T-TOUCH

—

Projektbericht

to-fuse
Interaction Design | Software Development

Projekt: Table Computing Multi-Player Game - Studienarbeit - HS/2009

Auftraggeber: to-fuse (Zürich), Christian Iten [christian.iten@to-fuse.ch]
Betreuer HSR: Prof. Dr. Markus Stolze, Institut für Software [mstolze@hsr.ch]
Studenten: Mischa Trecco [mischa@3cco.ch]
Ricardo Alvarez [ralvarez@hsr.ch]

Erklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.

Rapperswil, den 18. Dezember 2009

Mischa Trecco, Unterschrift:_____

Ricardo Alvarez, Unterschrift:_____

Danksagung

Die Autoren möchten sich für die vielfältige und kreative Unterstützung während dieser Studienarbeit bei dem to-fuse Team Christian Iten und Emanuel Zgraggen sowie bei unserem HSR-Betreuer Prof. Dr. Markus Stolze herzlich bedanken.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Abstract

Die HSR besitzt seit kurzer Zeit einen Multi-Touch Tisch der Firma to-fuse (Zürich). Für diese Platform existieren diverse Anwendungen wie z.B. der „Salestable“, welcher Unterstützung beim Vergleichen von Mobilgeräten bietet. Die Idee Multi-Player Spiele auf Multi-Touch fähige Oberflächen zu bringen ist nicht neu. Als Beispiel liefert Microsoft mit ihrem Microsoft Surface ein Tetris-ähnliches Spiel „Blox“ für 1-6 Spieler aus. Die Möglichkeit von Multi-Touch Gestiken wird aber bei allen von uns analysierten Spielen zu wenig ausgenützt. Dies wollten wir nun in einem Multi-Player Spiel umsetzen, welches ideal für zwei oder mehr Spieler ist und sich auch für Demonstrationen an Veranstaltungen für potentielle HSR Informatik-Studierende eignen soll.

Das Team entschied sich für den berühmten Spieleklassiker „Tetris“ als Grundlage. Bei der Erweiterung des Grundkonzepts um Multi-Player Funktionalität bot sich eine 3D Lösung an um die beschränkten Platzverhältnisse besser ausnutzen zu können. Unsere Recherchen haben gezeigt, dass keine Table-Top Anwendungen gibt, bei denen die Benutzer durch direkte Berührung 3D-Objekte auf der Multi-Touch Oberfläche manipulieren.

Mit dieser Entwicklung betraten wir Neuland und gerade die Problematik 3D Physics in Verbindung mit Finger Tracking haben wir als besondere Herausforderung identifiziert. Überraschend umfangreich war der benötigte Aufwand für die Entwicklung des 3D-Spieles gegenüber früheren Erfahrungen mit 2D-Spielen. „T-Touch“ hat uns aber gezeigt, dass Multi-Player Spiele in der Multi-Touch Welt viel Potential haben. Die aktuelle Version legt mit den gelösten 3D-Problemen die Grundlage für Weiterentwicklungen. Das Spiel kann nun visuell, interaktiv und allenfalls auch akustisch noch stark ausgebaut werden.

2 Paper

“T-Touch”: A Table Computing Multi-Player Game

Mischa Trecco, Ricardo Alvarez

Abstract—Multi-Touch fähige Oberflächen werden durch ihre neue Art der Kommunikation zwischen Mensch und Maschine immer populärer. Neuste Technologien ermöglichen insbesondere auch das gemeinsame und gleichzeitige Interagieren mehrerer Personen mit der Maschine. Diese Aspekte waren die Motivation das Multiplayer Spiel “T-Touch” für die to-fuse Multi-Touch Plattform zu entwickeln.
Das 3D-Spiel ist eine Neuauslegung des berühmten Spieleklassikers “Tetris” und für zwei Spieler konzipiert. “T-Touch” hat uns deutlich gezeigt, dass der physisch-soziale Aspekt der Plattform, kombiniert mit intuitiver Multi-Touch Technologie und einem guten Spielkonzept zu einem faszinierenden Spielerlebnis führt.
Leider konnte das Team bei der Entwicklung von “T-Touch” nicht auf die Unterstützung eines Grafikers zählen. Folglich halten sich die visuellen Reize der Anwendung in Grenzen. Obwohl sich dies je nach Zielgruppe nur in geringem Masse auf das Spielerlebnis auswirkt, könnte ansprechendere Grafik aber durchaus dazu beitragen das junge Zielpublikum potentieller Informatik-Studenten zu begeistern.

Index Terms—Multi-Touch, Multiplayer, Game, Tetris, Table Computing.

1 Introduction

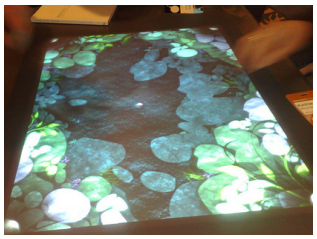
Multiplayer-Spiele gibt es unzählige. In der relativ jungen Multi-Touch Technologie Sparte haben wir jedoch nur wenige gute Referenzen diesbezüglich gefunden. Microsoft liefert mit ihrem Microsoft Surface ein Tetris-ähnliches Spiel “Blox” aus, bei dem es darum geht Reihen von gleichfarbigen Klötzen abzubauen. Dieses haben wir an der Ausstellung im Technorama zum 20 jährigen bestehen von Microsoft ausprobiert. Unserer Meinung nach nützt es aber zu wenig die interaktiven Möglichkeiten, wie z.B. Rotations-Gestik, des Tisches aus. Ein weiteres ist das Firefly von Carbonated Games bei welchem die Spieler Glühwürmchen einsammeln müssen. Andere Spiele existieren für Plattformen wie z.B. das IPhone, welche aber aufgrund der Display-Grösse oft nur im Singleplayer-Modus spielbar sind. An der Ausstellung im Technorama waren auf all den Multi-Touch fähigen Plattformen auch praktisch keine 3D-Visualisierungen zu sehen. Die einzige Applikation welche 3D nutzte, war eine Art “Bildschirmschoner” von Microsoft welcher Wasser simulierte und bei Berührung Wellen erzeugte. Interessante Forschungsresultate zum Thema Multi-Touch Interaktion mehrerer Personen auf derselben Oberfläche bietet das Paper “Exploring Multi-Touch Collaborative Play” der Carleton University in Kanada.

(a) Blox by Microsoft

(b) Firefly by Carbonated Games

Abbildung 1. Multi-Touch Spiele

(a) Labour of Loaf by Carleton University



(b) Watersimulation by Microsoft

Abbildung 2. Weitere Multi-Touch Anwendungen

2 TO-FUSE MULTI-TOUCH PLATTFORM

FÜR das Projekt stand dem Team die von der Firma to-fuse entwickelte "to-fuse Multi-Touch Plattform" zur Verfügung. Die Oberfläche wird von allen Seiten per Infrarot-Licht abgedeckt und Ablenkungen, wie sie durch Fingerberührung entstehen, durch eine Infrarot-Kamera und entsprechende Software erkannt. Die Plattform macht in der Anzahl gleichzeitiger Berührungspunkte keine Einschränkungen wie bei anderen Modellen wie z.B. dem HP TouchSmart600 PC welcher lediglich 2 Touch-Punkte erkennt.

Abbildung 3. to-fuse Multi-Touch Platform

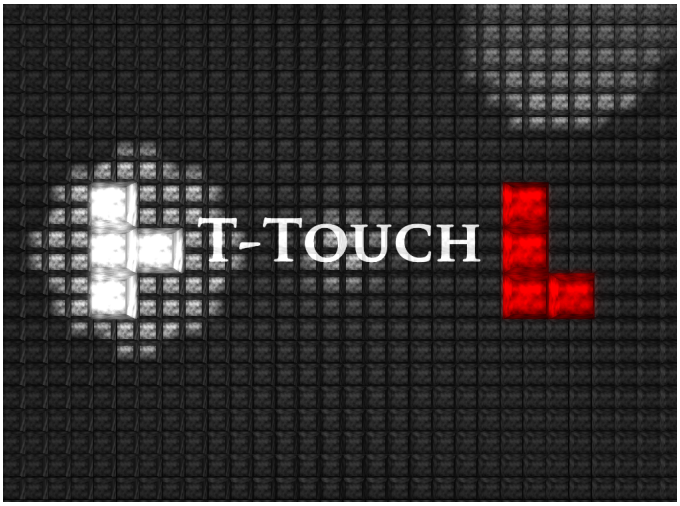


Abbildung 4. Bereitschafts-Screen

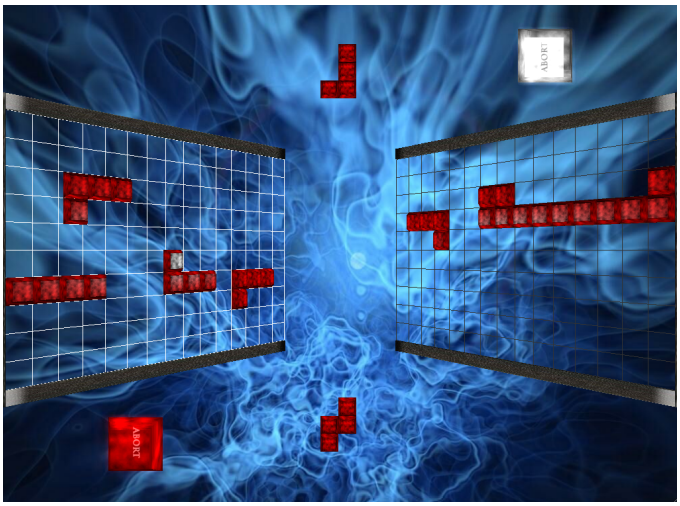


Abbildung 5. Game-Screen

3 SPIELKONZEPT

VOR Beginn des Spieles müssen beide Spieler zuerst ihre Bereitschaft signalisieren. Dazu wird vor jeden Spieler ein zufälliger Tetris-Stein gesetzt. Dieser fungiert als Button und wird von einem umherfahrenden Scheinwerfer beleuchtet sobald er berührt wurde. Im folgenden Bild hat der linke Spieler bereits seine Bereitschaft signalisiert und wartet auf den Gegenspieler.

Sind beide Spieler bereit wird ein neues Spiel gestartet und das Spielfeld sichtbar.

Jeder Spieler hat nun seinen persönlichen Bereich welcher sich in Funktionalität und Aussehen am klassischen Tetris orientiert. Darin können die Steine gedreht und zu Linien verbaut werden.

In der Mitte befindet sich der allgemeine Bereich in welchem regelmässig neue Steine erscheinen. In dieser Zone gibt es keine Einschränkungen und die Spieler kämpfen um die strategisch besten Steine, schieben dem anderen schlechte Steine unter oder zerstören für den anderen Spieler wichtige Steine.

Gelingt es, im eigenen Bereich Linien zu vollenden, gehen diese als unvollständige Straflinien auf den Gegner über. Ein Spieler hat verloren, wenn sich die Steine in seinem Feld bis zum oberen Spielfeldrand türmen.

3.1 Spielfeldanalyse

Um die optimale Anzahl Spieler sowie Form und Verteilung der Spielbereiche zu evaluieren haben wir diverse Prototypen entwickelt. Wir mussten schnell feststellen dass ein 2D Spielfeld keine Lösung war. Es können nicht genug

Linien pro Spielfeld dargestellt werden ohne dass die Steine zu klein und somit inakzeptabel für die Bedienung würden.

Die Idee die Spielfelder als Ebenen in den 3D Raum zu legen löste das Linien-Problem. Allerdings gelang es uns nicht, vier gleichwertige Spielfelder so anzuordnen, dass sich einerseits die Spieler nicht physikalisch behinderten und andererseits keine Spieler benachteiligt wurden. Die Lösung war eine Anordnung für zwei Spieler im 3D Raum. Mit einer quadratischen Oberfläche wäre einer vier Spieler Variante, wie aus dem folgenden Bild ersichtlich, nichts im Weg gestanden.

