

# Qualitätssicherungsbericht

(V. 1.0)

*PSE WS 2013/14*



*Auftraggeber:*

Karlsruher Institut für Technologie  
Institut für Betriebs- und Dialogsysteme  
Prof. Dr.-Ing. C. Dachsbacher

*Betreuer:*

Dipl.-Inf. Thorsten Schmidt  
Dipl.-Inform. M. Retzlaff

*Auftragnehmer:*

Tobias Schulz, Maximilian Reuter, Pascal Knodel,  
Gerd Augsburg, Christina Erler, Daniel Warzel

*6. März 2014*

# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>3</b>
<b>2. Tests</b>	<b>4</b>
2.1. Übersicht . . . . .	4
2.1.1. Kategorien . . . . .	4
2.2. Werkzeuge . . . . .	6
2.2.1. Manuell . . . . .	6
2.2.2. Automatisiert . . . . .	7
2.3. Pflichtenheft . . . . .	9
2.4. Protokoll . . . . .	11
2.5. Nicht getestet . . . . .	21
2.6. Statistik . . . . .	23
2.6.1. Abdeckung . . . . .	23
<b>3. Fehler</b>	<b>26</b>
3.1. Übersicht . . . . .	26
3.1.1. Klassifizierung . . . . .	26
3.2. Werkzeuge . . . . .	27
3.2.1. Manuell . . . . .	27
3.2.2. Automatisiert . . . . .	27
3.3. Protokoll . . . . .	29
3.4. Statistik . . . . .	30
<b>4. Änderungen</b>	<b>31</b>
4.1. Protokoll . . . . .	31
4.1.1. Geändert . . . . .	31
4.1.2. Nicht geändert . . . . .	32
4.1.3. Verschönert . . . . .	33
<b>5. Ausnahmen</b>	<b>34</b>
5.1. Behandlung . . . . .	34
5.2. Meldungen . . . . .	34
<b>6. Schluss</b>	<b>35</b>
6.1. Bewertung . . . . .	35
<b>A. Anhang</b>	<b>36</b>
A.1. Aufnahmen . . . . .	36
A.1.1. Testknoten . . . . .	37

# 1. Einleitung

In dem folgendem Bericht beschreiben wir, was in der Qualitätssicherungsphase von uns erarbeitet wurde.

Um ein bestmögliches Arbeiten des Teams zu gewährleisten nutzten wir das Ticket-System von Github. In das Ticket-System kann man alle wichtigen Aufgaben und Bugs eintragen und diese einem Teammitglied zuweisen. Dies hat den Vorteil, dass man eine Auflistung zu allen noch zu bearbeitenden Aufgaben hat und somit nichts vergessen kann.

## 2. Tests

### 2.1. Übersicht

#### 2.1.1. Kategorien

Wir gliedern die von uns durchgeführten Testfälle in verschiedene Kategorien:

##### **Funktionstests**

In dieser Kategorie testen wir die elementaren Spielfunktionen. Das sind Funktionen, welche das Spiel erst spielbar machen.

##### **Komponententests**

Zu fast jeder Komponente werden Tests durchgeführt. Wir schließen lediglich Grafik-Komponenten und reine Daten vom Testen durch Komponententests aus. Die Gründe dafür sind in Abschnitt 2.5 im Detail beschrieben. Zur Strukturierung der Tests spiegeln wir die organisatorische Struktur des Knot3-Projekts, welches den Programmcode enthält. D.h. zu jeder Komponente die wir testen gibt es eine Testklasse im Tests-Projekt. Eine Statistik zur Testabdeckung durch Komponententests ist unter Abschnitt 2.6.1 verfügbar.

##### **Negativtests**

Die Negativtests prüfen, ob das Spiel auf eine (falsche) Eingabe oder Bedienung, welche nicht den Anforderungen an die Anwendung entspricht erwartungsgemäß reagiert.

##### **Extremtests**

Bei den Extremtests prüfen wir, ob unser Spiel beispielsweise auch bei der Verarbeitung größerer Datenmengen erwartungsgemäß reagiert und es dabei

nicht zu Programmabstürzen kommt. Zudem führen wir einfaches Benchmarking durch, wobei wir den Zeitbedarf kritischer Funktionen messen und nach möglichen Flaschenhälsen Ausschau halten.

### **Abnahmetests**

Hierbei haben wir menschliche Tester unser Spiel spielen lassen. Die Anforderungen und Kritiken, welche die Tester an unser Spiel stellen, werden wir in den Abnahmetests beschreiben.

## 2.2. Werkzeuge

Zur Testdurchführung helfen uns einige Werkzeuge. Wichtig bei deren Wahl waren uns diese Kriterien:

- Kostenlos (für studentische Projekte)
- Aktuell
- Open-Source
- In Visual Studio integrierbar
- Mit Git(-Hub) verwendbar

Eine Anleitung über die Integration und Verwendung der Werkzeuge und hilfreiche Links haben wir auf GitHub im Wiki unseres Projekts zusammengefasst. Lokal unter Visual Studio installierte Werkzeuge sind NUnit, OpenCover und ReportGenerator. Für deren Integration in Visual Studio sind NuGet Pakete verfügbar. Um die drei Werkzeuge in Visual Studio verwenden zu können, müssen sie auch aufeinander abgestimmt werden. Dazu sind Build-Skripte nötig. Unter Windows übernimmt diese Aufgabe bei uns eine einfache Stapelverarbeitungsdatei. Die „Batch“-Datei läuft beim Erstellen des Testabdeckungsberichts in Visual Studio oder lässt sich direkt ausführen.

Einerseits war es uns wichtig die Werkzeuge lokal bei jedem Entwickler verfügbar zu machen. Andererseits ist die individuelle Erstellung und Ausführung von Tests alleine sehr zeitaufwendig, weshalb wir zusätzlich Automatismen (s. Abschnitt 2.2.2) einsetzen.

