL) in eine Massive-Szene integriert, wird lediglich ein Crowd Agent im Ursprung der Szene erzeugt. Um aus einer Agent-Datei mehrere Crowd Agents zu erzeugen, werden Instanzen des ursprünglichen Agents erzeugt. Diese Instanzierung der Agents erfolgt über die Erstellung von Positionsgebern (Locators) mit Hilfe des Massive Place Tools (siehe Abbildung 114). Die Place-Funktion generiert Instanzen des Agents in jedem erstellten Locator-Punkt. Der ursprüngliche Crowd Agent im Ursprung der Szene wird durch diesen Vorgang entfernt.

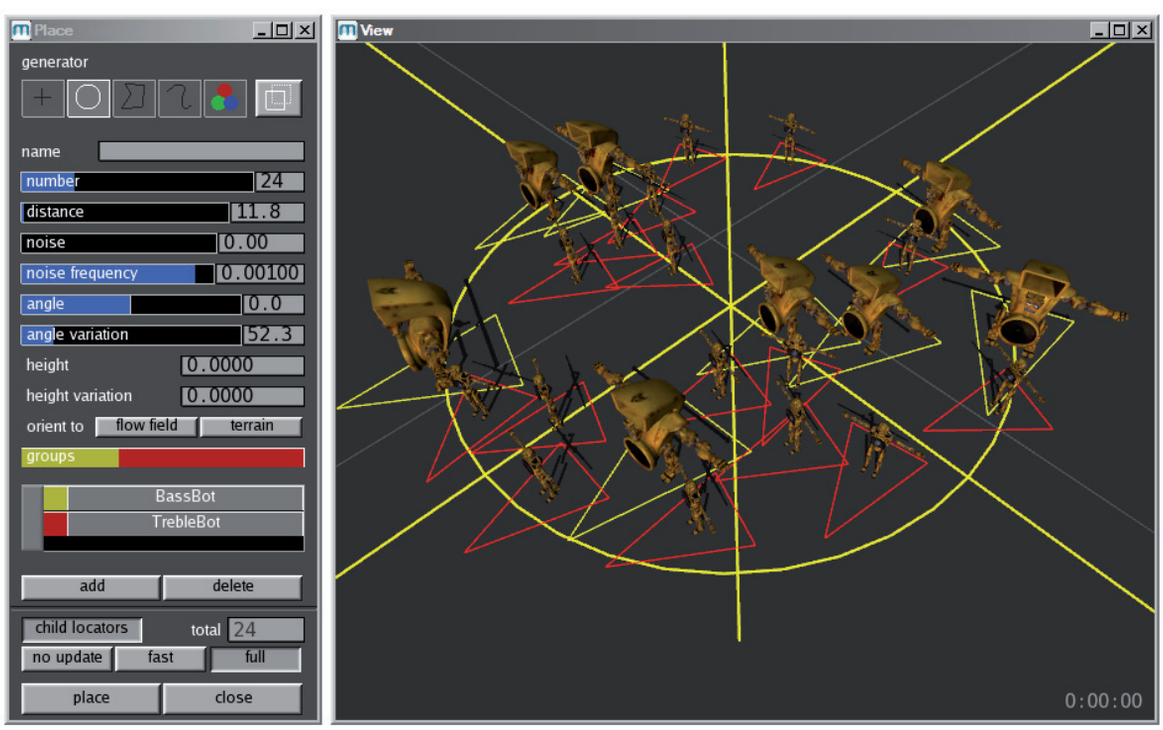


Abbildung 114: Platzierung von Agents

Die Erstellung der Locator-Punkte kann im Place Tool über verschiedene Funktionen erfolgen.

- **Point**
  Über die Point-Funktion können einzelne Locator-Punkte gezielt im Raum platziert werden.

- **Circle**
  Hier wird eine definierte Anzahl von Locator-Punkten innerhalb eines Kreises generiert. Der Mindestabstand zwischen den Positionsgebern sowie der Platzierungswinkel sind parametrisierbar. Um die Platzierung der Agents natürlicher zu verteilen, können Winkel und Positionen der Locator-Punkte über eine Random Funktion variiert werden.

- **Polygon**
  Die Platzierung von Agents über die Polygon-Funktion entspricht in den Grundzügen der Generierung von Locator-Punkten über die Circle-Funktion. Als Grundform für die Platzierung der Agents dient jedoch in diesem Fall ein Polygon mit einer beliebigen Anzahl an Eckpunkten, welche direkt im Viewport gesetzt werden.

- **Spline**
  Über die Spline-Funktion wird eine Kurve mit beliebig vielen Ankerpunkten erzeugt, die als Grundform für die Platzierung der Agents dient. Die Anzahl der generierten Locator-Punkte erfolgt entweder direkt über den Number Parameter oder resultiert aus der Definition der Anzahl an Spalten und Reihen. Die Locator-Punkte werden bei ihrer Generierung an den Tangenten der Kurve ausgerichtet.

- **Colour**
  Die Platzierung der Agents über die Colour-Funktion basiert auf der Farbinformation der Terrain-Textur. Auf den Flächen des Terrains, deren Farbwerte der im Place Tool definierten Farbe in Form eines RGB-Wertes entsprechen, werden Locator-Punkte für die Platzierung von Agents verteilt.

Werden mehrere Agent-Dateien in die Szene integriert, kann die Instanzierung der Agents in einem beliebigen Mischverhältnis erfolgen. Somit kann innerhalb einer Platzierungsfunktion definiert werden, für welche Agents Locator-Punkte erzeugt werden und in welchem Verhältnis sich die Anzahl der verschiedenen, generierten Agents zueinander verhält.

Eine weitere Möglichkeit Agents in der Szene zu platzieren, stellt die Spawning-Funktion dar. Im Gegensatz zur Platzierung der Agents über Locator-Punkte werden bei dieser Methode Instanzen der Crowd Agents während dem Simulationsdurchlauf erzeugt. Beim Spawning werden zwei unterschiedliche Crowd Agents benötigt: Spawner und Spawnee. Trägt eines der Segmente des Spawner Agents den Namen des Spawnee Agents kann durch die Aktivierung des Spawn Kanals im Brain des Spawner Agents eine Instanz des Spawnees aus diesem Segment generiert werden. Diese Methode ermöglicht die Emission von einfachen Crowd Agents, wie z.B. Pfeile und andere Wurfgeschoße bis hin zu komplexen humanoiden Charakteren, während der Laufzeit der Simulation. In der Fallstudie wurde ein einfacher Spawner Agent, bestehend aus einem Segment mit dem Titel „Banker Bot“, eingesetzt um Banker Bot Agents in zufälligen Zeitabständen aus dem GLOBOTEX Hauptgebäude zu emittieren.

10.10.2 Simulation

Um die Crowd Simulation zu berechnen, werden im Simulationsdurchlauf die Bewegungen jedes einzelnen Crowd Agents anhand der Animationsdaten und der Brain-Module kalkuliert. Bei der Simulation von großen Szenerien kann dieser Prozess ein hohes Ausmaß an Rechenleistung beanspruchen. Der Sim Dialog von Massive Prime bietet jedoch die Möglichkeit Simulationen in Dateien abzulegen, zu speichern und wiederzugeben (siehe Abbildung 115, S. 218). Das Speichern der Simulation kann in verschiedenen Dateiformaten erfolgen.

- Acclaim Motion Capture (.AMC)
  Dieses Format bildet die gesamte Crowd Simulation in einer Datei ab.

- Acclaim Motion Capture per Frame (.APF)
  Für jeden Frame der Simulation wird ein .APF File zur Beschreibung der Simulation erzeugt.

Die Simulationsdateien können in weiterer Folge als Input zugewiesen werden, um die gespeicherte Simulation wiederzugeben. Die Wiedergabe kostet zwar weniger Rechenleistung als die Kalkulation von Simulationen, lässt jedoch keine Änderung und Anpassung von Parametern im Brain der Agents zu. Aus diesem Grund wird im Produktions-Workflow in der Regel jede Szene simuliert, überprüft, verfeinert und erneut kalkuliert, bis das Ergebnis der Crowd Simulation den Anforderungen gerecht wird.

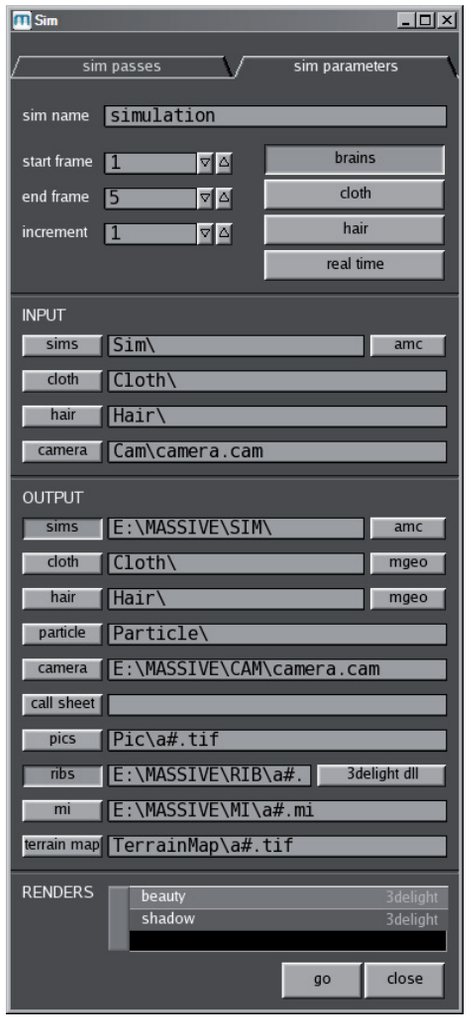


Abbildung 115: Sim Dialog

10.10.3 Rendering der Crowd Simulation

Der Render Workflow von Massive Prime basiert auf der Erstellung von Render Passes innerhalb einer Massive Szene. Die allgemeinen Parameter zum Rendern der Crowd Simulation werden im Render Dialog vorgenommen.

In erster Instanz werden die benötigten Render Passes angelegt. Jedem Render Pass wird ein Renderer zugewiesen. Massive Prime unterstützt zum Rendern der Simulation diverse Renderer, unter anderen RenderMan kompatible Renderer wie Pixar's PRMan oder 3Delight sowie Mental Ray Stand-Alone und den Massive openGL Renderer Velocity. Für jeden erstellten Render Pass können Shader und Shaderparameter in den Material Nodes der Agents sowie im Shader Tab des Terrains individuell angepasst werden. In weiterer Folge wird für jeden Rander Pass ein Render-Element angelegt. Diese Elemente enthalten renderspeziefische Parameter wie beispielsweise Auflösung, Bildformat und die Verknüpfung des Renderings mit einer Kamera (siehe Abbildung 116).

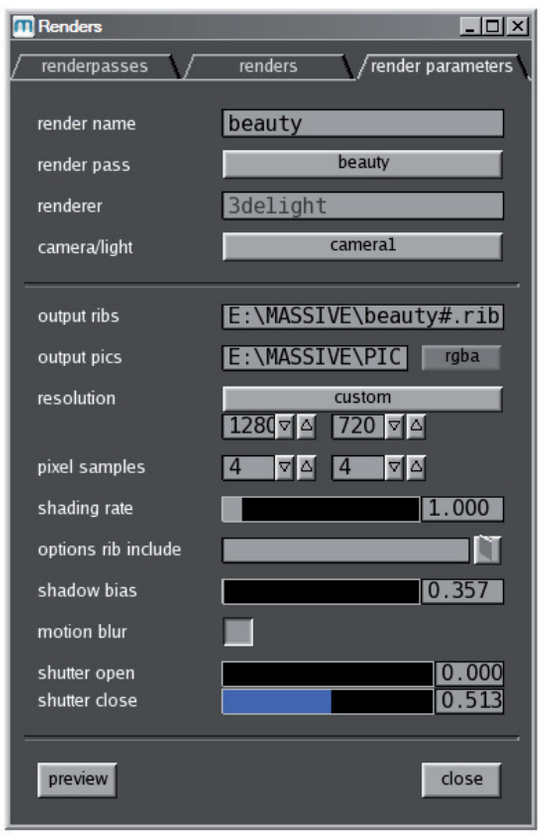


Abbildung 116: Render Parameters

Für die Berechnung der Bildsequenzen werden Render Dateien von Massive Prime erzeugt, welche die Beschreibung der gesamten Szene beinhalten. Die Daten werden im .RIB (Renderman Interface Bytestream) Format für RenderMan Renderer oder .MI (Mental Images) Format für Mental Ray Stand-Alone als Archive abgelegt. In Massive Prime wird zwischen drei verschiedenen Arten von Render Files unterschieden.

- **Main**
  Diese Datei beschreibt den wesentlichen Aufbau der Szene. Informationen zu Lichtern, Kameras und Renderparametern werden in diesem File abgelegt. Des Weiteren wird hier auf Agent- und Terrain-Daten referenziert. Diese Daten werden somit integriert und an den Renderer weitergeleitet.

- **Terrain**
  Das Terrain-Archiv enthält sämtliche Informationen über Geometrie und Shader des Terrains, die für den Rendervorgang benötigt werden. Diese Daten werden über das Main-Archiv an den Renderer übergeben.

- **Agent**
  Das Agent-Archiv enthält alle Informationen, die für den Rendervorgang der Agents in einer Szene benötigt werden. Diese Dateien beinhalten jedoch keinerlei Daten zur Geometrie und den Shadern der jeweiligen Agents. Es wird lediglich auf die Agent-Datei (.CDL), in welcher alle benötigten Informationen des jeweiligen Crowd Agents abgelegt sind, und die Sim-Datei verwiesen, welche die Positions- und Rotationsdaten der Agents enthält. Das Massive Render Plugin lädt diese Daten ein, interpretiert die Informationen aus der Agent-Datei und der Sim-Datei und übergibt die kalkulierten Geometrien und Shader zurück an den Renderer, wenn diese benötigt werden (siehe Abbildung 117, S. 221). Dieses Konzept minimiert den Verbrauch an Arbeitsspeicher und ermöglicht somit die effiziente Berechnung großer Szenerien mit einer hohen Anzahl an Agents.

In dieser Fallstudie wurde der RenderMan Compliant 3Delight zur Berechnung der Bildsequenzen eingesetzt. Die Render Files werden, wie die Sim Files, über den Sim-Dialog erzeugt und im .RIB Format im Dateisystem abgelegt. Bei der Generierung der Render Files ist die Angabe eines absoluten Dateipfades zur Integration der Daten ratsam. Werden relative Pfadangaben verwendet, können Fehler bei Verweisen innerhalb der Renderpipeline auftreten. Des Weiteren ist darauf zu achten, dass Dateipfade keine Sonder- oder Leerzeichen enthalten. Diese führen zu einem Fehler bei der Interpretation der Pfadangaben im Render

Plugin. Auf einem Windows System sollten daher die erzeugten Render Files nicht relativ zum Massive Intsallationsverzeichnis abgelegt werden, da der Ordner Program Files vom Massive Render Plugin fehlerhaft interpretiert wird und zu einem Abbruch des Renderings führt.

