HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

J3DEval

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2011

Autor(en): Mühlemann Lara, Walser Marion
Betreuer: Thomas Letsch

Inhalt

Java-3D-Library Evaluation/Refactoring

Studentinnen

- Lara Mühlemann
- Marion Walser

Einführung

Für eine bestehende 3D-Java-Applikation [1] soll die verwendete 3D-Bibliothek ersetzt werden.

Bei der oben aufgeführter Anwendung handelt es sich um eine Visualisierungs-Applikation, bei welcher der Zusammenhang zwischen Klassen und Objekten visuell veranschaulicht wird.

Dabei ist für den 3D-Teil eine Library eingesetzt (Java3D), welche direkt auf den Treibern der entsprechenden Video-Karte aufsetzt und somit sehr schnell, aber nicht mehr unabhängig von der Hardware ist und somit zu unterschiedlichem Verhalten je nach Hardware führt (das "Write-Once-Run-Everywhere"-Paradigma ist nicht mehr gegeben).
Zudem ist eine vorgängige Installation der Java3D-Library nötig.
Für den Einsatz im Unterricht wäre eine Unabhängigkeit von der darunterliegenden Hardware erstrebenswert.

Aufgabenstellung

Bei dieser Studienarbeit soll zuerst eine Evaluation durchgeführt werden, mit welcher bestimmt wird, welche hardwareunabhängige Technologie am besten für eine Ablösung verwendet werden soll.

Aufgrund dieses Entscheides soll darauf folgend ein Prove-of-Concept erstellt werden.

Unter Berücksichtigung von aktuellen Software-Engineering-Methoden soll ein geeigneter Entwicklungsprozess definiert werden und darauf basierend die Evaluation durchgeführt und darauf aufbauend der Prove-of-Concept gebaut werden.

Technologien

- Java
- Eclipse
- Enterprise Architect

Generelles

- Die Vorgaben der Abteilung Informatik [2] sind einzuhalten.
- Die "Generelle Richtlinien für Studien- und Bachelorarbeiten" [3] sind einzuhalten.
- Mit dem CASE-Tool Enterprise Architect ist ein UML-Modell zu führen, welches synchron mit den Programm-Sourcen und der Projekt-Dokumentation ist.
- Ein Java-Entwickler muss mit der Projekt-Dokumentation in die Lage versetzt werden, die Applikation in Betrieb zu nehmen und weiter entwickeln zu können.

Termine

- Montag 19.09.11 Beginn der Studienarbeit
- Freitag 23.12.11 17:00 Uhr Abgabe der Studienarbeit

Betreuung

Thomas Letsch
tletsch@hsr.ch
055 - 22 24 567 (HSR Büro 5.204); 055 - 214 43 50 (Geschäft)

Referenzen

[1] Bachelorarbeit "3D-Class-Object-Visualization 3D-COV"
(25.05.2007, Dario Vonäsch)
[2] www.hsr.ch: HSR-intern>Bachelor-Studiengänge>Informatik>Allgemeine Infos
Diplom-, Bachelor- und Studienarbeiten
https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html
[3] "Generelle Richtlinien für Studien- und Bachelorarbeiten"
(v1.5 / 18.09.2011, Thomas Letsch)

Rapperswil, 19.September 2011

Thomas Letsch

Projekt: J3DEval

Abstrakt

Version: 1.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
17.12.2011	1.0	Erstellung des Dokuments	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Marion Walser verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
17.12.2011	Lara Mühlemann

1.4. Inhalt

Abstrakt

In früheren Studien- und Bachelorarbeiten wurde ein Tool entwickelt namens 3DCOV, mit dem sich Klassen- und Objektdiagramme erstellen lassen. Das Objektdiagramm wird dabei in einer 3D-Darstellung angezeigt.

Die Umsetzung der 3D-Darstellung erfolgte mit der Java 3D Library. Dadurch dass die Java 3D Library nicht komplett plattformunabhängig ist und unterschiedliche Installation verlangte, wurde dem Grundsatz von Java „write once, run anywhere" nicht vollauf entsprochen.

In dieser Arbeit soll nun diese Plattformunabhängigkeit von Java wieder erreicht werden. Dafür wurde eine Evaluation durchgeführt, in der verschiedene 3D-Libraries auf ihre Plattformunabhängigkeit und Kompatibilität mit Java analysiert wurden. Dabei wurde erkannt, dass keine Library mit diesen Vorgaben existiert. Die 3D-Libraries werden grundsätzlich für eine hohe Performance entwickelt und greifen deshalb auf die Hardware zu. Dies verhindert eine Plattformunabhängigkeit.

Es wurden zwei mögliche Varianten aus diesen Erkenntnissen genauer angeschaut. Erstens die Entkopplung der Java 3D Library von der Hardware und der Aufbau derselben auf Java 2D. Zweitens die Eigenprogrammierung der 3D-Darstellung aufbauen auf 2D, dabei wird Java3D nicht mehr verwendet. Die erste Variante erwies sich als unmöglich, da die Struktur der Objekte in der Java 3D Library nicht verwendet werden kann, um diese über Java 2D zeichnen zu lassen. Die zweite Variante war erfolgreich und Bedenken bezüglich Performance erwies sich als nur teilweise berechtigt.

Die Implementation aufbauend auf Java 2D konnte danach durchgeführt werden. Es konnten alle Funktionalitäten von 3DCOV wieder hergestellt werden. Die Performance ist soweit akzeptabel, dass nur bei leistungsschwächeren Rechnern die Bewegung des Objektdiagramms nicht fliessend ist.

Projekt: J3DEval

Management Summary

Version: 1.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
16.12.2011	1.0	Erstellung des Dokuments	Lara Mühlemann
21.12.2011	1.0	Korrektur aus Besprechung mit Betreuer	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Lara Mühlemann verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
17.12.2011	Marion Walser

1.4. Inhalt

2. Ausgangslage

Das Programm 3DCOV wird im Programmierunterricht verwendet, um das Lernen der objektorientierten Programmierung visuell zu unterstützten. Es benutzt die Java-3D-Bibliothek, welche sich nicht auf allen Computern gleich verhält. In neueren Windows-Versionen änderte sich das Verhalten bei der Anzeige des Tooltips, so dass dieser nicht mehr lesbar war. Zusätzlich erfordert es eine, je nach Betriebssystem unterschiedliche Installation der Java-3D-Bibliothek.

In dieser Arbeit soll eine Evaluation durchgeführt werden, in der ermittelt wird, welche Technologie stattdessen benutzt werden kann. Innerhalb dieser Evaluation soll ein Proof of Concept durchgeführt werden.

3. Vorgehen, Technologien

Als erstes wurde im Internet und über Bücher nach möglichen Lösungen gesucht und daraufhin die Informationen ausgewertet. Für die beiden besten Möglichkeiten folgte ein Proof of Concept. Anschliessend wurde für die bessere dieser Varianten mit einer spezifischen Entwicklung für das Programm 3DCOV begonnen. Die Technologie die schlussendlich verwendet wurde, war Java2D.

4. Ergebnisse

Das Proof of Concept ergab, dass durch Java 2D das erwünschte Ergebnis erreicht werden kann. Nach der Durchführung des Proof of Concepts wurde das Ergebnis zusätzlich umgesetzt.

In der folgenden Abbildung ist das Programm 3DCOV zu sehen, welches für die Darstellung des Objektdiagramms (auf der rechten Seite) die neue Technologie verwendet.

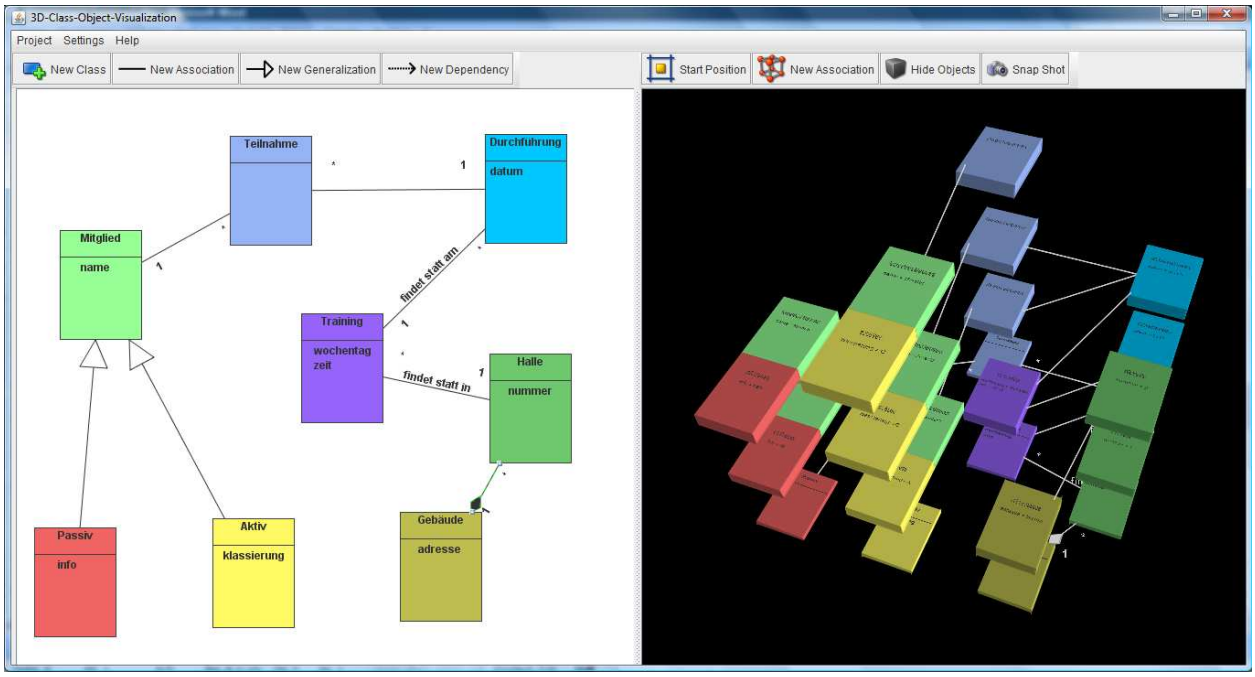


ABBILDUNG 1:3DCOV MIT NEUER LIBRARY FOCUSEDJ3D_ON_J2D

Die entstandene Lösung erlaubt das Hinzufügen von Verhalten wie:

- Rotation
- Verschiebung
- Zoom
- Selektion von Klassen, Objekten und Beziehungen

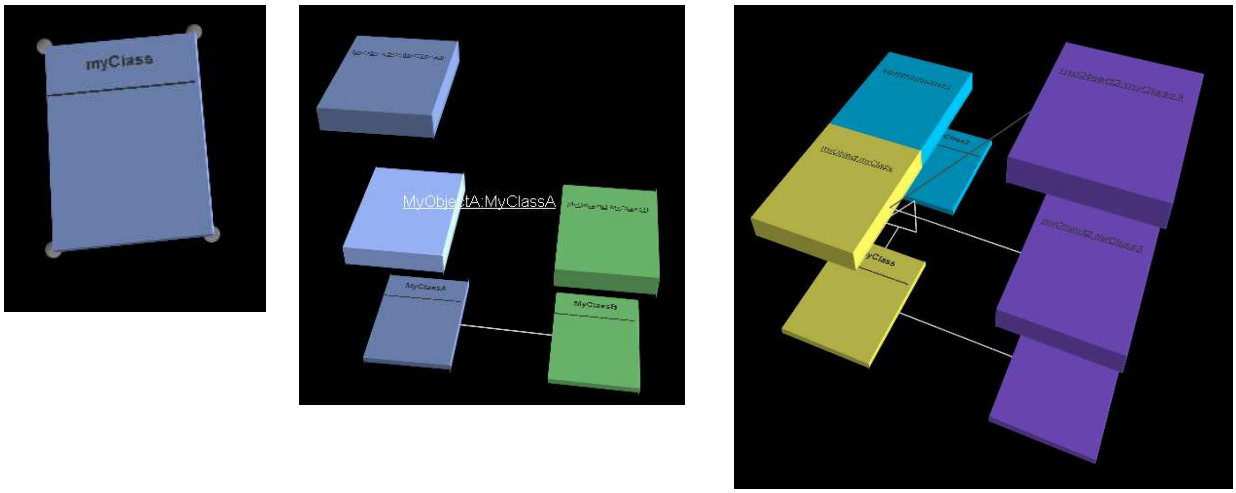


ABBILDUNG 2:V.L.N.R: SELEKTION VON KLASSEN, OBJEKTEN UND BEZIEHUNGEN

Zwei Einschränkungen sind bei dieser Variante zu berücksichtigen. Einerseits ist die Performance im Vergleich zur Originallösung erheblich schlechter. Bei leistungsstarken Rechnern zeigt sich dieser Effekt jedoch nur wenig. Weiter verschiebt sich das Diagramm bei Entfernen und Hinzufügen von Objekten bzw. Klassen leicht an eine andere Position. Diese Einschränkungen behindern das Arbeiten mit der Applikation nur unwesentlich.

5. Ausblick

Das Tool 3DCOV kann in Zukunft durch folgende Funktionalitäten erweitert weden:

- Ein Programm mit einfachem Klassendiagramm wird ausgeführt. Dieses Programm stellt 3DCOV sein Objektdiagramm zur Verfügung. 3DCOV merkt Änderungen in diesem Objektdiagramm automatisch und stellt diese dar (Vorschlag von Teambetreuer Herr Letsch).
- Ein Plugin für die Entwicklungsumgebung Eclipse, welches bei der Ausführung von einfachen Programmen in der Lage ist, deren Objektdiagramm auszulesen und 3DCOV zur Verfügung zu stellen.

Projekt: J3DEval

Technischer Bericht

Version: 1.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
12.12.2011	1.0	Erstellung des Dokuments	Lara Mühlemann
13.12.2011	1.0	Kapitel "Entkoppeln" ausgeführt	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Lara Mühlemann verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
21.12.2011	Lara Mühlemann

1.4. Inhalt

2. Einleitung und Übersicht

Dieser Arbeit liegt ein bereits bestehendes Tool zu Grunde, welches 3DCOV heisst. 3DCOV stellt Klassendiagramme in 2D und Objektdiagramme in 3D dar. 3DCOV nutzte bis anhin für die Darstellung des 3D-Objektdiagramms die Library Java3D. Dies birgt den Nachteil, dass jeder Nutzer von 3DCOV Java3D installieren muss, weshalb diese Lösung nicht komplett plattformunabhängig ist. Zusätzlich tritt bei Verwendung der Applikation auf den neusten Windows-Versionen ein Fehler beim Anzeigen des Tooltips im Objektdiagramm auf, was ein weiterer Grund lieferte für das Ersetzen der Library Java3D.

In der Arbeit J3DEval wird eine Evaluation durchgeführt, mit derer nach der besten plattformunabhängigen Lösung für die Darstellung des Objektdiagramms gesucht wird. Im Rahmen dieser Evaluation wird ein Proof of Concept erstellt.

3. Ergebnisse

3.1. Evaluation

Die Evaluation beinhaltet Kriterien, nach denen die Lösungen bewertet werden. In der Erwartung eine bestehende Library verwenden zu können, sind in der Evaluation 38 3D-Libraries geprüft. Darunter erfüllte jedoch keine alle Grundvoraussetzungen. Die erarbeiteten Lösungsvarianten sind aufgrund der Kriterien geprüft und mit Hilfe einer Matrix ausgewertet worden. Folgend sind die vier Lösungsvarianten aufgelistet:

- Ersetzen der Java3D API durch Eigenprogrammierung
- Ersetzen der Java 3D API und des Packages „cov3dmodel" im 3DCOV durch Eigenprogrammierung
- Java 3D API von Hardware entkoppeln
- LWJGL (Lightweight Java Game Library) von Hardware entkoppeln

Aufgrund der Auswertung wurde entschieden, dass für die Varianten „Ersetzen der Java3D API durch Eigenprogrammierung" und „Java 3D API entkoppeln" ein Proof of Concept erstellt wird.

3.2. Proof of Concept

3.2.1. Ersetzung der Java3D API durch Eigenprogrammierung

Da der Code von 3DCOV nicht verändert werden soll, wird die Schnittstelle von 3DCOV zu Java3D übernommen. Dies bedeutet für die Eigenprogrammierung, dass eine Struktur eingehalten werden muss, die für die Umsetzung mit Java 2D nicht ideal ist.

Trotzdem ist mit dem Prototyp zu Release 1 erwiesen, dass mittels Java2D 3D-Boxen gezeichnet, rotiert und verschoben werden können und die Performance mit vier Objekten auf den Testrechnern gut ist.

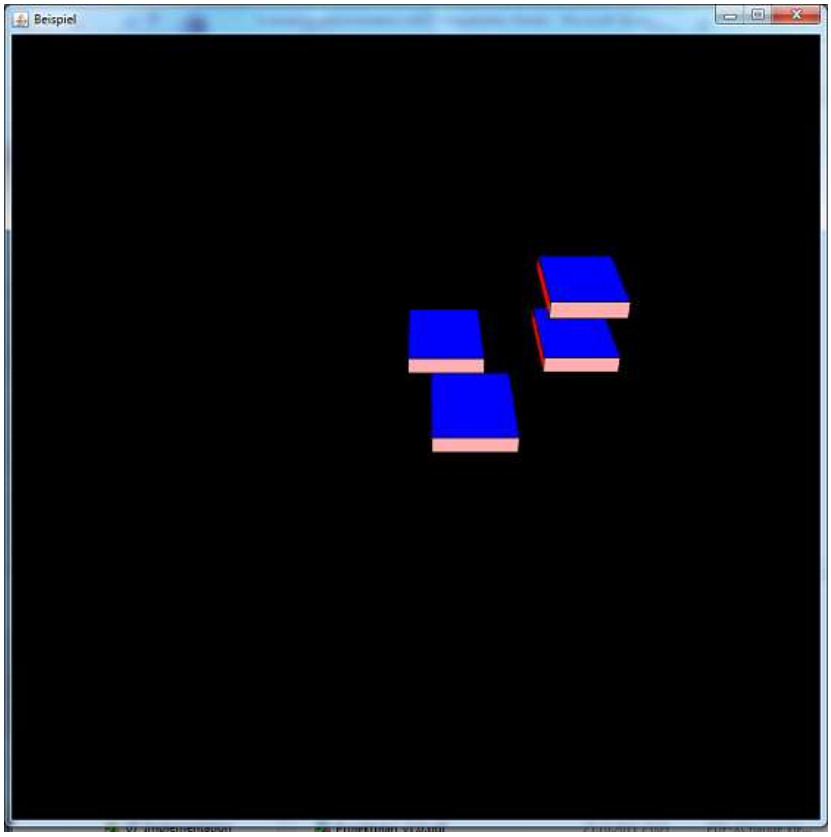


ABBILDUNG 1: PROTOTYP IN RELEASE 1

3.2.2. Java 3D API von Hardware entkoppeln

Aufgrund der Komplexität und Grösse der Library war nicht auf Anhieb ersichtlich, wie die Library mit dem Objekte umgeht und was auf welcher Ebene verwaltet und berechnet wird.

Zuerst wurden die vielen Klassen analysiert und einiges an Literatur dazu gelesen. Die Library bietet viele Möglichkeiten und eine hohe Performance, die auch für richtige Spielprogrammierung benötigt werden. Wie jedoch alle evaluierten Libraries entsprachen diese Fähigkeiten nicht den Anforderungen für die 3DCOV-Applikation.

Schlussendlich musste erkannt werden, dass die 3D-Objekte in der Java 3D API nicht so verwaltet wurden, die es erlaubt hätte, die Library von dem Direktzugriff auf die Hardware zu lösen und auf Java 2D aufzubauen. Die Objekte werden, bevor sie gezeichnet und auf 2D projektiert werden, in kleiner Teilobjekte unterteilt (entsprechend OpenGL), bevor jegliche Berechnung für Bewegung und Projektion ausgeführt wird.

Dadurch verschloss sich auch die Möglichkeit, Teile von Java 3D API in die aktuelle Library einfliessen zu lassen. Und es wurde entschieden, nicht weiter an diese Variante zu arbeiten.

3.2.3. Entscheidung

Aufgrund der Ergebnisse aus Release 1 wurde entschieden, dass mit der Variante „Ersetzung der Java3D API durch Eigenprogrammierung" weitergefahren wird.

3.3. Weiterführende Arbeiten

Die Library wurde in weiterführenden Arbeiten soweit entwickelt, dass 3DCOV diese als Ersatz für Java3D verwenden kann. Die folgende Abbildung zeigt ein Beispiel eines Objektdiagramms (in der rechten Hälfte des Bildes). Das Ergebnis läuft bei leistungsfähigen Computern mit einer guten Performance. Bei schwächeren Computern kann zum Teil stärkere Verzögerung beim Zeichnen auftreten.

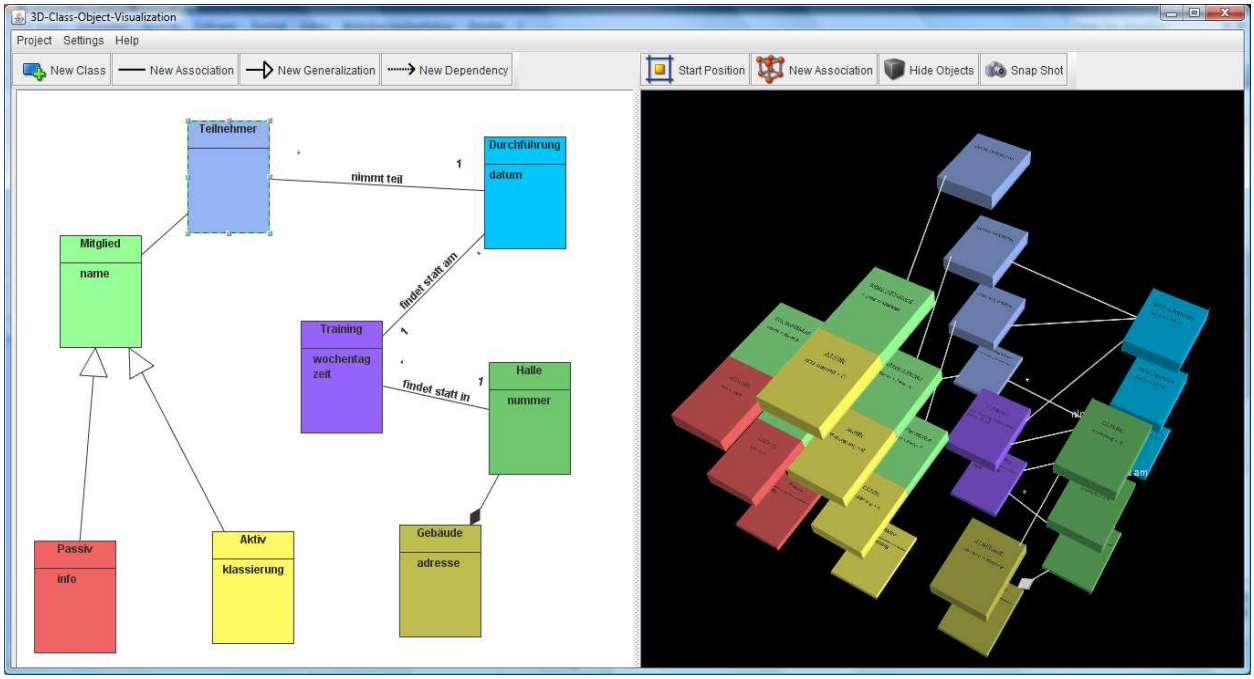


ABBILDUNG 2: ERGEBNIS ZU RELEASE 3

Folgend werden einige erwähnenswerte Eigenschaften von 3DCOV mit der neuen Library ausgeführt.

3.3.1. Behaviors

Die Library bietet die Möglichkeiten Verhalten zu Gruppen oder zum ganzen Canvas hinzuzufügen. In den folgenden Unterkapiteln sind die von 3DCOV umgesetzten Behaviors beschrieben.

3.3.1.1. Selektion

Im Objektdiagramm können Klassen, Objekte und Beziehungen selektiert werden. Klassen erhalten an den Ecken Kugeln, wenn sie selektiert sind, Objekte erscheinen heller und Beziehungen werden dunkelgrau.

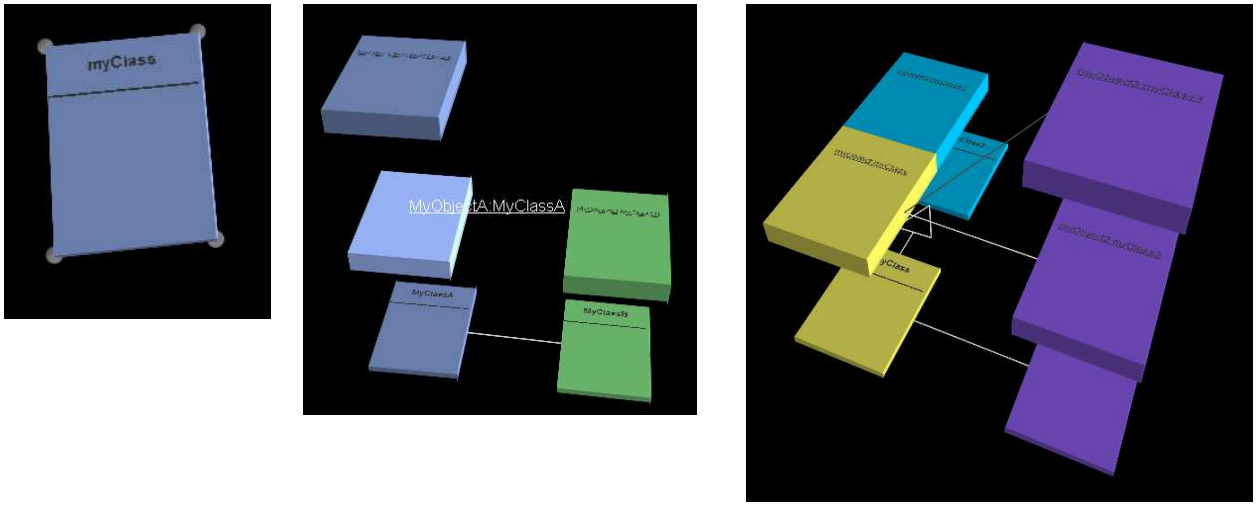


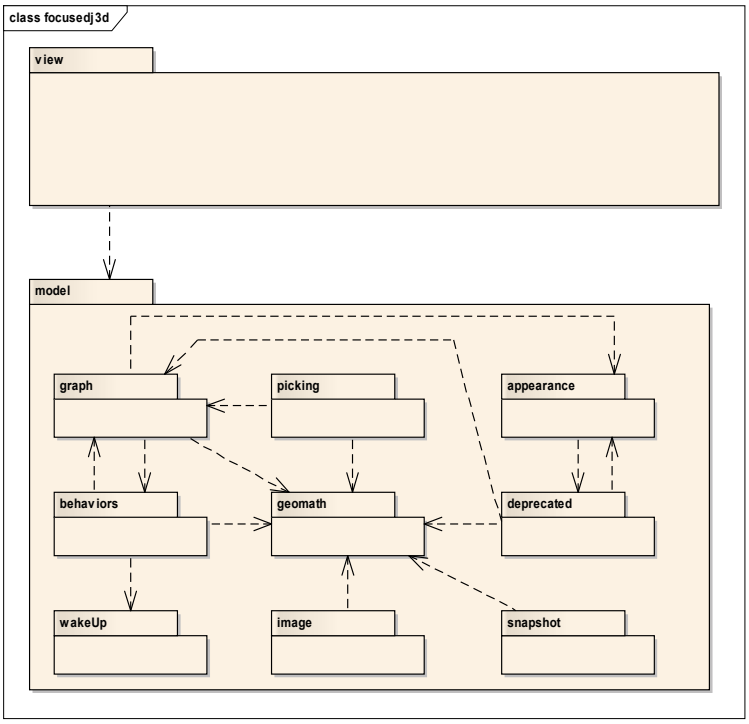
ABBILDUNG 3:V.L.N.R: SELEKTION VON KLASSEN, OBJEKTEN UND BEZIEHUNGEN

3.3.1.2. Tooltip

Wenn der Benutzer mit der Maus über ein Objekt fährt, erscheint ein Tooltip. Dieser zeigt den Klassennamen, den Objektnamen und die Attribute mit ihren Werten an.

ABBILDUNG 4:TOOLTIP

3.4. Architektur



Die Implementierung des User Iinterface's ist von der Domain-Logik getrennt. Dazu verwendet werden die Schichten view und model.

Da die Datenspeicherung von 3DCOV übernommen wird, ist keine Persistenzschicht notwendig.

Die Architektur ermöglicht einfach, neue Geometrieobjekte hinzuzufügen. Dazu muss lediglich eine Klasse implementiert werden, welche von DrawableObject ableitet. In dieser können die mathematischen Berechnungen implementiert werden. Das package view benötigt nur noch eine Ergänzung für das Zeichnen mit java2d.

ABBILDUNG 5:PACKAGE DIAGRAMM

4. Schlussfolgerungen

Im Verlaufe dieser Arbeit wurde nicht nur eine Evaluation und ein Proof of Concept erarbeitet. Die Library focusedj3d_on_j2d kann von 3DCOV zur Darstellung des Objektdiagramms benutzt werden. Dabei geht keine Funktionalität von 3DCOV verloren. Einschränkungen wurden bei der Darstellung von Beleuchtung gemacht. Es wird kein Lichteinfall simuliert, wie das Java3D umsetzt. Als Ersatz erscheinen die Seitenflächen der Boxen in verschiedenen Helligkeiten.

Java2D hat sich als geeignetes Werkzeug für das plattformunabhängige Zeichnen des Objektdiagrammes erwiesen. Allerdings ist eine gewisse Leistung des Computers für die flüssige Darstellung von Bewegungen notwendig.

Kleinere Verbesserungen in der Library focusedj3d_on_j2d können noch vorgenommen werden.

- Bei Veränderungen im Klassendiagramm, oder beim Hinzufügen/Löschen von Objekten springt das Objektdiagramm an eine leicht andere Position. Eine Lösungsvariante ist im System Architecture Document vermerkt.
- Die Beschriftung der Beziehungen und Rollen ist in der aktuellen Lösung immer horizontal ausgerichtet. Diese könnte in einer späteren Version mit rotieren. Dabei ist aber zu berücksichtigen, dass dies die Lesbarkeit des Textes einschränken könnte.

Das Tool 3DCOV kann in Zukunft durch folgende Funktionalitäten erweitert weden:

- Ein Programm mit einfachem Klassendiagramm wird ausgeführt. Über eine Schnittstelle (z.B XML) stellt dieses Programm 3DCOV sein Objektdiagramm zur Verfügung. 3DCOV registriert Änderungen in diesem Objektdiagramm automatisch und stellt diese dar.
- Ein Eclipse-Plugin welches bei der Ausführung von einfachen Programmen in der Lage ist deren Objektdiagramm auszulesen und mittels der im vorherigen Punkt beschriebenen Schnittstellen 3DCOV zur Verfügung zu stellen.

Projekt: J3DEval

Persönliche Berichte

Version: 1.0
Datum: 22. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
15.12.2011	1.0	Erstellung des Dokuments, erster Erfahrungsbericht	Marion Walser
19.12.2011	1.0	Zweiter Erfahrungsbericht	Lara Mühlemann

1.2. Verantwortlichkeit

Für dieses Dokument ist Marion Walser verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
19.12.2011	Marion Walser + Lara Mühlemann

1.4. Inhalt

2. Einführung

2.1. Zweck

In diesem Dokument beschreiben die Teammitglieder die Erfahrungen in diesem Projekt. Es sollen die Erfahrungen ausgeführt werden und eine Reflexion der Arbeit zeigen.

3. Erfahrungsbericht Marion Walser

3.1. Projektverlauf

Der Beginn war ziemlich harzig. Die Einarbeitung in 3D mit den vielen spezifischen Begriffen war aufwändig und die Beurteilung der vielen 3D-Libraries schwierig. Die Evaluation gestaltete sich als trockene Knochenarbeit, gab aber einen guten Einblick in die Welt der 3D-Computerprogramme.

Als es dann ans Programmieren ging, lief es etwas runder. Fleissig implementierten wir die Funktionalitäten. Wie von RUP empfohlen, haben wir iterativ die Use Cases implementiert. Im Nachhinein würde ich jedoch sagen, dass es in diesem Projekt vermutlich besser gewesen wäre, alle Use Cases zuerst zu analysieren. Damit hätten wir die Abhängigkeiten zwischen den Use Cases, die gleiche Funktionalitäten in der darunter liegenden Library nutzen, früher erkennen können. So mussten wir einzelne Male die Implementierung umstellen, damit sie auch für den nächsten Use Case passt.

3.2. Zusammenarbeit im Team

Die Verteilung der Arbeit auf die zwei Teammitglieder war etwas schwieriger als im letzten Projekt, da es zu Beginn der Programmierarbeit nicht solch eine klare Schichtenarchitektur erstellt werden konnte und viele Use Cases von der Implementierung anderen abhängig waren. Dadurch ergaben sich einzelnen Situationen, in den beide an der gleichen Aufgabe arbeiteten. Da wäre wohl manchmal eine Absprache sinnvoll gewesen.

Teilweise war es jedoch auch beabsichtigt, dass wir gemeinsam am gleichen Problem arbeiteten, damit wir mit vereinten Kräften die Lösung finden konnten.

3.3. Fazit

Ich konnte viel lernen in diesem Projekt, vor allem viel über 3D-Visualisierung. Weiter war es interessant, eine untere Schicht eines Programms zu ersetzen und zu erfahren, welchen Einschränkungen man vorgegeben bekommt. Auch kleinere Erfahrungen wie ein erster Einblick in AspectJ und Kennenlernen der XStream-Library ergaben sich in diesem Projekt.

Auch wenn es nicht mehr ganz so unterhaltsam war wie ein Computerspiel (SE2-Projekt) zu implementieren, war das Projekt interessant und die Erfahrungen werden mir sicher zukünftig nützlich sein.

4. Erfahrungsbericht Lara Mühlemann

4.1. Projektverlauf

Wir haben uns auf dieses Thema beworben, weil wir uns für die 3D-Programmierung interessierten. Jedoch haben wir vor diesem Projekt noch nie eine 3D-Applikation umgesetzt. Dementsprechend schwierig war es für mich, mir im Vorfeld dieses Projekt vorzustellen. Zu Beginn versuchten wir uns in die Thematik einzulesen, uns damit ein Bild über die 3D-Programmierung zu machen und uns über die vorhandenen Technologien zu informieren. Dabei habe ich rückblickend zu viele Bücher organisiert. Viele Bücher waren für uns nicht brauchbar und so erstickten wir etwas in der Informationsflut. Schlussendlich war nur eines dieser Bücher für uns von Nutzen.

Während der Evaluation untersuchten wir die verschiedenen Lösungsmöglichkeiten. Dieser Projektabschnitt war etwas enttäuschend, da wir sehr viele Libraries evaluierten und sich alle als ungeeignet herausstellten.

Beim Proof of Concept fand ich zu Beginn schwierig, dass uns nur ein Java2D-Beispiel zur Verfügung stand und dieses schwierig zu verstehen war, da der Code nicht refactored und die Namensgebung nicht sehr verständlich gewählt war. Trotzdem stellte sich schon relativ früh ein Erfolgserlebnis ein und es konnten 3 Boxen im Raum dargestellt werden.

Bei der Weiterentwicklung war die Hauptproblematik, dass man sehr oft durch die Schnittstelle eingeschränkt wurde. Es war zum Teil frustrierend, als ich ein gutes Design umsetzen wollte und dies durch die Schnittstelle nicht möglich war.

Trotzdem schritt das Projekt sehr zügig fort und einige Funktionalitäten konnten sogar schneller implementiert werden als gedacht.

4.2. Zusammenarbeit im Team

Da wir bereits mehrere Projekte zusammen erarbeitet haben, waren wir bereits ein eingespieltes Team. Über die wöchentlichen Sitzungen konnten die anstehenden Arbeiten koordiniert werden. Da zusätzlich auch oft am selben Ort gearbeitet wurde, konnten sich diese Besprechungen gegen Ende des Projektes sogar stark verkürzen.

4.3. Fazit

Das Projekt hat viel Spass gemacht, war sehr lehrreich und wir wurden vom Betreuer gut unterstützt.

Projekt: J3DEval

Glossar

Version: 4.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
27.09.2011	1.0	Erstellung des Dokuments	Lara Mühlemann
10.10.2011	2.0	Glossar auf Anforderungsspezifikation angepasst	Lara Mühlemann
14.10.2011	2.0	Begriff cov3dmodel ergänzt	Marion Walser
30.10.2011	3.0	Begriff Canvas ergänzt	Marion Walser
07.11.2011	3.0	Begriff Konstrukt und XStream ergänzt	Marion Walser

1.2. Überprüfung

Datum	Durchgeführt von
27.09.2011	Marion Walser
01.11.2011	Lara Mühlemann
17.12.2011	Marion Walser
19.12.2011	Lara Mühlemann

1.3. Inhalt

2. Glossar

Begriff	Beschreibung
3DCOV	Bestehende Applikation, bei der die 3D Library ersetzt werden soll.
Appearance	Appearance definiert das Aussehen der Oberfläche von dem Objekt, dem sie zugeordnet ist.
Attribut	Attribute gehören zu einer Klasse und deren Objekten und sind Eigenschaften oder verwaltete Daten. Dies ist ein Begriff aus der objektorientierten Softwareentwicklung.
BranchGroup	Die BranchGroup dient als Wurzel eines Baums von Objekten
Canvas	Canvas bedeutet Leinwand. Im Kontext von 3D-Zeichen ist dies der Bereich im View, in dem die 3D-Darstellung erfolgen soll.
cov3dmodel	Package in 3DCOV-Applikation, das die Darstellung der 3D-Objektdiagramm erstellt. Dieses Package hat in der 3DCOV Applikation die meisten Zugriffe auf die Java 3D API zu.
focusedj3d_on_j2d	Name der in diesem Projekt enstandenen Library
J3DEval	Name dieses Projektes, steht für Java 3D Evaluation
Java 3D API	Java-Bibliothek zur Darstellung von 3D-Grafiken
Klasse	Klassen werden mithilfe des Tools 3DCOV dargestellt. Dies ist ein Begriff aus der objektorientierten Softwareentwicklung.
Klassendarstellung	In der 3D-Visualisierung von 3DCOV werden Objekte und ihre zugehörigen Klassen dargestellt. Die Klassendarstellung ist die unterste Ebene dieses Diagramms und beinhaltet das flache Klassendiagramm.
Klassendiagramm	Fenster von 3DCOV, welches die Klassen in 2D darstellt und sich auf der linken Seite im Applikationsfenster befindet.
Konstrukt	3D-Darstellung der Objekte und Klassen im 3DCOV (Inhalt des Objektdiagramms)
Objekt	Objekte werden mithilfe des Tools 3DCOV dargestellt. Dies ist ein Begriff aus der objektorientierten Softwareentwicklung.
Objektdiagramm	Fenster von 3DCOV, welches die Objekte und die Klassen in 2D darstellt und sich auf der rechten Seite im

	Applikationsfenster befindet.
Proof of Concept	Nachweis, dass etwas umsetzbar ist
Transform3D	Transform3D definiert eine Bewegung (Translation, Rotations).
TransformGroup	TransformGroup enthält Objekte, die sich alle synchron bewegen müssen.
XStream	Library, die von 3DCOV für die XMI-Schnittstelle benutzt wird

Projekt: J3DEval

Literaturverzeichnis

Version: 1.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
16.12.2011	1.0	Erstellung des Dokumenten, Aufstellung der verwendeten Bücher und Referenzen aus Evaluation	Marion Walser
21.12.2011	1.0	Hinzufügen von Literatur	Lara Mühlemann

1.2. Verantwortlichkeit

Für dieses Dokument ist Marion Walser verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
21.12.2011	Lara Mühlemann + Marion Walser

1.4. Inhalt

2. Einführung

2.1. Zweck

Dieses Dokument listet die in dieser Arbeit verwendete Literatur und die verwendeten Referenzen auf.

2.2. Gültigkeitsbereich

Dieses Dokument ist für die Studienarbeit J3DEval gültig.

3. Literaturliste

3.1. Allgemeine Literatur über 3D-Darstellungen am Computer

[1] Java 3D Programming, Daniel Selman, Manning Verlag, 2002
[2] Grundkurs Computergrafik mit Java, Frank Klawonn, Vieweg + Teubner Verlag, 2005
[3] 3D-Computergrafische Darstellungen, K.D.Tönnies / H.U.Lemke, Oldenburg Verlag, 1994

3.2. Referenzen und Literatur zur Implementierung

[1] http://java.sun.com/products/java-media/2D/samples/suite/Colors/Rotator3D.java

3.3. Referenzen zu Libraries für Evaluation

[1] http://code.google.com/p/java-m3g/
[2] http://code.google.com/p/openskia/
[3] http://de.wikipedia.org/wiki/Java_3D
[4] http://de.wikipedia.org/wiki/Jogl
[5] http://de.wikipedia.org/wiki/Open_Inventor
[6] http://de.wikipedia.org/wiki/X3D
[7] http://developer.hoops3d.com/documentation/Hoops3DGS/tech_overview/TechnicalOverview_C.html#_Toc470077868
[8] http://en.wikipedia.org/wiki/Away3D
[9] http://en.wikipedia.org/wiki/ClanLib
[10] http://en.wikipedia.org/wiki/CrystalSpace
[11] http://en.wikipedia.org/wiki/Glide_API
[12] http://en.wikipedia.org/wiki/Graphics_library
[13] http://en.wikipedia.org/wiki/HOOPS_3D_Graphics_System
[14] http://en.wikipedia.org/wiki/JMonkey_Engine
[15] http://en.wikipedia.org/wiki/JT_%28visualization_format%29
[16] http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries
[17] http://en.wikipedia.org/wiki/Microsoft_Direct3D
[18] http://en.wikipedia.org/wiki/MiniGL

[19] http://en.wikipedia.org/wiki/Mobile_3D_Graphics_API
[20] http://en.wikipedia.org/wiki/O3D
[21] http://en.wikipedia.org/wiki/OGRE
[22] http://en.wikipedia.org/wiki/OpenGL
[23] http://en.wikipedia.org/wiki/OpenGL_Performer
[24] http://en.wikipedia.org/wiki/OpenSceneGraph
[25] http://en.wikipedia.org/wiki/OpenSG
[26] http://en.wikipedia.org/wiki/QSDK
[27] http://en.wikipedia.org/wiki/RenderMan_Interface_Specification
[28] http://en.wikipedia.org/wiki/RenderWare
[29] http://en.wikipedia.org/wiki/WebGL
[30] http://irrlicht.sourceforge.net/
[31] http://jausoft.com/gl4java.html
[32] http://jausoft.com/products/gl4java
[33] http://jmonkeyengine.org/downloads/
[34] http://lwjgl.org
[35] http://pointclouds.org/
[36] http://pointclouds.org/downloads/
[37] http://www.ambiera.com/copperlicht/
[38] http://www.jogl.info
[39] http://www.libsdl.org/
[40] http://www.mak.com/products/visualize/common-operating-picture.html
[41] http://www.nvidia.de/object/scenix-home.html
[42] http://www.opengl.org/wiki/Getting_started
[43] http://www.panda3d.org/
[44] http://www.plm.automation.siemens.com/en_sg/products/open/jtopen/technology/jt_showcase.shtml
[45] http://www.presagis.com/products_services/products/ms/visualization/vega_prime/
[46] http://www.quesa.org/
[47] http://www.segger.com/cms/emwin.html
[48] http://www.sfml-dev.org/index.php
[49] http://www.web3d.org/about/overview/

Projekt: J3DEval

Projektplan

Version: 2.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
20.09.2011	1.0	Erstellung des Dokuments. An Kapitel 1 - 2 geschrieben.	Lara Mühlemann
23.09.2011	1.0	Kapitel 2 - 4 geschrieben	Lara Mühlemann
25.09.2011	1.0	Kapital 5 – 9 ergänzt	Marion Walser
26.09.2011	1.0	Ergänzung in Qualitätsmanagement, Artefaktplan angepasst.	Lara Mühlemann
1.10.2011	1.0	Überarbeitung aufgrund Feedback von Betreuer	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Lara Mühlemann verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
27.09.2011	Lara Mühlemann, Marion Walser
07.10.2011	Lara Mühlemann
17.10.2011	Lara Mühlemann
17.12.2011	Lara Mühlemann, Marion Walser

1.4. Inhalt

2. Einführung

2.1. Zweck

Der Zweck dieses Dokuments ist, die Planung des Projektes J3DEval zu dokumentieren.

2.2. Gültigkeitsbereich

Dieses Dokument ist während der Projektdauer gültig.

2.3. Definitionen und Abkürzungen

Siehe Glossar.doc

2.4. Referenzen

[1] Zeitplan.xls

[2] Aufgabenstellung.pdf

[3] Iterationsplan: 01_Iterationsplan_Elaboration1.doc

[4] Systemtests.doc

[5] Code-Reviews.doc

[6] Dokuments „Code Conventions.pdf" von Sebastian Pötzsch

3. Projekt Übersicht

Im Projekt J3DEval wird die bestehende Applikation 3DCOV angepasst. Konkret geht es darum die verwendete Library Java3D durch eine betriebsystem- und hardware-unabhängige Lösung zu ersetzen.

3DCOV ist eine Applikation, die den Zusammenhang zwischen Klassen und Objekten visualisiert. Das Objekt/Klassendiagramm wird in 3D dargestellt.

Im Rahmen des Projektes J3DEval wird eine Evaluation durchgeführt. Darin wird nach der passendsten Lösung gesucht, welches durch ein Proof of Concept auf ihre Machbarkeit geprüft wird. Anschliessend wird die Lösung in die bestehende Applikation 3DCOV integriert.

3.1. Zweck und Ziel

Das Ziel des Projekts ist es, den 3D-Teil der 3DCOV Applikation so anzupassen, dass keine zusätzlichen Installationen notwendig sind, wie dies mit der Java 3D API erfolgen muss. Nach Möglichkeit soll auch keine zusätzliche Library mit der Applikation benötigt werden. Weiteres ist im Dokument „Aufgabenstellung.pdf" [2] zu finden.

3.2. Annahmen und Einschränkungen

Der Abgabetermin für das Projekt ist der Freitag, 23.12.2011 um 17:00 Uhr. Zusätzlich wird vom Auftraggeber der 30.11.11 um 7:00 Uhr als Termin für den 2. Release vorgegeben.

4. Projektorganisation

Das Team besteht aus zwei Personen, die gleichberechtigt sind. Probleme werden zusammen diskutiert und Entscheidungen gemeinsam gefällt.

Jedes Teammitglied ist für einzelne Bereiche verantwortlich. Diese Verantwortlichkeit besteht aus der Organisation und Koordination dieses Bereiches.

4.1. Organisationsstruktur


ABBILDUNG 1: MÜHLEMANN LARA (LINKS), WALSER MARION (RECHTS)

In der folgenden Tabelle ist aufgeführt, welches Teammitglied welche Verantwortlichkeiten als Projektleiter trägt.

Projektmitglied	Verantwortlichkeit
Mühlemann Lara	• Evaluation • 2. Release
Walser Marion	• 1. Release • 3. Release

Informationen, Fragen oder Aufträge sollen immer an beide Projektmitglieder gerichtet werden. Allfällige Arbeiten oder Abklärungen werden danach an der nächsten Besprechung zugeteilt bzw. besprochen.

4.2. Externe Schnittstellen

Name	Mailadresse	Rolle
Letsch Thomas	tletsch@hsr.ch	Verantwortlicher, Betreuer

Mühlemann Lara	lmuehlem@hsr.ch	Projektmitglied
Walser Marion	mwalser@hsr.ch	Projektmitglied

5. Management Abläufe

5.1. Projekt Kostenvoranschlag

Für das Projekt stehen 14 Wochen zur Verfügung. In dieser Zeit arbeiten zwei Teammitglieder gesamthaft 480 Stunden an dem Projekt. Dies gibt ein Zeitaufwand von 17 – 18 Stunden pro Woche.

5.2. Projektplan

5.2.1. Zeitplan

Siehe [1] Zeitplan.xls

5.2.2. Artefaktplan

Phase	**Inception**	**Elaboration 1 / Grobevaluation**		**Elaboration 2 /Feinevaluation**			**Construction 1**				**Construction 2**		**Transition**	
						1. Release				2. Release				3. Release
Zeitabschnitt beginnend am						2. 11.2011	3. 11.			28.11.2011	01.12.2011	08.12.2011	15.12.2011	18.12.2011
Projektwoche										SW11 (1)	SW 11 (2) / SW 12 (1)	SW 12 / 13 (1)	SW 13 (2)	SW 14
Projekt Management														
Projektplan	v1.0													v2.0
Zeitplan		v1.0		v2.0			v3.0				v4.0		v5.0	v6.0
Iterationspläne inkl. Risikomanagement		1		2			3				4		5	
Sitzungsprotokolle	1	2	4	5	6	7	8	9	10	11	12	13		14
Glossar	v1.0	v1.0	v2.0			v3.0								v4.0
ToDo und Bug-Liste														v1.0
Evaluation														
Evaluationsdokumentation			v1.0			v2.0								V3.0
Requirements														
Anforderungsspezifikation			v1.0			v2.0				v3.0				v4.0
Business Modeling														
Domainanalyse						v1.0				v2.0				v3.0
Architektur + Design														
System Architecture Document										v1.0			v2.0	v3.0
Implementation														
Funktionsumfang Releases				v1.0			v2.0				v3.0			v4.0
Release 1						v1.0								v2.0
Release 2										v1.0				v2.0
Release 3														v1.0
Test + Qualität														
Systemtests						v1.0				v2.0				v3.0
Code-Reviews						v1.0				v2.0				v3.0
Code-Conventions			v1.0											v2.0
Abschlussbericht-/dokumente														
Aufgabenstellung														v1.0
Abstract														v1.0
Management Summary														v1.0
Technischer Bericht														v1.0
Persönliche Berichte														v1.0
Literaturverzeichnis														v1.0
Dokumentenübersicht														v1.0
Kurzfassung														
Studienarbeit														v1.0
Poster														v1.0

5.2.3. Beschreibung Artefakte

Artefakte	Beschreibung
Projektplan	Beschreibung über den geplanten Ablauf des Projekts.
Zeitplan	Zeitplanung des Projektes mit Arbeitspaketen, Soll- und Ist-Stunden. Der Zeitplan wird jeweils anfangs Iteration mit den Soll-Zeiten für die Iteration ergänzt.
Risikomanagement	Aufstellung über die Risiken des Projekts, wird in den Iterationspläne erstellt bzw. aktualisiert
Iterationspläne	Definition und Einteilung der Aufgaben einer Iteration inkl. Neubewertung der Risiken zu Beginn der Iteration. Zu jeder Iteration wird ein eigenes Dokument erstellt.
Sitzungsprotokolle	Dokumentation der Sitzungen mit Entscheidungen und Beschlüssen
Glossar	Auflistung der projektspezifischen Begriffe mit Erklärungen
To-Do- und Bug-Liste	Liste der ToDos und Bugs mit Erstellungsdatum und Erledigt-Datum
Evaluationsdokumentation	Dokumentation zur Evaluationsphase
Anforderungsspezifikation	Detaillierte Auflistung und Beschreibung der funktionalen und nicht funktionalen Anforderungen
Domainanalyse	Beschreibung der Domäne mittels Darstellung der Konzepte in einem Domain Model. Use Cases werden mittels Sequenzdiagrammen dargestellt und die Kontrakte der wichtigsten Funktionen definiert.
System Architecture Document	Beschreibung und Darstellung der Architektur, Packages und des Designs (Klassen)
Funktionsumfang Releases	Definition des Funktionsumfangs der drei Releases, wird jeweils zu Beginn der Iteration erstellt
Release 1	Dokumentation zu Release 1. Siehe Kapitel Meilensteine: MSI *
Release 2	Dokumentation zu Release 2. Siehe Kapitel Meilensteine: MSII *
Release 3	Dokumentation zu Release 3. Siehe Kapitel Meilensteine: MSIII *
Systemtests	Beschreibung und Auswertung der Systemtest
Code-Reviews	Dokumentation der Ergebnisse aus den Code-Reviews

Code-Conventions	Codestyleguide, nach denen in diesem Projekt gearbeitet wird
Aufgabenstellung	Detaillierte Aufgabenstellung zur Studienarbeit
Abstract	Abriss des Projekts, richtet sich an den Spezialisten. *
Management Summary	Kompakte Darstellung des Projekt auf abstrakter Ebene. *
Technischer Bericht	Ergebnisse und Schlussfolgerungen aus dem Projekt. *
Persönliche Berichte	Bericht jedes Projektmitglied über die Erfahrungen bei der Arbeit
Literaturverzeichnis	Auflistung der verwendeten Quellen *
Dokumentenübersicht	Übersicht über die erstellten Dokumenten
Kurzfassung Studienarbeit	Inhalt des Abstract mit Metadaten. *.
Poster	Zusammenfassung der Arbeit auf einem Poster. Das Poster enthält Redundanz zu anderen Dokumenten.

* Diese Berichte enthalten Redundanz zu anderen Dokumenten.

5.2.4. Meilensteine

Meilen-stein	Beschreibung	Anforderungen an Releases	Dauer	Termin
MS I	Ende Elaboration 1 / Grobevaluation		3 Wochen	12.10.2011
MS II	Ende Elaboration 2 / Feinevaluation Release 1	Erste Funktionalitäten implementiert (z.B. Darstellung von drei Objekten im Raum und Rotation derselben).	3 Wochen	03.11.2011 07:00 Uhr
MS III	Ende Construction 1 Release 2	Weitere Funktionalitäten sind implementiert. Bei diesem Release soll das UML-Modell des Enterprise Architect und die Sourcen synchron sein (gemäss "Generelle Richtlinien für Studien- und achelorarbeiten").	3.5 Wochen	30.11.2011 07:00 Uhr
MS IV	Ende Construction 2		2 Wochen	14.12.2011
MS V	Ende Transition Release 3	Abgabe-Release der Studienarbeit.	8 Tage	23.12.2011 17:00 Uhr

Die offenen Daten werden, sobald sie bestimmt sind, in den Iterationsplänen ausgeführt und in der End-Version des Projektplans nachgeliefert.

