Darstellungsprototyp für 3D-Visualisationen zu Musik

Presentation prototype for 3D-visuals to music

Jan Swoboda

Bachelorarbeit an der
Freien Universität Berlin,
Institut für Informatik

Betreuer:
Prof. Dr. Lutz Prechelt
Dr. Alberto de Campo

Berlin, 2013

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bcher, Internetseiten oder hnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder hnlicher Form keiner anderen Prfungskommission vorgelegt und auch nicht verffentlicht.

4. Juni 2013,

Jan Swoboda

Zusammenfassung

In dieser Arbeit geht es um das Entwickeln eines Prototyps als VJ-Konzept. Das Ziel des Prototyps ist es, auf Basis einer Spiel-Engine einen Prototypen zu entwickeln, der als Framework dient 3D-Visuals zu erstellen und in einem Darstellungsprogramm, welches Teil des Prototyps ist, praktikabel performen können. Eine Soundanalyse-Komponente wird verwendet, um in den Visuals Animationen zu Klängen zu ermöglichen. Der Entwicklungsprozess teilt sich in Anforderungsanalyse, Entwurd und zwei Iterationsschritte für die Programmierung. Nach jedem Iterationsschritt wird der Prototyp anhand von Benutzertests hinsichtlich seiner Benutzbarkeit getestet und evaluiert. Die Ergebnisse aus dem ersten Test fließen als Verbesserungskriterien in den zweiten Version des Prototyp ein. Im zweiten Benutzertest werden die Auswirkungen der Verbesserungen bezüglich der Benutzbarkeit aus der zweiten Iteration untersucht.

Abstract

This work is about the developent of a prototype as a VJ concept. The purpose of the prototype based on a game engine is to be a framework for 3D visual creation and to display and perform those visuals in a practical manner in a performance tool wich is part of the prototype. A sound analysis component is used in order to be able to make visual animation depend on sound. The development process is divided requirements analysis, software draft and two iteration steps for programming. The Prototype is tested after each iteration to check and evaluate usability. Results from the first Test are influencing the second version of the prototype at usability aspects. The second user test shows the impact on usability after the second iteration.

Inhaltsverzeichnis

1 Einleitung

Seit langem spielt in der Musik auf Großveranstaltungen der visuelle Aspekt eine große Rolle. Neben Light Jockeys, die an Mischpulten Laser- und Lichtinstallationen steuern um eine Lichtshow zu erzeugen, beschäftigen sich Visual Jockeys im Kontext von Musik-veranstaltungen mit der Erzeugung von Visuals in Echtzeit. Visual Jockeys bedienen sich der Video- und Computertechnik, um aus visuellem Material wie Videoclips, Bildern oder 3D-Animationen eine Visualperformance auf Monitoren oder mit Videoprojektoren auf Großleinwänden zu realisieren. [1, pg 19-42] Das Mischen von Videoclips erfreut sich unter VJs von Anfang an großer Beliebtheit. Mit der Computerisierung entstanden Spezialprogramme für das Performen von Visuals. Viele dieser Programme konzentrieren sich dabei auf das Mischen von Videoclips. Das algorithmische Erzeugen von 3D-Animation als Performance in Echtzeit steht im Kontrast zu Performances basierend auf Videoclips. Es werden andere Werkzeuge für die Erstellung und Performance von Visualmaterial verwendet. Um Interaktivität mit Sound und 3D-Animationen zu ermöglichen, hängt die Benutzbarkeit eines VJ-Konzepts besonders von der Bedienbarkeit eines Darstellungsprogramms für die Visuals ab.

1.1 VJing

Unter VJing versteht man das Mischen von visuellem Material in Abstimmung zu Musik. Der VJ nimmt Rhytmus und Stimmung der Musik als Ausgangspunkt, um durch Bildkomposition die Gesamtwahrnehmung der Musik für die Audienz zu steigern. Ähnlich einem DJ, der die Musikauswahl aus seinem Repertoire trifft und mischt, wählt der VJ Visuals und mischt diese zur Musik.[1, pg 17-18]

Es gibt viele Möglichkeiten als VJ die Bildkompositionen in Echtzeit zu generieren. Der Computer hat sich dabei zur wichtigsten Plattform für die Erstellung und Darstellung von Visuals geworden. Letztendlich müssen Visuals präpariert werden, wofür Software zur Bild- und Videobearbeitung, sowie Programme zur 3D-Visualisierung und Animation verwendet werden. Ein Darstellungsprogramm für die präparierten Visuals stellt die Interaktion zwischen Visuals, und anderen Signalströmen wie Sound und Hardwareeingaben, her. Oft benutzen VJs eine Kombination aus spezieller Hardware und Software für die Interaktion mit den von ihnen erzeugten Visuals. Auch kommen teilweise selbstgebaute Eingabegeräte oder MIDI-Controller zum Einsatz, um schnell und mit Gefühl

auf die Musik zu reagieren und Änderungen an der Bildkomposition vorzunehmen. Dafür steuern Tasten und Regler Funktionen der Software. Der VJ entwickelt dafür sein eigenes Konzept.[3]

1.2 Motivation

Ein Gros der erhältlichen Software speziell für VJs ist proprietär. Die Ansätze für die Umsetzung von 3D-Visuals hängen von dem Funktionsumgang der Softwarelösungen ab. Die von VJs häufig eingesetzten Spezialprogramme sind hauptsächlich mit dem Gedanken an das Performen von Videoclips entstanden, sodass 3D-Animationen in den VJ-Programmen nur über Videoclips realisierbar sind. Mit Programmen aus der Videobearbeitung und 3D-Animation werden Videoclip mit 3D-Animationen erstellt. Dabei gibt es in der Computergrafik andere Wege, eine 3D-Animation zu erstellen und in Echtzeit auf Abruf auszuführen.

1.3 Ziel

In dieser Arbeit soll ein Konzept für das Erstellen und Performen von Visuals mithilfe einer Spiel-Engine erarbeitet werden, welches anhand eines Prototyps vorgestellt wird. Der Bedienung für das Performen von Visuals in einem 3D-Raum kommt große Bedeutung zu, da sie maßgeblich für die Interaktion des VJs mit seinen Visuals definiert. Die Visuals sollen aus Programmobjekten bestehen, deren Parameter während der Performance in Echtzeit verändert werden können. Dazu wird mittels Benutzertests die Bedienbarkeit geprüft und evaluiert. Mithilfe der Einsichten aus den Benutzertests soll in einem zweiten Iterationsschritt das Konzept überarbeitet werden um die Benutzbarkeit zu verbessern. Das Konzept soll dem VJ ein Framework für das Erstellen von Visuals und ein Werkzeug für das Performen mithilfe einer Spiel-Engine bereit stellen. Videoclips sollen unter dem Konzept auch berücksichtigt werden und zum Medienrepertoire für Visualideen gehören.

Im Rahmen einer Anforderungsanalyse sollen die Anforderungen an ein 3D-Darstellungsprogramm für Visuals ermittelt und in einem Entwurf für einen Prototypen berücksichtigt werden. Der Entwurf soll dabei helfen einen Prototypen zu entwickeln, der das Konzept umsetzt. Benutzertests sollen Mängel im Prototyp aufzeigen, welche in einer zweiten Iteration beseitigt werden sollen.

2 Anforderungsanalyse

Die Anforderungen an das VRC werden maßgeblich von der dreidimensionalen Characteristik des Konzepts beeinflusst.

2.1 Hardware und Software

Die Anforderungen an Hardware für das VRC orientieren sich an der Software. Durch den 3D-Raum muss auf Hardware geachtet werden, welche die Szenerie schnell genug darstellen kann. Deshalb sollte der Rechner, auf dem das Programm ausgeführt wird eine hardwareseitige 3D-Beschleunigung besitzen. Heutzutage haben fast alle Rechner eine hardwareseitige Unterstützung für OpenGL [5] oder DirectX, jedoch eignet sich nicht jede dazu eine Szenerie aus vielen Polygonen schnell genug darzustellen.

Hardware Da die Software dazu verwendet werden soll, 3D-Animationen und Effekte zu Musik zu performen, sollte der Computer neben der CPU mindestens einen dedizierten Grafikchip und dedizierten Grafikspeicher besitzen. Um von zu den Performances nicht so viel transportieren zu müssen sollte der Rechner ein Laptop sein. Programmierung der Visuals und Bedienung des Programms erfordern eine Tastatur. Die Soundaufnahme kann das interne Mikrofon des Laptops übernehmen

Software Die Software für das Konzept sollte ausschließlich Open-Source sein. Dabei ist darauf zu achten, dass die Software stabil ist und gut unterstützt wird. Die Software sollte nicht an eine Plattform gebunden sein und leicht auf andere Betriebssysteme portierbar sein. In dem Konzept werden Visuals zwar programmiert, jedoch sollte die Programmierung nicht allzu maschinennah sein. Eine abstrakte Sprache ist für die Verständlichkeit und Erlernbarkeit des Konzepts essentiell. Bei der ausschließlichen Verwendung von nativem OpenGL ist die Einstiegshürde selbst für VJs mit Programmiererfahrungen zu hoch. Daher ist es besser, ein Framework für OpenGL zu verwenden, mit dem leicht Animationen und Szenerien programmiert werden können. Die Softwarearchitektur sollte nachträgliche Erweiterungen nicht ausschließen.

2.2 Bedienbarkeit

Für die Bedienung sollen Standardeingabegeräte wie Tastatur und Maus genügen. Vor allem die Einstellung der Ansicht und Positionierung der Visuals in einem 3D-Raum muss auf einer Tastatur komfortabel möglich sein. Überblendungen und Visualwechsel beinhalten viele Aktionen wie das Ändern von Transparenzen oder Skalierungen und muss flüssig machbar sein. Die Ansicht durch eine virtuelle Kamera erfordert eine bewegliche Kamera. Bei Rotationen der Kamera ist es wichtig, die Orientierung nicht zu verlieren.

Das Anzeigen von Statusinformationen von Kamera und Einstellen von Soundanalyse und aktiven Visuals, sowie die Übersicht der verfügbaren Visuals erfordert eine geeignete GUI mit Panelen, an denen Parameter abgelesen und geändert werden können.

Als nichtfunktionale Anforderung steht vor allem schneller gezielter Zugriff auf Visuals und Kamera, sowie die Manipulation dergleichen im Vordergrund, um auf Änderungen in der Musik reagieren zu können.

2.3 Visuals

In diesem VJ-Konzept ist der Fokus auf eine liberale Visualgestaltung ausgerichtet. Operationen wie *Verschieben*, *Skalieren*, *Rotieren* und andere Transformationen zu Musik sollen jedem Visual aufgrund seines dreidimensionalen Charakters als Objekt in einem 3D-Raum möglich sein. Der Sound soll von Visuals genutzt werden können, um Effekte auszulösen oder zu beeinflussen.

Vor allem bei der Auswahl der Werkzeuge für die Visualserstellung ist wichtig, dass verschiedene Medien unterstützt werden. So soll der VJ Visuals aus Bilddateien, Videoclips als auch 3D-Modellen erstellen können. Für die Erstellung von 3D-Modellen soll 3D-Grafiksoftware verwendbar sein. So muss man geometrische Objekte nicht händisch als Punktkoordinaten schreiben, sondern kann eine Modelldatei ins Visual laden.

In einer Funktion nimmt das Visual die Soundwerte entgegen und kann entscheiden, was dazu passieren soll. So kann der Sound genutzt werden, um Tranformationen wie Skalierung oder Transparenz zu beeinflussen.

Eine *Namenskonvention* für einen Teil der Visualfunktionen ist nötig, um die Effekttasten an Funktionen zu binden. Beim Drücken einer Effekttaste soll für das aktive Visual eine selbstprogrammierte Effektfunktion ausgeführt werden können.