### 2.2.1. Manuell

Werkzeuge die uns beim Schreiben und Auswerten der Tests manuell unterstützen.

#### **NUnit** , *V. 2.6.3*

NUnit ist ein Framework für Komponententests für alle .NET-Sprachen.

Internetseite: <http://www.nunit.org/>

## 2.2.2. Automatisiert

Zusätzlich verwenden wir serverseitige, automatisierte Dienste für Testdurchläufe und die Erstellung von Berichten, welche so ständig auf den neuesten Stand gebracht werden. Die Ergebnisse sind online abrufbar. Über bestandene und fehlgeschlagene Tests werden zudem durch einen Benachrichtigungsservice bei jeder Änderung E-Mails an die Entwickler versandt.

### **Visual Studio Test-Explorer** , *VS-V. 12.0.21005.1*

Die Entwicklungsumgebung Visual Studio unterstützt uns beim durchführen der Tests und stellt die Ergebnisse der NUnit-Komponententests grafisch im Test-Explorer dar.

### **OpenCover** , *V. 4.5.1923*

OpenCover ermittelt die Testabdeckung unter .NET-Sprachen ab Version 2.0. Wir nutzen es, um die Testabdeckung durch NUnit-Komponententests zu berechnen.

Internetseite: <http://opencover.codeplex.com/>

### **ReportGenerator** , *V. 1.9.1.0*

ReportGenerator erstellt zu den von OpenCover produzierten XML-Daten einen übersichtlichen Bericht. Es sind verschiedene Formate möglich. Wir erzeugen z.B. eine HTML-Ausgabe des Berichts.

Internetseite: <http://reportgenerator.codeplex.com/>

Während unser Projekt läuft ist der automatisch erstellte Bericht über die Testabdeckung unter der Internetadresse

<http://www.knot3.de/development/coverage.php>

erreichbar.

## **Travis Continuous Integration (TCI)**

Für private GitHub-Repositories gibt es mit TCI die Möglichkeit nach jedem Commit Tests laufen zu lassen. Führt eine Änderung zu Fehlern in bereits vorhandenen Testfällen wird dies in einer E-Mail über die Testzustände nach dem Commit an den Entwickler mitgeteilt. Der Verlauf von fehlerfreien und fehlerhafter Commits ist während der Laufzeit des Projekts unter

<https://travis-ci.org/pse-knot/knot3-code/builds>

abrufbar.



## 2.3. Pflichtenheft

Die Tabelle 2.1 ordnet den im Pflichtenheft vorgegebenen Testfällen einen Verweis in das Testprotokoll - wo alle Tests beschrieben werden - unter Abschnitt 2.4 zu. Da sich die Beschreibungen beim Nachspezifizieren ändern und weiter aufgliedern, erleichtert die Zuordnung das Auffinden der Pflicht-Untersuchungen. Im PDF-Dokument zum QS-Bericht führt ein Klick auf einen Bezeichner in der Spalte **Testprotokoll** direkt zu der entsprechenden Stelle im Protokoll. Während der Testphase geben die Verweise eine ständige Übersicht zum aktuellen Fortschritt und verhindern, dass bei der Vielzahl von Tests etwas vergessen wird.

Tabelle 2.1.: Pflichtenheft-Testfälle, Referenzverweise

Testfall	Verweis	
	Pflichtenheft	Testprotokoll
<b>Funktionstests:</b>		
Einstellen gültiger Grafikauflösungen	/PTF_10/	FT_1
Bedienen der Nutzerschnittstellen	/PTF_20/	...
Transformieren von Knoten in gültige Knoten	/PTF_20/ /PTF_70/	FT_10
Erstellen von Challenge-Leveln	/PTF_30/	FT_20
Beenden des Programms	/PTF_50/	FT_40
Pausieren und Beenden von Spielen	/PTF_60/	FT_50
Manuelle Positionierung der Kamera	/PTF_80/	FT_80
Bestehen von Challenge-Leveln	/PTF_90/	FT_60
Speichern und Laden von Knoten	/PTF_100/	FT_70
Einrichten und Entfernen des Programms	/PTF_120/ /PTF_130/	FT_90

**Negativtests:**

Laden nicht gültiger Knoten-Dateien	/PTF_500/	NT_10
Erstellen von Challenge-Leveln aus gleichen Knoten	/PTF_510/	NT_20
Transformieren von Knoten in nicht gültige Knoten	/PTF_520/	NT_30
Löschen von Standard-Leveln	/PTF_530/	NT_40
Verhalten beim Drücken von nicht belegten Tasten	/PTF_1020/	NT_50

**Extremtests, Benchmarks:**

Laden großer Knoten-Dateien	/PTF_1000/	...
Erstellen von großen Challenge-Leveln	/PTF_1010/	...
Erfassen der maximal möglichen Eingabegeschwindigkeit	/PTF_1020/	...

## 2.4. Protokoll

In diesem Abschnitt sind zu jeder Testkategorie die durchgeführten Testfälle beschrieben. Über nicht durchgeführte Tests wird unter 2.5 berichtet.

- ↗ Funktionen
- ↗ Komponenten
- ↗ Negativ-Fälle
- ↗ Extremfälle
- ↗ Spielbarkeit

## Funktionstests

### FT\_10 Gültige Knoten-Transformationen.

Wir definieren eine Liste möglicher Transformationen ausgehend vom Startknoten. Jede Transformation ist einzeln ausführbar.