5.2.5. Besprechungen

Das Team trifft sich jeden Montag für etwa eine Stunde, um aktuelle Themen zu besprechen und die Aufgaben für die Woche zu definieren.

Am Mittwoch um 15 Uhr findet jeweils die Sitzung mit dem Betreuer statt. Das Protokoll dazu ist innert 24 Stunden nach Sitzungsende an den Betreuer abzuliefern.

6. Risikomanagement

Das Risikomanagement befindet sich jeweils in den Iterationsplänen. Die erste Version des Risikomanagements ist im Iterationsplan Elaboration 1 [3] und wird in den darauf folgenden Iterationsplänen neu beurteilt.

7. Arbeitspakete

Die Arbeitspakete sind im Zeitplan [1] definiert.

8. Infrastruktur

8.1. Räumlichkeiten

Die Besprechung mit dem Übungsbetreuer findet im Besprechungsraum 5.207 statt. Teamarbeit findet hauptsächlich in dem den Studenten zugeteilten Studienarbeitsraum statt.

Weiteres (gemeinsames) Arbeiten kann bei einem Teammitglied zu Hause oder in anderen Räumlichkeiten der HSR stattfinden.

8.2. Technische Hilfsmittel

Jedes Teammitglied arbeitet mit dem eigenen Notebook. Drucker stehen an der HSR und bei jedem Mitglied zu Hause zur Verfügung.

Die Server für Versionsmanagement und Projektautomation werden von der HSR zur Verfügung gestellt.

8.3. Verwendete Programme + Ausfallszenarios

Programmtyp	Programm	Szenario bei Ausfall	Aufwand bei Ausfall [h]
Betriebssystem	Windows 7 oder Windows Vista	Neuinstallation bzw. Neukauf Laptop / Arbeiten	5

		auf HSR-Computer	
Entwicklungstool	Java Development Kit 1.6 Eclipse 1.3	Neuinstallation	2
Versionsmanagement	Subversion, Repository unter https://svns.hsr.ch/DreiDCOV	Umstellung auf privaten Server	1
Projektautomation	Apache Ant in Eclipse, Buildserver unter http://sinv-19400.hsr.ch	Manuelles Build / Meldung der Dringlichkeit an Server-Verantwortlichen	1
Office-Programme	Microsoft Office 2003 (oder Datei abspeichern als 2003-Version)	Neuinstallation	2
PDF Erstellung	PDF24 Version 3.2.0	Download	1
UML-Modelling Tool	Enterprise Architect	Dringlichkeit an Server-Verantwortlichen	0
	Total Stunden für Risikomanagement (Übertrag in Iterationspläne)		12

9. Qualitätsmassnahmen

9.1. Arbeiten im Team und Dokumentation

Die Arbeit wird mittels Dokumentation festgehalten. Es wird darauf geachtet, dass die Dokumente ständig aktuell gehalten werden.

Am Ende jeder Iteration werden zusätzlich die Dokumente auf ihre Aktualität und Korrektheit von einem Projektmitglied überprüft. Dies wird in einer Tabelle am Anfang eines Dokumentes eingetragen.

Um ein einheitliches Design über alle Dokumente zu erhalten, wird eine Dokumentvorlage benutzt.

Die wöchentlichen Sitzungen werden protokolliert, damit bei Unstimmigkeiten oder Unsicherheiten auf diese Informationen zurückgegriffen werden kann. Während den Projektsitzungen werden die Aufgaben für die nächste Woche festgelegt. Zu Beginn jeder Projektsitzung wird überprüft, ob alle Aufgaben aus dem letzten Sitzungsprotokoll erledigt wurden.

Jedes Projektmitglied führt seine eigene Zeiterfassung, die täglich aktuell gehalten wird. Zusätzlich wird am Ende jeder Woche das Dokument „Zeitplan.xls" aktualisiert.

9.2. Konfigurationsmanagement

Für sämtliche Projektinhalte wird das Versionsmanagement-Tool TortoiseSVN benutzt.

Damit bei Ausfall des Tools kein zu grosser Mehraufwand entsteht, wird von allen Teammitgliedern regelmässig in das SVN Repository eingechecked und aktualisiert. Dadurch sind die 3 Kopien der Projektdaten immer sehr aktuell.

Sobald ein Java-Projekt existiert, wird für die Projektautomation Ant und Jenkins eingesetzt. Es wird jede Minute überprüft, ob sich im Repository etwas geändert hat. Bei Änderungen wird ein neuer Build erstellt.

9.3. Code-Qualität

Um eine gute Code-Qualität zu erreichen, werden sich die Projektmitglieder an die Code-Richtlinien halten. Dazu wird eine von den Projektmitgliedern überarbeitete Version des Dokuments „Code Conventions DE.pdf" von Sebastian Pötzsch [6] benutzt.

Für komplizierte Methoden in der Implementation wird JavaDoc eingesetzt.

Der bereits geschriebene Code wird am Ende jeder Iteration überprüft und wenn nötig umgeschrieben. Insbesondere wird darauf geachtet, dass die Code-Richtlinien eingehalten werden. Diese Reviews werden im Dokument „Code-Reviews.doc" [5] festgehalten.

9.4. Tests

Sämtliche Domain-Klassen werden mittels Unit-Tests überprüft. Diese Tests sind in der Projektautomation eingebunden. Zusätzlich werden die im nächsten Unterkapitel beschriebenen Systemtests durchgeführt.

Zu den Tests werden Protokolle mit den Findings erstellt und im Repository abgelegt („Systemtests.doc [4]"). In der darauf folgenden Besprechung werden die daraus entstehenden Arbeiten den Projektmitgliedern zugeteilt.

9.5. Systemtests

Die Systemtests werden jeweils am Ende der Iterationen durchgeführt:

- Ende Elaboration 2 / Proof of Concept
- Ende Construction 1
- Ende Construction 2

Universität Leipzig
Institut für Informatik
Lehrstuhl für Angewandte Telematik / e-Business
Prof. Dr. Volker Gruhn

Seminararbeit

Java Code Conventions

„Konzepte und Werkzeuge für die Softwareentwicklung mit Java“, SS 2005

Autor:	Sebastian Pötzsch
Matrikelnummer:	8768905
Studiengang:	Informatik (Diplom) 9. Semester
Betreuer:	Ralf Laue
Eingereicht am:	21.07.2005

Inhaltsverzeichnis

1 Einführung

In dieser Ausarbeitung geht es um Code-Konventionen für die Programmiersprache Java. Es soll dabei geklärt werden, was hinter der Grundidee der Code-Konventionen steckt, was ihre Vor- und Nachteile sind, was für Probleme bei ihrer Anwendung auftreten können und wie sie in der realen Welt umgesetzt werden.

In der Softwareentwicklung ist es üblich, dass viele Programmierer gemeinsam an einem Projekt arbeiten, oder in mehreren Projekten involviert sind. Das bedeutet für den einzelnen Programmierer, dass er so programmieren muss, dass sein Code für seine Teammitglieder lesbar und verständlich ist.

Es ist selten, dass ein Programmierer seinen Code während der gesamten Lebenszeit der Software betreut, d.h. der Code sollte so geschrieben sein, dass die Leute, die später die Software instandhalten müssen, sich schnell zurechtfinden. Man muss sich immer vor Augen halten, dass ungefähr 80% der Gesamtkosten von Software auf die Instandhaltung entfallen [sun]. Hier können Code-Konventionen helfen, genau diese Kosten zu senken. Sie beeinflussen, wie einfach es ist, Programmcode zu lesen, zu verstehen und zu überprüfen, selbst Monate nachdem man ihn geschrieben hat. Die Lesbarkeit, das Verständnis, das Abändern und die Fehlersuchen wird damit auch für die Leute erleichtert, die mit dem Code noch später arbeiten müssen. Einfache Code-Konventionen unterstützen also die Wiederverwendung und Wartbarkeit. Sie sind einfache Regeln für Programmierer, die wenn sie befolgt werden die Qualität der Software verbessern.

„In order to write great software, you have to write software greatly.“ [Net]

2 Die Konventionen

Also oberste Regel für Code-Konventionen gilt, dass sie die logische Struktur des Programms verdeutlichen sollen. Es geht hier also nicht darum, dass der Code nett aussieht. Ziel ist es, schon anhand des Layouts schlechten von guten Code zu unterscheiden. Techniken, die guten Code gut aussehen lassen und schlecht schlecht aussehen lassen, sind besser als Techniken, die jeden Code gut aussehen lassen.

In dieser Ausarbeitung werden vor allem die Code-Konventionen vorgestellt, die von Sun vorgeschlagen (siehe: [sun]) werden. Es sei hier jedoch gleich angemerkt, das Code-Konventionen auch größtenteils Geschmackssache sind und von Firma zu Firma, ja sogar von Projekt zu Projekt unterschiedlich gehandhabt werden. Doch spätestens innerhalb eines Projektes sollten sie einheitlich geregelt sein.

Bei Code-Konventionen geht es nicht um Detailfragen, ob man z.B. die öffnende geschwungene Klammer auf die selbe oder auf die nächste Zeile schreibt, sondern mehr um das Erreichen der Hauptziele:

1. Genaue Darstellung der logische Struktur des Codes.

Das erreicht man z.B. durch den Einsatz von Leerzeichen, Leerzeilen und Einzüge.

2. Konsequente Darstellung der logische Struktur.

   Das bedeutet, die Konventionen können bei jeden auftretenden Fall angewendet werden (ohne viele Ausnahmen).

3. Verbesserung der Lesbarkeit des Codes.

   Gute Konventionen erleichtern das Lesen des Codes und werden es niemals verschlechtern. Sie sollten auf keinen Fall guten Code schlechter machen.

4. Resistenz gegenüber Änderungen.

   Das Ändern von einer Zeile Code sollte nicht das Ändern von vielen Zeilen Code auslösen.

Es werden jetzt die einzelnen Richtlinien für die Code-Konventionen in Java vorgestellt. Sie wurden von den Mitgliedern des Java Communicator Teams verabschiedet [sun].

2.1 Dateinamen

2.1.1 Dateiendungen

In Java werden die folgenden Dateiendungen vorgeschrieben.

1. Für Quellcodedateien: *.java
2. Für die kompilierten Bytecodedateien: *.class

2.1.2 Weitere gebräuchliche Dateinamen

1. Für Information über das Programm: README
2. Für das Makefile: Makefile

2.2 Dateiaufbau

Die Quellcodedateien sollten mit einem Kommentar beginnen, der Informationen zum Klassennamen, zur Version, zum Datum und zum Urheberrecht enthält.

```
/*
 * Classname
```

```
 *
 * Version information
 *
 * Date
 *
 * Copyright notice
 */
```

Danach beschreiben die Package-Statements die Paket-Zugehörigkeit und die Import-Statements die importierte Klassen.

```
package java.awt;

import java.awt.peer.CanvasPeer;
```

Diesen folgen die Schnittstellen- und Klassendeklarationen. Die am besten mit einem JavaDoc-Kommentar beginnen, der nähere Informationen zur Klasse enthält (siehe: 2.4.2). Dabei sollte man nie mehr als eine `public` Klasse pro Datei deklarieren. Es sei denn es handelt sich um kleine Helferklassen, dann muss man aber darauf achten, dass in der Datei, die einzelnen Klassen klar getrennt sind. Es folgen das `class`- oder `interface`-Statement, die statischen Klassenvariablen, die Instanzvariablen, alle in der Reihenfolge `public`, `protected` und `privat`. Danach kommt der Konstruktur und die Methoden. Die Methoden sollten nach Funktionalität geordnet werden und nicht nach Zugriff oder gar alphabetisch.

2.3 Einzug

Als Standardeinzug sollten vier Leerzeichen verwendet werden. Man sollte immer Leerzeichen an Stelle des Tabulators verwenden, da dessen Länge auf unterschiedlichen Systemen unterschiedlich interpretiert werden kann. Dies könnte dann zu einer unübersichtlichen Formatierung führen. In dem Artikel „Program Indention and Comprehensibility“ [RJM93] hat man den Zusammenhang zwischen dem benutzten Einzug und der Verständlichkeit eines Programms untersucht. Die Studie ergab, dass 2 bis 4 Leerzeichen als Einzug optimal sind, während 6 Zeichen die Lesbarkeit wieder verschlechtern. Es ist weder hilfreich die Struktur eines Programms unterzubetonen, noch sie überzubetonen.

2.3.1 Zeilenlänge

Die Zeilen sollten nicht länger als 80 Zeichen sein. Hier sind gute Gründe für diese Regel:

- Zeilen länger als 80 Zeichen sind schwieriger zu lesen.

- Eine Begrenzung auf 80 Zeichen lässt Programmierer nochmals über die Sinnhaftigkeit ihrer längeren Zeilen nachdenken.
- Zeilen von mehr als 80 Zeichen lassen sich oft schlecht drucken.
- Die meisten alten Terminals (z.B. bei IBM Großrechnern) sind auf 80 Zeichen begrenzt, was zu unübersichtlichen Darstellungen führt.

2.3.2 Zeilenumbruch

Falls eine Zeile doch länger als 80 Zeichen ist, sollte man sie nach folgenden Regeln umbrechen (laut Sun).

- Umbrüche nach Kommas.
- Umbrüche vor Operatoren.
- Ziehe Umbrüche auf hohen Ebenen, denen auf niedrigen Ebenen vor.

```
// gut
longName1 = longName2 * (longName3 + longName4 - longName5)
          + 4 * longname6;

// schlecht
longName1 = longName2 * (longName3 + longName4
              - longName5) + 4 * longname6;
```

- Die umgebrochen Zeile wird im Einzug mit dem vorhergehenden Ausdruck angepasst. Sollte dies zu unübersichtlichen Code führen, verwendet man acht Leerzeichen.
- Innerhalb `if`-Statements verwendet man acht Leerzeichen, weil sonst leicht Zeilen übersehen werden.

```
// gut
if ((condition1 && condition2)
        || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}

// schlecht
if ((condition1 && condition2)
    || (condition3 && condition4)
    ||!(condition5 && condition6)) { // schlechter Umbruch
    doSomethingAboutIt();            // Zeile leicht überlesbar
}
```

2.4 Kommentare

Kommentare helfen beim Lesen und Verstehen des Codes. Schlecht angeordnete Kommentare können genau das Gegenteil bewirken. In Java gibt es zwei Arten von Kommentaren, Implementierungskommentare und Javadoc-Kommentare. Implementierungskommentare beginne mit `//` oder stehen in `/* ... */`. Sie beschreiben den Code näher und helfen dem Verständnis schwieriger Codeabschnitte. Sie sollten sparsam eingesetzt werden. Javadoc-Kommentare, sind spezielle Kommentare, die in `/** ... */` stehen und durch das `javadoc`-Werkzeug ausgewertet und in HTML-Dateien umgewandelt werden können. Kommentare können nicht geschachtelt werden.

2.4.1 Implementierungskommentare

Es gibt zwei unterschiedliche Arten die Implementierung zu kommentieren: Block-Kommentare und Zeilen-Kommentare.

1. Block-Kommentare: Block-Kommentare stehen in `/* ... */` und gehen über mehrere Zeilen. Sie dienen dazu Dateien, Klassen, Datenstrukturen und Methoden zu beschreiben. Sie sollten so ausgerichtet sein, wie der Code den sie beschreiben. Man kann sie auch verwenden, um ganze Codeabschnitte auszukommentieren.

   ```
   /*
    * Ein Blockkommentar
    */
   ```

2. Zeilen-Kommentare: Sie beginnen mit `//` und enden mit dem Zeilenende. Man kann sie einsetzen, um einzelne Zeilen Code auszukommentieren, oder um einzelne Codezeilen zu beschreiben. Man kann sie über den zu beschreibende Code setzen oder dahinter, wobei man dabei genügend Leerzeichen einfügen sollte, um den Kommentar klar vom Code zu trennen. Vor einer Kommentarzeile wird eine Leerzeile eingefügt.

   ```
   // schlecht
   radius = uselessFunction(sphere);

   // gut
   radius = calculateRadius(sphere);
   ```

2.4.2 Javadoc-Kommentare

Javadoc-Kommentare werden von dem Werkzeug `javadoc` geparst und können z.B. in HTML-Dateien oder PDF-Dateien umgewandelt werden. Sie stehen in

`/** ... */` und beschreiben die nachstehende Klasse, Schnittstelle, Variable oder Methode. `javadoc` verbindet immer automatisch den Javadoc-Kommentar mit der nächsten Deklaration. Durch spezielle Tags werden zum Beispiel Links, Eingabeparameter, Rückgabewerte, Exceptions usw. gekennzeichnet. Genauere Information zum Benutzen von `javadoc` findet man auf der Java-Homepage (http://java.sun.com/javadoc/).

```
/**
 * Ein javadoc Kommentar
 *
 * @param       a some integer
 * @param       b another integer
 * @return      guess what an integer
 * @author      Java Coder
 * @version     1.0, June 2005
 */
int foo( int a, int b ) {
 ...
```

2.5 Deklarationen

Es wird empfohlen eine Zeile pro Deklaration zu verwenden. Auch das Mischen von Datentypen ist zu vermeiden.

```
int sum;
int number; // gut

int sum, number; // schlecht

int sum, number[]; // schlecht
```

Variablendeklarationen lassen sich dadurch leichter auskommentieren. Mehr als eine Deklarationen ein einer Zeile kann zu Missverständnissen führen und die Fehlersuche erheblich erschweren. Hier mal ein C-Beispiel.

```
double* x, y; // C Beispiel
```

2.5.1 Initialisierung

Variablen sollten möglichst dort initialisiert werden, wo sie deklariert werden. Der einzige Grund, warum eine Variable erst deklariert wird und dann initialisiert, ist wenn ihr Wert von einer Berechnung abhängt.

2.5.2 Platzierung

Die Deklarationen sollten immer am Anfang eines Blockes stehen. Man wartet sollte nicht mit der Deklaration warten bis ma ndie Variable auch braucht. Das erschwert bloß das Lesen des Codes.

```
void myMethod() {
    int int1 = 0;         // Anfang des Blocks

    if (condition) {
        int int2 = 0;     // Anfang des If-Blocks
        ...
    }
}
```

Eine Ausnahme dieser Regel sind die Laufvariablen in `for`-Schleifen.

```
for (int i = 0; i < maxLoops; i++) {
    ...
}
```

Man sollte Deklarationen vermeiden, die Variablen auf höheren Ebenen überdecken.

```
int sum;

method() {
    if (condition) {
        int sum = 0; // schlecht
        ...
    }
    ...
}
```

2.5.3 Klassen und Schnittstellen

Bei Schnittstellen- und Klassendeklarationen gelten folgende Regeln.

- Kein Leerzeichen zwischen dem Methodennamen und der aufgehenden Klammer.
- Öffnende geschwungene Klammern sollten in der selben Zeile stehen.
- Schließende geschwungene Klammern sollten in einer eigenen Zeile stehen, außer bei leeren Statements, die sofort geschlossen werden.

```
class Sample extends Object {
    int ivar1;
    int ivar2;

    Sample(int i, int j) {
        ivar1 = i;
        ivar2 = j;
    }

    int emptyMethod() {}
    ...
}
```

- Methoden werden mit einer Leerzeile voneinander getrennt.

2.6 Statements

2.6.1 Einfache Statements

Man sollte immer eine Zeile pro Statement verwenden.

```
i = 0; j = 0; k = 0; DestroyBadLoopNames( i, j, k );  // schlecht
```

Hier ein paar gute Gründe für diese Regel:

- Das Verwenden von einer Zeile pro Statement veranschaulicht die Komplexität eines Programms besser.
- Mehrere Statements pro Zeile geben modernen Compilern keine Hinweise mehr für die Optimierung.
- Der Code liest sich leichter, wenn man einfach nur von oben nach unten liest, anstatt noch zusätzlich von rechts nach links.
- Es ist einfacher mit einem Debugger, den Code Schritt für Schritt durch zugehen und den Fehler zu lokalisieren.
- Es erleichtert das Ändern oder Auskommentieren der einzelnen Statements.
- Mehrere Operationen pro Zeile sind oft schwieriger zu verstehen und können zu Seiteneffekten führen.

  ```
  while ( *++t = *++s ) // C Beispiel
  ```

2.6.2 Blöcke

Die eingeschlossen Statements sollten ein Level mehr eingezogen sein als das umschließende Statement. Die öffnende Klammer sollte in der selben Zeile stehen wie das umschließende Statement (z.B. `for` oder `while`). Klammer werden auch um einzelne Statements gesetzt, wenn sie teil einer Kontrollstruktur wie `if-else` oder `for` sind. Das erleichtert es später weitere Zeilen einzufügen.

2.6.3 Return-Statement

Das `Return`-Statement sollte nicht in Klammern gesetzt werden, es sei denn dadurch wird der Rückgabewert verständlicher.

```
  return (size ? size : defaultSize); // Beipiel
```

2.6.4 If-Statement

`If`-Statements sollten folgende Form haben.

```
if (condition) {
    statements;
}

if (condition) {
    statements;
} else {
    statements;
}

if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```

2.6.5 For-Statement

`For`-Statements werden wie folgt geschrieben.

```
for (initialization; condition; update) {
    statements;
}
```

Bei leeren `For`-Statement werden die geschwungene Klammern weggelassen.

```
for (initialization; condition; update);
```

2.6.6 While-Statement

`While`-Statements haben folgende Form.

```
while (condition) {
    statements;
}
```

Ein leeres `While`-Statement wird ohne geschwungene Klammern geschrieben.

```
while (condition);
```

2.6.7 Do-While-Statement

`Do-While`-Statements werden wie folgt angelegt.

```
do {
    statements;
} while (condition);
```

2.6.8 Switch-Statement

Ein `switch`-Statement sollte folgende Form haben.

```
switch (condition) {
case ABC:
    statements;
    /* fällt durch */

case DEF:
    statements;
    break;
```

```
case XYZ:
    statements;
    break;

default:
    statements;
    break;
}
```

Jedes `switch`-Statement sollte ein `default`-Fall haben. Wenn ein „fall trough“ in einem Fall gewollt ist, sollte man das durch einen Kommentar verdeutlichen.

2.6.9 Try-Catch-Statement

Das Format für ein `try-catch`-Statement ist wie folgendes festgelegt.

```
try {
    statements;
} catch (ExceptionClass e) {
    statements;
} finally {
    statements;
}
```

2.7 Leerraum

Die Notwendigkeit von Leerraum sollen folgende zwei Beispiele demonstrieren. Das Codebeipiel in Abbildung 1 aus [McC02] soll verdeutlichen, wie der sparsame Einsatz von Leerzeichen das Lesen von Code erschweren kann. Dieser Code ist syn-

Abbildung 1: Schlechtes Layout

```
/* Use the insertion sort technique to sort the "data" array in
ascending order. This routine assumes that data[ firstElement ]
is not the first element in data and that data[ firstElement-1 ]
can be accessed. */ public void InsertionSort( int[] data, int
firstElement, int lastElement ) { /* Replace element at lower
boundary with an element guaranteed to be first in a sorted list.
*/ int lowerBoundary = data[ firstElement-1 ]; data[ firstElement-1 ]
 = SORT_MIN; /* The elements in positions firstElement through
 sortBoundary-1 are always sorted. In each pass through the loop,
 sortBoundary is increased, and the element at the position of
 the new sortBoundary probably isn't in its sorted place in the
 array, so it's inserted into the proper place somewhere between
firstElement and sortBoundary. */ for ( int sortBoundary =
firstElement+1; sortBoundary <= lastElement; sortBoundary++  )
{ int insertVal = data[ sortBoundary ]; int insertPos = sortBoundary;
while ( insertVal < data[ insertPos-1 ] ) { data[ insertPos ] =
data[ insertPos-1 ]; insertPos = insertPos-1; } data[ insertPos ]
= insertVal; } /* Replace original lower-boundary element */
data[ firstElement-1 ] = lowerBoundary; }
```

taktisch korrekt, ist gut kommentiert und hat gut gewählte Variabelnamen, dennoch hat er offensichtliche Nachteile gegenüber Codebeispiel in Abbildung 2.

2.7.1 Leerzeilen

Durch Leerzeilen wird die Lesbarkeit des Codes verbessert. Sie strukturieren die einzelnen Zeilen in logischen Codefragmente.

- Zwei Leezeilen sollte man zwischen Klassen- und Schnittstellen-Definitionen und zwischen einzelnen Codeabschnitten verwenden.
- Eine Leezeile sollte vor jeder Methode, vor jeden Block- oder Einzeilenkommentar (siehe: 2.4), zwischen logischen Abschnitten innerhalb einer Methode und zwischen den Variablendeklarationen und dem ersten Statement stehen.

2.7.2 Leerzeichen

Leerzeichen sollten nach jedem Schlüsselwort stehen (z.B. `while`, `if`, usw.), aber nie zwischen dem Methodennamen und der öffnenden Klammer. Außerdem sollten Leerzeichen nach Kommas in Parameterlisten, vor und nach binären Operatoren stehen und nach Cast-Konstrukten. Die Ausdrücke in `for`-Statements sollten durch Leerzeichen getrennt sein.

Abbildung 2: Gutes Layout

```
/* Use the insertion sort technique to sort the "data" array in ascending
order. This routine assumes that data[ firstElement ] is not the
first element in data and that data[ firstElement-1 ] can be accessed.*/
public void InsertionSort( int[] data, int firstElement, int lastElement ) {

    // Replace element at lower boundary with an element guaranteed to be
    // first in a sorted list.
    int lowerBoundary = data[ firstElement - 1 ];
    data[ firstElement - 1 ] = SORT_MIN;

    /* The elements in positions firstElement through sortBoundary-1 are
    always sorted. In each pass through the loop, sortBoundary
    is increased, and the element at the position of the
    new sortBoundary probably isn't in its sorted place in the
    array, so it's inserted into the proper place somewhere
    between firstElement and sortBoundary.
    */
    for ( int sortBoundary = firstElement + 1; sortBoundary <= lastElement;
        sortBoundary++ ) {
        int insertVal = data[ sortBoundary ];
        int insertPos = sortBoundary;

        while ( insertVal < data[ insertPos - 1 ] ) {
            data[ insertPos ] = data[ insertPos - 1 ];
            insertPos = insertPos - 1;
        }
        data[ insertPos ] = insertVal;
    }

    // Replace original lower-boundary element
    data[ firstElement - 1 ] = lowerBoundary;
}
```

2.8 Namenskonventionen

1. Pakete werden in Kleinbuchstaben geschrieben und sollten mit den typischen Domainnamen: com, edu, gov, mil, net, org, de, usw. beginnen, so wie sie im ISO Standard 3166, 1981 beschrieben sind. Außerdem sollte die Paket-Namen der inneren Organisation entsprechen.

   ```
   package com.sun.corba.se.extension;
   ```

2. Klassennamen sollten Substantive sein, die in der Einzahl stehen, wobei der Anfangsbuchstabe jedes inneren Wortes groß geschrieben sein sollte. Klassennamen sollten kurz und prägnant gewählt werden. Vermeiden sie, soweit möglich, Abkürzungen und Akronyme. Bei gebräuchliche Akronymen wird nur der erste Buchstaben groß geschrieben.

   ```
   class HtmlViewer;
   class XmlLoader;
   ```

3. Schnittstellen werden wie Klassen benannt.

   ```
   interface Loader;
   ```

4. Methodennamen sollten Verben sein, bei denen der ersten Buchstabe klein und jedes interne Wort groß geschrieben wird.

   ```
   getItem();
   isReady();
   computeStandardDeviation();
   ```

5. In Methodennamen sollte der Objektname vermieden werden.

   ```
   employee.getName(); // nicht employee.getEmployeeName()
   car.getColor(); // nicht car.getCarColor()
   ```

6. Beim Zugriff auf Klassenattribute sollte man die Präfixe *get* und *set* verwenden.

   ```
   line.getLength();
   line.setLength(10);
   ```

7. Beim Abfragen von boolschen Variablen sollte man den Präfix *is* verwenden.

   ```
   boolean isFinished();
   ```

8. Bei Funktionen, die etwas berechnen hat sich der Präfix *compute* durchgesetzt.

   ```
   data.computeMean();
   ```

9. Variablennamen beginnen klein und jedes interne Wort wird groß geschrieben. Die Namen sollten kurz und prägnant sein. Sie sollten selbsterklärend und besonders einprägsam gewählt werden. Einzelne Buchstaben als Variablen sollten vermieden werden, außer für temporäre Variablen, wie z.B. Indexvariablen.

   ```
   int i;
   float tableHeight;
   ```

10. Generische Variablen sollten den Namen ihres Typs tragen.

    ```
    // gut
    void connect(Database database);

    // schlecht
    void connect(Database db)
    ```

11. Bei Variablen, die eine Rolle haben, lassen sich oft ihre Rolle und ihr Typ vereinen.

    ```
    Name loginName;
    Matrix rotationMatrix;
    ```

12. Konstanten werden komplett groß geschrieben. Einzelne Wörter werden mit „_“ abgetrennt.

    ```
    static final int MAX_ROOMS = 99;
    static final int COLOR_RED = 2;
    ```

2.9 Programmiertechniken

2.9.1 Zugriff

Das Information Hiding Principle sollte eingehalten werden. Man sollte Klassenvariablen nicht unnötig als `public` deklarieren. Stattdessen verwendet man `get-` und `set`-Methoden. Ein guter Grund Klassenvariablen dennoch als `public` zu deklarieren, ist wenn ihre Klasse mehr einer Datenstruktur ähnelt.

2.9.2 Referenzen

Bei statischen Methoden und Klassenvariablen sollte man nie ein Objekt anlegen.

```
classMethod();              // gut
AClass.classMethod();       // gut
anObject.classMethod();     // schlecht
```

2.9.3 Konstanten

Konstanten sollten nie direkt verwendet werden. Man sollte nie magische Zahlen, die ohne Zusammenhang einfach im Code auftreten, verwenden. Ausnahme bilden hier die Zahlen 1, -1, und 0, die z.B. in `for`->Schleifen als Zählerwerte verwendet werden.

```
private static final int  TEAM_SIZE = 11;
Player[] players = new Player[TEAM_SIZE];
// NOT: Player[] players = new Player[11];
```

2.9.4 Klammern

Man verwendet Klammern in logischen Ausrücken, auch wenn die Operatorpriorität eindeutig ist. Es erhöht die Lesbarkeit und hilft unerfahrenen Programmierern.