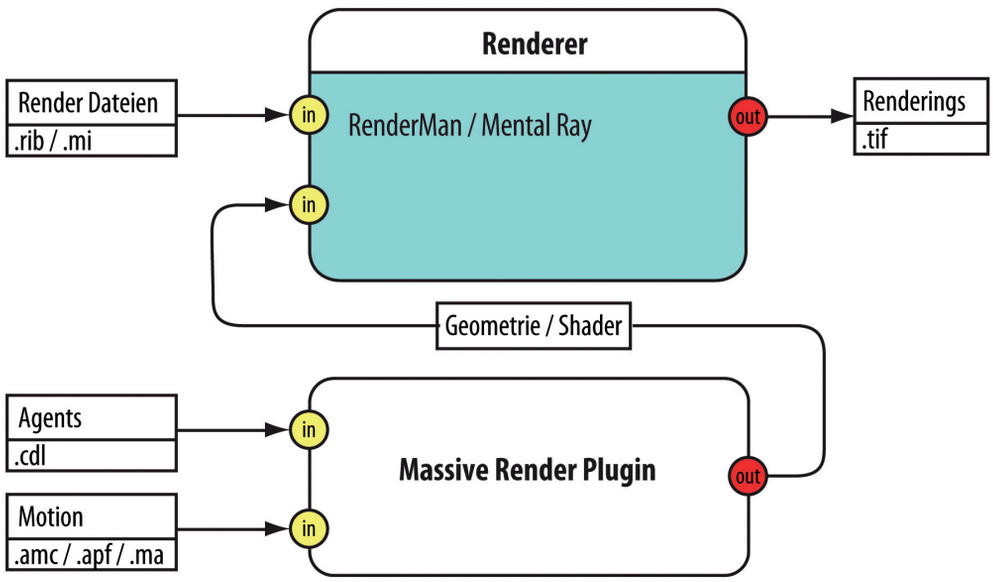


Abbildung 117: Massive Render Pipeline

11 Aufbau der Szenen

Um einen einheitlichen Look der einzelnen Einstellungen zu gewährleisten, erfolgt der Aufbau der Szenen und die Aufbereitung der Elemente für das finale Rendering in Autodesk Softimage. Im Zuge dieser Projektphase werden alle benötigten Daten, wie beispielsweise Assets aus 3ds Max oder Animationsdaten aus MotionBuilder bzw. Maya, in die Szenen integriert.

11.1 Integration der Daten

Um Änderungen möglichst effizient umsetzen zu können werden einzelne Elemente als „referenced Models“ integriert. Dies trifft sowohl für die urbane Umgebung zu, als auch für handlungstragende animierte Characters im Vordergrund der Einstellungen. Auch verschiedene weitere Assets, wie Solarpanels, Müllcontainer, Ampeln etc. werden auf diese Weise importiert. Dieser Zugang ermöglicht ein simultanes Arbeiten mehrerer Personen an einer Szene ohne sich gegenseitig zu behindern. Weiters kann an Details gearbeitet werden, sei es in den Bereichen der Character Animation, des Modelings oder auch der Shader-Kreation, ohne sich in komplexen großen Szenen zurechtfinden zu müssen. Wenn alle Einzelposten für eine Einstellung in einer Szene bereit stehen, kann die Kamerafahrt erstellt werden.

11.2 Kamera Setup

Bei dem Setup der Kameras in Softimage kommt das Shakey Cam Addon zum Einsatz, um eine natürlichere Kamerafahrt zu generieren, die keinen generischen Look aufweist. Zu diesem Zweck werden parametrisierbare Ungenauigkeiten in die Kamerapfade eingerechnet, um eine menschliche Komponente in der Kameraführung zu simulieren. Pfaden folgende Nullen dienen hierbei als gute Hilfsobjekte für Kamera und Interest.

Um die Kamera jedoch in Massive Prime einsetzen zu können, muss diese über Maya als .MA importiert werden. Die Kommunikation zwischen Maya und Softimage ist dank Autodesk Crosswalk flexibel. Mit Hilfe dieses Plug-Ins kann die Kamerafahrt exakt an Maya als .XSI übergeben werden. Das Shakey Cam Setup ist jedoch aus verschiedenen, hierarchisch eingegliederten Nullen aufgebaut, was den direkten Export für Massive unmöglich macht, da in der Massensimulationssoftware weder Nullen noch ein Kamera Interest existieren. Um die Kamera aus Maya für Massive exportieren zu können, müssen die Keyframes der Kamerabewegung direkt auf die Kamera selbst übertragen werden. Für diesen Zweck muss in Maya die Kamera kopiert und aus den Parents gelöst werden. Weiters erhält die kopierte Kamera sowohl einen Position- wie auch Orientation-Constraint auf die importierte Kamera. Auf diese Weise führen beide Kameras nun dieselbe Bewegung aus, jedoch hängt die kopierte Kamera in keiner komplexen Hierarchie. Nun kann ein Bake-Simulation Prozess für die kopierte Kamera durchgeführt werden, wodurch für jeden Frame absolute Translations- und Rotationswerte erzeugt werden. Die neu erzeugte Kamera kann nun für Massive als .MA exportiert werden.

Im nächsten Schritt werden die Einstellungen entsprechend ausgeleuchtet und für den Renderprozess aufbereitet.

12 Rendering und Post Prouktion

Die Berechnung der finalen Bilder erfolgt über die Kombination unterschiedlicher Render Outputs. Im Zuge der Compositing Phase werden die Renderings aus Autodesk Softimage, Massive Prime und Vue zum fertigen Endprodukt vereint.

12.1 Rendering und Pass Output

Das Rendering der urbanen Szenerie, sowie der handlungstragenden Figuren im Vordergrund erfolgt in Autodesk Softimage. Um eine hohe Flexibilität im Compositing zu ermöglichen wird das Rendering in verschiedene Passes aufgeteilt. Aus Performancegründen werden die erstellten Massensimulationen nicht in die Szenerie integriert, sonder über das Massive Render Plugin gerendert. Da der Vue interne Renderer zu deutlich besseren Ergebnissen führt als das Rendering innerhalb von Softimage mit dem Plug-In Vue xStream, wird die Vegetation in mehreren Render Passes in der Standalone Version von Vue berechnet.

Folgende Passes werden gerendert:

- Beauty Pass
- Specular Pass
- Abient Occlusion Pass
- Alpha Mask für Vordergrund Akteure und Objekte
- Z-Depht Pass
- Velocity Pass
- Reflection Pass
- Shadow Pass

12.2 Compositing

Das Composting erfolgt mit Hilfe von Adobe After Effects. Alle Rendersequenzen der unterschiedlichen Passes werden in diesem Softwarepaket zusammengefügt und mit einander Verknüpft, um den finalen Look des animierten Kurzfilms zu definieren.

13 Conclusio und Ausblick

Die Software CityEngine von Procedural Inc. hat sich im Zuge der Erarbeitung der Fallstudie als gute Lösung für die Generierung von urbanen Film Sets mit Hilfe prozeduraler Technikenerwiesen. Der prozedurale Ansatz der CityEngine ermöglicht es auf diese Art eine unglaubliche Menge an Gebäuden und Straßenzügen beliebigen Detailgrads zu erzeugen und erlaubt zusätzlich die sofortige Abänderung von Eigenschaften und Parametern der Modelle. Den Umgang mit der zu Grunde liegenden Formbeschreibungssprache CGA Shape Grammar zu erlernen stellte Anfangs eine gewisse Herausforderung dar, nach eindringlicher Beschäftigung mit der Materie war es jedoch auch ohne Vorwissen in Programmierungstechniken möglich die Funktionsweise und den Aufbau des Systems zu verstehen und anzuwenden.

Als problematisch stellte sich die Modellierung des Hauptverkehrsknotens heraus, dessen komplexe Formgebung und Proportionen nicht zufriedenstellend auf prozedurale Weise generiert werden konnten. Dieser Umstand machte es notwendig, dieses Straßenelement per Hand zu modellieren und texturieren, was einen erheblichen Mehraufwand an Arbeitszeit verursachte. Der Transfer der erzeugten Modelle von der CityEngine in die 3D-Software Softimage führte ebenfalls teilweise zu Problemen in Form von Systemabstürzen; die betroffene Export-Geometrie konnte jedoch letztlich mit Hilfe eines Workarounds - den Import der Objekte in Autodesk 3ds MAX gefolgt vom erneuten .OBJ-Export aus 3ds MAX - doch in die Zielsoftware Softimage importiert werden.

Die CityEngine verfügt in der aktuellen Version 2009.3 bereits über viele äußerst praktische Tools und Features, es könnten jedoch einige Arbeitsprozesse durch zukünftige Weiterentwicklungen erleichtert werden. Durch eine Optimierung der Erzeugung komplexer Straßenstrukturen wie etwa Unterführungen, Verkehrsknoten oder der nachträglichen Modifikationsmöglichkeiten der Shape-Struktur von Street Networks könnte der Arbeitsblauf bei vielen Projekten erheblich beschleunigt werden.

Das Design der 3D-Characters stellte einen weiteren zeitintensiven Prozess dar, da die Charaktere sowohl für die Anwendung als Crowd Agents in Massive Prime als auch für die hoch aufgelöste Darstellung in nahen Kameraeinstellungen geeignet sein mussten. Aufgrund des großen Zeitaufwandes bei der Fertigung der Modelle konnten von den ursprünglich angedachten Charakteren nur drei zur Gänze umgesetzt werden. Die Working Pipeline zur Realisierung weitere Modelle im Zuge der Fertigstellung des Kurzfilmprojektes konnte jedoch erfolgreich ermittelt werden.

Die Anforderung eine visuell heterogene Masse durch den Einsatz verschiedener Texturen, zahlreicher unterschiedlicher Modelle und austauschbarer Körperteilvarianten der Charaktere zu erzeugen, konnte ebenfalls aus Zeitgründen nicht vollständig erfüllt werden. Die nötigen Vorkehrungen für eine zukünftige Erweiterung der Masse unter Einbeziehung der genannten Maßnahmen konnten jedoch getroffen werden, so dass der fortschreitenden Heterogenisierung nichts mehr im Wege steht.

Beim Transfer von Geometriedaten zwischen den verschiedenen Softwarepaketen existieren durchaus noch Verbesserungsmöglichkeiten. So war es oftmals notwendig, den Umweg über Autodesk Maya zu nehmen, um die 3D-Geometrie der Characters problemlos zwischen Programmen weitergeben zu können.

Für die Erstellung von Bewegungsdaten für die Massensimulationssoftware Massive Prime hat sich Autodesk MotionBuilder als hilfreiches Tool herausgestellt, welches hervoragende Kommunikationseigenschaften aufweist, verschiedene Softwarepakete mit den nötigen Daten zu versorgen. Obwohl Massive Prime fast ausschließlich über Autodesk Maya kommuniziert, war es über die erstellte Pipeline möglich, alle Vorteile verschiedener Softwarepakete zur Gänze auszunutzen. Non-lineare Keyframe Animation, über Layer steuerbare Animation, prozedurale Animation und der Transfer von Motion Capture Daten boten eine Vielfalt an Animationstechniken, die es ermöglicht haben, variantenreiche Bewegungsabläufe zu erstellen. Erst der Einsatz verschiedener Techniken erlaubte es, die nötige Anzahl an Animationen für die Darstellung einer heterogenen Masse zu produzieren. Nicht jede Animationstechnik ist jedoch für die Erstellung aller gewünschten Bewegungsabläufe einsetzbar.

Das optische Tracking Verfahren per Hand hat sich als langwieriger und anstrengeder Prozess herausgestellt, der ein hohes Maß an Aufwand erfordert. Für handlungsspezifische, spezielle Aktionen ist diese Technik jedoch durchaus einzusetzen. Für weniger spezielle Aufgaben oder Standardaktionen bieten Motion Capture Libraries gute Ausgangsanimationen, die nach einem Cleaning Prozess an die jeweiligen Bedürfnisse angepasst werden können. Weiters dienen sie als gute Vorlage, um realistische Bewegungsabläufe rekonstruieren zu können. Für die Adaption von Bewegungsdaten unterschiedlicher Herkunft konnte mit Hilfe von MotionBuilder problemspezifisch immer der beste Manipulationsansatz (FK / IK) gewählt werden, um den für die jeweilige Animation gewünschten Effekt zu erzielen. Der prozedurale Workflow mit dem Character Animations Plug-In CAT hat sich bezüglich der Erstellung von Motion Loops wahrlich als „Timesaver“ etabliert.

Jede Animationstechnik hat spezielle Vor- beziehungsweise Nachteile. Wenn das Character Setup korrekt aufgebaut und flexibel für verschiedene Animationsansätze ausgelegt ist, gibt es in der Erstellung von Bewegungsdaten keinen richtigen oder falschen Weg. Erst die Kombination verschiedener Techniken ermöglicht einen flexiblen Ansatz für jede Problemlösung.

“Never get locked into only one way of solving a problem, expressing your creativity, or visually telling a story.“ (Kitagawa et al. 2008, S.126)

Obwohl sich die Kommunikationsfähigkeit zwischen unterschiedlichen Softwarepaketen, vor allem durch dein Einsatz von Autodesk Crosswalk, im Laufe der letzten Jahre deutlich verbessert hat, könnten wesentliche Arbeitsprozesse noch weiter optimiert werden. Der Einsatz verschiedener Rig-Strukturen in unterschiedlichen Programmen bewirkt einen Mehraufwand, der sich merklich auf die Effizienz im Produktionsablauf auswirkt. Einheitliche Rig Elemente würden für eine wesentlich flexiblere Arbeitsumgebung sorgen, wodurch die nötige Pipeline deutlich simpler angelegt werden könnte.

Der Umgang mit der Massensimulationssoftware Massive Prime erscheint auf den ersten Blick äußerst komplex. Erst nach der intensiven Auseinandersetzung mit der Software, sowie dem Handbuch, wird ein produktives und zielgerichtetes Arbeiten möglich. Der auf Nodes basierende programminterne Workflow ist gewöhnungsbedürftig, erweist sich nach einiger Zeit jedoch als überaus praktischer

und übersichtlicher Ansatz zur Umsetzung von hierarchischen Netzwerken. Diese visuelle Komponente erleichtert vor Allem die Konzeption von komplexen Brain Modulen zur Steuerung von Crowd Agents.