1. Jede einzelne Kante des Startknotens ist selektierbar.
2. Mehrere Kanten (zwei, drei oder vier) des Startknotens sind selektierbar.
3. Jede einzelne Kante des Startknotens ist in jede Richtung des dreidimensionalen Raumes um einen Schritt durch direktes Anklicken und anschließendes Ziehen mit der Maus verschiebbar.
4. Jede einzelne Kante des Startknotens ist in jede Richtung des dreidimensionalen Raumes um mehrere (mindestens zehn) Schritte durch direktes Anklicken und anschließendes Ziehen mit der Maus verschiebbar.
5. Mehrere (mindestens zwei) selektierte Kanten sind um einen Schritt durch direktes Anklicken und anschließendes Ziehen mit der Maus verschiebbar.
6. Mehrere (mindestens zwei) selektierte Kanten sind um mehrere (mindestens zehn) Schritte durch direktes Anklicken und anschließendes Ziehen mit der Maus verschiebbar.
7. Jede einzelne Kante des Startknotens ist in jede Richtung des dreidimensionalen Raumes um einen Schritt durch Anklicken der Navigationspfeile und anschließendes Ziehen mit der Maus verschiebbar.
8. Jede einzelne Kante des Startknotens ist in jede Richtung des dreidimensionalen Raumes um mehrere (mindestens zehn) Schritte durch Anklicken der Navigationspfeile und anschließendes Ziehen mit der Maus verschiebbar.
9. Mehrere (mindestens zwei) selektierte Kanten sind um einen Schritt durch Anklicken der Navigationspfeile und anschließendes Ziehen mit der Maus verschiebbar.
10. Mehrere (mindestens zwei) selektierte Kanten sind um mehrere (mindestens zehn) Schritte durch Anklicken der Navigationspfeile und anschließendes Ziehen mit der Maus verschiebbar.
11. Jede einzelne Kante des Startknotens lässt sich nach ihrer Verschiebung in die vorige Position durch direktes Anklicken und anschließendes Ziehen zurücksetzen.
12. Jede einzelne Kante des Startknotens lässt sich nach ihrer Verschiebung in die vorige Position durch Anklicken des „Undo“-Buttons zurücksetzen.
13. Jede einzelne Kante des Startknotens lässt sich nach ihrer Verschiebung in die vorige Position durch Anklicken des „Undo“-Buttons zurücksetzen.



und der „Redo“-Button macht die Aktion des „Undo“-Buttons rückgängig.

#### **FT\_20** *Erstellen von Challenge-Leveln*

1. Durch einen Klick auf den Text „NEW Creative“ öffnet sich das Creative-Menü.
2. Durch einen Klick auf den Text „NEW Challenge“ im Creative-Menü öffnet sich das Challenge-Konstruktions-Menü.
3. Im Challenge-Konstruktions-Menü in der linken Liste einen Zielknoten auswählen.
4. In der rechten Liste einen Startknoten auswählen.
5. Im Eingabefeld einen Namen für die Challenge eingeben und mit der „ENTER“-Taste bestätigen.

#### **FT\_30** *Nachbaubare Knoten-Beispiele*

Eine Sammlung von Beispiel-Knoten zum Nachbauen. Jeder Knoten deckt einen im Spielverlauf immer wieder-auftretenden Modellierungsfall einmalig ab.

1.  „Überleger“
2.  „Schlaufe“

#### **FT\_40** *Beenden des Programms.*

Ein Klick auf den „Exit“-Button im Hauptmenü beendet das Programm.

#### **FT\_50** *Pausieren und Beenden von Spielen.*

In beiden Spielmodi besteht die Möglichkeit ein Spiel zu pausieren und zu beenden.

Pausieren eines laufenden Spiels:

1. Durch Drücken der „ESC“-Taste im laufenden Spiel öffnet sich das Pausemenü. Im Challenge-Mode wird die Challenge-Zeit hierbei pausiert.
2. Durch ein Klick auf den Text „Back to Game“ im Pausemenü wird dieses Menü geschlossen und das Spiel fortgesetzt. Im Challenge-Mode läuft nach dem Schließen des Pausemenüs die Challenge-Zeit weiter.

Beenden eines laufenden Spiels im Challenge-Mode:

1. Durch Drücken der „ESC“ -Taste im laufendem Spiel öffnet sich das Pausemenü.
2. Durch einen Klick auf den Text „Abort Challenge“ schließt sich die laufende Challenge und öffnet das Hauptmenü.

Beenden eines laufenden Spiels im Creative-Mode:

1. Durch Drücken der „ESC“ -Taste im laufendem Spiel öffnet sich das Pausemenü.
2. Durch einen Klick auf den Text „Save and Exit“ wird der Knoten gespeichert:
  - Fall 1: Ist bereits ein Dateiname vorhanden, wird dieser beim Speichern verwendet.
  - Fall 2: Ist noch kein Dateiname vorhanden, öffnet sich der „Save As“-Dialog und fordert den Spieler auf einen Namen einzugeben. Der Knoten wird nach Bestätigen dieses Dialogs gespeichert und der Spieler gelangt ins Hauptmenü.
3. Durch einen Klick auf den Text „Discard Changes and Exit“ gelangt der Spieler ins Hauptmenü.

#### **FT\_60** *Bestehen von Challenge-Leveln.*

Nachdem der Spieler die letzte Transformation zur Beendigung der Challenge getätigt hat, reagiert das Spiel folgendermaßen:

1. Die Challenge-Zeit des Spielers wird gestoppt.
2. Es öffnet sich ein Dialog, welcher den Spieler auffordert seinen Spielernamen einzugeben.
3. Nachdem der Spieler den Spielernamen mit der „ENTER“ -Taste bestätigt hat, wird die Highscore-Liste geöffnet.
4. In der Highscore-Liste kann der Spieler die 10 besten Highscore-Einträge sehen. Wenn die Challenge-Zeit des Spielers gereicht hat, besitzt dieser auch einen Highscore-Eintrag.
5. Mit Hilfe der zwei Buttons („Restart challenge“ und „Return to menu“ ) kann der Spieler die Challenge noch einmal spielen oder zum Hauptmenü zurückkehren.