```
if (a == b && c == d)     // schlecht
if ((a == b) && (c == d)) // gut
```

3 Zusammenfassung

In der Praxis sieht es oft so aus, dass jede Firma ihre eigene Code-Konventionen hat. Code-Konventionen können sich auch von Projekt zu Projekt unterscheiden. Letztendlich sind viele Detailfragen eher Geschmackssache oder teil persönlicher Erfahrungen. Es gibt oft unterschiedliche Meinungen zu einzelnen Detailfragen und manche Entwickler vertreten sehr stur ihre Paradigmen. Es lohnt sich jedoch nicht sich bei solchen Details in „Religions-Kriege“ oder in philosophische Diskussionen verwickeln zu lassen. Wahre Experten können den Code auch verstehen, wenn er sich von ihren Layout-Vorstellungen unterscheidet. Es handelt sich hierbei nicht nur um objektive, sondern auch um ästhetische Anschauungen. Wichtig ist das man sich in seinem Team oder in der Firma auf einen Standard einigt.

Code-Konventionen sind auf keinen Fall unumstößliche Gesetze. So hat z.B. Microsoft jahrelang für C++ die ungarische Notation für die Variablen- und Methodenbenennung propagiert und sich kürzlich entschlossen sich davon wieder zu distanzieren.

Für einzelne Regelungen gibt es immer auch Vor- und Nachteile, die man der situationsabhängig beurteilen sollte, z.B. kann es von Vorteil sein die Zeilenlängenbegrenzung auf 80 Zeichen zu ignorieren, da dadurch vielleicht die Anweisung wieder leichter zu lesen ist.

Gute Programmierer sind aufgeschlossen gegenüber neuen Konventionen, die nachgewiesenermaßen besser sind als ihre bis dahin verwendeten Praktiken, selbst

wenn es die Anpassung mit einigen Unannehmlichkeiten verbunden ist. Es gibt eindeutig einige Konventionen die Vorteile haben gegenüber anderen haben und genau auf die sollte man sich konzentrieren. Man sollte nicht die Grundidee von Code-Konventionen vergessen, nämlich die Lesbarkeit, das Verständnis, das Überarbeiten und die Wartbarkeit von Code zu erleichtern.

A Anhang

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[McC02] McConnell, Steve: *Code Complete*. 2nd. Microsoft Press, September 2002. – ISBN 1556154844

[Net] Netscape: *Netscape's Software Coding Standards Guide For Java*. http://www.csa.iisc.ernet.in/old-website/Documentation/Tutorials/StyleGuides/netscape-codestyle.html

[RJM93] Richard J. Miara, Juan A. N.: Program indentation and comprehensibility. In: *Communications of the ACM* (1993)

[sun] *Code Conventions for the Java Programming Language*. http://java.sun.com/docs/codeconv/

Detailplan **J3DEval**

Phasen					Inception 1				Elaboration 1 / Grobevaluation			
Woche (von - bis)					SW 1				SW 2			
					19.09		23.09		26.09		30.09	
		proz.	Total		wals		mueh		wals		mueh	
Arbeitspaket	Geschätzt	verplant	SOLL	IST	S	I	S	I	S	I	S	I
Projekt Management	**89.8**	**59%**	**53.0**	**46.5**	13.5	9.5	11.0	12.0	3.0	3.5	4.0	5.5
Projektplan	22	93%	20.5	21.5	8.0	7.5	9.0	9.0	2.0	2.0	1.0	2.0
Zeitplan	7	86%	6.0	6.5	3.5	2.0						
Risikomanagement	3	100%	3.0	0.5	1.0							
Dokumentvorlagen	2	150%	3.0	5.5			2.0	3.0		1.0		
Codestyleguide	1	100%	1.0	0.0	1.0							
Besprechungen + Protokoll schreiben	28	54%	15.0	6.0					1.0	0.5	1.0	0.5
Iterationspläne	5	90%	4.5	6.5							2.0	3.0
Reserven aus Riskmanagement	21.8	0%	0.0	0.0								
Evaluation	**105.0**	**90%**	**94.0**	**57.0**	1.0	3.0	1.0	1.0	10.0	7.5	6.0	3.0
Evaluationskriterien erstellen	5	20%	1.0	1.5								
Lösungsvarianten suchen	15	60%	9.0	2.5								
Evaluation durchführen	30	100%	30.0	12.5					4.0	3.0		
Evaluationsdokumentation	25	96%	24.0	3.5								
Einarbeitung in 3DCOV-Code und 3D	24	100%	24.0	32.0	1.0	3.0	1.0	1.0	6.0	4.5	6.0	3.0
Beurteilung + Entscheid Lösungsvarianten	6	100%	6.0	5.0								
Qualitätsmassnahmen	**81.0**	**98%**	**79.5**	**95.0**	1.5	2.0	2.0	2.5	3.0	4.0	2.5	3.5
Reviews mit Betreuer + Protokoll schreiben	49	86%	42.0	36.0	1.5	2.0	2.0	2.5	2.0	2.5	1.5	2.0
Review Code	7	129%	9.0	14.0								
Refactoring	16	69%	11.0	12.5								
Dokumente überprüfen	9	194%	17.5	32.5					1.0	1.5	1.0	1.5
Requirements	**15.0**	**147%**	**22.0**	**28.0**	0.0	0.0	0.0	0.0	0.0	0.0	3.5	4.0
Anforderungsspezifikation	13	150%	19.5	27.5							3.0	3.5
Glossary	2	125%	2.5	0.5							0.5	0.5
Business Modeling	**11.0**	**182%**	**20.0**	**10.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Domain Model	4	125%	5.0	3.0								
System Sequenz Diagramm	4	275%	11.0	6.0								
Operation Contracts	3	133%	4.0	1.0								
Analyse und Design	**22.0**	**127%**	**28.0**	**17.5**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Sequenzdiagramm	6	67%	4.0	0.0								
Klassendiagramme	6	117%	7.0	6.5								
System Architecture Document	10	170%	17.0	11.0								
Implementation	**107.0**	**122%**	**130.5**	**193.5**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Funktionsumfang Releases	7	100%	7.0	3.0								
Release 1	25	120%	30.0	67.0								
Release 2	40	105%	42.0	79.5								
Release 3	35	147%	51.5	44.0								
Konfigurationsmanagement	**2.0**	**50%**	**1.0**	**3.5**	0.0	0.0	1.0	0.5	0.0	0.0	0.0	0.0
Version Control System	1	100%	1.0	0.5			1.0	0.5				
Projektautomation	1	0%	0.0	3.0								
Tests	**35.0**	**97%**	**34.0**	**33.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Systemtests (Protokoll erstellen)	9	89%	8.0	9.5								
Unit Tests	13	131%	17.0	9.0								
Bugfixing	13	69%	9.0	14.5								
Dokumentation und Deployment	**14.0**	**100%**	**14.0**	**25.5**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Schlussberichte erstellen	14	100%	14.0	25.5								
Wochen Subtotal					**16.0**	**14.5**	**15.0**	**16.0**	**16.0**	**15.0**	**16.0**	**16.0**
Wochentotal	**481.8**	**99%**	**476.0**	**509.5**			**31.0**	**30.5**			**32.0**	**31.0**
Kontrolle	**481.8**		**476.0**	**509.5**								
	0.0											

Phasen	Elaboration 1 / Grobevaluation								Elaboration 2 / Feinevaluation			
Woche (von - bis)	SW 3				SW 4				SW 5			
	03.10		07.10		10.10		14.10		17.10		21.10	
	wals		mueh		wals		mueh		wals		mueh	
Arbeitspaket	S	I	S	I	S	I	S	I	S	I	S	I
Projekt Management	1.0	2.0	1.0	1.0	3.5	1.0	1.5	2.0	0.5	0.5	0.5	0.5
Projektplan		0.5			0.5	0.5						
Zeitplan		0.5		0.5	1.0			0.5				
Risikomanagement							0.5	0.5				
Dokumentvorlagen												
Codestyleguide												
Besprechungen + Protokoll schreiben	1.0	0.5	1.0	0.5	1.0	0.5	1.0	0.5	0.5	0.5	0.5	0.5
Iterationspläne		0.5			1.0			0.5				
Reserven aus Riskmanagement												
Evaluation	10.5	13.0	11.0	5.5	9.0	7.5	10.0	6.0	8.5	0.5	8.0	2.0
Evaluationskriterien erstellen	0.5	0.5	0.5	1.0								
Lösungsvarianten suchen	1.0		1.5	1.5		1.0			3.5		3.0	
Evaluation durchführen	4.0	4.5	4.0		6.0	2.0	6.0	3.0	1.0		1.0	
Evaluationsdokumentation	2.0	1.0	2.0		2.0	2.5	3.0		2.0		2.0	
Einarbeitung in 3DCOV-Code und 3D	3.0	7.0	3.0	3.0		2.0		3.0	2.0	0.5	2.0	2.0
Beurteilung + Entscheid Lösungsvarianten					1.0		1.0					
Qualitätsmassnahmen	1.5	0.0	2.0	0.5	3.0	2.0	2.0	3.0	2.0	1.5	2.5	2.0
Reviews mit Betreuer + Protokoll schreiben	1.5		2.0		2.0	1.0	1.5	1.5	1.5	1.5	2.0	1.5
Review Code												
Refactoring												
Dokumente überprüfen				0.5	1.0	1.0	0.5	1.5	0.5		0.5	0.5
Requirements	2.0	0.0	2.0	7.0	1.0	0.0	2.5	0.5	0.0	0.0	0.0	0.0
Anforderungsspezifikation	2.0		2.0	7.0	1.0		2.0	0.5				
Glossary							0.5					
Business Modeling	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0
Domain Model									0.5		0.5	
System Sequenz Diagramm									0.5		0.5	
Operation Contracts												
Analyse und Design	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Sequenzdiagramm												
Klassendiagramme												
System Architecture Document												
Implementation	2.5	2.0	1.5	3.5	1.0	6.5	1.5	8.0	5.0	16.5	5.0	10.5
Funktionsumfang Releases	0.5		0.5			1.5	0.5					
Release 1	2.0	2.0	1.0	3.5	1.0	5.0	1.0	8.0	5.0	16.5	5.0	10.5
Release 2												
Release 3												
Konfigurationsmanagement	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Version Control System												
Projektautomation												
Tests	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Systemtests (Protokoll erstellen)												
Unit Tests												
Bugfixing												
Dokumentation und Deployment	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Schlussberichte erstellen												
Wochen Subtotal	**17.5**	**17.0**	**17.5**	**17.5**	**17.5**	**17.0**	**17.5**	**19.5**	**17.0**	**19.0**	**17.0**	**15.0**
Wochentotal			**35.0**	**34.5**			**35.0**	**36.5**			**34.0**	**34.0**

Phasen	Elaboration 2 / Feinevaluation				02.11.2011 1. Release						
Woche (von - bis)	SW 6				SW 7				SW 8		
	24.10		28.10		31.10		04.11		07.11		
	wals		mueh		wals		mueh		wals		ue
Arbeitspaket	S	I	S	I	S	I	S	I	S	I	S
Projekt Management	0.5	0.5	0.5	0.5	2.0	3.0	0.5	0.0	1.5	0.0	
Projektplan											
Zeitplan					0.5	0.5					
Risikomanagement					0.5						
Dokumentvorlagen						1.5			1.0		
Codestyleguide											
Besprechungen + Protokoll schreiben	0.5	0.5	0.5	0.5	0.5	0.5	0.5		0.5		
Iterationspläne					0.5	0.5					
Reserven aus Riskmanagement											
Evaluation	5.0	5.0	4.0	0.0	5.0	1.0	5.0	0.0	0.0	0.0	
Evaluationskriterien erstellen											
Lösungsvarianten suchen											
Evaluation durchführen	2.0		2.0								
Evaluationsdokumentation	3.0		2.0		3.0		3.0				
Einarbeitung in 3DCOV-Code und 3D						1.0					
Beurteilung + Entscheid Lösungsvarianten		5.0			2.0		2.0				
Qualitätsmassnahmen	2.0	1.5	1.5	1.5	3.5	10.5	4.0	11.5	1.0	2.5	
Reviews mit Betreuer + Protokoll schreiben	2.0	1.5	1.5	1.0	1.5	1.0	2.0	1.0	1.0	1.0	
Review Code					0.5	4.0	0.5	4.0			
Refactoring				0.5	1.0	3.0	1.0	4.0			
Dokumente überprüfen					0.5	2.5	0.5	2.5		1.5	
Requirements	1.0	0.0	1.0	2.0	0.0	2.0	0.0	2.0	2.0	0.0	
Anforderungsspezifikation	0.5		1.0	2.0		2.0		2.0	2.0		
Glossary	0.5										
Business Modeling	3.0	0.0	3.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	
Domain Model	1.0		1.0	3.0							
System Sequenz Diagramm	1.0		1.0	1.0							
Operation Contracts	1.0		1.0	1.0							
Analyse und Design	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	
Sequenzdiagramm											
Klassendiagramme											
System Architecture Document									2.0		
Implementation	5.0	11.0	6.0	9.0	7.0	1.0	7.0	2.5	10.0	19.5	
Funktionsumfang Releases	2.0		2.0								
Release 1	3.0	11.0	4.0	9.0	4.0		4.0	1.5			
Release 2					3.0	1.0	3.0	1.0	10.0	19.5	
Release 3											
Konfigurationsmanagement	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
Version Control System											
Projektautomation		1.0									
Tests	1.0	2.0	1.0	0.0	0.0	1.5	1.0	1.5	1.0	0.0	
Systemtests (Protokoll erstellen)			1.0			1.5		1.5			
Unit Tests	1.0	2.0							1.0		
Bugfixing							1.0				
Dokumentation und Deployment	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
Schlussberichte erstellen											
Wochen Subtotal	**17.5**	**21.0**	**17.0**	**18.0**	**17.5**	**19.0**	**17.5**	**17.5**	**17.5**	**22.0**	
Wochentotal			**34.5**	**39.0**			**35.0**	**36.5**			

Auswertung:

	Gesamt		wals		mueh		Diffenenz (Ist-Soll)			Diffenenz (Akkumuliert)		
	SOLL	IST	SOLL	IST	SOLL	IST	wals	mueh	Total	wals	mueh	Total
Inception 1	31.0	30.5	16.0	14.5	15.0	16.0	-1.5	1.0	-0.5	-1.5	1.0	-0.5
Elaboration 1 / Grobevaluation	102.0	102.0	51.0	49.0	51.0	53.0	-2.0	2.0	0.0	-3.5	3.0	-0.5
Elaboration 2 / Feinevaluation	103.5	109.5	52.0	59.0	51.5	50.5	7.0	-1.0	6.0	3.5	2.0	5.5
Construction 1	108.0	117.0	53.5	61.0	54.5	56.0	7.5	1.5	9.0	11.0	3.5	14.5
Construction 2	69.5	72.0	35.0	37.0	34.5	35.0	2.0	0.5	2.5	13.0	4.0	17.0
Transition 1	62.0	78.5	31.0	35.5	31.0	43.0	4.5	12.0	16.5	17.5	16.0	33.5
Total	**476.0**	**509.5**	**238.5**	**256.0**	**237.5**	**253.5**	**17.5**	**16.0**	**33.5**	**17.5**	**16.0**	**33.5**
Kontrolle	0.0	0.0										

				SOLL	IST	Plan
Total SOLL	**476.0**		**Total** wals	238.5	256.0	240
Total IST	**509.5**		**Total** mueh	237.5	253.5	240
Diff	**-33.5**		**Total**	**476.0**	**509.5**	**480**

Detailplan

J3DEval

Phasen

					Construction 1													Construction 2											Transition 1						
													30.11.2011: 2. Release																		23.12.2011: 3. Release			
					SW 9		SW 10						SW 11					SW 12					SW 13					SW 14						
Woche (von - bis)					18.11		21.11		25.11				28.11		02.12			05.12		09.12			12.12		16.12			19.12		23.12				
					mueh		wals		mueh				wals		mueh			wals		mueh			wals		mueh			wals		mueh				
Arbeitspaket	**Geschätzt**	**proz. verplant**	**Total SOLL**	**IST**	**S**	**I**	**S**	**I**	**S**	**I**			**S**	**I**	**S**	**I**		**S**	**I**	**S**	**I**		**S**	**I**	**S**	**I**		**S**	**I**	**S**	**I**			
Projekt Management	**89.8**	**59%**	**53.0**	**46.5**	0.5	0.0	0.5	0.0	0.5	0.0			0.5	0.0	2.0	0.0		0.5	1.5	0.5	0.5		0.5	1.0	2.0	1.0		0.0	1.0	0.0	0.0			
Projektplan	22	93%	20.5	21.5																													
Zeitplan	7	86%	6.0	6.5											0.5				0.5					1.0	0.5				1.0				
Risikomanagement	3	100%	3.0	0.5											0.5										0.5								
Dokumentvorlagen	2	150%	3.0	5.5																													
Codestyleguide	1	100%	1.0	0.0																													
Besprechungen + Protokoll schreiben	28	54%	15.0	6.0	0.5		0.5		0.5				0.5		0.5			0.5		0.5			0.5		0.5	0.5							
Iterationspläne	5	90%	4.5	6.5											0.5				1.0		0.5				0.5	0.5							
Reserven aus Riskmanagement	21.8	0%	0.0	0.0																													
Evaluation	**105.0**	**90%**	**94.0**	**57.0**	0.0	0.0	0.0	0.0	0.0	0.0			0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0			
Evaluationskriterien erstellen	5	20%	1.0	1.5																													
Lösungsvarianten suchen	15	60%	9.0	2.5																													
Evaluation durchführen	30	100%	30.0	12.5																													
Evaluationsdokumentation	25	96%	24.0	3.5																													
Einarbeitung in 3DCOV-Code und 3D	24	100%	24.0	32.0																													
Beurteilung + Entscheid Lösungsvarianten	6	100%	6.0	5.0																													
Qualitätsmassnahmen	**81.0**	**98%**	**79.5**	**95.0**	1.0	1.0	8.0	2.5	8.5	3.5			2.5	3.5	2.0	8.5		1.0	2.0	1.5	1.5		5.5	4.0	5.0	8.0		3.0	2.0	3.5	5.0			
Reviews mit Betreuer + Protokoll schreiben	49	86%	42.0	36.0	1.0	1.0	1.0	1.0	1.5	1.5			1.5	2.5	1.0	2.0		1.0	1.0	1.5	1.5		1.5	0.5	1.0	0.5		1.0	1.0	1.5	1.0			
Review Code	7	129%	9.0	14.0			2.0	1.5	2.0	2.0													2.0	0.5	2.0	2.0							
Refactoring	16	69%	11.0	12.5			3.0		3.0														1.0	1.0	1.0	2.5							
Dokumente überprüfen	9	194%	17.5	32.5			2.0		2.0				1.0	1.0	1.0	6.5			1.0				1.0	2.0	1.0	3.0		2.0	1.0	2.0	4.0			
Requirements	**15.0**	**147%**	**22.0**	**28.0**	1.0	0.0	0.5	1.0	0.0	0.0			0.0	0.0	0.0	0.0		1.0	0.0	1.0	0.0		0.0	3.5	0.0	0.0		0.5	1.0	0.0	0.0			
Anforderungsspezifikation	13	150%	19.5	27.5	1.0			1.0										1.0		1.0				3.5					1.0				
Glossary	2	125%	2.5	0.5			0.5																					0.5						
Business Modeling	**11.0**	**182%**	**20.0**	**10.0**	4.0	0.0	1.0	2.0	2.0	0.0			0.0	1.0	0.0	1.5		1.0	0.0	1.0	0.5		0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0			
Domain Model	4	125%	5.0	3.0	2.0																												
System Sequenz Diagramm	4	275%	11.0	6.0	2.0			2.0	1.0					1.0		1.5		1.0		1.0	0.5													
Operation Contracts	3	133%	4.0	1.0			1.0		1.0																								
Analyse und Design	**22.0**	**127%**	**28.0**	**17.5**	6.0	0.0	3.0	4.0	1.0	4.0			1.0	2.0	1.0	0.0		1.5	0.0	1.5	0.0		2.0	5.0	1.0	1.0		0.0	0.0	0.0	0.0			
Sequenzdiagramm	6	67%	4.0	0.0	2.0																												
Klassendiagramme	6	117%	7.0	6.5	2.0		2.0	4.0															1.0	2.5									
System Architecture Document	10	170%	17.0	11.0	2.0		1.0		1.0	4.0			1.0	2.0	1.0			1.5		1.5			1.0	2.5	1.0	1.0							
Implementation	**107.0**	**122%**	**130.5**	**193.5**	5.0	19.5	3.0	5.0	4.0	9.0			9.0	12.0	9.5	7.0		10.0	9.0	9.5	15.0		7.0	0.5	7.0	1.5		0.0	0.0	0.0	0.0			
Funktionsumfang Releases	7	100%	7.0	3.0														0.5			0.5			0.5								
Release 1	25	120%	30.0	67.0																													
Release 2	40	105%	42.0	79.5	5.0	19.5	3.0	5.0	4.0	9.0																							
Release 3	35	147%	51.5	44.0									9.0	12.0	9.5	7.0		9.5	9.0	9.5	14.5		7.0		7.0	1.5							
Konfigurationsmanagement	**2.0**	**50%**	**1.0**	**3.5**	0.0	0.0	0.0	0.0	0.0	0.0			0.0	2.0	0.0	0.0		0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0			
Version Control System	1	100%	1.0	0.5																													
Projektautomation	1	0%	0.0	3.0										2.0																			
Tests	**35.0**	**97%**	**34.0**	**33.0**	1.0	0.0	2.0	3.0	2.0	2.0			4.0	0.0	2.0	0.5		3.0	4.0	3.0	0.0		3.5	2.5	3.5	7.0		2.0	4.0	2.0	5.0			
Systemtests (Protokoll erstellen)	9	89%	8.0	9.5				1.0					2.0		2.0								0.5	2.0	0.5	0.5		1.0	1.0	1.0	2.0			
Unit Tests	13	131%	17.0	9.0	1.0		2.0		2.0	1.0								2.0		2.0			1.0		1.0	1.0		1.0	2.0	1.0	3.0			
Bugfixing	13	69%	9.0	14.5				2.0		1.0			2.0			0.5		1.0	4.0	1.0			2.0	0.5	2.0	5.5			1.0				
Dokumentation und Deployment	**14.0**	**100%**	**14.0**	**25.5**	0.0	0.0	0.0	0.0	0.0	0.0			0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0		1.0	5.0	1.0	7.5		6.0	6.0	6.0	7.0			
Schlussberichte erstellen	14	100%	14.0	25.5																			1.0	5.0	1.0	7.5		6.0	6.0	6.0	7.0			
Wochen Subtotal					18.5	20.5	18.0	17.5	18.0	18.5			17.0	20.5	16.5	17.5		18.0	16.5	18.0	17.5		19.5	21.5	19.5	26.0		11.5	14.0	11.5	17.0			
Wochentotal	**481.8**	**99%**	**476.0**	**509.5**	36.5	42.0			36.0	36.0					33.5	38.0				36.0	34.0				39.0	47.5				23.0	31.0			
Kontrolle	**481.8**		**476.0**	**509.5**																													
	0.0																																

Auswertung:

Inception 1
Elaboration 1 / Grobevaluation
Elaboration 2 / Feinevaluation
Construction 1
Construction 2
Transition 1

Total
Kontrolle

Projekt: J3DEval

Zeitplan und Auswertungen

Version: 1.0
Datum: 10. November 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
21.11.2011	1.0	Erstellung des Dokuments	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Marion Walser verantwortlich.

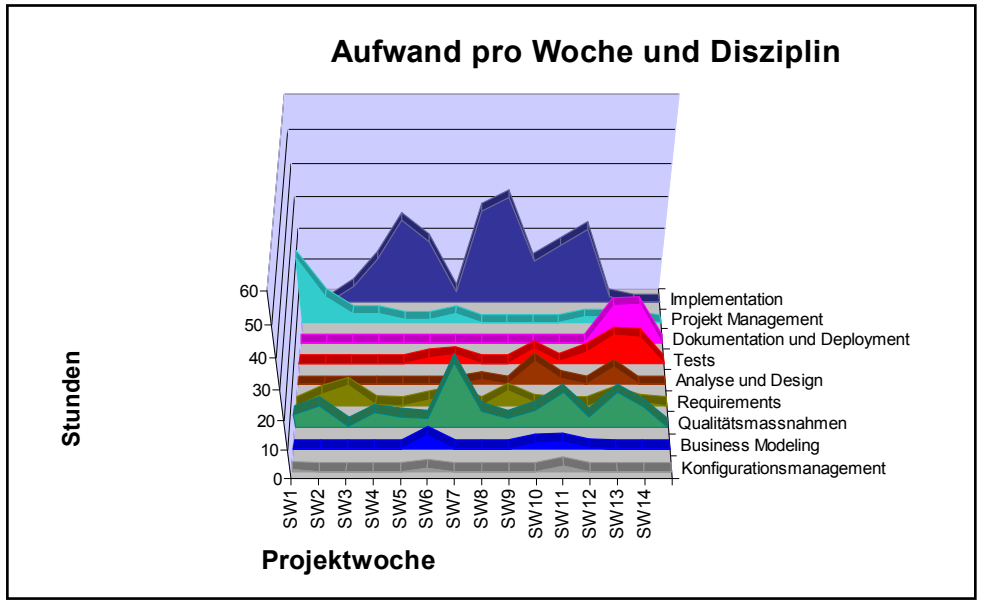
1.3. Überprüfung

Datum	Durchgeführt von
21.12.2011	Lara Mühlemann

1.4. Inhalt

2. Auswertungen zu Zeitplan

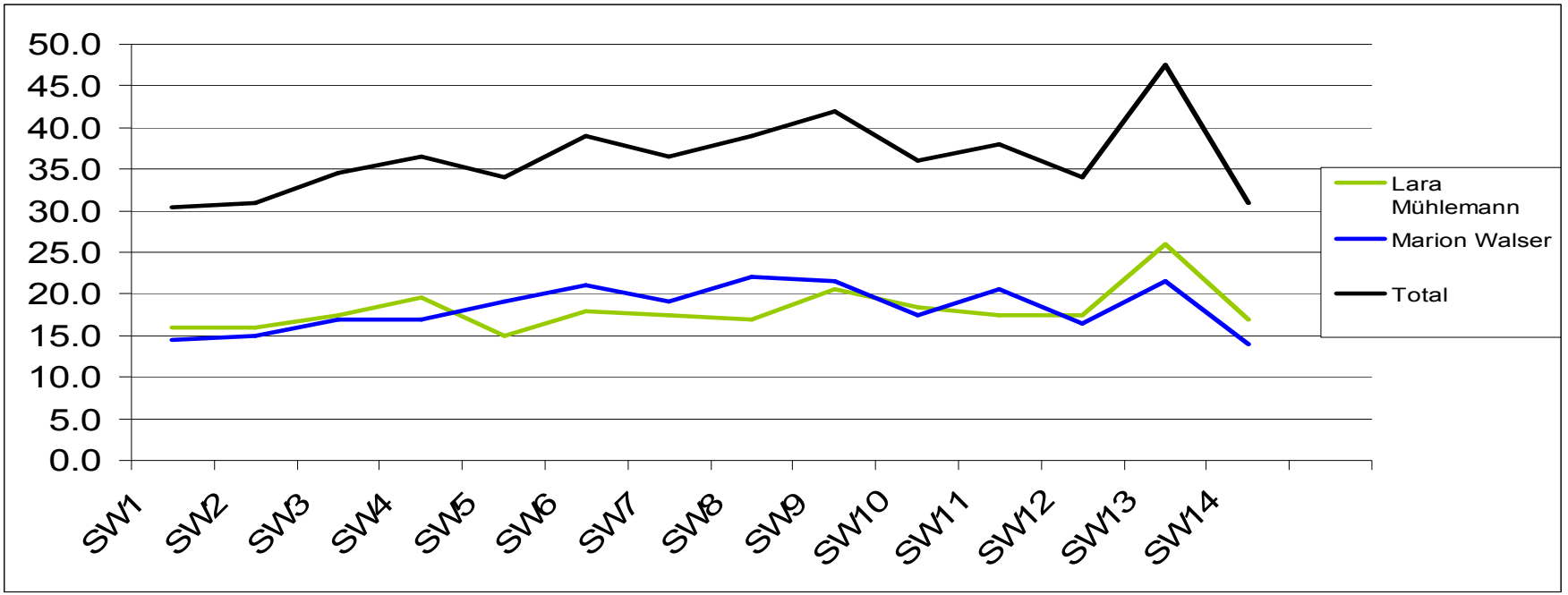
2.1. Ist-Aufwand pro Woche und Disziplin



Iteration	I1	E1			E2			C1			C2			T1		
Arbeitspaket	**SW1**	**SW2**	**SW3**	**SW4**	**SW5**	**SW6**	**SW7**	**SW8**	**SW9**	**SW10**	**SW11**	**SW12**	**SW13**	**SW14**		**Total**
Konfigurationsmanagement	0.5	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0		3.5
Business Modeling	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	2.0	2.5	0.5	0.0	0.0		10.0
Qualitätsmassnahmen	4.5	7.5	0.5	5.0	3.5	3.0	22.0	5.5	3.0	6.0	12.0	3.5	12.0	7.0		95.0
Evaluation	4.0	10.5	18.5	13.5	2.5	5.0	1.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0		57.0
Requirements	0.0	4.0	7.0	0.5	0.0	2.0	4.0	0.0	5.0	1.0	0.0	0.0	3.5	1.0		28.0
Analyse und Design	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.5	0.0	8.0	2.0	0.0	6.0	0.0		17.5
Tests	0.0	0.0	0.0	0.0	0.0	2.0	3.0	0.0	0.0	5.0	0.5	4.0	9.5	9.0		33.0
Dokumentation und Deployment	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	12.5	13.0		25.5
Projekt Management	21.5	9.0	3.0	3.0	1.0	1.0	3.0	0.0	0.0	0.0	0.0	2.0	2.0	1.0		46.5
Implementation	0.0	0.0	5.5	14.5	27.0	20.0	3.5	30.0	34.0	14.0	19.0	24.0	2.0	0.0		193.5
Total	**30.5**	**31.0**	**34.5**	**36.5**	**34.0**	**39.0**	**36.5**	**39.0**	**42.0**	**36.0**	**38.0**	**34.0**	**47.5**	**31.0**		**509.5**

2.2. Ist-Aufwand pro Woche und Person

Name	SW1	SW2	SW3	SW4	SW5	SW6	SW7	SW8	SW9	SW10	SW11	SW12	SW13	SW14		Total
Lara Mühlemann	16.0	16.0	17.5	19.5	15.0	18.0	17.5	17.0	20.5	18.5	17.5	17.5	26.0	17.0		253.5
Marion Walser	14.5	15.0	17.0	17.0	19.0	21.0	19.0	22.0	21.5	17.5	20.5	16.5	21.5	14.0		256.0
Total	**30.5**	**31.0**	**34.5**	**36.5**	**34.0**	**39.0**	**36.5**	**39.0**	**42.0**	**36.0**	**38.0**	**34.0**	**47.5**	**31.0**		**509.5**



2.3. Ist-Soll-Vergleich pro Paket

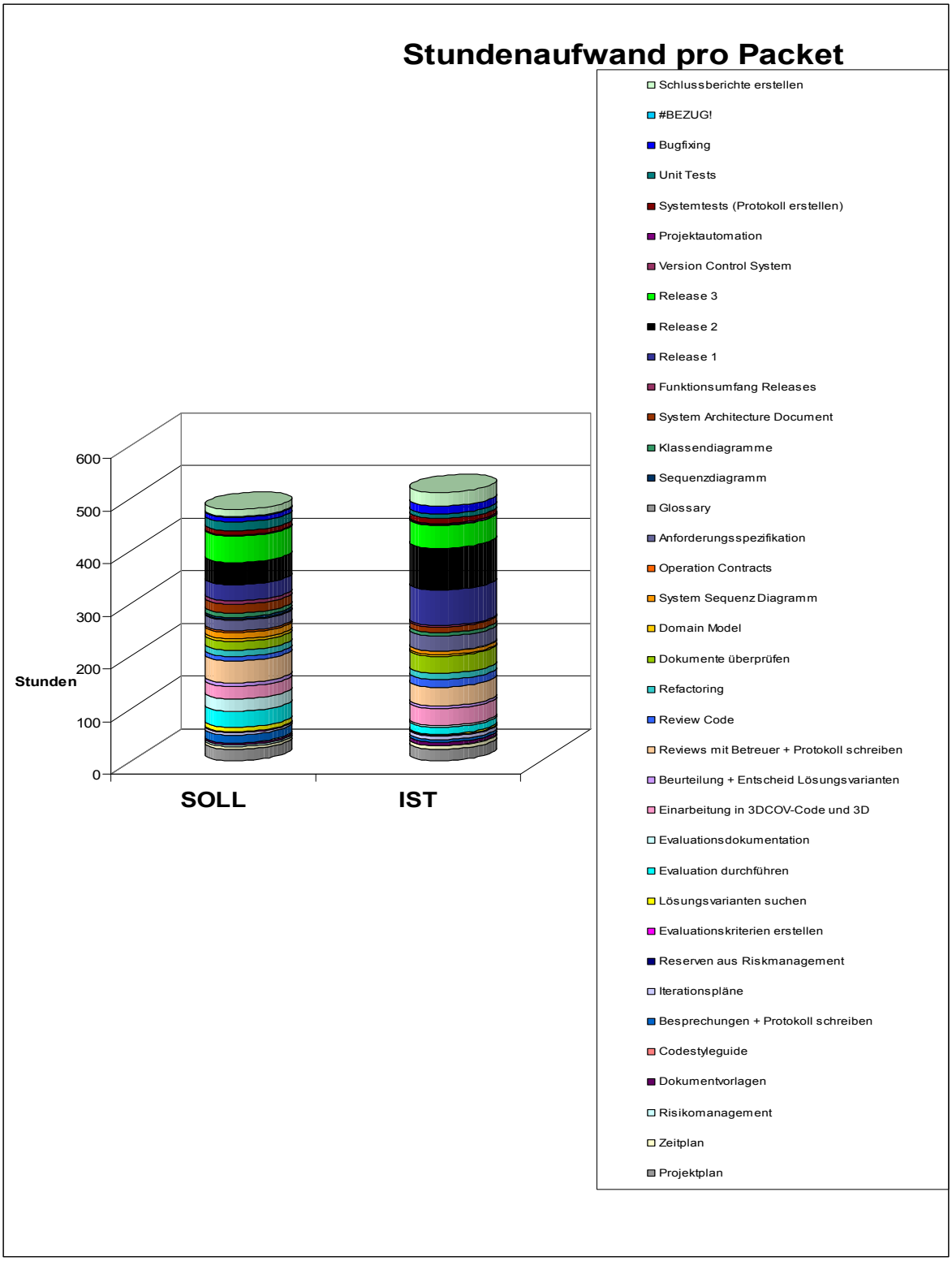
Folgende Differenzen beim Vergleich der Ist- und Sollzeiten bei den Paketen ist auffällig:

Für die Pakete der Evaluation wurde einiges zuviel Zeit eingeplant, da bei der Planung nicht berücksichtigt wurde, dass das Packet Release 1 ebenfalls zur Evaluation gehört. Dementsprechend gleicht sich der grosse Aufwand des Release 1 Pakets mit den Paketen der Evaluation wieder aus.

Bei vielen kleinen Paketen musste die geplante Zeit nicht voll genutzt werden. Aus diesem Grund konnte mehr Zeit in die Entwicklung für den Release 2 investiert werden.

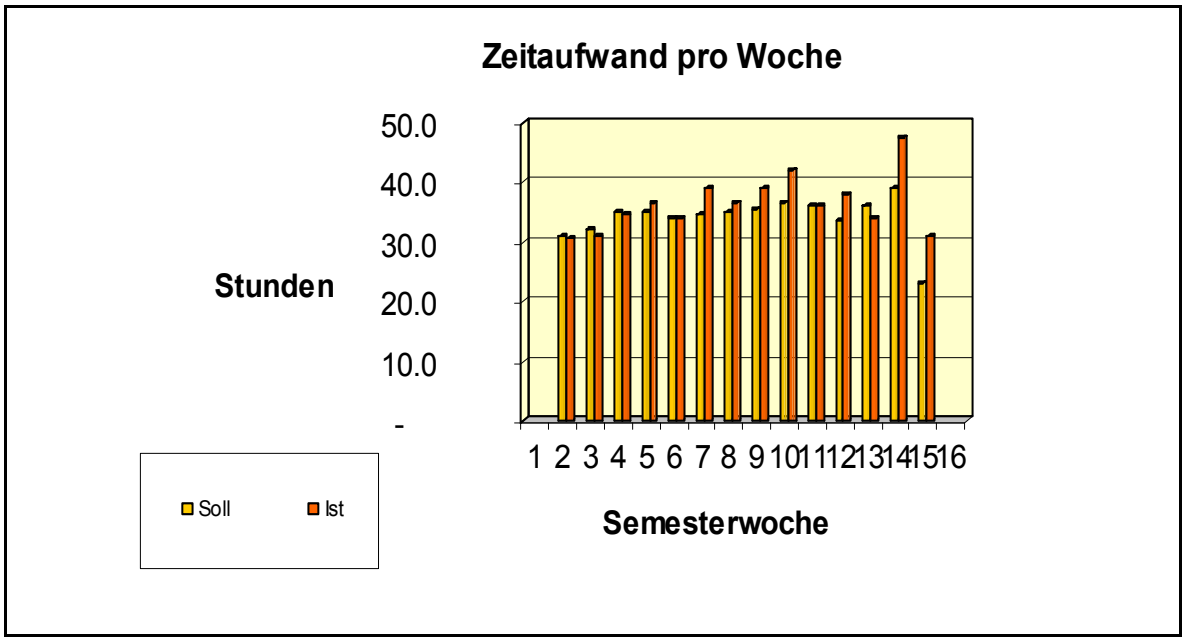
Stundenaufwand pro Packet		
Packet	**SOLL**	**IST**
Projektplan	20.5	21.5
Zeitplan	6.0	6.5
Risikomanagement	3.0	0.5
Dokumentvorlagen	3.0	5.5
Codestyleguide	1.0	0.0
Besprechungen + Protokoll schreiben	15.0	6.0
Iterationspläne	4.5	6.5
Reserven aus Riskmanagement	0.0	0.0
Evaluationskriterien erstellen	1.0	1.5
Lösungsvarianten suchen	9.0	2.5
Evaluation durchführen	30.0	12.5
Evaluationsdokumentation	24.0	3.5
Einarbeitung in 3DCOV-Code und 3D	24.0	32.0
Beurteilung + Entscheid Lösungsvarianten	6.0	5.0
Reviews mit Betreuer	42.0	36.0
Review Code	9.0	14.0
Refactoring	11.0	12.5
Dokumente überprüfen	17.5	32.5
Domain Model	5.0	3.0
System Sequenz Diagramm	11.0	6.0
Operation Contracts	4.0	1.0
Anforderungsspezifikation	19.5	27.5
Glossary	2.5	0.5
Sequenzdiagramm	4.0	0.0
Klassendiagramme	7.0	6.5
System Architecture Document	17.0	11.0
Funktionsumfang Releases	7.0	3.0
Release 1	30.0	67.0
Release 2	42.0	79.5
Release 3	51.5	44.0
Version Control System	1.0	0.5
Projektautomation	0.0	3.0
Systemtests (Protokoll erstellen)	8.0	9.5

Unit Tests	17.0	9.0
Bugfixing	9.0	14.5
Schlussberichte erstellen	14.0	25.5
Total	**476.0**	**509.5**



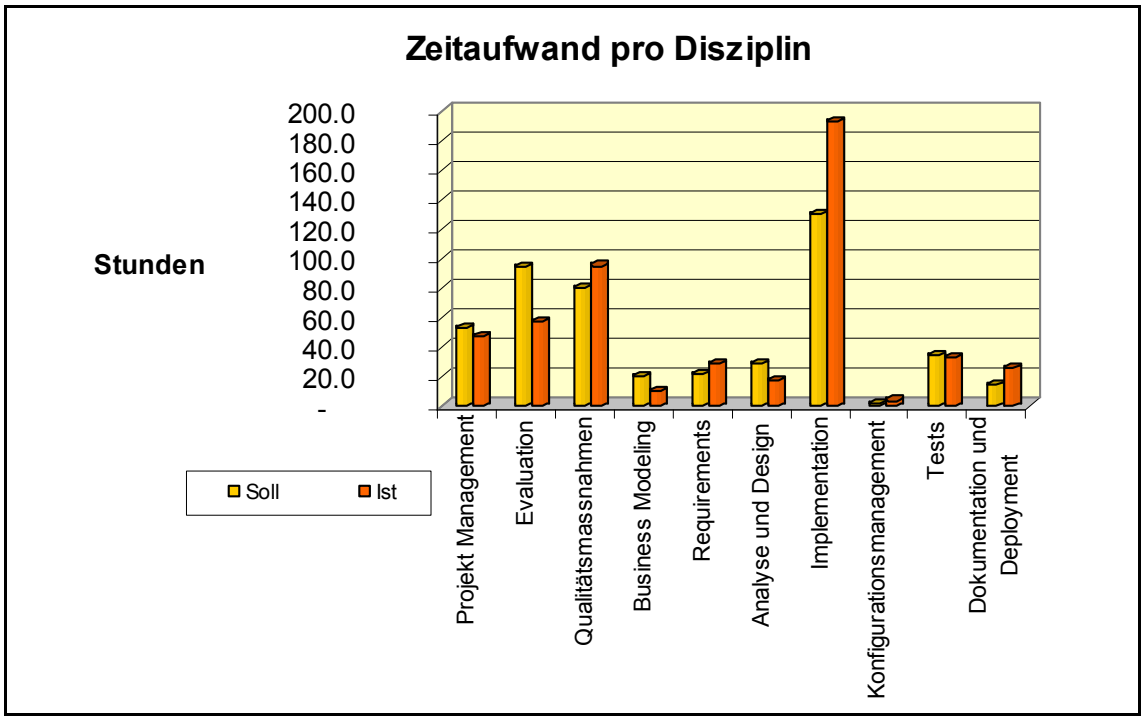
2.4. Soll-Ist-Vergleich pro Woche

Gesamter Zeitaufwand		
Woche	**Soll**	**Ist**
1	31.0	30.5
2	32.0	31.0
3	35.0	34.5
4	35.0	36.5
5	34.0	34.0
6	34.5	39.0
7	35.0	36.5
8	35.5	39.0
9	36.5	42.0
10	36.0	36.0
11	33.5	38.0
12	36.0	34.0
13	39.0	47.5
14	23.0	31.0
Total	**476.0**	**509.5**



2.5. Soll-Ist-Vergleich pro Disziplin

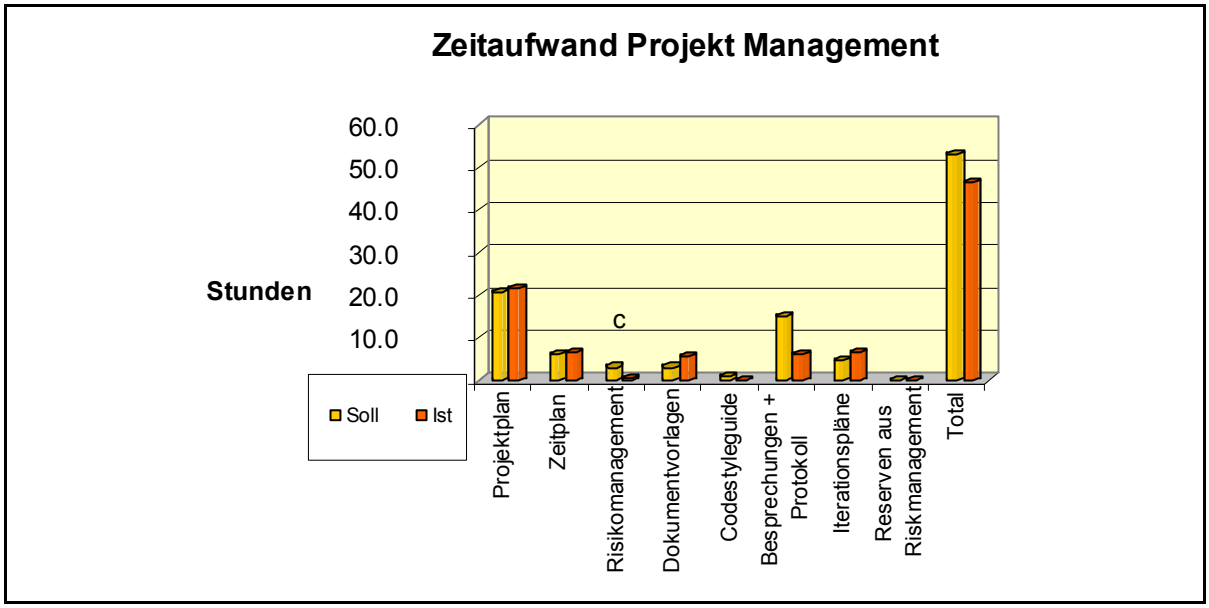
Zeitaufwand pro Disziplin		
	Soll	Ist
Projekt Management	53.0	46.5
Evaluation	94.0	57.0
Qualitätsmassnahmen	79.5	95.0
Business Modeling	20.0	10.0
Requirements	22.0	28.0
Analyse und Design	28.0	17.5
Implementation	130.5	193.5
Konfigurationsmanagement	1.0	3.5
Tests	34.0	33.0
Dokumentation und Deployment	14.0	25.5
Total	**476.0**	**509.5**



2.6. Soll-Ist-Vergleich pro einzelne Disziplin

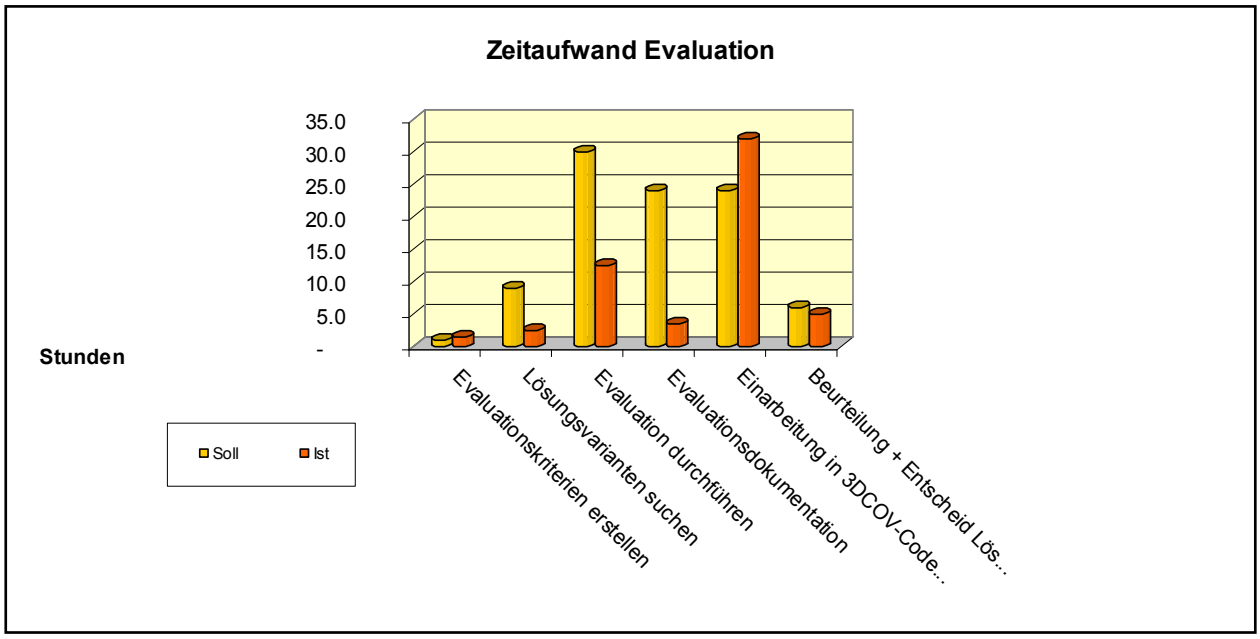
2.6.1. Soll-Ist-Vergleich Projekt Management

Zeitaufwand Projekt Management		
	Soll	**Ist**
Projektplan	20.5	21.5
Zeitplan	6.0	6.5
Risikomanagement	3.0	0.5
Dokumentvorlagen	3.0	5.5
Codestyleguide	1.0	-
Besprechungen + Protokoll	15.0	6.0
Iterationspläne	4.5	6.5
Reserven aus Riskmanagement	-	-
Total	**53.0**	**46.5**



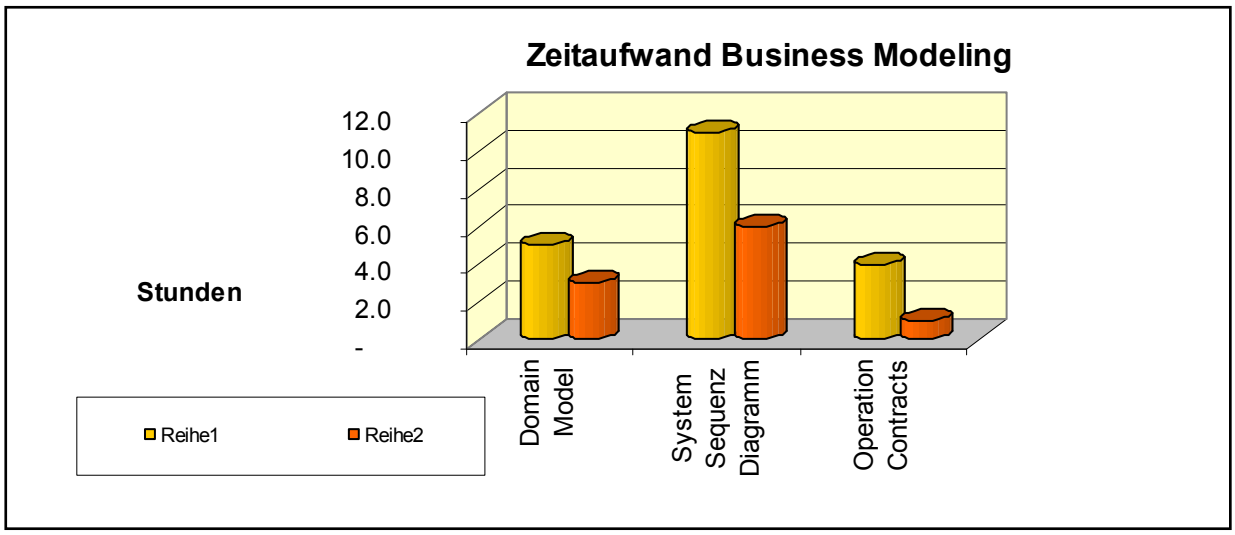
2.6.2. Soll-Ist-Vergleich Evaluation

Zeitaufwand Evaluation		
	Soll	Ist
Evaluationskriterien erstellen	1.0	1.5
Lösungsvarianten suchen	9.0	2.5
Evaluation durchführen	30.0	12.5
Evaluationsdokumentation	24.0	3.5
Einarbeitung in 3DCOV-Code und 3D	24.0	32.0
Beurteilung + Entscheid Lösungsvarianten	6.0	5.0
Total	**94.0**	**57.0**



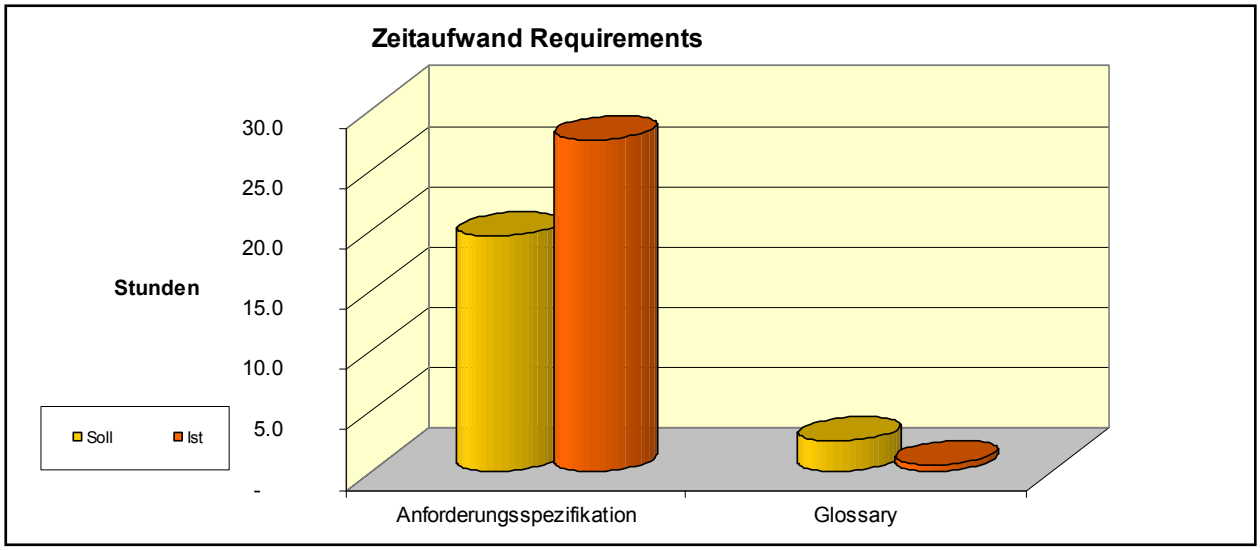
2.6.3. Soll-Ist-Vergleich Business Modeling

Zeitaufwand Business Modeling		
	Soll	Ist
Domain Model	5.0	3.0
System Sequenz Diagramm	11.0	6.0
Operation Contracts	4.0	1.0
Total	**20.0**	**10.0**



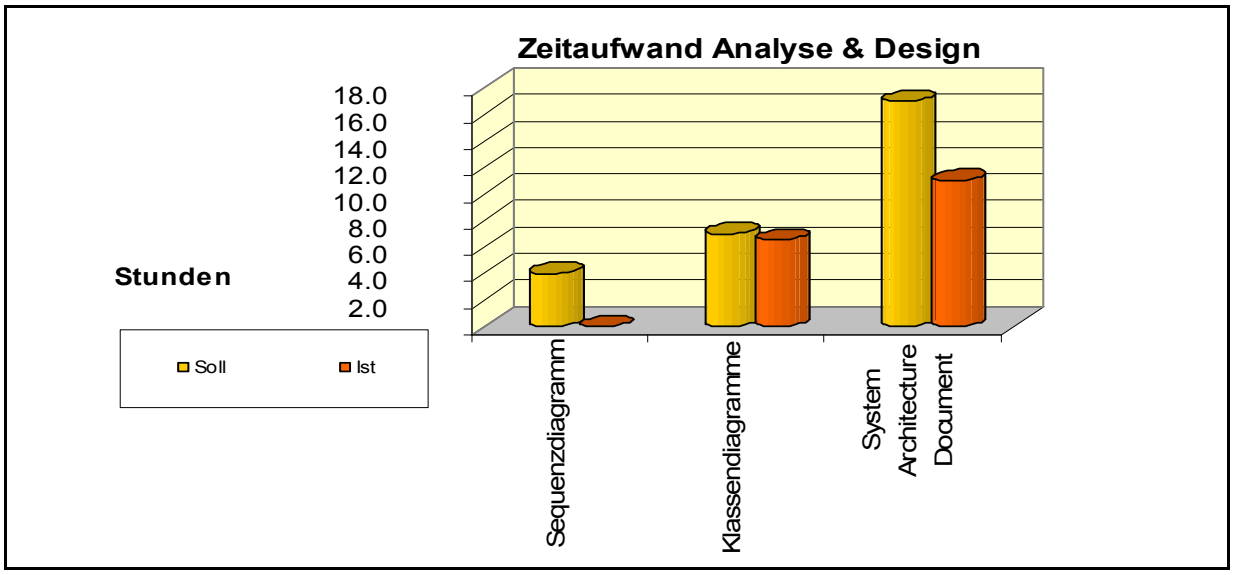
2.6.4. Soll-Ist-Vergleich Requirements

Zeitaufwand Requirements		
	Soll	Ist
Anforderungsspezifikation	19.5	27.5
Glossary	2.5	0.5
Total	**22.0**	**28.0**



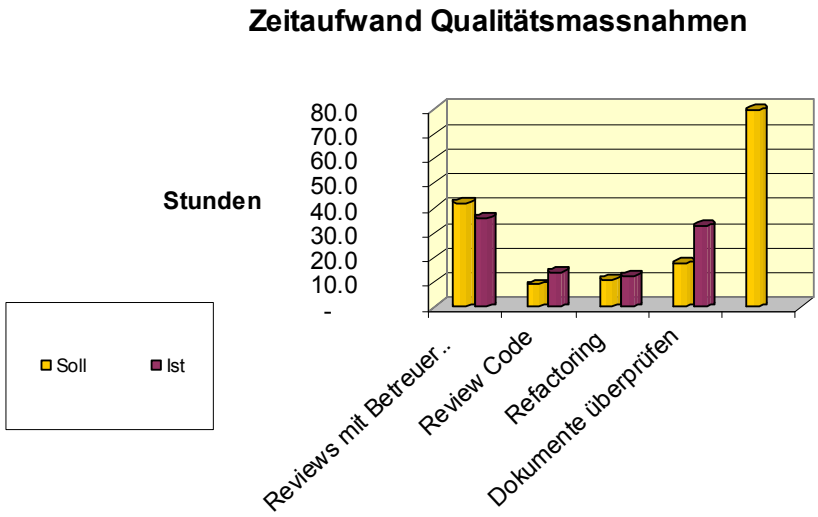
2.6.5. Soll-Ist-Vergleich Analyse & Design

Zeitaufwand Analyse & Design		
	Soll	Ist
Sequenzdiagramm	4.0	-
Klassendiagramme	7.0	6.5
System Architecture Document	17.0	11.0
Total	**28.0**	**17.5**



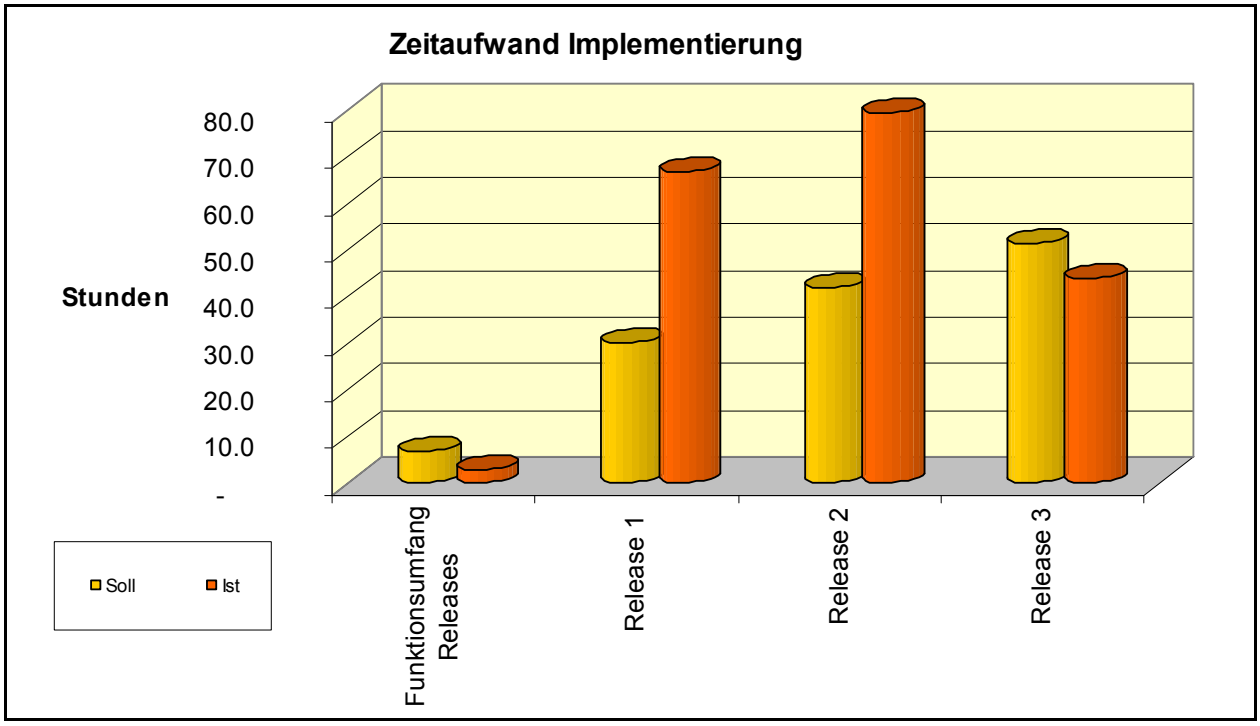
2.6.6. Soll-Ist-Vergleich Qualitätsmassnahmen

Zeitaufwand Qualitätsmassnahmen		
	Soll	Ist
Reviews mit Betreuer	42.0	36.0
Review Code	9.0	14.0
Refactoring	11.0	12.5
Dokumente überprüfen	17.5	32.5
Total	**79.5**	**95.0**



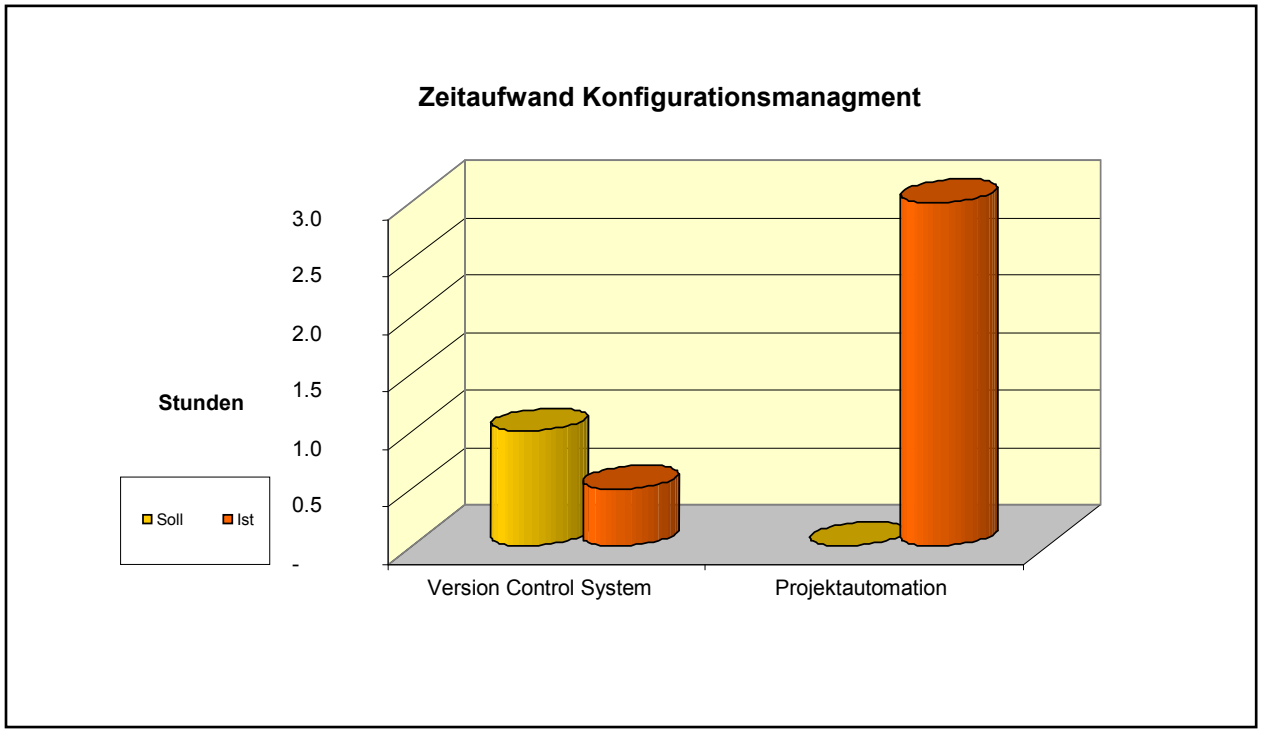
2.6.7. Soll-Ist-Vergleich Implementierung

Zeitaufwand Implementierung		
	Soll	Ist
Funktionsumfang Releases	7.0	3.0
Release 1	30.0	67.0
Release 2	42.0	79.5
Release 3	51.5	44.0
Total	**130.5**	**193.5**



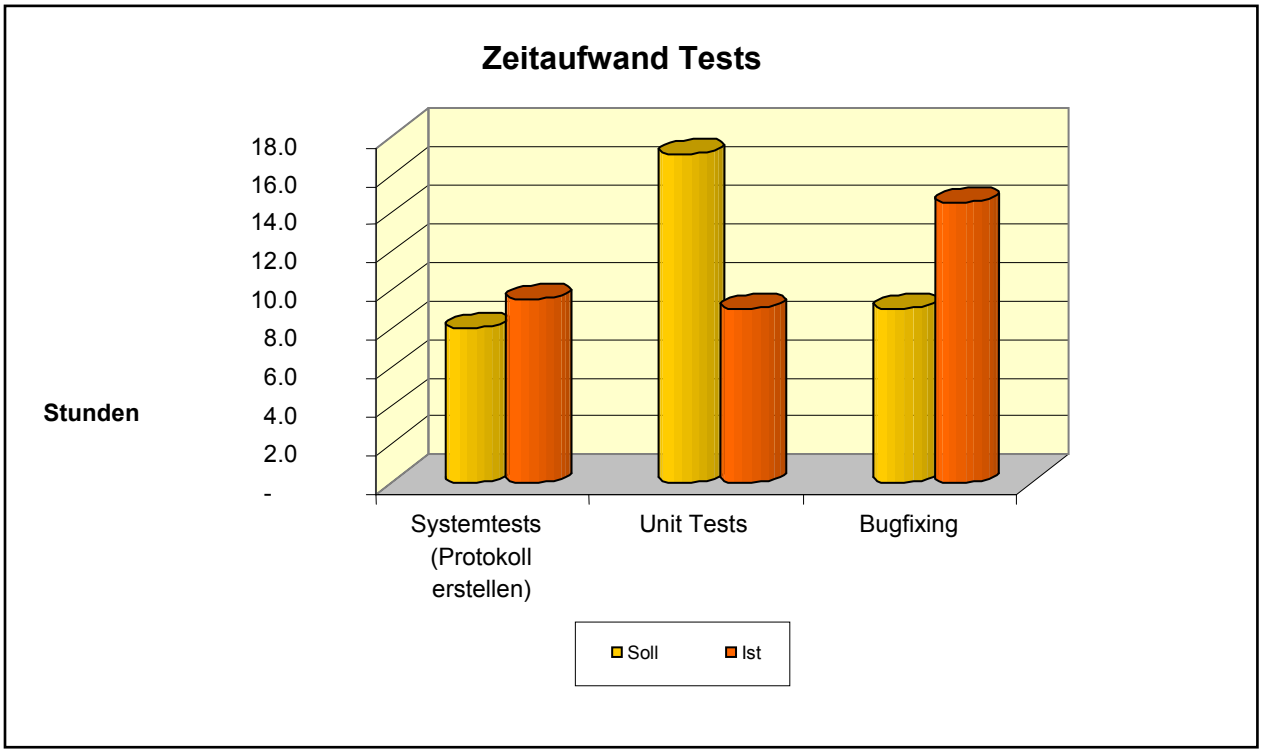
2.6.8. Soll-Ist-Vergleich Konfigurationsmanagement

Zeitaufwand Konfigurationsmanagement		
	Soll	Ist
Version Control System	1.0	0.5
Projektautomation	-	3.0
Total	**1.0**	**3.5**



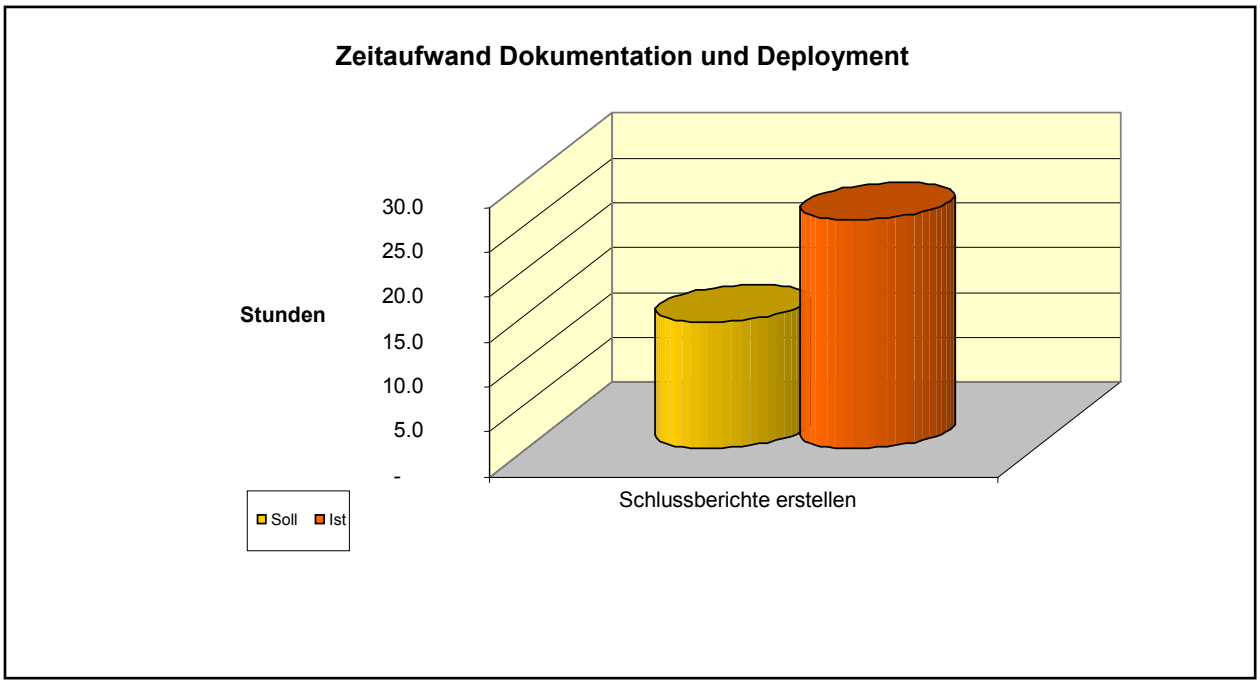
2.6.9. Soll-Ist-Vergleich Tests

Zeitaufwand Tests		
	Soll	Ist
Systemtests	8.0	9.5
Unit Tests	17.0	9.0
Bugfixing	9.0	14.5
Total	**34.0**	**33.0**



2.6.10. Soll-Ist-Vergleich Dokumentation und Deployment

Zeitaufwand Dokumentation und Deployment		
	Soll	Ist
Schlussberichte erstellen	14.0	25.5
Total	**14.0**	**25.5**



Projekt: J3DEval

Iterationsplan Elaboration 1

Version: 1.0
Datum: 7. Oktober 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
26.09.2011	1.0	Erstellung des Dokuments	Lara Mühlemann
02.10.2011	1.0	Überarbeitung Arbeitspakete	Marion Walser

1.2. Überprüfung

Datum	Durchgeführt von
27.09.2011	Lara Mühlemann, Marion Walser
07.10.2011	Lara Mühlemann

1.3. Inhalt

2. Einführung

2.1. Zweck

Der Zweck dieses Dokumentes ist es, die Aufgabenverteilung für die Phase Elaboration 1 zu unterstützen und die Ziele der aktuellen Iteration festzuhalten.

2.2. Gültigkeitsbereich

Dieser Iterationsplan gilt für die Elaboration 1 von Projektwoche 2 - 4.

2.3. Referenzen

[1] Projektplan.doc

[2] Zeitplan.xls

3. Plan

In dieser Iteration wird in hauptsächlich die Evaluation durchgeführt.

Um eine genauere Planung für die Iteration zu erhalten, wird in der folgenden Tabelle Redundanz zum Zeitplan [2] in Kauf genommen.

Regelmässig wiederkehrende Arbeitspakete wie z.B. Risikomanagement oder Besprechungen werden in der folgenden Tabelle nicht extra aufgeführt.

Arbeitspaket	Kommentar	Soll-stunden	Fertigungs-grad nach Iteration
Projektplan	Der Projektplan wird auf den aktuellen Stand korrigiert	3.5	93 %
Zeitplan	Soll-Stunden für Elaboration 2 planen	1	64 %
Iterationspläne	Iterationsplan für Elaboration 2 erstellen	3	60 %
Evaluationskriterien erstellen		1	20 %
Lösungsvarianten suchen		2.5	17 %
Evaluation durchführen	Lösungsvarianten im Detail evaluieren	24	80 %

Evaluations-dokumentation	Der Evaluationsbericht ist soweit fertig gestellt, dass mit dem Proof of Concept begonnen werden kann.	9	36 %
Einarbeitung in 3DCOV-Code und 3D		18	83 %
Beurteilung + Entscheid Lösungsvarianten		2	33 %
Dokumente überprüfen		3.5	44 %
Anforderungs-spezifikation	Dokument erstellen, Scope definieren	10	83 %
Glossary		1	50 %
Funktionsumfang Releases	Funktionsumfang von Release 1 definieren	1.5	21 %
Release 1	Beginn Proof of Concept für mögliche Lösungskandidaten	5	25 %

3.1. Meilensteine

In dieser Iteration wird auf den Meilenstein „Ende Elaboration 1" hingearbeitet. Das Datum dieses Meilensteins ist noch offen.

4. Ressourcen

Die zugeteilten Ressourcen sind im Zeitplan [2] ersichtlich.

5. Risikomanagement

ID	Risiko	Massnahme	Vorgehen bei Eintreffen	Wahrscheinlichkeit	Zusatzaufwand [h]	Gewichteter Zusatzaufwand [h]
R01	Verständnis für 3D-Graphic kann nicht erarbeitet werden	Früher Beginn mit Einarbeitung in Themengebiet	Anfrage bei Mathematik-Professor um Hilfe	20 %	30	6
R02	Nach Proof-of-Concept zeigt sich, dass sich doch nicht alle Anforderungen mit der neuen Library erfüllen lassen.	Proof of Concept soll möglichst viele Anforderungen erfüllen oder zumindest evaluiert worden sein.	Nicht erfüllbare Anforderungen werden mittels Eigenpro-grammierung ergänzt.	10 %	100	10
R03	Projekttool fällt aus	Im Projektplan unter „Verwendete Programme + Ausfallszenario" im Kapital „Infrastruktur" werden Ersatzprogramme bzw. Vorgehen definiert	Gemäss Kapitel „Infrastruktur" im Projektplan	5 %	12	0.6
R04	Datenverlust auf einem der Laptops oder auf svn-Server	Daten werden auf Laptops und auf svn-Server möglichst aktuell gehalten. Es sollten bei Datenverlust nicht mehr als 2 Stunden Arbeit verloren gehen.	Kopie einer aktuellen Version wird erstellt.	10 %	2	0.2

R05	Ausfall eines Teammitglieds durch Krankheit/Unfall	Gesunde und vernünftige Lebensweise	Arbeiten werden vom anderen Teammitglied übernommen. Falls nötig wird mit dem Auftraggeber besprochen, ob der Arbeitsumfang gekürzt werden kann.	5%	100	5
	Total gewichteter Zusatzaufwand					**21.8**

Projekt: J3DEval

Iterationsplan Elaboration 2

Version: 1.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
10.10.2011	1.0	Erstellung des Dokuments	Lara Mühlemann

1.2. Überprüfung

Datum	Durchgeführt von
12.10.2011	Marion Walser

1.3. Inhalt

2. Einführung

2.1. Zweck

Der Zweck dieses Dokumentes ist es, die Aufgabenverteilung für die Phase Elaboration 2 zu unterstützen und die Ziele der aktuellen Iteration festzuhalten.

2.2. Gültigkeitsbereich

Dieser Iterationsplan gilt für die Elaboration 2 von Projektwoche 5 - 7.

2.3. Referenzen

[1] Projektplan.doc

[2] Zeitplan.xls

3. Plan

Um eine genauere Planung für die Iteration zu erhalten, wird in der folgenden Tabelle Redundanz zum Zeitplan [2] in Kauf genommen.

Regelmässig wiederkehrende Arbeitspakete wie z.B. Risikomanagement oder Besprechungen werden in der folgenden Tabelle nicht extra aufgeführt.

Arbeitspaket	Kommentar	Soll-stunden	Fertigungs-grad nach Iteration
Lösungsvarianten suchen	Obwohl nicht alle ursprünglich geschätzten Stunden für dieses Arbeitspaket aufgebraucht wurde, ist dieses Arbeitspaket nach dieser Iteration fertig bearbeitet. Das Packet ist voraussichtlich weniger aufwändig als gedacht.	6.5	100%
Evaluation durchführen	Die Evaluation wird abgeschlossen.	6	100%
Evaluationsdokumentation	Nach dieser Iteration werden nur noch Korrekturen an der Evaluation nötig sein.	15	96%
Einarbeitung in 3DCOV-Code und 3D	Nach dieser Iteration werden die Teammitglieder mit dem bestehenden Code genügend vertraut sein	4	100%
Beurteilung + Entscheid	Nach dieser Iteration wird entschieden, welche Möglichkeit ins	4	100%

Lösungsvarianten	3DCOV integriert wird.		
Review Code	Die ersten Codereviews werden durchgeführt	1	14%
Refactoring	Refactoring aufgrund des Reviews	2	12%
Dokumente überprüfen		2	61%
Domain Model	Domainanalyse wird für den ersten Release erstellt	3	75%
System Sequenz Diagramm		3	75%
Operation Contracts		2	66%
Anforderungsspezifikation	Anforderungsspezifikation wird für diese Iteration erweitert	1.5	88%
Glossary		0.5	75%
Funtionsumfang Releases	Funktionsumfang für Release 2 wird definiert.	4	78%
Release 1	Der Release 1 wird in dieser Iteration bearbeitet.	25	100%
Systemtests	Die ersten Systemtests werden geschrieben	1	11%
Unit Tests	Erste Unit Tests werden geschrieben	1	7%
Bugfixing	Allfällige Fehler werden behoben.	1	7%

3.1. Meilensteine

In dieser Iteration wird auf den 1. Release hingearbeitet. Dieser wird auf den Mittwoch, 02. November festgelegt.

4. Ressourcen

Die zugeteilten Ressourcen sind im Zeitplan [2] ersichtlich.

5. Risikomanagement

ID	Risiko	Massnahme	Vorgehen bei Eintreffen	Wahrscheinlichkeit	Zusatzaufwand [h]	Gewichteter Zusatzaufwand [h]
R01	Verständnis für 3D-Graphic kann nicht erarbeitet werden	Früher Beginn mit Einarbeitung in Themengebiet	Anfrage bei Mathematik-Professor um Hilfe	15%	30	4.5
R02	Nach Proof-of-Concept zeigt sich, dass sich doch nicht alle Anforderungen mit der neuen Library erfüllen lassen.	Proof of Concept soll möglichst viele Anforderungen erfüllen oder zumindest evaluiert worden sein.	Nicht erfüllbare Anforderungen werden mittels Eigenpro-grammierung ergänzt.	10 %	100	10
R03	Projekttool fällt aus	Im Projektplan unter „Verwendete Programme + Ausfallszenario" im Kapital „Infrastruktur" werden Ersatzprogramme bzw. Vorgehen definiert	Gemäss Kapitel „Infrastruktur" im Projektplan	4 %	12	0.5
R04	Datenverlust auf einem der Laptops oder auf svn-Server	Daten werden auf Laptops und auf svn-Server möglichst aktuell gehalten. Es sollten bei Datenverlust nicht mehr als 2 Stunden Arbeit verloren gehen.	Kopie einer aktuellen Version wird erstellt.	8 %	2	0.2

R05	Ausfall eines Teammitglieds durch Krankheit/Unfall	Gesunde und vernünftige Lebensweise	Arbeiten werden vom anderen Teammitglied übernommen. Falls nötig wird mit dem Auftraggeber besprochen, ob der Arbeitsumfang gekürzt werden kann.	5%	100	5
	Total gewichteter Zusatzaufwand					**20.2**

Projekt: J3DEval

Iterationsplan Construction 1

Version: 1.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
04.11.2011	1.0	Erstellung des Dokuments	Marion Walser

1.2. Überprüfung

Datum	Durchgeführt von
07.11.2011	Lara Mühlemann

1.3. Inhalt

2. Einführung

2.1. Zweck

Der Zweck dieses Dokumentes ist es, die Aufgabenverteilung für die Phase Construction 1 zu unterstützen und die Ziele der aktuellen Iteration festzuhalten.

2.2. Gültigkeitsbereich

Dieser Iterationsplan gilt für die Construction 1 von Mitte Projektwoche 7 bis Mitte Projektwoche 11.

2.3. Referenzen

[1] Projektplan.doc

[2] Zeitplan.xls

3. Plan

Um eine genauere Planung für die Iteration zu erhalten, wird in der folgenden Tabelle Redundanz zum Zeitplan [2] in Kauf genommen.

Regelmässig wiederkehrende Arbeitspakete wie z.B. Risikomanagement oder Besprechungen werden in der folgenden Tabelle nicht extra aufgeführt.

Arbeitspaket	Kommentar	Soll-stunden	Fertigungs-grad nach Iteration
Dokumentenvorlage	Kopf- und Fusszeile ist besser sichtbar abzutrennen	1	100 %
Review Code	Code-Review für Implementationen in Release 2 ist durchzuführen	4	70 %
Refactoring	U.a. Umbenennen des Packages j3deval sowie logische Unterpackage-Struktur aufbauen	7	70 %
Dokumente überprüfen		4	80 %
Anforderungsspezifikation	Weitere Use Cases sind auszuführen	6	80 %

Glossary		0.5	90 %
Domain-Model	Domain-Model ist zu erweitern, um die Konzepte von Release 2	2	90 %
System Sequenz Diagramm	Weitere System Sequenz-Diagramme für die Use Cases von Release 2 sind zu erstellen	6	90 %
Operation Contracts		2	90 %
Sequenzdiagramm	Sequenzdiagramme für die komplexen Methoden sind zu erstellen	4	90 %
Klassendiagramme	Klassendiagramme sind im EA mit Source Code synchron zu halten	6	90 %
System Architecture Document		10	80 %
Funtionsumfang Releases	Funktionsumfang für Release 2 wurde auf Beginn Construction 1 verschoben.	1	90%
Release 2	Der Release 2 wird in dieser Iteration bearbeitet.	42	100%
Unit Tests		8	80 %

3.1. Meilensteine

In dieser Iteration wird auf den 2. Release hingearbeitet. Diese Fertigstellung erfolgt auf den Mittwoch, 30. November.

4. Ressourcen

Die zugeteilten Ressourcen sind im Zeitplan [2] ersichtlich.

5. Risikomanagement