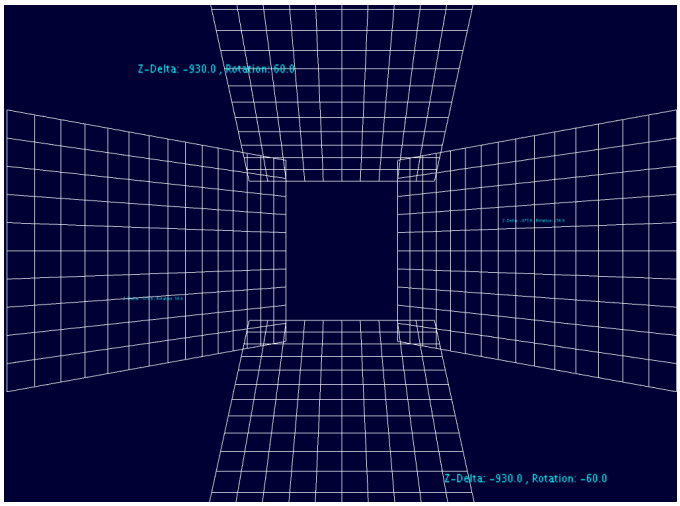
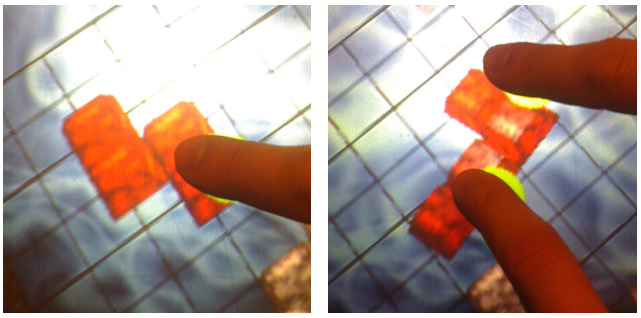


Abbildung 6. Spielfeld Prototyp für 4 Spieler

3.2 Interaktion

Die Spieler können die Tetris-Steine auf einfache Art und Weise ziehen und rotieren. Für die Translation reicht es das Objekt mit einem Finger zu berühren und an die gewünschte Stelle zu ziehen. Um eine Rotation zu erreichen muss der entsprechende Stein zuerst an zwei Stellen berührt werden. Danach kann der Stein durch Verschiebung der Berührungspunkte um deren Mittelpunkt rotiert werden. Eine Rotation beinhaltet auch immer eine Translation des Steins zum aktuellen Mittelpunkt der Berührungsstellen hin. Folgend sind die Interaktionen in Bildern illustriert:



(a) Verschieben (b) Drehen

Abbildung 7. Interaktion mit Tetris-Steinen

3.3 Räumlicher Eindruck

Ein weiterer interessanter Aspekt der 3D Lösung ist das räumliche Gefühl welches den Spielern vermittelt wird. Sie spielen nicht nur mit einfachen 2D Steinen sondern manipulieren diese im 3D Raum. Dies wirkt sich positiv auf das Spielerlebnis aus.

4 Technische Herausforderungen

Als grösste technische Herausforderung galt es die erste, auf dem to-fuse Application Framework basierende, 3D-Anwendung zu implementieren. Das Framework war im 3D-Bereich noch nicht eingesetzt worden. Die Architektur enthielt zwar bereits die nötigen Grundstrukturen, doch gewisse Funktionalitäten wie z.B. das Umrechnen von Treiberkoordinaten auf das Koordinatensystem der 3D-Umgebung, mussten noch angepasst werden. Das zu Beginn fehlende OpenGL-Know-How der Autoren erschwerte es zudem Fehlverhalten der Applikation als Probleme des Frameworks zu identifizieren.

4.1 2D vs. 3D

Der Entscheid die Applikation visuell um eine dritte Dimension zu erweitern führte zu einer Erhöhung der Komplexität. Ein erstes Problem, welches bei sämtlichen 3D Applikationen zum Zuge kommt, ist das Umrechnen der 2D Eingabe-Koordinaten eines Berührungspunktes auf den Punkt des berührten Objektes in der 3D Landschaft. Durch die Kamera-Perspektive

würde bei gleicher Berechnung wie in 2D der Finger dem Objekt davonfahren. Das Objekt legt zwar den gleichen Weg zurück wie der Finger, befindet sich aber durch den räumlichen Einfluss nicht immer unter ihm. Die Lösung ist die Szene ohne Perspektive zu berechnen, eine Gerade durch den Berührungspunkt in die Tiefe zu ziehen und den Schnittpunkt mit dem Objekt (Spielfeld) zu berechnen.

Ein weitere Herausforderung welche die 3D Lösung mit sich bringt ist die Rotation um mehrere Achsen. Rotiert ein Objekt nicht nur um eine Achse müssen die verschiedenen Rotationen kombiniert werden. Dies konnte aber über Multiplikation von Quaternionen gelöst werden.

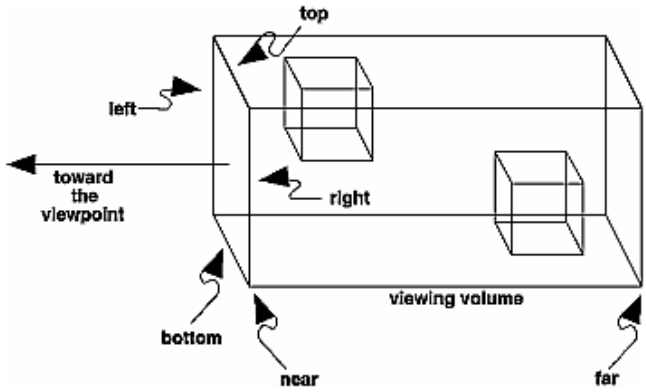


Abbildung 8. 2D Darstellungsfeld

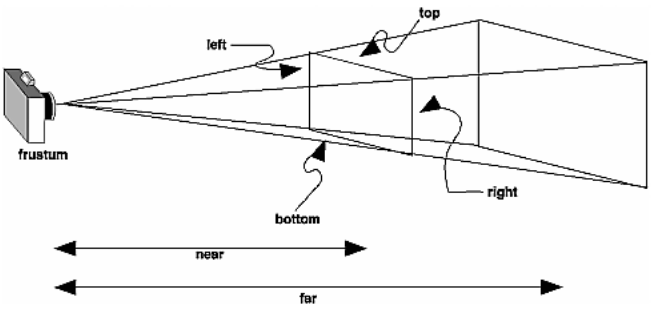


Abbildung 9. Perspektivisches Darstellungsfeld

5 USABILITY

WÄHREND der Entwicklung haben wir festgestellt, dass es auf einer Multi-Touch Oberfläche wenig Sinn macht den Benutzer in seiner Handlung einzuschränken. Am Beispiel Tetris-Stein auf dem Spielfeld lässt sich das gut erklären. Beschränkt man die Bewegungsfreiheit eines Steines auf das Spielfeld, verwirrt das den Spieler weil sein Finger den Stein eigentlich weiterziehen will, dieser aber am Rande des Spielfeldes hängen bleibt. Unserer Erkenntnis nach ist es wesentlich benutzerfreundlicher den Spieler nicht einzuschränken und statt dessen unterschiedlich auf seine Handlungen zu reagieren. In unserem Beispiel wird der Stein zerstört wenn er ausserhalb des Spielfeldes losgelassen wird.

6 AUSBLICK

T-TOUCH bietet diverse Erweiterungsmöglichkeiten. Dazu gehören nebst ansprechenderer Grafik und realistischen visuellen Effekten auch Sound-Unterstützung und die Möglichkeit durch neue Spielelemente weitere Interaktionsmöglichkeiten des Tisches wie z.B. das erkennen von Handflächen einzubringen.

7 CONCLUSION

MULTIPLAYER-SPIELE auf Multi-Touch Oberflächen können Spielern ein faszinierendes Spielerlebnis bieten. Um dieses Potential zu analysieren haben wir das Multi-Touch Spiel “T-Touch” entwickelt. Dabei handelt es sich um eine Neuauflage des berühmten Spieleklassikers “Tetris” und ist für zwei Spieler konzipiert. Bei unseren Beobachtungen haben wir festgestellt dass es auf der intuitiven Multi-Touch Umgebung benutzerfreundlicher ist, dem Spieler möglichst viele Freiheiten im Hantieren mit Objekten zu lassen als die Bewegungsfreiheit der Objekte einzuschränken. Weiter haben wir gesehen, dass sich ein 3D Spiel durch die Vermittlung des räumlichen Gefühls, positiv auf das Spielerlebnis auswirkt und Platzprobleme lösen kann. Der Mehraufwand für die Entwicklung einer 3D Anwendung, welche

von Grund auf komplexer als eine 2D Lösung ist, darf aber keineswegs unterschätzt werden.

"T-Touch" kann visuell und Interaktiv noch stark ausgebaut werden. Generell sind Multiplayer Spiele auf Multi-Touch Oberflächen noch nicht stark verbreitet und bieten noch wesentlich Potential für die Zukunft.

ANHANG A
2D ZU 3D KOORDINATENUMRECHNUNG

Die zweidimensionalen Inputkoordinaten rechnen wir mit folgendem Algorithmus in die dreidimensionalen Koordinaten der Spielwelt um:

1: Ensure input coordinates are based on lower-left origin
2: Calculate first point on line using gluUnProject with z=0
3: Calculate second point on line using gluUnProject with z=1
4: Define plane in space using three mounting points
5: Calculate intersection point between line and plane using

$$P_{hit} = A + c * t_{hit} \quad (1)$$

$$t_{hit} = \frac{n * (B - A)}{n * c} \quad (2)$$

where A is a point on the line, B a point on the plane, n the normale of the plane and c the normale of the line.

ANHANG B
KOMPLETTE LINIEN ABBAUEN

Das folgende Codefragment beschreibt den Algorithmus, der verwendet wird um komplette Linien im Tetrismodell abzubauen.

```
// lines contains the line-numbers to remove
Collections.sort(lines); //ascending
int shiftSize = 0;
int nextLineIndex = lines.size();
int nextClearY = lines.get(--nextLineIndex);
for (int y = nextClearY; y >= 0; y--) {
  if (y == nextClearY) {
    // clear line
    for (int x = 0; x < xSize; x++)
        data[y][x] = null;
    // set next clear line and increase shift size
    if (nextLineIndex > 0)
      nextClearY = lines.get(--nextLineIndex);
    shiftSize++;
  } else {
    // shift line down
    if (shiftSize > 0)
      for (int x = 0; x < xSize; x++) {
        data[y + shiftSize][x] = data[y][x];
        data[y][x] = null;
      }
  }
}
```

DANKSAGUNG

Die Autoren möchten sich für die vielfältige und kreative Unterstützung während dieser Studienarbeit bei dem to-fuse Team Christian Iten und Emanuel Zgraggen, sowie bei unserem HSR-Betreuer Prof. Dr. Markus Stolze herzlich bedanken.

REFERENZEN

[1] F.S. Hill, JR. und Stephen M. Kelley, *Computer Graphics using OpenGL*, 3rd ed. Upper Saddle River, USA: Pearson Prentice Hall, 2006.
[2] R. Khaled, P. Barr, H. Johnston, R. Biddle, *Let's Clean Up This Mess: Exploring Multi-Touch Collaborative Play.* In CHI '09, ACM Press 978-1-60558-247-4/09/04.
[3] S. Hsiao-Heng Chang, L. Stuart, B. Plimmer, B. Wünsche, *Origami Simulator: a Multi-Touch Experience.* In CHI '09, ACM Press 978-1-60558-247-4/09/04.
[4] *to-fuse* http://www.to-fuse.com
[5] *HSR* http://www.hsr.ch
[6] *Carbonated Games* http://www.carbonatedgames.com
[7] *Microsoft Surface* http://www.microsoft.com/surface/
[8] *Firefly-Review* http://sarcasticgamer.com/wp/index.php/2008/02/fi
the-first-game-for-microsoft-surface.html/1

Ricardo Alvarez Informatik-Student an der HSR Hochschule für Technik Rapperswil seit 2007.
email: ralvarez@hsr.ch

Mischa Trecco Informatik-Student an der HSR Hochschule für Technik Rapperswil seit 2007.
email: mtrecco@hsr.ch

3 Aufgabenstellung

Aufgabenstellung Studienarbeit Ricardo Alvarez und Mischa Trecco

Table Computing Multi-Player Game

1. Auftraggeber und Betreuer

Diese Studienarbeit findet in Zusammenarbeit mit der Firma to-fuse (Zürich) statt.