2.4 Darstellung und Performing

Das Performen stellt spezielle Anforderungen an die Übersichtlichkeit [2, pg 17]. Im virtuellen 3D-Raum werden Visuals angeordnet und gesteuert. Da für die Bedienung zum Performen ausschließlich Standardeingabegeräte benutzt werden, ist ein auf Tastatur und Maus zugeschnittenes Bedienkonzept nur mit einer übersichtlichen Gestaltung des Darstellungsprogramms realisierbar, das eine schnelle Reaktion innerhalb des Programms erlaubt.

Eine grafische Oberfläche hilft bei der Übersicht über den Zustand der Visuals und der Kamera, sowie der Soundeinstellung. Die grafische Oberfläche soll dabei Positions und Rotationsdaten anzeigen. Werte wie Transparenzen, Skalierung und Geschwindigkeit, sollen über Bedienelemente mit der Maus einstellbar sein. Aus Listen sollen Visuals auswählbar sein, um in den 3D-Raum geladen oder entfernt zu werden. In der GUI sollten alle Elemente schnell zugreifbar und übersichtlich angeordnet sein.

Das Performen der Visuals besteht aus dem Laden und Anordnen von Visuals. Bei Nichtverwendung von Visuals, werden diese aus dem Raum entfernt. Die Interaktion des VJs mit dem 3D-Raum besteht aus dem Ändern von Parametern und Werten der Visuals und der Kamera. Um mit den Visuals und der Kamera schnell auf Änderungen in der Musik zu reagieren, muss die Bedienung schnell und einfach zum Ziel führen. Durch das Betrachten der Szenerie mit einer virtuellen Kamera gibt der VJ die Visualität der 3D-Raums wider.

Die Ausgabe des 3D-Raum geschieht dabei separat auf einem anderen Fenster. Die Ausgabe beinhaltet nur das, was durch die Kamera vom 3D-Raum sichtbar ist, ohne Zusatzinformationen aus der grafischen Oberfläche oder dem Fenster, das für die Steuerung der 3D-Szene genutzt wird. Das Fenster zur Steuerung der Visuals und der Kamera dient gleichzeitig als Vorschaufenster. Dafür soll es möglich sein, in dem Vorschaufenster einen anderen Teil der Szenerie betrachten zu können, als auf dem Ausgabefenster aktuell zu sehen ist, um Schnitte vorzubereiten oder an einer anderen Stelle eine andere Szenerie aufzubauen.

Das Soundsignal kann über das in modernen Laptops enthaltene Mikrofon entnommen und anschließend verarbeitet werden. Dazu muss das Soundsignal beim Performen kontinuierlich analysiert werden, um rechtzeitig Beats zu erkennen, auf die selbstgeschriebene Funktionen in den Visuals reagieren können.

3 Entwurf

Beim Entwurf müssen Überlegungen getroffen werden, welche die Auswahl der Software im Vorfeld auf die in Frage kommenden Plattformen und nötige Programmbibliotheken reduziert. Für die Implementierung wichtige Datenstrukturen müssen so gewählt sein, dass man zur Laufzeit den Variablenzustand von Sound- und Bedienungseingabe geschickt an allen relevanten Stellen abrufen und ändern kann. Abbildung 3.1 zeigt einen groben Komponentenentwurf.

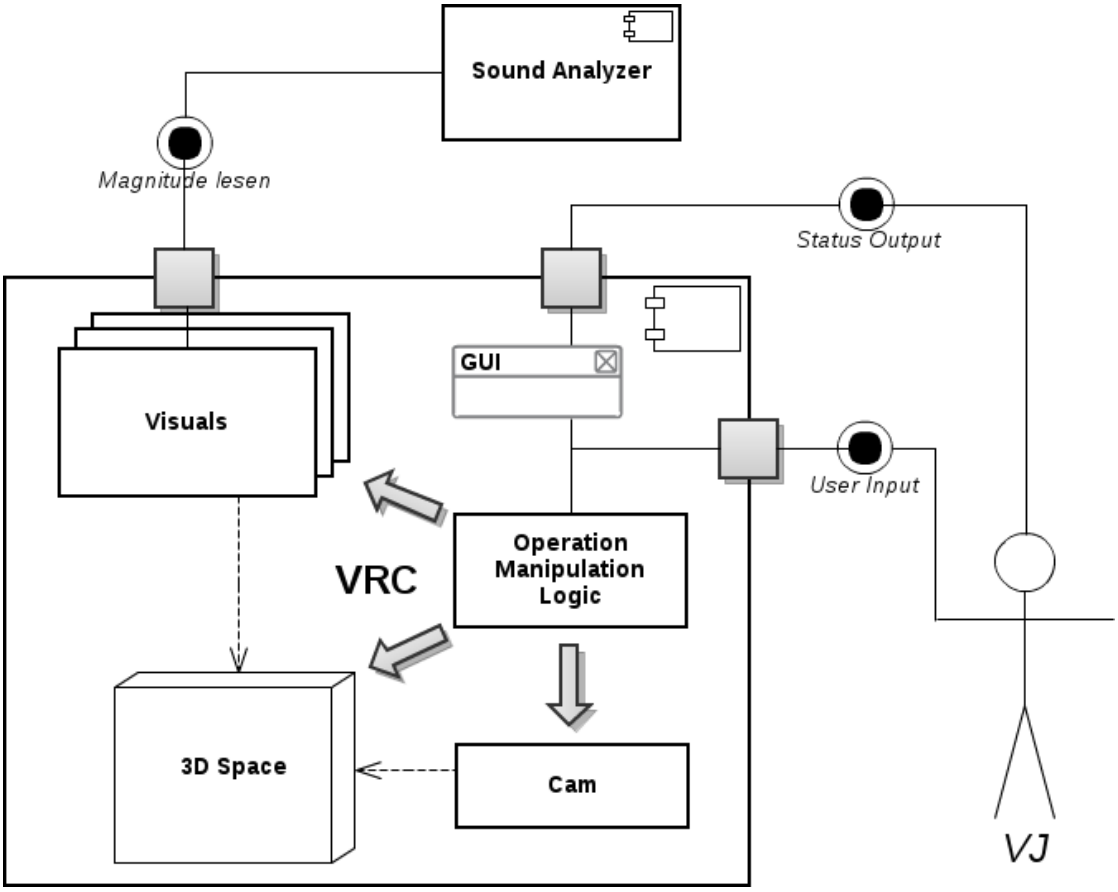

Abbildung 3.1: Strukturdarstellung des VJ-Konzeptprogramms VRC, bestehend aus Soundeingabe und einem Container für 3D-Raum und Visuals welche von einem User für die VJ-Performance genutzt werden.

Der Entwurf dient als Ideengrundlage, anhand der man mit einer Spiel-Engine das VRC-Konzept implementieren kann. Dabei ist wichtig, dass der Entwurf möglichst alle Anforderungen umsetzbar löst.

3.1 Überlegungen

Spiel-Engines als Programmiergerüst erscheinen vorteilhaft um 3D-Grafiken in Szene zu setzen [6]. Durch die Beschränkung auf Standardeingabegetäte sind Spiel-Engines hervorragend für die Verarbeitung von Benutzereingabe und Steuerung des 3D-Raums geeignet. Auch das Erstellen von Visuals erfährt durch die Verwendung von Spiel-Engines Vorteile. Elemente werden in Visuals gruppiert und mit Funktionen unter Einbezug verschiedener Parameter modifiziert/transformiert. Anhand des Soundsignals können ausgewählte Parameter dem musikalischen Zufall überlassen werden. Dadurch wird eine akustisch-visuelle Koppelung zwischen Visual und Umgebung möglich. Die Soundeingabe soll möglichst unabhängig von technischen Faktoren möglich sein. Laptops haben meist schon standardmäßig ein Mikrofon eingebaut, das man als Standardeingabegerät bei Performances nutzen kann.

Objektorientiertes Programmieren ermöglicht das Vererben und Überladen von Variablen und Funktionen. Allgemeine Variablen für den Zustand eines Visuals können in einer Klasse definiert werden und an das Visual vererbt werden. Genauso können auch allgemeine Funktionen für das Bewegen und Rotieren im 3D-Raum vererbt werden. Zusätzliche Funktionen für individuelle Animationen und Effekte können vom VJ selbst im Visual programmiert werden. Fortgeschrittene VJs können Funktionen der Visual-Klasse bei Bedarf auch überschreiben.

Mit dem Mikrofon hat man unabhängig von anderen Künstlern die Möglichkeit direkt den Sound aufzunehmen, ohne von anderen Faktoren wie Kabeln und abhängig zu sein. Um Beats zu erkennen, sollte das Mikrofon im Laptop reichen.

Nicht jeder VJ besitzt gute Programmierkenntnisse. Eine einfache und populäre Sprache kann den Einstieg in das Konzept für VJs attraktiver machen. Python als Programmiersprache erfreut sich einer großen Beliebtheit. Mit Python kann das VRC Anreiz für VJs sein, ihre Programmierkenntnisse zu vertiefen, indem sie lernen müssen, Visuals zu erstellen. Das Konzept verfolg unter anderem das Ziel, ein Programmierframework für Visuals zu sein. [7]

3.2 Spiel-Engine

Eine Spiel-Engine dient als Framework für die Implementierung des Konzepts, da sie sich besonder für die verschiedenen anfallenden Aufgaben eignet. So enthalten Spiel-Engines oft einen Taskmanager, der sich wiederholende Aufgaben ausführt. Spiel-Engines ermöglichen es, sich neben der Grafikprogrammierung vor Allem auf die Programmlogik zu konzentrieren [6].

Spiel-Engines besitzen Eigenschaften, die für die Realisierung der Virtual-Room-Concept-Programms essentiell sind. Vor allem bieten sie eine Datenstruktur zur Unterbringung der 3D-Geometrie - meist einen Szenengraphen in Baumstruktur. Auf den Szenengraphen wird über eine Schnittstelle in der Spiel-Engine zugegriffen. Ein Task-Manager, der kontinuierlich die Benutzereingaben und die Soundanalyse verwaltet, erleichtert das Programmieren der Ablauflogik innerhalb des VRC-Programms (citation here). Zum einen mussen die Visuals stets mit Soundwerten versorgt werden, zum Anderen müssen die Eingaben des Benutzers entgegengenommen und ausgeführt werden.

Für die Visualprogrammierung ist die Wahl einer Spiele-Engine vorteilhaft, da man eine vereinfachende Zwischenschicht für das Zusammenstellen von Geometrie und 3D-Modellen in den Szenengraphen hat. Auch für das Performen eignen sich Spiele-Engines gut und besitzen dank des Szenengraphen und seinen Schnittstellen eine einfache Möglichkeit Visuals zu laden und zu entfernen. Die Betrachtung der Szenerie durch eine virtuelle Kamera ist eine grundlegende Anforderung an Spiel-Engines (citation), sowie an das VRC-Programm. [6]

Das Angebot an freien Spiel-Engines ist relativ groß, jedoch sollte die Spiel-Engine so gewählt werden, dass sie sich gut mit den anderen Komponenten des VRC-Programms und ihren Bibliotheken verwenden lässt.

Neben vielen C++- und Java-Spiel-Engines existieren auch für viele andere Sprachen Spiel-Engines. Für das VRC ist entscheidend, dass die Spiel-Engine einen großen Benutzerkreis hat damit ein breites Interesse auf Seiten der Benutzer und Entwickler besteht, sodass die Spiel-Engine auch in Zukunft aktualisiert und weiterentwickelt wird.