## **FT\_70** *Speichern und Laden von Knoten.*

Hat der Spieler im Creative-Mode einen Knoten erstellt, so kann er diesen abspeichern und wiederum laden. Dazu muss man folgende Dinge tun:

1. Durch Drücken der „ESC“ -Taste öffnet sich das Pausemenü.
2. Im Pausemenü kann man den Knoten auf unterschiedliche Art und Weise speichern:
  - Durch einen Klick auf den Text „Save“ . wird man aufgefordert den Knotennamen einzugeben, welchen man mit der „ENTER“ -Taste bestätigt. Hat der Knoten bereits einen Knotennamen, so wird man nicht mehr aufgefordert diesen einzugeben. Daraufhin wird der Knoten unter diesem Namen gespeichert.
  - Durch einen Klick auf den Text „Save As“ wird man aufgefordert den Knotennamen einzugeben, welchen man mit der „ENTER“ -Taste bestätigt. Daraufhin wird der Knoten unter diesem Namen gespeichert.
  - Durch einen Klick auf den Text „Save and Exit“ wird man aufgefordert den Knotennamen einzugeben, welchen man mit der „ENTER“ -Taste bestätigt. Hat der Knoten bereits einen Knotennamen, so wird man nicht mehr aufgefordert diesen einzugeben. Nach der Bestätigung wird der Knoten gespeichert und das Spiel kehrt zurück zum Hauptmenü.
3. Im Hauptmenü auf den Text „Creative“ klicken.
4. Durch einen Klick auf den Text „LOAD Knot“ im Creative-Menü öffnet sich das Knoten-Lademenü.
5. Im Knoten-Lademenü kann man aus der linken Liste den zuvor abgespeicherten Knoten auswählen und mit dem „Load“ -Button laden.

## **FT\_80** *Manuelle Positionierung der Kamera.*

Mit den folgenden Tastatureingaben kann der Spieler die Kamera manuell bewegen. Die Tastatureingaben sind der Standardtastaturbelegung entnommen (siehe Spielanleitung/Kamerabewegung).

- Mit den „WASD“ Tasten bewegt der Spieler die Kamera nach oben/unten/rechts/links.
- Mit Hilfe der Tasten „R“ und „F“ bewegt der Spieler die Kamera in der Ebene nach vorne und hinten.
- Der Spieler zoomt mit den Tasten „Q“ und „E“ (alternativ mit dem Mausrad) herein- und heraus.
- Der Spieler rotiert die Kamera um eine Kante des Knotens, indem er sie mit der rechten Maustaste auswählt und mit den Pfeiltasten (alternativ durch gedrückt halten der rechten Maustaste) um die Kante rotiert.

- Mittels der „Space“ -Taste springt der Mittelpunkt der Kamera auf den Mittelpunkt der selektierten Kante.
- Mit der linken „Alt“ -Taste wird die Kamera frei gegeben. Mit der Maus schaut sich der Spieler um. Durch erneutes Klicken der linken „Alt“ -Taste rastet die Kamera wieder ein.
- Durch drücken der „ENTER“ -Taste setzt der Spieler die Kamera zurück.

## **FT\_90** *Einrichten und Entfernen des Programms*

Hinweis: Es gibt in der Endversion von Knot3 keine automatische Installation/Deinstallation.

1. Das Archiv in dem sich alle für das Spiel relevanten Dateien befinden lässt sich auf dem Zielsystem entpacken.
2. Durch einen Doppel-Klick auf die ausführbare Datei „Knot3.exe“ startet das Spiel erstmalig im Fenstermodus und das Hauptmenü wird angezeigt. Dabei wird auf dem Zielsystem auch ein Ordner für Einstellungen und Speicherspeicherstände angelegt.
3. Die Deinstallation erfolgt manuell. D.h. alle zu Knot3 gehörigen Ordner sind vom System zu löschen.

## **FT\_100** *Spiel-Modi Starten*

Creative-Mode:

1. Durch einen Klick auf den Text „Creative“ im Hauptmenü öffnet sich das Creative-Menü.
2. Durch einen Klick auf den Text „NEW Knot“, startet der Creative-Mode zum Erstellen eines neuen Knotens.

Challenge-Mode:

1. Durch einen Klick auf den Text „Challenge“ im Hauptmenü öffnet sich das Challenge-Menü.
2. Im Challenge-Menü wird in der Challenge-Liste eine Challenge ausgewählt und durch einen Klick auf den Start-Button gestartet.



## **Komponententests**

Beschreibung der durchgeführten Tests je Komponente.

## Negativtests

Problemeingaben die bereits bekannt sind werden explizit getestet.

### **NT\_10** *Laden nicht gültiger Knoten-Daten.*

Es ist nicht möglich ungültige Knoten-Daten zu laden, da nur gültige Knoten-Daten in der Auswahlliste angezeigt werden. Dies ist durch eine vorherige Überprüfung der Knoten auf Gültigkeit möglich.

### **NT\_20** *Erstellen von Challenge-Leveln aus gleichen Knoten.*

Das Erstellen einer Challenge mit gleichen Knoten ist nicht möglich. Wählt der Spieler beim Erstellen einer Challenge zwei gleiche Knoten aus, so kann er nicht auf den „Create!“ -Button drücken.

### **NT\_30** *Transformieren von Knoten in nicht gültige Knoten.*

Da man nur gültige Transformationen durchführen kann, ist es nicht möglich einen ungültigen Knoten mit Hilfe von Transformationen zu erstellen.

### **NT\_40** *Löschen von Standard-Leveln.*

### **NT\_50** *Verhalten beim Drücken von nicht belegten Tasten.*

Drückt der Spieler nicht belegte Tasten so passiert einfach gar nichts. Das Spiel läuft ohne Probleme weiter und stürzt auch nicht ab. Das Spiel verhält sich so in allen Spielmenüs und wenn man sich in den Spielmodis befindet.