Ansprechpartner Auftraggeber:

- Christian Iten, to-fuse (Zürich)
  christian.iten@to-fuse.ch

Betreuer HSR:

- Prof. Dr. Markus Stolze, Institut für Software mstolze@hsr.ch

2. Ausgangslage

Die HSR besitzt seit kurzer Zeit einen Multi-Touch Tisch der Firma to-fuse (Zürich). Die Erstellung von Multi-Touch User Interfaces für Interaktive Tische erfordert die Verwendung neuer Programmier-Methoden und neuer Interaktionstechniken.

3. Ziele und Aufgabenstellung

In dieser Studienarbeit soll ein Multi-Player Spiel für den to-fuse Tisch entwickelt werden. Dieses Spiel soll die interaktiven Fähigkeiten des Tisches demonstrieren und ein Spielerlebnis ermöglichen welches auf traditionellen Computern mittels Maus, Tastatur oder Joystick nicht erreichbar ist. Die Anwendung soll sich insbesondere gut für die Demonstration an Veranstaltungen für potentielle HSR Informatik-Studierende eignen.

4. Zur Durchführung

Mit den HSR-Betreuern finden in der Regel wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen. Besprechungen mit dem Auftraggeber werden nach Bedarf durchgeführt.

Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das den Betreuern, dem technischen Berater und dem Auftraggeber per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

5. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in 5 Exemplaren abzugeben. Auf Wunsch ist für den Auftraggeber eine gedruckte Version zu erstellen. Zudem ist eine kurze Projektresultatdokumentation im Wiki von Prof. Stolze zu erstellen. Weiterhin erwünscht ist die Erstellung eines kurzen Videos sowie die Erstellung einer Kurzzusammenfassung welche die wichtigsten Resultate und Erkenntnisse der Arbeit auf maximal fünf Seiten zusammenfasst.

6. Termine

Siehe auch Terminplan auf https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html

Montag, den 14. September 2009	Beginn der Studienarbeit, Ausgabe der Aufgabenstellung durch die Betreuer
18.12.2009	Abgabe Kurzbeschreibung an das Abteilungssekretariat mit folgendem Formular gemäss https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html
Freitag, den **18. Dezember 2009, 17:00**	**Abgabe** des Berichtes an die Betreuer

Studenten sollten möglichst bald mit to-fuse einen Termin für eine Einführung in das to-fuse Programming Framework abmachen.

7. Beurteilung

Eine erfolgreiche Studienarbeit erhält 8 ECTS-Punkten (1 ECTS Punkt entspricht einer Arbeitsleistung von ca. 25 bis 30 Stunden). Für die Modulbeschreibung der Studienarbeit siehe https://unterricht.hsr.ch/staticWeb/allModules/10938_M_SAI.html

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/5
2. Berichte (Abstract, Mgmt Summary, techn. u. persönliche Berichte) sowie Gliederung, Darstellung, Sprache der gesamten Dokumentation	1/5
3. Inhalt*)	3/5

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit festgelegt.

Im Übrigen gelten die Bestimmungen der Abt. Informatik zur Durchführung von Studienarbeiten.

Rapperswil, den 14. September 2009

Prof. Dr. Markus Stolze, Institut für Software, Hochschule für Technik Rapperswil

4 Projektplan

4.1 Änderungshistorie

Rev.	Datum	Wer	Was wurde geändert
0	20.09.2009	ral	Neues Dokument, Initialversion
1	29.09.2009	ral, mtr	Ergänzungen
1	30.09.2009	mtr	Qualitätskontrolle

4.2 Einführung

4.2.1 Zweck

Dieses Dokument bietet einen Überblick über Organisation und Planung des Projektes.

4.2.2 Gültigkeitsbereich

Das Projekt basiert auf diesem Dokument. Somit bleibt die Gültigkeit während der gesamten Projektdauer bestehen.

4.2.3 Definitionen und Abkürzungen

Definitionen und Abkürzungen sind im Glossar, einem separaten Dokument, zu finden.

4.2.4 Übersicht

In den folgenden Abschnitten werden verschiedene Details der Projektplanung beschrieben. Diese sollen beim Projektmanagement als Leitfaden dienen und die Kontinuität während dem Projekt wahren.

4.3 Projektübersicht

Bei diesem Projekt wird das Spiel Tetris für eine Multitouch-Oberfläche implementiert. Das Spielprinzip orientiert sich stark am klassischen Tetris, wird allerdings primär für mehrere Spieler ausgelegt sein und einige Erweiterungen aufweisen, um die Vorzüge einer Multitouch-Oberfläche Nutzen zu können. Es folgt eine kleine Erläuterung der Spielelemente und des Spielablaufs.

Das Spielfeld teilt sich in verschiedene Bereiche. In der Mitte ist ein allgemeiner Bereich, wo die Steine erscheinen. Darunter hat jeder Spieler seinen persönlichen Bereich, der sich in Funktionalität und aussehen am klassischen Tetris orientiert. Darin können die Steine gedreht und bewegt werden und müssen zu Linien gruppiert werden wie in der Urfassung. Diesen Bereich können die Steine nicht mehr verlassen.

Der grosse Unterschied liegt im Erscheinen der Steine im mittleren Bereich. Die Steine erscheinen in definierten Gebieten und bewegen sich in Richtung eines persönlichen Spielfelds. Die Steine können hier von jedem Spieler in beliebiger Richtung verschoben werden und besitzen ein physikalisches Verhalten, so dass sie auch angestossen werden können. Ziel ist es, sich benötigte Steine zu sichern und in den eigenen Bereich zu ziehen, um sie zu verbauen. Falls zwei Spieler den selben Stein greifen, wird der Stein auseinander gezogen bis zu einer Maximalgrösse und spickt dann in seine Ursprungsform zurück. Danach ist der Stein wieder frei, muss also neu ergriffen werden.

Gelingt es, im eigenen Bereich Linien zu vollenden, gehen diese als unvollständige Straflinien auf die Gegner über, die diese wieder abbauen müssen. Ein Spieler hat verloren, wenn sich die Steine in seinem Feld bis zu einer gewissen Höhe türmen.

Als weiteres Element gibt es explodierende Steine, welche entsprechend gekennzeichnet sind. Diese können normal verbaut werden und explodieren erst, wenn sie Teil einer vollständigen Linie sind. Dabei nehmen sie noch ein paar umliegende Steine mit. Natürlich gibt es entsprechend mehr Strafsteine für die Gegner.

Um ein schnelles Ende zu ermöglichen ist ein Sudden Death Modus geplant. Jeder Spieler hat einen Knopf, mit dem er seine Bereitschaft für diesen Modus signalisieren kann. Sobald alle zugestimmt haben, wird er aktiviert und die erhaltenen Straflinien werden multipliziert, so dass das Spiel bald zu einem Ende kommt.

Die Spieleranzahl in einem Spiel soll dynamisch sein. Das heisst ein Spieler kann dazukommen und sein Feld aktivieren. Sobald dieses erschienen ist, bekommt er einen Teil der unvollständigen Linien der anderen Spieler, um ein faires Spiel zu garantieren. Umgekehrt läuft es, wenn jemand das Spiel verlässt.

4.3.1 Ziel

Ziel ist es, die Vorzüge und Eigenheiten eines Multitouch-Tisches im Rahmen eines Spieles auszuloten und mit den Gegebenen Möglichkeiten ein interessantes und interaktives Spielerlebnis zu bieten.

4.4 Projektorganisation

Das Projektteam besteht aus zwei einander gleichgestellten Personen. Prof. Dr. Markus Stolze fungiert dabei als Betreuer, d.h. er unterstützt das Projekt bei offenen Fragen und ist auch für die Notengebung zuständig.

Aufgrund der Teamgrösse und Organisation gemäss Scrum werden die Aufgaben bei den regelmässigen Meetings verteilt und Verantwortlichkeiten als Team getragen.

4.4.1 Externe Schnittstellen

Betreuer des Projekts ist Prof. Dr. Markus Stolze, vom Institut für Software an der Fachhochschule Rapperswil.

Als Industriepartner stellt uns die Firma to-fuse aus Zürich nebst Know-How die Hardware und das Framework zur Verfügung, um die Entwicklung des Spieles zu ermöglichen.

4.5 Management Abläufe

4.5.1 Projektkostenvoranschlag

Dem Projekt stehen pro Woche und Teammitglied ca. 16 Stunden zur Verfügung. Der Projektstart wurde auf den 14. September 2009 festgelegt und der Abgabetermin auf den 18. Dezember 2009. Sollten unerwartete Probleme auftreten, kann die Arbeitszeit pro Woche um max. 10 Stunden angehoben werden.

4.5.2 Projektplan

Das Projekt wird gemäss der Richtlinien den Softwareentwickung nach SCRUM geführt. Da es um die Entwicklung eines neuen Produktes geht, dass zu Beginn nur als grobe Idee existiert, ist es wichtig einen agilen Entwicklungsprozess zu haben, um flexibel auf Erkenntnisse reagieren zu können.

Sprints / Meilensteine

Gemäss den Richtlinien von Scrum ist die Entwicklung in Sprints unterteilt, welche zwei bis drei Wochen dauern. Die folgende Liste bietet einen Überblick der Sprints dieses Projektes.

Sprint	Beschreibung	Ende
1	Projektplanung, Risikoeinschätzung	SW 2
2	Steine manipulieren, Spielerfeld-Visualisierung	SW 5
3	Spielerfeld-Funktionalität	SW 8
4	Gesamtes Spielfeld (ink. Battlezone) spielbar	SW 11
5	Spiel fertigstellen	SW 13
6	Projektabschluss und Abgabe	SW 14

Zeitplan

Den Überblick über den Zeitplan des Projekts bietet der *Product Backlog*. Darin ist der noch zu erwartende Aufwand und die bisher geleisteten Arbeitsstunden über das ganze Projekt aufgelistet.

Je einzelnen Sprint gibt es ein *Sprint Backlog* mit der detaillierten Darstellung der Arbeitszeit pro Aufgabe, auch hier aufgeteilt in geleistete Stunden und noch zu erwartender Aufwand.

Die *Zeitpläne inkl. Aufwandschätzungen* werden täglich nachgeführt, so dass der aktuelle Stand des Projektes jederzeit ersichtlich ist und falls nötig Massnahmen ergriffen werden können.

Änderungen in der Projektplanung

Allfällige Änderungen im Projektplan werden an den Meetings besprochen und müssen von allen Teammitgliedern abgesegnet werden. Werden Änderungen beschlossen werden diese in den betreffenden Dokumenten nachgeführt.

Besprechungen

Wöchentliche Sitzungen mit dem Betreuer Prof. Dr. Markus Stolze finden in der Regel Dienstags statt. In diesem Meeting werden Wochenplanung, Pendenzen und allfällige Probleme besprochen und in einem Protokoll festgehalten.

4.6 Infrastruktur

4.6.1 Räumlichkeiten

Der Arbeitsplatz ist im Zimmer 1.262 in der HSR in Rapperswil, wo auch die *to-fuse Multi-Touch Platform* zur Verfügung steht.

4.6.2 Hardware

- *to-fuse Multi-Touch Platform*
- HSR-Arbeitsrechner in den Übungsräumen der HSR

- Private Laptops der Teammitglieder
- Private Drucker oder Drucker der HSR
- Versionsverwaltungsserver der HSR

4.6.3 Software

- Eclipse SDK 3.5.0
- Versionsverwaltungssoftware SVN

4.7 Qualitätsmassnahmen

4.7.1 Dokumentation

Damit das Projekt eine hohe Qualität erreichen kann, werden die Dokumente stets aktuell gehalten, damit sich die Gruppenmitglieder auf deren Inhalt verlassen können.

4.7.2 Sitzungen

Über sämtliche Sitzungen wird Protokoll geführt. Das schriftliche Festhalten des Besprochenen hilft weitere Arbeiten zu koordinieren und eventuelle Missverständnisse auszuräumen.

Die Protokolle werden dem Betreuer und dem Industriepartner zur Information weitergeleitet.

4.7.3 Reviews

Sämtliche Dokumente wie auch der Code werden von einem weiteren Teammitglied korrekturgelesen und mit dem Verfasser nachbesprochen. So wird die Qualität des Dokuments erhöht.

4.7.4 Codequalität und Codestyle

Codequalität und Stil halten sich an die gängigen Konventionen der Javaprogrammierung.