Zu Beginn der Fallstudie wurde mit der Massive Prime Version 3.0.3 gearbeitet. Sowohl auf Windows Vista, als auch auf Ubuntu führten einzelne Funktionen dieser Version zu Fehlermeldungen und Systemabstürzen. Mit der Version 3.5.5 für Windows wurde ein Großteil dieser Fehler behoben. Die Crossfade Funktion im Action Editor, zur Überblendung von Motion Loops führt jedoch auch in dieser Version zu einer Fehlermeldung, gefolgt von einem Programmabsturz. Aus diesem Grund muss die Aufbereitung und Überblendung von Animationszyklen extern in MotionBuilder erfolgen. Des weiteren erzwingt die Integration von Massive in eine zielgerichtete Produktionspipeline die Verwendung von Autodesk Maya, als Bindeglied zwischen Massive und weiteren 3D Softwarepaketen. Zwar wird der Import von Rigs und Animationsdaten als .XSI oder .FBX Daten unterstützt, führt jedoch in den meisten Fällen zu Fehlermeldungen oder mangelhaften Ergebnissen. Die Erweiterung der Kommunikationsfähigkeit und möglicher Schnnittstellen zu 3D Softwarepaketen von Drittanbietern würde die Handhabung der Software und die Eingliederung in verschiedenste Produktionspipelines vereinfachen. Seit Februar 2010 ist Massive Prime in der Version 4.0 auf dem Markt, welche einige Verbesserungen im Wokflow, Bugfixes und diverse neue Funktionen auf diesem Gebiet verspricht.

Obwohl Massive Prime erst vor wenigen Jahren im Zuge der “Herr der Ringe – Trilogie” entwickelt wurde, ist die Software im Bereich der Filmproduktion bereits heute marktführend in der Umsetzung von Massensimulationen. Nicht zuletzt auf Grund der einfachen Handhabung , dem effizienten Renderprozess von komplexen Szenerien und der Steuerung der Agents durch die Implementierung von künstlicher Intelligenz, bietet Massive Prime eine Vielzahl an Vorteilen gegenüber vergleichbaren Softwarelösungen.

Glossar

bake
Durch den bake Vorgang werden Translations- und Rotationsdaten eines, mit Hilfe von Operationen gesteurten 3D Objekts, auf absolute Keyframes reduziert.

Biped
Als Biped wird die Skelettstruktur (Rig) eines virtuellen menschlichen Characters bezeichnet.

Brain
Auf Fuzzy Logic basierendes Netzwerk aus Nodes zur Definition des Verhalten einzelner Crowd Agents, sowie gruppendynamischer Abläufe der Agents zueinander.

Capture Volumen
Das Capture Volumen beschreibt die räumlichen Ausmaße, die bei der Aufzeichnung von Bewegungen im Rahmen eines Motion Capture Verfahrens zur Gänze erfasst werden.

CCD (Carge coupled Device)
Lichtempfindlicher Bildsensor.

CGA (Computer Generated Architecture)
Computergenerierte Architektur.

CGI (Computer Generated Imagery)
Mit Hilfe von 3D Softwarelösungen generierte Bilder im Bereich der Filmproduktion.

Cluster
Definierte Gruppe von Polygonen eines 3D Objekts. In den meisten Fällen werden Cluster zur gezielten Überlagerung von Materialien eingesetzt.

Constraint
Unter einem Contraint wird die translations- und/oder rotationsgebende Verknüpfung zweier Objekte im dreidimensionalen Raum beschrieben.

Crowd Agent
Virtueller Akteur als Teil einer simulierten Masse.

Dirt Map
Überlagernde Textur zur Erzeugung von optischen Unreinheiten.

DOF (Degrees of Freedom)
Einschränkung der Translations- und Rotationsfreiheit eines Objekts im dreidimensionalen Raum.

Effector
Beschreibt den Endpunkt einer Bone Kette, welcher die Rotationswerte der in der Kette befindlichen Joints durch Translation des Effektors beeinflusst.

Epipolare Linie
Spiegelt die Achse der Kamera wider, die einen optischen Marker - im Zuge eines Motion Tracking Verfahrens - im Raum erfasst hat.

Height Map
Formgebendes Graustufenbild zur Definition von Höhenünterschieden geometrischer Objekte im dreidimensionalen Raum.

Loop
Ein Motion Loop oder Cycle definiert einen nahtlos wiedholbaren Bewegungszyklus.

LOD (Level of Detail)
Der Detailgrad einer dargestellten Geometrie, definiert durch die Anzahl an Polygonen.

Lot
Zweidimensionale, polygonale Form zur Definition von Grundstücksflächen.

Marker
Im Zuge eines Motion Tracking Verfahrens beschreibt ein Marker ein optisches Hilfsobjekt, welche Bewgungsinformationen in Form von Translationdaten erfasst und wiedergibt.

Mesh
Polygonnetz zur Beschreibung von dreidimensionalen Geometrien.

Motion Tree
Der Motion Tree beschreibt ein, aus einzelnen Bewegungen bestehendes, hierarchisches Netzwerk, welches alle Möglichkeiten an kombinierten Bewegungsfolgen eines Crowd Agents abbildet.

Node
Ein Node beschreibt einen Knotenpunkt eines hierarchisch aufgebauten Netzwerks aus parametrisierbaren Elementen. Dieser Ansatz bietet im Gegensatz zur Code basierenden Programmierung eine visuelle Komponente, welche die Übersicht über Datenflüsse und Abhängigkeiten innerhalb des Netzwerks verbessert.

Normals
Normalvektoren im Schwerpunkt von Polygonflächen.

Null Objekt
Ein nicht bildgebendes Objekt im Bereich der 3D Animation, welches verwendet wird um Positionen im dreidimensionalen Raum zu definieren.

Occlusion
Im Rahmen eines Motion Capturing Verfahrens beschreibt Occlusion die temporäre Überdeckung von optischen Markern durch Körperteile oder andere Marker.

Parser
Ein Computerprogramm, das in der Computertechnik für die Zerlegung und Umwandlung einer beliebigen Eingabe in ein für die Weiterverarbeitung brauchbares Format zuständig ist.

Physiognomie
Unter Physiognomie wird das äussere Erscheinungsbild eines Körpers bezeichnet.

plot
siehe bake.

prozedural
Mathematische und physikalische Algorithmen zur automatisierten Erstellung von parametrisierbaren 3D Modellen, Texturen und Animationen.

Random
Funktion zur generierung von zufälligen Werten.

Referenced Model
Als Referenz integrierte 3D Objekte, welche die unabhängige Editierung von referenzierten Modellen ermöglicht, ohne die Szene zu beeinflussen.

Rig
Für Animationszwecke aufbereitete Skelettstruktur eines virtuellen Characters.

Rigging
Dieser Prozess beschreibt die Erstellung der Skelettstruktur, sowie benötigter Relationen und Hilfsobjekte eines virtuellen Characters.

Street Graph
Auf Vektoren basierendes Strassennetz. Aus diesen Vektordaten wird in weiterer Folge der Grundriss von Strassen und Kreuzungen (Street Shapes) erzeugt.

Street Shape
Zweidimensionaler Grundriss von Strassenverläufen. Auf Basis der Street Shapes wird die Geometrie der Strassennetzwerke erzeugt.

Subdivision Surface
Auf Vektoren basierender Algorithmus zur rekursiven Unterteilung von Polygonflächen. Durch die Intopaltion werden Kanten der Geometrie abgerundet.

Talent
Schauspieler für die Bewegungsaufzeichnung im Rahmen eines Motion Capturing Verfahrens.

Texture Map
Das Mapping von Texturen beschreibt die Projektion der Textur auf eine dreidimensionale Geometrie anhand der UV Koordinaten.

Tile
Textur, welche sich selbst ohne optische Unterbrechungen wiederholen lässt.

UV
Koordinatensystem zur Beschreibung der Oberfläche einer Geometrie für die Projektion von Texturen.

Literaturverzeichnis

CityEngine Manual (2009)

Curdes, G./Haase, A./Rodriguez-Lores, J. (1989): Stadtstruktur: Stadtbild und Wandel. Beiträge zur stadtmorphologischen Diskussion, Köln

Demers, O. (2002): Texturing & Painting, New Riders Publishing, USA

Ebert, D./Kenton Musgrave, F./Peachey, D./Perlin, K./Worley, S. (2003): Texturing & Modeling. A Procedural Approach, 3. Aufl. Morgan Kaufmann Publishers, San Francisco

Huber, B. (1997): Städtebau-Raumplanung, 4. Aufl. In: Band 1: Städtebau. Grundlagen und Materialien. Verlag der Fachvereine an den schweizerischen Hochschulen und Techniken, Zürich

Kerlow, I. (2004): The Art of 3D Computer Animation and Effect, third Edtion, John Willey & Sons Inc, New Jersey

Kerlow, I. (2009): The Art of 3D Computer Animation and Effects, 4. Aufl. John Wiley & Sons Inc., Hoboken, New Jersey

Kitagawa, M./Windsor, B. (2008): MoCap for Artists, Workflow and Techniques for Motion Capture, Elsevier Inc, Oxford

Liverman, M. (2004): The Animator's Motion Capture Guide, Organizing, Managing, and Editing, Charles River Media, Hingham

Maestri, G. (1999): Digital Character Animation 2, Volume 1: Essential Techniques, New Riders Publishing, Indianapolis

Maestri, G. (2002): Character Animation 2 - advanced techniques. New Riders Publishing, USA

Massive Software (2008): Massive Manual, Version 3.5.1 revision 109

Massive Software (2010a): What is Massive?, http://www.massivesoftware.com/whatismassive/, Zugriff am 20.03.2010

Massive Software (2010b): Massive 4.0 Supporting V-Ray Rendering is Now Available, http://www.massivesoftware.com/massive-4-0-supporting-v-ray-rendering-is-now-available/, Zugriff am 25.03.2010

Meade, T./Arima, S. (2007): Maya 8: The Complete Reference, Mcgraw-Hill Professional, New York

Menache, A. (2000): Understanding Motion Capture for Computer Animation and Video Games, Academic Press, San Diego

Müller, P./Wonka, P./Haegler, S./Ulmer, A./Van Gool, L. (2006): Procedural Modeling of Buildings. ACM SIGGRAPH 2006 Papers. Boston, Massachusetts

Müller, Y./Müller, P. (2001): Procedural Modeling of Cities. ACM SIGGRAPH 2001 Papers. Los Angeles, California

Petrasch, T./Zinke, J. (2003): Einführung in die Videofilmproduktion, Carl Hanser Verlag Leipzig

Rich, E. (1983): Artificial Intelligence, McGraw Hill Inc., New York

Schmidth, U. (2005): Professionelle Videotechnik, 4. Aktualisierte und erweiterte Auflage, Springer Verlag, Berlin Heidelberg

Schönherr, M. (2000): Maya 3, Ästhetik & Technik von High-End 3D-Animationen, Addison-Wesley Verlag, München

Schubiger-Banz, S. (2009): Prozedurale Modellierung. In: Digital Production Nr.6/09

Thalmann, D./Raup Musse, S. (2007): Crowd Simulation, Springer, London

Vince, J. (2000): Essential Computer Animation, How to Understand the Techniques of Computer Animation, Springer-Verlag, London

Vogel, N./Sheridan, S./Coleman, T. (2000): Maya 2, Character Animation, 3D Story Concept Development & Modeling, New Riders Publishing, Indianapolis

Wenisch, T. (2005): Kurzlehrbuch Physik Chemie Biologie, Elsevier Urban & Fischer, München
Wheeler, P. (2005): Practical Cinematography, Second Edition, Focal Press, Oxford

Winter, R. (2001): Grundlagen der formalen Logik, 2. Aufl., Wissenschaftlicher Verlag Harri Deutsch GmbH, Frankfurt am Main

3ds Max Cat Help (2010a): CATMotion, http://download.autodesk.com/esd/3dsmax/cat-help-2010/index.html?url=WS7af5cac11814013a-2ee5b21011fde8c01a2-8000.htm,topicNumber=d0e6403, Zugriff am 12.03.2010

3ds Max Cat Help (2010b): Hubs, http://download.autodesk.com/esd/3dsmax/cat-help-2010/index.html?url=WS7af5cac11814013a17ba0fbf11fde8bf84b-7fb2.htm,topicNumber=d0e1449, Zugriff am 12.03.2010

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

Agent Channels

channel	description	input / output	min	max	example
tx ty tz	translation rate of agent	both	-inf	inf	ty
rx ry rz	rotation rate of agent	both	-inf	inf	ry
lx ly lz	lean of agent in world space (see lean)	input			ly
collide	indicates the occurance and depth of a collision with another agent	input		inf	collide
collide.v	velocity of collision	input		inf	collide.v
collide.x collide.y collide.z	magnitude of collision in agent space (normal x depth)	input	-inf	inf	collide.x
collide.vx collide.vy collide.vz	velocity vector of collision in agent coordinates	input	-int	int	collide.vx
collide.nx collide.ny collide.nz	normal vector of collision in agent coordinates	input			collide.nx
collide.px collide.py collide.pz	position of collision in agent coordinates	input	-inf	inf	collide.px
ground	height of terrain relative to agent (see terrain)	input	-inf	inf	ground
ground.r ground.g ground.b ground.a	colour of terrain beneath agent	input			ground.r
ground.dx ground.dz	slope of terrain along agent x and z axes	input	-inf	inf	ground.dx
ground.r.dx ground.g.dx ground.b.dx ground.r.dz ground.g.dz ground.b.dz	gradient of terrain R G and B with respect to agent X and Z axes.	input	-inf	inf	ground.r.dx
ground.flow	terrain flow direction relative to agent orientation	input			ground.flow
balance.x balance.z	distance that agent is off balance in x and z	input	-inf	inf	balance.x
sound.f sound.a sound.f1f sound.f1a sound.f2f sound.f2a	sound emmission / reception (see sound)	both		inf	sound.f
sound.d sound.x sound.z	see sound	input			sound.d
wind.x wind.y wind.z	velocity of wind (see wind)	both	-inf	inf	wind.x

wind.a wind.f	amplitude and frequency of wind turbulence	output		inf	wind.a
dynamics.active	activates rigid body dynamics for the agent (see-dynamics)	output			dynamics.active
dynamics.rx.fixed dynamics.ry.fixed dynamics.rz.fixed dynamics.tx.fixed dynamics.ty.fixed dynamics.tz.fixed	causes agent space in certain transformation to remain unchanged before, during and after dynamics (default value for each channel varies, seeBlending Dynamics with Kinematics for more detail)	output			dynamics.rx.fixed
vision.active	switches off vision processing if set to zero (default value is one).	output			vision.active
active	switches as agent off if set to zero	output			active
parent	causes an agent to become parented to an agent with which it is colliding	output			parent
keyboard	Ascii code for the key that was pressed. For realtime keyboard control of agents.	input			keyboard
controller.joy1.x controller.joy1.y controller.joy2.x controller.joy2.y controller.joy3.x controller.joy3.y controller.slider controller.b1 . . controller.b8	Gamepad controller joysticks, d pad, trim slider and buttons. For realtime control of agents. (see joystick/keyboard controls)	input			controller.joy3.x
servo.force	magnitude of servo force to be used for all degrees of freedom for all segments (see dynamics)	output		inf	servo.force
paint.r, paint.g, paint.b, paint.a	level of red/green/blue/alpha channels that agent paints on the ground	output			paint.r
paint.size	size of brush agent uses to paint on the ground	output		inf	paint.size

colour	colour of agent’s skeleton	input/ output			colour
lane.x	lateral position of agent in lane: -1 = left edge, 0 = center, 1 = right edge	input			lane.x
lane.ox	relative angle of lane to agent in degrees, similar to ground.flow	input			lane.ox
lane.h	hue of lane, similar to vision.h	input			lane.h