## **Extremtests**

Für extreme Testfälle, z.B. die Verarbeitung großer Datenmengen führen wir Tests durch. Zusätzlich messen wir die Zeit, welche zur Ausführung der Aktion benötigt wird.

## **Abnahmetests**

Wir lassen das Spiel Knot3 von menschlichen Testern spielen. Welche Aufgaben diese dabei auszuführen haben werden hier beschrieben.

## 2.5. Nicht getestet

Hier beschreiben wir nicht getestetes Verhalten und begründen im konkreten Fall unsere Entscheidung einen Test nicht durchzuführen.

### Funktionstests

#### **FT\_1** *Einstellung der Grafikauflösung.*

Die möglichen Einstellungen werden dynamisch vom Betriebssystem angefordert. D.h. die Werte, welche dem Spieler zur Auswahl stehen sind bereits vom Betriebssystem auf Gültigkeit überprüft worden (siehe: `Microsoft.Xna.Framework.Graphics.SupportedDisplayModes`).

### Komponententests

Von den Klassen, die wir für das Unit-Testing in Betracht gezogen haben, mussten wir diejenigen ausschließen, die für einen entscheidenden Teil ihrer Funktionalität eine oder mehrere Instanzen der Klassen `Game`, `GraphicsDeviceManager`, `GraphicsDevice` und `ContentManager` benötigen. Das bedeutet, dass sie Instanzen dieser Klassen entweder im Konstruktoren erstellen, im Konstruktoren als Parameter erwarten, dass sie teilweise Wrapper um diese Klassen sind oder dass ihre Funktionalität sich auf einige wenige Methoden beschränkt, die mit diesen Instanzen arbeiten.

Dazu gehören einerseits alle von `IRenderEffect` ererbenden Klassen wegen der intensiven Nutzung von `GraphicsDevice` und `GraphicsDeviceManager` sowie teilweise von Instanzen der Klasse `Effect`, das den Zugriff auf Shader kapselt und ebenfalls von `GraphicsDevice` und `ContentManager` abhängt.

Andererseits gehören dazu auch die `GameModels` und davon ererbende Klassen, weil diese eine Instanz von `Model` enthalten, das über einen `ContentManager` geladen werden muss und damit ein `GraphicsDevice` benötigen. Auch die `InputHandler`, deren hauptsächliche Funktion es ist, in bestimmten Methoden, die eventbasiert aufgerufen werden, Listen von `GameModels` zu erstellen, sind damit von `ContentManager` und vom `GraphicsDevice` abhängig.

**Negativtests**

**Extremtests**

**Abnahmetests**

## **2.6. Statistik**

### **2.6.1. Abdeckung**

Auf den folgenden Seiten steht der aus den Daten von OpenCover generierte Bericht zur Testabdeckung durch Komponententests. Die Berechnung der Testabdeckung erfolgt zeilenweise.

## Summary

**Generated on:** 06.03.2014 - 10:13:08  
**Parser:** OpenCoverParser  
**Assemblies:** 2  
**Classes:** 44  
**Files:** 44  
**Coverage:** 59.3%  
**Covered lines:** 1570  
**Uncovered lines:** 1076  
**Coverable lines:** 2646  
**Total lines:** 7601

## Assemblies

<b>Knot3</b>	<b>66.5%</b>
Knot3.Framework.Audio.Knot3Sound	100%
Knot3.Game.Audio.Knot3AudioManager	90.9%
Knot3.Game.Data.Challenge	0%
Knot3.Game.Data.ChallengeFileIO	0%
Knot3.Game.Data.ChallengeMetaData	0%
Knot3.Game.Data.CircleEntry‘1	92.3%
Knot3.Game.Data.CircleExtensions	100%
Knot3.Game.Data.Direction	100%
Knot3.Game.Data.Edge	100%
Knot3.Game.Data.Knot	84.8%
Knot3.Game.Data.KnotFileIO	27.7%
Knot3.Game.Data.KnotMetaData	64%
Knot3.Game.Data.KnotStringIO	73.1%
Knot3.Game.Data.Node	73.7%
Knot3.Game.Data.NodeMap	88.5%
Knot3.Game.Data.RectangleMap	0%
Knot3.Game.Utilities.FileIndex	44.4%
Knot3.Game.Utilities.SavegameLoader‘2	38.7%
<b>Knot3.Framework</b>	<b>51.9%</b>
Knot3.Framework.Audio.AudioManager	15.5%
Knot3.Framework.Audio.LoopPlaylist	0%
Knot3.Framework.Audio.OggVorbisFile	0%
Knot3.Framework.Audio.Sound	28.5%
Knot3.Framework.Audio.SoundEffectFile	0%
Knot3.Framework.Core.Camera	65.3%
Knot3.Framework.Core.DisplayLayer	98.1%
Knot3.Framework.Core.World	15.9%
Knot3.Framework.Math.Angles3	100%
Knot3.Framework.Math.BoundingCylinder	90.4%
Knot3.Framework.Math.Bounds	94.5%
Knot3.Framework.Math.RayExtensions	68%
Knot3.Framework.Math.ScreenPoint	51.8%
Knot3.Framework.Platform.SystemInfo	100%
Knot3.Framework.Storage.BooleanOption	100%
Knot3.Framework.Storage.Config	66.6%
Knot3.Framework.Storage.ConfigFile	100%
Knot3.Framework.Storage.DistinctOption	100%
Knot3.Framework.Storage.FileUtility	50%
Knot3.Framework.Storage.FloatOption	92%
Knot3.Framework.Storage.IniFile	60.6%
Knot3.Framework.Storage.KeyOption	100%
Knot3.Framework.Storage.Language	0%
Knot3.Framework.Storage.LanguageOption	0%



---

Knot3.Framework.Storage.Localizer	0%
Knot3.Framework.Storage.Option	100%

## 3. Fehler

### 3.1. Übersicht

#### 3.1.1. Klassifizierung

**Programmfehler** Fehler im Programm.

**Grafikfehler** Fehlerhafte grafische Darstellung.

**Fehlend** Fehlender Bestandteil.

**Fraglich**

## 3.2. Werkzeuge

Bei der Fehlersuche unterstützen uns mehrere Programme. Je nach Fehlerklasse (s. 3.1.1) sind verschiedene Werkzeuge hilfreich.