4.7.5 Test

Unit-Test

Sämtlicher Code soll wo möglich mit Unit-Tests geprüft werden, insbesondere bevor dieser ins SVN eingecheckt wird.

Usability Test

Testpersonen überprüfen die Prototypen. Das Feedback aus diesen Tests soll Schwächen im Programmablauf und der Benutzerschnittstelle aufzeigen.

4.7.6 Risikomanagement

Das Risikomanagement wird im separaten *Riskmanagement* geführt.

4.7.7 Bug Reporting

Fehler im Programm, die während des Testens gefunden werden, sind in der *Bug List* festgehalten und verwaltet.

5 Anforderungsspezifikation

5.1 Änderungshistorie

Rev.	Datum	Wer	Was wurde geändert
0	18.11.2009	mtr	Neues Dokument
1	23.11.2009	mtr	Allgemeine Beschreibung
2	23.11.2009	ral	Funktionale Anforderungen
2	24.11.2009	mtr	Qualitätskontrolle funktionale Anforderungen
3	25.11.2009	mtr	Nichtfunktionale Anforderungen
3	25.11.2009	ral	Qualitätskontrolle nichtfunktionale Anforderungen
4	7.12.2009	mtr	Anpassungen gemäss Feedback Betreuer vom 5.12.06

5.2 Allgemeine Beschreibung

5.2.1 Einführung

Die Softwareanforderungsspezifikation gibt Auskunft über funktionale und nichtfunktionale Anforderungen des Projektes.

5.2.2 Ziel und Zweck von T-Touch

Bei T-Touch handelt es sich um ein Multiplayer Spiel, das auf dem berühmten Spieleklassiker ”Tetris” aufbaut und für die *to-fuse Multi-Touch Platform* entwickelt wurde. Das Spiel setzt auf die innovativen Interaktionsmöglichkeiten der Platform und ermöglicht so ein einzigartiges Spielerlebnis, welches auf traditionellen Computern mittels Maus, Tastatur oder Joystick nicht erreichbar ist. Die Anwendung soll sich insbesondere gut für die Demonstration an Veranstaltungen für potentielle HSR Informatik-Studierende eignen.

5.2.3 Produkt Perspektive

Da das Spiel spezifisch für die *to-fuse Multi-Touch Platform* entwickelt wird, ist die Verbreitung abhängig vom Erfolg und Einsatz dieser Platform. Die HSR welche im Besitz einer solchen Platform ist, will das Spiel bei Informationsveranstaltungen der Informatik-Abteilung zu Demonstrationszwecken einsetzen. Ob und wie oft das Spiel tatsächlich eingesetzt wird, hängt aber nicht zuletzt mit dem erreichten Spielerlebnis des Produktes zusammen.

5.2.4 Produkt Funktion

T-Touch ist für zwei Spieler konzipiert, die gegeneinander antreten. Der Spieler wird aber im Gegensatz zum ”Tetris”-Klassiker nicht durch ein immer schneller werdendes Spiel, sondern durch den Erfolg seines Gegners gefordert. Das heisst, wenn der eine Spieler erfolgreich spielt schadet das dem anderen. Die innovative Möglichkeit, die Steine mit den Fingern zu bewegen, ermöglicht es auch, diese in bereits vorhandene ”Löcher” zwischen verbauten Reihen zu platzieren.

5.2.5 Produkte Umfeld

Spiele die auf Tetris basieren gibt es unzählige. In der relativ jungen Multi-Touch Technologie Sparte sind jedoch sehr wenige gute Referenzen zu finden. Microsoft liefert mit ihrem Microsoft Surface ein ähnliches Spiel aus, bei dem es darum geht Reihen von gleichfarbigen Klötzen abzubauen. Dieses haben wir an der Ausstellung im Technorama zum 20 jährigen bestehen von Microsoft ausprobiert. Unserer Meinung nach nützt es aber zu wenig die interaktiven Möglichkeiten, wie z.B. Rotations-Gestik, des Tisches aus. Andere Spiele existieren für Plattformen wie z.B. das IPhone, welche aber aufgrund der Display-Grösse oft nur im Singleplayer-Modus spielbar sind. An der Ausstellung im Technorama waren auf all den Multi-Touch fähigen Plattformen auch praktisch keine 3D-Visualisierungen zu sehen. Die einzige Applikation welche 3D nutzte, war eine Art "Bildschirmschoner"welcher Wasser simulierte und bei Berührung Wellen erzeugte.

5.2.6 Benutzer Charakteristik

Zur Zielgruppe gehören potentielle HSR Informatik-Studierende.

5.2.7 Einschränkungen

Einzige Einschränkung ist die vorgegebene Umgebung in der das Spiel laufen muss. Es handelt sich dabei um die von to-fuse entwickelte *to-fuse Multi-Touch Platform* , zu der auch ein auf OpenGL basierendes Framework gehört. Zu den einschränkenden technischen Faktoren der Plattform sind folgende in Betracht zu ziehen:

Grafikauflösung
Die Pixelauflösung schränkt die Anzahl der möglichen Spielfelder und deren Grösse ein.

Reaktionszeit und Genauigkeit
Das Spielerlebnis hängt direkt damit zusammen wie schnell und genau die Plattform Finger-Gesten erkennen kann.

Abbildung 5.1: *to-fuse Multi-Touch Platform*

5.2.8 Prototyping

Zu Beginn des Projektes war nicht klar, wie tauglich die Plattform für das Spiel ist, welche Möglichkeiten genutzt werden können und welche Konzepte und Ideen zu einem guten Spielerlebnis führen werden. Aus diesem Grund wurde entschieden, diverse Ideen und Konzepte in Prototypen auszutesten und zu analysieren, bevor diese endgültig in das Endprodukt fliessen.

Diese Erfahrungen sind im *Entwicklungsprotokoll* dokumentiert.

Aus den Ergebnissen dieser Pre-Production Phase, welche mehrere Sprints beinhaltete, wurde schlussendlich der endgültige Funktionsumfang (siehe *Kapitel 5.8: "Funktionale Anforderungen"*) für das Endprodukt "T-Touch" definiert.

5.3 Qualitätsanforderungen

5.3.1 Benutzbarkeit

Verständlichkeit und Erlernbarkeit

Der Benutzer muss ohne Anleitung, innerhalb von 5 Minuten, Spielkonzept und -steuerung so verstanden haben, dass er spielen kann.

Einheitlichkeit

Die Bedienung muss für jeden teilnehmenden Spieler äquivalent sein.

5.3.2 Effizienz

Verbrauchsverhalten

Das Spiel darf maximal die von der *to-fuse Multi-Touch Platform* zur Verfügung gestellten Hardware-Ressourcen ausnutzen.

Zeitverhalten / Framerate

Die Spieler sollen kein Flimmern/Stocken wahrnehmen. Das Zeichnen der Oberfläche muss daher mit mind. 25 Frames pro Sekunde erfolgen.

5.3.3 Produktnutzungszeitraum

Produzierbarkeit

Das Spiel wird in Form einer Archiv-Datei, die als Projekt in Eclipse importiert werden kann, sowie in Form einer Library welche über eine mitgelieferte .bat-Datei gestartet werden kann, ausgeliefert.

Langlebigkeit

Die Langlebigkeit der Applikation wird durch regelmässige Codereviews mit allfälligem Refactoring sowie der Verwendung von gängigen Code-Richtlinien (siehe *Projektplan*) sichergestellt.

5.3.4 Übertragbarkeit

Installierbarkeit

Das Spiel muss auf der *to-fuse Multi-Touch Platform* vorinstalliert werden, die Installierbarkeit wird nicht weiter unterstützt.

5.3.5 Zuverlässigkeit

Fehlertoleranz

Interaktionen von Spielern auf der *to-fuse Multi-Touch Platform* , die keine Auswirkungen auf das Spiel haben führen nicht zu Fehlverhalten des Spiels.

Reife

Das Spiel muss während einer Stunde nach Start der Applikation stabil, d.h. ohne Absturz verursacht durch die Applikation selbst, laufen.

Wiederherstellbarkeit

Bei einem aufgetretenen Applikationsfehler kann das Spiel von einer Person, die mit der *to-fuse Multi-Touch Platform* bereits vertraut ist, innerhalb von 10 Minuten wieder in Betrieb genommen werden.

5.3.6 Funktionalität

Angemessenheit

Die interaktiven Möglichkeiten der *to-fuse Multi-Touch Platform* sollen durch das Spiel möglichst ausgenutzt werden. Natürlich gilt dies nur für Funktionalität bei der entsprechende Interaktionen angemessen und sinnvoll sind. Im Minimum müssen aber die Bereiche Rotation, Drag & Drop, Multiplayer abgedeckt sein.

5.4 Technische Anforderungen

5.4.1 Lösungen

Programmiersprache

Von der *to-fuse Multi-Touch Platform* wird als Programmiersprache Java vorgegeben.

Software / Architektur

T-Touch muss auf das bestehende Applikationsframework der *to-fuse Multi-Touch Platform* aufbauen. to-fuse lässt die Möglichkeit offen das Applikationsframework wo sinnvoll zu erweitern.

Hardware

Als Hardware-Umgebung muss das Spiel auf der *to-fuse Multi-Touch Platform* aufsetzen. Die HSR stellt diese zur Verfügung.

5.5 Anforderungen an sonstige Lieferbestandteile

5.5.1 Software Dokumentation

Allgemein

Die Dokumentation muss den Vorgaben für Diplom-, Bachelor- und Studienarbeiten der Abteilung Informatik entsprechen. Diese sind auf der HSR-Website hinterlegt. Alle Dokumente müssen bei Abgabe den Stand der Arbeit in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in 5 Exemplaren abzugeben. Zudem ist eine kurze Projektresultatdokumentation im Wiki von Prof. Dr. Markus Stolze zu erstellen. Weiterhin ist ein kurzes Video sowie ein Paper, welches die wichtigsten Resultate und Erkenntnisse der Arbeit auf maximal fünf Seiten zusammenfasst, zu erstellen.

to-fuse Applikationsframework

Erweiterungen an dem Applikationsframework der *to-fuse Multi-Touch Platform* müssen nachvollziehbar sein und dokumentiert werden.

Installationshandbuch

Es ist ein Installationshandbuch in Form eines Readme-Files zu erstellen.

5.5.2 Projekt Management

Es ist ein Projektplan zu erstellen. Ein kontinuierlicher und sichtbarer Arbeitsfortschritt muss jederzeit nachvollziehbar sein.

5.6 Anforderungen an durchzuführende Tätigkeiten

5.6.1 Projekt Management

Kommunikationsrichtlinien

Mit dem HSR-Betreuer finden in der Regel wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen. Besprechungen mit dem Auftraggeber werden nach Bedarf durchgeführt. Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das dem Betreuer und dem Auftraggeber per E-Mail zugestellt wird.

5.7 Rechtlich-vertragliche Anforderungen

5.7.1 Vertragliche Anforderungen

Es muss gemäss Vorlage auf der HSR-Website eine Vereinbarung über Urheber- und Nutzungsrechte von Auftraggeber, HSR und den Studierenden unterzeichnet werden.

5.8 Funktionale Anforderungen

5.8.1 Spielablauf

Das Spiel ist für zwei Spieler ausgelegt, welche sich gegenüberstehen und gegeneinander spielen. Zu Beginn sehen die Spieler den Startbildschirm. Sobald beide ihre Bereitschaft kundgetan haben, startet das Spiel.

Jeder Spieler hat einen persönlichen Bereich des Spielfeldes, auf dem er die Tetrissteine gemäss dem Spielziel zu Linien verbaut. Zwischen den beiden Spielergrids befindet sich die Battlezone, wo die neuen Steine für beide Spieler erscheinen. Diese beiden Bereiche werden unten genauer erläutert.

Das Spiel endet, wie in der klassischen Version, wenn bei einem der beiden Spieler die Tetris Steine auf seinem Grid keinen Platz mehr haben, so dass ein Teil des Steines ausserhalb

platziert werden müsste. Alternativ kann das Spiel beendet werden, wenn beide Spieler einverstanden sind. Dafür hat jeder Spieler einen Knopf, den er zum Beenden drücken kann. Das Spiel wird allerdings erst abgebrochen, wenn beide Spieler ihren Knopf gedrückt haben. Das heisst, das Spiel wird erst beendet, wenn beide Spieler damit einverstanden sind. Nach Spielende kehrt die Applikation zum Startbildschirm zurück.