Python erfreut sich in der 3D-Gestaltung einer hohen Popularität. Programme wie Blender[1] und auch Modul8[2] besitzen eine Python-Sprachanbindung. Blender verfügt sogar über eine eigene Spiel-Engine in Python.[3] Mit Python als Sprache für eine Spiel-Engine erleichtert man durch die relativ einfache Syntax VJs den Einstieg in das Konzept. Eine gute Dokumentation der Sprache und der Spiel-Engine hilft bei der Erstellung des Prototyps und der Visuals enorm. Anhand dieser Aspekte lässt sich die Wahl einengen

[1] Freies 3D-Modellierungsprogramm: `http://www.blender.org/`
[2] Proprietäres VJing-Programm: `http://www.modul8.ch/`
[3] `http://www.blender.org/features-gallery/features/`

in eine OpenGL-basierte Spiel-Engine für Python. Bekannte Vertreter von Spiel-Engines für Python sind jMonkeyEngine, Blender oder Panda3d.[4]

Weitere Eigenschaften von Spiel-Engines, die für das VJing Vorteilhaft sein können:

- Konvertierungswerkzeuge für viele 3D-Modellingtools
- Shader-Generatoren für Spezialeffekte
- Kollisionsdetektion und Unterstützung eines Physik-Systems

3.2.1 Datenstrukturen

Um die Visuals im Programm vorzuhalten, kann eine Liste mit Verweisen auf die Visuals verwendet werden. Für die Visual-Elemente muss die Spiel-Engine eine Datenstruktur vorhalten, in den die Elemente hineingeladen werden können. Dafür eignet sich eine Baumstruktur ganz gut. Der 3D-Raum stellt die Wurzel des Baums dar, jedes Visual ist ein Kind des 3D-Raum-Wurzelknotens. Ein Visual besteht aus einem Knoten, der als Wurzel für die Visual-Elemente fungiert (siehe Abbildung 3.2).

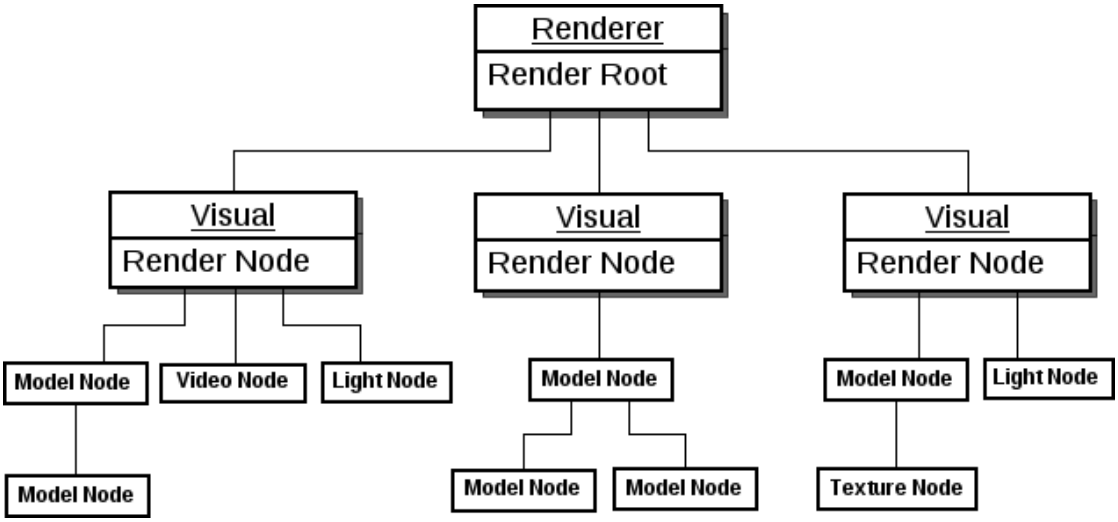

Abbildung 3.2: Skizze eines Szenengraphen, mit geladenen Objekten der Visuals, wie 3D-Modelle, Texturen, Videos.

[4]Suche nach Open-Source Python-Spiel-Engines `http://devmaster.net/devdb/engines` (Zugriff 2013-07-03)

3.3 Soundanalyse

Es gibt verschiedene Möglichkeiten Sound zu analysieren. Bei diesem Entwurf wird Wert darauf gelegt, dass Soundwerte auf einfache Art in den Visuals verarbeitet werden können. Es sollen Bässe, Mitten und Höhen als positive Gleitkommawerte abgelegt werden.

Die Soundanalysekomponente muss kontinuierlich die Soundeingabe verarbeiten. Das aufgenommene Frequenzband wird unterteilt in Bassband, Mittelband und Höhenband. Das Bassband erstreckt sich über die Frequenzen 0Hz bis 100Hz, die Mitteltöne über 100Hz bis 1000Hz, und die Hoehen über 1000Hz bis 22050Hz. Für jedes Band wird eine Magnitude aus den Samples berechnet, welche die Intensität auf den drei Bändern angibt. Dem Mikrofonsignal werden kontinuierlich Samples entnommen, um die Magnitude über den Frequenzbändern zu messen und in Soundzustandsvariablen zu speichern. Das Soundsignalsample ist ein Vektor aus 16Bit Integer-Werten - für jede Frequenz ein Integer-Wert.

Eine einfache Methode die Musik zu analysieren und einen Beat zu erkennen ist beispielsweise das Vergleichen der Magnitude mit einem Grenzwert. Liegt die Magnitude über dem Grenzwert, so kann das als Beat interpretiert werden, worauf eine Aktion folgen kann.

3.4 Bedienung

Da in den Anforderungen die Benutzung von Standardeingabegeräten gefordert ist, stellt vor allem die Tastatur zur Steuerung von Visuals eine gute Plattform dar. Das Virtual Room Concept verhält sich von der Navigion im 3D-Raum her ähnlich dem Bewegen von Aktoren in Computerspielen. In Computerspielen ist das, was der Spieler sieht, eine Ansicht einer virtuellen Kamera auf die ihn virtu-ell umgebende Szenerie. Eine Grundlage für Computerspiele ist das Bewegen des Agenten in einer Spiellandschaft, was dem Bewegen einer virtuellen Kamera in einer Szenerie in einem 3D-Raum entspricht.

Viele Spiele, allen voran First-Person-Spiele (FPS) haben oft für das Bewegen des virtuellen Agenten standardmäßig die selbe Tastenbelegung. Auch andere Funktionen wie die Auswahl von Gegenständen ist, standardmäßig oft ähnlich belegt: die Tasten "w", ä", ß", "d", so wie ßpace"werden häufig in Kombination mit der Maus für die Bewegung, des Agenten genutzt, die Zahlentasten bedienen Gegenstände oder führen spezielle Aktionen aus.[5] In diesem Entwurf wird die Bedienung von FPS für die Kamera und Visuals adaptiert. Für die Rotationen wird auch die Tastatur verwendet. Hierfür werden

[5] Forumsdiskussion `https://thetfp.com/tfp/tilted-gaming/103390-fps-keyboard-layout.html#axzz2XzqaKI3y` (Zugriff 2013-07-03)

zusätzlich Belegungen definiert. Diese unterscheiden sich von denen in Computerspielen. In Computerspielen wird für die Steuerung des Nick- und Gier-Winkel häufig die Maus verwendet. Da für Kamera und Visual jeweils Rotationen als auch Bewegungen entlang der drei Achsen im 3D-Raum möglich sein sollen, wird die Bedienung für diese Aktionen vereinfacht, indem die selbe Belegung für das Bewegen und Rotieren für Visual als auch Kamera gilt.

Die Abbildung 3.3 zeigt den Entwurf für eine mögliche Tastenbelegung. Um gezielt auf

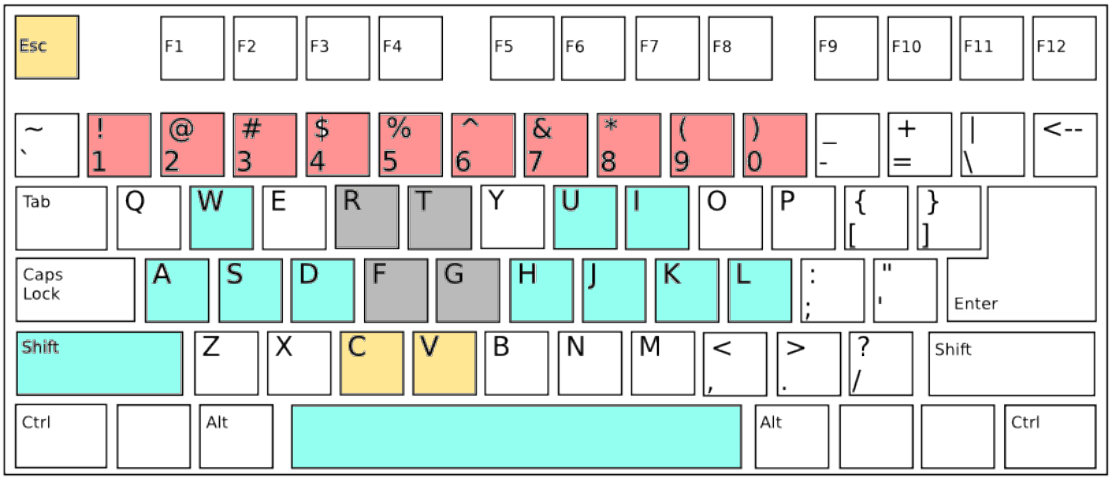

Abbildung 3.3: Skizze eines Szenengraphen

einem Visual oder der Kamera operieren zu können, werden Bedienmodi für Visuals und Kamera definiert. Befindet man sich im Visual-Bedienmodus, können nur Visuals gesteuert werden, im Kamera-Modus nur die Kamera. Während die Bewegungs- und Rotationstasten in beiden Modi das gleiche bewirken, braucht man für Effekte im Visual-Modus zusätzliche Tasten. Auch die Kamersteuerung, um die Ausgabekamera mit der Vorschaukamera zu synchronisieren oder unabhängig von einander zu bedienen benötigt eine Bedienmöglichkeit. Durch die Bedienmodi ist es möglich verschiedene Bedienfunktionen mit einer Taste zu belegen.

Die Maus soll vor allem genutzt werden, um die grafische Oberfläche zu bedienen.

3.5 Grafischen Oberfläche

In der grafischen Oberfläche werden Daten aus dem Zustand des VRC aufbereitet, um Statusinformationen über die Visuals und Kamera anzuzeigen. Außerdem werden über die grafische Oberfläche die Visuals in den 3D-Raum hinzugefügt oder entfernt. Wichtige

gemeinsame Funktionen der Visuals können in der grafische Oberfläche zusammengefasst und der Manipulation durch den Benutzer zugänglich gemacht werden.

Es müssen die wichtigen Entitäten zum VJing schnell abrufbar sein. Gleichzeitig hat man aber nur begrenzten Platz auf dem Bildschirm, um Vorschaufenster und Menus unterzubringen. Die Bedienelemente für die verschiedenen Komponenten würden alle zusammen auf einer Fläche chaotisch wirken. Deshalb können sie über ein Reitermenü nach Komponenten gruppiert dargestellt werden. Die verschiedenen Programmkomponenten Sound, Visuals und Kamera haben auch ihre eigenen Anforderungen an Bedienschnittstellen. So haben Visuals eine Schnittstelle für Skalierung und Transparenz, die Kamera aber nicht. Da es sich um eine überschaubare Menge an Komponenten handelt ist ein Reiterlayout für den VJ beim VJing ausreichend, um Kontrolle über den Status des Programms während der Laufzeit zu behalten. Abbildung 3.4 zeigt eine Layoutskizze, an der sich im VRC für das Darstellungsprogramm orientiert wird. Das Resultat im Vorschaufenster