Action Channels

channel	description	input / output	min	max	example
action	A value of 1 triggers the action. A value of 2 triggers the action irrespective of latch state (see actions)	output			walk
action	The activation level of the action	input			walk
action1->action2	Blends action2 over action1	output			walk->sidestepL
[channel]:scale	Scales the action values applied to the channel.	output	-inf	inf	[head:ry]:scale
[channel]:offset	Offsets the action values applied to the channel.	output	-inf	inf	[head:ry]:offset
[channel]:origin	Sets the origin for scaling of the channel	output	-inf	inf	[r_ toes:ik.y]:origin
action:rate	Sets the playback rate. 0.5 causes the action to play at half speed. -1 causes it to play backwards.	both	-inf	inf	walk:rate
action:running	A binary value indicating wether the the action is playing or not.	input			walk:running
action:phase	The phase of the playback of the action	input			walk:phase
phase	The phase of the most active playback on track 1	input			phase
phase2 ... phase8	The phase of the most active playback on tracks 2 to 8	input			phase5
action:phase_ offset	Sets the starting phase offset for the action.	output			walk:phase_offset
latch	The state of the latch for track 1	intput			latch

Segment Channels

channel	description	input / output	min	max	example
tx ty tz	segment translation	both	-inf	inf	head:ty
rx ry rz	segment rotation	both	-inf	inf	head:ry
x y z	segment location in agent space	input	-inf	inf	head:y
lx ly lz	lean of segment in world space (see lean)	input			head:ly
h	height of segment above ground in world space	input	-inf	inf	r_hand:h
mass	segment mass	output		inf	head:mass
dynamics. active	activates dynamics for the segment and it's descen-dants	output			head:dynamics.active
dynamics. detach	specfies whether segment, if dynamic, will detach from non-dynamic parent (1) or remain constrained to parent (0) according to rotation order and limits set in segment's 'dof' tab. Only works with ODE solver.	output			l_shoulder:dynamics. detach
grab	when set to 1, causes the segment to remain in the same position as ("grab onto") a colliding segment while maintaining 3 degrees of freedom in rotation. If there is currently no collid-ing segment, no constrain-ing will occur until there is one. When set to 0 this channel causes the segment to let go of the colliding segment. This channel can be switched on and off any number of times while the dynamics engine is still run-ning.Only works with ODE solver.	output			l_hand:grab
force.x force.y force.z	External force applied to segment. Forces only have effect when dynamics is active	output	-inf	inf	r_shoulder:force.z

force.px force.py force.pz	Location in segment space at which force is applied.	output	-inf	inf	r_shoulder:force.px
force.rx force.ry force.rz	Torque applied to segment.	output	-inf	inf	head:force.ry
ground	height of terrain under segment relative to agent (see terrain)	input	-inf	inf	r_foot:ground
ground.r ground.g ground.b ground.a	colour of terrain beneath segment	input			l_foot:ground.r
ground.r.dx ground.g.dx ground.b.dx ground.r.dz ground.g.dz ground.b.dz	gradient of terrain R G and B with respect to segment X and Z axes.	input	-inf	inf	l_foot:ground.r.dx
ground.flow	terrain flow direction relative to segment orientation	input			r_foot:ground.flow
vision.x vision.y vision.z vision.h vision.i	see vision	input			head:vision.x
ik.x ik.y ik.z ik.twist ik.active ik.hold ik.grab	see Inverse kinematics	both			r_hand:ik.x
rc.x rc.y rc.z rc.active	see rotation constraint	both			r_hand:rc.x
collide	indicates the occurance and depth of a collision with another segment or agent	input		inf	head:collide
collide.v	velocity of collision	input		inf	head:collide.v
collide.x collide.y collide.z	magnitude of collision in agent space (normal x depth)	input	-inf	inf	head:collide.x
collide.vx collide.vy collide.vz	velocity vector of collision in agent coordinates	input	-int	int	head:collide.vx

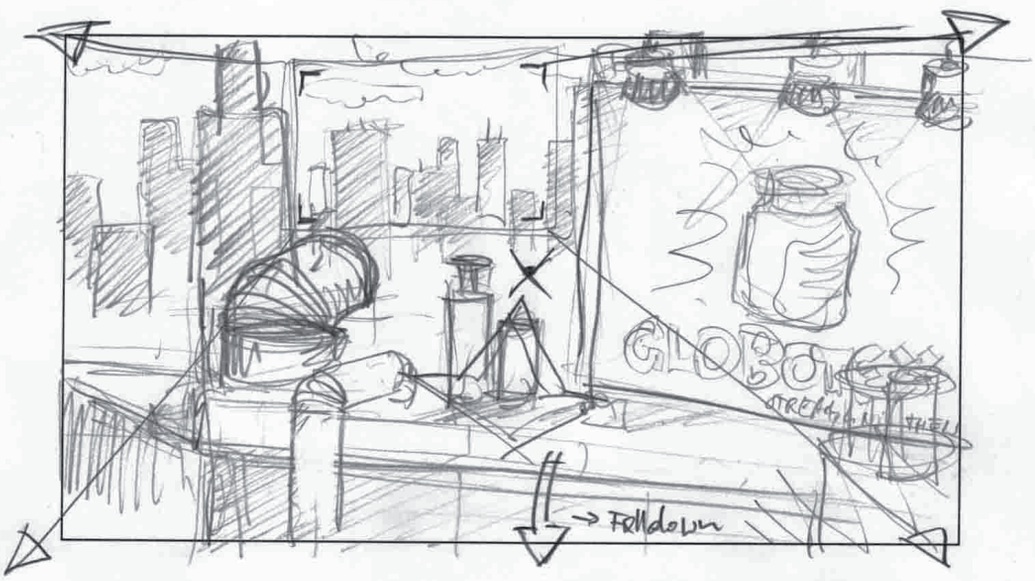
collide.nx collide.ny collide.nz	normal vector of collision in agent coordinates	input			head:collide.nx
collide.px collide.py collide.pz	position of collision in agent coordinates	input	-inf	inf	head:collide.px
spawn	A rising edge (i.e. going from 0 to 1) triggers the creation of a new instanced agent. The new agent will instanced from an original agent with the same name as the segment.	output			arrow:spawn
servo.force	magnitude of servo force to be used for all degrees of freedom for the segment (seedynamics)	output		inf	head:servo.force
servo.force.x servo.force.y servo.force.z	magnitude of servo force to be used for individual degrees of freedom for the segment (see dynamics)	output		inf	head:servo.force.x
servo.rx servo.ry servo.rz	sets the target rotation values for the servos for the segment. Note that these values are overriden by actions (seedynamics)	output	-inf	inf	head:servo.rx
paint.r, paint.g, paint.b, paint.a	level of red/green/blue/alpha channels that the segment paints on the ground	output			head:paint.r
paint.size	size of brush the segment uses to paint on the ground	output		inf	head:paint.size
colour	colour of segment	input/ output			head:colour