5.8.2 Spielfeld

Battlezone

In diesem Bereich erscheinen regelmässig neue Tetris-Steine. Damit ein interessanteres Spiel entsteht, bewegen sich die Steine stets auf ein Spielerfeld zu und gehen auf dieses über wenn sie es erreicht haben. Der Spieler muss sich dann gezwungener Massen auch um die unerwünschten Steine kümmern.

Tetris-Steine die sich in der Battlezone befinden, können von beiden Spielern in ihren persönlichen Bereich gezogen werden. Natürlich können gewiefte Spieler auch unerwünschte Steine dem Gegner unterjubeln, indem sie diese auf deren Feld schieben.

Persönlicher Bereich

Dieser Bereich besteht aus quadratischen Feldern, auf welchen die Steine bewegt werden. Komplettierte Linien verschwinden und werden beim Gegner zur Strafe als unvollständige Linien eingefügt. Dabei werden alle bereits platzierten Steine nach oben geschoben und die Strafsteine erscheinen auf der untersten Linie. Steine, welche nicht vom Spieler geführt werden, fahren nach unten auf dem Gitter, bis sie am unteren Ende oder an bereits platzierten Steinen angekommen sind. Sobald dies eintrifft, werden diese Steine fest platziert und können nicht mehr durch den Spieler manipuliert werden.

5.8.3 Tetris-Steine

Aussehen

Ein Tetris-Stein besteht aus vier Basissteinen. Ein Basisstein hat eine quadratische Grundfläche mit der selben Kantenlänge wie ein Feld im Spielergrid. Die Oberseite ist auch quadratisch, allerdings mit kleinerer Kantenlänge, damit die Seiten des Steins nicht rechtwinklig sonder schräg auf der Grundfläche liegen. Dies führt zu einem interessanteren graphischen Effekt.

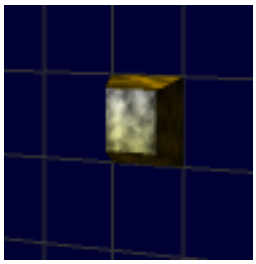


Abbildung 5.2: Ein Basestone, der Grundbaustein der Tetris-Steine

Diese vier Basissteine werden auf alle möglichen Arten kombiniert, so dass neun verschieden aussehende Tetris-Steine entstehen.

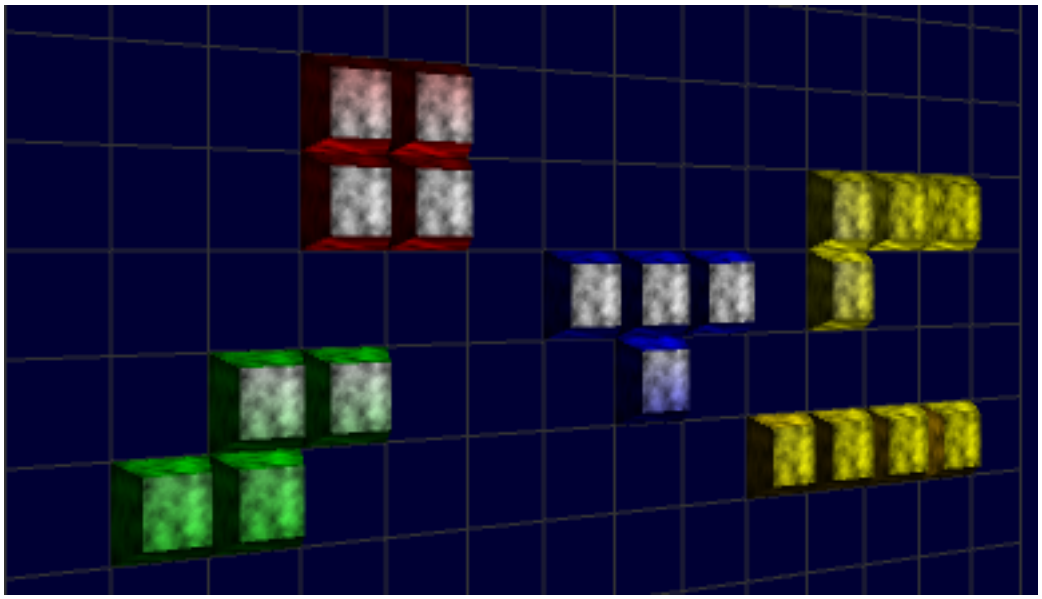


Abbildung 5.3: Verschiedenen Tetris-Steine

Interaktionsmöglichkeiten

In der Battlezone bewegen sich die Tetris-Steine von aussen Richtung Mitte und dann auf eines der persönlichen Felder. Diese Bewegung kann durch die Spieler unterbrochen werden, in dem sie diese in die gewünschte Richtung ziehen.

Auf dem persönlichen Spielbereich des Spielers bewegen sich die Tetris-Steine auf den Feldern des Gitters. Folglich ist ihr Bewegungsspielraum stark eingeschränkt. Allerdings unterliegt der Spieler bei seiner Interaktion mit den Spielsteinen nicht diesen Beschränkungen, sondern kann den Stein frei bewegen und rotieren, solange er ihn führt. Sobald er ihn loslässt platziert sich der Stein in den nächstgelegenen Feldern und passt seine Rotation entsprechend an, da mit schief gelegenen Steinen keine Reihenbildung möglich ist. Auch in Bezug auf die Ränder des Spielfelds wird die Bewegung nicht eingeschränkt. Des weiteren werden die Spielsteine, welche von einem Spieler geführt werden, über die anderen Steine erhoben so dass eine Kollision nicht möglich ist. Dies hat den Hintergrund, dass die Steine an einem Objekt hängenbleiben würden, dies aber nicht vereinbar ist mit den Eingabemöglichkeiten der *to-fuse Multi-Touch Platform* . Die genauen Überlegungen hierzu sind im *Entwicklungsprotokoll* dargelegt.

5.9 Use Cases

5.9.1 Use Case Model

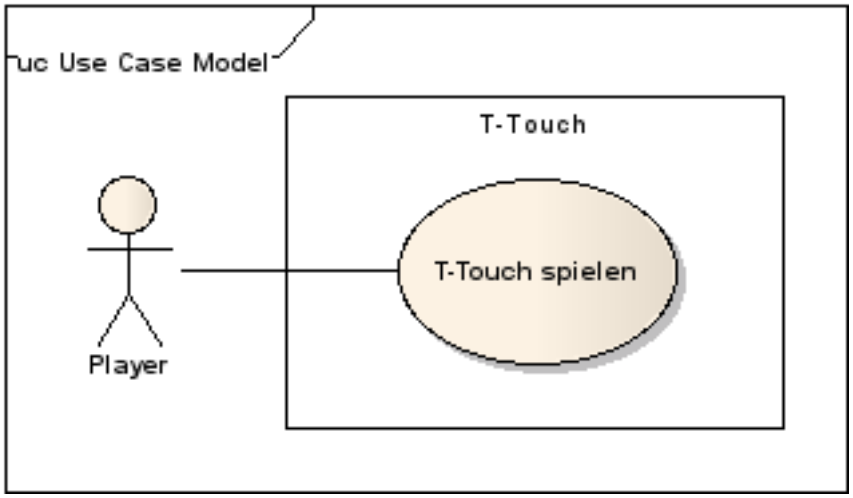


Abbildung 5.4: Use Case Model

5.9.2 Use Case brief

UC1: T-Touch spielen

Ein neues T-Touch Spiel in dem zwei Spieler gegeneinander antreten bestreiten.
Der Spielablauf ist unter *Kapitel 5.8.1: "Spielablauf"* genauer beschrieben.

5.9.3 Aktoren und Stakeholders

Der einzige Aktor des Systems ist der Player (Spieler).

5.9.4 Use Case fully dressed

UC1: T-Touch spielen

Use Case ID	UC1
Umfang	TTouch Applikation
Ebene	Anwenderziel
Primärakteur	User
Stakeholder und Interessen	User: Will T-Touch spielen
Vorbedingung	Zwei Spieler wollen spielen
Nachbedingung	keine
Standardablauf	Schritt 1wird wiederholt, bis beide Spieler diese ausgeführt haben 1. Spieler bestätigt Bereitschaft 2. Spiel wird gestartet Schritte 3-8 werden wiederholt, bis ein Spieler verliert 3. Steine erscheinen in Battlezone 4. Spieler zieht Tetris Stein in sein Feld 5. Spieler zieht und dreht Stein in Position 6. Spieler erstellt Reihen 7. Komplette Reihen werden abgebaut 8. Allfällige Strafsteine werden hinzugefügt 9. Spieler gewinnt 10. Rückkehr zum Startbildschirm
Erweiterungen	3-8a. Spieler markiert Bereitschaft zum Spielabbruch 3-8b. Alle Spieler haben abgebrochen 4a. Stein bewegt sich selber auf ein Spielerfeld 5a. Stein fährt auf Spielerfeld selbständig nach unten, wenn nicht durch Spieler manipuliert 9a. Spiel wird abgebrochen
Spezielle Anforderungen	*to-fuse Multi-Touch Platform*
Häufigkeit des Auftretens	Immer: Dies ist die einzige Möglichkeit zu spielen

5.10 Personas

5.10.1 Ausstellungsbesucher

Profil

Abbildung 5.5: Ausstellungsbesucher

- **Max Welsheim**
- 20 Jahre jung
- Kurz vor Abschluss einer Informatiker-Lehre mit BMS
- Fragt sich was er nach dem Lehrabschluss machen soll
- Besitzt ein IPhone
- Spielt täglich eine Stunde Computerspiele
- Hat das klassische Tetris früher auf dem Game Boy gespielt

Szenario

Max arbeitet täglich auf einem Computer und kennt die Welt der Informatik bereits sehr gut. Er programmiert sehr gerne und findet: ”Am liebsten würde ich selber ein Spiel entwickeln!". Was aktuelle Spiele auf dem Computer angeht ist er immer auf dem neuesten Stand. Was ihm zurzeit am meisten Spass macht ist sein IPhone: ”Das trage ich immer bei mir und teste auch neue Spiele während der Zugfahrt in die Schule.”.

5.10.2 Aussteller

Profil

Abbildung 5.6: Aussteller

- **Hans Jung**
- 35 Jahre alt
- Arbeitet für die HSR
- Organisiert regelmässig Ausstellungen
- Kann die *to-fuse Multi-Touch Platform* an Ausstellungen in Betrieb nehmen

Szenario

Hans hat an den Ausstellungen jeweils ein kleines Kribbeln im Bauch: ”Meine grösste Sorge ist immer ob ich es schaffe die Demo-Applikationen rechtzeitig zu starten!”. Hans schätzt es desshalb sehr wenn alles sauber vorinstalliert und gut zu finden ist.

6 Entwicklungsprotokoll

6.1 Änderungshistorie

Rev.	Datum	Wer	Was wurde geändert
0	13.11.2009	ral	Neues Dokument
1	18.11.2009	ral	Einführung und erste Kapitel
2	20.11.2009	ral	Restliche Prototypen
3	24.11.2009	mtr	Qualitätskontrolle
4	7.12.2009	mtr	Anpassungen gemäss Feedback Betreuer vom 5.12.06

6.2 Einführung

6.2.1 Zweck

Zu Begin des Projektes war nicht klar wie tauglich die Platform für das Spiel ist, welche Möglichkeiten genutzt werden können und welche Konzepte und Ideen zu einem guten Spielerlebnis führen werden. Aus diesem Grund wurde entschieden diverse Ideen und Konzepte in Prototypen auszutesten und zu analysieren bevor diese endgültig in das Endprodukt fliessen.

6.2.2 Definitionen und Abkürzungen

Definitionen und Abkürzungen sind im *Glossar* zu finden.

6.2.3 Übersicht

Diese Erfahrungen sind in diesem Dokument beschrieben. Nach Beschreibung der Spielidee und der Auflistung der Prototypen und ihrer Risiken werden die Erfahrungen und Erkenntnisse, die bei der Umsetzung dieser Ideen gemacht wurden dokumentiert. Aufgrund dieses Dokuments werden die endgültigen Spezifikationen festgelegt.