### 3.2.1. Manuell

#### **FastStone Capture** , V. 7.7

FastStone Capture erstellt Bildschirmaufnahmen. Damit lassen sich Screenshots und Videos von mehreren Fenstern machen. In den Videos werden auch Benutzerinteraktionen eingezeichnet. Gerade bei Fehlern, die sich durch grafisches Fehlverhalten äußern und um diese zu dokumentieren, kommt dieses Werkzeug zum Einsatz. Die Screenshots helfen bei der Fehlerbeschreibung. Zudem lassen sich die Videos - deren Größe wenige MB beträgt - einfach ins GIF-Format konvertieren. Das ist besonders hilfreich, da sich bis jetzt unter GitHub nur GIF-Animationen in die textuelle Beschreibung direkt einfügen lassen. Im Gegensatz zu den anderen Werkzeugen ist diese Software Shareware.

Internetseite: <http://www.faststone.org>

#### **GitHub-Issues**

Das durch GitHub bereit gestellte Ticket-System nutzen wir zur Fehlerverfolgung. Sämtliche Probleme sind dort unter

<https://github.com/pse-knot/pse-knot/issues>  
aufgelistet.

### 3.2.2. Automatisiert

#### **Gendarme**

Gendarme durchsucht anhand von Regeln .NET-Code und gibt einen Fehlerbericht aus. Das Werkzeug kontrolliert u. A.:

- Code-Style
- Code-Conventions
- Änderungen, welche die Performance verbessern
- ...

Internetseite: <http://www.mono-project.com/Gendarme>

Während der Laufzeit des Projekts liegt der Gendarme-Bericht unter

<http://www.knot3.de/development/gendarme.html>

vor.

### **3.3. Protokoll**

**Programmfehler**

**Grafikfehler**

**Fehlende Bestandteile**

## 3.4. Statistik

### Gendarme

Anzahl der durch Gendarme gefundenen Probleme verteilt über die Monate von Januar bis März 2014:

Januar	Februar	März
> 1200	~ 600	~ ...

# 4. Änderungen

## 4.1. Protokoll

### 4.1.1. Geändert

Fehler die wir verbessert haben oder Ergänzungen werden hier beschrieben. Kleinigkeiten fließen nicht in das Protokoll ein.

#### **Startmenü, Credits**

vorher ( ↻ )

#### **Transformations-Vorschau**

#### **Spielbeschreibung**

### **4.1.2. Nicht geändert**

Hier werden Probleme die wir nicht behoben haben oder nicht ändern konnten beschrieben.