Storyboard

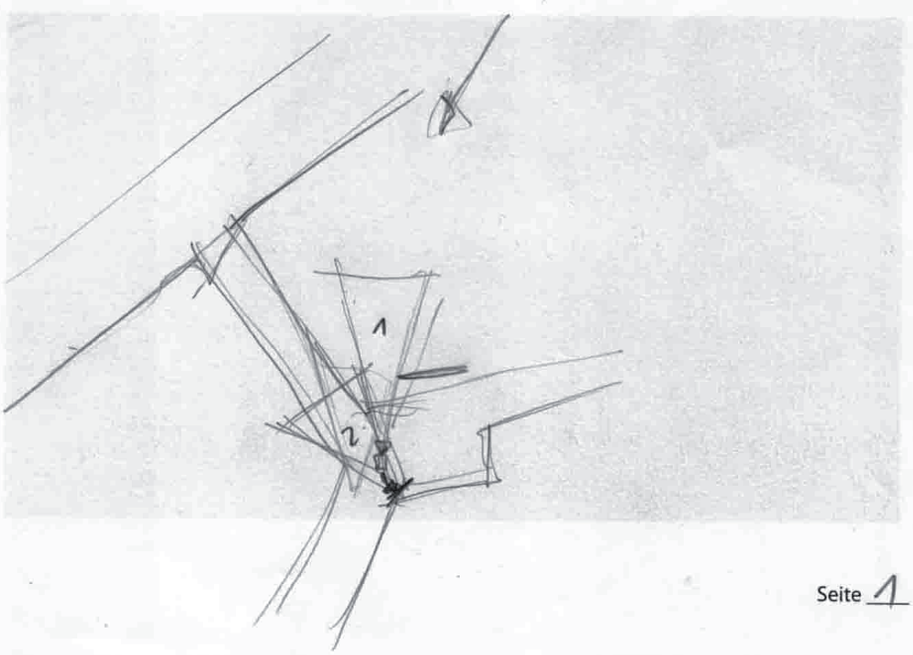
SZENE - Nr.: 1.1 SHOT-DAUER: 15s

SET: ROOFTOP

CAMERA SETUP: Weitwinkel - Establishing Shot - Dolly Back to Crane Folldown



SHOT SETUP

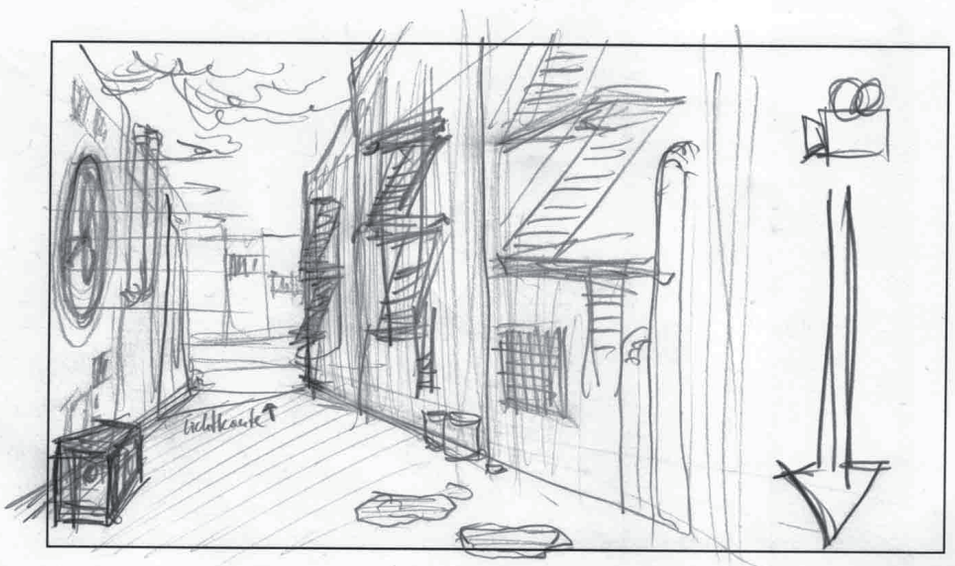


Seite 1

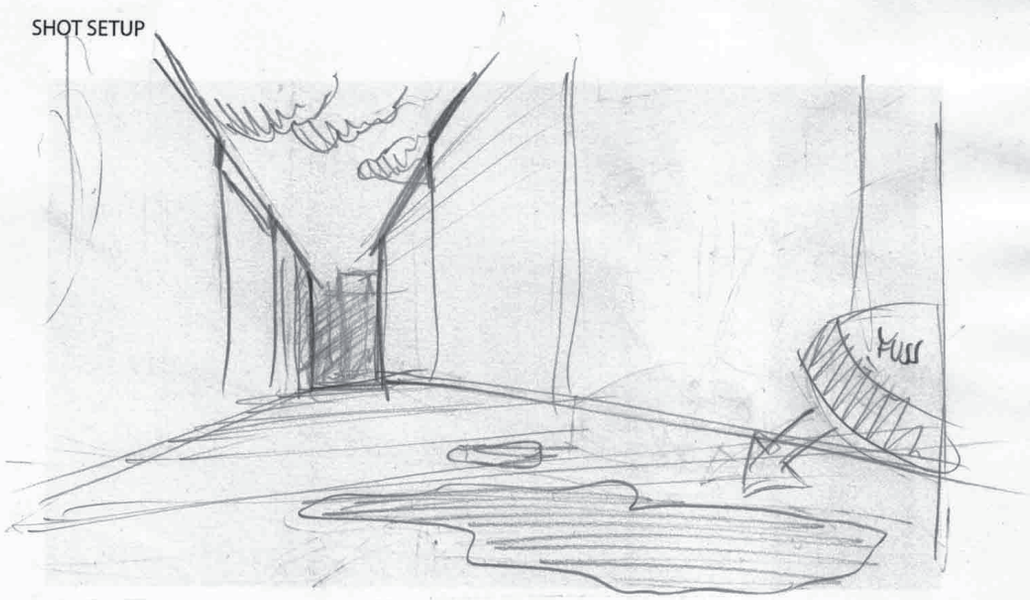
SZENE - Nr.: 1.1(P) SHOT-DAUER: 15s

SET: SEITENSTRASSE BODEN

CAMERA SETUP: _____



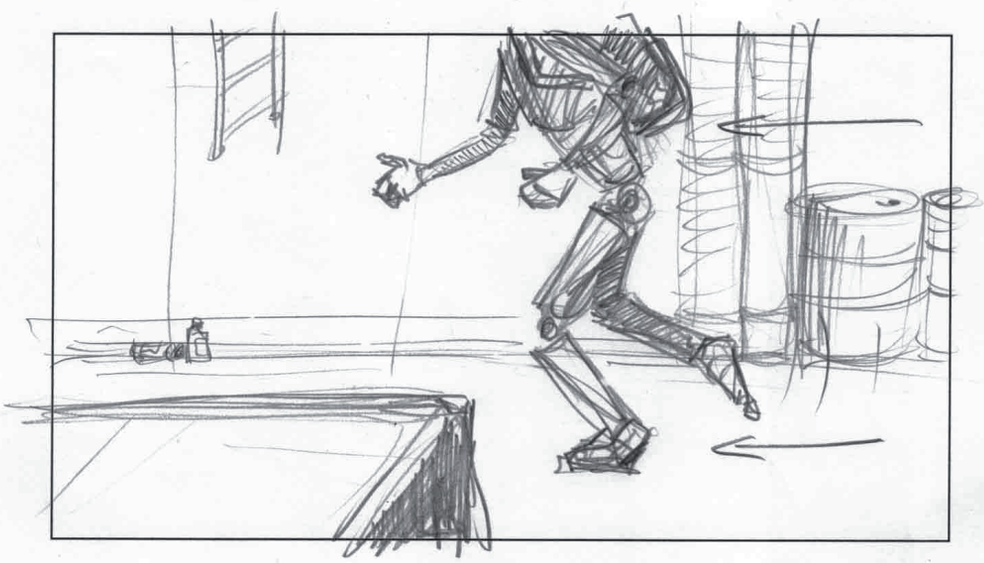
SHOT SETUP



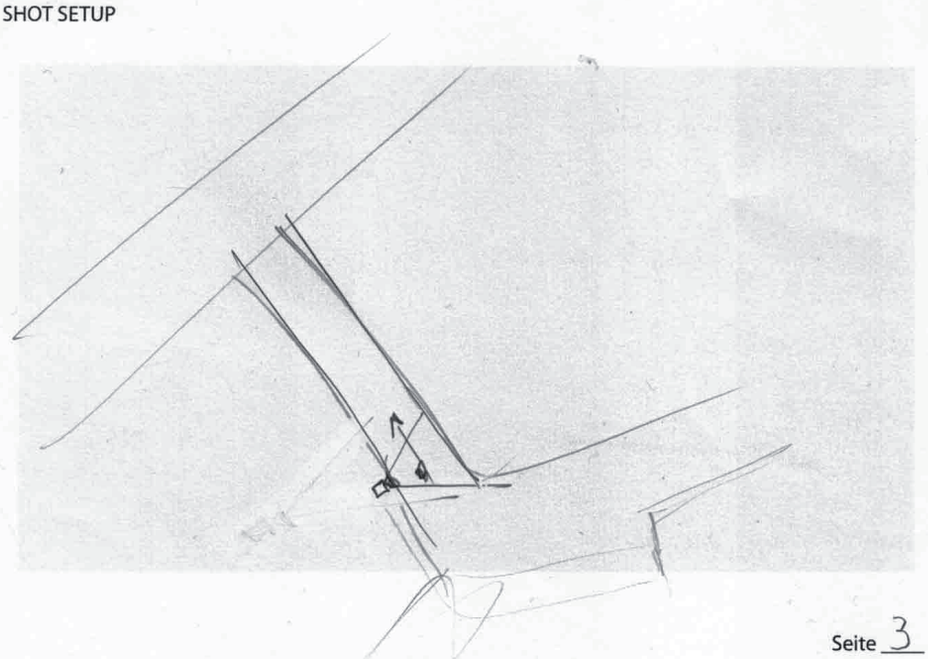
SZENE - Nr.: 2.1 SHOT-DAUER: 2-3s

SET: Seitengasse – Später Nachmittag/Abendstimmung

CAMERA SETUP: HALF SHOT – Dutch Angle



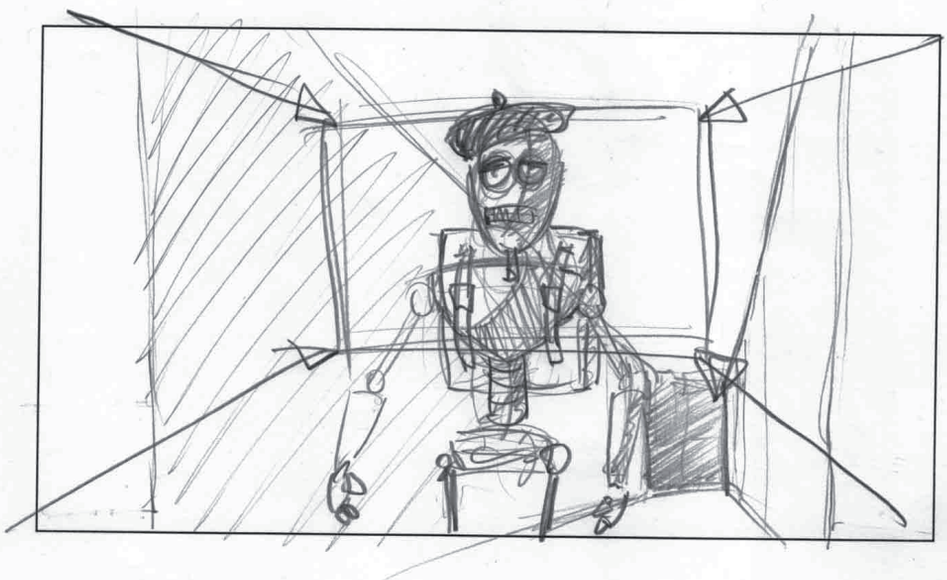
SHOT SETUP



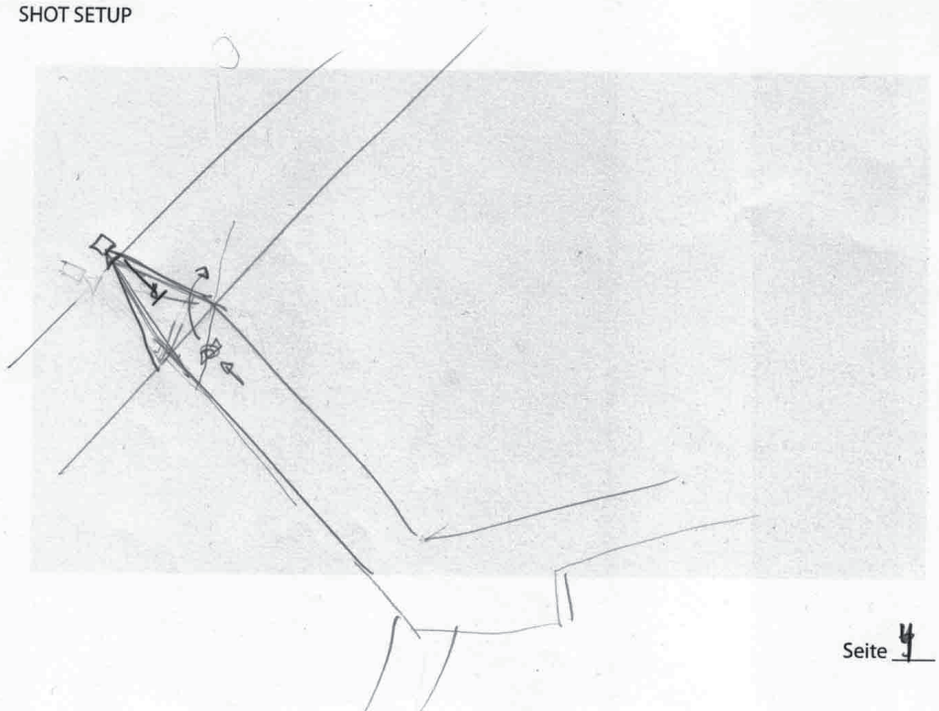
SZENE - Nr.: 3.1 SHOT-DAUER: 7s

SET: ~~Ext~~ - Allee (Seitengasse)

CAMERA SETUP: Medium - Medium Close contract Dolly



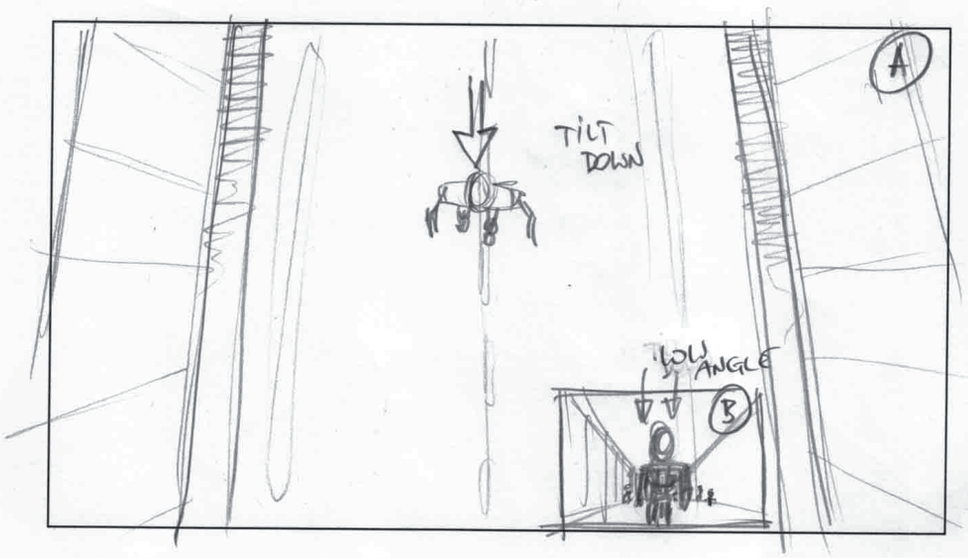
SHOT SETUP



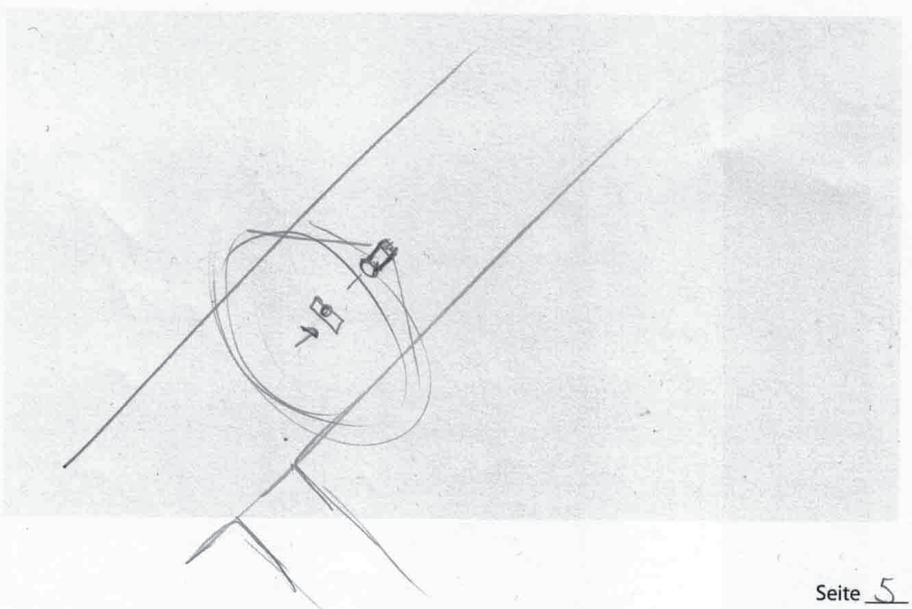
SZENE - Nr.: 3.2 SHOT-DAUER: 4-5s

SET: Ext - Alle

CAMERA SETUP: Wide shot Fly over - Top -> Front



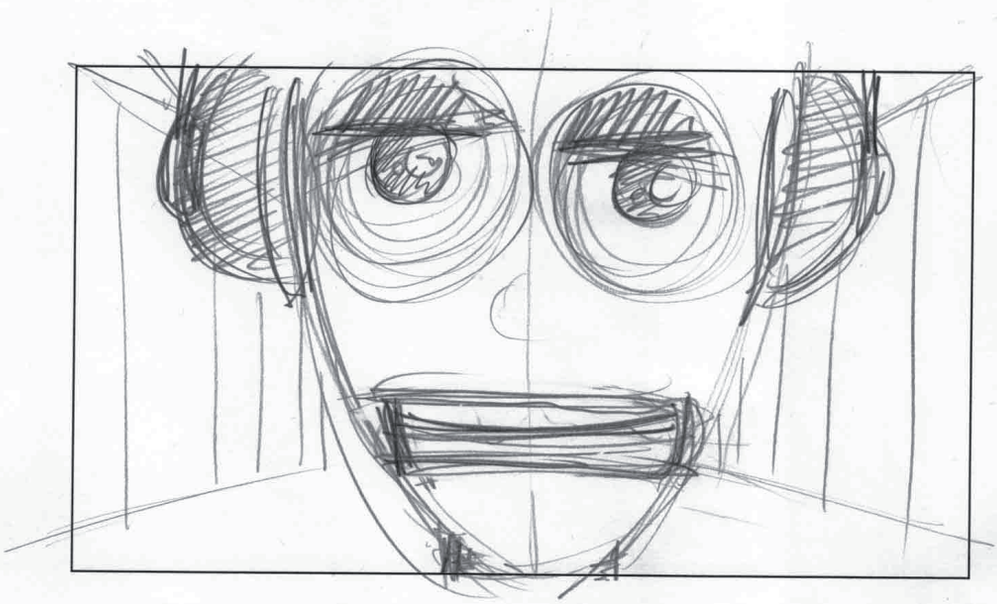
SHOT SETUP



SZENE - Nr.: 3.3 SHOT-DAUER: 2s

SET: EXT - ALLEE - später NM.