6.3 Spielldee

6.3.1 Übersicht

Bei diesem Projekt wird das Spiel Tetris für eine Multitouch-Oberfläche implementiert. Das Spielprinzip orientiert sich stark am klassischen Tetris, wird allerdings primär für mehrere

Spieler ausgelegt sein und einige Erweiterungen aufweisen, um die Vorzüge einer Multi-Touch-Oberfläche Nutzen zu können. Es folgt eine kleine Erläuterung der geplanten Spielelemente und des Spielablaufs.

Das Spielfeld teilt sich in verschiedene Bereiche. In der Mitte ist ein allgemeiner Bereich, wo die Steine erscheinen. Darunter hat jeder Spieler seinen persönlichen Bereich, der sich in Funktionalität und Aussehen am klassischen Tetris orientiert. Darin können die Steine gedreht und bewegt werden und müssen zu Linien gruppiert werden wie in der Urfassung. Diesen Bereich können die Steine nicht mehr verlassen.

Der grosse Unterschied liegt im Erscheinen der Steine im mittleren Bereich. Die Steine erscheinen in definierten Gebieten und bewegen sich in Richtung eines persönlichen Spielfelds. Die Steine können hier von jedem Spieler in beliebiger Richtung verschoben werden und besitzen ein physikalisches Verhalten, so dass sie auch angestossen werden können. Ziel ist es, sich benötigte Steine zu sichern und in den eigenen Bereich zu ziehen, um sie zu verbauen. Falls zwei Spieler den selben Stein greifen, wird der Stein auseinander gezogen bis zu einer Maximalgrösse und spickt dann in seine Ursprungsform zurück. Danach ist der Stein wieder frei, muss also neu ergriffen werden.

Gelingt es, im eigenen Bereich Linien zu vollenden, gehen diese als unvollständige Straflinien auf die Gegner über, die diese wieder abbauen müssen. Ein Spieler hat verloren, wenn sich die Steine in seinem Feld bis zu einer gewissen Höhe türmen.

Als weiteres Element gibt es explodierende Steine, welche entsprechend gekennzeichnet sind. Diese können normal verbaut werden und explodieren erst, wenn sie Teil einer vollständigen Linie sind. Dabei nehmen sie noch ein paar umliegende Steine mit. Natürlich gibt es entsprechend mehr Strafsteine für die Gegner.

Um ein schnelles Ende zu ermöglichen ist ein Sudden Death Modus geplant. Jeder Spieler hat einen Knopf, mit dem er seine Bereitschaft für diesen Modus signalisieren kann. Sobald alle zugestimmt haben, wird er aktiviert und die erhaltenen Straflinien werden multipliziert, so dass das Spiel bald zu einem Ende kommt.

Die Spieleranzahl in einem Spiel soll dynamisch sein. Das heisst ein Spieler kann dazukommen und sein Feld aktivieren. Sobald dieses erschienen ist, bekommt er einen Teil der unvollständigen Linien der anderen Spieler, um ein faires Spiel zu garantieren. Umgekehrt läuft es, wenn jemand das Spiel verlässt.

6.3.2 Risiken und Prototypen

Diese Liste zeigt eine Übersicht über die Risiken und deren Prototypen, wie sie sich beim ersten erstellen des Spielkonzeptes boten:

Risiko	Beschreibung
Berührungserkennung	Erkennungsfähigkeit und Reaktion der Elemente
Spielergrid	Lage, Grösse, Form des Spielergrid
Tetris-Steine	Form und verhalten der Spielsteine
Beleuchtung	Diverse Lichtquellen und ihre Auswirkungen
klassische Tetrisversion	Einzelspielerversion mit Reihenbildung und -abbau

6.4 Berührungserkennung

6.4.1 Problem und Ansatz

Ein erstes Risiko, welches wir aufgrund von Erfahrungen mit anderen Multitouch-Applikationen sahen, ist die Erkennungs- und Reaktionsgeschwindigkeit der Spielelemente. Das Spiel setzt eine gewisse Mindestgeschwindigkeit voraus, damit Spielspass und Spannung aufkommen kann und die Spieler durch ungenaues ergreifen der Spielelemente nicht frustriert werden.

Um dieses Risiko auszuschalten haben wir einen einfachen Prototypen mit mehreren einfachen Elementen erstellt. Die Elemente sind in der Form Tetris-Steinen ähnlich und in der Gösse skalierbar, um das Verhalten zu analysieren. Mögliche Interaktionen sind die Elemente zu ziehen und zu rotieren.

6.4.2 Erfahrungen und Erkenntnisse

Bei Tests mit diesem Prototypen stellte sich schnell heraus, dass die Erkennungs- und Reaktionsfähigkeit des Multitouch-Tisches von to-fuse gut ist und für unsere Zwecke ausreicht. Die Frage zur Grösse der Spielelemente konnte in diesem Schritt noch nicht abschliessend beantwortet werden, da diese unmittelbar vom Spielfeld abhängt, in welchem sich die Elemente bewegen. Diese Frage wird mitunter im nächsten Abschnitt untersucht.

6.5 Spielergrid

6.5.1 Problemstellung

Die nächste wichtige Frage, die es zu klären galt, ist die Form und Verteilung der Spielbereiche. Aufgrund des Spielprinzips ist das Spielfeld ein Gitter mit Quadraten, welches den Bewegungsrahmen der Spielsteine definiert. Dieses Netz haben wir in einem Prototypen dargestellt, um Form, Verteilung und Anzahl der Spieler bestimmen zu können. Der Prototyp stellt auch die relevanten Parameter wie Position und Rotation dar, um eine Reproduktion zu erlauben. Gemäss vorhergehender Überlegungen, wie das Spiel werden soll, sind die Gitter zweidimensional auf einer Ebene angeordnet.

6.5.2 Erfahrungen und Erkenntnisse

Das grösste Problem, die Spielfelder zu verteilen, liegt darin, dass der Bildschirm nicht Quadratisch ist und in der Mitte ein genügend grosses gemeinsames Feld frei bleiben muss, auf dem die Spielelemente erscheinen und die Spieler gemeinsam Zugriff haben. Der erste Ansatz, der diesem Problem Rechnung trägt, ist die Gitter an der schmalen Seite nebeneinander anzuordnen. Allerdings wurde dies schnell verworfen, da sich die Spieler gegenseitig stören wenn sie so nahe beieinander stehen müssen.

Beim nächsten Versuch haben wir die vier Gitter je an einer Seite des Bildschirmes platziert. Um den Platz zu nutzen lag es nahe, die Spielfelder zu verziehen. Allerdings muss dann bei jeder Bewegung eines Steines seine Verformung berechnet werden, was sehr viel Aufwand bedeutet. Dies legte den Schluss nahe, die Applikation dreidimensional zu gestalten, so dass die Spielfelder als nichtverzerrte Ebenen dargestellt werden können. Ein weiterer Vorteil einer dreidimensionalen Spielwelt läge in den Effekten wie zum Beispiel Licht, welche sich einfach erzeugen liessen.

Wir haben diverse Anordnungen der Gitter ausprobiert. Allerdings blieb die Problematik der ungleichen Länge der Seiten des Bildschirms erhalten. Es war uns nicht möglich, die Felder so anzuordnen, dass alle Spieler das selbe Spielerlebnis geboten bekommen. Weitere Ideen, wie falt- oder scrollbare Gitter hatten den selben Mangel. Uns war es wichtig allen Spielern das selbe Erlebnis zu bieten und diesen Mangel konnten wir noch nicht beheben. Auch Ansätze die Spieler an den Ecken des Tisches zu platzieren waren nicht praktikabel, da entweder die Felder so zu klein geworden wären, so dass sich das Spielen schwierig gestaltet hätte oder bei schiefen Anordnungen zwar der Platz gut hätte genutzt werden können, aber Eindruck und Spielgefühl dann leiden aufgrund der ungewohnten Sicht auf das Spielfeld.

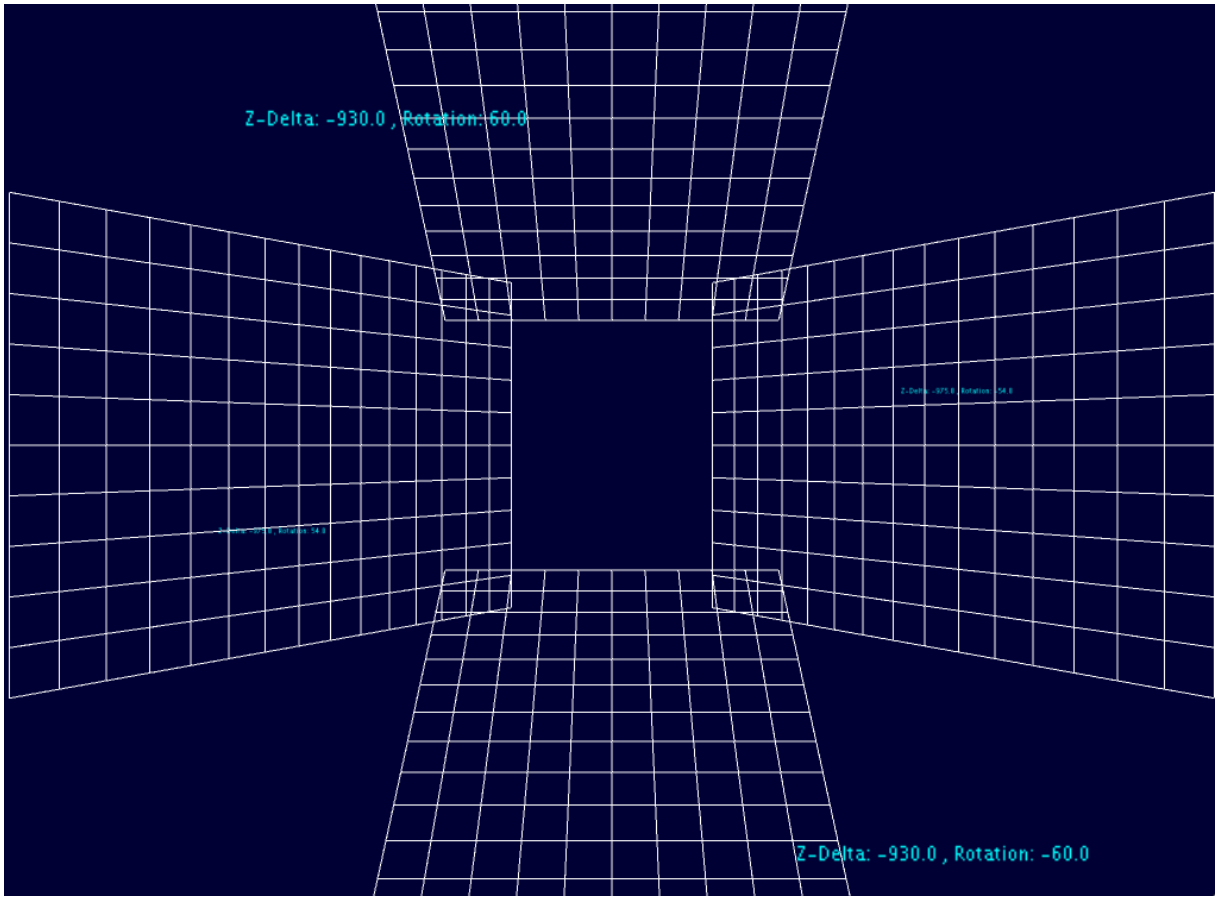


Abbildung 6.1: Prototyp mit 4 Grids in 3D

Dies führte uns zur Lösung mit nur zwei Spielern, so dass die Spielfelder symmetrisch angeordnet werden können. Zwar war auch diese Lösung nicht optimal, da so neben den Gittern einiges an nicht genutztem Raum blieb. Es waren aber bereits Ideen vorhanden wie dieser allenfalls noch genutzt werden könnte. Mit Hilfe des Prototypen haben wir daraufhin die Felder möglichst optimal aufgestellt und die Werte der Verschiebungen und Rotationen der Gitter gespeichert.

6.5.3 Schlussfolgerung

Die in diesem Entwicklungsschritt gemachten Erkenntnisse haben dazu geführt, dass wir von einer 2D auf eine 3D Applikation gewechselt haben, was zum einen Erleichterungen brachten, wie zum Beispiel, dass wir die Spielfelder nicht verziehen müssen, sondern die Felder quadratisch bleiben, andererseits aber auch Schwierigkeiten mit sich brachte, da drei Dimensionen die Berechnungen erschweren. Die Vorteile überwiegen aber klar. Weitere Konsequenz dieser Analysen ist die Verringerung der Spieleranzahl von vier auf zwei aus den oben beschriebenen Überlegungen.