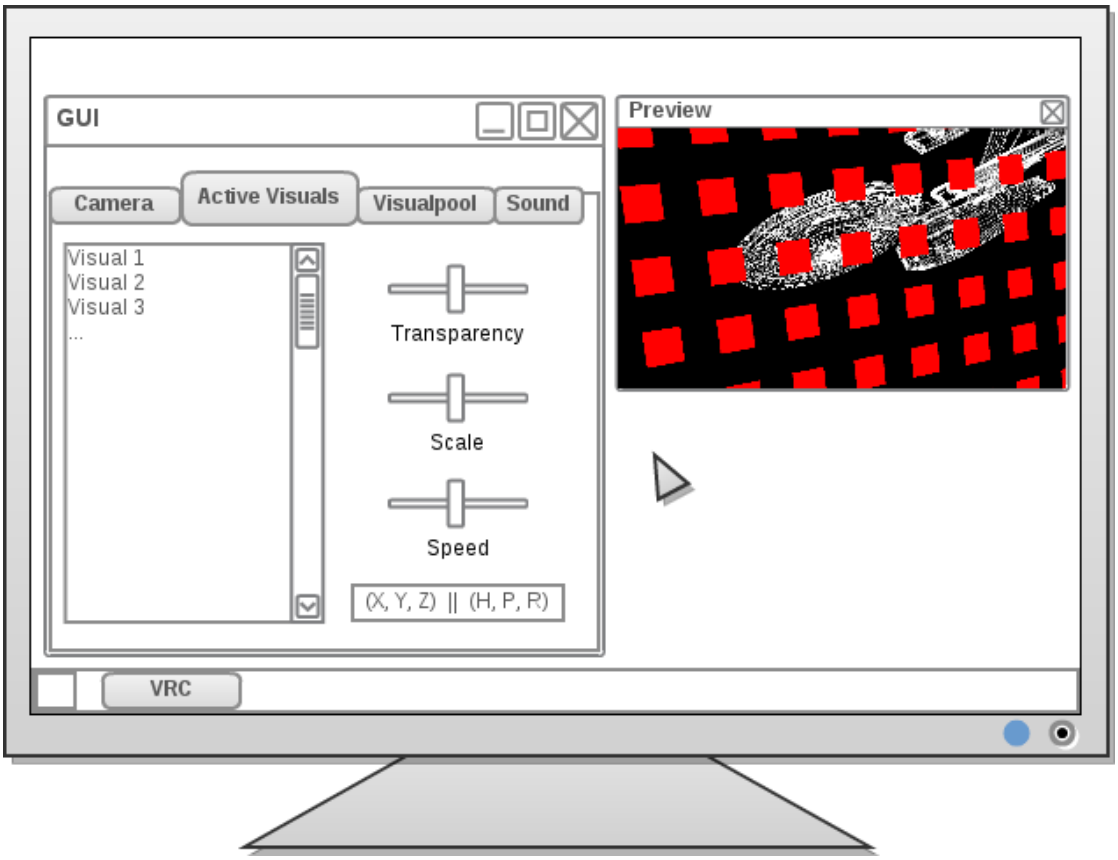

Abbildung 3.4: Skizze der GUI auf einem Monitor als Fenster für das Virtual Room Concept und die Vorschau

wird über einen zweiten Videoausgang ohne Fensterrahmen auf einem externen Ausgabegerät angezeigt.

3.6 Visual-Klasse

Visuals besitzen durch ihre Anforderungen an Beweglichkeit, Skalierung und andere gewünschte Manipulationsmöglichkeiten eine Menge an Funktionen, die für jedes Visual verwendet werden können. Da Visuals als Objekte gedacht sind, lässt sich deren Charakterisierung von einer Klasse ableiten, wie in der Abbildung 3.5 am Vererbungsmuster verdeutlicht wird.

Durch die richtige Initialisierung und einer Standardeinstellung ist das Überschreiben und Erweitern der Funktionen der Visual-Klasse im Visual-Objekt möglich. Die Visual-Klasse besitzt einen Konstruktor, mit dem die Initialisierung für das VRC-Programm vorgenommen wird. So wird der Knoten, an den alle weiteren 3D-Elemente geknüpft werden, erstellt. Referenzen auf andere Komponenten des VRC-Programms werden angelegt. Weiter werden Funktionen für die Bewegung und Rotation der Visuals schon in der Visual-Klasse definiert. Mit diesen Funktionenen ist es möglich, die Visuals im 3D-Raum anzuordnen.

Diese Eigenschaften können von Visuals geerbt werden und stehen ihnen dann zur Verfügung. Mit der Verwendung einer Visual-Klasse erreicht man, dass sich Visuals trotz individueller Programmierung einheitlich in das VRC-Programm einfügen lassen und steuerbar sind.

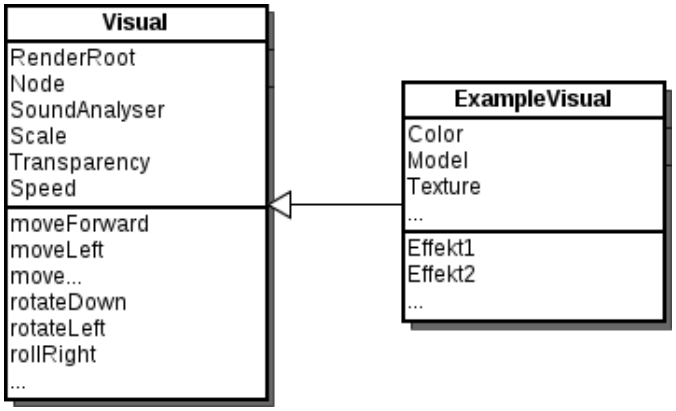

Abbildung 3.5: Vererbungsmuster Hauptvisual-Klasse und Visual. Die Hauptvisualklasse implementiert für VRC benötigten Funktionen

4 Prototyp

Der Prototyp für das Konzept orientiert sich an den Ideen aus dem Entwurf, um die Anforderungen an das Konzept umzusetzen. Da im Entwurf noch nicht klar ist, welche Software zum Einsatz kommt, wird während des Entwicklungsprozesses nach Software gesucht, mit der das Konzept umzusetzen ist.

4.1 Softwarewahl

Nach erfolglosen Versuchen mit Processing[1] einen Prototypen umzusetzen begann die Suche nach Alternativen. Erst wurden noch javabasierte Bibliotheken für OpenGL ausprobiert. Nachdem aber schnell klar wurde, dass keine der angetesteten Bibliotheken in der Lage war, simpel und komfortabel zwei Fenster mit dem selben Inhalt darzustellen, fiel der Fokus auf Spiel-Engines. Eine gute Dokumentation und eine gut aufgestellte freie Entwicklergemeinschaft sind wichtig für den Erfolg des VRC-Projekts. Durch eine gute Dokumentation ist es leichter, die verwendete Software zu erlernen, dank einer aktiven Benutzer- und Entwicklergemeinschaft werden spezifische Fragen in Internet-Foren oder Chat-Räumen schnell beantwortet.

Letzendlich fiel die Wahl der Game-Engine auf Panda3d[2], da Panda3d Open-Source ist und sich für schnelles Prototyping eignet [4]. Mit Panda3d wurden zu Beginn kleine Testprogramme erstellt, mit denen die Anforderungen an Funktionalität ausgetestet werden. So wurde erst einmal versucht die Ausgabe des Programms auf einem undekorierten Fenster auf ein Ausgabegerät in Form von Videoprojektor auszugeben. Als diese Funktionalität durch kleine Testprogramme erreicht worden war, ging es darum zu sehen, wie sich die Ansicht auf die 3D-Szenerie bedienen lässt. Dazu wurde ein kleines Programm mit ein paar geladenen Models geschrieben und eine Kamera hineingesetzt. Nachdem erfolgreich die Kamerabedienung programmiert worden war, waren das die Kriterien für die Entscheidung, bei der verwendeten Spiel-Engine zu verbleiben und darauf aufzubauen. Wichtig bei der Wahl der Spiel-Engine war auch, dass sie nicht veraltet ist und Techniken wie Shaderprogrammierung, Technologien für Schatten und Licht unterstützt und eine klare Syntax hat. Panda3d enthält auch eine Physik-Engine und einen Taskmanager.

[1] Javabasierte Programmiersprache für 2D/3D Ausgabe `http://processing.org` (Zugriff 2013-07-03
[2] Spiel-Engine für C++ und Python `http://python.org` (Zugriff 2013-07-03)

Panda3d bietet die Möglichkeit verschiedene Wrapper für GUI-Toolkits zu verwenden, darunter wxPython und TKinter [4]. Tkinter eignet sich zum schnellen Prototypenbau für grafische Oberflächen und ist im Python-Sprachpaket direkt enthalten.[3] Für den VRC-Prototypen wurde die Tkinter-Bibliothek gewählt.

Für die Soundanalyse-Komponente ist wichtig, ohne große Schwierigkeiten die Hardware anzusprechen und den Sound darüber auszulesen. Bei der Berechnung der Magnituden muss man über einen Vektor aus 22050 natürlichen Zahlen iterieren, welche die Samples des Signals sind. Dabei muss man die Laufzeit beachten. Es gibt Bibliotkeken für Python zum Auslesen des Signals über Pulse-Code-Modulation (referenz auf python-alsa*) Einen Algorithmus zur Berechnung der Magnitude auf den Frequenzbändern der tiefen, mittleren und hohen Töne konnte aus einem C++-Programm übersetzt werden, welches eine änliche Soundanalyse-Komponente verwendete (reference auf liveGL von phreax)

4.2 Datenstrukturen

Innerhalb der Testprogramme wurden Datenstrukturen für die Bedienung und das Handhaben der Visuals verwendet. Dabei erwies sich assoziatives Datenfeld als geeignet, um die Tastatureingabe entgegenzunehmen und für ein Bedienschlüsselwort zu speichern. Panda3d verarbeitet Tastendruck- und Tastenfreigabe-Events. Die Tastenzustände lassen sich binär als 1 und 0 darstellen, wobei 1 gedrückt, und 0 nicht gedrückt bedeutet. Um nun für eine Operation den Zustand zu speichern wird die Tastatur zyklisch abgefragt. Wenn eine Taste gedrückt wird und einer Operation entspricht so wird der Wert dieser Operation auf 1 gesetzt, wie anhand des Codebeispiels gezeigt.

```
if key == "a": setOperationMap('move-forward', 1)
if key == "a-up": setOperationMap('move-forward', 0)
```

So wird der Zustand der Eingabe an einer Stelle im Speicher vorgehalten und ist in anderen Teilen des Prototypen leicht abrufbar.

In einer Liste werden die Verweise auf geladene Visuals vorgehalten. Dadurch können Funktionen leicht auf allen geladenen Visual-Objekten ausgeführt werden, wie beispielsweise das Versorgen mit den Sound-Messwerten.

Der Szenengraph ist eine baumartige Datenstruktur. Panda3d besitzt Schnittstellen um Knoten und deren Unterknoten aus dem Baum zu entfernen und an einer anderen Stelle im Szenengraphen anzuhängen. So können Visuals, die nicht mehr performt werden sollen aus dem Graph entfernt werden wodurch Rechenressourcen frei werden.

[3] Python Wiki `http://www.cs.mcgill.ca/~hv/classes/MS/TkinterPres/#Overview` (Zugriff 2013-07-03)

4.3 Software-Architektur

Das Zusammenführen von Visuals, Sound-Messwerten und GUI erfordert eine durchdachte Architektur für das VRC-Programm. Die erforderlichen Komponenten lassen sich zu Klassen zusammenfassen. Das Klassendiagramm (siehe Abb. 4.1) zeigt die Assoziationen der Klassenobjekte untereinander.

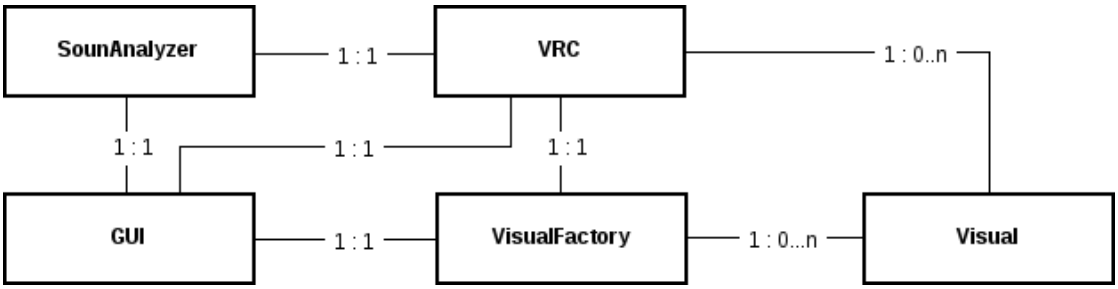

Abbildung 4.1: grobes Klassendiagramm mit Assoziationen

VRC-Klasse

Die VRC-Klasse erbt von dem Panda3d-Framework. So bekommt sie unter Anderem die Puffer für den 3D-Raum und den Taskmanager von der Spiel-Engine vererbt, die die Grundlage für das Konzept bilden. Mithilfe des Taskmanagers kann man wiederkehrende Tasks wie das Abfragen des SoundAnalyzers und des Tastaturzustands ausführen. Dadurch vermeidet man das Programmieren in einer Hauptschleife. In ihrem Konstruktor initialisiert die VRC-Klasse, die für sich benötigten Objekte der anderen Klassen. In Variablen und einer Liste für Visuals werden die Referenzen auf die Objekte gespeichert.