ID	Risiko	Massnahme	Vorgehen bei Eintreffen	Wahrscheinlichkeit	Zusatzaufwand [h]	Gewichteter Zusatzaufwand [h]
R01	Verständnis für 3D-Graphic kann nicht erarbeitet werden	Früher Beginn mit Einarbeitung in Themengebiet	Anfrage bei Mathematik-Professor um Hilfe	2 %	30	0.6
R02	Nach Proof-of-Concept zeigt sich, dass sich doch nicht alle Anforderungen mit der neuen Library erfüllen lassen.	Proof of Concept soll möglichst viele Anforderungen erfüllen oder zumindest evaluiert worden sein.	Nicht erfüllbare Anforderungen werden mittels Eigenpro-grammierung ergänzt.	5 %	100	5
R03	Projekttool fällt aus	Im Projektplan unter „Verwendete Programme + Ausfallszenario" im Kapital „Infrastruktur" werden Ersatzprogramme bzw. Vorgehen definiert	Gemäss Kapitel „Infrastruktur" im Projektplan	3 %	12	0.4
R04	Datenverlust auf einem der Laptops oder auf svn-Server	Daten werden auf Laptops und auf svn-Server möglichst aktuell gehalten. Es sollten bei Datenverlust nicht mehr als 2 Stunden Arbeit	Kopie einer aktuellen Version wird erstellt.	5 %	2	0.2

		verloren gehen.				
R05	Ausfall eines Teammitglieds durch Krankheit/Unfall	Gesunde und vernünftige Lebensweise	Arbeiten werden vom anderen Teammitglied übernommen. Falls nötig wird mit dem Auftraggeber besprochen, ob der Arbeitsumfang gekürzt werden kann.	3%	80	2.4
	Total gewichteter Zusatzaufwand					**8.6**

Projekt: J3DEval

Iterationsplan Construction 2

Version: 1.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
05.12.2011	1.0	Erstellung des Dokuments	Lara Mühlemann
05.12.2011	1.0	Arbeitspakete überarbeitet	Marion Walser

1.2. Überprüfung

Datum	Durchgeführt von
06.12.2011	Lara Mühlemann

1.3. Inhalt

2. Einführung

2.1. Zweck

Der Zweck dieses Dokumentes ist es, die Aufgabenverteilung für die Phase Construction 2 zu unterstützen und die Ziele der aktuellen Iteration festzuhalten.

2.2. Gültigkeitsbereich

Dieser Iterationsplan gilt für die Construction 2 von Mitte Projektwoche 11 bis Mitte Projektwoche 13.

2.3. Referenzen

[1] Projektplan.doc

[2] Zeitplan.xls

3. Plan

Um eine genauere Planung für die Iteration zu erhalten, wird in der folgenden Tabelle Redundanz zum Zeitplan [2] in Kauf genommen.

Regelmässig wiederkehrende Arbeitspakete wie z.B. Risikomanagement oder Besprechungen werden in der folgenden Tabelle nicht extra aufgeführt.

Arbeitspaket	Kommentar	Soll-stunden	Fertigungs-grad nach Iteration
Refactoring		2	95 %
Review Code		4	95 %
Anforderungsspezifikation	Weitere Use Cases sind auszuführen	2	100 %
System Sequenz Diagramm	Weitere System Sequenz-Diagramme für die Use Cases von Release 3 sind zu erstellen	2	100 %
Klassendiagramme	Klassendiagramme sind im EA mit Source Code synchron zu halten	1	100 %
System Architecture Document		5	100 %

Funtionsumfang Releases	Funktionsumfang für Release 3 wurde auf Beginn Construction 1 verschoben.	0.5	100 %
Release 3		37	100%
Unit Tests		4	100 %
Systemtests		2	95 %
Dokumente überprüfen		2	95%
Bugfixing		2	95 %

3.1. Meilensteine

In dieser Iteration wird auf den 3. Release hingearbeitet.

4. Ressourcen

Die zugeteilten Ressourcen sind im Zeitplan [2] ersichtlich.

5. Risikomanagement

ID	Risiko	Massnahme	Vorgehen bei Eintreffen	Wahrscheinlichkeit	Zusatzaufwand [h]	Gewichteter Zusatzaufwand [h]
R01	Verständnis für 3D-Graphic kann nicht erarbeitet werden	Früher Beginn mit Einarbeitung in Themengebiet	Anfrage bei Mathematik-Professor um Hilfe	0 %	30	0
R02	Nach Proof-of-Concept zeigt sich, dass sich doch nicht alle Anforderungen mit der neuen Library erfüllen lassen.	Proof of Concept soll möglichst viele Anforderungen erfüllen oder zumindest evaluiert worden sein.	Nicht erfüllbare Anforderungen werden mittels Eigenpro-grammierung ergänzt.	3 %	100	3
R03	Projekttool fällt aus	Im Projektplan unter „Verwendete Programme + Ausfallszenario" im Kapital „Infrastruktur" werden Ersatzprogramme bzw. Vorgehen definiert	Gemäss Kapitel „Infrastruktur" im Projektplan	1 %	12	0.1
R04	Datenverlust auf einem der Laptops oder auf svn-Server	Daten werden auf Laptops und auf svn-Server möglichst aktuell gehalten. Es sollten bei Datenverlust nicht mehr als 2 Stunden Arbeit	Kopie einer aktuellen Version wird erstellt.	2 %	2	0

		verloren gehen.				
R05	Ausfall eines Teammitglieds durch Krankheit/Unfall	Gesunde und vernünftige Lebensweise	Arbeiten werden vom anderen Teammitglied übernommen. Falls nötig wird mit dem Auftraggeber besprochen, ob der Arbeitsumfang gekürzt werden kann.	1%	50	0.5
	Total gewichteter Zusatzaufwand					**3.6**

Projekt: J3DEval

Iterationsplan Transition

Version: 1.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
17.12.2011	1.0	Erstellung des Dokuments	Lara Mühlemann

1.2. Überprüfung

Datum	Durchgeführt von
17.12.2011	Marion Walser

1.3. Inhalt

2. Einführung

2.1. Zweck

Der Zweck dieses Dokumentes ist es, die Aufgabenverteilung für die Phase Transition zu unterstützen und die Ziele der aktuellen Iteration festzuhalten.

2.2. Gültigkeitsbereich

Dieser Iterationsplan gilt für die Transition von Mitte Projektwoche 13 bis zum Projektende am Freitag in Woche 14.

2.3. Referenzen

[1] Projektplan.doc

[2] Zeitplan.xls

3. Plan

Um eine genauere Planung für die Iteration zu erhalten, wird in der folgenden Tabelle Redundanz zum Zeitplan [2] in Kauf genommen.

Regelmässig wiederkehrende Arbeitspakete wie z.B. Risikomanagement oder Besprechungen werden in der folgenden Tabelle nicht extra aufgeführt.

Arbeitspaket	Kommentar	Soll-stunden	Fertigungs-grad nach Iteration
Refactoring		1	100 %
Review Code		1	100 %
Unit Tests		4	100 %
Systemtests		3	100 %
Dokumente überprüfen		2	100%
Bugfixing		2	100 %
Schlussberichte erstellen		14	100%

3.1. Meilensteine

In dieser Iteration wird auf den 3. Release hingearbeitet.

4. Ressourcen

Die zugeteilten Ressourcen sind im Zeitplan [2] ersichtlich.

5. Risikomanagement

ID	Risiko	Massnahme	Vorgehen bei Eintreffen	Wahrscheinlichkeit	Zusatzaufwand [h]	Gewichteter Zusatzaufwand [h]
R01	Verständnis für 3D-Graphic kann nicht erarbeitet werden	Früher Beginn mit Einarbeitung in Themengebiet	Anfrage bei Mathematik-Professor um Hilfe	0 %	30	0
R02	Nach Proof-of-Concept zeigt sich, dass sich doch nicht alle Anforderungen mit der neuen Library erfüllen lassen.	Proof of Concept soll möglichst viele Anforderungen erfüllen oder zumindest evaluiert worden sein.	Nicht erfüllbare Anforderungen werden mittels Eigenpro-grammierung ergänzt.	0 %	100	0
R03	Projekttool fällt aus	Im Projektplan unter „Verwendete Programme + Ausfallszenario" im Kapital „Infrastruktur" werden Ersatzprogramme bzw. Vorgehen definiert	Gemäss Kapitel „Infrastruktur" im Projektplan	0 %	12	0
R04	Datenverlust auf einem der Laptops oder auf svn-Server	Daten werden auf Laptops und auf svn-Server möglichst aktuell gehalten. Es sollten bei Datenverlust nicht mehr als 2 Stunden Arbeit	Kopie einer aktuellen Version wird erstellt.	1 %	2	0

		verloren gehen.				
R05	Ausfall eines Teammitglieds durch Krankheit/Unfall	Gesunde und vernünftige Lebensweise	Arbeiten werden vom anderen Teammitglied übernommen. Falls nötig wird mit dem Auftraggeber besprochen, ob der Arbeitsumfang gekürzt werden kann.	1%	12	0.1
	Total gewichteter Zusatzaufwand					**0.1**

Projekt: 3DEval

Evaluationsdokumentation

Version: 3.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
21.09.2011	1.0	Dokument erstellt, gefundene Libraries ergänzt	Marion Walser
03.10.2011	1.0	Libraries evaluiert	Marion Walser
10.10.2011	1.0	Libraries evaluiert	Lara Mühlemann
11.10.2011	1.0	Auswertung erstellt	Marion Walser
30.10.2011	2.0	Beurteilung der Lösungsvarianten nach Prototyping erstellt und Entscheid	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Marion Walser verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
12.10.2011	Lara Mühlemann
01.11.2011	Lara Mühlemann
17.12.2011	Lara Mühlemann

1.4. Inhalt

2. Einführung

2.1. Zweck

In diesem Dokument wird dokumentiert, was bei der Evaluation erarbeitet wurde. Es werden Notizen über gelesene Artikel, Bücher und andere Unterlagen erfasst und dabei entstandene Gedankengänge aufgeschrieben.

2.2. Gültigkeitsbereich

Dieses Dokument ist während der Projektdauer gültig.

2.3. Definitionen und Abkürzungen

Siehe Glossar.doc

2.4. Referenzen

[1] http://java.sun.com/products/java-media/2D/samples/suite/Colors/Rotator3D.java
[2] http://code.google.com/p/java-m3g/
[3] http://code.google.com/p/openskia/
[4] http://de.wikipedia.org/wiki/Java_3D
[5] http://de.wikipedia.org/wiki/Jogl
[6] http://de.wikipedia.org/wiki/Open_Inventor
[7] http://de.wikipedia.org/wiki/X3D
[8] http://developer.hoops3d.com/documentation/Hoops3DGS/tech_overview/TechnicalOverview_C.html#_Toc470077868
[9] http://en.wikipedia.org/wiki/Away3D
[10] http://en.wikipedia.org/wiki/ClanLib
[11] http://en.wikipedia.org/wiki/CrystalSpace
[12] http://en.wikipedia.org/wiki/Glide_API
[13] http://en.wikipedia.org/wiki/Graphics_library
[14] http://en.wikipedia.org/wiki/HOOPS_3D_Graphics_System
[15] http://en.wikipedia.org/wiki/JMonkey_Engine
[16] http://en.wikipedia.org/wiki/JT_%28visualization_format%29
[17] http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries
[18] http://en.wikipedia.org/wiki/Microsoft_Direct3D
[19] http://en.wikipedia.org/wiki/MiniGL
[20] http://en.wikipedia.org/wiki/Mobile_3D_Graphics_API
[21] http://en.wikipedia.org/wiki/O3D
[22] http://en.wikipedia.org/wiki/OGRE
[23] http://en.wikipedia.org/wiki/OpenGL
[24] http://en.wikipedia.org/wiki/OpenGL_Performer
[25] http://en.wikipedia.org/wiki/OpenSceneGraph
[26] http://en.wikipedia.org/wiki/OpenSG
[27] http://en.wikipedia.org/wiki/QSDK

[28] http://en.wikipedia.org/wiki/RenderMan_Interface_Specification
[29] http://en.wikipedia.org/wiki/RenderWare
[30] http://en.wikipedia.org/wiki/WebGL
[31] http://irrlicht.sourceforge.net/
[32] http://jausoft.com/gl4java.html
[33] http://jausoft.com/products/gl4java
[34] http://jmonkeyengine.org/downloads/
[35] http://lwjgl.org
[36] http://pointclouds.org/
[37] http://pointclouds.org/downloads/
[38] http://www.ambiera.com/copperlicht/
[39] http://www.jogl.info
[40] http://www.libsdl.org/
[41] http://www.mak.com/products/visualize/common-operating-picture.html
[42] http://www.nvidia.de/object/scenix-home.html
[43] http://www.opengl.org/wiki/Getting_started
[44] http://www.panda3d.org/
[45] http://www.plm.automation.siemens.com/en_sg/products/open/jtopen/technology/jt_showcase.shtml
[46] http://www.presagis.com/products_services/products/ms/visualization/vega_prime/
[47] http://www.quesa.org/
[48] http://www.segger.com/cms/emwin.html
[49] http://www.sfml-dev.org/index.php
[50] http://www.web3d.org/about/overview/

3. Datenerhebung

3.1. Auflistung der existierenden und gefundenen 3D-Libraries

Nr.	Library	Type	Quelle	Stand Evaluation
1	X3D	Low-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
2	OpenGL (and the OpenGL Shading Language)	Low-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
3	RenderMan	Low-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
4	RenderWare	Low-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
5	Glide API	Low-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
6	ClanLib	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
7	CrystalSpace	High-level 3D	http://en.wikipedia.org/wiki/List_	erl.

		API	of_3D_graphics_libraries	
8	HOOPS 3D Graphics System	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
9	Irrlicht Engine	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
10	Java 3D	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	Lösungs-variante
11	JMonkey Engine	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
12	Mobile 3D Graphics API (M3G; JSR-184)	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
13	JT Open from Siemens PLM Software	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
14	Nvidia Scene Graph (NVSG)	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
15	OGRE	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
16	OpenGL Performer	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
17	OpenSceneGraph	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
18	OpenSG	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
19	QSDK	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
20	Quesa	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
21	Vega Prime by Presagis	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
22	VR-Vantage by VT MAK	High-level 3D API	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
23	CopperLicht	JavaScript-based engines	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
24	O3D	JavaScript-based engines	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
25	Away3D	Flash-based engines	http://en.wikipedia.org/wiki/List_of_3D_graphics_libraries	erl.
26	Direct3D		http://en.wikipedia.org/wiki/Graphics_library	erl.
27	MiniGL		http://en.wikipedia.org/wiki/Graphics_library	erl.
28	WebGL		http://en.wikipedia.org/wiki/Graphics_library	erl.
29	Openskia		http://en.wikipedia.org/wiki/Graphics_library	erl.

30	Open Inventor	http://en.wikipedia.org/wiki/Graphics_library	erl.
31	emWin (an Embedded Graphics Library)	http://en.wikipedia.org/wiki/Graphics_library	erl.
32	SFML	http://en.wikipedia.org/wiki/Graphics_library	erl.
33	Simple DirectMedia Layer	http://en.wikipedia.org/wiki/Graphics_library	erl.
34	Panda3D	http://www.panda3d.org/	erl.
35	Lightweight Java Game Library (LWJGL)	http://lwjgl.org/	Lösungs-variante
36	Point Cloud Library (or PCL)	http://pointclouds.org/	erl.
37	JOGL	http://www.jogl.info	erl.
38	**GL4Java**	**http://jausoft.com/products/gl4java**	erl.

3.2. Kriterien

- Zeitaufwand zur Erstellung der Lösungsvariante
- Abhängigkeiten (Betriebssystem / Grafikkarte)
- Lizenzen
- Notwendigkeit einer Installation
- Zusätzliche Library benötigt
- Dokumentation vorhanden und qualitativ gut
- Performanz der Transformation
- Aktualität der Library
- Gewährleistung von Aktualisierungen gegeben

4. Lösungsvarianten

4.1. Beschreibung Lösungsvarianten (vor Proof of Concept)

- Ersetzen der Java 3D API durch Eigenprogrammierung: Ganze Darstellung aufbauend auf Java 2D API neu programmieren (Beispiel Rotator 3D [1] zeigt, dass dies von der Performanz machbar sein sollte). Die Schnittstelle zu 3DCOV ist das Package „cov3dmodel".

- Ersetzen der Java 3D API und des Packages "cov3dmodel" im 3DCOV durch Eigenprogrammierung: Durch das zusätzliche Ersetzen des Packages „cov3dmodel" im 3DCOV ergibt sich eine andere Schnittstelle, die keinen Bezug zu der Java 3D API mehr hat.
- Java 3D API entkoppeln: Java 3D API so umstellen, dass die Grafikkarte nicht mehr direkt angesprochen wird.
- LWJGL (Lightweight Java Game Library) entkoppeln inkl. Schnittstelle zu 3DCOV: Zusätzlich zur Lösungsvariante „Java 3D API entkoppeln" muss dafür eine Kopplung zwischen Package „cov3dmodel" im 3DCOV und der LWJGL programmiert werden.

Die Variante mit einer externen Library wird nicht aufgeführt, da keine passende Library gefunden wurde (siehe Kapitel „Analyse einzelner 3D Libraries").

5. Evaluationsauswertung

5.1. Vorgehen

Die Kriterien sind nach Priorität gewichtet. Damit wird berücksichtigt, dass die Präferenzen des Kunden und die einfache Handhabung des Programms stärker berücksichtigt werden. Die einzelnen Kriterien wurden anschliessend pro Lösungsvariante mittels einer Skala von 0 bis 10 eingeschätzt, wobei der Wert 10 die beste Bewertung ist.

Anhand des gewichteten Totals kann eine Grundlage für eine Entscheidung gegeben werden.

Diese Auswertung erfolgte in der Grobevaluation und berücksichtigt nicht die Umsetzbarkeit, die in der darauf folgenden Phase ermittelt wurde. Für den definitiven Entscheid einer Lösungsvariante wurde das Proof of Concept miteinbezogen. Weitere Informationen dazu sind im Kapitel „Beurteilung der Lösungsvarianten nach Prototyping" und „Entscheid für „Ersetzen der Java3D API durch Eigenprogrammierung"" ersichtlich.

5.2. Auswertung

Lösungsvarianten / Kriterien	Zeitaufwand zur Erstellung	Abhängigkeiten	Lizenzen	Notwendigkeit einer Installation	Zusätzliche Library benötigt	Dokumentation vorhanden und qualitativ gut	Performanz der Transformation	Aktualität der Library	Gewährleistung von Aktualisierungen	Total Punkte gewichtet
Gewichtung	5	20	20	20	15	5	5	5	5	**100**
Ersetzen der Java 3D API durch Eigen-programmier-ung	2	10	10	10	10	5	2	8	8	**875**
Ersetzen der Java 3D API und „cov3dmodel"	0	10	10	10	10	5	2	8	8	**865**
Java 3D API entkoppeln	6	10	10	10	10	7	2	4	8	**885**
LWJGL entkoppeln	4	10	10	10	10	7	2	4	8	**875**

5.3. Erklärungen zur Auswertung

5.3.1. Ersetzen der Java 3D API durch Eigenprogrammierung

Die Variante verursacht viel Programmieraufwand und die Performanz könnte Probleme verursachen. Die Unabhängigkeit der Lösungsvariante und Chance, dass das Programm auch auf neuen Betriebssysteme bzw. Grafikkarten läuft, sind gross, da zu erwarten ist, dass Java 2D weiterhin aktualisiert wird.

Da die Schnittstelle beibehalten wird, ist keine Änderung am 3DCOV Code nötig. Der Nachteil davon ist, dass die Schnittstellen Klassenstrukturen vorgeben, die für die Programmierung mit Java2D nicht ideal sind.

5.3.2. Ersetzen der Java 3D API und des Packages "cov3dmodel" im 3DCOV durch Eigenprogrammierung

Hier ist ebenso der Programmieraufwand der Negativpunkt der Variante. Dies verstärkt sich noch zur vorhergehenden Variante dadurch, dass auch die spezifische Darstellung von 3DCOV ersetzt werden würde.

5.3.3. Java 3D API entkoppeln

Dadurch, dass die ganzen Funktionalitäten für die Darstellung und die Schnittstelle zu 3DCOV erhalten bleibt, ist keine Neuprogrammierung erforderlich. Jedoch ist die Komplexität der Java 3D API nicht zu unterschätzen. Die Entkopplung erfordert ein gutes Verständnis der Library und den Zugriff in die tiefen des Computers.

5.3.4. LWJGL entkoppeln inkl. Schnittstelle zu 3DCOV

Zusätzlich zur letzten Variante muss hier noch eine Kopplung zu 3DCOV erstellt werden. Dadurch entsteht wieder Programmieraufwand. Möglicherweise ist jedoch die LWJGL leichter verständlich als die Java 3D API und deshalb leichter zu entkoppeln.

6. Beurteilung der Lösungsvarianten nach Prototyping

6.1. Ersetzen der Java 3D API durch Eigenprogrammierung

Obwohl die Struktur, die von 3DCOV durch die Verwendung der Java 3D API, vorgegeben wird, nicht optimal ist, konnte in dieser Variante ein Prototype erstellt werden. Die Klassen, die in der Java 3D API von 3DCOV angesprochen werden, konnten so umfunktioniert werden, dass ein Zeichnen der Objekte auf dem Canvas über die Java 2D API erreicht wurde. Die Projektion und die Rotation konnte mittels einfacher geometrischer Berechnung umgesetzt werden.

Die Systembelastung durch die ständige Neuberechnung ist hoch und bei nicht sehr leistungsfähigen Computern kann es schnell zu einem Flackern kommen, wenn die Körper bewegt werden. 3DCOV's Priorität ist jedoch nicht die Bewegung an sich, sondern die statische Ansicht aus verschiedenen Richtungen. Deshalb muss das Konstrukt nach dem Stoppen der Rotation sauber dargestellt werden. Während der Rotation bzw. Verschiebung ist die Anzeige weniger bis gar nicht relevant.

Für moderne Geräte, mit teilweise sogar Multi-Core-Prozessoren, wird jedoch auch dies kein Problem sein, soweit der Computer nicht durch andere Prozesse bereits ausgelastet ist.

Die Implementierung der weiteren Funktionalitäten von 3DCOV sollte kein Problem in Bezug auf die Belastung bereiten. Der Hauptbestandteil des Konstrukts sind die Objekte und Klassen. Es wird nicht erwartet, dass das Zeichnen der Linien (Assoziationen, Vererbung etc) höhere Belastung verursacht.

Nach dem ersten Analysieren, wie die Schnittstellen für die weiteren Funktionalitäten von 3DCOV vorgegeben werden, ist zu erwarten, dass nicht alle Klassen auf die Umsetzung auf Java 2D API passend sein werden. Dies bedeutet, dass Methoden unter Umständen leer implementiert werden, weil die Aufgabe in einer anderen Funktion bzw. Klasse ausgeführt werden muss.

Weiter ist zu erwarten, dass gewisse Details in der Darstellung nicht umgesetzt werden können oder der Aufwand für eine Umsetzung unverhältnismässig wäre. Dazu gehört die bis jetzt erkannte Unterscheidung, wie weit eine Seite eines Objekts dem Betrachter zugewendet ist und entsprechend heller oder dunkler dargestellt wird. Dadurch, dass solche Details nicht implementiert werden, wird die Übersichtlichkeit oder die Aussagekraft der Darstellung nicht beeinträchtigt.

Die Umsetzung der kompletten Funktionalitäten von 3DCOV innerhalb des Projekts erscheint nach dieser Evaluationsphase mit dieser Lösungsvariante sehr wahrscheinlich.

6.2. Ersetzen der Java 3D API und des Packages "cov3dmodel" im 3DCOV durch Eigenprogrammierung

Da vom Kunden eine Veränderung der 3DCOV-Applikation nicht erwünscht ist und nur im äussersten Fall erfolgen darf, wurde diese Variante nicht weiterverfolgt. Dies wurde auch dadurch unterstützt, dass die erste Variante ohne Änderung des Packages „cov3dmodel" soweit erfolgreich war.

6.3. Java 3D API entkoppeln

Dies war die zweite Variante, an der intensiv gearbeitet wurde. Mit über 800 Klassen und davon 400 Klassen innerhalb eines Packages, Methoden, die mehrere hundert, eine bis tausend Zeilen Code lang sind, war der Anfang schwer zu finden. Zusätzlich machte das Multi-Threading, welches in dieser API intensiv angewendet wurde, die Aufgabe nicht einfacher.

Java 3D API baut auf OpenGL beziehungsweise Direct3D auf und benötigt deshalb eine Installation, unterschiedlich je nach Betriebssystem. Das Ziel war nun, diese Abhängigkeit zu eliminieren und allenfalls bereits aufbereitete Objekte und Bilder mittels Java 2D darstellen zu lassen.

Wenn eine Applikation gestartet wird, die auf Java3D aufbaut, werden zuerst einige Systemproperties übers Betriebssystem bzw. von der Grafikkarte an die API geliefert. Abhängig von diesen Properties wird das Rendering unterschiedlich durchgeführt. Dies erklärt ein Teil der Komplexität vieler ausufernden Methoden.

Eine Java 3D Applikation enthält zusätzlich zu den Threads, von der JVM gestartet, noch einige weitere Threads. Zu diesen gehören unter anderem der Rendering-Thread und mehrere Update-Threads. Der MasterControl-Thread startet die anderen Threads und verwaltet das Messaging, welches die Kommunikation zwischen den Threads ermöglicht. Wie der Update-Threads bereits impliziert, wird das Rendering auf die Veränderung der Objekte ausgeführt und dabei sehr detailliert ermittelt, was wirklich neu gezeichnet werden muss.

Im CanvasViewCache wird die Projektion auf 2D berechnet. Dies passiert aus der Methode „updateViewCache" im Canvas3D heraus, welche durch den RenderBin, den Renderer oder GraphicsContext3D aufgerufen werden kann. Im RenderBin wird aufgeführt, welche Rendering-Operationen ausgeführt werden müssen. Der Renderer, der einen Loop über „doWork" ausführt, unterscheidet nach Mode, Request, diverse Einstellungen und rendert entsprechend unterschiedlich.

Der GraphicsContext3D wird nur im Immediate Mode genutzt. Beim Immediate Mode kann genauer bestimmt werden, was wann gezeichnet wird. In der 3DCOV-Appplikation wird hauptsächlich der Retained Mode benutzt. Dies ist in diesem Fall auch sinnvoll.

Die vielen Threads, die von Java 3D erzeugt werden, beanspruchen viel Prozessorzeit. Die Unterteilung in die Threads macht vor allem Sinn, wenn mehrere Prozessoren damit

angesprochen werden können, was bei Einbezug der Grafikkarte ausgenutzt werden kann. Bei einer Implementierung von 3DCOV auf Java 2D würde dies jedoch nur Performanz-Probleme generieren.

Innerhalb Java 3D werden die 3D-Objekte in OpenGL-Strukturen umgewandelt. Dies passiert in den Retained-Objekten, welche immer parallel zu den vom Programmierer angesprochenen Klassen gehalten werden. In GeometryRetained werden die einzelnen Geometry-Typen aufgeführt. Die Weitergabe an die Schicht unter Java 3D erfolgt demzufolge weder in Java-3D-Objekten noch in einer 2-dimensionalen Version, sondern in einer Form dazwischen, die für OpenGL verwendbar ist.

Java 3D API wurde für die Spielentwicklung erstellt. Dies ist in der ganzen Struktur und in der ganzen Komplexität dieser API bemerkbar. Die Architektur ist nicht dafür gebaut, auf Java 2D umgeleitet zu werden. Dafür sind die Schritte, die erfolgen, bevor ein Objekt dargestellt wird, zu sehr darauf ausgerichtet, möglichst effizient bzw. performant darzustellen und von der darunterliegenden Schnittstelle abhängig.

Diese Lösungsvariante wird deshalb nicht weiterverfolgt.

6.4. LWJGL entkoppeln inkl. Schnittstelle zu 3DCOV

LWJGL bietet keine bereits fertigen geometrischen Objekte wie dies von 3DCOV gefordert ist und von Java 3D bereitgestellt wird. Deshalb wurde diese Variante nicht weiterverfolgt.

7. Entscheid für „Ersetzen der Java 3D API durch Eigenprogrammierung“

Bis zu diesem Zeitpunkt wurde für die Variante „Ersetzen der Java 3D API durch Eigenprogrammierung“ ein komplettes Proof Of Concept erstellt. Das Risiko, dass grössere Probleme bei der weiteren Umsetzung auftauchen, ist klein. Die Aussichten sind gut, dass bis Ende des Projektes ein wieder komplett funktionierendes 3DCOV abgeliefert werden kann.

8. Analyse einzelner 3D Libraries

8.1. Irrlicht Engine

Beschreibung

The Irrlicht Engine is an open source high performance realtime 3D engine written and usable in C++ and also available for .NET languages. It is completely cross-platform, using D3D, OpenGL and its own software renderer, and has all of the state-of-the-art features which can be found in commercial 3d engines. Quelle http://irrlicht.sourceforge.net/

Relevanz für Projekt

Diese Library ist nicht für dieses Projekt verwendbar, da sie nicht mit Java benutzt werden kann.

8.2. JMonkey Engine

Beschreibung

jMonkeyEngine (jME) is a game engine made especially for modern 3D development, as it uses shader technology extensively. jMonkeyEngine is written purely in Java and uses LWJGL as its default renderer. OpenGL 2 through OpenGL 4 is fully supported.

Quelle: http://en.wikipedia.org/wiki/JMonkey_Engine

Requirement: Operating system Mac OS X, Windows, Linux, or Solaris

Quelle: http://jmonkeyengine.org/downloads/

Relevanz für Projekt

Da diese Library nicht komplett plattformunabhängig ist, ist sie für das Projekt nicht relevant.

8.3. Mobile 3D Graphics API (M3G; JSR-184)

Beschreibung

The Mobile 3D Graphics API, commonly referred to as M3G, is a specification defining an API for writing Java programs that produce 3D computer graphics. It extends the capabilities of the Java ME, a version of the Java platform tailored for embedded devices such as mobile phones and PDAs.

Quelle: http://en.wikipedia.org/wiki/Mobile_3D_Graphics_API

Relevanz für Projekt

Diese API ist für Mobiltelefone gemacht. Es gibt eine Desktop-Implementation Java-M3G, allerdings nur für Ubuntu und Android. Quelle: http://code.google.com/p/java-m3g/

8.4. JT Open from Siemens PLM Software

Beschreibung

JT is a 3D data format developed by Siemens PLM Software (formerly UGS Corp.) and is used for product visualization, collaboration, and CAD data exchange. It can contain any combination of approximate (faceted) data, exact boundary representation surfaces (NURBS), Product and Manufacturing Information (PMI), and Metadata (textual attributes) either exported from the native CAD system or inserted by a product data management (PDM) system. It is probably the most widely used 3D visualization format for discrete manufacturing with over 4,000,100 JT-enabled licenses of software in use. On 2009 September 18 the ISO stated officially that the JT specification has been accepted for publication as an ISO Publicly Available Specification (PAS).

Quelle: http://en.wikipedia.org/wiki/JT_%28visualization_format%29

To view the JT showcase you must have JT2Go installed on your computer.

Quelle:
http://www.plm.automation.siemens.com/en_sg/products/open/jtopen/technology/jt_showcase.shtml

Relevanz für Projekt

Da eine JT2Go Installation notwendig ist, um diese Library zu benutzen, ist diese für das Projekt nicht relevant.

8.5. Nvidia Scene Graph (NVSG)

Beschreibung

The SceniX scene management engine is an established solution in its sixth release as a cross-platform, object-oriented programming solution for software developers to quickly create interactive OpenGL applications having the highest degree of interactive performance and realism. Unlike most scene graphs, SceniX was designed around shader usage and renderer independence for applications requiring high image quality and the flexibility to share shaders via CgFX. SceniX also includes mature support for interactive ray tracing using OptiX, and the RTFx ray tracing effect interchange format.

Quelle: http://www.nvidia.de/object/scenix-home.html

Relevanz für Projekt

Benutzt OpenGL und ist daher nicht relevant für das Projekt.

8.6. OGRE

Beschreibung

OGRE (Object-Oriented Graphics Rendering Engine) is a scene-oriented, flexible 3D rendering engine (as opposed to a game engine) written in C++ designed to make it easier and intuitive for developers to produce applications utilizing hardware-accelerated 3D graphics. The class library abstracts the details of using the underlying system libraries like Direct3D and OpenGL and provides an interface based on world objects and other high level classes.

Quelle: http://en.wikipedia.org/wiki/OGRE

Relevanz für Projekt

Da in C++ geschrieben und da es Direct3D und OpenGL benutzt ist OGRE nicht plattformunabhängig und darum nicht relevant für das Projekt.

8.7. OpenGL Performer

Beschreibung

OpenGL Performer, formerly known as IRIS Performer and commonly referred to simply as Performer, is a commercial library of utility code built on top of OpenGL for the purpose of enabling hard real-time visual simulation applications. OpenGL Performer was developed by SGI which continues to maintain and enhance it. OpenGL Performer is available for IRIX,

Linux, and several versions of Microsoft Windows. Both ANSI C and C++ bindings are available.

Quelle: http://en.wikipedia.org/wiki/OpenGL_Performer

Relevanz für Projekt

Da nicht plattformunabhängig nicht relevant für das Projekt.

8.8. OpenSceneGraph

Beschreibung

OpenSceneGraph is an open source 3D graphics application programming interface,[1] used by application developers in fields such as visual simulation, computer games, virtual reality, scientific visualization and modeling. The toolkit is written in standard C++ using OpenGL,[1] and runs on a variety of operating systems including Microsoft Windows, Mac OS X, Linux, IRIX, Solaris and FreeBSD.

Quelle: http://en.wikipedia.org/wiki/OpenSceneGraph

Relevanz für Projekt

Da nicht plattformunabhängig nicht relevant für das Projekt.

8.9. OpenSG

Beschreibung

OpenSG is a scene graph system to create real-time graphics programs, e.g. for virtual reality applications. It is developed following Open Source principles, LGPL licensed, and can be used freely. It runs on Microsoft Windows, Linux, Solaris and Mac OS X and is based on OpenGL. Quelle: http://en.wikipedia.org/wiki/OpenSG

Relevanz für Projekt

Da nicht plattformunabhängig nicht relevant für das Projekt.

8.10. QSDK

Beschreibung

The QSDK is a streaming scene graph retained-mode Application Programming Interface (API) combined with a cell-portal system to connect scene graphs. Audio and animation are fully supported. It is available on Macintosh, Playstation 2 and Xbox platforms, and for free on PC platforms.

Quelle: http://en.wikipedia.org/wiki/QSDK

Relevanz für Projekt

Da nicht plattformunabhängig, nicht relevant für das Projekt.

8.11. Quesa

Beschreibung

Quesa is a high-level, Open Source, 3D graphics library that offers binary and source level compatibility with Apple's QuickDraw™ 3D API. Quesa does not contain any Apple source code, and was developed without access to Apple's QD3D implementation. Quesa has some APIs and capabilities that were not present in QD3D.

Quelle: http://www.quesa.org/

Relevanz für Projekt

Da nicht plattformunabhängig nicht relevant für das Projekt.

8.12. Vega Prime

Beschreibung

Vega Prime, with its cross-platform scalable environment is the most productive COTS visualization tool for real-time 3D development and deployment of simulation applications. Ideal for both high-performance and low cost hardware platforms, Vega Prime's extensible plug-in architecture facilitates the rapid design and prototyping of real time 3D applications by utilizing the most sophisticated technology available, within an easy-to-use toolkit.

Quelle: http://www.presagis.com/products_services/products/ms/visualization/vega_prime/

Relevanz für Projekt

Nicht kostenlos, daher nicht relevant für das Projekt.

8.13. VR-Vantage

Beschreibung

VR-Vantage XR is VT MÄK's premier battlefield visualization tool.

Quelle: http://www.mak.com/products/visualize/common-operating-picture.html

Relevanz für Projekt

Da für Kämpfe gedacht, nicht relevant für das Projekt.

8.14. Copperlicht

Beschreibung

CopperLicht is a WebGL library and JavaScript 3D engine for creating games and 3d applications in the webbrowser. It uses the WebGL canvas supported by modern browsers and is able to render hardware accelerated 3d graphics without any plug-in.

Quelle: http://www.ambiera.com/copperlicht/

Relevanz für Projekt

Da für Webbrowser nicht relevant für das Projekt.

8.15. O3D

Beschreibung

O3D is an open source (BSD license) JavaScript API[2] created by Google for creating interactive 3D graphics applications that run in a web browser window or in a XUL desktop application.

Quelle: http://en.wikipedia.org/wiki/O3D

Relevanz für Projekt

Da für Webbrowser nicht relevant für das Projekt.

8.16. Away3D

Beschreibung

Away3D is an open source 3D graphics engine, written for the Adobe Flash platform in ActionScript 3, and runs in modern web browsers that utilize Adobe Flash Player.

Quelle: http://en.wikipedia.org/wiki/Away3D

Relevanz für Projekt

Da Adobe Flash Player benötigt nicht relevant für das Projekt.

8.17. X3D

Beschreibung

Extensible 3D, kurz X3D, ist eine auf XML (XML-Encoding, Datei-Endung .x3d) basierende Beschreibungssprache für 3D-Modelle, die in einem Webbrowser angezeigt werden können. Quelle: http://de.wikipedia.org/wiki/X3D

Relevanz für Projekt

X3D dient für Anzeige 3D-Objekte im Webbrowser. Dies ist nicht, das was gesucht wird.

8.18. Panda3D

Beschreibung