CAMERA SETUP: CLOSE UP



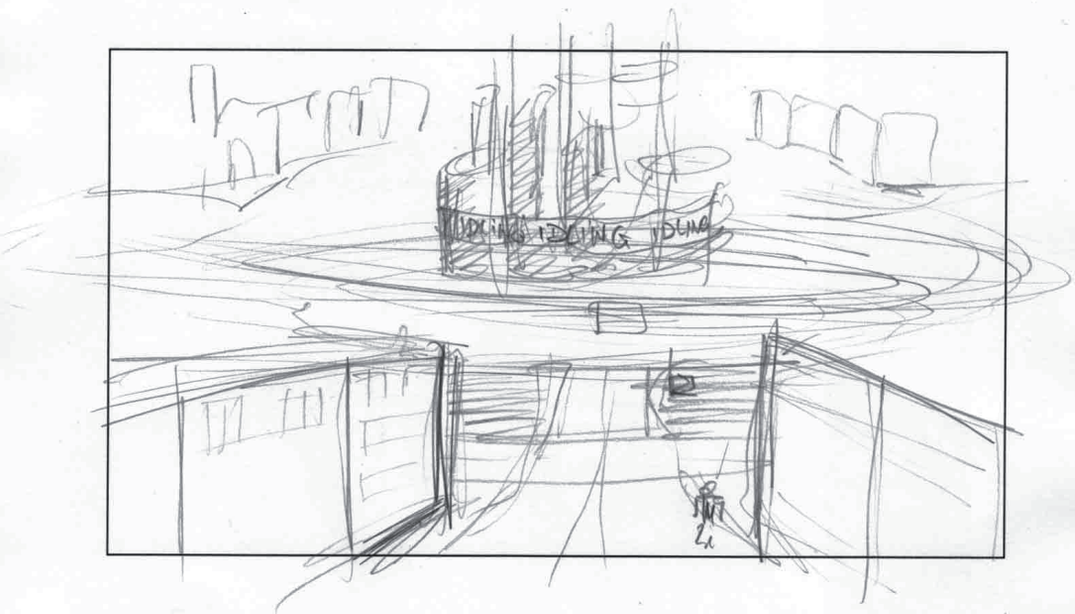
SHOT SETUP

DIVERSE CLOSE-UPS
/EXTREME CLOSE UPS

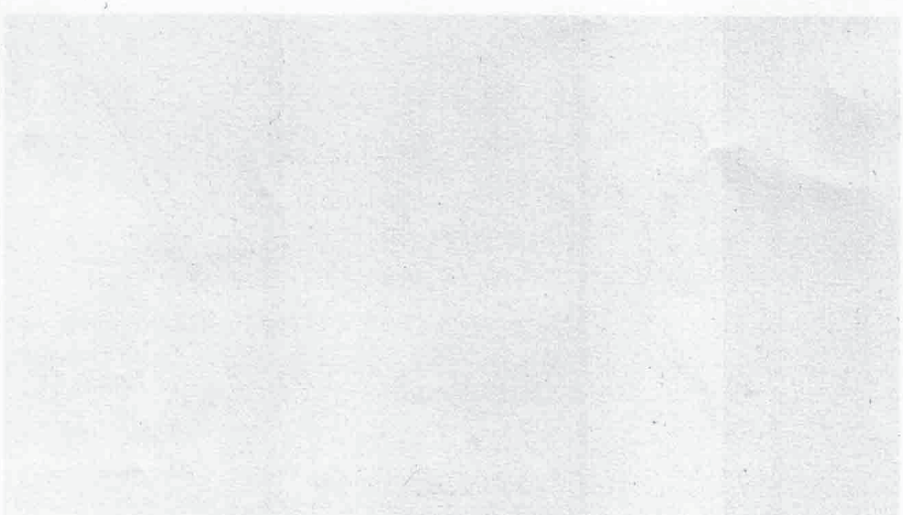
SZENE - Nr.: 4.1 SHOT-DAUER: 5 s

SET: Ext - Kreuzung

CAMERA SETUP: Depth Dolly



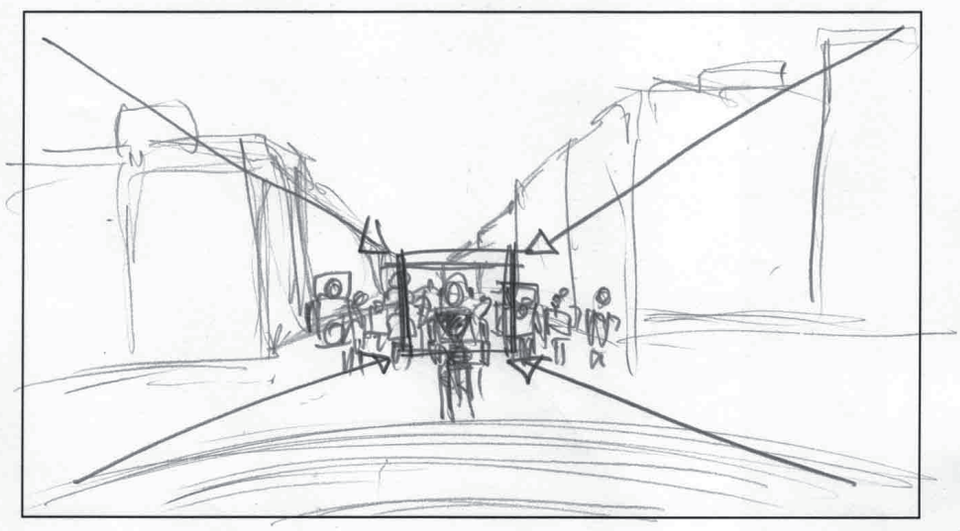
SHOT SETUP



SZENE - Nr.: 4.2 SHOT-DAUER: 4s

SET: EXT - KREUZUNG

CAMERA SETUP: Full Shot - Contract Dolly

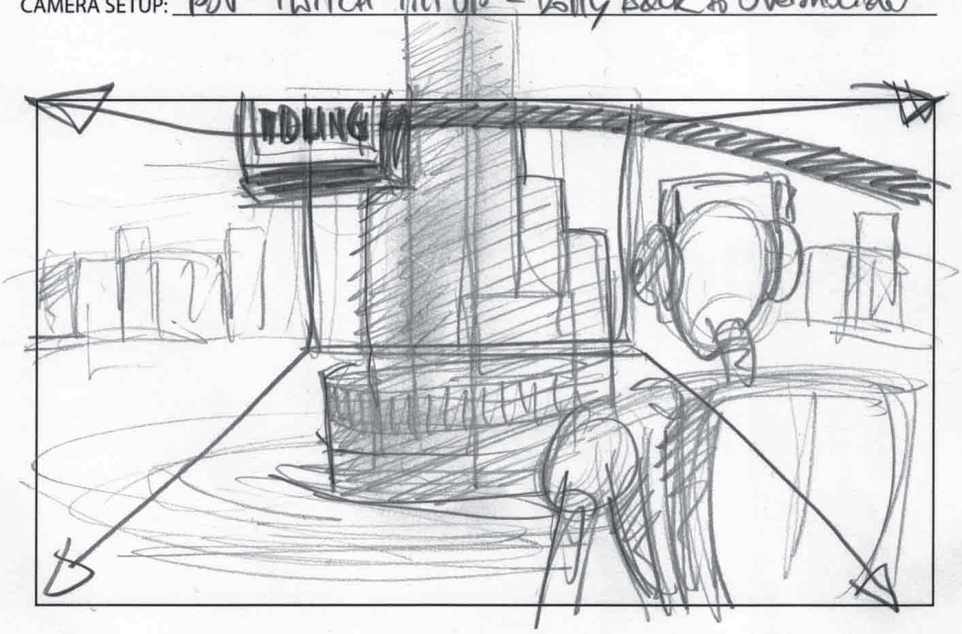


SHOT SETUP

SZENE - Nr.: 4.3 SHOT-DAUER: 5s

SET: EXT - KREUZUNG

CAMERA SETUP: POV - TWITCH TILT UP - Dolly Back to Overshoulder

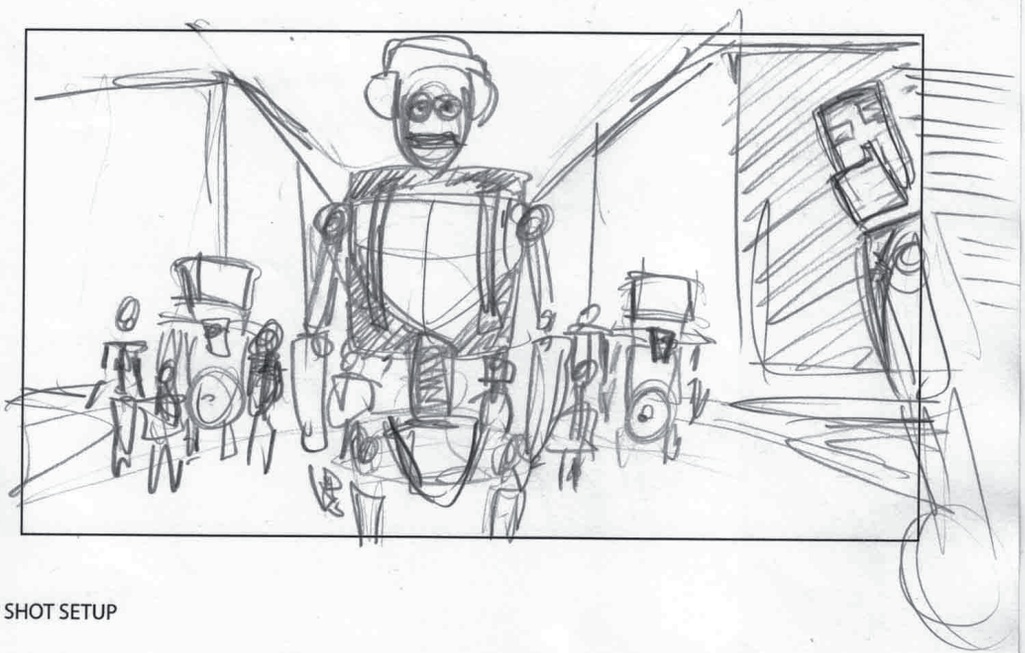


SHOT SETUP Ampel switcht auf „BUSY"