Neben den für das Konzept wichtigen Objekte, wird in der VRC-Klasse auch die Initialisierung der Spiel-Engine vorgenommen. Unter anderem werden Fenster erstellt, und die Kameras den Fenstern zugewiesen.

Außerdem wird in der VRC-Klasse der Bedienmechanismus implementiert. Die Spiel-Engine implementiert anhand von Event-Polling das Belegen von Tasten mit Funktionen. Der Bedienmechanismus schaltet zwischen Visualmodus und Kameramodus zwischen Funktionen um, die für die Tastatureingabe in dem assoziativen Datenfeld Variablen für die Operationen richtig schalten. Dafür wird mithilfe des Taskmanagers der Spiel-Engine der Tastaturzustand kontinuierlich abgefragt. Anhand der Abfrage werden die relevanten Aktionen ausgeführt. Da die VRC-Klasse von der Spiel-Engine erbt kann man ganz Einfach im Konstruktor die Belegung der nötigen Tasten initialisieren und an Funktionen knüpfen.

```
self.accept('a', self.setOperation, ['a'])
```

```
self.accept('a-up', self.setOperation, ['a-up'])
...
self.taskMgr.add(self.executeOperation, 'keyboardaction')
self.taskMgr.add(self.spreadTheBeat, 'sound')
```

In der executeOperation-Funktion wird anhand des Bedienmodus die entschieden ob die Kamera-Operationen oder Visual-Operationen ausgeführt werden. Die spreadTheBeat-Funktion führt auf allen Visuals eine Methode zum Reagieren auf. Dadurch erhalten die Visuals gleichzeitig einen Art Tick, anhand derer Sie selbst kontinuierlich Funktionen implementieren können.

4.3.1 SoundAnalyzer-Klasse

In der Soundanalyzer-Klasse werden Bibliotheken für die Schnelle Fouriertransformation und für den Zugriff auf das PCM-Signal verwendet, um über die drei Frequenzbänder Bässe, Mitteltöne und Höhen das Spektrum zu analysieren und als Gleitkommazahl zu repräsentieren.

Der in dem Prototypen verwendete Soundanalyzer basiert auf dem Algorithmus aus dem LiveGL-Visualframework[4] von dem Github-Benutzer *phreax*, da sich der Algorithmus für die Soundanalyse aus der Spectral-Analyzerklasse[5] für die Zwecke des VRC eignet. Dazu wurde der Soundspektrumanalyzer aus dem Programm als Vorlage verwendet und in Python mithilfe von Pythonbibliotheken für lineare Algebra umgesetzt. Die Bibliothek NumPy[6] ist ein fundamentales Paket für wissenschaftliche Berechnungen in linearer Algebra, mit der Vektoren für die Soundanalyse als Puffer für die Eingabewerte des Mikrofons und Ausgabewerte der Schnellen Fourier-Transformation angelegt worden sind. Für die Schnelle Fourier-Tranformation wird die Pythonanbindung an die FFTW3-Subroutine von fftw.org verwendet.

4.3.2 Visual-Klasse

Die Visual-Klasse stellt Methoden für das Bewegen und Rotieren der Visuals bereit. Sie wird mit einer Referenenz auf den SoundAnalyzer und auf den 3D-Raum initialisiert. Es wird ein Platzhalter-Knoten für den Szenengraph angelegt, der Ausgangspfad für die anderen Elemente des Visuals ist. Dieser Knoten wird in den Bewegungsmethoden verwendet, um das Visual durch den Raum bewegen zu können. Alle Elemente, die an diesen Knoten angefügt werden, bewegen sich dann relativ zum Platzhalter-Knoten mit.

[4] Framework für OpenGL-Shader-Visuals `https://github.com/phreax/livegl/` (Zugriff 2013-07-03)
[5] `https://github.com/phreax/livegl/blob/master/src/spectral.cpp` (Zugriff 2013-07-03)
[6] `http://www.numpy.org/` (Zugriff 2013-07-03)

```
def __init__(self, render, loader, snd):
    self.snd = snd
    self.loader = loader
    self.render = render
    self.path = self.render.attachNewNode('dummy')
    self.path.setPos(0,0,0)
    ...
    self.setup()

def moveRight(self):
    self.path.setX(self.path, +self.visualMovementSpeed)
```

Außerdem werden noch initiale Zustandsvariablen für die Bewegungsgeschwindigkeit, Transparenz und Skalierung definiert. Der Konstruktor ruft zum Abschluss noch eine weitere leere Setup-Funktion auf. Diese Funktion ist dafür gedacht, als Framework-Funktion in den, von der Visual-Klasse abgeleiteten Visuals, überschrieben zu werden. Dadurch kann man an diesem Punkt die visualspezifischen Elemente laden und einstellen.

Weiter wird eine Methode für das Entgegennehmen der Soundanalyse implementiert. Diese Methode ruft noch eine weitere Platzhalter-Methode auf, die im eigentlichen Visual dann mit einem Algorithmus für das Reagieren auf den Sound überschrieben werden kann.

Für die Effekte werden leere Methoden angelegt, die aus der VRC-Klasse durch die Bedienung ausgelöst werden. In den einzelnen Visuals können diese Methoden überschrieben werden. Dadurch hat man ein Framework, um sich für die Tasten eigene Effekte anlegen zu können.

4.3.3 VisualFactory-Klasse

Der Prototyp verwendet für das Initialisieren der Visuals eine Visual-Objektfabrik, da die Initialisierung eines Visuals mehr als das Initialisieren des Konstruktor erfordern kann. Dadurch hat man als Programmierer mehr Freiheiten aus dem Visual-Framework auszubrechen und eigene Komponenten programmieren, um diese dann an die Visuals anzubinden. Die eigenen Komponenten können wiederum andere Bibliotheken und Techniken verwenden. Der Konstruktor nimmt Referenzen auf 3D-Raum und SoundAnalyse-Objekt entgegen, um damit die Visuals zu initialisieren. Die Visual-Objekte werden unter einem Namen in einem Wörterbuch abgelegt.

4.3.4 GUI

In der GUI befinden sich Slider, Buttons und Listen mit denen die Einstellungen an Kamera, Visuals und SoundAnalyzer vorgenommen werden können. So wird über die GUI mithilfe von Schiebern die Transparenz, die Bewegungsgeschwindigkeit und Skalierung für die Visuals eingestellt. Mit einem Button können die Visuals in den Raum gesetzt und entfernt werden. Standardmäßig werden die Visuals am Nullpunkt gesetzt. Die Namen der geladenen Visuals erscheinen in einer Auswahlliste. Dort kann man ein Visual auswählen, um es im Visualmodus zu bedienen. Für die Kamera kann die Sensitivität über einen Schieber eingestellt werden. Zur Soundanalyse können anhand von drei Schiebern ein Grenzwert eingestellt werden, mit dem dann in den Visuals der aktuelle Messwert verglichen werden kann. Dadurch kann man z.B. bei basslastigem Klang den Grenzwert erhöhen, um die Reaktion der Visuals auf starke Bässe zu trimmen.

Für den Überblick sind Labels vorgesehen, in denen die Positions- und Rotationsinformationen der Kamera und der Visuals anzeigent werden. Außerdem werden Informationen aus dem Operationswörterbuch angezeigt, wie beispielsweise ob die Kamera im Vorschaufenster synchron mit der Kamera im Präsentationsfenster ist, oder ob die Kamera auf ein Visual fixiert oder frei ist.

5 Usability Test

Die Gebrauchstauglichkeit und Benutzbarkeit wird qualitativ anhand von vier Testpersonen ermittelt. Der Test besteht aus zwei Teilen: eine Einarbeitungsphase und ein fiktives Szenario. Während der Tests werden Notizen gesammelt, um die Nutzererfahrung festzuhalten. Danach werden die Ergebnisse evaluiert. Die Tests sollen Fehler und Schwächen im ersten Prototypen aufzeigen.

Zunächst wird den Testpersonen das VJing-Konzept skizziert. Danach werden die Testpersonen in die Bedienung eingearbeitet. Die Testpersonen sollen spielerisch die Bedienung ausprobieren und Fragen stellen, wenn sie Hilfe benötigen. Dabei werden die Testpersonen mit den verschiedenen Spezialfunktionen für Effekte, Visualmanipulation und Soundanalyseeinstellung vertraut gemacht.

Nachdem sich die Testperson mit Übergangsmethoden und Effekten vertraut gemacht hat und gelernt hat, sich im 3D-Raum zu orientieren, Visuals zu Positionieren und die Kamerabewegungen beherrscht, wird ein Szenario gestellt. Die Testperson soll sich vorstellen auf einem Konzert zu drei Liedern Visuals zu performen. Bei elektronischer Tanzmusik sind Mitteltöne, Bässe und Höhen meistens gut voneinander getrennt. Oft besteht diese Musik aus einer Beatspur, mit Beatanordnungen im Bassbereich und in den Höhen, sowie anderen Spuren für Melodien, Vocals und andere stilistische Elemente. (http://howtomakeelectronicmusic.com/how-to-create-electro-lead-bass) Oft werden bei DJ-Sets mit Tief- und Hochpassfiltern gearbeitet, um sogenannten Druck aufzubauen. Dabei wird die Musik mit Tiefpassfiltern gedämpft und wieder aufgedreht. In solchen Momenten kann man zum Beispiel eine Aktion mit einem Visual oder mit der Kamera durchführen.

Während des Test werden die Äußerungen der Testpersonen in Stichwörtern notiert. Da die Testpersonen unterschiedlichen Hintergrund haben, bringen sie unterschiedliche Fähigkeiten mit. Es testeten ein Fussballer ohne viel Computerkenntnisse, eine Linguistikstudentin, die Programmiergrundlagen beherrscht, ein Filmemacher, der Dozent für neue Medien und Film an der Universität Siegen ist, und ein Student mit einer Leidenschaft für Computerspiele.

Die Aussagen der Testperson werden im Folgenden anhand der Notizen zusammengefasst und einer Evaluation unterzogen. Die Ergebnisse sollen Kriterien für den zweiten Prototypen werden.

5.1 Testperson 1: Normaluser

Die erste Testperson hat in ihrem Beruf gar nichts mit Computern zu tun. Privat verwendet die Person den Computer nur um Sportergebnisse nachzuschauen und Musik im Internet zu hören. Die Testperson war trotzdem offen für ein Konzept, das keine Gemeinsamkeiten mit den Erfahrungen der Testperson hat.

Spielerisch probierte die Person die Tastatureingaben aus. Die Bedienung konnte schnell erlernt werden. Jedoch hatte die Person Schwierigkeiten eine Performance zu gestalten und Übergänge oder Effekte in den richtigen Momenten vorzunehmen.

Zur Tastenbelegung der Bedienung machte die Person keine Angaben. Bei der Benutzung der grafischen Oberfläche fand die Person es aber irritierend, dass das zuletzt angeklickte Visual nicht automatisch das aktive Visuals ist.

Für Szenenkomposition wünschte sich die Person eine Speicherfunktion, so dass man Szeneneinstellungen abspeichern kann. Dabei soll die Position, Rotation, Kameraposition und Blickwinkel gespeichert werden können.

Die Testperson hatte zudem Schwierigkeiten mit der Orientierung im 3D-Raum, da es keine Orientierungspunkte gibt und schlug Reset-Funktion vor, mit der man die Kamera wieder auf Aufgangslage bringen kann.

5.2 Testperson 2: Linguistikstudentin

Die zweite Testperson verfügte über grundlegende Programmierkenntnisse und war außerdem bereit auch Visuals zu erstellen.