Panda3D is a game engine, a framework for 3D rendering and game development for Python and C++ programs. Quelle: http://www.panda3d.org/

Relevanz für Projekt

Panda3D ist eine Game Engine für Spiel-Entwicklung in Python und C++. Dies passt nicht.

8.19. Simple DirectMedia Layer

Beschreibung

SDL is written in C, but works with C++ natively, and has bindings to several other languages, including Ada, C#, D, Eiffel, Erlang, Euphoria, Go, Guile, Haskell, Java, Lisp, Lua, ML, Objective C, Pascal, Perl, PHP, Pike, Pliant, Python, Ruby, Smalltalk, and Tcl.

SDL supports Linux, Windows, Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX. The code contains support for AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS, and OS/2, but these are not officially supported. Quelle: http://www.libsdl.org/

Relevanz für Projekt

Die Library wird für verschiedene Betriebssysteme zur Verfügung gestellt und eine dem Betriebssystem entsprechende Version muss geladen und installiert werden. Dies entspricht nicht den Anforderungen des Projekts.

8.20. SFML

Beschreibung

SFML is a free multimedia C++ API that provides you low and high level access to graphics, input, audio, etc. Quelle: http://www.sfml-dev.org/index.php

Relevanz für Projekt

Die Library benötigt unterschiedliche Version pro Betriebssystem. Dies entspricht nicht den Anforderungen des Projekts.

8.21. emWin

Beschreibung

emWin is designed to provide an efficient, processor- and LCD controller-independent graphical user interface (GUI) for any application that operates with a graphical LCD.

It is compatible with single-task and multitask environments, with a proprietary operating system or with any commercial RTOS. emWin is shipped as "C" source code.
It may be adapted to any size physical and virtual display with any LCD controller and CPU. Quelle: http://www.segger.com/cms/emwin.html

Relevanz für Projekt

Diese Library ist spezialisiert auf Embedded Software, entspricht nicht dem, was gesucht wird.

8.22. Open Inventor

Beschreibung

Open Inventor ist eine freie objektorientierte C++-Programmbibliothek zur Erstellung von 3D-Grafiken unter Verwendung von OpenGL. Quelle: http://de.wikipedia.org/wiki/Open_Inventor

Relevanz für Projekt

Betriebssystemunabhängigkeit ist nicht gegeben.

8.23. Openskia

Beschreibung

This vector graphics rendering software makes highend visual effects possible on feature phones. It is tiny in size and is capable of delivering very high quality. Skia's engine is the graphics core of both Google Android and Google Chrome. Quelle: http://code.google.com/p/openskia/

Relevanz für Projekt

Dies entspricht nicht den Anforderungen, da nicht plattformunabhängig.

8.24. WebGL

Beschreibung

WebGL (Web-based Graphics Library) is a software library that extends the capability of the JavaScript programming language to allow it to generate interactive 3D graphics within any compatible web browser. WebGL code executes on a computer display card's Graphics Processing Unit (GPU), which must support shader rendering. Quelle: http://en.wikipedia.org/wiki/WebGL

Relevanz für Projekt

Die Library ist für Web Browser und greift auf Graphik-Hardware zu; plattformunabhängigkeit ist nicht gegeben.

8.25. MiniGL

Beschreibung

The term **MiniGL** was applied to a wide range of incomplete OpenGL implementations provided by graphics card hardware companies including 3dfx, PowerVR and Rendition in the late 1990s. They owe their genesis to the computer game *Quake*. Quelle: http://en.wikipedia.org/wiki/MiniGL

Relevanz für Projekt

Plattformunabhängigkeit ist nicht gegeben.

8.26. Direct3D

Beschreibung

Direct3D is part of Microsoft's DirectX application programming interface (API). Quelle: http://en.wikipedia.org/wiki/Microsoft_Direct3D

Relevanz für Projekt

Diese Library ist Microsoft-spezifisch und deshalb nicht relevant für das Projekt.

8.27. X3D

Beschreibung

X3D is a royalty-free open standards file format and run-time architecture to represent and communicate 3D scenes and objects using XML. Quelle: http://www.web3d.org/about/overview/

Relevanz für Projekt

Diese Library dient hauptsächlich zum Austausch von 3d-Objekten. Dies entspricht nicht den Anforderungen.

8.28. OpenGL

Beschreibung

OpenGL is a low-level, procedural API, requiring the programmer to dictate the exact steps required to render a scene. This contrasts with descriptive (aka scene graph or retained mode) APIs, where a programmer only needs to describe a scene and can let the library manage the details of rendering it. OpenGL's low-level design requires programmers to have a good knowledge of the graphics pipeline, but also gives a certain amount of freedom to implement novel rendering algorithms. Quelle: http://en.wikipedia.org/wiki/OpenGL

To program using the OpenGL API, you need the driver and the development package (depends on platform and programming language). More platform-specific details are described in the sections below. Quelle: http://www.opengl.org/wiki/Getting_started

Relevanz für Projekt

Diese Library greift direkt auf Hardware zu. Dafür müssen die entsprechenden Grafik-Treiber installiert sein. Dies kann aber für die 3DCOV-Applikation nicht vorausgesetzt werden.

8.29. RenderMan

Beschreibung

The RenderMan Interface Specification,[1] or *RISpec* in short, is an open API developed by Pixar Animation Studios to describe three-dimensional scenes and turn them into digital photorealistic images. It includes the RenderMan Shading Language

The RenderMan shading language allows material definitions of surfaces to be described not only by adjusting a small set of parameters, but in an arbitrarily complex fashion by using a C-like programming language to write shading procedures commonly known as procedural textures and shaders.

Quelle: http://en.wikipedia.org/wiki/RenderMan_Interface_Specification

Relevanz für Projekt

Die Library ist zu komplex für das Projekt und benötigt eine Komplett-Installation.

8.30. RenderWare

Beschreibung

RenderWare was widely cross-platform: It ran on Windows as well as Apple Mac OS X-based applications and many video game consoles such as Nintendo GameCube, Wii, Xbox, Xbox 360, PlayStation 2, PlayStation 3, and PlayStation Portable.

RenderWare is no longer available for purchase, although EA still honors old contracts, meaning that external developers who licensed the technology before the Criterion acquisition may still use the technology. What was RenderWare 4 has dissolved into the rest of EA internal tech.

Quelle: http://en.wikipedia.org/wiki/RenderWare

Relevanz für Projekt

RenderWare hätte man kaufen müssen, ist aber nicht mehr erhältlich. Daher ist sie für dieses Projekt nicht relevant.

8.31. GlideAPI

Beschreibung

Glide is based on the basic geometry and "world view" of OpenGL. Quelle: http://en.wikipedia.org/wiki/Glide_API

Relevanz für Projekt

Diese Library baut auf OpenGL auf und benötigt deshalb zusätzliche Installationen.

8.32. ClanLib

Beschreibung

ClanLib is a cross-platform C++ game SDK, currently supporting Microsoft Windows, Linux and Mac OS X. It has full hardware accelerated graphics support through OpenGL, and also a software renderer. ClanLib also helps in playing sound, using the Vorbis or MikMod libraries, and has classes for collision detection, GUIs, XML, networking, and other things that may be helpful to a game programmer.

Quelle: http://en.wikipedia.org/wiki/ClanLib

Relevanz für Projekt

Diese Library ist nicht betriebssystemunabhängig.

8.33. Crystal Space

Beschreibung

Crystal Space is a framework for developing 3D applications written in C++ by Jorrit Tyberghein and others. It is typically used as a game engine but the framework is more general and can be used for any kind of 3D visualization. It is very portable and runs on Microsoft Windows, GNU/Linux, UNIX, and Mac OS X

Quelle: http://en.wikipedia.org/wiki/CrystalSpace

Relevanz für Projekt

Diese Library ist eine C++ Library und nicht komplett plattformunabhängig und daher nicht relevant für das Projekt.

8.34. HOOPS 3D Graphics System

Beschreibung

The HOOPS 3D Graphics System is a 3D Graphics API, part of The HOOPS 3D Application Framework. The HOOPS 3D Application Framework has three main elements the first of which is the core HOOPS 3D Graphics System itself, a 3D scene-graph API. The second is a rendering pipeline which can drive a number of lower-level API's including OpenGL and Direct3D. Finally, a layer of "application-level" functionality sits atop the scene-graph.

Quelle: http://en.wikipedia.org/wiki/HOOPS_3D_Graphics_System

HOOPS/3dGS automatically defines a segment named "/driver". Underneath this segment are a dozen or more pre-created sub-segments- one segment for each kind of device for which HOOPS/3dGS has a device driver.

Quelle:
http://developer.hoops3d.com/documentation/Hoops3DGS/tech_overview/TechnicalOverview_C.html#_Toc470077868

Relevanz für Projekt

Diese Library ist eine C++ Library. Ausserdem benötigt sie Treiber. Aus diesem Grund ist diese Library nicht relevant für das Projekt.

8.35. Java 3D

Beschreibung

Java 3D wurde seit 1997 von Sun Microsystems entwickelt. Die Version 1.0 erschien im Dezember 1998. Die Version 1.4 ist seit März 2006 verfügbar, als wichtiges Leistungsmerkmal ist hier die Möglichkeit der Shader-Programmierung für aktuelle Grafikhardware hervorzuheben. Aktuell ist die Version 1.5, die unter anderem die Rendering-Pipeline Jogl auf allen Plattformen einführt.

Quelle: http://de.wikipedia.org/wiki/Java_3D

Relevanz für Projekt

Java 3D benötigt eine Installation, die abhängig ist vom Betriebssystem bzw. von der Grafikkarte.

Die Library könnte noch als Vorlage für eine auf Java 2D aufbauende Library dienen, indem man die Komponenten, die auf die Grafikkarte zugreifen, eliminiert.

8.36. JOGL

Beschreibung

Java Bindings for OpenGL (JOGL) ist ein im Jahr 2003 durch die Zusammenarbeit von Sun Microsystems und SGI geschaffenes Open-Source Projekt. Ursprünglich wurde es von Kenneth Russell und Chris Kline begonnen. Ziel von JOGL war es, dass so die Spielindustrie auf die Programmiersprache Java bei Spielentwicklungen zugreift, da es vorher keine Unterstützung für OpenGL gab.

Quelle: http://de.wikipedia.org/wiki/Jogl

Relevanz für Projekt

Ist eine Binary Bibliothek. Der Endbenutzer müsste jeweils die Native Library für sein Betriebsystem installieren. Aus diesem Grund ist diese Library nicht relevant für das Projekt.

Quelle: http://www.jogl.info/installation.htm

8.37. GL4Java

Beschreibung

OpenGLTMfor JavaTM maps the complete OpenGLTM 1.3 API and the complete GLU 1.2 API to JavaTM and integrates all managment functions, while using the JavaTM-Native-Interface (JNI) and the JDirect-Interface of MSTM-JVM. OpenGLTMfor JavaTM uses the native OpenGLTM library of the underlying operating System !

Quelle:http://jausoft.com/gl4java.html

Relevanz für Projekt

Ist eine Binary Bibliothek. Der Endbenutzer müsste jeweils die Native Library für sein Betriebsystem installieren. Aus diesem Grund ist diese Library nicht relevant für das Projekt.

Quelle: http://jausoft.com/products/gl4java/gl4java_install.html

8.38. LWJGL

Beschreibung

LWJGL is not meant to make writing games particularly easy; it is primarily an enabling technology which allows developers to get at resources that are simply otherwise unavailable or poorly implemented on the existing Java platform. We anticipate that the LWJGL will, through evolution and extension, become the foundation for more complete game libraries and "game engines" as they have popularly become known, and hide some of the new evils we have had to expose in the APIs. Quelle: http://lwjgl.org

Im Source-Code der heruntergeladenen Libraries sind Native-Klassen enthalten unterteilt nach Betriebssystemen.

Relevanz für Projekt

Durch die Unterscheidung nach Betriebssystemen geht die Garantie verloren, dass bei Änderungen der Betriebssysteme bzw. Graifkarten die Library weiterhin funktioniert. Deshalb kann die Library für das Projekt nicht weiter berücksichtigt werden.

Die Library könnte noch als Vorlage für eine auf Java 2D aufbauende Library dienen, indem man die Komponenten, die auf die Grafikkarte zugreifen, eliminiert.

8.39. PCL

Beschreibung

Point Cloud Library (PCL) runs on many operating systems, and prebuilt binaries are available for Linux, Windows, and Mac OS X. In addition to installing PCL, you will need to download and compile a set of 3rd party libraries that PCL requires in order to function. Select the operating system of your choice below to continue. If your platform is not supported, please contact us. Quelle: http://pointclouds.org/downloads/

PCL ist in C++ programmiert.

Relevanz für Projekt

Eine Installation, vom Betriebssystem abhängig, muss ausgeführt werden. Dies ist entspricht nicht den Anforderungen.

Projekt: J3DEval

Anforderungsspezifikation

Version: 4.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
03.10.2011	1.0	Erstellung des Dokuments. Erste Version der Kapitel 1-4 geschrieben.	Lara Mühlemann
10.10.2011	1.0	Schnittstellen in Dokument integriert	Lara Mühlemann
25.10.2011	2.0	Kapitel Use Cases Struktur erstellt + Use Case brief von UC01 geschrieben.	Lara Mühlemann
30.10.2011	2.0	Im Kapitel Use Cases gearbeitet	Lara Mühlemann
31.10.2011	2.0	Use Cases umstrukturiert	Lara Mühlemann, Marion Walser
18.11.2011	3.0	Use Cases weitergeführt	Marion Walser
28.11.2011	3.0	Fehler im Kapitel Schnittstelle korrigiert (Klasse Leaf ist nicht gefordert, darum gelöscht, public Variabeln ergänzt, die Exception beim XML-Import werfen, wenn sie fehlen.).	Lara Mühlemann
12.12.2011	3.0	Use Cases weitergeführt	Marion Walser
16.12.2011	3.0	XML-Schnittstelle erwähnt	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Lara Mühlemann verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
12.10.2011	Marion Walser
17.10.2011	Lara Mühlemann
01.11.2011	Marion Walser + Lara Mühlemann
28.11.2011	Lara Mühlemann
19.12.2011	Lara Mühlemann

1.4. Inhalt

2. Einführung

2.1. Zweck

In diesem Dokument werden die funktionalen und nicht funktionalen Anforderungen an die Library focusedj3d_on_j2d aufgeführt. Diese dienen als Grundlage für die weiteren Design- und Programmierarbeiten in diesem Projekt.

2.2. Gültigkeitsbereich

Dieses Dokument ist während der Projektdauer gültig.

2.3. Definitionen und Abkürzungen

Siehe Glossar.doc

2.4. Referenzen

[1] Evaluationsdokumentaion.doc

3. Allgemeine Beschreibung

3.1. Produktperspektive

Focusedj3d_on_j2d wird als Studienarbeit an der HSR erstellt. Es soll die Java3D Library des Produkts 3DCOV ablösen.

3.2. Produkt Funktionen

Die Applikation 3DCOV soll nach der Ablösung der Java 3D Library dieselben Informationen darstellen können wie vorher.

In der Abbildung 1 ist die ursprüngliche Version der 3D-Darstellung von 3DCOV ersichtlich. Nach der Ablösung soll die Darstellung wieder gleich sein.

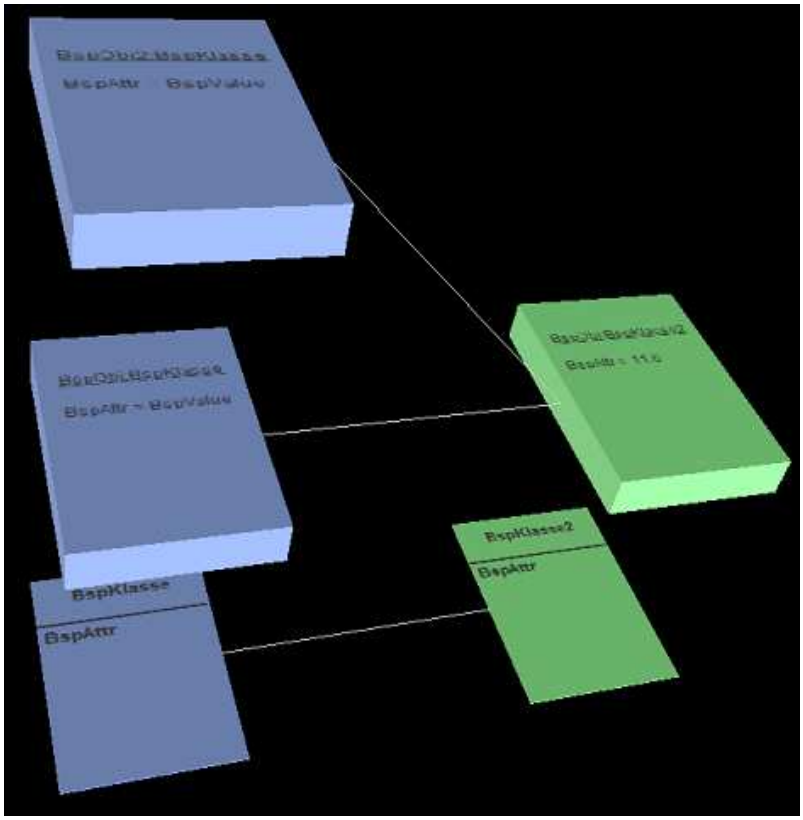


ABBILDUNG 1: ALTE VERSION DER 3D-DARSTELLUNG VON 3DCOV

Dies beinhaltet:

- Darstellung der Klassen als Rechteck. Ersichtlich sind der Klassenname und die Attribute.
- Darstellung der Objekte. Diese liegen senkrecht über der Klassendarstellung und werden als Quader dargestellt. Es werden der Objektname und die Attribute inklusive deren Wert abgebildet.
- Es sind sowohl die Beziehungen zwischen den Klassen sowie die Beziehungen zwischen den Objekten ersichtlich.
- Das Konstrukt ist in horizontale und in vertikale Richtung drehbar.
- Es kann gezoomt werden.
- Es kann verschoben werden.
- Die Klassen, Objekte und Beziehungen können selektiert werden.

3.3. Benutzer Charakteristik

Die Library focusedj3d_on_j2d, die in J3DEval entwickelt wird, ist speziell für die 3DCOV Applikation. 3DCOV ist daher das einzige Zielpublikum.

3.4. Einschränkungen

Die Applikation soll unabhängig von der darunterliegenden Hardware sein.

Aufgrund der Eingrenzung des Zielpublikums (in Unterkapitel ‚Benutzer Charakteristik' ersichtlich) werden nicht alle Funktionalitäten von Java3D umgesetzt, sondern lediglich die von der Applikation 3DCOV benötigten.

3.5. Annahmen

Keine Annahmen vorhanden.

3.6. Abhängigkeiten

Da die Applikation 3DCOV in Java geschrieben wurde, wird focusedj3d_on_j2d ebenfalls in Java geschrieben. Für die Ausführung der Applikation wird eine Java Virtual Machine und Java Runtime Environment mit Java 2D vorausgesetzt.

4. Spezifische Anforderungen

4.1. Zuverlässigkeit

Es wurden keine Angaben über die Zuverlässigkeit gemacht.

4.2. Leistung

Zugunsten der Plattformunabhängigkeit wird bewusst auf eine gute Leistung verzichtet. Bewegungen müssen nachvollziehbar sein, dürfen aber ruckeln. Um die Bewegungen fliessender zu gestalten, darf auch auf Detailzeichnungen wie z.B. Text während der Bewegung verzichtet werden. Es ist möglich, dass während der Bewegung nur die Konturen und erst bei Stillstand der Text wieder angezeigt werden.

4.3. Wartbarkeit

Mithilfe der Dokumentationen soll ein Java-Entwickler die Applikation in Betrieb nehmen und weiterentwickeln können.

4.4. Schnittstellen

In diesem Kapitel werden die Schnittstellen beschrieben, welche vom Programm 3DCOV vorausgesetzt werden.

4.4.1. Schnittstelle zwischen 3DCOV und Java 3D API

Nachfolgend wird die Schnittstelle nach der Lösungsvariante „Ersetzen der Java 3D API durch Eigenprogrammierung" gemäss Evaluationsdokumentation [1] dargestellt. Es werden sämtliche Klassen inklusive public-Methoden aufgelistet, welche 3DCOV von der Java 3D API benutzt.

Die Strukturierung der folgenden Auflistung erfolgt gemäss den Packages aus der Java 3D API, die bis anhin von 3DCOV benutzt wurde. Die Package-Namen können jedoch im Verlauf des Projektes J3DEVal ändern sowie die Strukturierung der Klassen.

4.4.1.1. javax.media.j3d

Klasse	Methoden/Konstanten
WakeupCriterion extends WakeupCondition	
WakeupOnAWTEvent extends WakeupCriterion	WakeupOnAWTEvent(int) AWTEvent[] getAWTEvent()
WakeupOr extends WakeupCondition	WakeupOr(WakeupCriterion[])
WakeupCondition	
AmbientLight extends Light	AmbientLight(Color3f)
Light extetends Node	setInfluencingBounds(BoundingSphere)
Background extends Node	Background(Color3f) static final int ALLOW_COLOR_WRITE static final int ALLOW_COLOR_READ setApplicationBounds(Bounds)
BoundingSphere extends Bounds	BoundingSphere(Point3d, double) BoundingSphere(Bounds)
DirectionalLight extends Light	DirectionalLight(Color3f, Vecor3f)
GraphicsContext3D	readRaster(Raster)
Group extends Node	static final int ALLOW_CHILDREN_WRITE static final int ALLOW_CHILDREN_READ static final int ALLOW_CHILDREN_EXTEND removeAllChildren() addChild(Node) removeChild(Node)
Node	setCapability(int) setUserData(Object) getUserData getLocalToVworld(Transform3D) Bounds getBounds()
Bounds	
abstract Behavior extends Node	abstract void initialize() wakeupOn(WakeupCondition)

	abstract void processStimulus(Enumeration) setSchedulingBounds(Bounds)
ImageComponent	static final int FORMAT_RGB
ImageComponent2D	ImageComponent2D(Int, java.awt.image.BufferedImage) Java.awt.image.RenderedImage getImage()
Raster	static final int RASTER_COLOR Raster(Point3f, int, int, int, int, int, ImageComponent2D, Object) ImageComponent2D getImage()
View	setMinimumFrameCycleTime(Long) setBackClipDistance(double) setFrontClipDistance(float)
Canvas3D extends java.awt.Canvas	Canvas3D(java.awt.GraphicsConfiguration) java.awt.Graphics2D getGraphics() postSwap() GraphicsContext3D getGraphicsContext3D() java.awt.Dimension getSize() setCursor(java.awt.Cursor)
ColoringAttributes	setColor(Color3f)
PolygonAttributes	static final int CULL_NONE setCullFace(int)
TriangleFanArray	static final int COORDINATES TriangleFanArray(int, int, int[]) setCoordinates(int, Point3f[])
Texture2D	setEnable(Boolean)
Material	Material(Color3f, Color3f, Color3f, Color3f, float)
Transform3D	get(Vector3d) setTranslation(Vector3d) setRotation(AxisAngle4f) lookAt(Point3d, Point3d, Vector3d) invert() double[] mat; int type; int dirtyBits;

	boolean autoNormalize;
GeometryArray	static final int COORDINATES static final int COLOR_3 static final int ALLOW_COLOR_WRITE setCoordinates(int, Point3f[]) setCoordinates(int, Point3d[]) setCapability(int) setColors(int, Color3f[])
PickInfo	static final int PICK_GEOMETRY static final int NODE static final int CLOSEST_INTERSECTION_POINT Node getNode()
LineArray extends GeometryArray	LineArray(int, int)
Shape3D extends Node	static final int ALLOW_GEOMETRY_WRITE Shape3D(TriangleFanArray, Appearance) Shape3D() setGeometry(LineArray) setAppearance(Appearance) Appearance getAppearance() setUserData(Object)
Appearances	static final int ALLOW_COLORING_ATTRIBUTES_WRITE setLineAttributes(LineAttributes) setColoringAttributes(ColoringAttributes) setTexture(Texture2D) setMaterial(Material) setCapability(int) setPolygonAttributes(PolygonAttributes) PolygonAttributes getPolygonAttributes()
BranchGroup extends Group	static final int ALLOW_DETACH compile()
LineAttributes	static final int PATTERN_DASH static final int PATTERN_USER_DEFINED setPatternMask(int) setLinePattern(int)
TransformGroup extends Group	static final int ALLOW_TRANSFORM_WRITE static final int ALLOW_TRANSFORM_READ

	TransformGroup() TransformGroup(Transform3D) setTransform(Transform3D)

4.4.1.2. javax.vecmath

Klasse	Methoden/Konstanten
Point3d	int x int y int z Point3d(double, double, double) Point3d()
Vector3f	Vector3f(float, float, float)
AxisAngle4f	AxisAngle4f(int, int, int, float)
Color3f	Color3f(java.awt.Color) Color3f(Color3f) Color3f(float, float, float)
Vector3d	double x double y double z Vector3d(double, double, double) Vector3d() set(Point3d)
Point3f	Point3f(float, float, float) float getX() float getY() float getZ() setX(float) setY(float) setZ(float) float x float z float y

4.4.1.3. com.sun.j3d.utils

Klasse	Methoden/Konstanten
geometry.Box extends	Box(float, float, float, int, Appearance)

Primitive	static final int TOP Shape3D getShape(int) setAppearance(Appearance)
geometry.Primitive	static final int GENERATE_TEXTURE_COORDS static final int GENERATE_NORMALS static final int ENABLE_APPEARANCE_MODIFY
geometry.Text2D extends Shape3D	Text2D(String, Color3f, String, int, int)
geometry.Sphere	Sphere(float, Appearance) setPickable(boolean)
images.TextureLoader	TextureLoader(String, java.util.Observer) Texture2d getTexture()
universe.SimpleUniverse	SimpleUniverse(Canvas3D) static GraphicsConfiguration getPreferredConfiguration() viewingPlatform getViewingPlatform() Viewer getViewer() addBranchGraph(BranchGroup) Canvas3D getCanvas()
Universe.Viewer	View getView()
universe.ViewingPlatform	getViewPlatformTransform() setViewPlatformBehavior(Behavior)
picking.PickCanvas extens PickTool	PickCanvas(Canvas3D, BranchGroup) setTolerance(float) setShapeLocation(int, int)
picking.PickIntersection	Point3d getClosestVertexCoordinatesVW() Point3d[] getPrimitiveCoordinatesVW()
picking.PickResult	PickIntersection getClosestIntersection(Point3d) Node getObject()
picking.PickTool	static final int GEOMETRY_INTERSECT_INFO setMode(int) Point3d getStartPosition() PickResult[] pickAllSorted PickResult pickClosest()
pickfast PickCanvas extends pickfast.PickTool	PickCanvas(Canvas3D, BranchGroup)

	setShapeLocation(int, int) setTolerance(float)
pickfast.PickTool	PickInfo pickClosest() PickInfo[] pickAllSorted() setMode(int) setFlags(int)
behaviors.vp.OrbitBehavior extends ViewPlatformAWTBehavior	static final int REVERSE_ALL OrbitBehavior(Canvas3D, int) setRotationCenter(Point3d) double getRotXFactor() double getRotYFactor double getTransXFactor() double getTransYFactor() double getZoomFactor() setTransFactors(double, double) setRotYFactor(double) setRotXFactor(double) setRotFactors(double, double) setZoomFactor(double) setRotateEnable(Boolean) setTranslateEnable(Boolean) setZoomEnable(Boolean) protected processMouseEvent(java.awt.event.MouseEvent) protected processAWTEvents(java.awt.AWTEvent[])
behaviors.vp.ViewPlatformAWTBehavior extends Behavior	static final int KEY_LISTENER setHomeTransform(Transform3D) goHome()
behaviors.mouse.MouseTranslate extends Behavior	setTransformGroup(TransformGroup) setSchedulingBounds(BoundingSphere)

	double getYFactor() setFactor(double, double) setEnable(Boolean)

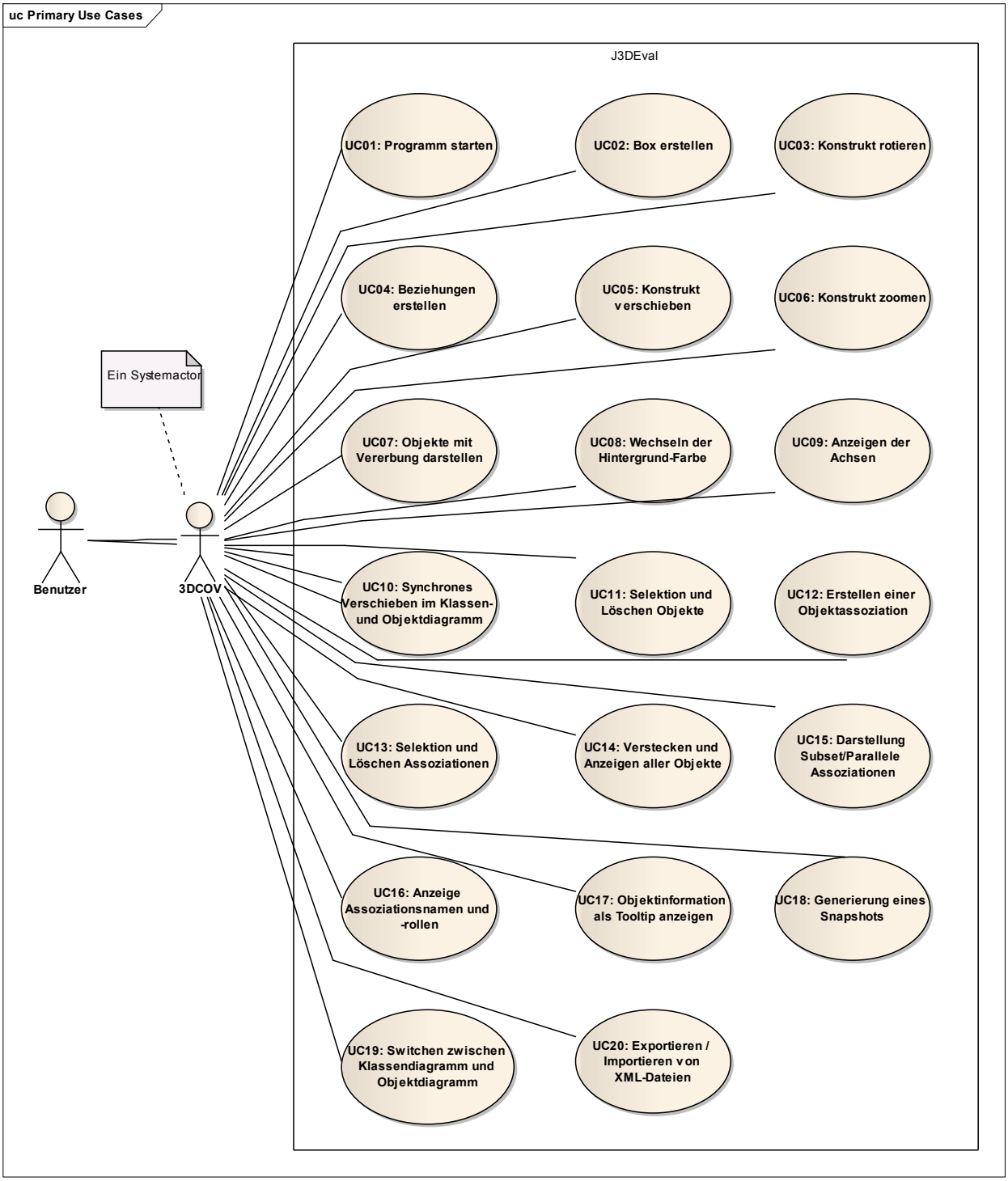
4.4.2. XML-Schnittstelle von 3DCOV

3DCOV stellt eine XML-Schnittstelle zur Verfügung, womit eine Datenspeicherung von den Diagrammen erreicht werden kann.

Es soll weiterhin möglich sein, früher erstellte und abgespeicherte XML-Dateien einzulesen und weiterzubearbeiten.

5. Use Cases

5.1. Use Case Diagram



5.2. Aktoren & Stakeholders

Aktor	Beschreibung
Benutzer	Person, welche das Programm 3DCOV bedient.
3DCOV	Programm, welches die Library focusedj3d_on_j2d benutzt.

5.3. Use Case brief

Use Case	Beschreibung
UC01: Programm starten	**Aus Benutzersicht:** Der Benutzer startet die 3DCOV-Applikation und sieht die erste Ansicht, in dem das Klassendiagramm und das Objektdiagramm noch leer sind. **Aus 3DCOV-Sicht:** Das SimpleUniverse wird aufbereitet. Dies beinhaltet das Erstellen des Canvas, der BranchGroups und die Zuordnung dieser dem SimpleUniverse. Das Canvas wird zusätzlich dem Cov3dModelView hinzugefügt. Weiter werden die Behaviors erstellt. Die Behaviors für das Canvas werden der ViewingPlatform vom SimpleUniverse angehängt. Die Behaviors für die einzelnen Objekte werden der dazugehörigen BranchGroup angehängt.
UC02: Box erstellen	**Aus Benutzersicht:** Der Benutzer erstellt im 3DCOV eine Klasse mit Bezeichnung und Attributen. Sowohl im Klassendiagramm, wie auch im Objektdiagramm erscheint die erstellte Klasse. **Aus 3DCOV-Sicht:** 3DCOV fordert von focusedj3d_on_j2d eine Box an und erstellt ein Image von der Oberansicht der Klasse mit Klassennamen und Attributen. Das Image wird lokal abgespeichert. 3DCOV fügt die Box der ClassBranchGroup hinzu. Danach wird die paint-Methode von Canvas aufrufen, welche die Boxen und das Image mittels Java2D zeichnet.
UC03: Konstrukt rotieren	**Aus Benutzersicht:** Der Benutzer will das Konstrukt nacheinander in horizontale und vertikale Richtung rotieren. **Aus 3DCOV-Sicht:** 3DCOV fordert von focusedj3d_on_j2d das Rotieren des Konstrukts nacheinander in horizontaler und vertikaler Richtung sowie eine folgende Neuzeichnung an.
UC04: Beziehungen erstellen	**Aus Benutzersicht:** Der Benutzer von 3DCOV will die Beziehungen zwischen 3 bestehenden Klassen A, B und C erstellen. Als erstes erstellt er eine Generalisierung. Klasse A erbt von Klasse B. Danach erstellt er eine Assoziation von Klasse B zu Klasse C. Anschliessend

	erstellt der Benutzer eine Dependency von Klasse C zu A. Bei der Dependency wird ein Beziehungsnamen eingegeben. **Aus 3DCOV-Sicht:** 3DCOV erstellt eine Beziehung zwischen den Klassen mittels eines LineArrays und verpackt diese in ein Shape3D. Beides wird bei focusedj3d_on_j2d angefordert. Zusätzlich werden weitere LineArrays von focusedj3d_on_j2d angefordert zum Zeichnen des Beziehungstyp (siehe LineEnd). Beim Erstellen des Beziehungsnamens wird von focusedj3d_on_j2d ein Text2D angefordert. Die Berechnung der Punkte, wo die LineArrays gezeichnet werden müssen, sowie die Position des Text2Ds werden von 3DCOV berechnet. Der LineArray für die Beziehung und der Text2d werden über eine TransformGroup der dazugehörigen BranchGroup angehängt, während die LineArrays für den Beziehungstyp direkt der BranchGroup angehängt wird.
UC05: Konstrukt verschieben	**Aus Benutzersicht:** Der Benutzer verschiebt das Konstrukt in horizontale und vertikale Richtung. **Aus 3DCOV-Sicht:** 3DCOV fordert von focusedj3d_on_j2d das Verschieben des Konstrukts in horizontaler und vertikaler Richtung sowie eine folgende Neuzeichnung an.
UC06: Konstrukt zoomen	**Aus Benutzersicht:** Der Benutzer lässt das Konstrukt grösser bzw. kleiner erscheinen. **Aus 3DCOV-Sicht:** 3DCOV fordert von focusedj3d_on_j2d, dass das Konstrukt grösser bzw. kleiner gezeichnet wird.
UC07: Objekte mit Vererbung darstellen	**Aus Benutzersicht:** Der Benutzer erstellt drei Klassen und je ein dazugehöriges Objekt. Die Klassen stehen in einer Hierarchie zueinander (Generalisierung). Der Benutzer will die Oberklassen der Objekte von ableitenden Klassen in einer Hierarchie von Objekten dargestellt kommen. **Aus 3DCOV-Sicht:** 3DCOV fordert von focusedj3d_on_j2d, bei Erstellung von Objekte von ableitenden Klassen mehrere Boxen aufeinander gemäss Klassenhierarchie zu erstellen. Dies entspricht dem Ablauf gemäss Use Case 2 (3 Boxen erstellen) und wird deshalb weder hier noch in der Domainanalyse weiter ausgeführt.
UC08: Wechseln der Hintergrund-Farbe	**Aus Benutzersicht:** Der Benutzer will die Hintergrund-Farbe von schwarz zu weiss wechseln und umgekehrt. **Aus 3DCOV-Sicht:** 3DCOV fordert ein Background von focusedj3d_on_j2d an, in welchem 3DCOV die Farbe abspeichert. 3DCOV hängt den Background der obersten BranchGroup an. Danach wird ein Neuzeichnen von focusedj3d_on_j2d erwartet.

UC09: Anzeigen der Achsen	**Aus Benutzersicht:** Der Benutzer will die drei Achsen im Raum anzeigen lassen. **Aus 3DCOV-Sicht:** 3DCOV lässt eine BranchGroup und eine TransformGroup für die Achsen erstellen. Weiter werden LineArrays für die Achsen verlangt, die wie bei der Assoziation einem Shape3D zugeordnet werden und der TransformGroup angehängt wird.
UC10: Synchrones Verschieben im Klassen- und Objektdiagramm	**Aus Benutzersicht:** Der Benutzer will das Verschieben einer oder mehreren Klassen im Klassendiagramm auch im Objektdiagramm ausgeführt haben. **Aus 3DCOV-Sicht:** 3DCOV berechnet die Neupositionen der Klassen, Objekte und Linien. Das Neuzeichnen des Objektdiagramm mit den neuen Positionen wird danach bei focusedj3d_on_j2d angefordert.
UC11: Selektion und Löschen Objekte	**Aus Benutzersicht:** Der Benutzer will, dass Objekte im Objektdiagramm selektiert und gelöscht werden können. **Aus 3DCOV-Sicht:** 3DCOV will, dass bei Selektion auf das Objektdiagramm processStimulus() im SelectionBehavior aufgerufen wird. Darin fordert 3DCOV über das PickCanvas von focusedj3d_on_j2d das selektierte Shape3D an. 3DCOV fordert danach BranchGroup, TransformGroup und Sphere an, welche dem Baum angehängt werden und erwartet, dass focusedj3d_on_j2d neuzeichnet.
UC12: Erstellen einer Objektassoziation	**Aus Benutzersicht:** Der Benutzer will, dass er zwischen zwei Objekten im Objektdiagramm eine Assoziation erstellen kann unter der Voraussetzung, dass zwischen den zwei zugrunde liegenden Klassen bereits eine Assoziation besteht. **Aus 3DCOV-Sicht:** 3DCOV fordert BranchGroup, TransformGroup und LineArray für die zu zeichnende Linie an und hängt diese dem Baum an. Darauf soll die Beziehung gemäss Vorgaben von 3DCOV von focusedj3d_on_j2d gezeichnet werden.
UC13: Selektion und Löschen Assoziationen	**Aus Benutzersicht:** Der Benutzer will Assoziationen zwischen Objekten im Objektdiagramm selektieren und löschen können. **Aus 3DCOV-Sicht:** Dies funktioniert gleich wie UC11 „Selektion und Löschen Objekte
UC14: Verstecken und Anzeigen aller Objekte	**Aus Benutzersicht:** Der Benutzer will im Objektdiagramm alle Objekte ausblenden und anzeigen lassen können. **Aus 3DCOV-Sicht:** 3DCOV entfernt die Objekte vom zu zeichnenden Baum und erwartet ein Neuzeichnen von focusedj3d_on_j2d. Wenn die Objekte wieder angezeigt werden sollen, fügt 3DCOV die Objekte dem von focusedj3d_on_j2d zu zeichnenden Baum wieder hinzu.

UC15: Darstellung Subset/Parallele Assoziationen	**Aus Benutzersicht:** Der Benutzer will Restriktionen zwischen zwei Assoziationen erstellen können. **Aus 3DCOV-Sicht:** Dies funktioniert gleich wie UC04 „Beziehungen erstellen"
UC16: Anzeige Assoziationsnamen und -rollen	**Aus Benutzersicht:** Der Benutzer will zu den Assoziationen Bezeichnungen der Beziehungen und der Rollen hinzufügen. **Aus 3DCOV-Sicht:** 3DCOV fordert BranchGroup, TransformGroup und Text2D von focusedj3d_on_j2d an und fügt diese dem Baum hinzufügt. Danach wird ein Neuzeichnen von focusedj3d_on_j2d erwartet.
UC17: Objektinformation als Tooltip anzeigen	**Aus Benutzersicht:** Der Benutzer will, dass ein Tooltipp erscheint, wenn er mit der Maus über ein Objekt im Objektdiagramm fährt. **Aus 3DCOV-Sicht:** 3DCOV erwartet, dass postSwap() im Cov3DCanvas von focusedj3d_on_j2d aufgerufen wird, sobald die Maus über ein Objekt fährt. In dieser Methode zeichnet 3DCOV den Tooltipp an der richtigen Stelle.
UC18: Generierung eines SnapShots	**Aus Benutzersicht:** Der Benutzer will ein Snapshot vom Objektdiagramm erstellen und als separate Datei abspeichern können. **Aus 3DCOV-Sicht:** 3DCOV erwartet, dass ein Snapshot des Canvas in ein BufferedImage abspeichert wird, welches vorher dem ImageComponent2D von focusedj3d_on_j2d mitgegebenen wurde, und über ein Raster von focusedj3d_on_j2d bezogen werden kann.
UC19: Switchen zwischen Klassendiagramm und Objektdiagramm	**Aus Benutzersicht:** Der Benutzer will zwischen den Diagrammen als Totalansicht oder Halbansicht wechseln. **Aus 3DCOV-Sicht:** 3DCOV fordert eine neue TransformGroup an, auf der sie lookAt() aufruft und drei Punkte (eye, center, up) mitgibt. Danach wird invert() auf der TransformGroup des SimpleUniverse von focusedj3d_on_j2d aufgerufen und die TransformGroup der Platform von focusedj3d_on_j2d angehängt. Zusätzlich lässt 3DCOV das Rotationscenter neu berechnen und setzt dieses.
UC20: Exportieren / Importieren von XML-Dateien	**Aus Benutzersicht:** Der Benutzer will sein vorher erstelltes Diagramm exportieren und später wieder importieren können. **Aus 3DCOV-Sicht:** 3DCOV erstellt bzw. exportiert und importiert XML-Dateien über XStream.

5.4. Use Case fully dressed

Nachfolgend sind einige fully dressed Use Cases aufgelistet. Da diese Redundanz zu den System-Sequenz-Diagrammen darstellen, wurde auf die restlichen fully dressed Use Cases verzichtet.

5.4.1. UC01: Programm starten

Umfang	**Focusedj3d_on_j2d**
Ebene	Anwenderziel
Primärakteur	3DCOV
Stakeholder und Interessen	Benutzer: • Will 3DCOV Applikation benutzen 3DCOV • Will, dass die Zeichnungsfläche vorbereitet wird.
Vorbedingungen	Keine
Nachbedingungen	Der Benutzer sieht die gestartete 3DCOV-Applikation.