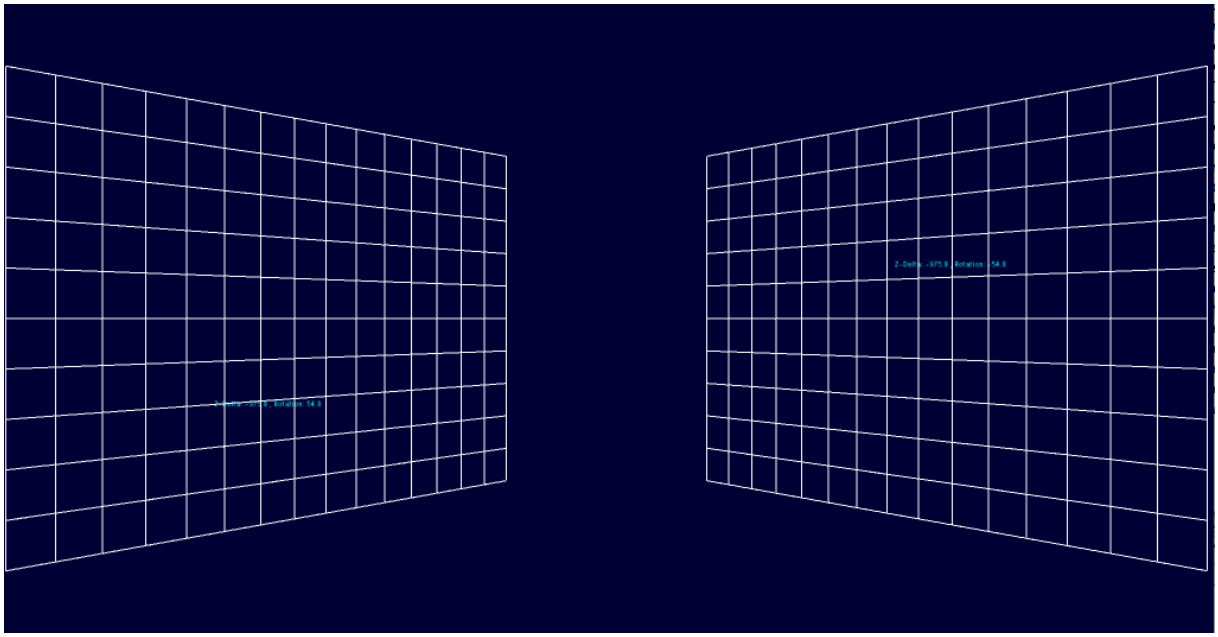


Abbildung 6.2: Endgültige Platzierung der beiden Spielergrids

6.6 Tetris-Steine und ihr Verhalten

6.6.1 Problemstellung

Parallel zu den Gittern haben wir die Spielsteine gestaltet. Dies beinhaltet sowohl ihre Form wie auch Texturen. Als weiterer Schritt folgte dann die Kombination obiger Elemente, so dass sich die Steine auf den Gittern bewegten. Bei diesem Schritt haben wir keine speziellen Probleme erwartet.

6.6.2 Erfahrungen und Erkenntnisse

Die Form der Steine ergab sich schnell aus den Vorgaben. Ein Tetris-Stein wird aus vier Basissteinen zusammengesetzt. Weil die Felder des Gitters quadratisch sind, muss auch der Grundriss des Basissteins quadratisch sein. Um dem Stein mehr Kontur zu geben, haben wir entschieden, die Oberseite kleiner zu machen, so dass die Seiten schräg zu liegen kommen. Dies führt in Verbindung mit der Beleuchtung der Szene zu interessanten Effekten. Des Weiteren haben wir eine Textur auf den Stein gelegt und die Steine eingefärbt, was ihnen Struktur verleiht. Je vier Basissteine werden zu den bekannten Tetris-Steinen kombiniert und erhalten unterschiedliche Farben, so dass sie einfach zu unterscheiden sind.

Die Steine mit den Spielfeldern zu kombinieren stellte keine Probleme dar. Das Framework ist so aufgebaut, dass die Elemente als Child an ein anderes Element angehängt werden können und so dessen Charakteristika übernehmen. In unserem Fall wird der Tetris-Stein ans Spielergrid angehängt und liegt somit schon richtig im Raum. Er bewegt sich dadurch relativ zum Spielfeld,

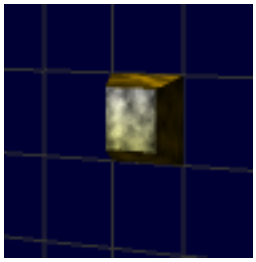


Abbildung 6.3: Ein Basestone, der Grundbaustein der Tetris Steine

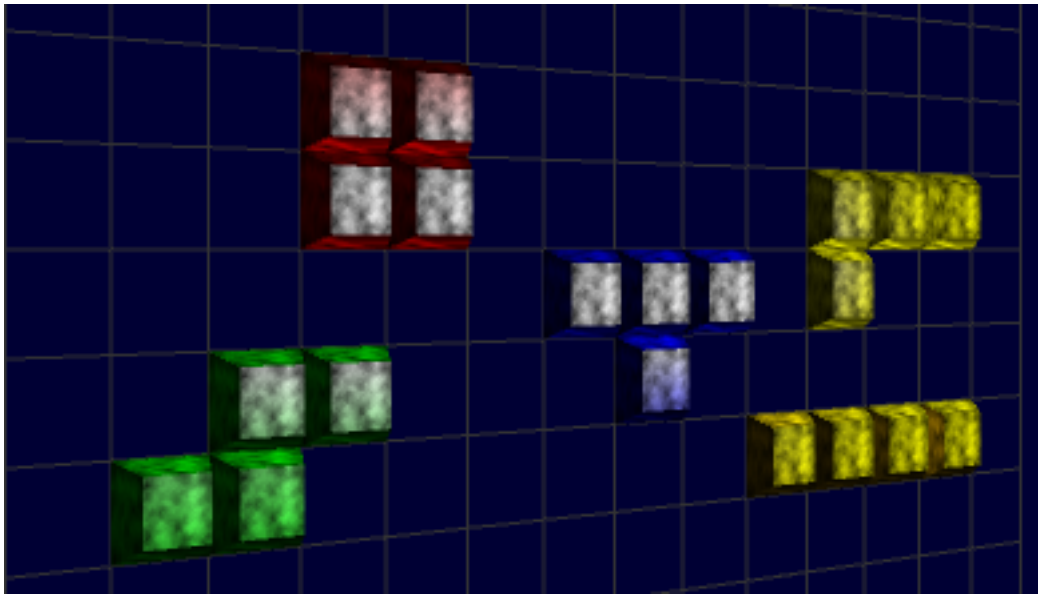


Abbildung 6.4: Alle verschiedenen Tetris-Steine

so dass die Bewegungen des Steines von uns nur noch in zwei Dimensionen kontrolliert werden müssen, was die komplexität erheblich erleichtert.

6.7 Beleuchtung der Szene

6.7.1 Problemstellung

Die richtige Beleuchtung einer Szenerie trägt viel zum Ambiente bei und bietet die Möglichkeit, Effekte ins Spiel einzubauen. Allerdings enthält OpenGL verschiedene Lichtquellen mit einer Unzahl an Parametern und Einstellungsmöglichkeiten. Um diese Möglichkeiten kennenzulernen, haben wir einen Prototypen erstellt, der die Einstellung des Lichtes mittels Tastatur erlaubt und die aktuellen Parameter anzeigt.

6.7.2 Erfahrungen und Erkenntnisse

Dieser Prototyp hat uns in einem ersten Schritt erlaubt, die Parameter und unterschiedlichen Lichtquellen etwas kennenzulernen. Interessanter wird aber in einem späteren Schritt die Möglichkeit sein, diese tastaturgesteuerte Anpassung der Beleuchtung bei Spielszenen zu benutzen, um die optimalen Einstellungen zu finden.

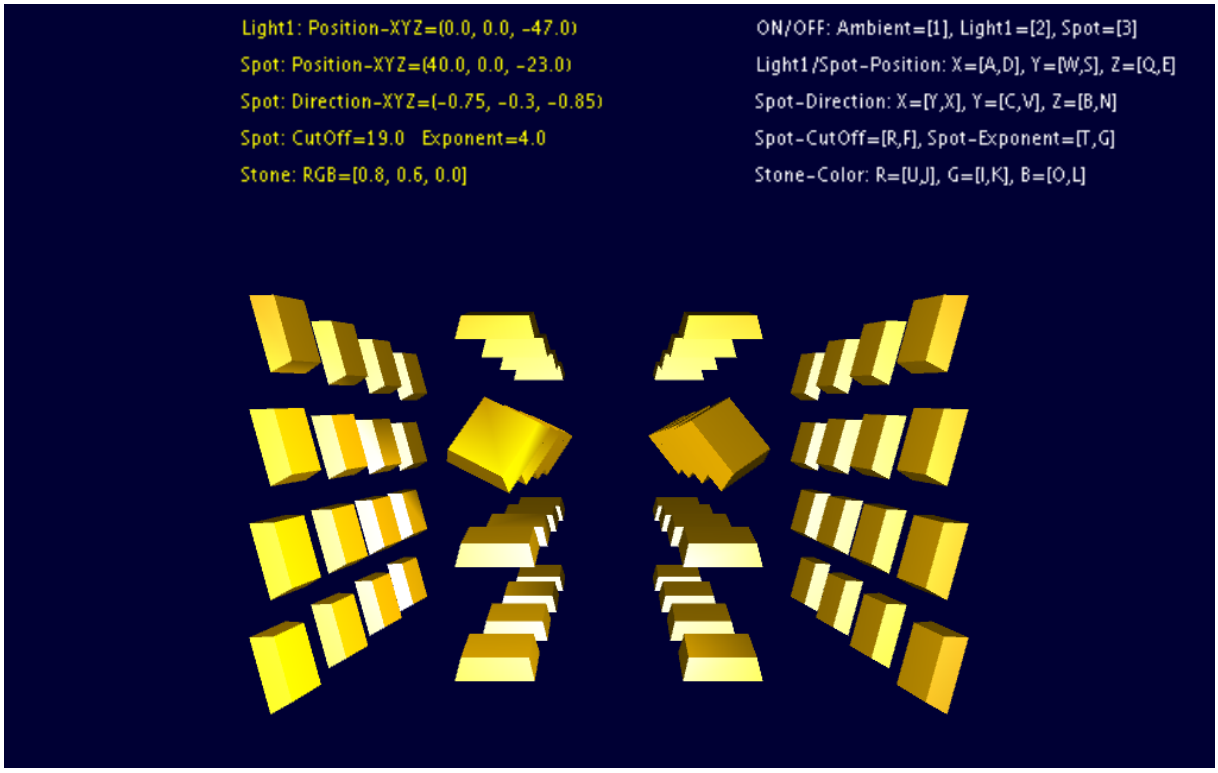


Abbildung 6.5: Prototyp für die Beleuchtung mit diversen zugeschalteten Lichtquellen

6.8 3D Erweiterung des to-fuse Applicaton Frameworks

6.8.1 Problemstellung

Beim Testen der Tetris-Steine auf dem Gitter sind zwei Probleme aufgetreten. Beide Probleme sind darauf zurückzuführen, dass dies die erste Applikation auf dem to-fuse Applicaton Framework in 3D ist und somit die entsprechenden Funktionen bisher nicht benötigt wurden.

Zum Einen bewegten sich die Steine in Y-Richtung in die zur Eingabe des Spielers entgegengesetzten Richtung. Dies lag daran, dass die von der Kamera gemessene Fingerposition falsch ins Koordinatensystem der Spielszene übersetzt wurde. Grund dafür war, dass eine 3D Applikation im Gegensatz zu einer 2D Applikation ein normales Koordinatensystem benutzt mit Ursprung in der Bildschirmmitte. Diese Umrechnung musste folglich angepasst werden.