Bei der Benutzung fand die Tastaturbedienung für Neigung, Rotation und Drehung unlogisch und fand die Bedienung nicht intuitiv. Trotzdem sagte die Testperson, die Bedienung sei schnell erlernbar. Die Benutzung allgemein wurde als unübersichtlich kritisiert, da die Person nicht schnell auffassen konnte, in welchem Modus sie sich befand.

Auch diese Person hatte Probleme sich in dem Raum zu orientieren. Für das Erlernen der Funktionen wie Soundanalyse hielt die Person kleine Erklärungskästen für hilfreich. Als Orientierungshilfe schlug sie eine Gitterstruktur auf der XY-Ebene vor.

Zur Performance kritisierte die Person einen zu komplizierten Ablauf. Da man ein Fenster hat, in dem die Spiel-Engine läuft und ein Fenster für die grafische Oberfläche, muss man zwischen den Fenstern wechseln, damit die Tastatureingabe wieder in der Spiel-Engine funktioniert, wenn man davor etwas in der grafischen Oberfläche verstellt hat.

Auch diese Person fand, dass eine Abspeicherfunktion von Positionierung und Performance hilfreich wäre.

Beim Erstellen von Visuals fand die Person das Framework ersichtlich und sinnvoll gestaltet. Zusammen wurde innerhalb einer Stunde ein Visual erstellt. Das Visual beinhaltet eine Funktion, welche die 3D-Elemente zum Beat rotiert.

5.3 Testperson 3: Filmemacher mit VJ-Erfahrung

Die dritte Testperson hat Erfahrung mit Videobearbeitungsprogrammen, unter anderem AfterEffects von Adobe, mit dem man unter anderem 3D-Animationen erstellen kann. Dabei arbeitet man auch mit einer virtuellen Kamera in einem 3D-Raum. Dadurch hatte die Testperson keine Schwierigkeiten mit dem Betrachtungskonzept und arbeitete vor allem damit. Außerdem hat die Testperson schon einmal als VJ auf einer Party eine Performance mit Resolume (footnote here) und Videoclips realisiert.

Die Testperson fand das Bedienkonzept ganz gut. Mit der Orientierung hatte sie keine Probleme und benutzte die Kamerafunktionen ausgiebig. Schließlich fand sie sich auch mit Spezialfunktionen für die Kamera zurecht und konnte Schnitte einsetzen.

Bei der Performance passten die von mir präparierten Visuals aber nicht. Die Testperson wünschte sich andere Visuals, welche mehr Raum einnehmen, wie beispielsweise Landschaften. Auch fand diese Testperson viele Labels in der GUI unnötig und die Umschaltung der Bedienmodi ein bisschen aufwändig, meinte aber, dass man sich daran gewöhnen kann. Die Idee, alle Funktionen, die momentan nur in der grafischen Oberfläche verfügbar sind, durch Tastaturkürzel zu regeln, wurde angesprochen, um Bedienwege zu verkürzen. Eine reduzierte grafische Oberfläche wurde vorgeschlagen. Vor allem die verschiedenen Reiter zum Umschalten wurden kritisiert. Die Zusammenführung aller wichtigen Regler auf einer gemeinsamen Fläche sei sinnvoll. So würden die Bedienwege verkürzt. Die wichtigsten Labels der Koordinaten könne man auch platzsparend in einer Zeile plazieren.

Zwar konnte der Person das Erstellen von einem Visual erklärt werden, aber aufgrund keiner Kenntnisse im Programmieren wurde nur darüber diskutiert. Jedoch aus eigener VJ-Erfahrung gefiel der Person das Konzept des getesten Prototyps.

5.4 Testperson 4: Computerspieler

Die vierte Testperson hatte aufgrund ihrer Erfahrungen mit verschiedenen Computerspielen viele Verbesserungsvorschläge bezüglich der Bedienung. Die Bedienung wurde als

schlecht bewertet. Das wechseln der Bedienmodi wurde kritisiert. Anstatt die ESC-Taste zum Verlassen des aktuellen Modus zu drücken, wurde angeregt, man könne auch direkt in den gewünschten Modus wechseln.

Trotz allem unterbreitete die Person viele Verbesserungsvorschläge. Die Testperson bemerkte die Tasten für die Kameraspezialfunktionen und argumentierte, dass man genauso gut mit den Tasten auf Visual-Einstellungen wie Transparenz, Skalierung und Geschwindigkeit für das Inkrementieren und Dekrementieren der Werte benutzen könnte. So würde man eine Taste drücken und anschließend mit zum Beispiel den Pfeiltasten oder dem Mausrad den Wert verändern. Dadurch würde die grafische Oberfläche nur noch zur Orientierung und Statusanzeige existieren.

Die Testperson schlug außerdem vor, mit der Maus per Klick in das Vorschaufenster mit der Mausbewegung die Rotation der Szenerie zu ermöglichen, während man die Maustaste gedrückt hält. An der Kamera wurde eine zu hohe Rotationssensitivität kritisiert.

5.5 Evaluation

Der erste Prototyp genügte den Anforderungen an Benutzbarkeit offensichtlich noch nicht ganz. Man kann zwar schon alle Funktionen bedienen, jedoch stiftet die Trennung zwischen Spiel-Engine und grafischer Oberfläche Verwrirrung. Bei dem Testversuch waren sich alle Personen einig, dass die Bedienung hinsichtlich der Zugriffschnelligkeit verbessert werden muss. Das Wechseln der Bedienmodi und das häufige Umschalten zwischen den Fenstern verlängert Bedienwege. Oft hatten die Testpersonen Schwierigkeiten, wenn sie etwas im Fenster für die grafische Oberfläche eingestellt hatten und direkt wieder in der Spiel-Engine die nächste Aktion ausühren wollten. Das Wechseln der Fenster beeinträchtigte das auf den richtigen Moment abgestimmte Performen. Häufig passierte im Spiel-Engine-Fenster dann auch nichts, da die Testperson sich noch in keinem Bedienmodus befand.

Sobald eine Szenerie angeordnet worden war und die Testpersonen damit ihre Idee performen konnten, wünschten sie sich eine Abspeicherfunktion. Die Abspeicherfunktion wurde mit dem vorhergehenden Aufwand begründet, eine passende Szenerie nicht jedes mal neu erstellen zu müssen.

Für die Verbesserung des Prototyps konnte die vierte Testperson gute Ideen beitragen. Zwar wurden von allen Testpersonen Anregungen geäußert, die Maus für Bedienung mit zu verwenden. Die vierte Testperson hatte konkrete Vorschläge für Bedienmöglichkeiten mit der Maus. Indem man eine Maustaste gedrückt hält, kann die Spiel-Engine so programmiert werden, dass die Mausbewegung für die Kamerarotation verwendet werden kann. Das Mausrad kann verwendet werden, um die Werte für Transparenz, Skalierung

oder Geschwindigkeit zu verändern. Dadurch können Bedienwege verkürzt werden, da dafür nicht mehr jedes mal das Fenster gewechselt werden muss.

6 Verbesserter Prototyp

Der zweite Prototyp basiert auf dem ersten Prototypen. Es werden Verbesserungen an der Bedienung vorgenommen. Dafür wird der Prototyp um Funktionen erweitert. Die Ergebnisse aus der ersten Testreihe werden zur Überarbeitung der Bedienung verwendet. Außerdem werden fehlende Anforderungen, wie das Speichern und Laden einer Szenerie, implementiert.

Die Softwarearchitektur wird dabei beibehalten, da sie genügend Flexibilität für Anpassungen bietet. Darüber hinaus wird die Codebasis um überflüssigen Code bereinigt und stellenweise vereinfacht.

6.1 Neue Funktionen

Dem Prototypen wird das Laden und Speichern von Szenerien hinzugefügt. Bei der Steuerung kommen neue Mausfunktionen hinzu. Die Maus kann verwendet werden um die Transparezn, Skalierung oder Geschwindigkeit der Visuals einzustellen oder die Kamera zu rotieren. Eine Anzeige im Fenster der Spiel-Engine zeigt Kameraposition und Rotation, sowie Visualposition und Rotation an.

6.1.1 Speichern und Laden von Szenerien

Da das Speichern und Laden einer Szenerie während der Tests eine von allen Testpersonen geäußerte Kritik war, wird bei der Verbesserung des Protoyps eine Funktion für das Speichern der erstellten Szenerie implemetiert. Die relevanten Daten für das rekonstruieren der Szenerie werden in einer Datei als String gespeichert. Um diesen String wieder einlesen zu können wird als Format JSON (footnote here) verwendet. Dadurch können Daten wie Position, Rotation, Transparenz, Skalierung und Geschwindigkeit der Visuals in einer Liste für jedes Visual gespeichert werden. Mit JSON kann man Datenstrukturen in einen String umwandeln, um sie wieder einlesen und komfortabel auslesen zu können. So wird für jedes Visual ein assoziatives Array mit den relevanten Werten angelegt und in einer Liste gespeichert. Der String wird in einer Datei gespeichert und dank des JSON-Formats wieder in Python-Datenstrukturen umgewandelt . Um die Sze-

nerie wieder herzustellen, wird über die Liste iteriert und die in der Liste enthaltenen Visuals aktiviert und positioniert.

6.1.2 Maus

Die Verbesserungsvorschläge der vierten Testperson fließen direkt bei der Überarbeitung der Maus mit ein. Um die Mauseingabe zu kontrollieren soll mit der Maus nur die Bewegung abgefragt werden, wenn eine Maustaste gedrückt ist. Dadurch wird die Benutzung der Maus bewusst ausgeführt, da erst eine Taste gedrückt werden muss. Erst dann wird die Maus an eine Funktion gebunden. So soll eine Taste gedrückt sein, um die Kamera mit der Maus zu schwenken. Das Mausrad kann verwendet werden, um die Werte für die Transparenz, Skalierung oder Geschwindigkeit zu manipulieren.

Für diese Funktionserweiterungen wird die Bedienlogik um Verzweigungen für die Verarbeitung der Mauseingabe erweitert. Dabei muss für jeden Bedienmodus neue Fallunterscheidungen getroffen werden, welche in der Visualklasse oder Kamera die entsprechenden Variablen setzen.

6.2 Usability

Bei dem ersten Prototyp fielen während der Tests viele Unzulänglichkeiten der Software in der Benutzung auf. Die Unzulänglichkeiten behinderten die Benutzer schnell und zielstrebig die gewünschten Aktionen vorzunehmen. Der zweite Prototyp greift zu Verbesserung der Usability die Testergebnisse auf und setzt diese in der Bedienung und der grafischen Oberfläche um.

6.2.1 Grafische Oberfläche

Die Testergebnisse zeigten Unzulänglichkeiten in der grafischen Oberfläche des ersten Prototypen auf. Vor allem müssen Bedienwege innerhalb der grafischen Oberfläche verkürzt werden. Dazu werden die Bedienelemente aus den Reitern für aktive Visuals, Kamera und Soundeinstellungen unter einem Reiter zusammengefasst. Bedienelemente für das Laden und Speichern von Szenerien werden im Reiter für das Laden und Entfernen von Visuals untergebracht. Letztendlich hat man nur noch zwei Reiter, was zudem der Übersichtlichkeit dient.

In den Tests verwirrten Beschriftungen zur Anzeige des Zustands aus der OperationMap die Benutzer. Deshalb werden diese Elemente aus der grafischen Oberfläche entfernt.

Dadurch wird das Fenster für die Oberfläche schmaler und bleibt neben dem Fenster für die Spiel-Engine besser sichtbar.

6.2.2 Bedienung

Aus den Testergebnissen wurden Vorschläge für die Verbesserung der Bedienung aufgegriffen. Auch hier spielt der Verkürzung der Bedienwege eine große Rolle.

Kameraschwenks mit der Maus sind nun möglich. Werte für die Geschwindigkeit von Kamera und Visuals sind mit dem Mausrad veränderbar. Transparenz und Skalierung der Visuals können auch mit dem Mausrad verändert werden. Das Spiel-Engine-Fenster verfügt nun auch über eine Anzeige zur Information über Kamera- und Visualstatus und Bedienmodus. Dadurch hat man eine schnelle Übersicht über das ausgewählte Visual und seine Einstellung. So muss der Blick nicht so häufig auf die grafische Oberfläche wandern.