Standardablauf	1. 3DCOV fordert eine Canvas3D an, um die Zeichnungsfläche vorzubereiten. 2. System liefert das Canvas3D zurück. 3. 3DCOV fordert ein SimpleUniverse für den Canvas3D. 4. System liefert das SimpleUniverse zurück. 5. 3DCOV setzt die BackClipDistance auf 500. 6. System speichert BackClipDistance. 7. 3DCOV setzt die FrontClipDistance auf 0.01. 8. System speichert FrontClipDistance. 9. 3DCOV fordert eine BranchGroup an. 10. System liefert die BranchGroup zurück. 11. 3DCOV fordert ein OrbitBehavior an. 12. System liefert ein OrbitBehavior. 13. 3DCOV fügt das Behavior der ViewingPlatform des SimpleUniverse an. 14. 3DCOV fordert Behaviors für die Selection. 15. System liefert das Behavior zurück. 16. 3DCOV fordert ein PickCanvas an. 17. System liefert ein PickCanvas zurück. 18. 3DCOV fügt das Behavior der passenden BranchGroup hinzu. 19. Schritte 14 – 18 wiederholen sich für zwei weitere Behaviors (MouseOverBehavior und MouseDragBehavior). 20. 3DCOV fügt die BranchGroup dem SimpleUniverse hinzu. 21. 3DCOV fügt das Canvas seinem Cov3dModelView hinzu. 22. 3DCOV fordert focusedj3d_on_j2d auf, den Canvas zu zeichnen.
Erweiterungen oder alternative Abläufe	Keine

Spezielle Anforderungen	Keine
Liste der Technik und Datenvariationen	focusedj3d_on_j2d wird bei 3DCOV als Library eingebunden. Es wird Java und Java2D verwendet.
Häufigkeit des Auftretens	Einmal pro Start der 3DCOV-Applikation
Offene Fragen	Keine

5.4.2. UC02: Box erstellen

Umfang	**Focusedj3d_on_j2d**
Ebene	Anwenderziel
Primärakteur	3DCOV
Stakeholder und Interessen	Benutzer: • Will Klassen und Objekte dargestellt bekommen 3DCOV • Will das Zeichnen einer Box an focusedj3d_on_j2d übergeben.
Vorbedingungen	3DCOV ist gestartet und benutzt die Library focusedj3d_on_j2d (UC01).
Nachbedingungen	Der Benutzer sieht eine Box.
Standardablauf	1. Der Benutzer erstellt eine Klasse oder Objekt. 2. 3DCOV fordert eine Appearance an, um das Aussehen der Box zu definieren. 3. System liefert die Appearance zurück. 4. 3DCOV fordert eine Box mit der Appearance an. 5. System liefert die Box zurück. 6. 3DCOV fordert eine Appearance für die Oberseite der Box an, um auf dieser ein Bild darzustellen. 7. 3DCOV setzt die Appearance auf die obere Seite der Box. 8. 3DCOV fordert eine Transform3D an. 9. System liefert eine Transform3D zurück. 10. 3DCOV fordert einen Vector3D für eine kleine Verschiebung nach rechts an. 11. System liefert den passenden Vector3D zurück. 12. 3DCOV fordert focusedj3d_on_j2d auf, die Verschiebung um den Vektor3D in Transform3D zu speichern. 13. 3DCOV fordert eine TransformGroup für die Transform3D an. 14. System liefert die TransformGroup zurück. 15. 3DCOV fordert focusedj3d_on_j2d auf, die Box der TransformGroup hinzuzufügen.

	16. 3DCOV fordert focusedj3d_on_j2d auf, die TransformGroup der BranchGroup hinzuzufügen. 17. 3DCOV fordert von focusedj3d_on_j2d die Bounds vom ClassesBranch an. 18. System liefert die Bounds vom ClassesBranch zurück. 19. 3DCOV speichert diese in die BoundingSphere ab. 20. System berechnet das Zentrum der BoundingSphere. 21. 3DCOV fordert das Zentrum der BoundingSphere an. 22. 3DCOV fordert focusedj3d_on_j2d auf das Canvas zu zeichnen.
Erweiterungen oder alternative Abläufe	Keine
Spezielle Anforderungen	Keine
Liste der Technik und Datenvariationen	focusedj3d_on_j2d wird bei 3DCOV als Library eingebunden. Es wird Java und Java2D verwendet.
Häufigkeit des Auftretens	Kann ständig auftreten
Offene Fragen	Keine

5.4.3. UC03: Konstrukt rotieren

Umfang	**Focusedj3d_on_j2d**
Ebene	Anwenderziel
Primärakteur	3DCOV
Stakeholder und Interessen	Benutzer: • Will das Konstrukt aus Klassen und Objekte als Ganzes rotieren 3DCOV • Will das synchrone Rotieren aller Objekte an focusedj3d_on_j2d übergeben.
Vorbedingungen	3DCOV ist gestartet und benutzt die Library focusedj3d_on_j2d (UC01). Konstrukt bzw. 3D-Objekte sind in Objektdarstellung vorhanden (z.B. UC02).
Nachbedingungen	Der Benutzer sieht das Konstrukt von einem anderen Winkel.
Standardablauf	1. Der Benutzer lässt das Konstrukt rotieren. 2. System lässt das Konstrukt rotieren gemäss Einstellungen aus 3DCOV. 3. System fordert von Java2D ein Neuzeichnen an.
Erweiterungen	Keine

oder alternative Abläufe	
Spezielle Anforderungen	Keine
Liste der Technik und Datenvariationen	focusedj3d_on_j2d wird bei 3DCOV als Library eingebunden. Es wird Java und Java2D verwendet.
Häufigkeit des Auftretens	Kann ständig auftreten
Offene Fragen	Keine

5.4.4. UC04: Beziehungen erstellen

Umfang	**Focusedj3d_on_j2d**
Ebene	Anwenderziel
Primärakteur	3DCOV
Stakeholder und Interessen	Benutzer: • Will eine Beziehung zwischen zwei Klassen erstellen 3DCOV • Will das Zeichnen einer Linie zwischen den zwei Klassen an focusedj3d_on_j2d übergeben
Vorbedingungen	3DCOV ist gestartet und benutzt die Library focusedj3d_on_j2d (UC01). Konstrukt bzw. 3D-Objekte sind in Objektdarstellung vorhanden (z.B. UC02).
Nachbedingungen	Der Benutzer sieht eine Linie zwischen den ausgewählten zwei Klassen.
Standardablauf	1. Der Benutzer erstellt eine Assoziation. 2. 3DCOV fordert eine neue BranchGroup von focusedj3d_on_j2d an. 3. System liefert eine BranchGroup zurück. 4. 3DCOV fordert eine neue TransformGroup von focusedj3d_on_j2d an und hänge diese der BranchGroup an. 5. System liefert eine TransformGroup zurück. 6. 3DCOV fordert ein LineArray von focusedj3d_on_j2d an mit den im Voraus berechneten Punkten 7. System liefert ein LineArray zurück. 8. 3DCOV fordert ein Shape3D von focusedj3d_on_j2d an und hängt den LineArray an die Shape3D. 9. System liefert ein Shape3D zurück. 10. 3DCOV hängt die Shape3D der TransformGroup an. 11. 3DCOV hängt die BranchGroup der Linie dem BranchGroup für die Klassen-Assoziationen an.

	12. System fordert von Java2D ein Neuzeichnen an.
Erweiterungen oder alternative Abläufe	Keine
Spezielle Anforderungen	Keine
Liste der Technik und Datenvariationen	Focusedj3d_on_j2d wird bei 3DCOV als Library eingebunden. Es wird Java und Java2D verwendet.
Häufigkeit des Auftretens	Kann ständig auftreten
Offene Fragen	Keine

5.4.5. UC05: Konstrukt verschieben

Umfang	**Focusedj3d_on_j2d**
Ebene	Anwenderziel
Primärakteur	3DCOV
Stakeholder und Interessen	Benutzer: • Will das Konstrukt aus Klassen und Objekte als Ganzes verschoben wird 3DCOV • Will das synchrone Verschieben aller Objekte an focusedj3d_on_j2d übergeben.
Vorbedingungen	3DCOV ist gestartet und benutzt die Library focusedj3d_on_j2d (UC01). Konstrukt bzw. 3D-Objekte sind in Objektdarstellung vorhanden (z.B. UC02).
Nachbedingungen	Der Benutzer sieht das Konstrukt an einem anderen Ort im Fenster dargestellt.
Standardablauf	1. Der Benutzer verschiebt das Konstrukt. 2. System verschiebt das Konstrukt gemäss 3DCOV-Einstellungen. 3. System fordert von Java2D ein Neuzeichnen an.
Erweiterungen oder alternative Abläufe	Keine
Spezielle Anforderungen	Keine
Liste der Technik und Datenvariationen	focusedj3d_on_j2d wird bei 3DCOV als Library eingebunden. Es wird Java und Java2D verwendet.
Häufigkeit des Auftretens	Kann ständig auftreten

Offene Fragen	Keine

Projekt: J3DEval

Domainanalyse

Version: 3.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
24.10.2011	1.0	Erstellung des Dokuments	Lara Mühlemann
25.10.2011	1.0	Konzeptbeschreibung fertiggestellt, Systemsequenzdiagramm UC01 erstellt	Lara Mühlemann
30.10.2011	1.0	Contracts UC01 geschrieben	Lara Mühlemann
07.11.2011	1.0	Änderung am Domainmodel gemäss Besprechung.	Lara Mühlemann
14.11.2011	1.0	Abgeleitete Assoziation im Domain-Modell ergänzt	Marion Walser
27.11.2011	2.0	Sequenzdiagramme für UC03 und UC04 erstellt	Marion Walser
19.12.2011	3.0	Domain Model mit Behavior und Text2D ergänzt	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Lara Mühlemann verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
01.11.2011	Marion Walser + Lara Mühlemann
28.11.2011	Lara Mühlemann
19.12.2011	Lara Mühlemann

1.4. Inhalt

2. Einführung

2.1. Zweck

Der Zweck dieses Dokuments ist, die Domainanalyse des Projektes J3DEval zu dokumentieren.

2.2. Gültigkeitsbereich

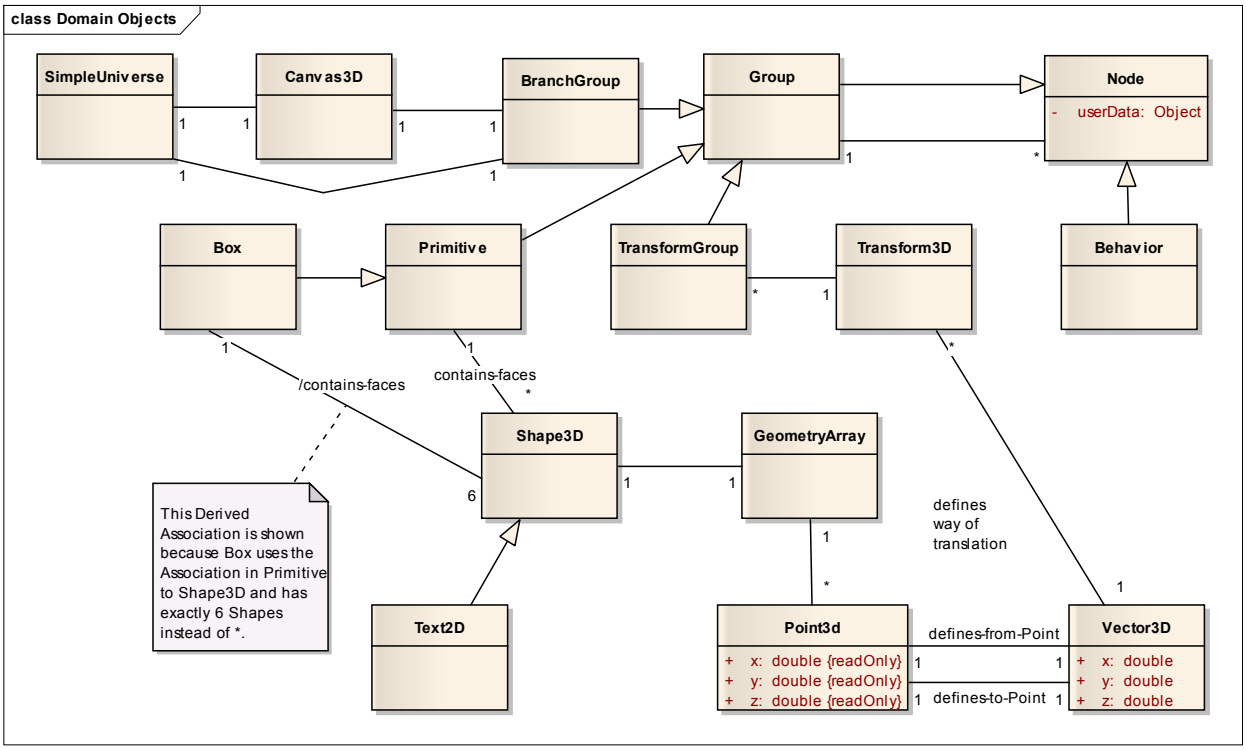
Dieses Dokument ist während der Projektdauer gültig.

2.3. Definition und Abkürzungen

Siehe Glossar.doc

3. Domain-Modell

3.1. Strukturdiagramm



3.2. Konzeptbeschreibung

Group	
Beschreibung	Eine Group wird benutzt um mehrere Nodes zu gruppieren.
Attribute	
Beziehungen	Group ist eine Unterklasse von Node, da in einer Gruppe auch Untergruppen hinzugefügt werden können. Group hat Referenzen auf Nodes, welche in dieser Group gruppiert werden.

BranchGroup	
Beschreibung	Ein BranchGroup ist eine Group. Auf die Erweiterungen gegenüber einer Group wird in diesem Release nicht eingegangen.
Attribute	
Beziehungen	BranchGroup erbt von Group.

TransformGroup	
Beschreibung	Eine TransformGroup ist eine Group, welche als Ganzes transformiert werden kann.
Attribute	
Beziehungen	TransformGroup hat eine Referenz auf Transform3D.

Transform3D	
Beschreibung	Speichert eine Transformation
Attribute	
Beziehungen	Transform3D hat eine Referenz auf Vector3D um mit dem Vektor den Weg einer Transformation zu beschreiben.

Node		
Beschreibung	Eine Node ist ein Objekt, welches einer Group hinzugefügt werden kann.	
Attribute	userData:	Ein Objekt, welches der Benutzer von

	focusedj3d_on_j2d als Hilfsobjekt abspeichern und wieder auslesen kann. (Wird von der Schnittstelle vorgegeben)
Beziehungen	

SimpleUniverse	
Beschreibung	SimpleUniverse wird von der Schnittstelle vorgegeben. SimpleUniverse repräsentiert eine 3D-Umgebung.
Attribute	
Beziehungen	SimpleUniverse hat eine Referenz auf ein Canvas3D und eine Referenz auf eine BranchGroup. Java3D erlaubte es, mehrere Canvas3D's und BranchGroups zu referenzieren. Da der Schnittstelle jeweils eins davon genügt, wird zur Vereinfachung auch nur je eins zur Verfügung gestellt. Obwohl BranchGroup ebenfalls von Canvas3D referenziert wird, ist diese Abhängigkeit nicht entfernbar, da sie von der Schnittstelle vorgegeben wird.

Canvas3D	
Beschreibung	Canvas3D ist ein Canvas. Ein Canvas ist ein Rechteck, auf welchem gezeichnet werden kann.
Attribute	
Beziehungen	Canvas3D erweitert java.awt.Canvas Canvas3D enthält eine Referenz auf eine BranchGroup. Dies ist nötig, damit beim Zeichnen mit Java2D der Zeichnungsprozess über die von Canvas geerbte paint Methode gestartet wird und an die BranchGroup weitergeleitet werden kann. Durch diese Notwendigkeit entsteht der Nachteil, dass die BranchGroup doppelt referenziert wird, von SimpleUniverse und von Canvas3D.

Box	
Beschreibung	Eine Box repräsentiert einen Quader. Dies ist ein 3D Objekt. Ein Quader besteht aus 6 Seitenflächen (Shape3D).
Attribute	
Beziehungen	Box ist eine Unterklasse von Primitive Box erbt von Primitive die Beziehung zu Shape3D. Der Spezialfall Box

	enthält genau 6 Shape3Ds.

Primitive	
Beschreibung	Ein Primitive repräsentiert ein 3D Objekt, welches aus beliebig vielen Seitenflächen (Shape's) besteht.
Attribute	
Beziehungen	Primitive ist eine Unterklasse von Group. (Wird von der Schnittstelle vorgegeben) Ein Primitive enthält mehrere Shape3Ds.

Shape3D	
Beschreibung	Eine Shape3D repräsentiert eine Seitenfläche eines 3D Objekts.
Attribute	
Beziehungen	Ein Shape3D enthält ein GeometryArray, wodurch die Eckpunkte des Shape3D beschrieben werden.

GeometryArray	
Beschreibung	Eine GeometryArray kapselt mehrere Point3ds und ist verantwortlich für deren Verschiebung und Rotation.
Attribute	
Beziehungen	Ein GeometryArray enthält mehrere Point3ds.

Point3d		
Beschreibung	Point3d kapselt die Positionsinformationen eines Punktes in einem Koordinatensystem.	
Attribute	x	Der Wert der Abszisse
	y	Der Wert der Ordinate
	z	Der Wert der Applikate
Beziehungen		

Vector3d

Beschreibung	Vector3D repräsentiertden Abstand und die Richtung zwischen zwei Punkten im dreidimensionalen Raum.	
Attribute	x:	Differenz zwischen dem X-Wert des Anfangspunktes und des Endpunktes
	y	Differenz zwischen dem Y-Wert des Anfangspunktes und des Endpunktes
	z	Differenz zwischen dem Z-Wert des Anfangspunktes und des Endpunktes
Beziehungen	Vector3d hat zwei Referenzen zu Point3d. Ein Point3d repräsentiert den Startpunkt des Vektors, der andere den Endpunkt.	

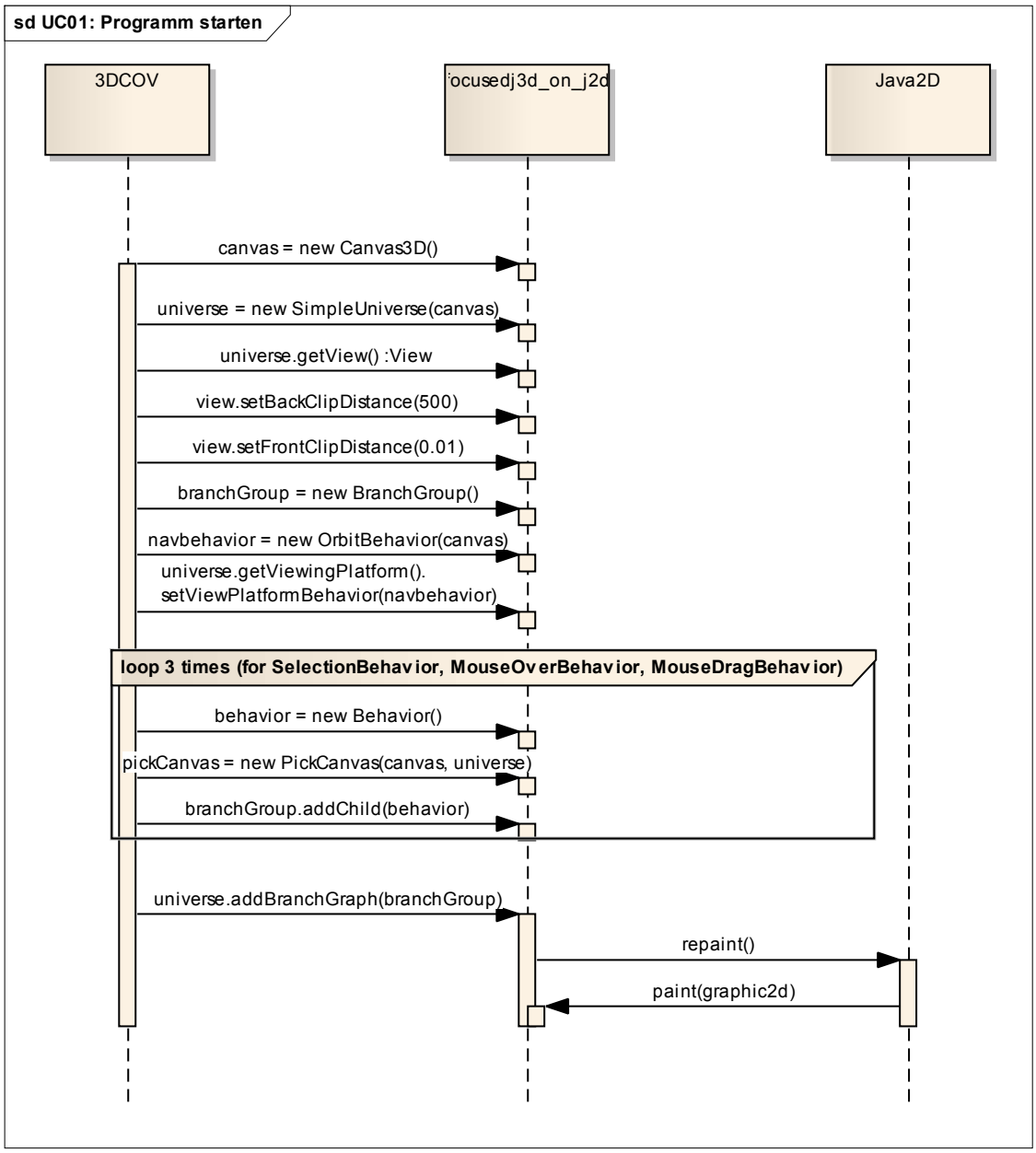
Text2D	
Beschreibung	Text2D enthält den Namen, die Rollen und die Kardinalitäten einer Assoziation.
Attribute	
Beziehungen	Text2D ist eine Unterklasse von Shape3D.

Behavior	
Beschreibung	Behavior enthält das Verhalten der Group, der es hinzugefügt wurde.
Attribute	
Beziehungen	Behavior ist eine Unterklasse von Node.

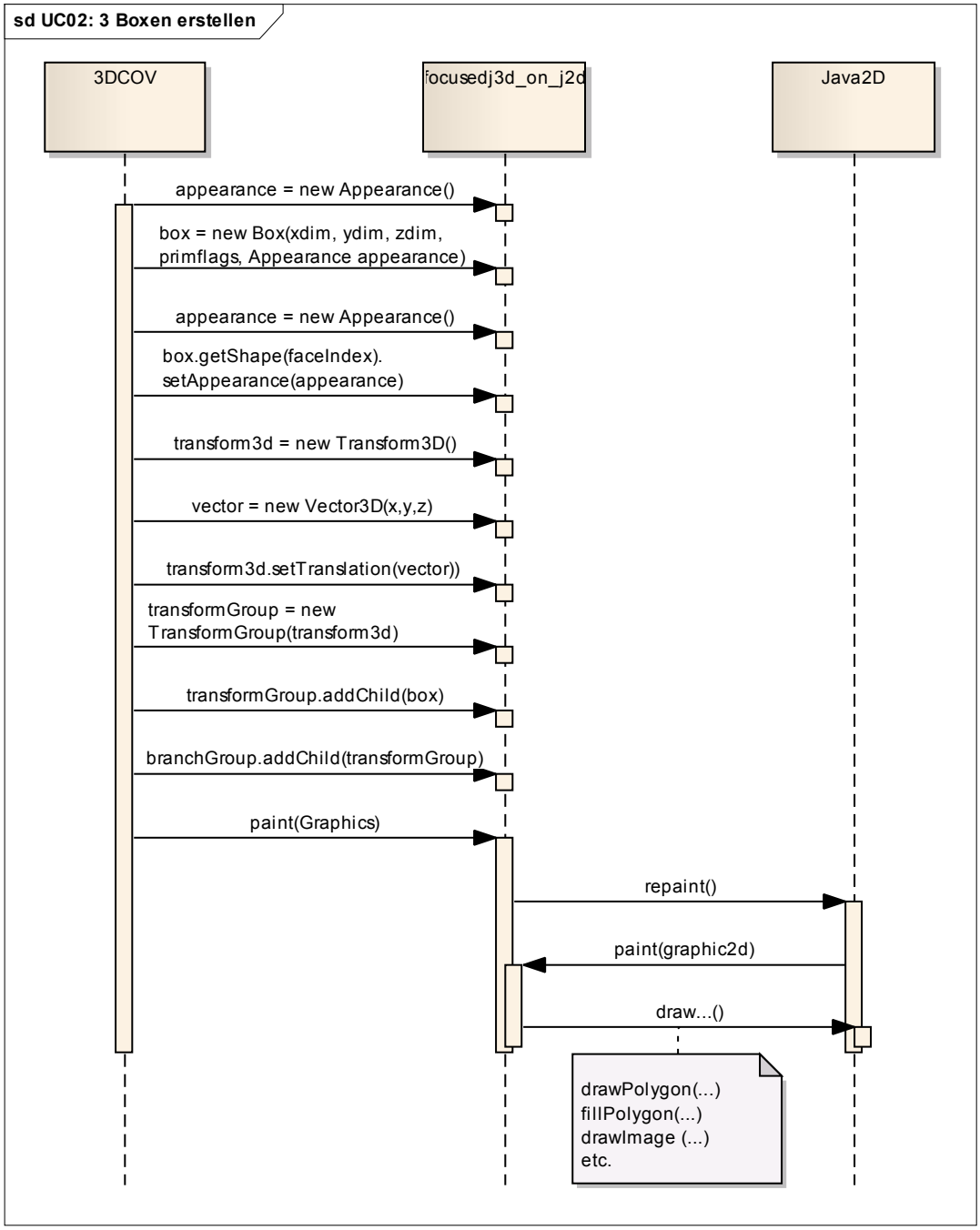
4. System Sequenzdiagramme

Von den wichtigsten Use Cases folgen hier die System-Sequenzdiagrammen.

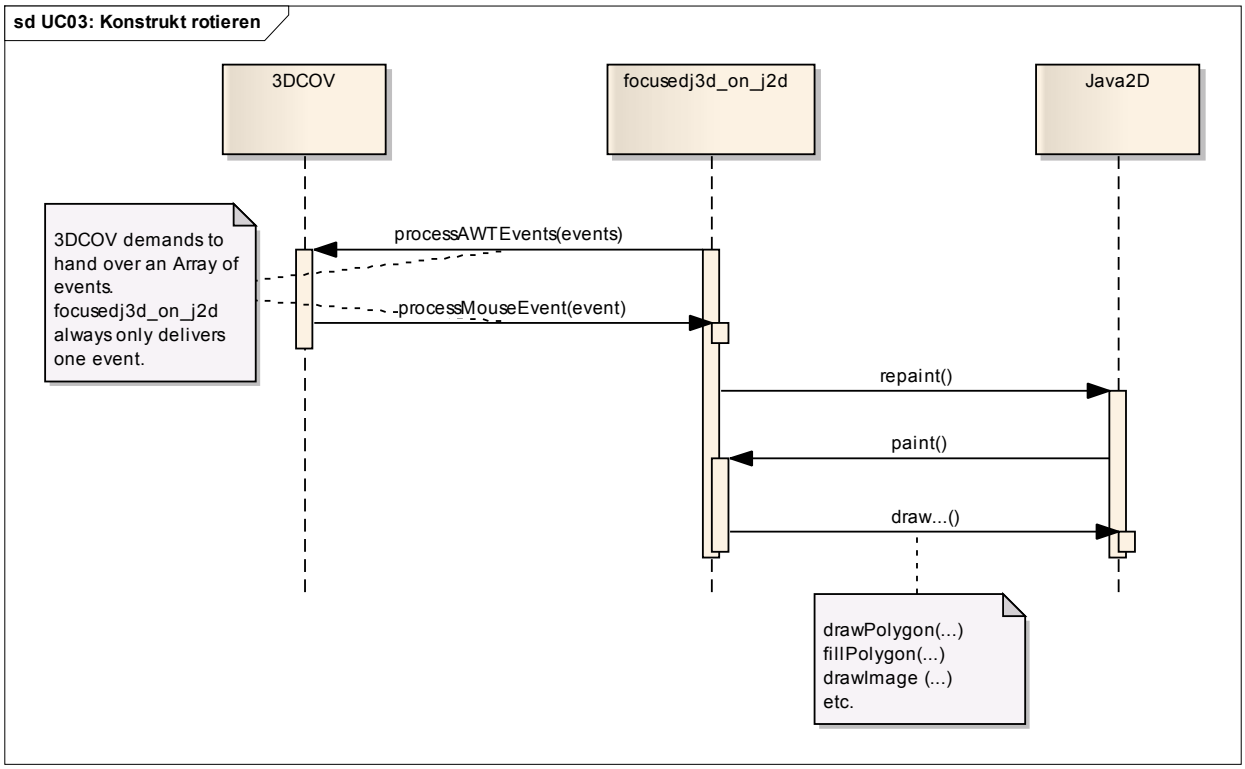
4.1. UC01: Programm starten



4.2. UC02: Box erstellen

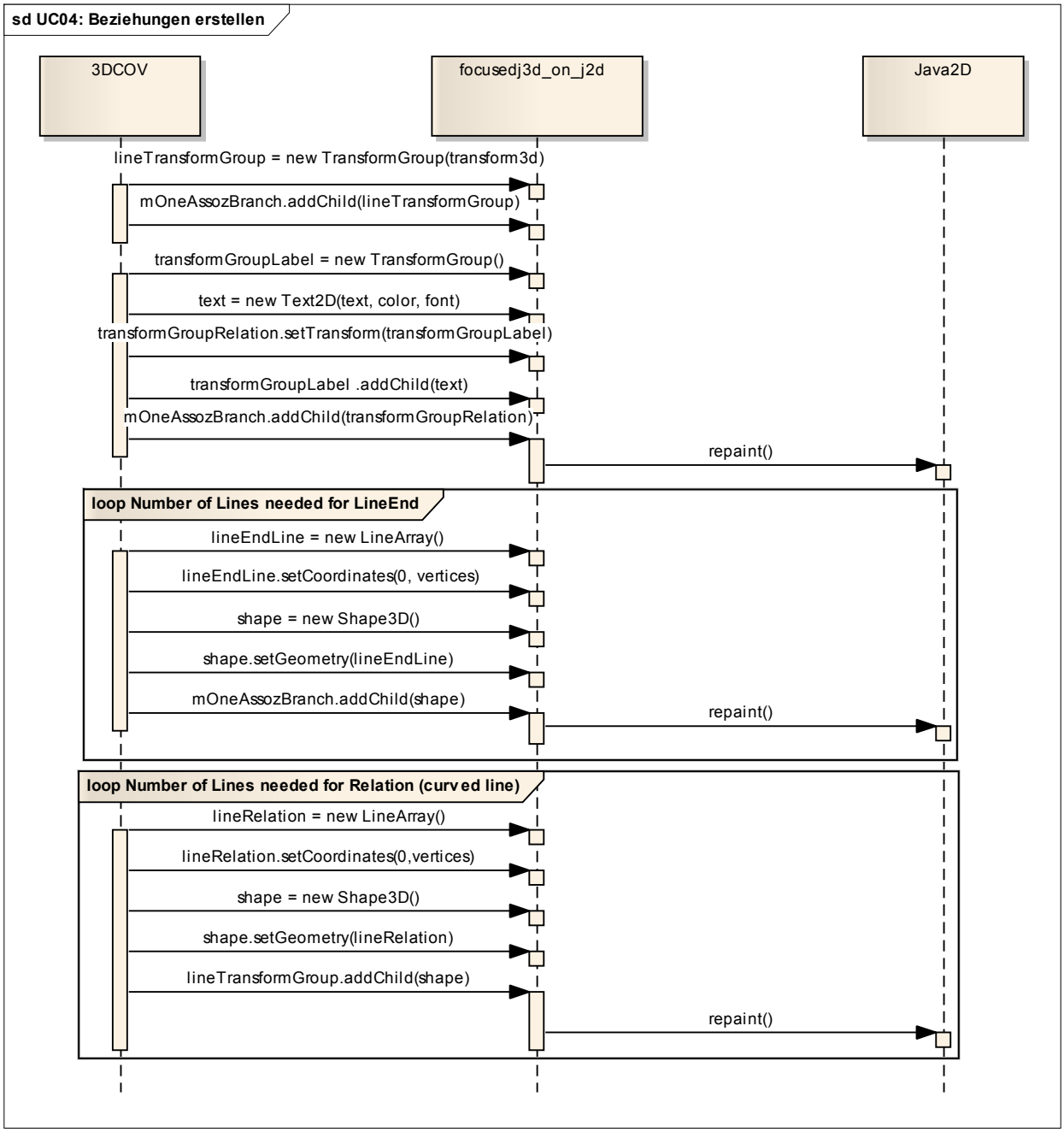


4.3. UC03: Konstrukt rotieren



4.4. UC04: Beziehungen erstellen

Es folgt ein vereinfachtes Sequenzdiagramm. Aufrufe, wie die Verschiebung/Rotation oder das Setzen des Aussehens (Appearance), welche für diesen Use Case nicht essenziell sind, sind nicht aufgeführt.



4.5. UC05: Konstrukt verschieben

Dieser Use Case entspricht im Ablauf dem Use Case „Konstrukt rotieren" (UC03) und wird deshalb hier nicht nochmals dargestellt.

4.6. UC06: Konstrukt zoomen

Dieser Use Case entspricht im Ablauf dem Use Case „Konstrukt rotieren" (UC03) und wird ebenfalls nicht nochmals dargestellt.

5. Systemoperationen

Im diesem Kapitel werden die interessantesten Contracts beschrieben.

5.1. Contract UC01: Programm starten

Operation	**new SimpleUniverse(canvas:Canvas3D)**
Querverweise	Use Case: Programm starten
Vorbedingung	• Eine Canvas3D wurde erstellt.
Nachbedingung	• Eine Instanz universe von SimpleUniverse wurde erstellt. • canvas wurde mit universe verknüpft. • Es wurde eine Instanz viewingPlatform von ViewingPlatform erstellt. • viewingPlatform wurde mit universe verknüpft. • Es wurde eine Instanz viewer von Viewer erstellt. • viewer wurde mit universe verknüpft

5.2. Contract UC02: Box erstellen

Operation	**new Box(xdim:float, ydim:float, zdim:float, primflags:int, appearance:Appearance)**
Querverweise	Use Case: Box erstellen
Vorbedingung	• Eine Appearance wurde erstellt.
Nachbedingung	• Eine Instanz box von Box wurde erstellt • Für jede Seite der Box wurde eine Instanz vom Typ Shape erstellt.

Projekt: J3DEval

System Architecture Document

Version: 3.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
07.11.2011	1.0	Erstellung des Dokuments	Lara Mühlemann
22.11.2011	1.0	Klassendiagramme ergänzt	Marion Walser
26.11.2011	1.0	Texte zu den Klassendiagrammen ergänzt	Lara Mühlemann
27.11.2011	1.0	Klassendiagramme aktualisiert	Marion Walser
17.12.2011	2.0	Aktualisierung auf neusten Stand	Lara Mühlemann
17.12.2011	2.0	Klassendiagramm graph neu formatiert, weitere textuelle Ergänzungen	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Lara Mühlemann verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
28.11.2011	Lara Mühlemann

1.4. Inhalt

2. Einführung

2.1. Zweck

Der Zweck dieses Dokumentes ist, die Software Architektur und das Design der Library „focusedj3d_on_j2d" zu beschreiben.

2.2. Gültigkeitsbereich

Dieses Dokument ist während der Projektdauer gültig.

2.3. Referenzen

[1] Anforderungsspezifikation.doc

2.4. Definition und Abkürzungen

Siehe Glossar.doc

3. Architektonische Ziele & Einschränkungen

3.1. Ziele

- Die Architektur soll in die Schichten model und view unterteilt werden.
- Die Architektur soll ermöglichen, dass einfach neue Geometrieobjekte hinzugefügt werden können.

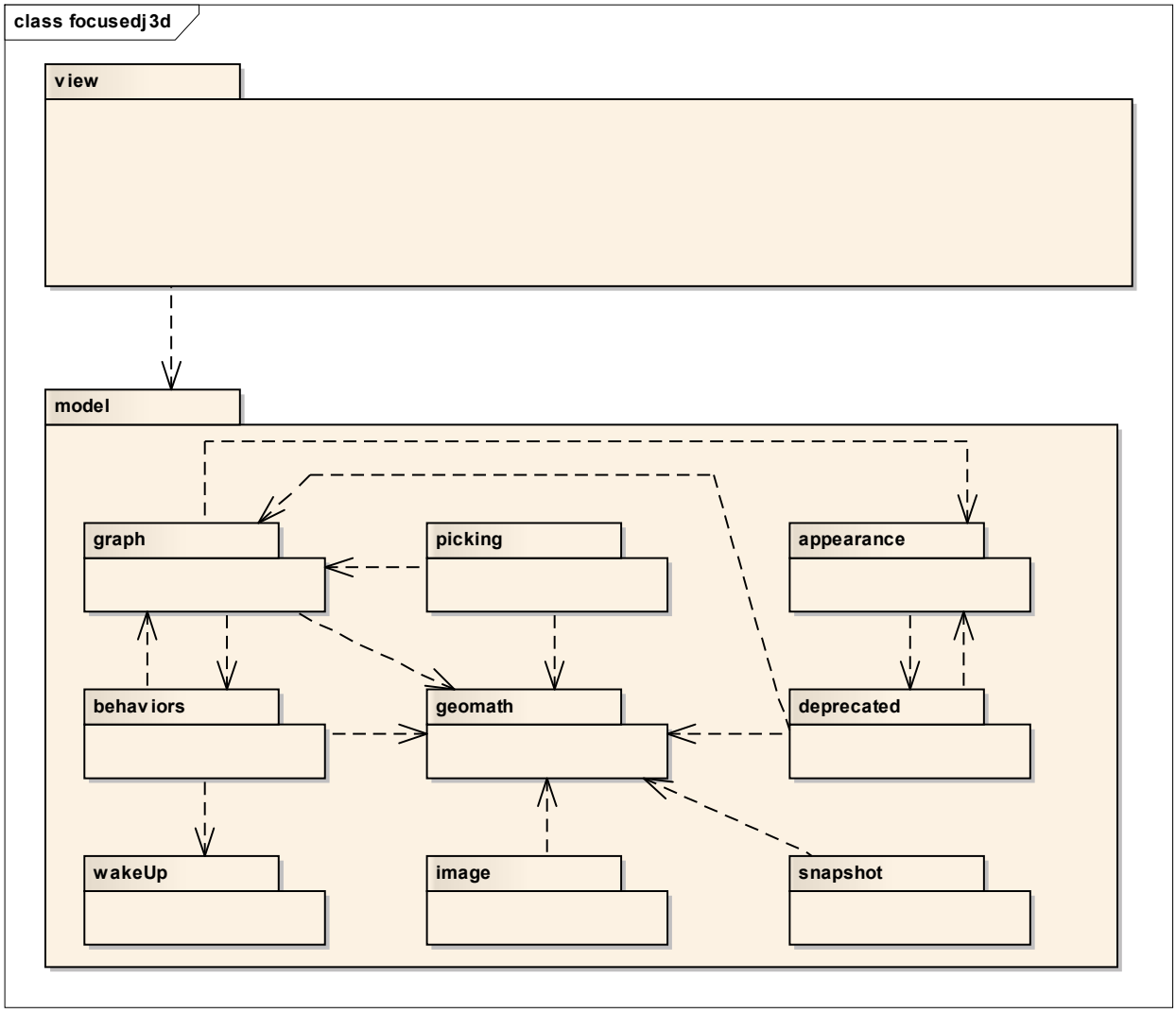
3.2. Einschränkungen

- Die Schnittstelle, welche von 3DCOV vorgegeben wird, muss eingehalten werden. (Siehe Kapitel Schnittstelle in der Anforderungsspezifikation[1])

4. Logische Architektur

4.1. Übersicht

Focusedj3d_on_j2d verwendet eine 2-Layer-Architektur mit den Packages view und model. Der Zugriff erfolgt von oben nach unten. Das bedeutet, dass das Package view auf das Package model greift, das Package model jedoch nicht auf das Package view.



Innerhalb des Package model bestehen zyklische Beziehungen.

Die Beziehung von graph zu behaviors ist im Kapitel Design unter der Beschreibung der Klasse Node (Package model.graph) erklärt.

Die zweite zyklische Beziehung wird vom Package deprecated verursacht und von der Schnittstelle vorgegeben.

5. Prozesse und Threads

Es wird zu den Threads, welche Java2D benutzt, keine zusätzlichen Threads oder Prozesse erstellt.

6. Datenspeicherung

Es werden keine Daten persistent gespeichert, beziehungsweise übernimmt 3DCOV die Datenspeicherung.

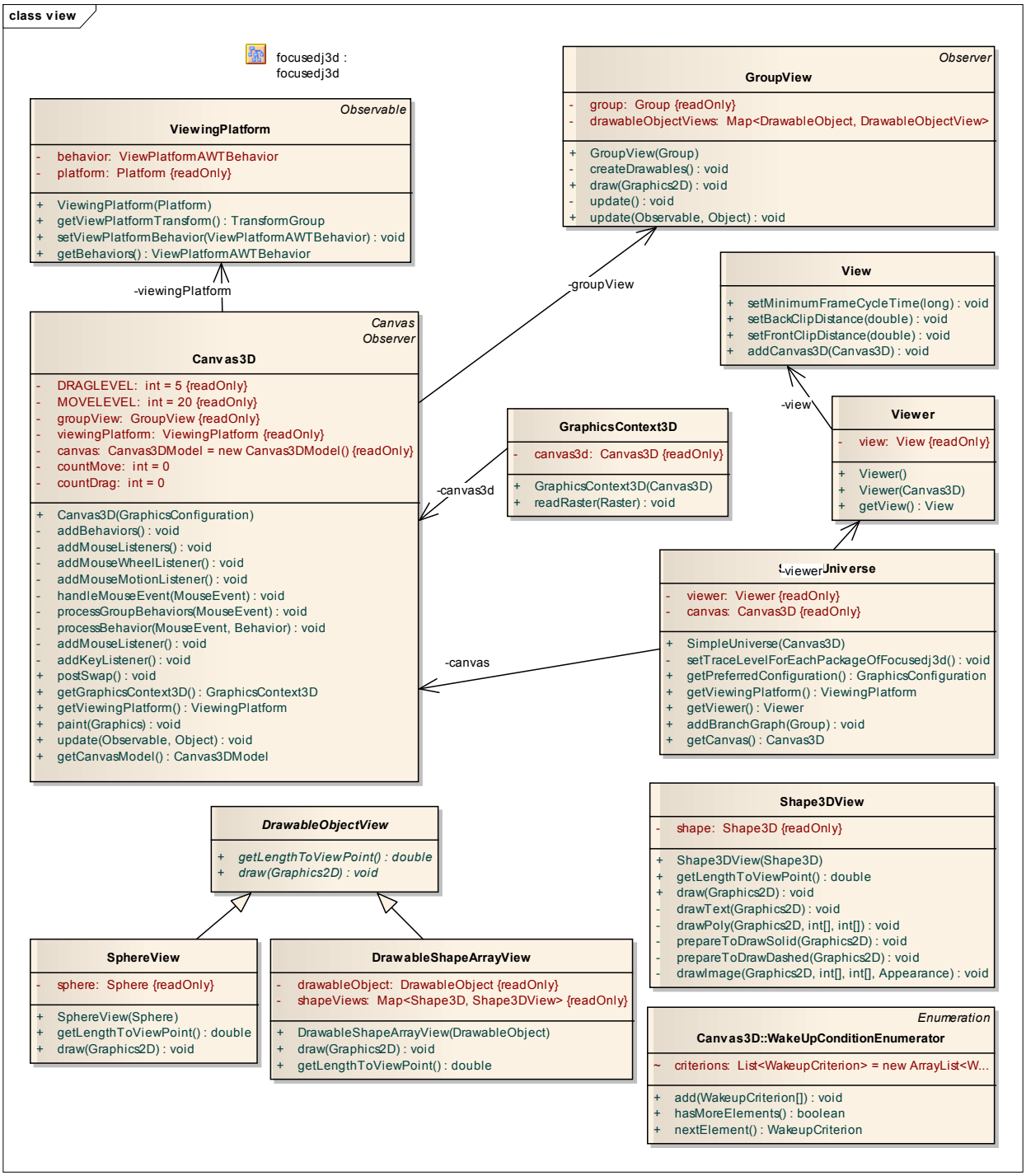
3DCOV benutzt die Library „XStream" für die Implementierung der XML-Schnittstelle. Dadurch, dass früher erstellte XML-Dateien weiterhin importierbar sein sollen, mussten in verschiedenen Klassen spezielle Erweiterungen, vor allem im Package geomath, eingefügt werden. Diese sind im Kapital „Design" weiter ausgeführt.

Früher erstellte XML-Dateien können weiterhin eingelesen und verwendet werden. Jedoch ist es nicht möglich, aus focusedj3d_on_j2d erstellte XML-Dateien in die frühere 3DCOV-Version, die mit Java 3D arbeitet, einzulesen (keine Rückwärtskompatiblität.

7. Design

Sequenzdiagramme wurden auf Anraten des Betreuers nicht erstellt, da sie aufgrund ihrer Änderungsanfälligkeit nur mühsam aktuell zu halten und für die Entwicklung nicht sehr nützlich sind.

7.1. Package view



Im Package view befinden sich die Klassen, welche für das Zeichnen mittels Java2D verantwortlich sind.

Die wichtigste Klasse in diesem Package ist die Klasse `Canvas3D`. Sie erbt von der java.awt Klasse `Canvas` und repräsentiert dementsprechend eine rechteckige Leinwand, in die gezeichnet werden kann.

Vorbereiten zum Zeichnen der Elemente

Um das Zeichnen der Elemente wie z.B. einer Box oder einer Linie vorzubereiten, wird zuerst in `Canvas3D` eine `GroupView` erstellt. Eine `GroupView` repräsentiert eine `Group` aus dem Package model. Die Klasse `GroupView` ist dafür verantwortlich, dass alle Elemente innerhalb der `Group` gezeichnet werden, die das Interface `DrawableObject` implementieren. Die `GroupView` erstellt für jedes `DrawableObject` eine Instanz von `DrawableObjectView` und speichert diese in einer Liste.

`DrawableObjectView` erstellt für jede `Shape`, die das `DrawableObject` enthält eine `ShapeView`.

Zeichnen der Elemente

Das Zeichnen der Elemente wie z.B. eine Box oder eine Linie erfolgt folgendermassen:

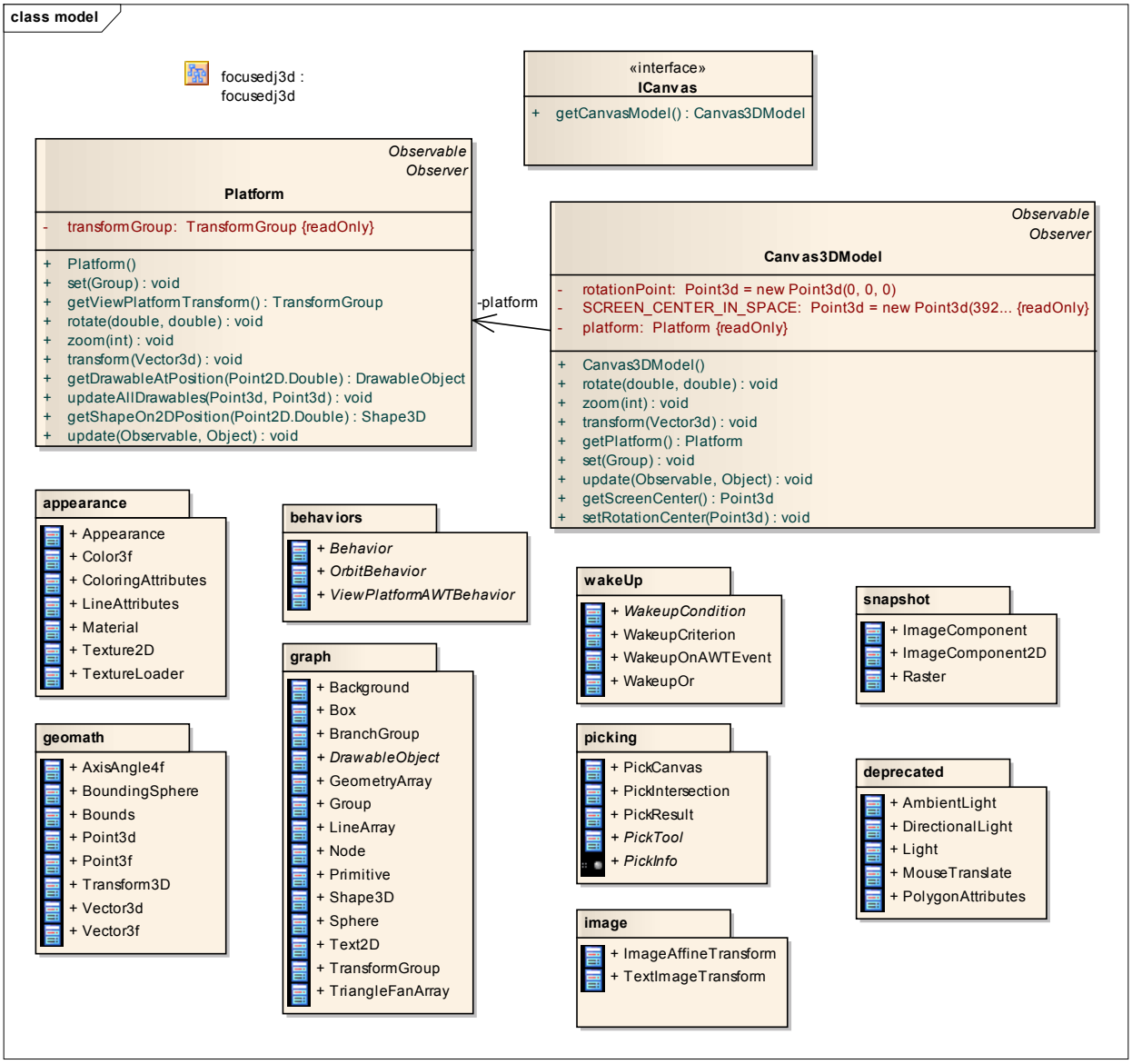
In der Klasse `Canvas3D` wird die Methode paint aufgerufen. Auf der `GroupView`-Instanz wird innerhalb der paint Methode die Methode draw aufgerufen. In dieser werden anschliessend die `DrawableObjectViews` so sortiert, dass zuerst diejenigen Objekte gezeichnet werden, welche am weitesten vom ViewPoint entfernt sind. So wird sichergestellt, dass die hinteren Objekte vor den vorderen gezeichnet und damit von diesen überzeichnet werden. Anschliessend wird in der richtigen Reihenfolge die draw Methode aller `DrawableObjectViews` aufgerufen. Die `ShapeViews` werden in `DrawableObjectViews` ebenfalls sortiert und deren draw Methode aufgerufen, in der die `Shapes` schliesslich mittels Java2D gezeichnet werden.

Tooltip

Nach dem Zeichen der Elemente wird postSwap aufgerufen, welche in 3DCOV überschrieben wird. Dadurch wird der Tooltip der Objekte gezeichnet. Um nach dem Verlassen eines Objekts ein Repaint anzufordern, wurde der Code von 3DCOV verändert. Dies geschah in der Methode draw in der Klasse ch.hsr.cov.view.cov3dmodel.Cov3DCanvas.

Um das Flimmern beim Überfahren der Objekte zu reduzieren wurde die Konstante MOVELEVEL eingeführt. Diese definiert wie oft (in Millisekunden) neu gezeichnet werden soll. Dies bei Dragevents ebenfalls einzuführen hat sich nicht bewährt, da bei leistungsschwächeren PCs die verursachte Verzögerung beim Rotieren für den Benutzer unangenehmer sind.

Package model



Das Package model ist für die Verwaltung der von 3DCOV angeforderten Objekte verantwortlich sowie für die Logik, zu der die 3D- und 2D-Berechnungen gehören.

ICanvas

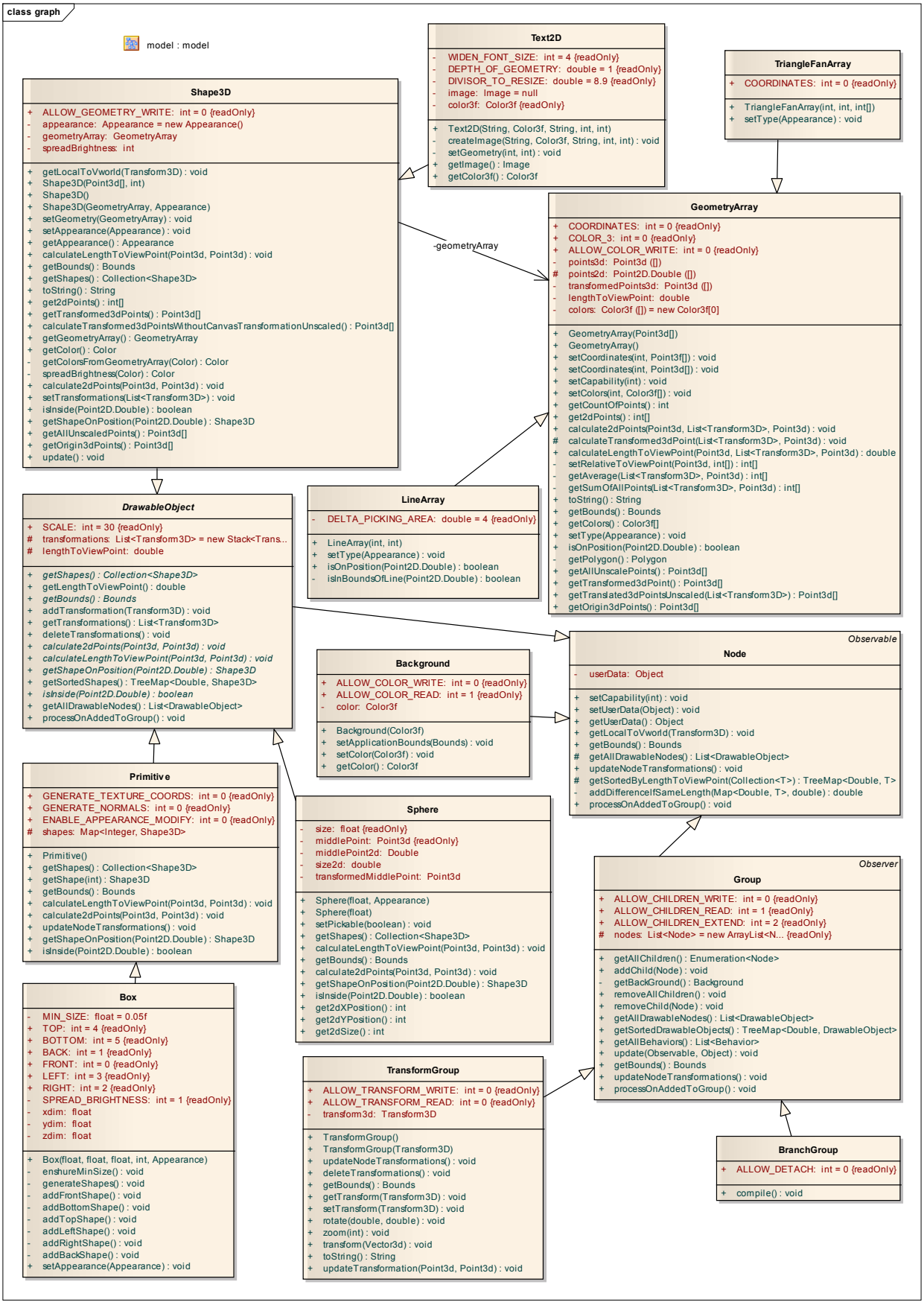
Das Interface ICanvas muss von der Canvas3D Klasse im View implementiert werden. Dadurch wird sichergestellt, dass diese ein Objekt der Klasse Canvas3DModel zurückgeben kann. Durch die Einführung dieses Interfaces konnte eine Beziehung vom Package model zum Package view vermieden werden.

Canvas3DModel/Platform

Diese Klassen sind für die Funktionalitäten zuständig, die vom Canvas des view-Packages verwendet werden. Dadurch dass `Canvas3DModel` ein Observer der `Platform`, die `Platform` von dessen `Group` und die `Group` von ihren `Childs` ist, registriert `Canvas3DModel` alle relevanten Änderungen der enthaltenen `DrawableObjects`. In der Hook-Methode update

wird die Methode updateAllDrawables der `Platform` aufgerufen, wo unter anderem alle 2D-Punkte neu berechnet werden. Dies geschieht hier, um den Aufruf von paint in `Canvas3D` zu entlasten und das Zeichnen dadurch schneller erfolgt.

7.1.1. Package model.graph



Baumstruktur

Das Hinzufügen von Elementen wird über eine Baumstruktur gelöst. Alle Klassen, deren Objekte dem Baum hinzugefügt werden können, erweitern die Klasse `Node`. Diese Klassen können einer `Group` mit Aufruf der Methode addChild hinzugefügt werden. Da einer `Group` auch eine `Group` angehängt werden kann, erbt `Group` ebenfalls von `Node`.

TransformGroup

`TransformGroup` erweitert die Klasse `Group` und fügt die Fähigkeit hinzu, die Gruppe zu transformieren.

Die Information über die Transformation wird über die Methode updateNodeTransformations an die Kinder der TransformGroup weitergeleitet. In den Kindern wird die TransformGroup in einer Collection gespeichert. Folgend wird beim Berechnen der transformierten Punkte die Transformation wie bei einem Stack nach LIFO abgearbeitet.

Da beim Hinzufügen oder Verschieben der Klassen oder Objekte ein neuer Rotationspunkt berechnet wird, springt das Konstrukt zum Teil an eine neue Position. Dies könnte behoben werden, indem in der Klasse TransformGroup statt einer einzelnen Transform3D eine Map verwendet wird, welche als Key den Rotationspunkt benutzt. Dadurch entsteht aber bei längerer Laufzeit des Programms und häufigen Neuberechnungen des Rotationspunkts eine grosse Map, was ein Performanceverlust beim Berechnen der rotierten Punkte bedeutet.

BranchGroup

Die Klasse `BranchGroup` wird von der Schnittstelle vorgegeben. Diese Klasse bringt gegenüber der Klasse `Group` keine Erweiterung und ist daher für diese Library unnötig und als deprecated markiert.

Primitive

Die Klasse `Primitive` repräsentiert eine dreidimensionale geometrische Figur, welche aus mehreren `Shape3D besteht.`

Shape3D

Diese Klasse repräsentiert eine Seitenfläche eines `Primitives` oder eine Linie. Ein `Shape3D` enthält ein `GeometryArray`, welches die Eckpunkte bestimmt und eine `Appearance`, welche das Aussehen definiert.

Sphere

Über die Klasse `Sphere` wird eine Kugel definiert. 3DCOV verwendet die Klasse, um im Objektdiagramm eine Klasse als selektiert zu markieren. Dazu erstellt sie an jeder Ecke der Klasse eine Sphere.

Box

Eine `Box` ist ein Quader. Eine Box wird von 3DCOV dazu verwendet, Klassen und Objekte darzustellen. Sie enthält damit immer sechs `Shape3D`.

GeometryArray

Ein `GeometryArray` ist eine Sammlung von Punkten. Die Schnittstelle gibt vor, dass darin mehrere Farbe (`Color3f`) gesetzt werden kann. Focusedj3d_on_j2d unterstützt jedoch nur das Setzen einer Farbe pro `Shape3D`. Deshalb wurden die dazugehörigen Methoden und das Attribut als deprecated markiert.

LineArray

`LineArray` erweitert `GeometryArray`. `LineArray` wird von 3DCOV benutzt, wenn eine Linie gezeichnet werden soll.

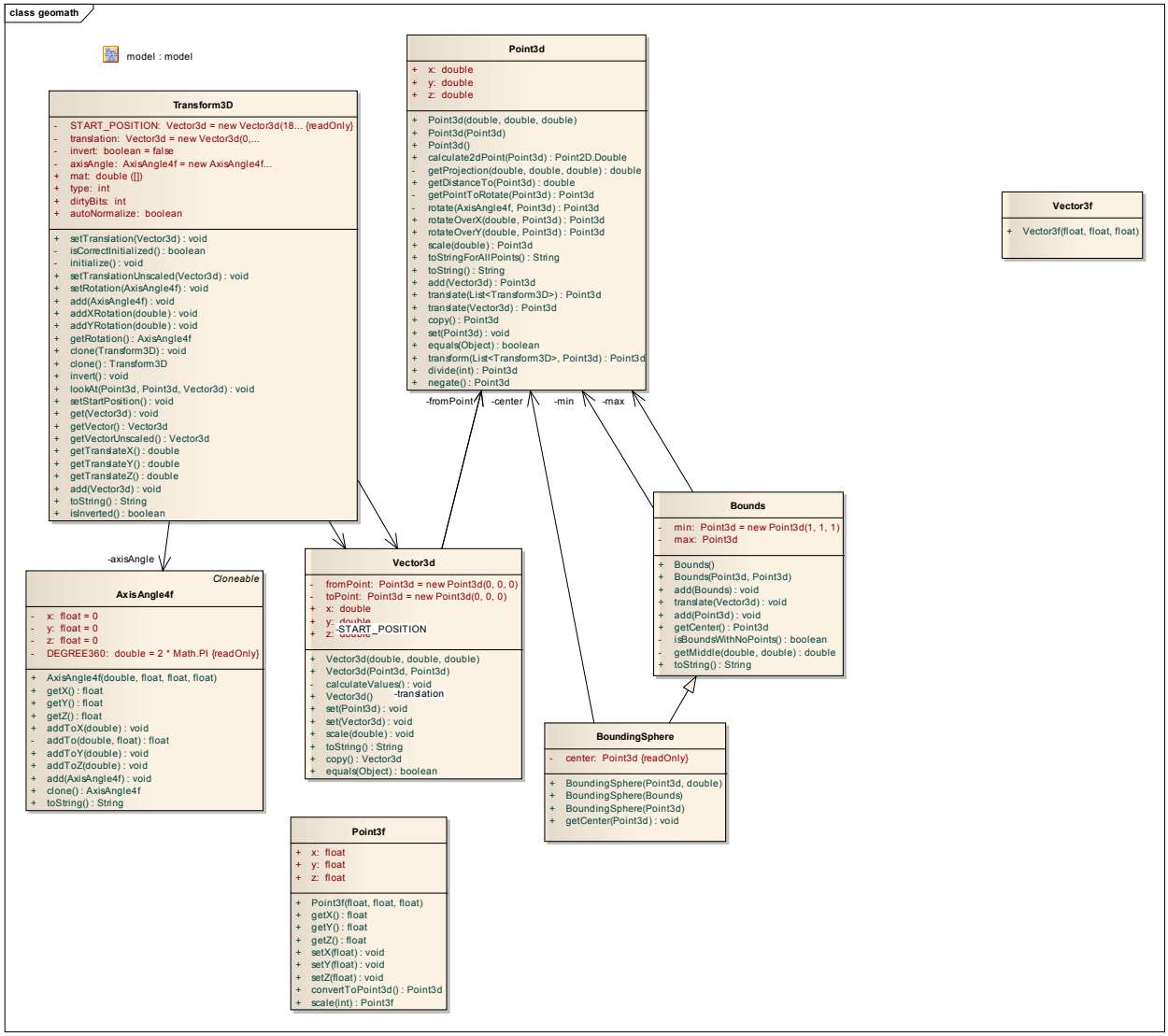
DrawableObject

`DrawableObject` ist ein Interface, welches von allen Objekten implementiert wird, welche gezeichnet werden (sichtbar sein) können. Ausserdem ist hier eine Konstante SCALE enthalten. Diese Konstante wird benötigt um die kleinen Werte, die von 3DCOV übergeben werden zu vergrössern.

Text2D

Diese Klasse wird benutzt um Texte darzustellen. Sie werden von 3DCOV benutzt, um die Assoziationen zu beschriften.

7.1.2. Package model.geomath



Im Package geomath sind diejenigen Klassen enthalten, welche geometrische Berechnungen durchführen.

Point3d/Point3f

`Point3d` stellt einen Punkt in einem dreidimensionalen Koordinatensystem dar. Diese Klasse kann einen Punkt rotieren, verschieben und seine Position im zweidimensionalen Koordinatensystem berechnen. Die Schnittstelle fordert sowohl einen solchen Punkt, dessen Position mit doubles wie auch mit floats repräsentiert wird. Da focusedj3d_on_j2d nicht mit floats rechnet, enthält `Point3f` die von der Schnittstelle geforderten Methoden und eine zusätzliche, welche den `Point3f` in einen `Point3d` konvertiert.

Bounds

Die Klasse `Bounds` bestimmt den Quader, welches ein `DrawableObject` oder eine `Group` im Raum einnimmt. Diese Klasse enthält eine Methode, die den Mittelpunkt des Quaders errechnet, welcher für die Rotation benötigt wird.

Transform3D

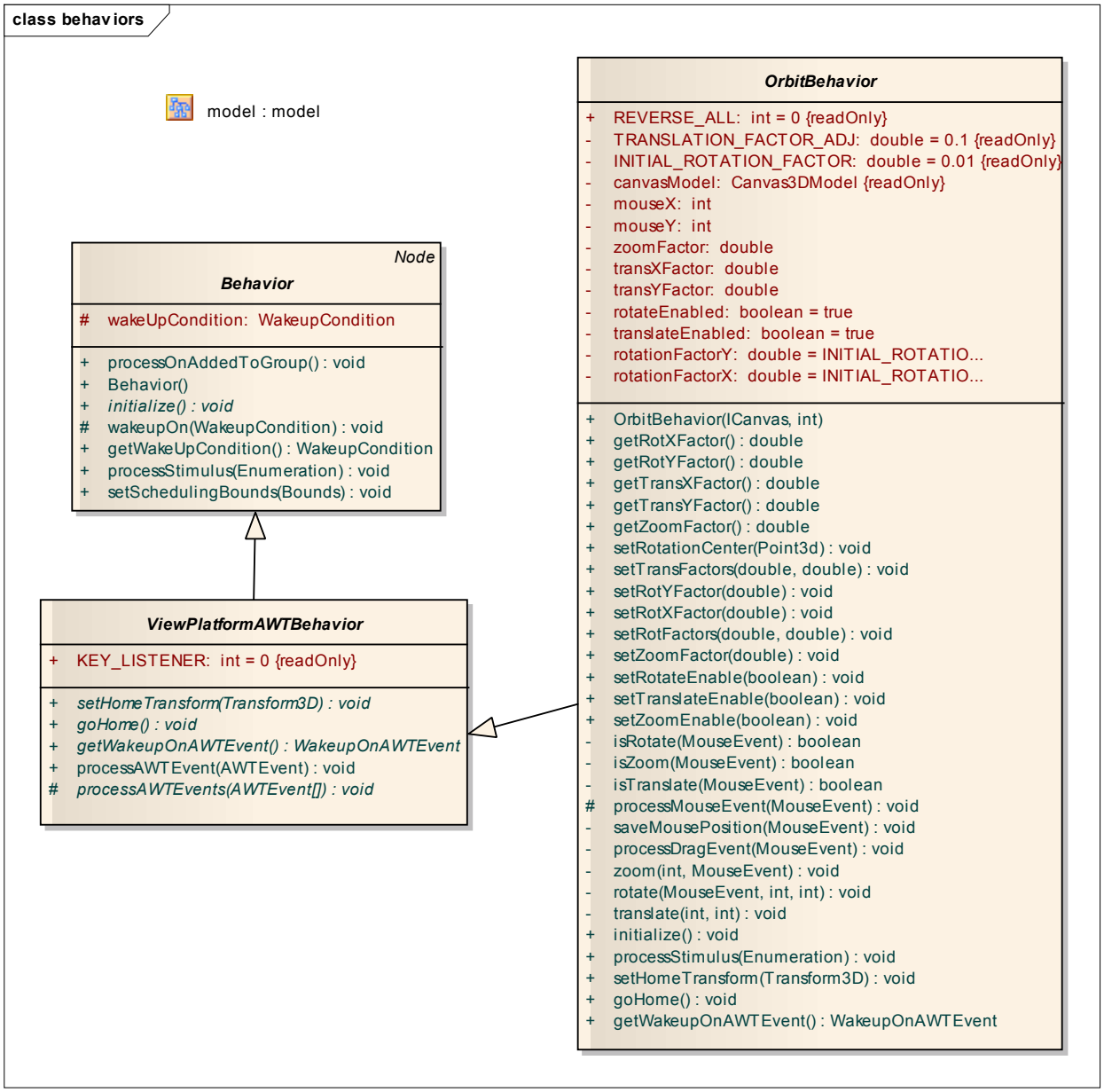
`Transform3D` speichert eine Verschiebung (Translation) und eine Rotation. Damit werden die Punkten von den einzelnen `DrawableObject` an die unterschiedliche Position gesetzt.

In dieser Klasse mussten Instanzvariablen sowie Guards in den Methoden zusätzlich implementiert werden, um sicherzustellen, dass frühere XML-Dateien weiterhin eingelesen werden. Die Instanzvariablen wurden benötigt, um die XML-Tags einlesen zu können, welche diese referenzieren. Die Guards stellen sicher, dass die von focusedj3d_on_j2d neu erstellten Instanzvariablen auch initialisiert sind, da XStream die Objekt nicht über Ausführung eines Konstruktors erstellt.

Vector3d/Vector3f

Ein `Vector3d` enthält einen Anfangs- und einen Endpunkt. Da 3DCOV auf die Felder x, y, z direkt zugreift, werden diese Felder bei jeder Veränderung der Endpunkte neu berechnet. Die Float-Version `Vector3f` wird nicht unterstützt. Sie ist von der Schnittstelle gefordert, aber wird nur für das Setzen von Lichteinfall benutzt. Lichteinfall wird jedoch von dieser Library nicht unterstützt wird. `Vector3f` hat aus diesem Grund keine Funktionalität.

7.1.3. Package model.behaviors



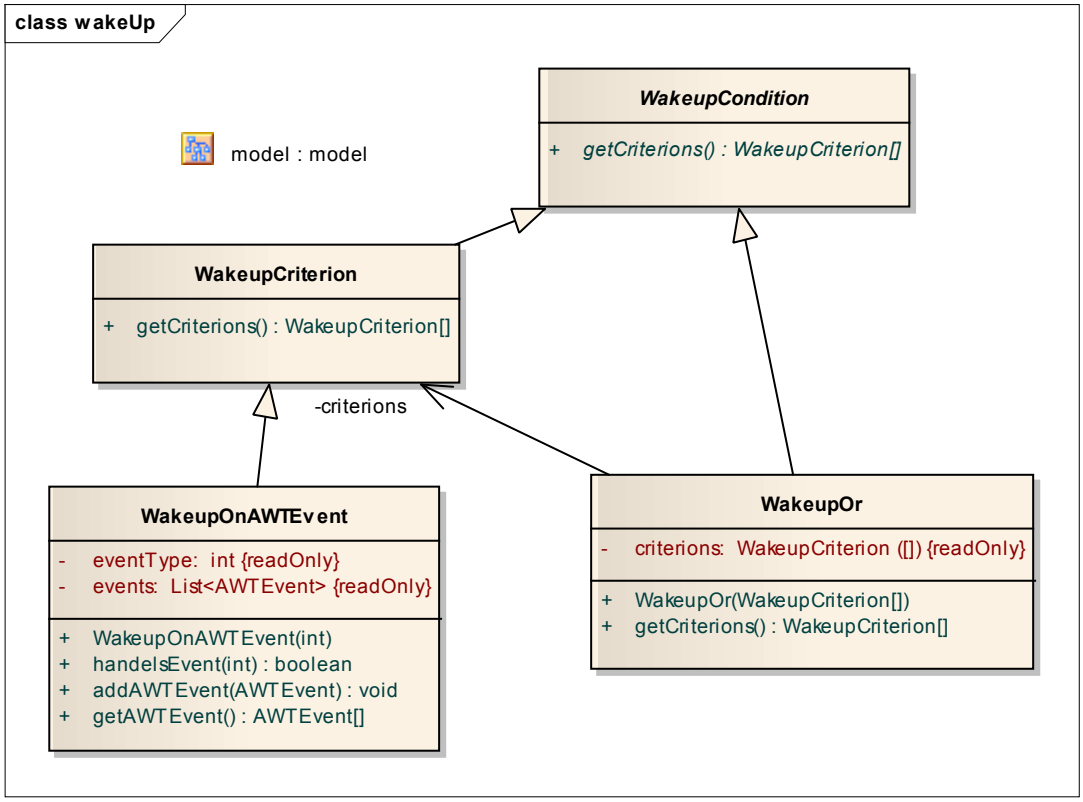
Die unterschiedlichen Verhaltenweisen, wie zum Beispiel das Rotieren per Maus oder Tastatur oder das Selektieren der Elemente wird über die Klasse `Behavior` implementiert. Dabei werden zwei Arten unterschieden:

Es gibt ein `Behavior`, welches für das gesamte Canvas gilt. Dieses wird durch die von der Schnittstelle vorgegebene abstrakte Klasse `ViewPlatformAWTBehavior` und der Implementation derselben, das `OrbitBehavior`, repräsentiert. `OrbitBehavior` wird in 3DCOV erweitert und dem `ViewPlatform` von `Canvas3D` hinzugefügt.

`Behavior`, welche nicht für das gesamte Canvas gelten, wie zum Beispiel das Selektieren einer einzelnen Box, werden der dazugehörigen `Group` hinzugefügt. Um ein Behavior zu erstellen, wird in 3DCOV eine neue Klasse erstellt, die von `Behavior` ableitet und das entsprechende Verhalten in der Methode processStimulus implementiert.

Bei welchen Events ein `Behavior` ausgelöst wird, wird über `WakeUpConditions` gesteuert. Diese befinden sich im Package wakeUp.

7.1.4. Package model.wakeUp



Alle Klassen, Methoden und Beziehungen in diesem Package sind von der Schnittstelle vorgegeben. Die Klassen `WakeupCriterion` und `WakeupCondition` wären in dieser Library nicht nötig, da nur eine Unterklasse vorhanden ist.

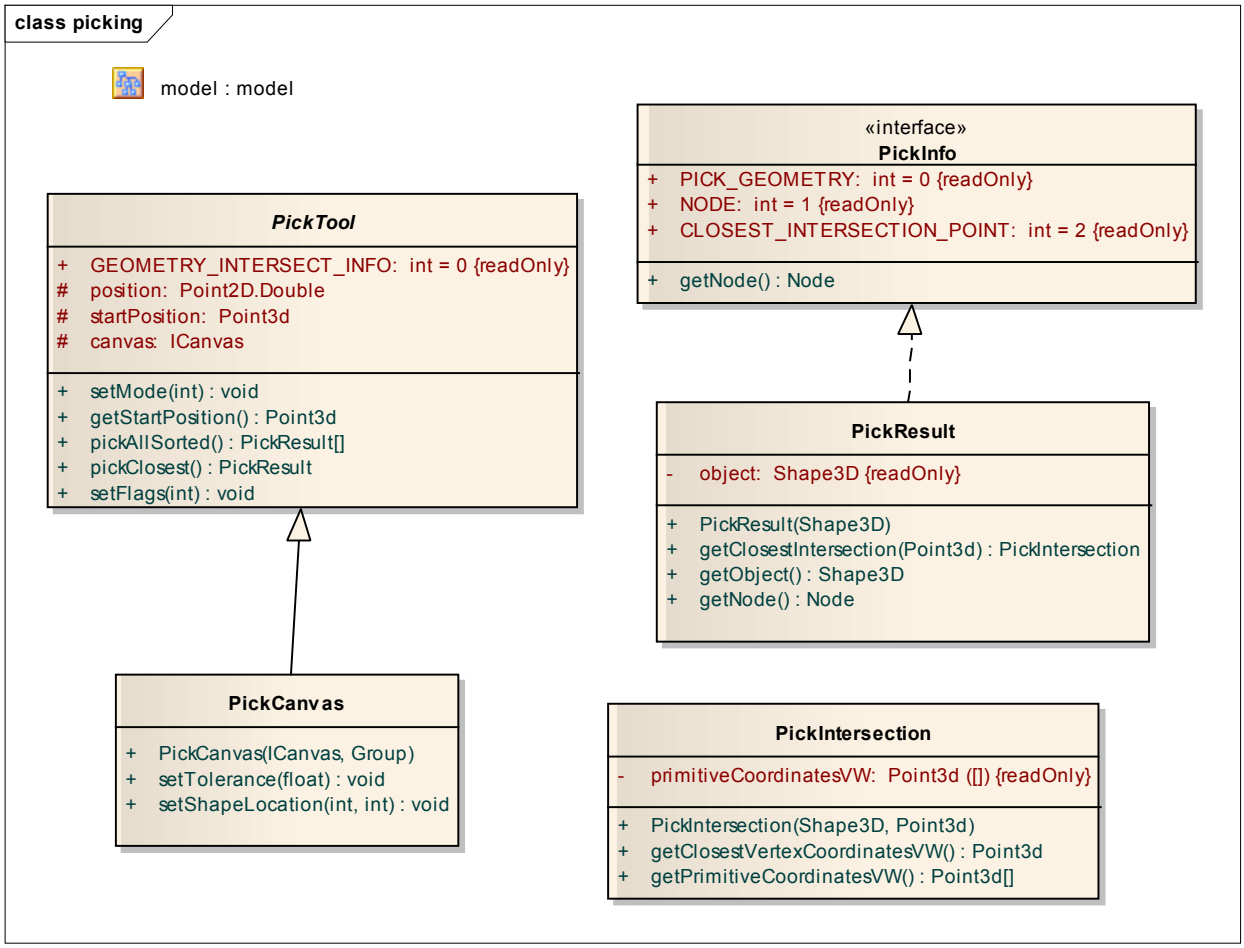
WakeupOnAWTEvent

Der Klasse `WakeupOnAWTEvent` wird im Konstruktor angegeben, auf welchen AWT-Event sie reagiert. Wenn ein Event auftritt, wird aus der Klasse `Canvas3D` die Methode addAWTEvent aufgerufen. Falls dieser Event dem im Konstruktor definierten Typ entspricht, wird der Event gespeichert. Bei der Ausführung der Behaviors werden die hinzugefügten Events über die Methode getAWTEvent zurückgegeben.

WakeupOr

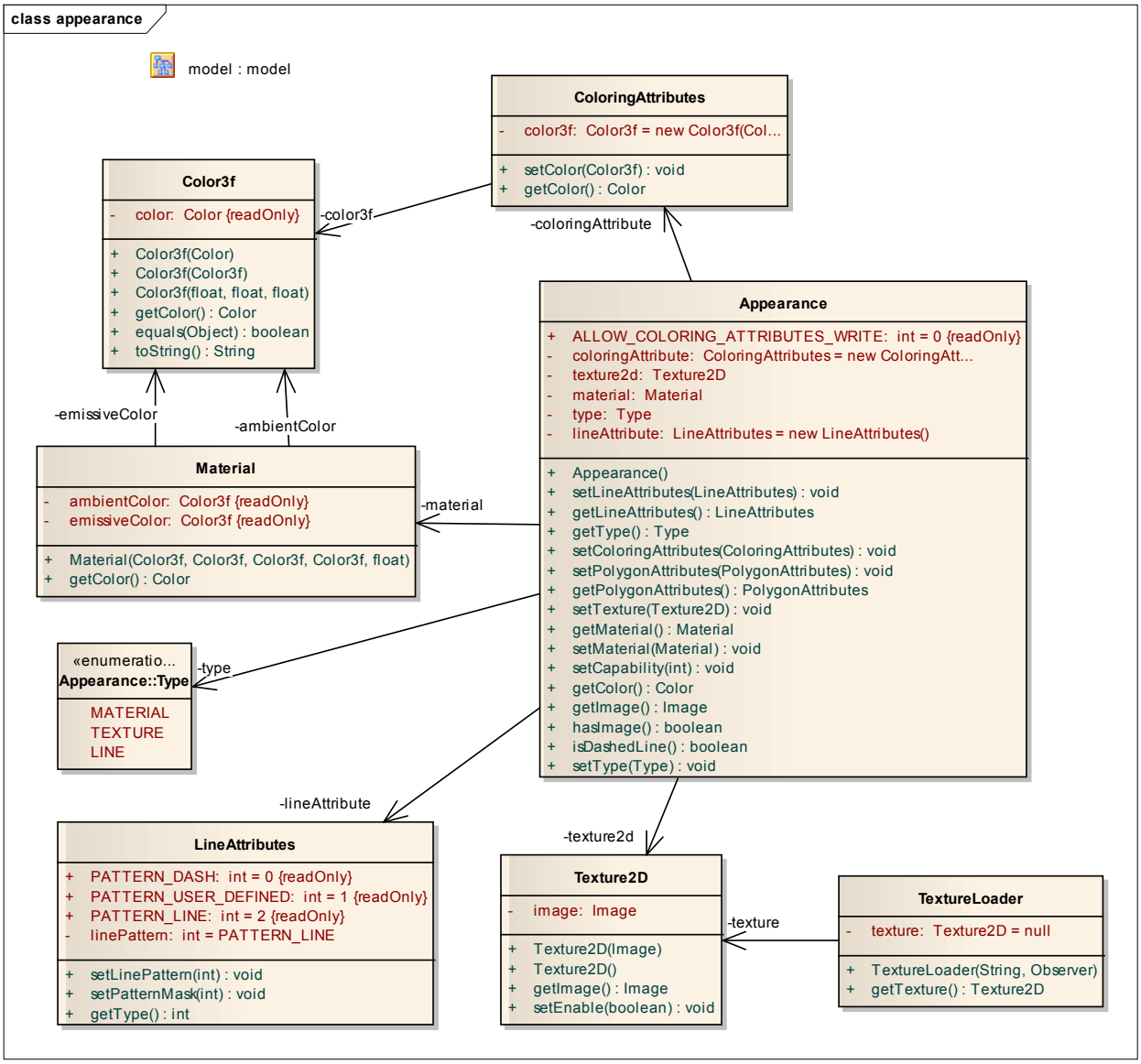
Über die Klasse `WakeupOr` können mehrere `WakeupCriterions` (in dieser Library nur `WakeupOnAWTEvents`) kombiniert werden.

7.1.5. Package model.picking



Die Klassen in diesem Package werden benutzt, damit die Behaviors von 3DCOV eine Anfrage stellen können, welche Objekte sich auf einer bestimmten Position befinden. Alle Klassen sind von der Schnittstelle vorgegeben.

7.1.6. Package model.appearance



In diesem Package befinden sich Klassen, welche für das Aussehen eines Objektes verantwortlich sind.

Color3f, Material, ColoringAttributes

Diese drei Klassen werden von der Schnittstelle vorgegeben. In der Library focusedj3d_on_j2d würde die Klasse java.awt.Color genügen.

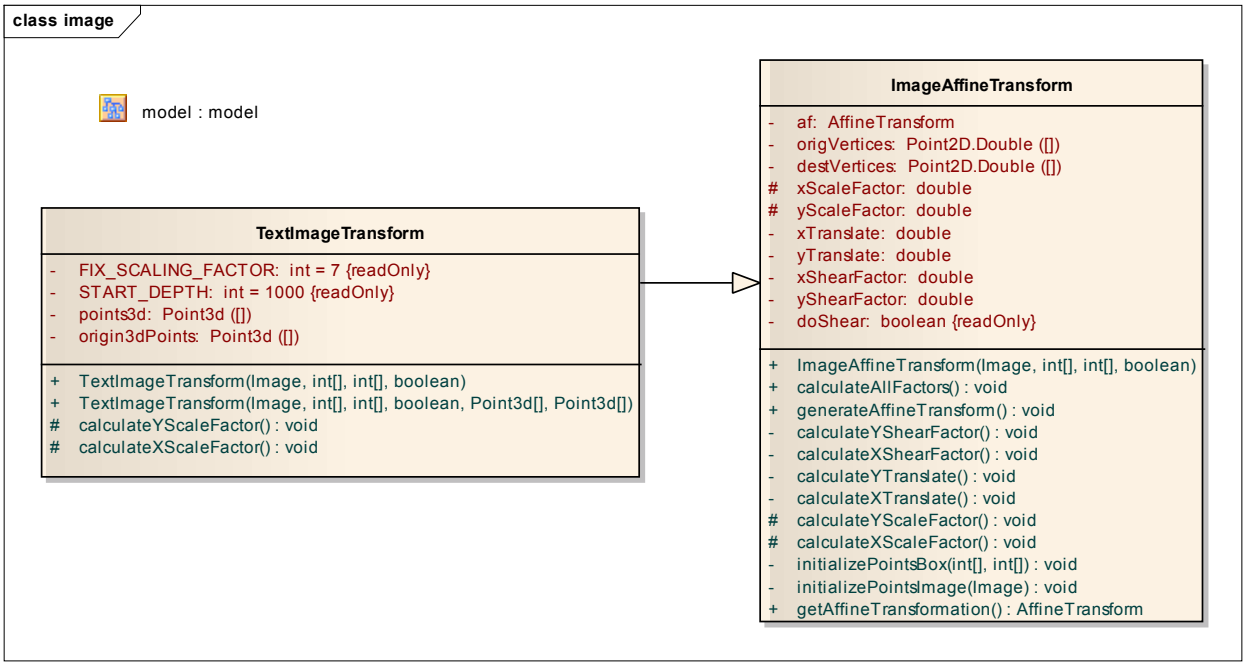
Texture2D

`Texture2D` wird verwendet um die Beschriftung einer `Box`, welche als Bild an die Library focusedj3d_on_j2d übergeben wird, zu speichern.

LineAttributes

`LineAttributes` wird verwendet um anzugeben, ob eine Linie durchgezogen oder gestrichelt gezeichnet werden soll.

7.1.7. Package model.image



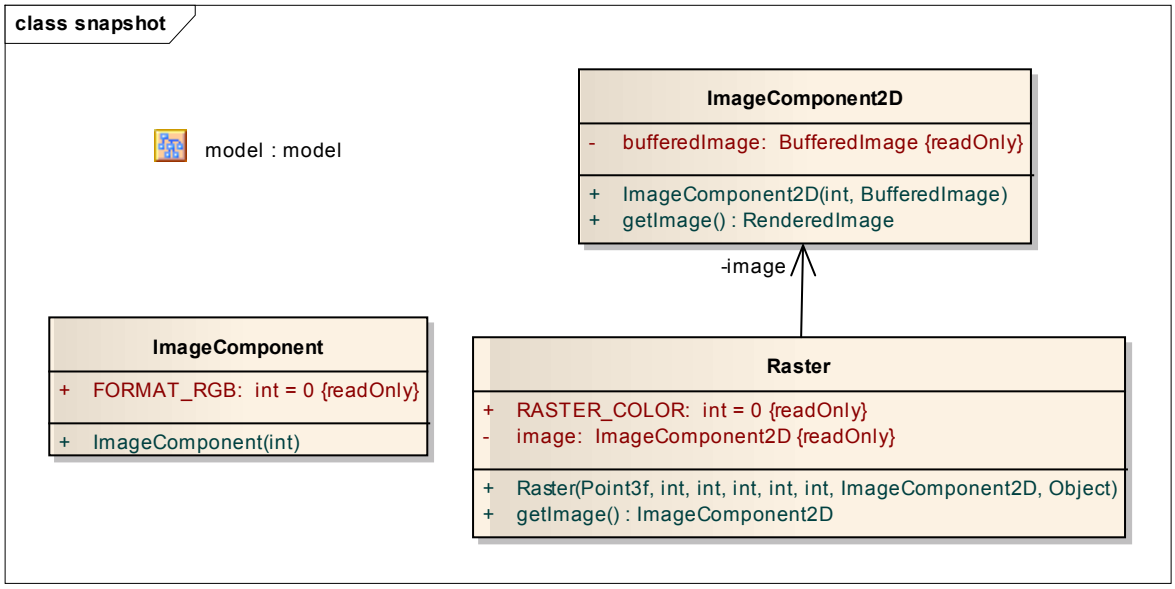
ImageAffineTransformation

`ImageAffineTransformation` ist für die Transformation eines java.awt.Image verantwortlich. Im Konstruktor werden die Zielkoordinaten (destVertices) übergeben, welche dem Polygon (`Shape3d`) entsprechen, auf die das Bild gezeichnet werden soll. Die Originalkoordinaten des Bildes werden im Feld origVertices gespeichert.

Da mit einer java.awt.geom.AffineTransform kein Polygon, sondern nur ein Parallelogramm erreicht werden kann, geht dabei die Perspektive verloren. Momentan wird die Grösse des Bildes mittels Clipping auf die Grösse der dazugehörigen `Shape3D` reduziert.

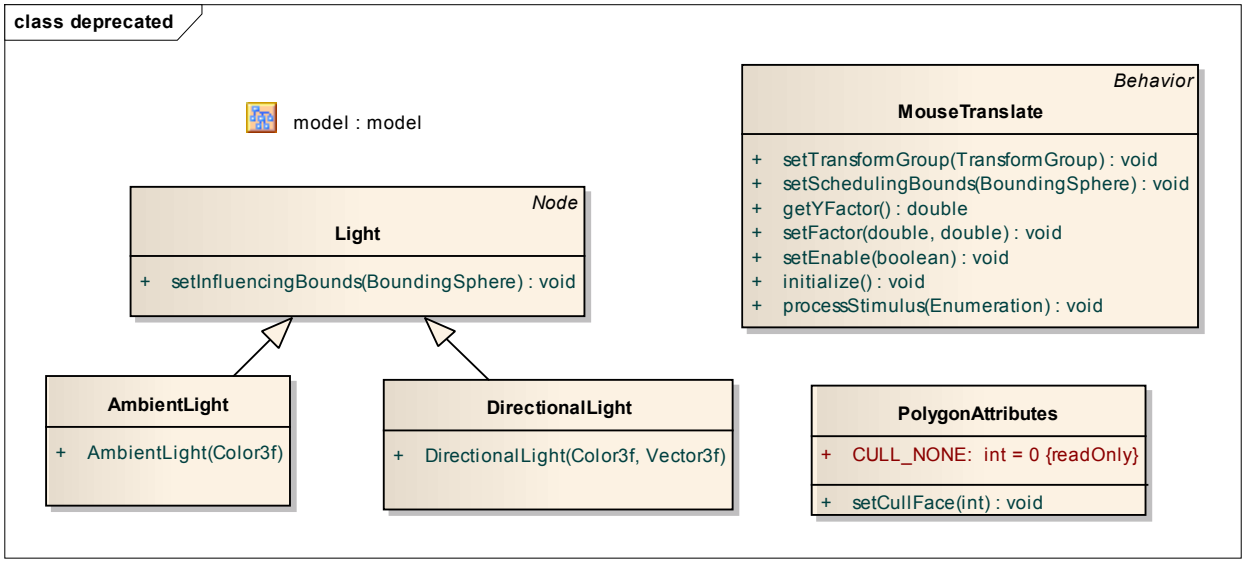
Eine Alternative dazu wäre, dass der 3DCOV so abgeändert wird, dass kein Bild mehr übergeben wird, sondern der Text. Da Text nicht transformiert werden kann, wäre es nötig eine eigene Schrift zu definieren, die aus geraden Linien besteht (z.B. eine Digit-Schrift). Dies könnte allerdings zu Performance-Problemen führen, da dadurch sehr viele Punkte berechnet und viele kleine Linien gezeichnet werden müssten.

7.1.8. Package model.snapshot



Im Package snapshot befinden sich diejenigen Klassen, die benötigt werden, um einen Snapshot des Canvas zu machen. Dabei hat man sich auf die Anforderungen von 3DCOV konzentriert, welche immer ein Bild des ganzen Canvas verlangt.

7.1.9. Package model.deprecated



In diesem Package befinden sich einige Klassen, welche von der Schnittstelle vorgegeben sind, jedoch in dieser Library keine Funktionalität zur Verfügung stellen.

Der Lichteinfall wird in dieser Library nicht unterstützt.

Projekt: J3DEval

Funktionsumfang Release

Version: 4.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
10.10.2011	1.0	Dokument erstellt und Funktionalitäten aufgelistet	Marion Walser
07.11.2011	2.0	Überarbeitung für Release 2	Lara Mühlemann
14.11.2011	2.0	Funktionalität „Tooltipp auf Objekt" ergänzt	Marion Walser
27.11.2011	2.0	Überarbeitung für Abgabe „Release 2"	Marion Walser
29.11.2011	3.0	Zwei weitere Funktionalitäten ergänzt	Marion Walser
05.12.2011	3.0	Überarbeitung für Release 3	Lara Mühlemann
11.12.2011	3.0	Überarbeitung für Abgabe „Release 3"	Marion Walser
21.12.2011	3.0	Bug mit Zoomen ergänzt	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Marion Walser verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
12.10.2011	Lara Mühlemann
07.11.2011	Marion Walser
06.12.2011	Marion Walser
19.12.2011	Marion Walser

1.4. Inhalt

2. Einführung

2.1. Zweck

Dieses Dokument beschreibt den Funktionsumfang der Releases. Dabei wird eine Auflistung der Funktionalitäten erstellt und Priorität und Status dieser aufgeführt. Zusätzlich werden (noch) nicht behobene Probleme festgehalten.

2.2. Gültigkeitsbereich

Dieses Dokument ist während der Projektdauer gültig.

2.3. Definitionen und Abkürzungen

Siehe Glossar.doc

2.4. Referenzen

[1] Aufgabenstellung.pdf

[2] Projektplan.doc

[3] Anforderungsspezifikation.doc

[4] Release1.doc

[5] Release2.doc

[6] Release3.doc

2.5. Redundanz

Dieses Dokument enthält Redundanz zu den Dokumenten Release1[4], Release2[5], Release3[6].

3. Funktionalitäten 3DModel

Folgend werden die Funktionalitäten aufgelistet, die implementiert werden müssen. Diese werden weitestgehend auf Ebene Applikation 3DCOV aufgelistet.

Die Priorität gibt den Release an, in dem die Funktionalität implementiert werden soll. Diejenigen Funktionalitäten mit „-„ werden voraussichtlich in diesem Projekt nicht umgesetzt.

Nr.	Priorität	Beschreibung Funktionalität	Status
1	1	Darstellung von drei 3D-Objekten im Raum	Erledigt in Release 1
2	1	Rotation 3D-Objekte	Erledigt in Release 1

3	2	Darstellung Klasse mittels Klasse „Box"	Erledigt in Release 1
4	2	Darstellung Objekt mittels Klasse „Box"	Erledigt in Release 1
5	2	Listeners von 3DCOV berücksichtigen	Erledigt in Release 2
6	2	Darstellung Abhängigkeit	Erledigt in Release 2
7	2	Darstellung Assoziation	Erledigt in Release 2
8	2	Darstellung Vererbung	Erledigt in Release 2
9	2	Darstellung Mehrfach-Vererbung	Erledigt in Release 2
10	2	Zoomen Diagramm	Erledigt in Release 2
11	2	Verschiebung Diagramm	Erledigt in Release 2
12	2	Beschriftung von Klassen und Objekten	Erledigt in Release 2
13	3	Wechseln der Hintergrund-Farbe (weiss – schwarz)	Erledigt in Release 3
14	3	Anzeigen der Achsen	Erledigt in Release 3
15	3	Synchrones Verschieben von Objekten in Klassendiagramm und 3D-Model	Erledigt in Release 2
16	3	Selektion und Löschen Objekt	Erledigt in Release 3
17	3	Mehrfach-Selektion und -Löschen Objekte	Erledigt in Release 3
18	-	Erstellen einer Objektassoziation über Kontext-Menu von Objekt	Erledigt in Release 3
19	-	Selektion und Löschen Assoziation	Erledigt in Release 3
20	3	Selektion aller Objekt einer Klasse mittels Kontext-Menu der Klasse im Klassendiagramm	Erledigt in Release 3
21	3	Verstecken/Anzeigen aller Objekte	Erledigt in Release 3
22	-	Darstellung Subset/Parallele Assoziationen bei Objekten	Erledigt in Release 3
23	-	Anzeige Assoziationsnamen und -rollen	Erledigt in Release 3
24	3	Anzeige reflexive Assoziationen	Erledigt in Release 2
25	-	Korrektes Positionieren + Skalierung gemäss Ausgangslage	Offen
26	3	Objektinformationen als Tooltipp anzeigen, wenn Maus über Objekt fährt	Erledigt in Release 3
27	3	Generierung eines Bildes für SnapShot	Erledigt in Release 2