### **4.1.3. Verschönert**

**Blinkende Sterne**

## **5. Ausnahmen**

### **5.1. Behandlung**

### **5.2. Meldungen**

## **6. Schluss**

### **6.1. Bewertung**

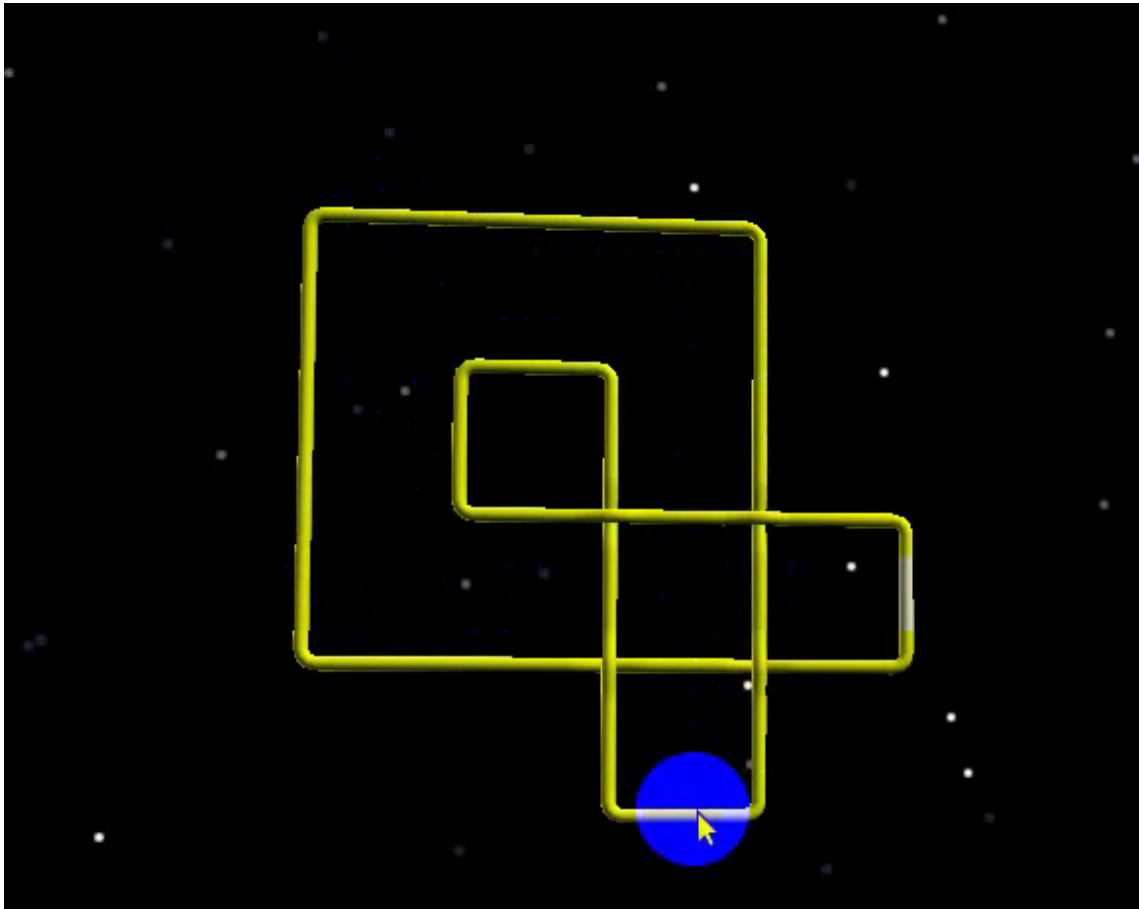
# **A. Anhang**

## **A.1. Aufnahmen**

### A.1.1. Testknoten

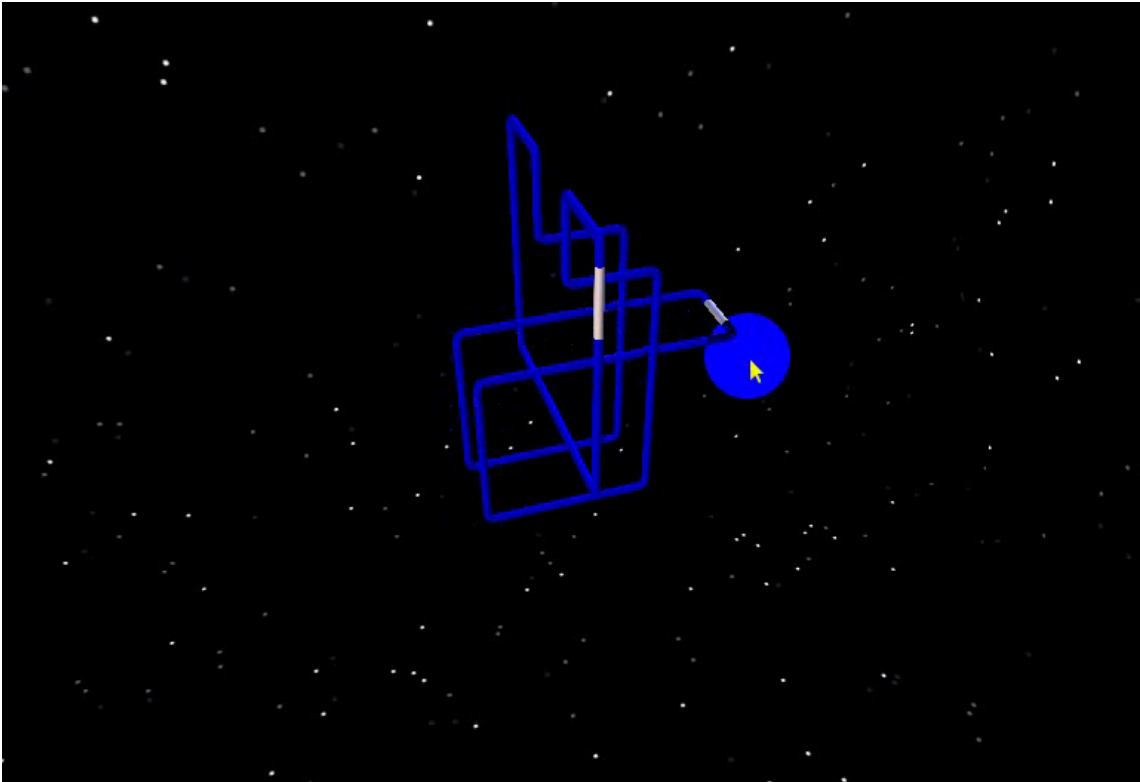
Ein Bilderkatalog der die Knoten zeigt, deren Erstellbarkeit wir explizit prüfen. Hinweis: Mit Adobe Reader ab Version 9 ist es möglich den Knoten-Bau im PDF-Dokument als Animation abzuspielen.

„Überleger“



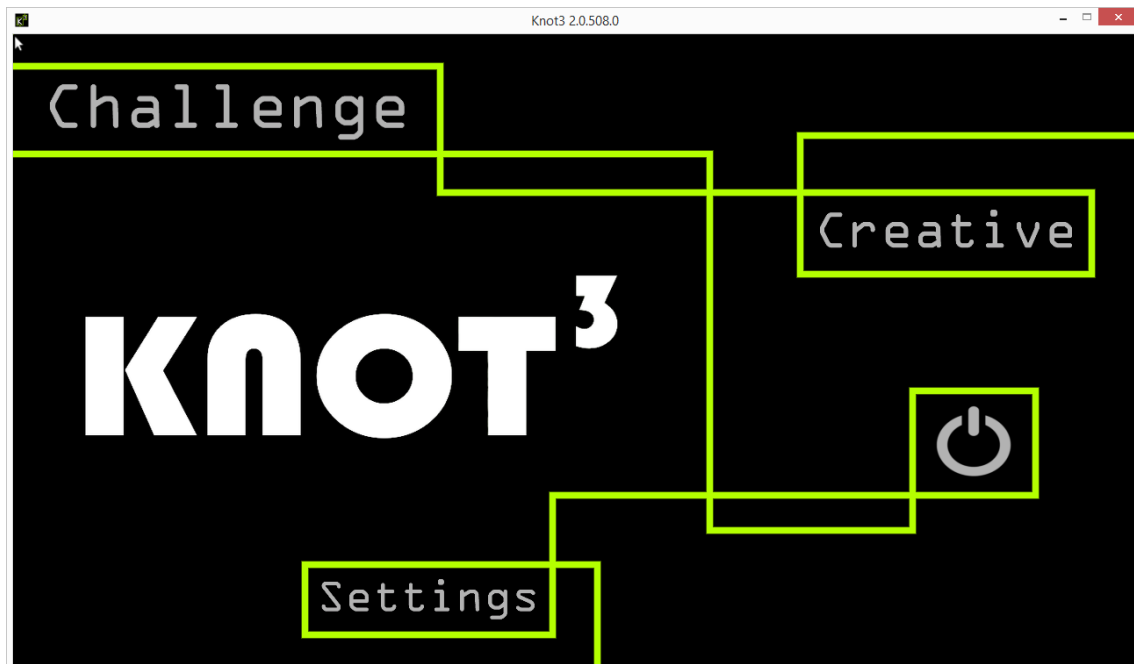
↗ Siehe Funktionstest FT\_30:1.