Die Veränderung der Werte für Geschwindigkeit, Transparenz und Skalierung eines Visuals kann im Fenster der Spiel-Engine mittels Tastatureingabe gewählt werden und anschließend mit dem Mausrad eingestellt werden. Mittels einer Taste wird die Manipulation des gewünschten Wertes an das Mausrad gekoppelt. Für die Transparenz, Skalierung und Geschwindigkeit sind die Tasten t", fünd "gïn der selben Reihenfolge vorgesehen. Wenn eine von den Tasten gedrückt wird, erscheint in der Anzeige im Spiel-Engine-Fenster eine Information, ob die Transparenz-, Skalierungs- oder Geschwindigkeitsvariable ausgewählt ist. Diese Tasten reagieren nur im Visualmodus für diese Funktion.

Da die Kamera keine Variable für Transparenz und Skalierung hat, entfällt das Binden von Variablen auf das Mausrad mittels Tastatureingabe. Man muss nur die Geschwindigkeit manipulieren. Deswegen verändert das Mausrad direkt den Geschwindigkeitswert der Kamera. Außerdem sind die Tasten t", fünd "g"bereits belegt mit Funktionen aus dem ersten Prototypen für das synchronisieren der zwei virtuellen Kameras.

Der Bedienweg, um das zu bedienende Visual zu wechseln, war ein wesentlicher Mangel im ersten Prototypen. Deshalb wird eine Taste zum Durchwechseln der geladenen Visuals eingeführt. Mit der tabTaste kann der Benutzer im Visualmodus das nächste Visual aus der Liste der geladenen Visuals auswählen.

7 Zweiter Test

Das Virtual Room Konzept richtet sich vor allem an VJs mit Programmierkenntnissen, da die Erstellung der Visuals das Programmieren vorraussetzt. Deshalb sind bei der zweiten Testreihe Informatiker und Künstler als Testpersonen interessant, um die Usability des zweiten Prototyps für das VJing zu ermitteln. Für die Tests werden die selben Beispielvisuals verwendet, welche für den ersten Testlauf programmiert worden sind.

Mit dieser Testreihe soll ermittet werden, ob der neue Prototyp den Überblick und die Orientierung für das Performen in Echtzeit mit den Verbesserungen erleichtert. Mit den verkürzten Bedienwegen wird versucht, die Zugriffszeiten auf Funktion zu verringern.

Ähnlich zu den Tests aus der ersten Testreihe wird versucht, in einer konstruierten Umgebung zu Musik passend Visuals zu performen. Jedoch werden die Testpersonen wegen des zugenommenen Bedienumfangs zu Beginn eingehender mit der Bedienung vertraut gemacht.

In der Testreihe soll die Verständlichkeit der Software ermittelt werden. Während der Tests wird untersucht, wie schnell die Benutzer sich mit der Bedienung zurecht finden. Mit der reduzierten grafischen Oberfläche und der Orientierungsfläche soll eine verbesserte Orientierung erzielt werden. Deshalb liegt während der Tests das Augenmerk auch darauf, ob die Testpersonen weniger Schwierigkeiten haben mit der Orientierung im Programm als bei den ersten Benutzertests.

Nach ca. einer Stunde Einarbeitung sollen die Testpersonen versuchen mit dem Programm gezielt zu Musik zu Performen. Danach werden die Testpersonen nach ihren Eindrücken befragt um die Verständlichkeit des Konzepts bewerten zu können.

7.1 Testperson 5: Künstlerin

Die fünfte Testperson ist eine Künstlerin, die vor allem im Videobereich tätig ist. Sie gibt Kurse an einer Volkshochschule in Berlin zur Bild- und Videobearbeitung und arbeitet schon seit Jahren mit Spezialprogrammen in diesem Bereich. Das Programm war für die Testperson mit keinem Programm, das sie benutzt, vergleichbar. Da die Bedienung mit der Tastatur der eines Computerspiels ähnlich ist, war die Testperson mit einem

unvertrauten Konzept konfrontiert. Vorteilhaft zur Verdeutlichung des Konzepts war die Verwendung eines Projektors in einem Atelier.

Nach der einstündigen Einarbeitungsphase hatte die Person immernoch Schwierigkeiten mit der Orientierung in dem virtuellen dreidimensionalen Raum. Der Wechsel zwischen den Bedienmodi für Kamera und Visuals gelang nicht flüssig. Dadurch bewegte die Testperson unfreiwillig die Kamera, anstatt, wie gewollt, das Visual zu bewegen.

Die Bedienung war für die Testperson nicht intuitiv. Auch nach der Einarbeitung benutzte die Testperson viele Funktionen des Prototyps nur unter Nachfrage. Das Mausrad wurde wenig benutzt, eher wurden die Werte für Transparen, Skalierung und Geschwindigkeit über die grafischen Oberfläche manipuliert. Trotzdem befand die Testperson die Bedienung erlernbar.

Als die Testperson sich bereit fühlte zu Musik zu performen, wurde das Programm neu gestartet. Die Person war nach der Einarbeitung in der Lage Visuals zu laden und im Raum zu bewegen. Die Testperson spielte mit der Skalierung und Transparenz. Durch die Verwirrung bei der Orientierung gelang eine Bewegungszustandsspeicherung eines Visuals mittels Modusswitch sehr schwierig. Mit dem Ergebnis ihrer Performance während des Tests war die Testperson nicht zufrieden. Ihr gelangen Übergänge nicht, durch den Orientierungsverlust passierten ungewollte Aktionen.

Trotzdem zeigte sich die Testperson beeindruckt über das Potenzial des Konzepts und hält es auch nach dem Test für erlernbar, obwohl die Bedienung häufig misslang.

7.2 Testperson 6: BWL-Student

Bei der sechsten Testperson handelt es sich um einen technikinteressierten Studenten der Betriebswirtschaftslehre, der schon mehrere Computerspiele gespielt hat.

Die Testperson befand die Bedienung als nicht intuitiv, erlernte sie aber schnell vollständig. Mit der Orientierung im 3D-Raum hatte die Testperson keine Schwierigkeiten, Visuals zu bewegen, auch wenn sie verdreht sind. So gewöhnt die Testperson sich schnell daran, dass je nach Visualrotation bei einem Tastendruck auf äßum Bewegen nach links, das Visual unter Umständen auf dem Bildschirm nach rechts bewegt.

Trotz der anfangs nicht intuitiven Bedienung kann die Testperson das Programm schnell erlernen. Auch die neuen Funktionen wie die Tasten für das Manipulieren der Werte für Transparenz, Skalierung und Geschwindigkeit findet die Testperson heraus. Der Modusswitch in den Escaped-Mode zum Speichern einer Tastatureingabe wird verstanden und schnell erlernt.

Nach der Einarbeitungsphase ist die Testperson in der Lage zu Musik mit allen Bedienmöglichkeiten des Programms zu performen. Zunächst baut sich die Testperson nach dem Neustart des Programms eine Szene zusammen und positioniert die Kamera. Anschließend wird die Szene gespeichert und als Ausgangslage verwendet. Mit Hilfe des Escaped-Mode speichert die Testpersonen Bewegungen ab. In der Performance wusste die Testperson Visuals in das Blickfeld zu fliegen und mittels Mausrad im Fenster für die Spiel-Engine das Visual sanft zu stoppen. Durch das Mausrad kann eine negativer Wert eingestellt werden. Durch diese Eigenschaft konnte die Testperson die Bewegungen invertieren.

Im Anschluss äußert sich die Testperson begeistert über das Bedienkonzept und empfindet Spaß bei der Benutzung.

7.3 Testperson 7: Bioinformatikstudent

Die siebte Testperson ist Bioinformatikstudent an der Freien Universität Berlin. In seiner Freizeit interessiert er sich für Computerspiele und spielt verschiednste Computerspielgenres wie Strategie, Rollenspiele oder Egoshooter. Durch den Informatikhintergrund sind Programmierkenntnisse in Python vorhanden.

Bei der Orientierung hatte die Testperson wenig Probleme. Die Relativbewegung, die sich aus der Rotation der Visuals und Kamera im Raum ergibt, kann die Testperson in den meisten Fällen richtig einschätzen und zügig die Positionierung der Visuals meistern. Die Testperson beachtet auch bewusst die Informationen zu Koordinaten und Rotation der Kamera und Visuals aus der neuen Anzeige im Spiel-Engine-Fenster.

Mit der Bedienung fand sich die Testperson auch schnell zurecht. Nach kurzer Zeit war die Testperson mit allen Funktionen vertraut und hatte sich an die Bedienung gewöhnt. Trotzdem findet die Testperson die Bedienung nicht intuitiv.

Als Performance setzte die Testperson verschiedene Einstellungen ein, indem gespeicherte Szenerien geladen wurden. An musikalischen Höhepunkten setzte die Testperson auch Effekte ein.

Im anschließenden Gespräch wurde auch auf die Erstellung der Visuals eingegangen. Technische Aspekte konnten diskutiert werden. So findet die Testperson das Frameworkkonzept für die Erstellung der Visuals interessant und ist dank der Spiel-Engine-Syntax in der Lage den Code zu verstehen.

7.4 Testperson 8: Wirtschaftsinformatikerin

Der letzte Test aus der zweiten Testreihe wird mit einer Wirtschaftsinformatikstudentin durchgeführt. In ihrer Freizeit beschäftigt sich die Testperson viel mit Malerei. Unter anderem benutzt sie auch Vektorgrafikprogramme für ihre Kunst. Daneben war sie leidenschaftliche Taktik-Shooter-Spielerin.

In der Einarbeitungsphase hatte sie keine Probleme, sich im 3D-Raum zu orientieren. Mit der Bedienung fand sie sich schnell zurecht, obwohl sie auch insgesamt die Bedienung nicht intuitiv findet. Nach einer Weile verinnerlichte die Testperson den Moduswechsel. Im Kameramodus bediente sie sich oft der Maussteuerung, um die Kamera zu schwenken. Die Rotationstasten wurden ausschließlich für die Visuals verwendet.

Bei dem Performance-Teil des Tests arbeitete die Testperson viel mit der Kamera. Die Funktionen für das Lösen der Kamerasynchronisation und Resynchronisierung waren ein Bestandteil der Performance. Mit der Maussteuerung wurden Kamerafahrten entlang der Oberfläche groß skalierter Visuals durchgeführt. Die Testperson performte aufgrund der extensiven Kameranutzung die Visuals einzeln, anstatt sich eine Szenerie zurecht zu legen. Das Mausrad wurde am häufigsten dazu benutzt, die Geschwindigkeit der Kamera- oder der Visualbewegung zu verändern.

7.5 Evaluation

Mit der Evaluation sollen die Testergebnisse hinsichtlich der Bedienbarkeit betrachtet werden. Die Testergebnisse geben Aufschluss daruüber, ob die Orientierung innerhalb des Programms besser möglich ist. Das Bedienkonzept hat sich nur in Details geändert.

Bei den Tests des zweiten Prototyps trat die grafische Oberfläche in den Hintergrund. Durch die verbesserte Bedienung und die Anzeige im Fenster der Spiel-Engine rückte der Fokus in das Spiel-Engine-Fenster. Die durch die TabTaste neu eingeführte Funktion die Visuals zu wechseln trug wesentlich dazu bei die Bedienwege zu verkürzen. So mussten die Fenster nicht mehr umgeschaltet werden, um die Visuals zu wechseln, was die Bedienwege verkürzt. Zur reduzierten grafischen Oberfläche wurden von den Testpersonen im zweiten Testdurchlauf weniger bis gar keine Fragen gestellt. Nur zu den Soundanalyse-Einstellungen wurden Fragen gestellt, was die Schieber zum Enstellen der Vergleichswerte bewirken.