28	3	Paint-Aufrufe optimieren	Erledigt in Release 3
30	3	Switchen zwischen Klassendiagramm und Objektdiagramm mit Ctrl+1,2,3	Erledigt in Release 3
31	3	Zurücksetzen des Konstrukt auf Startposition	Erledigt in Release 3
32	3	Ändern der Farbe der Klasse und Objekt aus Aufforderung von Klassendiagramm	Erledigt in Release 3
33	3	Ändern der Texture aus Aufforderung von Klassendiagramm	Erledigt in Release 3

4. Nicht behobene Probleme

Nr.	Priorität	Beschreibung Funktionalität	Status
1	3	Beschriftung der Klassen und Objekte ist etwas schlechter lesbar als bei der Java 3D-Version	Offen
2	3	Bei einem grossen Konstrukt mit vielen Objekten erfolgt Neuzeichnen aufbauend, Objekte erscheinen nacheinander	Offen, siehe nächstes Kapital
3	3	Beim extremen Zoomen in die Nähe verschwindet das Konstrukt und erscheint dann wieder, jedoch gespiegelt.	Offen

5. Beschreibung zur Umsetzung

Bezeichnung	**Beschreibung**
Korrektes Positionieren + Skalierung gemäss Ausgangslage (Nr. 25)	Werte zur Positionierung aus 3DCOV werden mehrheitlich nicht übernommen. Es wurde lediglich darauf geschaut, dass sich die Startposition etwa am gleichen Ort befindet.
Performance bzw. Neuzeichnen	Bei leistungsschwächeren Rechnern und bei grossen Diagrammen (grosse Anzahl von Klassen und Objekten) erfolgt das Neuzeichnen immer noch aufbauend, d.h. von den Klassen her nach oben. Auch ist die Reaktionszeit länger, da viele Kalkulationen ausgeführt werden müssen. Die Bedienung mit der Tastatur ist bei solchen Rechnern angenehmer. Bei starken Rechnern funktioniert das Bewegen des Konstrukts fast optimal.

Projekt: J3DEval

Release 1

Version: 2.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	**Version**	**Änderung**	**Autor**
31.10.2011	1.0	Erstellung des Dokuments und Auflistung der Funktionalitäten	Lara Mühlemann

1.2. Verantwortlichkeit

Für dieses Dokument ist Lara Mühlemann verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
01.11.2011	Lara Mühlemann, Marion Walser
19.12.2011	Lara Mühlemann

1.4. Inhalt

2. Einführung

2.1. Zweck

Dieses Dokument dient zur Dokumentation des Release 1.

2.2. Gültigkeitsbereich

Dieses Dokument ist während der Projektdauer gültig.

2.3. Definitionen und Abkürzungen

Siehe Glossar.doc

2.4. Referenzen

[1]Funktionsumfang_Release.doc

2.5. Redundanz

Dieses Dokument enthält Redundanz zum Dokument Funktionsumfang_Release[1].

3. Funktionalitäten

Folgend sind alle Funktionalitäten aufgelistet, welche gemäss Funktionsumfang_Release[1] in diesem Release geplant waren oder welche zusätzlich umgesetzt werden konnten.

Nr.	Funktionalitäten	Geplant	Status
1	Darstellung von drei 3D-Objekten im Raum		
	• Berechnung der Anfangspositionen der Eckpunkte im 3D Raum.	Geplant	Umgesetzt
	• Umrechnung der 3D Positionen zu 2D Positionen	Geplant	Umgesetzt
	• Sicherstellung, dass Objekte die weiter vorne liegen, die dahinterliegenden Objekte verdecken.	Geplant	Umgesetzt
	• Farben der Boxen können gesetzt werden	Nicht geplant	Teilweise umgesetzt
2	Rotation 3D-Objekte		
	• Horizontale Rotation	Geplant	Umgesetzt
	• Vertikale Rotation	Geplant	Umgesetzt
3	Darstellung Klasse mittels Klasse „Box"	Nicht geplant	Umgesetzt
4	Darstellung Objekt mittels Klasse „Box"	Nicht geplant	Umgesetzt

Kommentar zur Tabelle

Die Farben der Boxen können mithilfe der Klasse Appearance über setColor gesetzt werden. Noch nicht möglich ist das Setzen der Farbe mittels der Methode setMaterial.

Projekt: J3DEval

Release 2

Version: 2.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
07.11.2011	1.0	Erstellung des Dokuments und Auflistung der Funktionalitäten	Marion Walser
27.11.2011	1.0	Überarbeitung für Abgabe „Release 2"	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Marion Walser verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
28.11.2011	Lara Mühlemann
19.12.2011	Lara Mühlemann

1.4. Inhalt

2. Einführung

2.1. Zweck

Dieses Dokument dient zur Dokumentation des Release 2.

2.2. Gültigkeitsbereich

Dieses Dokument ist während der Projektdauer gültig.

2.3. Definitionen und Abkürzungen

Siehe Glossar.doc

2.4. Referenzen

[1]Funktionsumfang_Release.doc

2.5. Redundanz

Dieses Dokument enthält Redundanz zum Dokument Funktionsumfang_Release [1].

3. Funktionalitäten

Folgend sind alle Funktionalitäten aufgelistet, welche gemäss Funktionsumfang_Release[1] in diesem Release geplant waren oder welche zusätzlich umgesetzt werden konnten.

Nr.	Funktionalitäten	Geplant	Status
3	Darstellung Klasse mittels Klasse „Box"	Erledigt in Release 1	Umgesetzt
4	Darstellung Objekt mittels Klasse „Box"	Erledigt in Release 1	Umgesetzt
5	Listeners von 3DCOV berücksichtigen	Geplant	Umgesetzt
6	Darstellung Abhängigkeit	Geplant	Umgesetzt
7	Darstellung Assoziation	Geplant	Umgesetzt
8	Darstellung Vererbung	Geplant	Umgesetzt
9	Darstellung Mehrfach-Vererbung	Geplant	Umgesetzt
10	Zoomen Diagramm	Geplant	Umgesetzt
11	Verschiebung Diagramm	Geplant	Umgesetzt
12	Beschriftung von Klassen und Objekten	Geplant	Umgesetzt
15	Synchrones Verschieben von Objekten in Klassendiagramm und 3D-Model	Nicht geplant	Umgesetzt
24	Anzeige reflexive Assoziationen	Nicht geplant	Umgesetzt

27	Generierung eines Bildes für SnapShot	Nicht geplant	Umgesetzt

4. Nicht behobene Probleme

Nr.	Priorität	Beschreibung Funktionalität	Status
1	3	Beschriftung der Klassen und Objekte ist etwas schlechter lesbar als bei der Java 3D-Version	Offen
2	3	Texture mit Beschriftung der Klassen und Objekte ist nicht genau deckungsgleich mit der Box (Umformung durch Projektion fehlt)	Offen, evtl. Weiterbearbeitung in Release 3
3	3	Bei vielen Objekten erfolgt Neuzeichnen aufbauend, Objekte erscheinen nacheinander	Wird in Release 3 optimiert

Projekt: J3DEval

Release 3

Version: 1.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
03.12.2011	1.0	Dokument erstellt	Marion Walser
11.12.2011	1.0	Ergänzt, was umgesetzt wurde	Lara Mühlemann

1.2. Verantwortlichkeit

Für dieses Dokument ist Marion Walser verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
05.12.2011	Lara Mühlemann
19.12.2011	Lara Mühlemann

1.4. Inhalt

2. Einführung

2.1. Zweck

Dieses Dokument dient zur Dokumentation des Release 3.

2.2. Gültigkeitsbereich

Dieses Dokument ist während der Projektdauer gültig.

2.3. Definitionen und Abkürzungen

Siehe Glossar.doc

2.4. Referenzen

[1]Funktionsumfang_Release.doc

2.5. Redundanz

Dieses Dokument enthält Redundanz zum Dokument Funktionsumfang_Release[1].

3. Funktionalitäten

Folgend sind alle Funktionalitäten aufgelistet, welche gemäss Funktionsumfang_Release[1] in diesem Release geplant waren oder welche zusätzlich umgesetzt werden konnten.

Nr.	Funktionalitäten	Geplant	Status
13	Wechseln der Hindergrund-Farbe (weiss – schwarz)	Geplant	Umgesetzt
14	Anzeigen der Achsen	Geplant	Umgesetzt
15	Synchrones Verschieben von Objekten in Klassendiagramm und 3D-Model	Erledigt in Release 2	Umgesetzt
16	Selektion und Löschen Objekt	Geplant	Umgesetzt
17	Mehrfach-Selektion und -Löschen Objekte	Geplant	Umgesetzt
18	Erstellen einer Objektassoziation über Kontext-Menu von Objekt	offen	Umgesetzt
19	Selektion und Löschen Assoziationen	offen	Umgesetzt
20	Selektion aller Objekt einer Klasse mittels Kontext-Menu der Klasse im Klassendiagramm	Geplant	Umgesetzt
21	Verstecken/Anzeigen aller Objekte	Geplant	Umgesetzt
22	Darstellung Subset/Parallele Assoziationen bei Objekten	offen	Umgesetzt
23	Anzeige Assoziationsnamen und -rollen	offen	Umgesetzt
24	Anzeige reflexive Assoziationen	Erledigt in Release 2	Umgesetzt

25	Korrektes Positionieren + Skalierung gemäss Ausgangslage	offen	Teilweise umgesetzt
26	Objektinformationen als Tooltipp anzeigen, wenn Maus über Objekt fährt	Geplant	Umgesetzt
27	Generierung eines Bildes für SnapShot	Erledigt in Release 2	Umgesetzt
28	Paint-Aufrufe optimieren	Geplant	Umgesetzt
30	Switchen zwischen Klassendiagramm und Objektdiagramm mit Ctrl+1,2,3	Geplant	Umgesetzt
31	Zurücksetzen des Konstrukt auf Startposition	Geplant	Umgesetzt
32	Ändern der Farbe der Klasse und Objekt aus Aufforderung von Klassendiagramm	Geplant	Umgesetzt
33	Ändern der Texture aus Aufforderung von Klassendiagramm	Geplant	Umgesetzt

Projekt: J3DEval

Code-Reviews

Version: 3.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
31.10.2011	1.0	Erster Review dokumentiert	Marion Walser, Lara Mühlemann
24.11.2011	2.0	Review Memorized Translation	Lara Mühlemann
25.11.2011	2.0	Review für Release 2	Marion Walser, Lara Mühlemann
26.11.2011	2.0	Package-Namen aktualisiert	Marion Walser
12.12.2011	3.0	Review für Release 3	Lara Mühlemann
17.12.2011	3.0	Package-Namen aktualisiert	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Marion Walser verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
01.11.2011	Lara Mühlemann
28.11.2011	Lara Mühlemann

1.4. Abkürzungen

- mw: Marion Walser
- lm: Lara Mühlemann

1.5. Inhalt

2. Code-Reviews

2.1. Package ch.hsr.focusedj3d.view

Klasse: Canvas

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	Listener werden für Release 2 wieder entfernt und werden daher nicht refactored.	

Klasse: SimpleUniverse

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	Es werden nicht Listen von Canvas und BranchGroup geführt, da dies von 3DCOV nicht benötigt wird, Javadoc ergänzt	

Klasse: DrawableShapeArrayView

Datum / Reviewer	Kommentar	Refactoring
13.12.2011	Sortierung der Shape gehört nicht ins view package	Berechnung neu in DrawableObject

2.2. Package ch.hsr.focusedj3d.model.graph

Klasse: Shape3D

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	getLengthToViewPoint und CompareTo problematisch wegen duplicated code zu Klasse Box	TreeMap erstellt, welche sortiert. Dies erfolgt in den Klassen Primitive für Shapes und Group für Nodes(unter anderem die Boxen).

Klasse: Background

Datum / Reviewer	Kommentar	Refactoring
13.12.2011 lm	Soweit ok.	

Klasse: Group

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	getDrawableNodes umgestellt, so dass es die Box etc. sortiert werden (TreeMap)	

Klasse: GeometryArray

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	get2dPoint und getLengthToViewPoint von Shape3D weiter in GeometryArray verschieben	Move Method
	getLengthToViewPoint in Untermethoden unterteilt	Extract Method

Klasse: Node

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	Soweit ok	

Klasse: TransformGroup

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	Draw Methode, die nur super aufruft, gelöscht	

Klasse: BranchGroup

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	Compile wird in dieser Library nicht umgesetzt, da kein Caching über Retained-Objekte erfolgt. Daher wurde diese Methode auf Depricated gesetzt.	

Klasse: Primitive

Datum	Kommentar	Refactoring
31.10.2011 / mw_lm	drawPoly ist Long Method, soll eine Untermethoden erstellt werden und gehört in Shape3D	Extract Method / Move Method
31.10.2011 / mw_lm	get2dPoints wird verschoben von Primitive zu Point3d, da Featury Envy	Move Method
31.10.2011 / mw_lm	get2dPoints mehr selbsterklärend machen	Extract Method
31.10.2011 / mw_lm	For-Schleife in drawPoly über Shape-Punkte wird Shape3D verschoben und gibt ein zweidimensionales Array von int zurück	Extract Method

Klasse: Box

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	Verschiebung auf 3D-Koordination erklärend machen	Extract Local Variable / Extract Method
31.10.2011 / mw_lm	generateShapes unterteilt	Extract Method

2.3. Package ch.hsr.focusedj3d.model.geomath

Klasse: AxisAngle4f

Datum / Reviewer	Kommentar	Refactoring
13.12.2011 / lm	Methode get360Degree enthält nur fixe Zahlen	ersetzt durch Konstante
13.12.2011 / lm	Duplicated Code in addToX, addToY, addToZ	Extract Method. Da x/y/z keine Referenzen mit Rückgabewert.
13.12.2011 / lm	Cohesion könnte besser sein. Grund: getter-und add-Methoden. Keine Idee zur Verbesserung.	

Klasse: BoundingSphere

Datum / Reviewer	Kommentar	Refactoring
13.12.2011 / lm	Soweit ok	

Klasse: Bounds

Datum / Reviewer	Kommentar	Refactoring
13.12.2011 / lm	transform nimm ein Transform3D entgegen. Nutzte aber nur dessen Vector.	Umbenennen in translate. Übergabewert auf Vector3d geändert

Klasse: Transform3D

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	Soweit ok	

Klasse: Point3d

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	get2dPoints auf Einzahl get2dPoint gesetzt	Rename
13.12.2011 / lm	Kohäsion wird belassen	
13.12.2011 / lm	Methode rotate übernimmt ein Transform3D, braucht aber nur dessen AxisAngle + Methode ist public, wird aber nur in Point3d benutzt	Parameter geändert zu AxisAngle4f + zu private geändert
13.12.2011 / lm	Add(Point3d point) wird nicht benutzt	gelöscht

Klasse: Point3f

Datum / Reviewer	Kommentar	Refactoring
13.12.2011 / lm	Kohäsion wird belassen	

Klasse: Vector3d

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	Attribute x,y,z sind public, da von 3DCOV direkt zugegriffen wird. Problematisch, da in dem Moment kein Neuberechnung des Vektors erfolgt.	Nicht möglich ohne 3DCOV Code zu verändern

Klasse: MemorizedTranslation

Datum / Reviewer	Kommentar	Refactoring
24.11.2011 lm	Um Cohesion zu verbessern müssen 2 Unterklassen erstellt werden. Eine für die GroupTransformation und eine für die ObjectTransformation	Hat sich erledigt aufgrund der Verschiebung von der Speicherung der Transformation des Universe von Group zu Canvas
25.11.2011 / lm_mw	getTranslation wird nicht benutzt	Gelöscht
25.11.2011 / lm_mw	getVector3d() gehört in Klasse Transform3d	getVector3d() in Transform3D verschoben, damit getVector3d(Vector 3d) in Klasse Transform3D direkt aufgerufen
	Klasse wurde entfernt da nicht mehr gebraucht	

2.4. Package ch.hsr.focusedj3d.model.appearance

Klasse: Color3f

Datum / Reviewer	Kommentar	Refactoring
12.12.2011 lm	Soweit ok.	

Klasse: Appearance

Datum / Reviewer	Kommentar	Refactoring
12.12.2011 lm	Soweit ok. Diese Klasse hat keine hohe Kohäsion, dies ist allerdings durch die Schnittstelle gegeben.	

Klasse: ColoringAttributes

Datum / Reviewer	Kommentar	Refactoring
12.12.2011 lm	Soweit ok. Hat Smell „Lazy class" . Klasse kann aber nicht entfernt werden da von der Schnittstelle vorgegeben.	

Klasse: LineAttributes

Datum / Reviewer	Kommentar	Refactoring
12.12.2011 lm	Soweit ok.	

Klasse: Material

Datum / Reviewer	Kommentar	Refactoring
12.12.2011 lm	Soweit ok. Konstruktor enthält zuviele Parameter. Dies ist jedoch von der Schnittstelle vorgegeben.	

Klasse: Texture2D

Datum / Reviewer	Kommentar	Refactoring
12.12.2011 lm	Soweit ok. I	

2.5. Package ch.hsr.focusedj3d.model.image

Klasse: ImageAffineTransform

Datum / Reviewer	Kommentar	Refactoring
25.11.2011 / lm_mw	Klasse hat schlechte Kohesion	Refactoring würde Code unübersichtlicher machen

2.6. Package ch.hsr.focusedj3d.model.behaviors

Klasse: OrbitBehavior

Datum / Reviewer	Kommentar	Refactoring
25.11.2011 / mw_lm	ProcessMouseEvent mittels switch-case implementieren. Somit ist es übersichtlicher.	If-else in switch-case umwandeln

2.7. Package ch.hsr.focusedj3d.model.snapshot

Klasse: Raster

Datum / Reviewer	Kommentar	Refactoring
25.11.2011 / lm_mw	Zu viele Parameter im Konstruktor	Kein Refactoring möglich, da 3DCOV dies fordert, Javadoc geschrieben

2.8. Testpackage ch.hsr.focusedj3d.model.graph

Klasse: GeometryArrayTest

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	Soweit ok	

Klasse: PrimitiveTest

Datum / Reviewer	Kommentar	Refactoring
31.10.2011 / mw_lm	Soweit ok	

Projekt: J3DEval

Systemtests

Version: 3.0
Datum: 21. Dezember 2011

Von: Marion Walser, Lara Mühlemann

1. Dokumentinformationen

1.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
01.11.2011	1.0	Erstellung des Dokuments. Einführung und Testspezifikation für UseCase 01-03 geschrieben	Marion Walser + Lara Mühlemann
27.11.2011	2.0	Test-Abläufe für Release 2 erstellt Systemtest durchgeführt	Marion Walser
28.11.2011	2.0	Systemtest durchgeführt	Lara Mühlemann
02.12.2011	3.0	Test-Abläufe erweitert	Marion Walser
14.12.2011	3.0	Test-Abläufe erweitert	Marion Walser
16.12.2011	3.0	Systemtest durchgeführt	Lara Mühlemann + Marion Walser
19.12.2011	3.0	Systemtest durchgeführt	Lara Mühlemann
19.12.2011	3.0	Systemtest mit Laptop1 ausgeführt.	Marion Walser

1.2. Verantwortlichkeit

Für dieses Dokument ist Marion Walser verantwortlich.

1.3. Überprüfung

Datum	Durchgeführt von
01.11.2011	Lara Mühlemann
28.11.2011	Lara Mühlemann
19.12.2011	Marion Walser

1.4. Inhalt

2. Einführung

2.1. Zweck

In diesem Dokument werden die Systemtests definiert und die Resultate der Tests dokumentiert.

2.2. Gültigkeitsbereich

Dieses Dokument ist während der Projektdauer gültig.

2.3. Definitionen und Abkürzungen

Siehe Glossar.doc

2.4. Referenzen

[1]SoftwareArchitectureDocument.doc

3. Voraussetzung

Es werden für die Test die folgenden Computer verwendet:

Computer	Ausrüstung
Laptop1	Packard Bell, 4 GB RAM, Intel® Core™2 Solo CPU U355 1.40 GHz,32 Bit-Betriebssystem, Windows Vista Home Premium, Java Version 6
Laptop2	Mysn, 4 GB RAM, Intel® Core™2 Duo CPU P9700 2.80 GHz,64 Bit-Betriebssystem, Windows 7 Professional, Java Version 6
PC	FUJITSU, 3.7 GB RAM, 4 Intel® Core™ i5 CPU @ 3.33 GHz, Kernel Linux 2.6.35.6-45.fc14.x86_64

4. Testspezifikation

4.1. UseCase01: Programm starten

4.1.1. Voraussetzungen

Keine Voraussetzungen

4.1.2. Vorbereitungen

Keine Vorbereitung

4.1.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender startet die Applikation 3DCOV		Ein Fenster erscheint, in dem zwei grössere leere Fenster sichtbar sind. Das linke hat einen weissen Hintergrund, das rechte einen schwarzen.

4.2. UseCase02: Box erstellen

Um das Erstellen von Boxen intensiver zu testen wird nicht nur der Use Case mit einer Box getestet, sondern der Testfall wird auf 3 Boxen erweitert. Zwei Boxen repräsentieren eine Klasse, eine Box repräsentiert ein Objekt.

4.2.1. Voraussetzungen

Applikation ist gestartet, die zwei Diagramme enthalten keine Objekte.

4.2.2. Vorbereitungen

UseCase01 wurde ausgeführt.

4.2.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender drückt auf den Button „New Class".		Ein neues, kleines Fenster öffnet sich, in dem die Daten für die neue Klasse eingegeben werden können.
T02	Der Anwender gibt die Klassendaten ein und drückt auf den OK-Button.	Klassenname: A Attribut: a	Kleines Fenster wird geschlossen und je eine blaue Klasse im Klassendiagramm und im Objektdiagramm erscheinen. Beide enthalten den Klassennamen A und das Attribut a.
T03	Der Anwender wiederholt T02 mit weiteren Klassendaten	Klassenname: B Attribut: b	Gleiches Verhalten wie in T02 wird erwartet, jedoch mit den anderen Testdaten und die Klasse erscheint grün. Zusätzlich überdeckt die Klasse B die Klasse A leicht.
T04	Der Anwender verschiebt die Klasse B mit Drag'n'Drop etwas nach rechts, so dass die Klasse A nicht mehr überdeckt wird.		In beiden Diagrammen verschiebt sich die Klasse B nach rechts.
T05	Der Anwender erstellt ein Objekt mit dem Kontext-Menü der Klasse A im Objektdiagramm und der Auswahl des Menüpunkt „Create Object".		Fenster „New Object" öffnet sich.
T06	Der Anwender gibt die Objektdaten ein und drückt auf den OK-Button.	Objektname: aObject Wert für Attribut a: 1011	Fenster „New Object" verschwindet. Im Objektdiagramm erscheint ein blaues Objekt mit dem unterstrichenen Text „aObject:A" und mit nicht unterstrichenem Text eine Zeile weiter unten „a = 1011". Dies ist in der Erstansicht nicht erkennbar, da die Perspektive den Text verzerrt.

4.3. UseCase03: Konstrukt rotieren

4.3.1. Voraussetzungen

Applikation ist gestartet, die zwei Diagramme enthalten keine Objekte.

4.3.2. Vorbereitungen

UseCase01 wurde ausgeführt.

4.3.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender erstellt eine neue Klasse.	Klassenname: Z	Eine Klasse erscheint im Klassendiagramm und im Objektdiagramm
T02	Der Anwender lässt die Klasse rotieren.		Die einzelne Klasse dreht nur um sich selbst.
T03	Der Anwender führt den TestCase für UseCase02 aus.		Das Konstrukt wird um die Klassen und Objekte gemäss TestCase erweitert.
T04	Der Anwender zieht mit gedrückter, linker Maustaste auf dem Objektdiagramm von links nach rechts, bis sich das Konstrukt um 90 Grad gedreht hat.		Das Konstrukt dreht sich um etwa 90 Grad um die Ordinaten-Achse im Gegenuhrzeigersinn.
T05	Der Anwender zieht mit gedrückter, linker Maustaste auf dem Objektdiagramm von unten nach oben, bis sich das Konstrukt um 180 Grad gedreht hat.		Das Konstrukt dreht sich um etwa 180 Grad, wobei sich die Vorderseite zuerst nach oben bewegt.
T06	Der Anwender drückt eine Pfeil Taste		Das Konstrukt dreht sich je nach gedrückter Pfeiltaste. Die Vordseite des

	Konstrukts bewegt sich in die selbe Richtung wie die Pfeiltaste.

4.4. UseCase04: Beziehungen erstellen

4.4.1. Voraussetzungen

Das Konstrukt enthält drei Klassen A, B und C.

4.4.2. Vorbereitungen

Die drei Klassen wurden erstellt.

4.4.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender drückt auf den Button **„New Assoziation"**. Dann drückt er die linke Maustaste, wenn die Maus auf der Klasse B ist und belässt die Maustaste gedrückt. Er schiebt die Maus zur Klasse C und lässt die Maustaste dann wieder los.		Eine Beziehung erscheint im Klassendiagramm und im Objektdiagramm zwischen Klasse B von C.
T02	Der Anwender drückt auf den Button **„New Dependency".** Dann drückt er die linke Maustaste, wenn die Maus auf der Klasse C ist und belässt die Maustaste gedrückt. Er schiebt die Maus zur Klasse A und lässt die Maustaste dann wieder los.		Eine Abhängigkeit erscheint im Klassendiagramm und im Objektdiagramm zwischen Klasse B von C.
T03	Der Anwender drückt auf den Button **„New Generalization".** Dann drückt er die linke		Eine Generalisierungsbeziehung erscheint im Klassendiagramm und im Objektdiagramm, die

	Maustaste, wenn die Maus auf der Klasse A ist und belässt die Maustaste gedrückt. Er schiebt die Maus zur Klasse B und lässt die Maustaste wieder los.	die Klasse A von B ableiten lässt.
T04	Der Anwender klickt auf die Mitte der Generalisierungsbeziehung, drückt rechte Maustaste und wählt aus dem Kontext-Menü „Change Relation Type" – „**Association**" aus.	Die Generalisierungsbeziehung wechselt im Klassendiagramm und im Objektdiagramm in eine Assoziation.
T05	Der Anwender klickt auf die Mitte der Generalisierungsbeziehung, drückt rechte Maustaste und wählt aus dem Kontext-Menü „Change Relation Type" – „**Dependency**" aus.	Die Generalisierungsbeziehung wechselt im Klassendiagramm und im Objektdiagramm in eine Abhängigkeit.
T06	Der Anwender klickt auf die Mitte der Generalisierungsbeziehung, drückt rechte Maustaste und wählt aus dem Kontext-Menü „Change Relation Type" – „**Generalisation**" aus.	Die Generalisierungsbeziehung wechselt im Klassendiagramm und im Objektdiagramm in eine Generalisierung.
T07	Der Anwender klickt auf die Mitte der Generalisierungsbeziehung, drückt rechte Maustaste und wählt aus dem Kontext-Menü „Change Relation Type" – „**Aggregation**" aus.	Die Generalisierungsbeziehung wechselt im Klassendiagramm und im Objektdiagramm in eine Aggregation.
T08	Der Anwender klickt auf die Mitte der Generalisierungsbeziehung, drückt rechte Maustaste und wählt aus dem Kontext-Menü „Change Relation Type" – „**Composition**" aus.	Die Generalisierungsbeziehung wechselt im Klassendiagramm und im Objektdiagramm in eine Komposition.
T09	Der Anwender drückt auf den Button **„**New Assocation**".** Dann drückt er die linke Maustaste, wenn die Maus auf der Klasse C ist und lässt die	Es erscheint eine **reflexive Assoziation** im Klassendiagramm und im Objektdiagramm für Klasse C.

	Maustaste gleich wieder los.

4.5. UseCase05: Konstrukt verschieben

4.5.1. Voraussetzungen

Drei Boxen (Klassen und Objekte) sind erstellt.

4.5.2. Vorbereitungen

UseCase02 wurde ausgeführt.

4.5.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender zieht mit gedrückter, rechter Maustaste auf dem Objektdiagramm von links nach rechts, von oben nach unten etc.		Das Konstrukt verschiebt sich entsprechend der Richtung der Maus.
T02	Der Anwender drückt die Ctrl-Taste und eine Pfeiltaste.		Das Konstrukt verschiebt sich entsprechend der gedrückten Pfeiltaste.

4.6. UseCase06: Konstrukt zoomen

4.6.1. Voraussetzungen

Drei Boxen (Klassen und Objekte) sind erstellt.

4.6.2. Vorbereitungen

UseCase02 wurde ausgeführt.

4.6.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender dreht das Mausrädchen auf dem Objektdiagramm hoch und runter.		Das Konstrukt verkleinert und vergrössert sich.
T02	Der Anwender drückt die Alt-Taste und die Pfeil-Hoch-Taste. Danach drückt er die Alt-Taste und die Pfeil-nach-unten-Taste.		Das Konstrukt verkleinert sich zuerst und danach vergrössert es sich.
T03	Drückt die mittlere Maustaste und verschiebt die Position der Maus hoch und runter.		Das Konstrukt verkleinert und vergrössert sich.

4.7. UseCase07: Objekte mit Vererbung erstellen

4.7.1. Voraussetzungen

Zwei Klassen A und B sind erstellt, wobei die Klasse B von A erbt.

4.7.2. Vorbereitungen

UseCase04 wurde ausgeführt.

4.7.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender erstellt ein Objekt der Klasse B, indem er auf der Klasse B die rechte Maustaste drückt und vom Kontext-Menü „Create Object" auswählt.		Fenster „New Object" öffnet sich.
T02	Der Anwender gibt den Objektnamen ein und drückt „OK".	Objektnamen: bObjekt	Im Objektdiagramm wird ein Objekt erstellt, das mittels zwei aufeinander stehenden Boxen dargestellt wird. Die

	untere Box hat die Farbe der Klasse B und die obere Box die Farbe der Klasse A.

4.8. UseCase08: Wechseln der Hintergrund-Farbe

4.8.1. Voraussetzungen

Mind zwei Klassen A und B sind erstellt, wobei die Klasse B eine Beziehung zu A hat.

4.8.2. Vorbereitungen

UseCase02 wurde ausgeführt.

4.8.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender wählt unter „Settings" „Change Background Color" aus.		Die Hintergrund-Farbe des Objektdiagramms wechselt von schwarz auf weiss. Die Farben der Klassen bleiben gleich.
T02	Der Awender wählt nochmals „Change Background Color" unter „Settings" aus.		Die Hintergrund-Farbe des Objektdiagramms wechselt zurück auf schwarz.

4.9. UseCase09: Anzeigen der Achsen

4.9.1. Voraussetzungen

Mind zwei Klassen A und B sind erstellt, wobei die Klasse B eine Beziehung zu A hat.

4.9.2. Vorbereitungen

UseCase02 wurde ausgeführt.

4.9.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender wählt unter „Settings" „Show Model Axis" aus.		Die drei Achsen werden im Objektdiagramm rot angezeigt.

4.10. UseCase10: Synchrones Verschieben im Klassen- und Objektdiagramm

4.10.1. Voraussetzungen

Mind zwei Klassen A und B und ein Objekt zu Klasse A sind erstellt, wobei die Klasse B eine Beziehung zu A hat.

4.10.2. Vorbereitungen

UseCase02 wurde ausgeführt.

4.10.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender verschiebt im Klassendiagramm die Klasse A in die untere rechte Ecke.		Die Klasse A und deren Objekt verschiebt sich im Objektdiagramm entsprechend der Verschiebung im Klassendiagramm. Die Beziehungslinie führt weiterhin von Klasse

	A nach Klasse B.

4.11. UseCase11: Selektion und Löschen Objekte

4.11.1. Voraussetzungen

Mind zwei Klassen A und B und ein Objekt zu Klasse A sind erstellt, wobei die Klasse B eine Beziehung zu A hat.

4.11.2. Vorbereitungen

UseCase02 wurde ausgeführt.

4.11.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender selektiert mit der Maus ein Objekt der Klasse A im Objektdiagramm.		Das selektierte Objekt verschwindet aus dem Objektdiagramm.

4.12. UseCase12: Erstellen einer Objektassoziation

4.12.1. Voraussetzungen

Mind zwei Klassen A und B und zwei Objekte zu Klasse A und ein Objekt zu Klasse B sind erstellt, wobei die Klasse B eine Beziehung zu A hat.

4.12.2. Vorbereitungen

UseCase02 wurde ausgeführt sowie ein Objekt der Klasse B erstellt.

4.12.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten

T01	Der Anwender wählt den Button „New Association" oberhalb des Objektdiagramms aus, fährt mit der Maus zum unteren Objekt der Klasse A, drückt die linke Maustaste und zieht die Maus mit gedrückter Taste zum Objekt der Klasse B. Dann lässt er die Maustaste los.		Zwischen dem Objekt der Klasse A und dem Objekt der Klasse B erscheint eine Linie.
T02	Der Anwender fährt mit der Maus zum oberen Objekt der Klasse A, drückt die rechte Maustaste und wählt aus dem Kontext-Menü „New Association". Danach klickt er auf dasselbe Objekt, drückt die linke Maustaste, die er gedrückt hält bis zum Objekt der Klasse B und dann die Maustaste loslässt..		Es erscheint eine Assoziation zwischen dem oberen Objekt der Klasse A und dem Objekt der Klasse B.

4.13. UseCase13: Selektion und Löschen Assoziationen

4.13.1. Voraussetzungen

Mind zwei Klassen A und B und zwei Objekte zu Klasse A und ein Objekt zu Klasse B sind erstellt, wobei die Klasse B eine Beziehung zu A hat. Es besteht mind. eine Objektbeziehung zwischen den zwei ersten Objekten der beiden Klassen.

4.13.2. Vorbereitungen

UseCase12 wurde ausgeführt.

4.13.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender wählt eine Objektbeziehung aus und drückt die Taste „Delete"		Die ausgewählte Beziehung zwischen den Objekten verschwindet.

4.14. UseCase14: Verstecken und Anzeige aller Objekte

4.14.1. Voraussetzungen

Mind zwei Klassen A und B und zwei Objekte zu Klasse A und ein Objekt zu Klasse B sind erstellt, wobei die Klasse B eine Beziehung zu A hat. Es besteht mind. eine Objektbeziehung zwischen den zwei ersten Objekten der beiden Klassen.

4.14.2. Vorbereitungen

UseCase12 wurde ausgeführt.

4.14.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender wählt den Button „HideObjects" oberhalb des Objektdiagramms an.		Die Objekte der Klasse A und B verschwinden sowie die Objektbeziehungen.
T02	Der Anwender wählt den Button „HideObjects" oberhalb des Objektdiagramms nochmals an.		Die Objekte der Klasse A und B erscheinen wieder sowie die Objektbeziehungen.

4.15. UseCase15: Darstellung Subset/Parallele Assoziationen

4.15.1. Voraussetzungen

Mind. zwei Klassen A und B sind erstellt, wobei die Klasse B eine Generalisierung und eine Assoziation zu A hat.

4.15.2. Vorbereitungen

UseCase02 wurde ausgeführt sowie eine weitere Beziehung zwischen Klasse A und B erstellt.

4.15.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender wählt im Klassendiagramm die Assoziation aus, drückt die linke Maustaste und wählt aus dem Kontext-Menü „Restriction" aus. Dann drückt er auf die Assoziation und fährt mit gehaltener rechter Maustaste zur Generalisierung.		Eine gestrichelte Verbindung zwischen der Generalisierung- und der Assoziationslinie erscheint im Objektdiagramm, ebenso im Klassendiagramm mit dem Name {subsets}.

4.16. UseCase16: Anzeige Assoziationsnamen und -rollen

4.16.1. Voraussetzungen

Mind zwei Klassen A und B sind erstellt.

4.16.2. Vorbereitungen

UseCase02 wurde ausgeführt.

4.16.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender erstellt eine Assoziation zwischen den Klassen. Danach drückt er zweimal auf die Linie der Assoziation und es erscheint Feld, in dem er den Assoziationsnamen eingibt.	Assoziationsnamen: enthält ein	Im Objektdiagramm erscheint der Name der Assoziation bei der Assoziationslinie.

4.17. UseCase17: Objektinformation als Tooltip anzeigen

4.17.1. Voraussetzungen

Mind zwei Klassen A und B und ein Objekt der Klasse A mit Namen sind erstellt.

4.17.2. Vorbereitungen

UseCase02 wurde ausgeführt.

4.17.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender fährt mit der Maus über das Objekt der Klasse A		Der Name des Objekts sowie eventuelle Attribute und Attributwerte erscheinen in weisser Schrift in der Nähe des Objekts im Objektdiagramm.

4.18. UseCase18: Generierung eines Snapshots

4.18.1. Voraussetzungen

Mind zwei Klassen A und B und ein Objekt der Klasse A mit Namen sind erstellt.

4.18.2. Vorbereitungen

UseCase02 wurde ausgeführt.

4.18.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender drückt den Button „Snap Shot"		Es erscheint ein Fenster „Save Snap Shot".

	oberhalb des Objektdiagramms.		
T02	Der Anwender gibt den Namen der Datei ein, in der er den Snapshot abspeichern will. Er merkt sich den Ort, wo die Datei abgespeichert wird.	Dateiname: Test.png	Das Fenster verschwindet wieder.
T03	Der Awender schaut sich den Snapshot in dem Verzeichnis an.		In der png-Datei wurde das aktuelle Bild des Objektdiagramms abgebildet.

4.19. UseCase19: Switchen zwischen Klassendiagramm und Objektdiagramm

4.19.1. Voraussetzungen

Mind zwei Klassen A und B und ein Objekt der Klasse A sind erstellt.

4.19.2. Vorbereitungen

UseCase02, UseCase04, UseCase07 und UseCase15 wurde ausgeführt.

4.19.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender drückt Ctrl+1, um nur noch das Klassendiagramm zu sehen.		Das Objektdiagramm verschwindet und nur noch das Klassendiagramm ist zu sehen.
T02	Der Anwender drückt Ctrl+2, um nur noch das Objektdiagramm zu sehen.		Das Klassendiagramm verschwindet und das Objektdiagramm wird gross angezeigt. Das Konstrukt wurde an die Startposition zurückgesetzt.
T03	Der Anwender verschiebt das Konstrukt im Objektdiagramm etwas nach oben und dreht es		

	etwas im Uhrzeigersinn.		
T04	Der Anwender drückt Ctrl+3, um nur noch das Objektdiagramm zu sehen.		Es werden wieder beide Diagramme dargestellt. Das Konstrukt im Objektdiagramm ist wieder auf der Startposition.

4.20. UseCase20: Exportieren und Importieren von XML-Dateien

4.20.1. Voraussetzungen

Diverse Klassen, Objekte und Beziehungen sind erstellt.

4.20.2. Vorbereitungen

UseCase02 wurde ausgeführt.

4.20.3. Testspezifikation

Nr.	Aktion	Testdaten	Erwartetes Verhalten
T01	Der Anwender wählt „Save Project" über den Eintrag „Projekt" in der Menüleiste aus.		Ein Fenster erscheint, in dem der Dateiname eingegeben werden kann.
T02	Der Anwender gibt den Dateinamen ein, merkt sich das Verzeichnis und drückt „Save".	Test.xml	Das Fenster schliesst sich und eine Datei in dem im Fenster angezeigten Namen wird erstellt.
T03	Der Anwender schliesst die Applikation und startet sie neu.		
T04	Der Anwender wählt „Load Project" über den Eintrag „Projekt" in der Menüleiste aus.		Ein Fenster erscheint, in dem die vorher exportierte Datei aufgelistet wird.

T05	Der Anwender wählt die Datei „Test.xml" aus und drückt „Save".		Das vorher exportierte Diagramm wird im Klassendiagramm und im Objektdiagramm dargestellt.
T06	Der Anwender wählt „Load Project" über den Eintrag „Projekt" in der Menüleiste aus und wählt als zu importierende Datei „Clubschule.xml" aus.		Das Beispiel-Projekt wird geladen und sauber dargestellt.