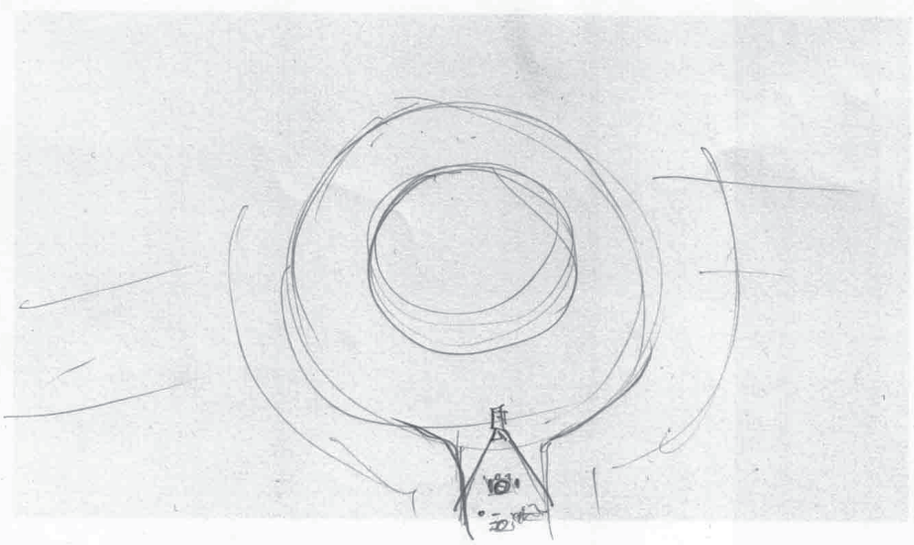
SZENE - Nr.: 4.4 SHOT-DAUER: 4s

SET: Ext - Kreuzung

CAMERA SETUP: Medium Shot



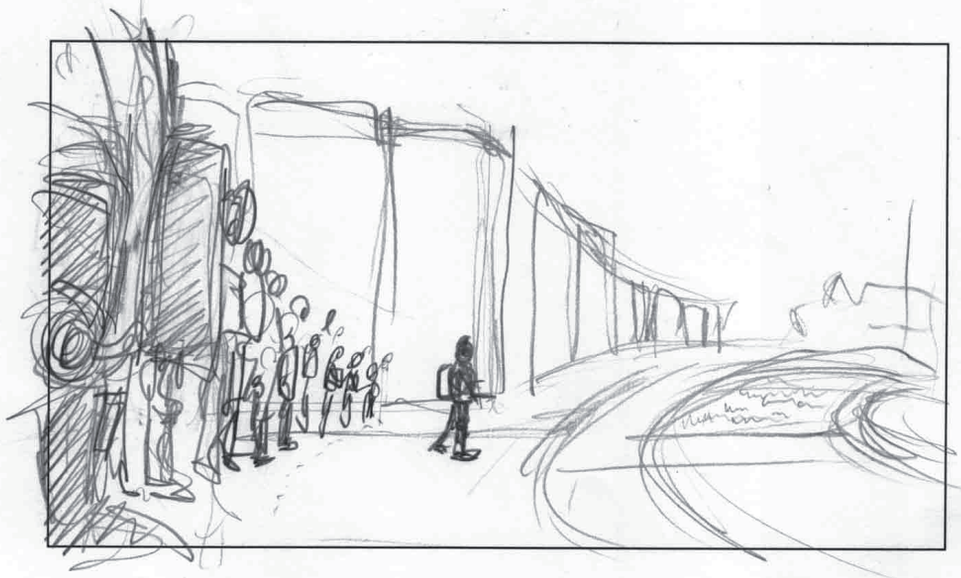
SHOT SETUP



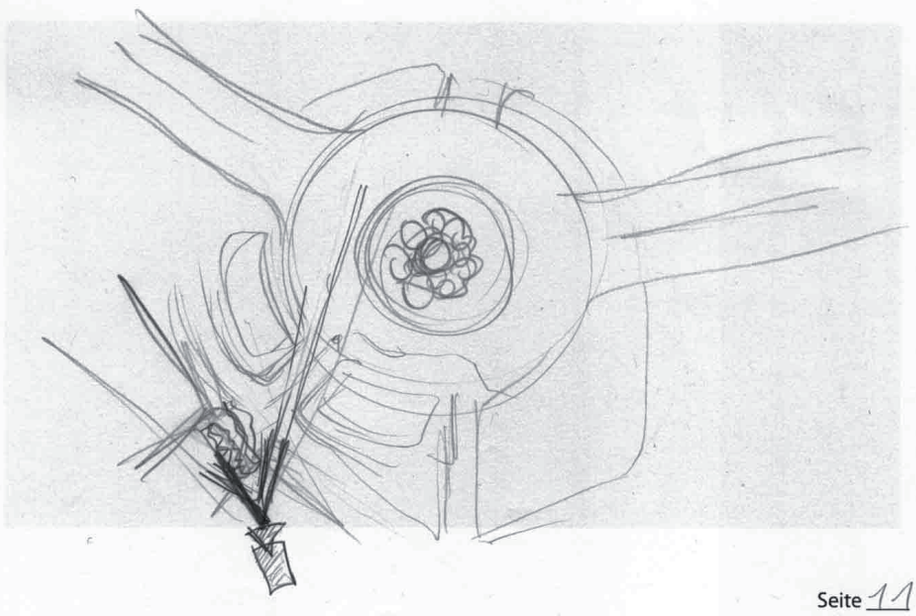
SZENE - Nr.: 4+5 SHOT-DAUER: 5s

SET: EXT - KREUZUNG

CAMERA SETUP: Low Angle Depth Staging



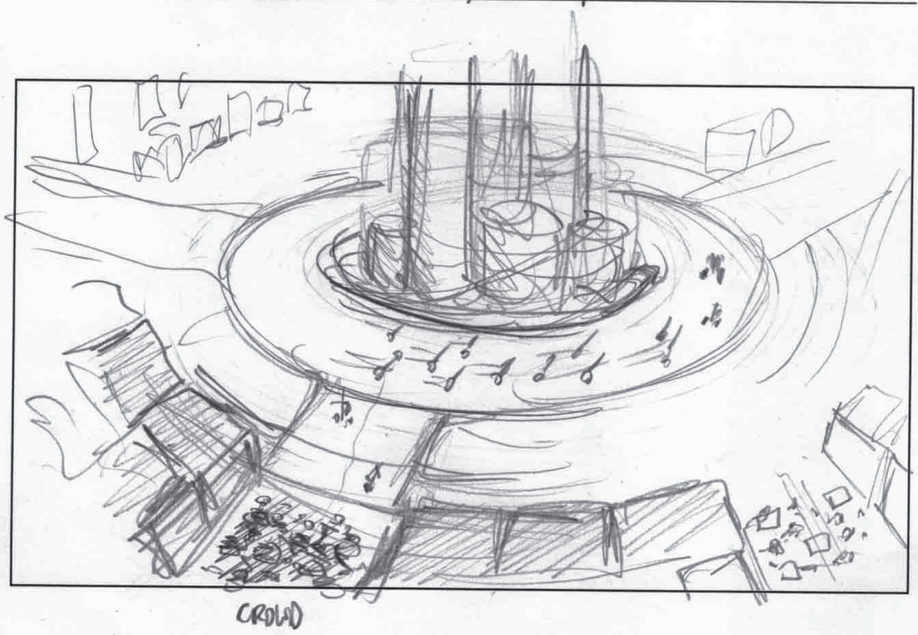
SHOT SETUP



SZENE - Nr.: 4.6 SHOT-DAUER: 5-6 s

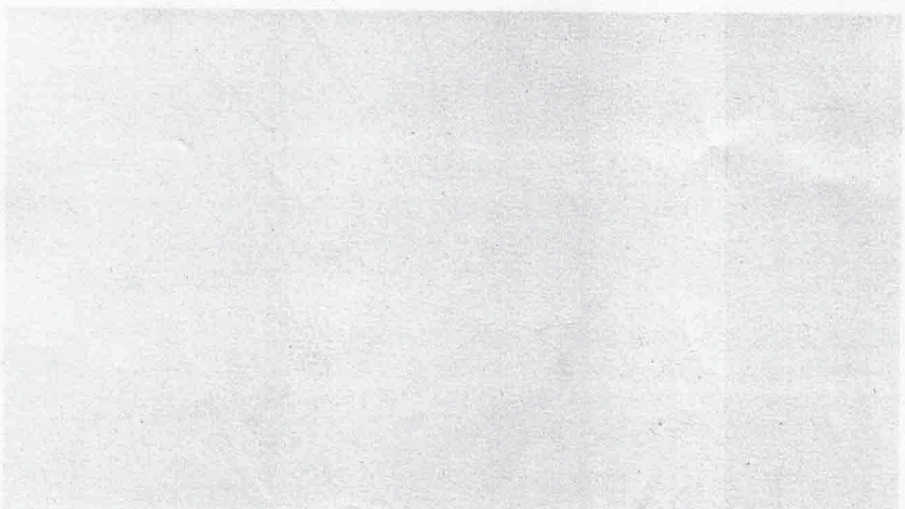
SET: EXT KREUZUNG

CAMERA SETUP: LOOK DOWN - STILL / SLOW PAN



CROWD

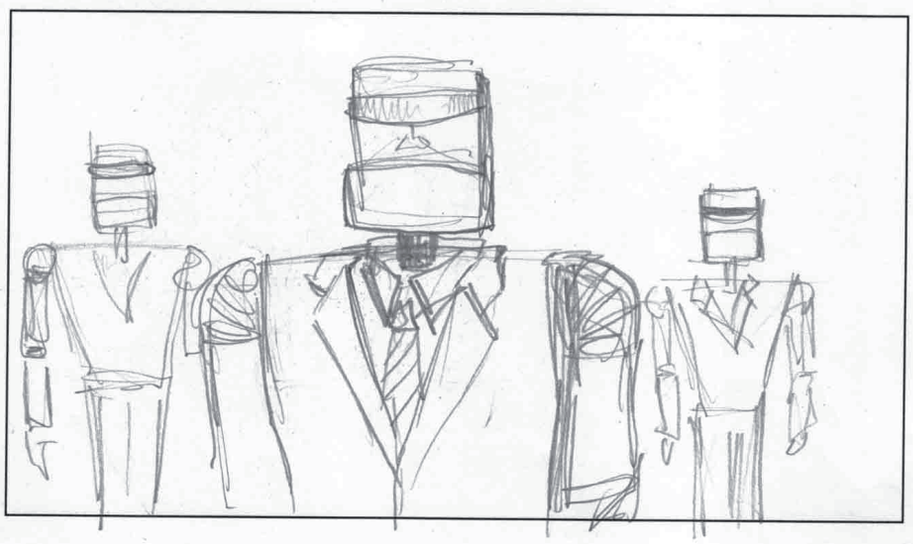
SHOT SETUP



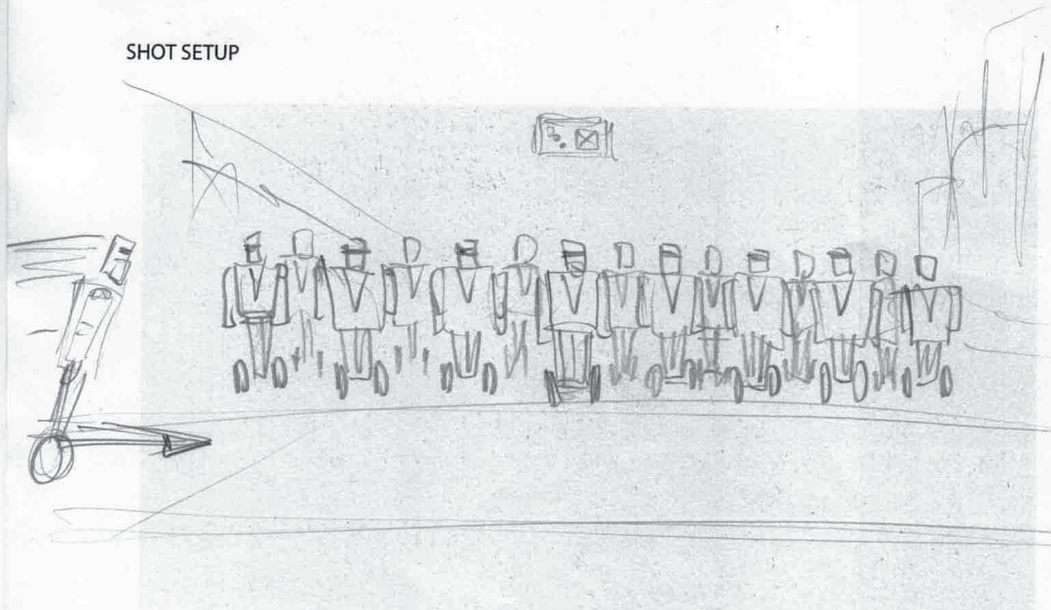
SZENE - Nr.: 5.1 SHOT-DAUER: 6-7s

SET: _____

CAMERA SETUP: MEDIUM SHOT



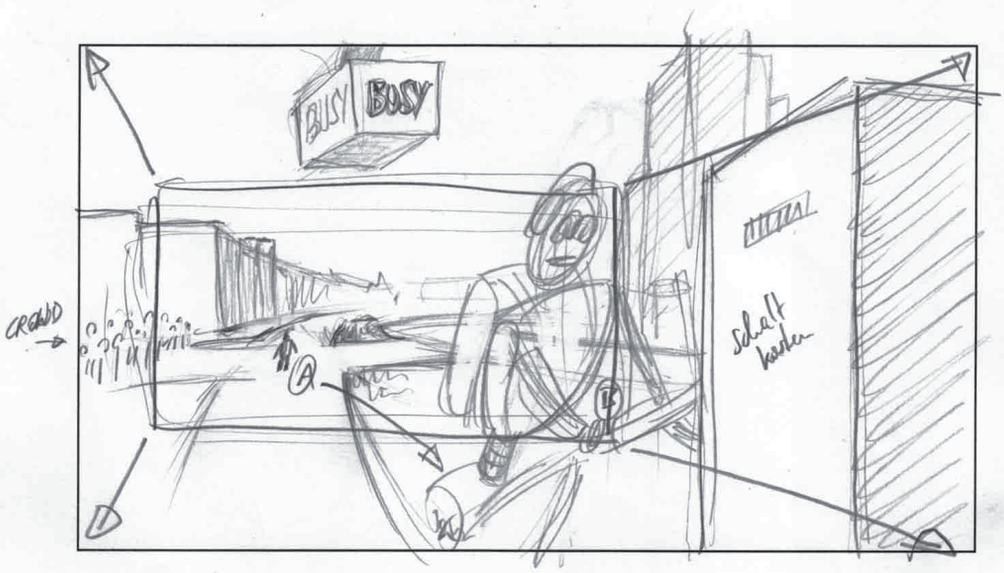
SHOT SETUP



SZENE - Nr.: 6.1 SHOT-DAUER: _____

SET: KREUZUNG

CAMERA SETUP: FULL / MEDIUM - Shot, FOCUS PULL

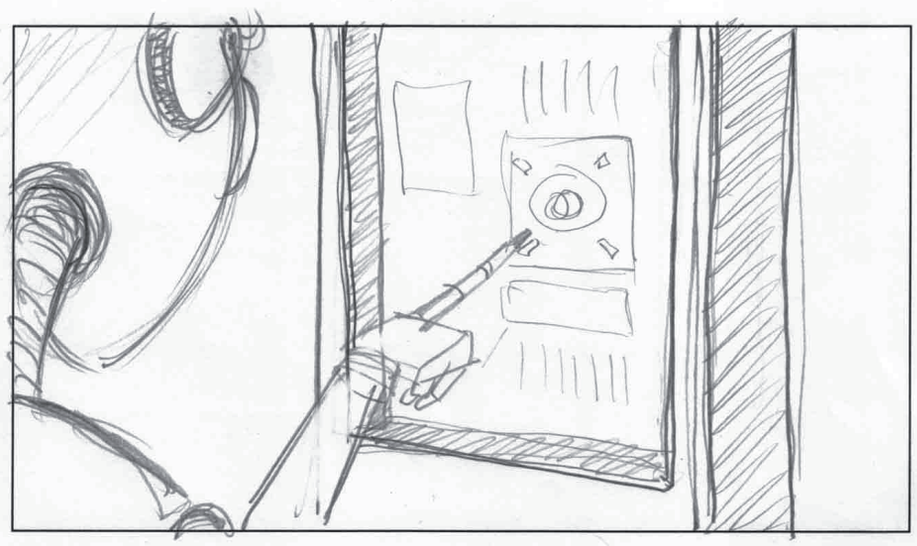


SHOT SETUP

SZENE - Nr.: 6.2 SHOT-DAUER: 25

SET: KREUZUNG

CAMERA SETUP: CLOSE SHOT

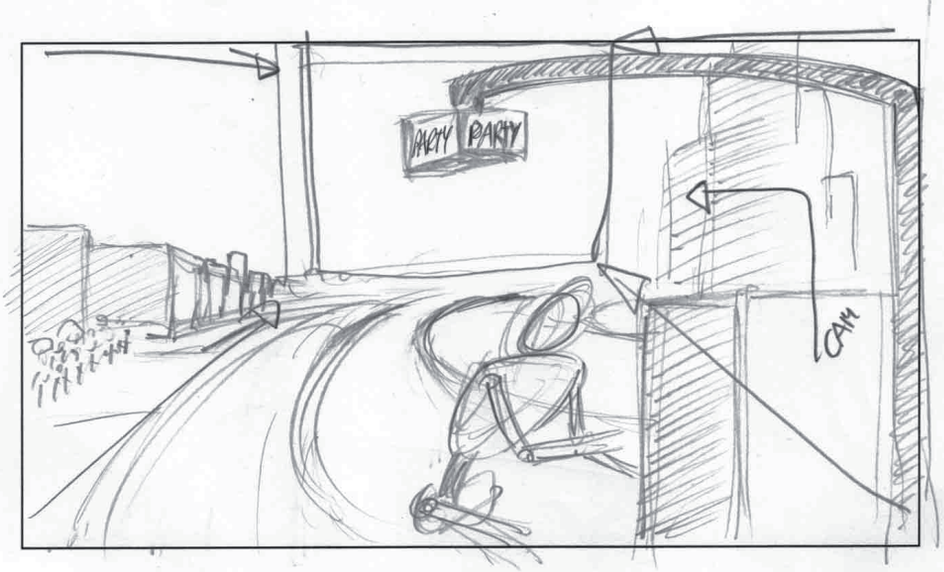


SHOT SETUP

SZENE - Nr.: 6.3 SHOT-DAUER: 5s

SET: KREUZUNG

CAMERA SETUP: wie 6.1 → +CRANE CAM

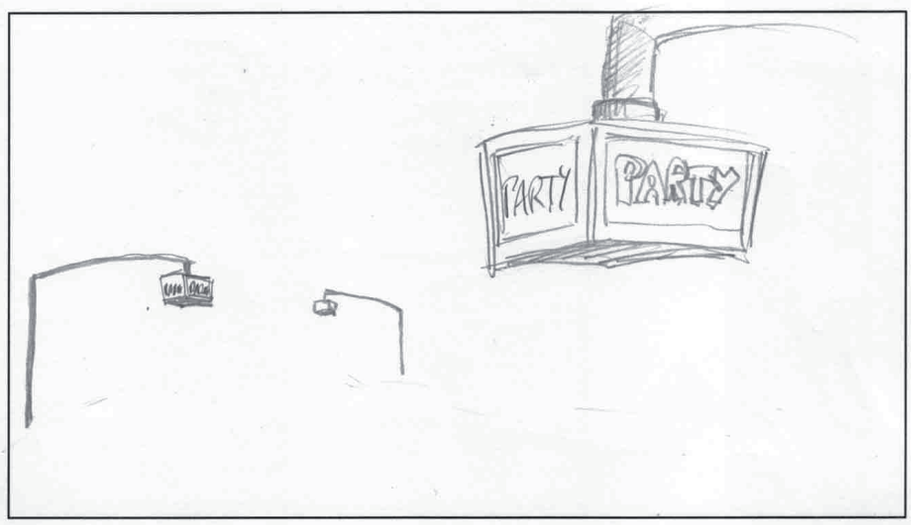


SHOT SETUP

SZENE - Nr.: 6.4 SHOT-DAUER: 2s

SET: _____

CAMERA SETUP: MID ANGLE SHOT - PERSPECTIVE

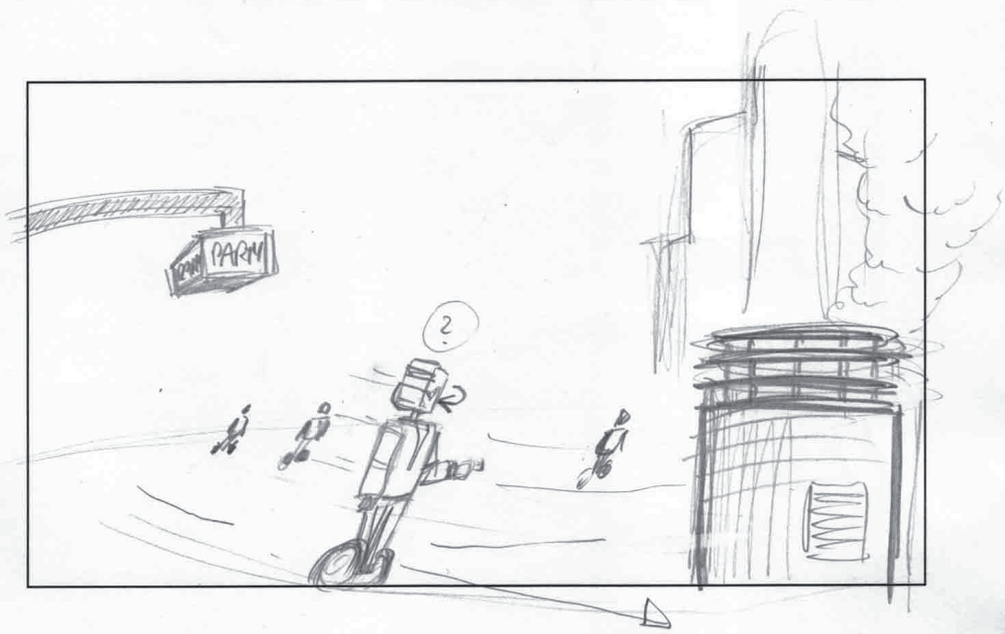


SHOT SETUP