Während des Performens nutzte jede Testperson die Möglichkeit Bewegungen zu invertieren. Mit der Gitterebene als optische Konstante im 3D-Raum fiel es den Testpersonen viel leichter die Visuals und die Kamera zu kontrollieren. Die Kamera-Toggles für das Lösen und wiedersynchronisieren der zwei Kameras im Raum stifteten jedoch immern-

och Verwirrung. Mit der neuen Rotationsfunktion für die Kamera mit der Maus und der schnell verstellbaren Kamerageschwindigkeit konnte bei einem Irrtum schnell wieder zum Visual geflogen werden, was hilfreich ist, wenn der Benutzer seine Kamera nicht auf das Visual fixiert hat.

Mit dem Speichern und Laden von Szenerien wurde eine Anforderung aus dem ersten Test implementiert. Nach dem zweiten Testlauf hatten die Testpersonen im Anschluss eher Anregungen anzumerken, die sich nicht mehr mit denen der anderen Testpersonen deckt. Im Vordergrund standen eher Fragen, welche Möglichkeiten mit dem Konzept außer den Beispielvisuals möglich sind. Den Testpersonen wurde daraufhin das Framework-Konzept erkärt. Testpersonen mit Informatikhintergrund konnten den Code nachvollziehen. Testperson 7 und 8 fanden die Idee der Visualklassenvererbung gut, wollten aber kein Visual programmieren.

Insgesamt reduzierte die neue grafische Oberfläche zusammen mit dem überarbeiteten Fenster für den 3D-Raum den Bedienaufwand. Mit den Ergebnissen aus der ersten Testreihe konnte eine Bedienung gefunden werden, die von allen Personen als erlernbar eingestuft worden ist. Mit den hinzugefügten Mausfunktionen und der Speicher- und Lade-Funktion für Szenerien wurde die Benutzbarkeit gesteigert.

8 Resultat und Fazit

Im Rahmen dieser Arbeit konnte ein Werkzeug für das Performen von selbst programmierten Visuals in einem dreidimensionalen Raum umgesetzt werden. Die Visuals in dieser Arbeit sind bloß Testvisuals, die das Potenzial des Konzepts bereits jetzt verdeutlichen. Der ganze Funktionsumfang für Grafikprogrammierung der Spiel-Engine steht zur Verfügung, um Visuals zu erstellen. Mithilfe der Anordnung von Lichtquellen kann die Szenerie beleuchtet werden, Schatteneffekte werden möglich. Durch die Unterstützung von Shadern können auch komplexe mathematische Algorithmen in Echtzeit für Visuals genutzt werden. Da sich die Arbeit auf die Entwicklung eines VJ-Frameworks konzentrierte, wurden die Möglichkeiten zur Erstellung von Visuals nicht ausgeschöpft.

Der zweite Iterationsschritt verbesserte den Prototypen in seiner Bedienbarkeit, was man bei den zweiten Benutzertests deutlich wahrnehmen konnte. Das Testen deckte viele Schwächen bei der Bedienbarkeit im ersten Prototypen auf, aber auch der zweite Prototyp hat noch viel Spielraum für Verbesserungen. Dank Softwarearchitektur konnten Nachbesserungen am ersten Prototyp vorgenommen werden ohne die Architektur für den zweiten Prototypen groß zu ändern. Stattdessen konnte stellenweise das Programmverhalten erweitert und gegebenenfalls geändert werden.

Das Konzept kann als Vorlage für die Entwicklung ähnlicher Projekte unter Verwendung einer anderen Spiel-Engine oder Programmiersprache dienen. Da die Arbeit als Framework mit Fokus auf Benutzbarkeit während der Performance entwickelt wurde, schöpfen die Beispielvisuals nicht alle Möglichkeiten der Spiel-Engine für die Erstellung von Visuals aus. Der Funktionsumfang von Spiel-Engines birgt ein großes Potenzial für die Realisierung von Animationen und Effekten, das in mit dieser Arbeit nicht untersucht worden ist. Die Architektur des Frameworks zeigt einen Weg auf, mit dem der VJ seine Visualideen mithilfe einer Visual-Klasse und einer Visual-Factory entwickeln kann. Nachdem eine neue Klasse für ein Visual angelegt und in der Factory geladen wird, kann der VJ anfangen in der setup-Funktion den Funktionsumfang der Spiel-Engine auszuprobieren.

Insgesamt lässt sich festhalten, dass das Konzept für das VJing in Echtzeit verwendet werden kann. Weitere Arbeiten zeigen welche künstlerischen Potenziale in der Verwendung einer Spiel-Engine in der Digitalen Kunst rund um das VJing liegen.

Literatur

[1] W. 375. *VJing*. August 2010. Edited during Mapping Festival, Geneva (mappingfestival.com), Assempled during Hyperactivity, Centre d'Art Contemporain de Neuchatel (www.can.ch). Greyscale Publishing, 2010.

[2] T. Alatalo. *A System for Using 3d Animations in Live Performances*. pdfio.com (http://www.pdfio.com/k-1391628.html, accessed 2013-07-03). original work published on studio.kyperjokki.fi (unaccessible 2013-07-03). 2006.

[3] A. Engström, M. Esbjörnsson und O. Juhlin. "Nighttime visual media production in club environments". In: *CHI 2008 Proceedings, Workshops : Presented at the Night and darkness: Interaction after dark*. 2008.

[4] M. Goslin und M. Mine. "The Panda3D graphics engine". In: *Computer* 37.10 (2004), S. 112–114. ISSN: 0018-9162. DOI: 10.1109/MC.2004.180.

[5] T. K. Group. *OpenGL Overview*. http://opengl.org/about (accessed 2013-07-03).

[6] M. Lewis und J. Jacobson. "Game Engines In Scientific Research". In: *Communications of the ACM* 45/1.0001-0782 (2002).

[7] G. van Rossum. *Python Workshop*. Slides, http://www.python.org/doc/essays/ppt/acmws/sld001.htm (accessed 2013-07-03). 1999.

Abbildungsverzeichnis

A Appendix

A.1 Verwendete Abkürzungen

GUI Graphical User Interface, engl. für grafische Benutzeroberfläche.

VJ Kurzform für Visual-Jockey.

VJing Des VJs Akt der Darstellung/Performance,

VRC Virtual Room Concept, der Name des in dieser Arbeit beschriebenen VJ-Konzepts, das in einem virtuellen 3D-Raum stattfindet.

A.2 Screenshots des Prototypen

In den Screenshots in Abb. A.1 und A.2 sieht man die ungenutzte Fläche in der GUI. Das was in dem Vorschaufenster sichtbar ist, wird ohne die Statusanzeigen direkt an ein anderes undekoriertes Fenster ausgegeben, das auf Beamer gelegt werden kann.

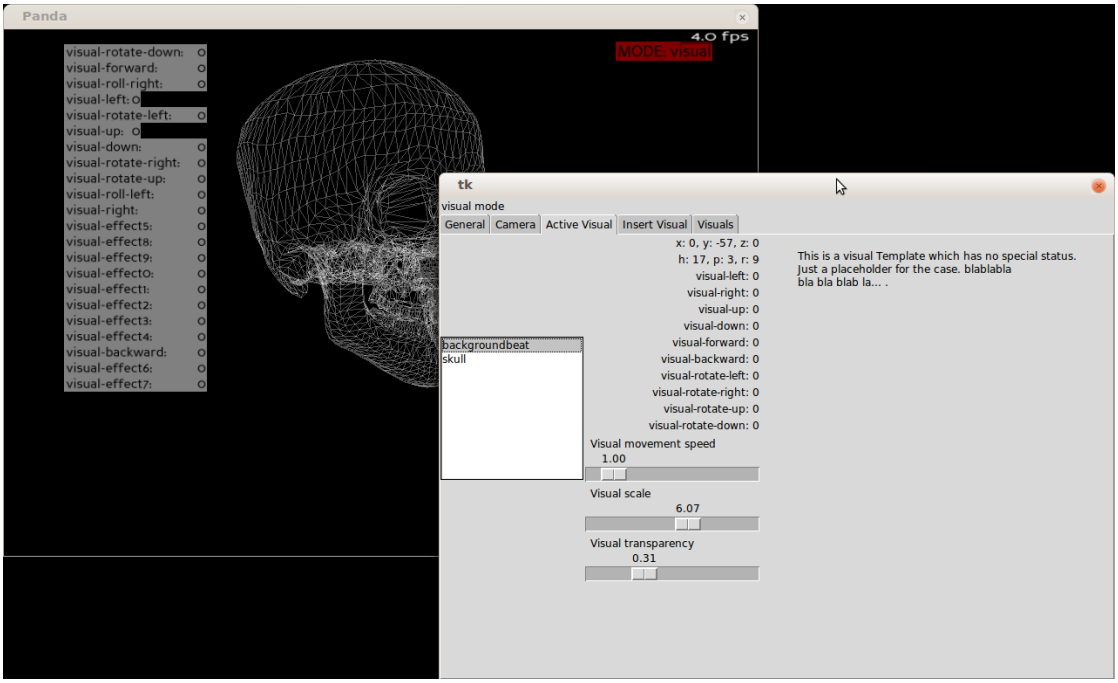

Abbildung A.1: Screenshot der Oberfläche der ersten Version des Prototyps. Zu sehen ist das Vorschaufenster und das zusätzliche Fenster für die GUI

Abbildung A.2: Screenshot der Oberfläche der zweiten Version des Prototyps mit reduzierter GUI und Orientierungsfläche

A.3 Implementierung eines Beispielvisuals

Über die Effekttasten kann eingestellt, werden, ob das Visual zu Musik skaliert oder nicht, Außerdem kann man zwischen den Rendermodi Gittermodell und ausgeüllt umschalten. Für die Skalierungsfunktion werden *LerpInterval*-Funktionen definiert, welche als Parameter die Zeit angegeben haben, um innerhalb davon von einem Wert zu einem anderen zu gelangen, wenn die Funktion aufgerufen wird.

Im Folgenden ist der Code des Totenkopf-Visuals aus Abb. A.1 und A.2 zu sehen. Die Funktionen *setup()*, *performBeat()* und *effect1()* bis *effect4()* sind dabei die aus der Visual-Klasse überschriebenen Funktionen, welche gedacht sind, um Platzhalter für die Visualelemente und -Algorithmen zu sein.

```
from panda3d.core import CardMaker, MovieTexture,
NodePath, TextureStage, TransparencyAttrib
from panda3d.core import DirectionalLight, Vec3, Vec4
from panda3d.core import PandaNode
from direct.interval.LerpInterval import *
from direct.interval.IntervalGlobal import Sequence,
LerpFunc, Parallel
import math, sys, colorsys
from visual import visual

class Skull(visual):

    def setup(self):
        self.skull = self.loader.loadModel('skull')
        self.skull.setP(90)
        self.skull.setX(3)
        self.skull.setZ(-2)
        self.skull.reparentTo(self.path)
        self.skull.setRenderModeWireframe()

        self.colorLerp0 = LerpColorInterval(
            self.skull,
            0.15,
            Vec4(0,0,0,1),
            Vec4(1,1,1,1)
        )
        self.colorLerp1 = LerpColorInterval(
            self.skull,
            0.85,
            Vec4(1,1,1,1),
            Vec4(0,0,0,1)
        )
        self.scaleLerp0 = LerpScaleInterval(
            self.skull,
            0.15,
```

```
            self.scale + self.scale/5,
            self.scale
        )
        self.scaleLerp1 = LerpScaleInterval(
            self.skull,
            0.85,
            self.scale,
            self.scale + self.scale/5
        )

        self.yFunc = self.nothing

    def performBeat(self):
        if self.sndX > self.snd.xThreshold:
            Sequence(self.colorLerp0, self.colorLerp1).start()
        if self.sndY > self.snd.yThreshold:
            self.yFunc()

    def scaleFunc(self):
        Sequence(self.scaleLerp0, self.scaleLerp1).start()

    def effect1(self):
        self.skull.setRenderModeFilled()

    def effect2(self):
        self.skull.setRenderModeWireframe()

    def effect3(self):
        self.yFunc = self.nothing

    def effect4(self):
        self.yFunc = self.scaleFunc

    def nothing(self):
        pass
```