## „Schlaufe“



↩ Siehe Funktionstest FT\_30:2.

## Änderungen



↗ Siehe Startmenü-Änderung.

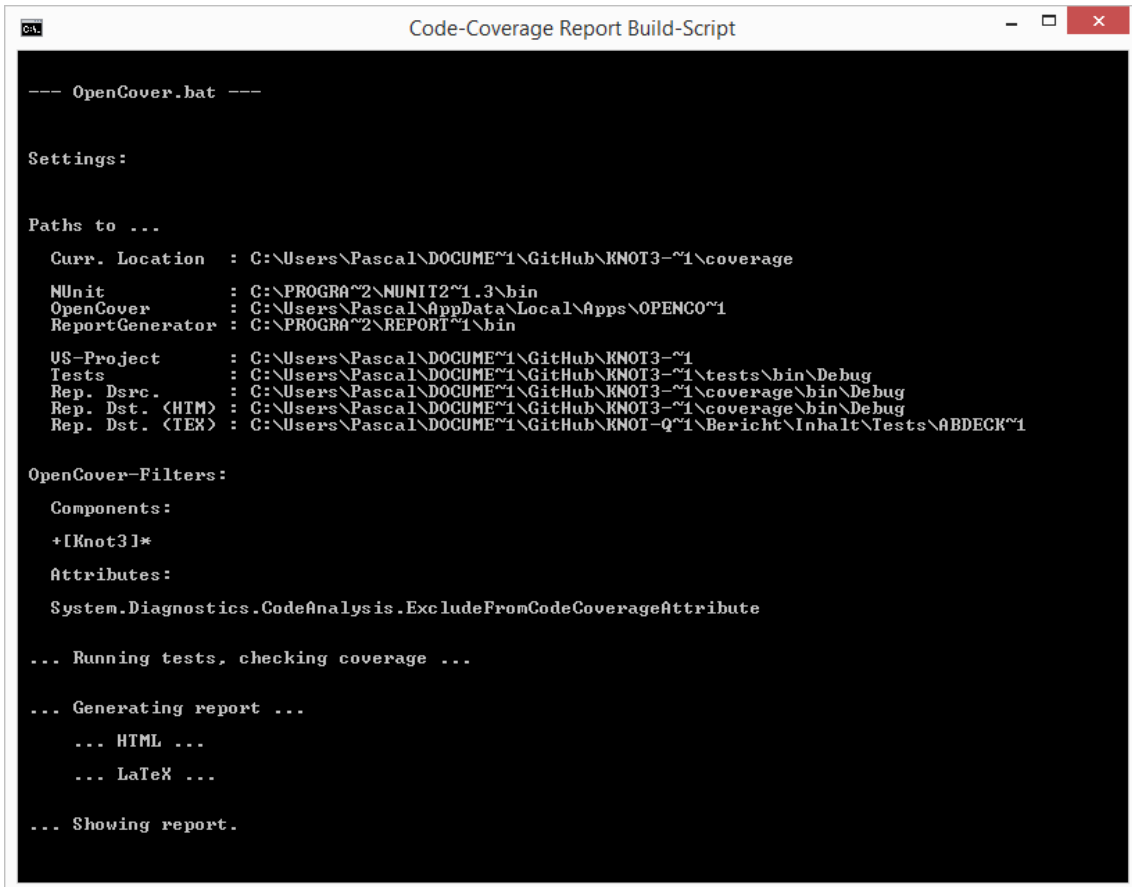
**Fehler**

**Grafikfehler**



## Fehlende Bestandteile

## Werkzeuge



```
Code-Coverage Report Build-Script

--- OpenCover.bat ---

Settings:

Paths to ...

  Curr. Location   : C:\Users\Pascal\DOCUME~1\GitHub\KNOT3-~1\coverage
  NUnit            : C:\PROGRA~2\NUNIT2~1.3\bin
  OpenCover        : C:\Users\Pascal\AppData\Local\Apps\OPENC0~1
  ReportGenerator  : C:\PROGRA~2\REPORT~1\bin

  US-Project       : C:\Users\Pascal\DOCUME~1\GitHub\KNOT3-~1
  Tests            : C:\Users\Pascal\DOCUME~1\GitHub\KNOT3-~1\tests\bin\Debug
  Rep. Dsrc.       : C:\Users\Pascal\DOCUME~1\GitHub\KNOT3-~1\coverage\bin\Debug
  Rep. Dst. <HTM>  : C:\Users\Pascal\DOCUME~1\GitHub\KNOT3-~1\coverage\bin\Debug
  Rep. Dst. <TEX>  : C:\Users\Pascal\DOCUME~1\GitHub\KNOT-~1\Bericht\Inhalt\Tests\ABDECK~1

OpenCover-Filters:

  Components:

  +[Knot3]*

  Attributes:

  System.Diagnostics.CodeAnalysis.ExcludeFromCodeCoverageAttribute

... Running tests, checking coverage ...

... Generating report ...

  ... HTML ...

  ... LaTeX ...

... Showing report.
```

Abbildung A.1.: Build-Batch für die Erstellung des Testabdeckungs-Berichts.