T07	Der Anwender erstellt eine neue Klasse.	Klassennamen: Ausrüstung	Die Klasse wird zusätzlich im Klassendiagramm und Objektdiagramm dargestellt.
T08	Der Anwender rotiert und verschiebt das Konstrukt		Das Konstrukt rotiert und transformiert wie in UseCase03 und 05

5. Ausführungen

Die Ausführungen werden nach Ausführungsdatum und fortlaufender Nummer strukturiert.

5.1. Ausführung von 27.11.2011

Für Ausführung der Tests wurde Laptop1 benutzt.

Test-Nr.	Status (Gut, OK, Fehler)	Fehler/Unschönheiten	Mögliche Verbesserungen
UC01	Gut	Keine	Keine
UC02	OK	Klassen erscheinen sehr klein, in einem grossen Winkel zum Betrachter und mit der Unterseite nach oben	Startposition besser setzen
UC03	Fehler	Nach oben und unten rotiert das Konstrukt in die falsche Richtung	Keine
UC04	Gut	Keine	Keine
UC05	Gut	Keine	Keine
UC06	Gut	Keine	Keine

5.2. Ausführung von 28.11.2011

Für Ausführung der Tests wurde Laptop2 benutzt.

Test-Nr.	Status (Gut, OK, Fehler)	Fehler/Unschönheiten	Mögliche Verbesserungen
UC01	Gut	Keine	Keine
UC02	OK	Klassen erscheinen sehr klein, in einem grossen Winkel zum Betrachter und mit der Unterseite nach oben	Startposition besser setzen.
UC03	Gut	Keine	Keine
UC04	Gut	Keine	Keine
UC05	Gut	Keine	Keine
UC06	Gut	Keine	Keine

5.3. Ausführung von 16.12.2011

Für Ausführung der Tests wurde PC benutzt.

Test-Nr.	Status (Gut, OK, Fehler)	Fehler/Unschönheiten	Mögliche Verbesserungen
UC01	Gut	Buttons von 3DCOV werden zum Teil abgeschnitten	
UC02	OK	Konstrukt hüpft beim Erstellen des Objekts	Verbesserungsmöglichkeit ist im Dokument SoftwareArchtectureDocument [1] aufgeführt.
UC03	Gut		
UC04	Gut		
UC05	Gut		
UC06	Gut		
UC07	Gut		
UC08	Gut		
UC09	Gut		
UC10	OK	Das ganze Konstrukt verschiebt sich zum Teil an eine andere Position	Siehe Verbesserungsmöglichkeit UC02
UC11	Gut		
UC12	Gut		
UC13	Gut		
UC14	Gut		
UC15	Gut		
UC16	Gut		
UC17	Gut		
UC18	Gut		
UC19	Gut		
UC20	OK	Beim Rotieren/Verschieben stockt es leicht	Performance verbessern

5.4. Ausführung von 19.12.2011

Für Ausführung der Tests wurde Laptop2 benutzt.

Test-Nr.	Status (Gut, OK, Fehler)	Fehler/Unschönheiten	Mögliche Verbesserungen
UC01	Gut		
UC02	OK	Konstrukt hüpft beim Erstellen des Objekts	Verbesserungsmöglichkeit ist im Dokument SoftwareArchitectureDocument [1] aufgeführt.
UC03	Gut		
UC04	Gut		
UC05	Gut		
UC06	Gut		
UC07	Gut		
UC08	Gut		
UC09	Gut		
UC10	OK	Das ganze Konstrukt verschiebt sich zum Teil an eine andere Position	Siehe Verbesserungsmöglichkeit UC02
UC11	Gut		
UC12	Gut		
UC13	Gut		
UC14	Gut		
UC15	Gut		
UC16	Gut		
UC17	Gut		
UC18	Gut		
UC19	Gut		
UC20	Gut		

5.5. Ausführung von 19.12.2011

Für Ausführung der Tests wurde Laptop1 benutzt.

Test-Nr.	Status (Gut, OK, Fehler)	Fehler/Unschönheiten	Mögliche Verbesserungen
UC01	Gut		
UC02	Gut		
UC03	OK	Konstrukt flackert.	Performance verbessern
UC04	Gut		
UC05	OK	Konstrukt flackert leicht.	Performance verbessern
UC06	OK	Konstrukt flackert leicht.	Performance verbessern
UC07	Gut		
UC08	Gut		
UC09	Gut		
UC10	OK	Auch die Klasse B verschiebt sich etwas.	Verbesserungsmöglichkeit ist im SoftwareArchitectureDocument [1] aufgeführt
UC11	Gut		
UC12	Gut		
UC13	Gut		
UC14	Gut		
UC15	Gut		
UC16	Gut		
UC17	OK	Konstrukt flackert, wenn Maus darüber fährt.	Verbesserungsmöglichkeit ist im SoftwareArchitectureDocument [1] aufgeführt
UC18	Gut		
UC19	Gut		
UC20	Gut		

Nr.	ToDo Beschreibung	Erf.Datum	Zuteilung	Dringlichkeit	Status	Ref. Auf	Erl.Datum	Bemerkung
1	In SAD Alternativen beschreiben zur aktuellen Lösung beim Zeichnen der Texture	23. Nov	Lara	1	erledigt			
2	offenes Refactoring in MemorizedTranslation	24. Nov	Lara	2	erledigt			
3	Felder aus Klasse Transform3D ins Schnittstellendokument	25. Nov	Lara	1	erledigt			
4	Light Klassen in Package deprecated verschieben	26. Nov	Lara	2	erledigt			
5	Besseren Namen für MemorizedTranslation suchen	28. Nov	Lara	2	erledigt			
6	Alle Berechnungen im view Package ins model bringen	28. Nov	Marion	2	erledigt			
7	Domain-Modell: Text2D ergänzen	28. Nov		2	erledigt			
8	Übergabewert von 3DCOV im COV3dModelController berücksichtigen: view.setBackClipDistance	28. Nov		2	nicht in dieser Arbeit erledigt			
9	Switchen zwischen Diagrammen: Ursprünglich wird beim Switchen das Konstrukt wieder auf Startposition gesetzt, evtl. ebenso implementieren	02. Nov	Marion	2	erledigt			
10	Kantensichtbarkeit der Boxen: drei Helligkeitsabstufungen einsetzen, je zwei Seitenflächen in unterschiedlicher Helligkeit, damit zwischen allen Flächen ein Helligkeitsunterschied besteht.	02. Nov	Marion	2	erledigt			
11	Assoziationsnamen (Text2d) sollte noch in die richtige Tiefe gesetzt werden, sowie Refactoring der Klasse TextImageTransform	02. Nov	Marion	2	erledigt			
12	Testen auf anderem Betriebsystem	12. Dez	Lara + Marion	2	erledigt			
13	Titelblatt erstellen	12. Dez	Lara	2	erledigt			
14	Aufgabenstellung mitabgeben	12. Dez		2	erledigt			
15	Erklärung über eigenständige Arbeit	12. Dez	Lara + Marion	2	erledigt			
16	Abstract	12. Dez	Marion	2	erledigt			
17	Management Summary	12. Dez	Lara	2	erledigt			
18	Technischer Bericht der Arbeit	12. Dez	Lara	2	erledigt			
19	Persönliche Berichte	12. Dez		2	erledigt			
20	Glossar	12. Dez		2	erledigt			
21	Literaturverzeichnis	12. Dez		2	erledigt			
22	Dokumente des Projekts	12. Dez		2	erledigt			
23	Kurzfassung für Studienarbeit	12. Dez	Marion	2	erledigt			
24	Poster	12. Dez	Lara	2	erledigt			
25	Veröffentlichung der Arbeit im Internet?	12. Dez		2	erledigt			
26	ToDos code	12. Dez	Marion	2	erledigt			
27	Sequenzdiagramme	12. Dez	Marion	2	erledigt			
28	Systemtests	12. Dez	Lara	2	erledigt			
29	Use Cases	12. Dez	Marion	2	erledigt			
30	Unit Tests	12. Dez		2	erledigt			
31	Refactoring	12. Dez	Lara	2	erledigt			
32	SAD	12. Dez	Lara	2	erledigt			
33	Im SAD Idee beschreiben, wie das Gumpen verhindert weden kann (Map)	12. Dez	Lara	2	erledigt			
34	Nach Besprechung AxisAngle, BoundingSphere, Bounds kommentar raus	13. Dez	Lara	3	erledigt			Kommentar gemäss Betreuer belassen
35	Tooltip braucht Änderung im 3DCOV -> im Designpapier bemerken.	15. Dez	Lara	2	erledigt			
36	Im CodeReview packages umbenennen	15. Dez	Marion	2	erledigt			
37	Invert bei eigenSpeicherung falsch	17. Dez	Marion	2	erledigt			
38	SAD: Move und Draglevel beschreiben	17. Dez	Lara	2	erledigt			
39	Javadoc erstellen	21. Dez		1	erledigt			
40	Auf CD: EA, Eclipse, def. Versionen, JDK	21. Dez		1	erledigt			

Nr.	Bug Beschreibung	Erf.Datum	Zuteilung	Dringlichkeit	Status	Ref. Auf	Erl.Datum	Bemerkungen
1	Rotation: einzelne Box dreht sich nicht um sich selbst	21. Nov	Lara	2	erledigt			
2	Texture: ist nicht optimal deckungsgleich mit Box	21. Nov	Lara	2	erledigt			Gelöst mit Schablone (clip(Polygon))
3	Texture: Punkte im Polygon sind noch nicht aktuell, wenn paint() aufgerufen wird.	21. Nov	-	1	erledigt			Durch Optimierung der paint-Aufrufe erledigt
4	Zoomen: bei zu hohem Zoomen erscheint Konstrukt wieder verkehrt	21. Nov		2	nicht in dieser Arbeit erledigt			
5	Zoomen: mit Tastatur und mit Mausrädchen zoomt in unterschiedlichen Geschwindigkeiten	21. Nov	Lara	2	erledigt			
6	Rotieren: Beim Rotieren springt die Box zum Teil an einen anderen Ort	24. Nov	Marion	1	erledigt			
7	Rotieren: wenn man nach Gegenuhrzeigersinn rotiert mit der Maus dann dreht es auf einmal in die falsche Richtung.	25. Nov	Marion	1	erledigt			in AxisAngle: angle von int auf double gesetzt, zu testen, ob es etwas genützt hat
8	Nach dem Löschen von einzelnen Boxen verschiebt sich das Konstrukt bei erneutem Bewegen zu einem anderen Punkt	01. Dez		2	nicht in dieser Arbeit erledigt			
9	Rotation: bei erst horizontalem Draggen und nachher vertikalem Draggen ohne Unterbrechung des Draggen rotiert das Konstrukt weiter in horizontaler Richtung.	02. Dez	Lara	2	erledigt			
10	Rotation: Bei Richtungswechsel nach Mausloslassen rotiert Konstrukt noch wenig weiter bevor es Richtung ändert.	02. Dez	Lara	2	erledigt			
11	Zoomen funktioniert umgekehrt als ursprünglich bei 3DCOV (Anpassen so wie in Benutzerhandbuch)	02. Dez	Marion	2	erledigt			
12	Vertikale Rotation ist verkehrt	02. Dez	Marion	2	erledigt			
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								
29								

Zeiterfassung

Projekt: **J3DEval**
Teammitglied: **Marion Walser**

Durchschnittliche Soll-Arbeitszeit: **16.0**

Woche	Tag	Datum	Tätigkeit	Arbeitspaket	Zeit	Zeit/Wo	Total	Soll
Woche 1	Montag	19-09-2011	Grobplanung mit Artefakten	Projektplan	3.5			
	Montag	19-09-2011	Zeitplan	Zeitplan	2.0			
	Mittwoch	21-09-2011	Sitzung mit T. Letsch	Reviews mit Betreuer	2.0			
	Donnerstag	22-09-2011	Grobplanung mit Artefakten angepasst	Projektplan	1.0			
	Freitag	23-09-2011	3DCOV-Projekt im Eclipse erstellt	Einarbeitung in 3DCOV-Code und 3D	1.5			
	Samstag	24-09-2011	mich über OpenGL informiert und 3DCOV-	Einarbeitung in 3DCOV-Code und 3D	1.5			
	Sonntag	25-09-2011	Projektplan ergänzt	Projektplan	3.0			
						14.5	**14.5**	16.0
Woche 2	Montag	26-09-2011	Projektplan ergänzt	Projektplan	0.5			
	Montag	26-09-2011	Einarbeitung in Java 3D API und 3DCOV	Einarbeitung in 3DCOV-Code und 3D	4.5			
	Dienstag	27-09-2011	Dokumente überprüft	Dokumente überprüfen	1.5			
	Dienstag	27-09-2011	Teambesprechung	Besprechungen	0.5			
	Mittwoch	28-09-2011	Sitzung mit T. Letsch	Reviews mit Betreuer	2.0			
	Donnerstag	29-09-2011	Protokoll schreiben	Reviews mit Betreuer	0.5			
	Freitag	30-09-2011	Dokumentvorlage angepasst	Dokumentvorlagen	1.0			
	Samstag	01-10-2011	aktuelle dll gesucht und nicht gefunden	Evaluation durchführen	2.0			
	Sonntag	02-10-2011	Projektplan + Iterationsplan aktualisiert aufgrund Feedback von Betreuer	Projektplan	1.5			
	Sonntag	02-10-2011	3D Libraries evaluiert	Evaluation durchführen	1.0	**15.0**	**29.5**	32.0
Woche 3	Montag	03-10-2011	3D Libraries evaluiert	Evaluation durchführen	4.5			
	Montag	03-10-2011	Teambesprechung	Besprechungen	0.5			
	Montag	03-10-2011	Arbeitspakete aktualisiert	Iterationspläne	0.5			
	Montag	03-10-2011	Soll Elaboration 1 angepasst	Zeitplan	0.5			
	Montag	03-10-2011	Java 3D Source Code in Projekt 3DCOV integriert	Einarbeitung in 3DCOV-Code und 3D	2.0			
	Mittwoch	05-10-2011	Downloaden und Aufarbeiten Java 3D API	Einarbeitung in 3DCOV-Code und 3D	1.0			
	Freitag	07-10-2011	Überarbeiten Layout	Projektplan	0.5			
	Freitag	07-10-2011	Kriterien synchronisiert	Evaluationskriterien erstellen	0.5			
	Samstag	08-10-2011	Einarbeitung in Java 3D API	Einarbeitung in 3DCOV-Code und 3D	3.0			
	Sonntag	08-10-2011	Einarbeitung in Java 3D API	Einarbeitung in 3DCOV-Code und 3D	1.0			
	Sonntag	08-10-2011	Erarbeitung Struktur Java 3D API	Release 1	2.0			
	Sonntag	08-10-2011	Evaluationsmatrix	Evaluationsdokumentation	1.0	**17.0**	**46.5**	48.0
Woche 4	Montag	10-10-2011	Funktionsumfang Release erstellt	Funktionsumfang Releases	1.5			
	Montag	10-10-2011	Evaluationsmatrix dokumentiert	Evaluationsdokumentation	1.5			
	Montag	10-10-2011	Teambesprechung	Besprechungen	0.5			
	Montag	10-10-2011	Struktur Java 3D API abgegrenzt	Evaluation durchführen	2.0			
	Dienstag	11-10-2011	Evaluationsmatrix dokumentiert	Evaluationsdokumentation	1.0			
	Dienstag	11-10-2011	Lösungsvarianten gesammelt	Lösungsvarianten suchen	1.0			
	Dienstag	11-10-2011	Erarbeitung Struktur Java 3D API	Einarbeitung in 3DCOV-Code und 3D	2.0			
	Mittwoch	11-10-2011	Evaluation 1 Doks	Dokumente überprüfen	1.0			
	Freitag	13-10-2011	Sitzung mit T. Letsch	Reviews mit Betreuer	1.0			
	Freitag	13-10-2011	Korrekturen aus Sitzung	Projektplan	0.5			
	Sonntag	15-10-2011	Von unten nach oben durch den Code gewühlt, aufwändig, aber evtl. möglich. Entkopplung von oben nach unten ergibt zu viele Fehler	Release 1	5.0			
						17.0	**63.5**	64.0
Woche 5	Montag	17-10-2011	Arbeiten an Entkopplung	Release 1	6.0			
	Montag	17-10-2011	Teambesprechung	Besprechungen	0.5			
	Dienstag	18-10-2011	Buch Selman lesen	Einarbeitung in 3DCOV-Code und 3D	0.5			
	Mittwoch	19-10-2011	Sitzung mit T. Letsch	Reviews mit Betreuer	1.0			
	Mittwoch	19-10-2011	Zeichnen mit Lara (Rotation)	Release 1	1.0			
	Mittwoch	19-10-2011	Sitzungsprotokoll schreiben	Reviews mit Betreuer	0.5			
	Donnerstag	20-10-2011	Projektion-Berechnung umstellen	Release 1	2.0			
	Freitag	21-10-2011	Projektion-Berechnung umstellen	Release 1	2.5			
	Samstag	22-10-2011	Projektion-Berechnung umstellen	Release 1	2.0			
	Sonntag	23-10-2011	Arbeiten an Entkopplung	Release 1	3.0			
						19.0	**82.5**	80.0
Woche 6	Montag	24-10-2011	Arbeiten an Entkopplung	Release 1	4.0			
	Dienstag	25-10-2011	Arbeiten an Entkopplung	Release 1	2.0			
	Mittwoch	26-10-2011	Sitzung mit T. Letsch	Reviews mit Betreuer	1.0			
	Mittwoch	26-10-2011	Sitzung mit T. Letsch	Reviews mit Betreuer	0.5			
	Mittwoch	26-10-2011	Teambesprechung	Besprechungen	0.5			
	Donnerstag	27-10-2011	Jenkins eingerichten	Projektautomation	1.0			
	Donnerstag	27-10-2011	Test GeometryArray aktualisiert	Unit Tests	2.0			
	Freitag	28-10-2011	RotationPoint ins Canvas erstellt	Release 1	1.0			
	Samstag	29-10-2011	Arbeiten an Entkopplung	Release 1	4.0			
	Sonntag	30-10-2011	Beurteilung nach Prototyping schreiben	Beurteilung + Entscheid Lösungsvarianten	5.0			
						21.0	**103.5**	96.0
Woche 7	Montag	31-10-2011	Use Cases überarbeitet	Anforderungsspezifikation	2.0			
	Montag	31-10-2011	Erster Code-View ausgeführt	Review Code	4.0			
	Montag	31-10-2011	Teambesprechung	Besprechungen	0.5			
	Dienstag	01-11-2011	Erster Code-View ausgeführt	Refactoring	3.0			
	Dienstag	01-11-2011	Struktur erstellt	Systemtests	1.5			
	Dienstag	01-11-2011	Systemtests	Dokumente überprüfen	2.5			
	Mittwoch	02-11-2011	Besprechung mit Betreuer	Reviews mit Betreuer	1.0			
	Mittwoch	02-11-2011	Besprechen ob MVC Pattern	Release 2	1.0			
	Donnerstag	03-11-2011	AspectJ Verständnis erarbeitet	Einarbeitung in 3DCOV-Code und 3D	1.0			
	Freitag	04-11-2011	Planung für Construction 1 erstellt	Zeitplan	0.5			
	Freitag	04-11-2011	Planung für Construction 1 erstellt	Iterationspläne	0.5			
	Freitag	04-11-2011	Kopf- und Fusszeile geändert	Dokumentvorlagen	1.5			
						19.0	**122.5**	112.0
Woche 8	Montag	07-11-2011	Dokumente überprüfen + Abgabe	Dokumente überprüfen	1.5			
	Montag	07-11-2011	Release 2 Dok erstellt	Release 2	0.5			
	Montag	07-11-2011	Arbeiten an Rotation Use Case	Release 2	2.0			
	Dienstag	08-11-2011	Arbeiten an Update und Verschiebung	Release 2	2.0			
	Dienstag	08-11-2011	Linie zeichnen	Release 2	2.0			
	Mittwoch	09-11-2011	Besprechung mit Betreuer	Reviews mit Betreuer	1.0			
	Donnerstag	10-11-2011	TraceLib importiert	Release 2	1.0			
	Freitag	11-11-2011	Arbeiten an Update und Verschiebung	Release 2	2.0			
	Donnerstag	10-11-2011	Transformation umstrukturiert	Release 2	10.0			
						22.0	**144.5**	128.0
Woche 9	Montag	14-11-2011	Use Cases weitergeführt	Anforderungsspezifikation	1.0			
	Montag	14-11-2011	Korrekturen aus Sitzung, Dokumente für	Dokumente überprüfen	1.0			
	Montag	14-11-2011	Use Cases weitergeführt	Anforderungsspezifikation	3.0			
	Dienstag	15-11-2011	Boxoberfläche zeichnen	Release 2	2.0			
	Dienstag	15-11-2011	Boxoberfläche zeichnen	Release 2	2.0			
	Mittwoch	16-11-2011	Besprechung mit Betreuer	Reviews mit Betreuer	1.0			

Zeiterfassung

Projekt: **J3DEval**
Teammitglied: **Marion Walser**

Durchschnittliche Soll-Arbeitszeit: **16.0**

Woche	Tag	Datum	Tätigkeit	Arbeitspaket	Zeit	Zeit/Wo	Total	Soll
Woche	Mittwoch	16-11-2011	Boxoberfläche zeichnen	Release 2	1.0			
	Donnerstag	17-11-2011	Boxoberfläche zeichnen	Release 2	2.0			
	Freitag	18-11-2011	Use Cases weitergeführt	Anforderungsspezifikation	1.0			
	Samstag	19-11-2011	an Rotation gearbeitet	Release 2	3.0			
	Samstag	19-11-2011	gestrichelte Linie	Release 2	1.0			
	Sonntag	19-11-2011	Text zeichnen	Release 2	1.0			
	Sonntag	20-11-2011	LineEndPoint der Abhängigkeiten	Release 2	1.0			
	Sonntag	20-11-2011	Snapshot	Release 2	1.5	**21.5**	**166.0**	144.0
Woche 10	Montag	21-11-2011	Bug mit erstem Zeichnen	Bugfixing	1.0			
	Montag	21-11-2011	an Texture gearbeitet	Release 2	5.0			
	Dienstag	22-11-2011	Klassendiagramme erstellt	Klassendiagramme	0.5			
	Mittwoch	23-11-2011	Besprechung mit Betreuer	Reviews mit Betreuer	1.0			
	Mittwoch	23-11-2011	Problem mit Einlesen der Texture	Bugfixing	1.0			
	Donnerstag	24-11-2011	Klassendiagramme erstellt, mit EA rumgeschlagen	Klassendiagramme	2.0			
	Freitag	25-11-2011	Review für Release 2	Review Code	1.5			
	Sonntag	27-11-2011	SSD erstellt	System Sequenz Diagramm	2.0			
	Sonntag	27-11-2011	Use Cases ergänzt	Anforderungsspezifikation	1.0			
	Sonntag	27-11-2011	Klassendiagramme aktualisiert	Klassendiagramme	1.5			
	Sonntag	27-11-2011	Test für Release 2 ergänzt, Tests ausgeführt	Systemtests	1.0	**17.5**	**183.5**	160.0
Woche 11	Montag	28-11-2011	Package Diagramm erstellt	System Architecture Document	2.0			
	Montag	28-11-2011	Jenkins eingerichten	Projektautomation	2.0			
	Montag	28-11-2011	Arbeit mit Lara	System Sequenz Diagramm	1.0			
	Dienstag	29-11-2011	Abgabe Release 2 vorbereiten	Dokumente überprüfen	1.0			
	Dienstag	29-11-2011	Zeichnen Komposition implementiert und Linienfarben angepasst	Release 3	2.0			
	Mittwoch	29-11-2011	Besprechung mit Betreuer	Reviews mit Betreuer	2.0			
	Mittwoch	29-11-2011	Sitzungsprotokoll schreiben	Reviews mit Betreuer	0.5			
	Donnerstag	30-11-2011	Kalkulation 2d-Points verschoben	Release 3	4.0			
	Sonntag	04-12-2011	Zurück auf Startposition implementiert	Release 3	3.0			
	Sonntag	04-12-2011	mich an Assoziationsnamen versucht	Release 3	3.0	**20.5**	**204.0**	176.0
Woche 12	Montag	05-12-2011	Arbeitspakete erstellt	Iterationspläne	1.0			
	Montag	05-12-2011	für Release 3 aktualisiert	Zeitplan	0.5			
	Montag	05-12-2011	Zeichen Assoziationsnamen	Release 3	4.0			
	Montag	05-12-2011	Background zeichnen	Release 3	1.0			
	Montag	05-12-2011	unterschiedliche Flächen auf Boxen	Release 3	1.0			
	Dienstag	06-12-2011	Abgabe Woche 12	Dokumente überprüfen	1.0			
	Dienstag	06-12-2011	Tiefe zeichnen der Linien und Assoz-Texten	Bugfixing	2.0			
	Dienstag	06-12-2011	Rotation	Bugfixing	2.0			
	Mittwoch	07-12-2011	Besprechung mit Betreuer	Reviews mit Betreuer	1.0			
	Sonntag	11-12-2011	an Texture gearbeitet	Release 3	3.0			
						16.5	**220.5**	192.0
Woche 13	Montag	12-12-2011	UpdateAllDrawables bei setRotationPoint ergänzt, damit Neupositionierung bei Änderung RotationPoint ausgeführt wird und nicht erst beim Anklicken	Bugfixing	0.5			
	Montag	12-12-2011	Line-Funktion in GeometryArray zu LineArray	Refactoring	0.5			
	Montag	12-12-2011	Schnittstelle Vererbung angepasst	Anforderungsspezifikation	0.5			
	Montag	12-12-2011	TODO teilweise erledigt, Konstanten erstellt	Refactoring	0.5			
	Montag	12-12-2011	Problem ergänzt	Funktionsumfang Releases	0.5			
	Montag	12-12-2011	Use Cases fertig geschrieben	Anforderungsspezifikation	3.0			
	Dienstag	13-12-2011	Docs für Abgabe Woche 13 erstellt	Dokumente überprüfen	0.5			
	Dienstag	13-12-2011	Kapitel "Entkoppeln" ergänzt, durchgelesen	Schlussberichte erstellen	1.0			
	Mittwoch	14-12-2011	Besprechung mit Betreuer	Reviews mit Betreuer	0.5			
	Mittwoch	14-12-2011	Test zu den Use Cases erstellt	Systemtests	2.0			
	Donnerstag	15-12-2011	Auswertung und Titelblatt aktualisiert,	Zeitplan	1.0			
	Donnerstag	15-12-2011	Persönlicher Bericht erstellt	Schlussberichte erstellen	1.0			
	Freitag	16-12-2011	Erklärung eigenständige Arbeit	Schlussberichte erstellen	0.5			
	Freitag	16-12-2011	Neu generiert und im SAD aktualisiert	Klassendiagramme	2.0			
	Freitag	16-12-2011	XML Erkenntnisse ergänzt	System Architecture Document	0.5			
	Samstag	17-12-2011	Package aktualisiert, durchgeschaut	Review Code	0.5			
	Samstag	17-12-2011	Kurzfassung erstellt	Schlussberichte erstellen	1.5			
	Samstag	17-12-2011	Poster bearbeitet	Schlussberichte erstellen	0.5			
	Samstag	17-12-2011	Neues Objektdiagramm-Beispiel erstellt	Schlussberichte erstellen	0.5			
	Samstag	17-12-2011	Management Summary durchgelesen	Dokumente überprüfen	0.5			
	Samstag	17-12-2011	Diagramm Graph besser formatiert	Klassendiagramme	0.5			
	Samstag	17-12-2011	Abgabe Woche 13 erstellt	Dokumente überprüfen	1.0			
	Samstag	17-12-2011	überarbeitet für Abgabe	System Architecture Document	2.0			
						21.5	**242.0**	208.0
Woche 14	Montag	19-12-2011	XML-Use Case ergänzt, Domain Model aktualisiert	Anforderungsspezifikation	1.0			
	Montag	19-12-2011	invert geändert, damit alle Varianten richtig erscheinen	Bugfixing	1.0			
	Montag	19-12-2011	Test mit eigenem Laptop	Systemtests	1.0			
	Montag	19-12-2011	Formeln berichtigt	Zeitplan	1.0			
	Montag	19-12-2011	Dokumente durchlesen + korrigiereren	Dokumente überprüfen	1.0			
	Mittwoch	20-12-2011	Box und Sphere	Unit Tests	2.0			
	Mittwoch	21-12-2011	Besprechung mit Betreuer	Reviews mit Betreuer	1.0			
	Mittwoch	21-12-2011	Zusammenstellung der Dokumente	Schlussberichte erstellen	6.0	**14.0**	**256.0**	224.0

Total Ist - Arbeitszeit: **256.0**
Total Soll - Arbeitszeit: **224.0**
Differenz: **32.0**

Zeiterfassung

Projekt: J3DEval
Teammitglied: Lara Mühlemann

Durchschnittliche Soll-Arbeitszeit: 17.1

Woche		Tag	Datum	Tätigkeit	Arbeitspaket	Zeit	Zeit/Wo	Total	Soll
Woche 1	SW 1	Montag	19-09-2011	Dokumentvorlage, Sitzungsprotokollvorlage, Sitzungsprotokoll, SVN einrichten	Dokumentvorlagen	3.0			
	SW 1	Montag	19-09-2011	svn aufgesetzt	Version Control System	0.5			
	SW 1	Montag	19-09-2011	Grobplanung mit Artefakten	Projektplan	4.0			
	SW 1	Dienstag	20-09-2011	Sitzungsprotokoll, Projektplan	Projektplan	1.5			
	SW 1	Mittwoch	21-09-2011	Büchersuche	Einarbeitung in 3DCOV-Code und 3D	1.0			
	SW 1	Mittwoch	21-09-2011	Logo kreeirt, Projektplan	Projektplan	1.5			
	SW 1	Mittwoch	21-09-2011	Besprechung mit T.Letsch	Reviews mit Betreuer + Protokoll schreiben	2.0			
	SW 1	Mittwoch	21-09-2011	Protokoll schreiben	Reviews mit Betreuer + Protokoll schreiben	0.5			
	SW 1	Freitag	23-09-2011	Projektplan	Projektplan	2.0			
	SW 1						**16.0**	**16.0**	17.1
Woche 2	SW 2	Montag	26-09-2011	Projektplan	Projektplan	2.0			
	SW 2	Montag	26-09-2011	Iterationsplan	Iterationspläne	3.0			
	SW 2	Montag	26-09-2011	Internetrecherche über 3D	Einarbeitung in 3DCOV-Code und 3D	3.0			
	SW 2	Dienstag	27-09-2011	Überprüfung Projektplan + Iterationsplan	Dokumente überprüfen	1.0			
	SW 2	Dienstag	27-09-2011	Glossar erstellen	Glossary	0.5			
	SW 2	Dienstag	27-09-2011	Besprechung mit Marion	Besprechung + Protokoll schreiben	0.5			
	SW 2	Dienstag	27-09-2011	PDFs generiert und verschickt	Dokumente überprüfen	0.5			
	SW 2	Mittwoch	30-09-2011	Besprechung mit T.Letsch	Reviews mit Betreuer + Protokoll schreiben	2.0			
	SW 2	Samstag	01-10-2011	Rausgeschrieben, welche Klassen und Methoden zur verfügung gestellt werden müssen und Code-Gerüst für entsprechende erstellt	Anforderungsspezifikation	3.5			
	SW 2						**16.0**	**32.0**	34.2
Woche 3	SW 3	Montag	03-10-2011	Anforderungsspezifikation erstellt	Anforderungsspezifikation	5.0			
	SW 3	Montag	03-10-2011	Evaluationskriterien erstellen	Evaluationskriterien erstellen	0.5			
	SW 3	Montag	03-10-2011	Besprechung mit Marion	Besprechung + Protokoll schreiben	0.5			
	SW 3	Montag	03-10-2011	Überarbeitung Zeitplan	Zeitplan	0.5			
	SW 3	Montag	03-10-2011	Lösungsvarianten suchen	Lösungsvarianten suchen	1.5			
	SW 3	Dienstag	04-10-2011	Schnittstellen	Anforderungsspezifikation	2.0			
	SW 3	Dienstag	04-10-2011	Einarbeitung Code	Einarbeitung in 3DCOV-Code und 3D	2.0			
	SW 3	Mittwoch	06-10-2011	Würfel in Java2D erstellen	Release 1	1.5			
	SW 3	Donnerstag	07-10-2011	Würfel in Java2D erstellen	Release 1	2.0			
	SW 3	Freitag	08-10-2011	Evaluationskriterien erstellen	Evaluationskriterien erstellen	0.5			
	SW 3	Freitag	08-10-2011	Java2D Beispiel verstehen	Einarbeitung in 3DCOV-Code und 3D	1.0			
	SW 3	Freitag	08-10-2011	Überarbeitung überprüfen und abschicken	Dokumente überprüfen	0.5			
	SW 3						**17.5**	**49.5**	51.3
Woche 4	SW 4	Montag	10-10-2011	Grobevaluation durchführen	Evaluation durchführen	3.0			
	SW 4	Montag	10-10-2011	Besprechung mit Marion	Besprechung + Protokoll schreiben	0.5			
	SW 4	Montag	10-10-2011	Schnittstellen in Anforderungsspezifikation	Anforderungsspezifikation	0.5			
	SW 4	Montag	10-10-2011	Zeitplan für Elab2	Zeitplan	0.5			
	SW 4	Montag	10-10-2011	Iterationsplan für Elab2	Iterationspläne	0.5			
	SW 4	Montag	10-10-2011	Risikomanagement für Elab2	Risikomanagement	0.5			
	SW 4	Montag	10-10-2011	Mit 3D-2D Berechnung vertraut machen	Einarbeitung in 3DCOV-Code und 3D	2.5			
	SW 4	Dienstag	11-10-2011	Quader statt Würfel zeichnen, Translation + Rotationen aufgetrennt	Release 1	3.0			
	SW 4	Mittwoch	12-10-2011	Dokumente überprüfen, PDFs machen und abschicken	Dokumente überprüfen	1.5			
	SW 4	Mittwoch	12-10-2011	Einarbeitung Code	Einarbeitung in 3DCOV-Code und 3D	0.5			
	SW 4	Freitag	14-10-2011	Besprechung mit Betreuer	Reviews mit Betreuer + Protokoll schreiben	1.0			
	SW 4			Protokoll schreiben	Reviews mit Betreuer + Protokoll schreiben	0.5			
	SW 4	Samstag	15-10-2011	Drehen nur in eine Richtung	Release 1	3.0			
	SW 4	Sontag	16-10-2011	Versucht drehung nicht im Körper sondern um	Release 1	2.0			
	SW 4						**19.5**	**69.0**	68.4
Woche 5	SW 5	Montag	17-10-2011	Rotation mit Maus + weitere Boxen	Release 1	5.0			
	SW 5	Montag	17-10-2011	Abgabe für Besprechung vorbereiten	Dokumente überprüfen	0.5			
	SW 5	Montag	17-10-2011	Besprechung	Besprechung + Protokoll schreiben	0.5			
	SW 5	Montag	17-10-2011	Ablauf von 3D Aufrufen genauer analysieren	Einarbeitung in 3DCOV-Code und 3D	2.0			
	SW 5	Dienstag	18-10-2011	Rotation weitergearbeitet	Release 1	4.0			
	SW 5	Mittwoch	19-10-2011	Reihenfolge von Shapes beim Zeichnen	Release 1	1.5			
	SW 5	Mittwoch	19-10-2011	Besprechung mit Betreuer + Protokoll	Reviews mit Betreuer + Protokoll schreiben	1.5			
	SW 5						**15.0**	**84.0**	85.5
Woche 6	SW 6	Montag	25-10-2011	unterstützen im Entkoppeln	Release 1	1.5			
	SW 6	Montag	25-10-2011	Domainanalyse schreiben	Domain Model	2.0			
	SW 6	Montag	25-10-2011	Rotation mit richtigen Zahlen (geht immer noch nicht) + OwnObject in Primitive integriert + Sortiertes Zeichnen der Boxen (hat noch eine schwachstelle)	Release 1	5.5			
	SW 6	Dienstag	25-10-2011	rotation fehler korrigieren / importieren von diagrammen ins 3dcov fehler behoben	Release 1	2.0			
	SW 6	Dienstag	25-10-2011	Domainanalyse: Konzeptbeschreibung	Domain Model	1.0			
	SW 6	Dienstag	25-10-2011	UC01 Systemsequenzdiagramm, UC01 Brief	System Sequenz Diagramm	1.0			
	SW 6	Mittwoch	26-10-2011	Besprechung mit Betreuer + Protokoll	Reviews mit Betreuer + Protokoll schreiben	1.0			
	SW 6	Mittwoch	26-10-2011	Besprechung mit Marion	Besprechung + Protokoll schreiben	0.5			
	SW 6	Mittwoch	26-10-2011	Packages umbenennen	Refactoring	0.5			
	SW 6	Sontag	30-10-2011	An UseCases gearbeitet	Anforderungsspezifikation	2.0			
	SW 6	Sontag	30-10-2011	Contracts UC 01 geschrieben	Operation Contracts	1.0			
	SW 6						**18.0**	**102.0**	102.6
Woche 7	SW 7	Montag	31-10-2011	Javadoc für Klassen eingefügt	Release 1	1.0			
	SW 7	Montag	31-10-2011	Release1 Dokument geschrieben	Release 1	0.5			
	SW 7	Montag	31-10-2011	Use Cases überarbeitet	Anforderungsspezifikation	2.0			
	SW 7	Montag	31-10-2011	Code-Review	Review Code	4.0			
	SW 7	Montag	31-10-2011	Sortierung so gemacht dass kein setzen von rotationspunkten mehr nötig sein wird	Refactoring	1.0			
	SW 7	Dienstag	01-11-2011	Code-Review	Refactoring	3.0			
	SW 7	Dienstag	01-11-2011	Systemtestdokument	Systemtests (Protokoll erstellen)	1.5			
	SW 7	Dienstag	01-11-2011	Dokumente überprüfen, PDFs machen und abschicken	Dokumente überprüfen	2.5			
	SW 7	Mittwoch	02-11-2011	Besprechung mit Betreuer	Reviews mit Betreuer + Protokoll schreiben	1.0			
	SW 7	Mittwoch	02-11-2011	Besprechen ob MVC Pattern	Release 2	1.0			
	SW 7						**17.5**	**119.5**	119.7
Woche 8	SW 8	Montag	07-11-2011	Packages aufgeteilt in view und model	Release 2	2.0			
	SW 8	Montag	07-11-2011	Funktionsumfang Releases	Funktionsumfang Releases	0.5			
	SW 8	Montag	07-11-2011	Dokumente überprüfen + Abgabe	Dokumente überprüfen	1.5			
	SW 8	Montag	07-11-2011	SAD begonnen	System Architecture Document	1.5			
	SW 8	Montag	07-11-2011	Observerpattern	Release 2	2.0			
	SW 8	Montag	07-11-2011	mit behaviors vertraut machen	Einarbeitung in 3DCOV-Code und 3D	2.0			
	SW 8	Dienstag	08-11-2011	Anfangstransformation anders gemacht	Release 2	2.0			
	SW 8	Mittwoch	09-11-2011	Besprechung mit Betreuer	Reviews mit Betreuer + Protokoll schreiben	1.5			
	SW 8	Donnerstag	10-11-2011	Behavior von Maus	Release 2	2.0			
	SW 8	Freitag	11-11-2011	Probleme bei Linien behoben	Release 2	2.0			
	SW 8						**17.0**	**136.5**	136.8

Zeiterfassung

Projekt: **J3DEval**
Teammitglied: **Lara Mühlemann**

Durchschnittliche Soll-Arbeitszeit: **17.1**

Woche		Tag	Datum	Tätigkeit	Arbeitspaket	Zeit	Zeit/Wo	Total	Soll
Woche 9	SW 9	Montag	14-11-2011	MemorizedTranslation ins Group und nicht mehr static. Translation flicken.GetBounds. Rotation anders. Rotationspunkt von 3DCOV	Release 2	8.0			
	SW 9	Montag	14-11-2011	Transformation veruschen zu flicken	Release 2	1.0			
	SW 9	Montag	14-11-2011	Color/Matrial/Texture beim zeichnen	Release 2	2.0			
	SW 9	Dienstag	15-11-2011	reihenfolge von shapes beim zeichnen wieder flicken	Release 2	2.0			
	SW 9	Mittwoch	16-11-2011	Besprechung	Reviews mit Betreuer + Protokoll schreiben	1.0			
	SW 9	Mittwoch	16-11-2011	Arbeit an Bild auf Oberfläche einfügen	Release 2	1.0			
	SW 9	Donnerstag	17-11-2011	Arbeit an Bild auf Oberfläche einfügen	Release 2	2.0			
	SW 9	Freitag	18-11-2011	Arbeit an Bild auf Oberfläche einfügen	Release 2	2.0			
	SW 9	Samstag	19-11-2011	Transformation aufgeteilt in Objekttransformation und Gruppentransformation	Release 2	1.5			
	SW 9						**20.5**	**157.0**	153.9
Woche 10	SW 10	Montag	21-11-2011	Bild versucht zu machen aus Text2D	Release 2	4.0			
	SW 10	Dienstag	22-11-2011	Text2D	Release 2	0.5			
	SW 10	Dienstag	22-11-2011	Package umstrukturiereung	Release 2	1.5			
	SW 10	Dienstag	22-11-2011	Transformation umstrukturiert damit sich Pfeile die an BranchGroup angehängt sind auch darstellen lassen	Release 2	1.5			
	SW 10	Mittwoch	23-11-2011	Transformation umstrukturiert damit sich Pfeile die an BranchGroup angehängt sind auch darstellen lassen	Release 2	1.0			
	SW 10	Mittwoch	23-11-2011	Besprechung mit Betreuer + Protokoll	Reviews mit Betreuer + Protokoll schreiben	1.5			
	SW 10	Donnerstag	24-11-2011	Unit Tests flicken + schreiben	Unit Tests	1.0			
	SW 10	Donnerstag	24-11-2011	Bug fix	Bugfixing	1.0			
	SW 10	Freitag	25-11-2011	Code Review	Review Code	2.0			
	SW 10	Samstag	26-11-2011	SAD Text zu diagrammen + refactorings	System Architecture Document	4.0			
	SW 10	Samstag	26-11-2011	Farben + Zoom geschwindkgiet	Release 2	0.5	**18.5**	**175.5**	171.0
Woche 11	SW 11	Montag	28-11-2011	Dokumente korrekturgelesen	Dokumente überprüfen	5.0			
	SW 11	Montag	28-11-2011	System Sequenzdiagramme überarbeitet	System Sequenz Diagramm	1.5			
	SW 11	Montag	28-11-2011	Bug behoben bei rotierung	Bugfixing	0.5			
	SW 11	Dienstag	29-11-2011	Abgabe	Dokumente überprüfen	1.5			
	SW 11	Mittwoch	30-11-2011	Review 2	Reviews mit Betreuer + Protokoll schreiben	2.0			
	SW 11	Mittwoch	30-11-2011	Transformationen über TransformGroup, nicht mehr MemorizedTransformaiton	Release 3	3.0			
	SW 11	Donnerstag	31.11.2011	Fehler in Translation behoben	Release 3	0.5			
	SW 11	Samstag	03-12-2011	Selectionsbehavior	Release 3	3.5			
	SW 11						**17.5**	**193.0**	188.1
Woche 12	SW 12	Montag	05-12-2011	Iterationsplan	Iterationspläne	0.5			
	SW 12	Montag	05-12-2011	Funktionsumfang Releases	Funktionsumfang Releases	0.5			
	SW 12	Montag	05-12-2011	Tooltip	Release 3	1.0			
	SW 12	Montag	05-12-2011	Sphere (Selektion von Klassen)	Release 3	6.0			
	SW 12	Dienstag	06-12-2011	Sphere (Selektion von Klassen)	Release 3	1.0			
	SW 12	Dienstag	06-12-2011	SAD Design einzeln	System Architecture Document	0.5			
	SW 12	Dienstag	06-12-2011	Selektion Objekte Helligkeit	Release 3	0.5			
	SW 12	Dienstag	06-12-2011	Background ohne 3DCOV code verändern	Release 3	1.0			
	SW 12	Dienstag	06-12-2011	Sphere positionierung	Release 3	1.0			
	SW 12	Mittwoch	07-12-2011	Selektion von Linien	Release 3	1.0			
	SW 12	Mittwoch	07-12-2011	Besprechung	Reviews mit Betreuer + Protokoll schreiben	1.5			
	SW 12	Sontag	11-12-2011	Tracing	Release 3	3.0			
	SW 12						**17.5**	**210.5**	205.2
Woche 13	SW 13	Montag	12-12-2011	Bugfix für Selektion von Linie	Bugfixing	2.0			
	SW 13	Montag	12-12-2011	Performance verbessern	Release 3	0.5			
	SW 13	Montag	12-12-2011	Technischer Bericht	Schlussberichte erstellen	4.0			
	SW 13	Montag	12-12-2011	Invert gefixed	Bugfixing	1.0			
	SW 13	Montag	12-12-2011	Review	Review Code	2.0			
	SW 13	Dienstag	13-12-2011	Refactoring	Refactoring	2.0			
	SW 13	Dienstag	13-12-2011	Draglevel für Bugfix	Bugfixing	1.0			
	SW 13	Mittwoch	14-12-2011	getLengthToViewPoint nur noch im Model	Refactoring	0.5			
	SW 13	Mittwoch	14-12-2011	Schablone für Bild	Bugfixing	1.0			
	SW 13	Mittwoch	14-12-2011	Besprechung	Reviews mit Betreuer + Protokoll schreiben	0.5			
	SW 13	Donnerstag	15-12-2011	Poster	Schlussberichte erstellen	2.0			
	SW 13	Donnerstag	16-12-2011	ifs beim tracing für Performance + MoveLevel	Release 3	1.0			
	SW 13	Donnerstag	16-12-2011	Unit Tests schreiben	Unit Tests	1.0			
	SW 13	Freitag	19-12-2011	Titelblatt, Management Summery, Technischer Bericht usw	Schlussberichte erstellen	1.0			
	SW 13	Freitag	19-12-2011	Problem mit Umlauten	Bugfixing	0.5			
	SW 13	Freitag	19-12-2011	Besprechung	Besprechung + Protokoll schreiben	0.5			
	SW 13	Freitag	19-12-2011	Iterationsplan + Zeitplan	Iterationspläne	0.5			
	SW 13	Freitag	19-12-2011	Systemtests auf Linux	Systemtests (Protokoll erstellen)	0.5			
	SW 13	Samstag	20-12-2011	Erklärung eigenständige Arbeit	Schlussberichte erstellen	0.5			
	SW 13	Samstag	20-12-2011	Dokumente durchlesen	Dokumente überprüfen	3.0			
	SW 13	Samstag	20-12-2011	SAD aktualisiert	System Architecture Document	1.0			
	SW 13						**26.0**	**236.5**	222.3
Woche 14	SW 14	Montag	19-12-2011	Dokumente durchlesen + korrigiereren	Dokumente überprüfen	4.0			
	SW 14	Montag	19-12-2011	Systemtests durchführen	Systemtests (Protokoll erstellen)	2.0			
	SW 14	Montag	19-12-2011	Architektur im Technischen Bericht + Schlussberichte	Schlussberichte erstellen	1.0			
	SW 14	Dienstag	20-12-2011	Unittests	Unit Tests	3.0			
	SW 14	Mittwoch	21-12-2011	Besprechung mit Betreuer	Reviews mit Betreuer + Protokoll schreiben	1.0			
	SW 14	Mittwoch	21-12-2011	Zusammenstellung der Dokumente	Schlussberichte erstellen	6.0			
	SW 14						**17.0**	**253.5**	239.4

Total Ist - Arbeitszeit: **253.5**
Total Soll - Arbeitszeit: **239.4**
Differenz: **14.1**