SZENE - Nr.: 6.5 SHOT-DAUER: 5-6s

SET: KREUZUNG

CAMERA SETUP: LOW ANGLE – CAM TILTED UP

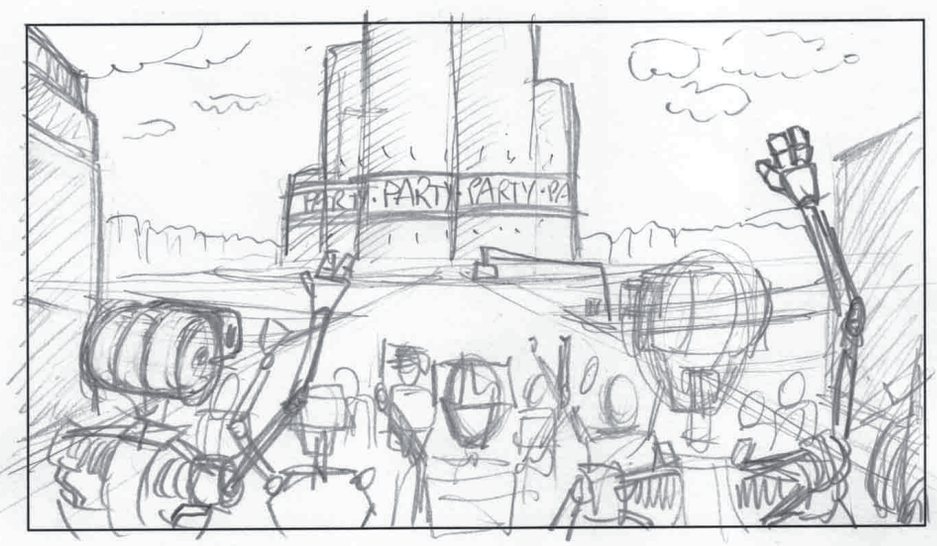


SHOT SETUP

SZENE - Nr.: 6.6 SHOT-DAUER: 4 s

SET: KREUZUNG

CAMERA SETUP: WIDE - SHOT - POV - DEPTH STAGING - FOCUS PULL

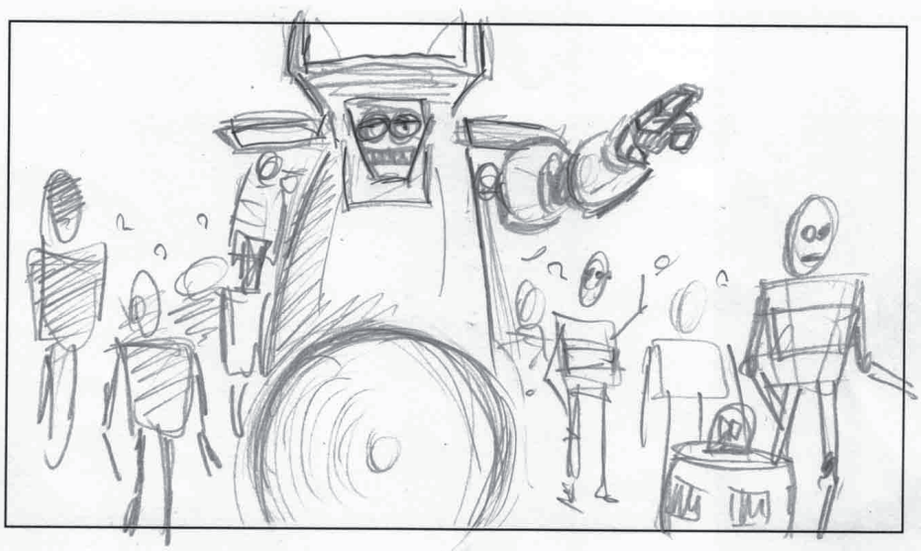


SHOT SETUP

SZENE - Nr.: 6.7 SHOT-DAUER: 1-2s

SET: _____

CAMERA SETUP: MEDIUM CLOSE SHOT – DUTCH ANGLE

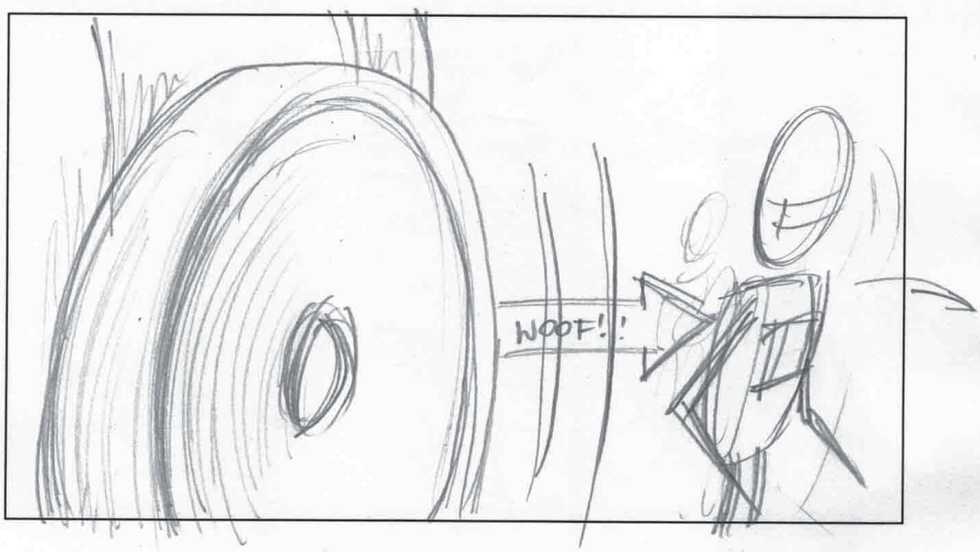


SHOT SETUP

SZENE - Nr.: 6.8 SHOT-DAUER: 2s

SET: _____

CAMERA SETUP: CLOSE UP - DRAMATIC ANGLE

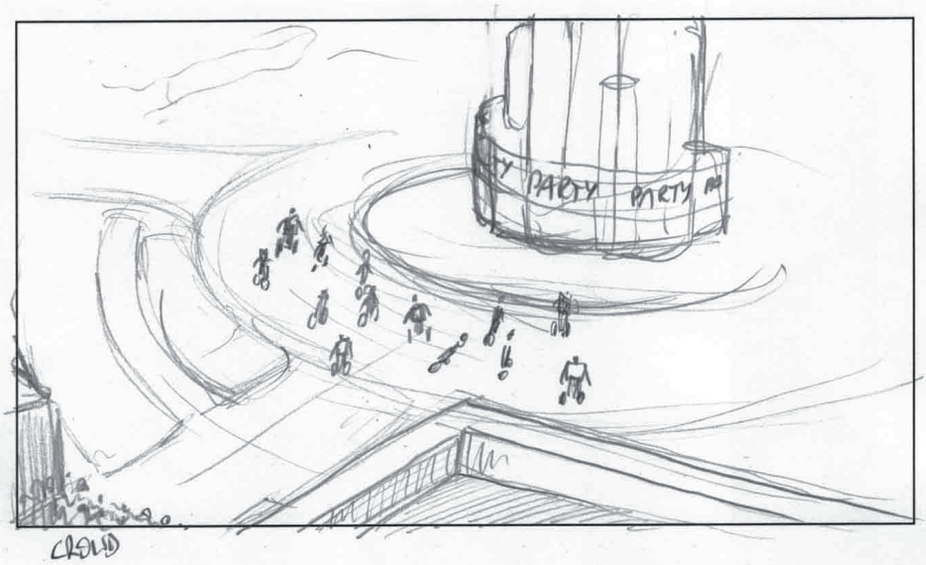


SHOT SETUP

SZENE - Nr.: 6.9 SHOT-DAUER: 6 s

SET: ROOFTOP / KREUZUNG

CAMERA SETUP: FULL SHOT - HIGH ANGLE



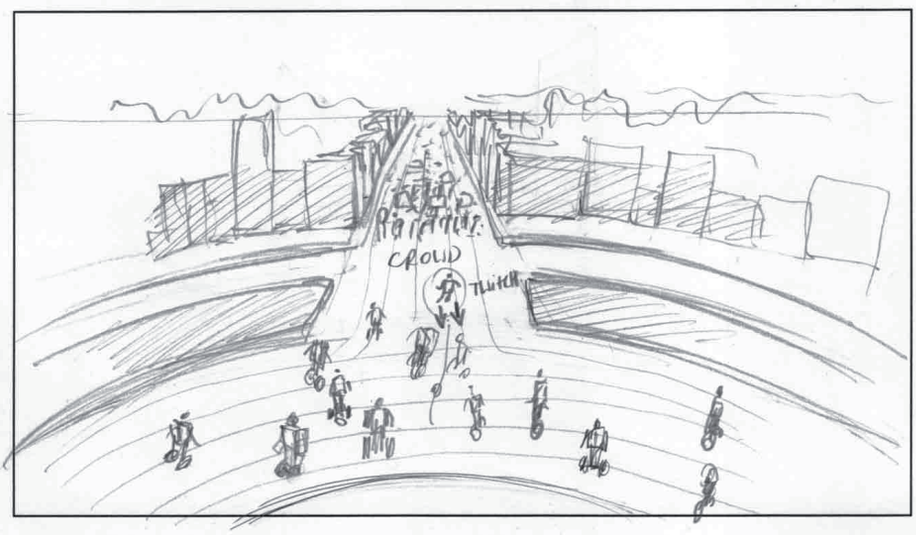
CLOUD

SHOT SETUP

SZENE - Nr.: 6.10 SHOT-DAUER: 4 s

SET: KREUZUNG

CAMERA SETUP: MEDIUM SHOT - GEGENSCHUSS

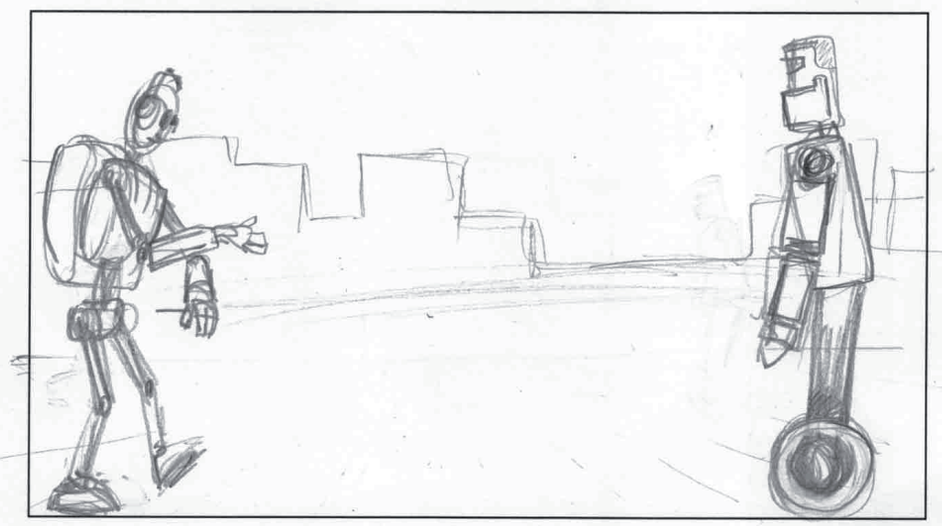


SHOT SETUP

SZENE - Nr.: 6.11 SHOT-DAUER: 15s

SET: KREUZUNG

CAMERA SETUP: FULL SHOT – PLANAR STAGING

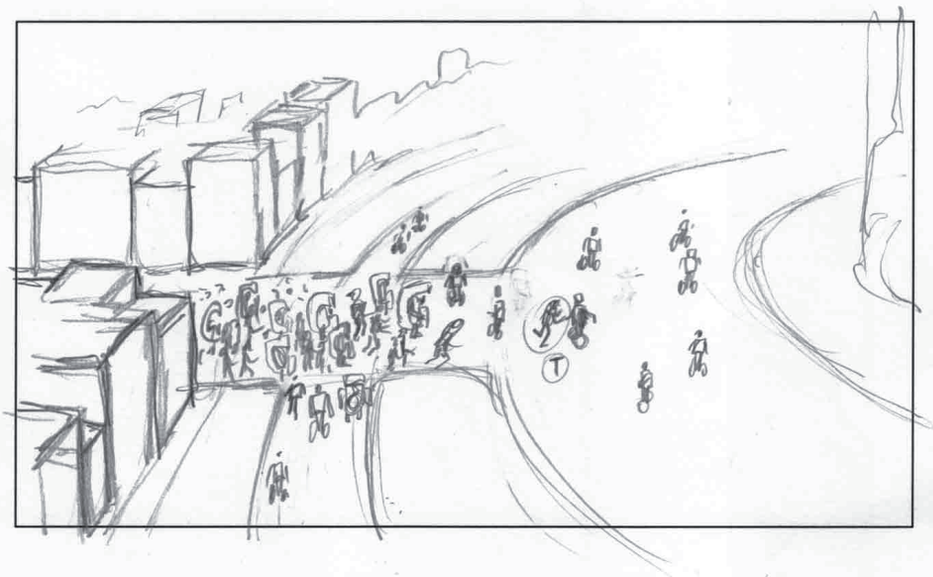


SHOT SETUP

SZENE - Nr.: 6.12 SHOT-DAUER: 5.1

SET: KREUZUNG

CAMERA SETUP: WIDE SHOT – CRANE UP

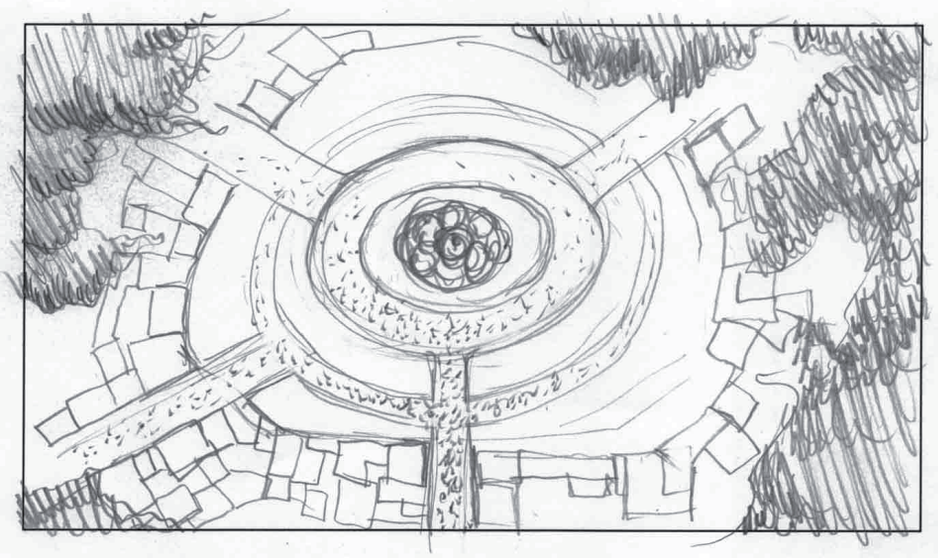


SHOT SETUP

SZENE - Nr.: 6.14 SHOT-DAUER: 10s

SET: KREATUR → AIR

CAMERA SETUP: MASTER SHOT - WIDE ANGLE - CRANE UP LOOK DOWN



SHOT SETUP