Das zweite Problem hat ebenfalls seinen Ursprung im unterschiedlichen Verhalten zwischen 2D und 3D Szenerien. Durch die Kamera-Perspektive würde bei gleicher Berechnung wie in 2D der Finger dem Objekt davonfahren. Das Objekt legt zwar den gleichen Weg zurück wie der Finger, befindet sich aber durch den räumlichen Einfluss nicht immer unter ihm.

6.8.2 Lösungsansätze und Erkenntnisse

Das erste Problem versuchten wir mit einer einfachen Anpassung des Kamerainputs zu lösen. Allerdings führte dies zu einer falschen Umrechnung, so dass Objekte nicht mehr manipuliert werden konnten. Das Problem lag tiefer im Framework. Mit Hilfe von Emanuel Zgraggen von to-fuse konnte schliesslich eine Lösung gefunden werden. Die Umrechnung der Koordinaten war an mehreren Stellen nötig, damit sämtliche Eingaben richtig umgesetzt wurden.

Die Lösung des zweiten Problems hat etwas mehr Zeit in Anspruch genommen. Bei einer 2D Applikation reicht es, ein Objekt an die Position der Input Koordinaten zu verschieben. In 3D genügt dieser Ansatz wie oben beschrieben nicht. Die Lösung ist die Szene ohne Perspektive zu berechnen, eine Gerade durch den Berührungspunkt in die Tiefe zu ziehen und den Schnittpunkt mit dem Objekt (Spielfeld) zu ermitteln. Das Objekt kann dann am Schnittpunkt positioniert werden. Mit Hilfe dieser angepassten Berechnung bewegt sich das Objekt exakt mit dem Finger des Spielers mit und erlaubt eine angenehme und präzise Bedienung.

6.9 Klassische Tetris-Spielelemente

6.9.1 Problemstellung

Als letzten Schritt haben wir die obigen Elemente kombiniert. Ziel war es einen spielbaren Prototypen zu erhalten, bei dem sich die Steine ans untere Ende des Spielfeldes bewegen und Reihen gebildet werden können. Die abgebauten Reihen werden daraufhin dem Gegner mit zufälligen Lücken versehen als Strafe von unten ins Spielfeld geschoben.

6.9.2 Erfahrungen und Erkenntnisse

Die Animationen an sich konnten wir schnell einbauen und boten keine nennenswerte Probleme. Allerdings mussten wir feststellen, dass es spezielle Situationen gab, die ein Synchronisieren von mehreren, gleichzeitig laufenden Animationen erforderten. Als Beispiel ist hier das Nachrücken aller Steine oberhalb einer kompletten Linie zu nennen. Werden zwei Linien kurz nacheinander abgebaut, werden die Steine oberhalb nachgerückt. Es werden also zwei Animationen gestartet. Wenn diese Animationen aber nicht koordiniert werden stören sie sich gegenseitig. Beide verändern die Position sämtlicher Steine oberhalb und es kommt zu unerwünschten, visuellen Effekten wie z.B. einem ”Zurückspringen” der Steine.

Ein weiteres Problem, das uns beim Testen der Tetris-Steine auf dem Spielfeld auffiel ist das Verhalten, wenn ein vom Spieler geführter Stein auf ein Hindernis stösst wie z.B. ein anderer Stein oder den Rand des Spielfeldes. Ursprünglich war der Lösungsansatz den Stein an dieser Stelle zu blockieren. Allerdings stellt sich die Schwierigkeit, dass der Spieler seinen Finger weiterziehen kann, das Objekt aber zurückbleibt, was eine intuitive Bedienung äusserst schwierig gestaltet. Aus dieser Erkenntnis sind wir zu der Lösung übergegangen den Tetris-Stein während des Ziehens über die anderen zu heben. So können keine Kollisionen entstehen während ein Spieler den Stein führt. Geprüft wird erst wenn er den Stein wieder loslässt. Sollte sich der Stein zu diesem Zeitpunkt ausserhalb des Spielfeldes befinden, wird er zerstört. Weiter besteht die Möglichkeit den Stein innerhalb des Spielfeldes auf einem anderen Stein loszulassen. In diesem Fall werden sämtliche betroffenen Steine zerstört. Wie sich dieses Verhalten auf den Spielspass auswirkt ist Gegenstand des weiterer Tests im Rahmen des Spielbalancings.

6.10 Abschluss der Preproduction

Dank den bis hierhin gewonnenen Erfahrungen können wir nun festlegen, welche Funktionalität wir ins Spiel integrieren wollen. Der nächste Schritt ist das festlegen der Produktspezifikation, welche im Dokument *Anforderungsspezifikation* festgehalten wird.

7 Domainanalyse

7.1 Änderungshistorie

Rev.	Datum	Wer	Was wurde geändert
0	24.11.2009	mtr	Neues Dokument
1	30.11.2009	mtr	Domain Model inkl. Konzeptbeschreibung
2	01.12.2009	mtr	System Sequenz Diagramme, Contracts
2	01.12.2009	ral	Qualitätskontrolle
3	7.12.2009	mtr	Anpassungen gemäss Feedback Betreuer vom 5.12.06

7.2 Einführung

7.2.1 Zweck

Dieses Dokument beschreibt die Analyse der Domain für das Projekt T-Touch auf Basis der *Anforderungsspezifikation*.

7.2.2 Definitionen und Abkürzungen

Definitionen und Abkürzungen sind im *Glossar* zu finden.

7.2.3 Referenzen

Die Analyse wurde aufgrund der *Anforderungsspezifikation* erstellt.

7.3 Domain Model

7.3.1 Strukturdiagramm

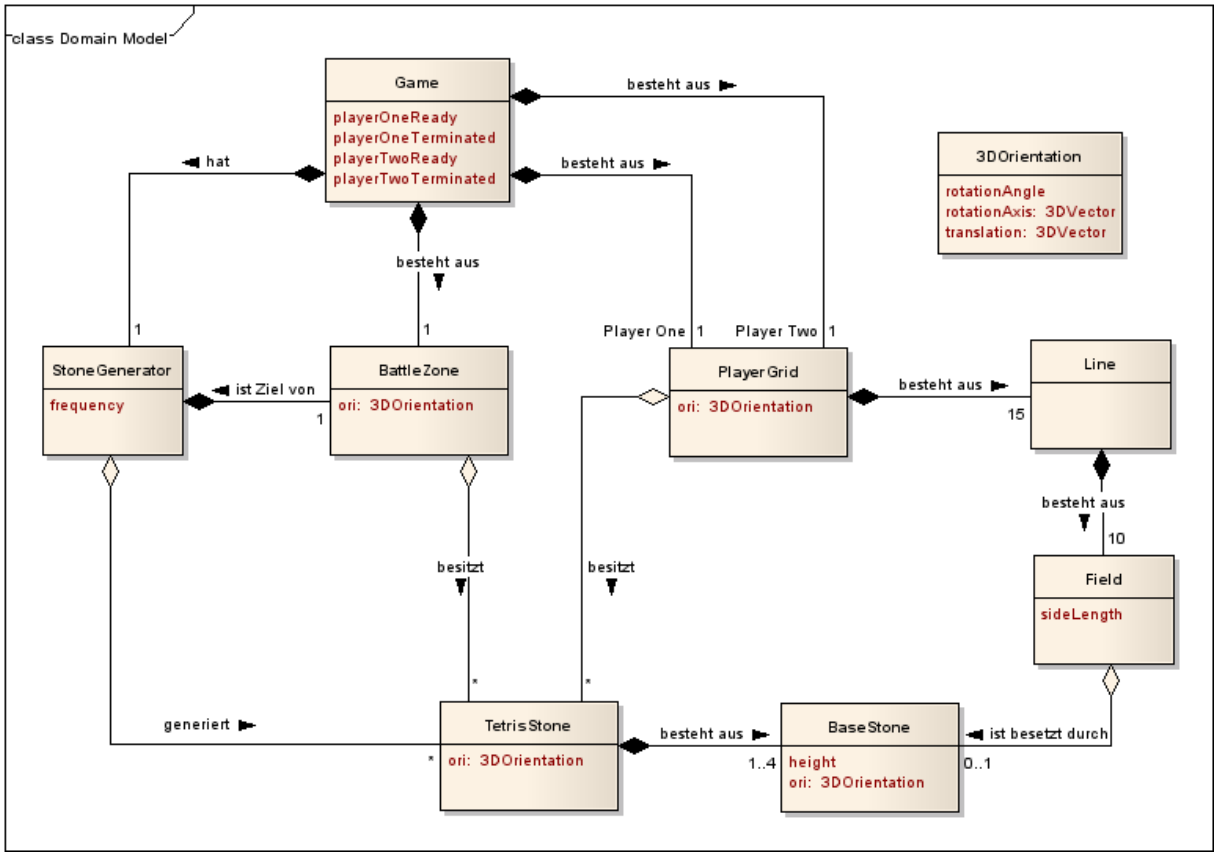


Abbildung 7.1: Domain Model

7.3.2 Konzeptbeschreibung

Game

Das laufende Spiel.

Attribut: playerOneReady
Definiert ob Spieler Eins für ein Spiel bereit ist oder nicht.

Attribut: playerOneTerminated
Definiert ob Spieler Eins das Spiel beenden möchte oder nicht.

Attribut: playerTwoReady
Definiert ob Spieler Zwei für ein Spiel bereit ist oder nicht.

Attribut: playerTwoTerminated

Definiert ob Spieler Zwei das Spiel beenden möchte oder nicht.

Beziehungen

Besteht aus einem *StoneGenerator*, einer *BattleZone* und je Spieler einem *PlayerGrid*.

StoneGenerator

Der StoneGenerator generiert neue *TetrisStone*'s und fügt diese in die *BattleZone* ein.

Attribut: frequency

Anzahl zu generierende Steine pro Zeiteinheit.

Beziehungen

Zu *TetrisStone* um diese generieren zu können und zur *BattleZone* als Ziel-Container der generierten *TetrisStone*'s.

BattleZone

Repräsentiert den Mittelbereich zwischen den Spielfeldern. Hier werden neue *TetrisStone*'s ins Spiel gebracht und Spieler können innerhalb dieser Zone um diese kämpfen. Verlässt ein Stein die BattleZone wird die Zuständigkeit für diesen Stein auf das entsprechende *PlayerGrid* verschoben.

Attribut: ori

Ausrichtung des Objektes im 3D-Raum.

Beziehungen

Kennt seine *TetrisStone*'s.

PlayerGrid

Das Spielfeld eines Spielers. Jeder Spieler hat ein PlayerGrid. In diesem kann er *TetrisStone*'s platzieren.

Attribut: ori

Ausrichtung des Objektes im 3D-Raum.

Beziehungen

Das PlayerGrid besteht aus 15 *Line*'s und keinen oder mehreren *TetrisStone*'s.

TetrisStone

Ein "klassischer" Tetris-Spielstein bestehend aus *BaseStone*'s.

Attribut: ori

Ausrichtung des Objektes im 3D-Raum.

Beziehungen

Ein Spielstein besteht aus mind. einem bis max. 4 *BaseStone*'s.

BaseStone

Ein einzelner, quadratischer Basis-Stein. Aus ihm werden *TetrisStone*'s gebildet. Er passt bezüglich Grösse in genau in ein *Field*.

Attribut: height

Die Höhe des Steines.

Attribut: ori

Ausrichtung des Objektes im 3D-Raum.

Line

Repräsentiert eine Linie des *PlayerGrid*'s.

Beziehungen

Eine Linie besteht aus 10 *Field*'s.

3DOrientation

Definiert durch seine Attribute die Orientierung eines Objektes im 3D-Raum.

Attribut: rotationAngle

Definiert den Winkel um den gedreht werden soll.

Attribut: rotationAxis

Definiert über einen 3DVector {x,y,z} um welche Achsen rotiert werden soll.

Attribut: translation

Die Verschiebung definiert durch einen 3DVector {x,y,z}.

Field

Repräsentiert ein quadratisches Feld (Zelle) auf dem *PlayerGrid*. Ein Feld kann optional jeweils von max. einem *BaseStone* besetzt sein.

Attribut: sideLength

Die Seitenlänge des Feldes.

Beziehungen

Wenn dieses Feld besetzt ist kennt es den entsprechenden *BaseStone*, sonst nicht.

7.4 System Sequenz Diagramme

7.4.1 UC1: T-Touch spielen

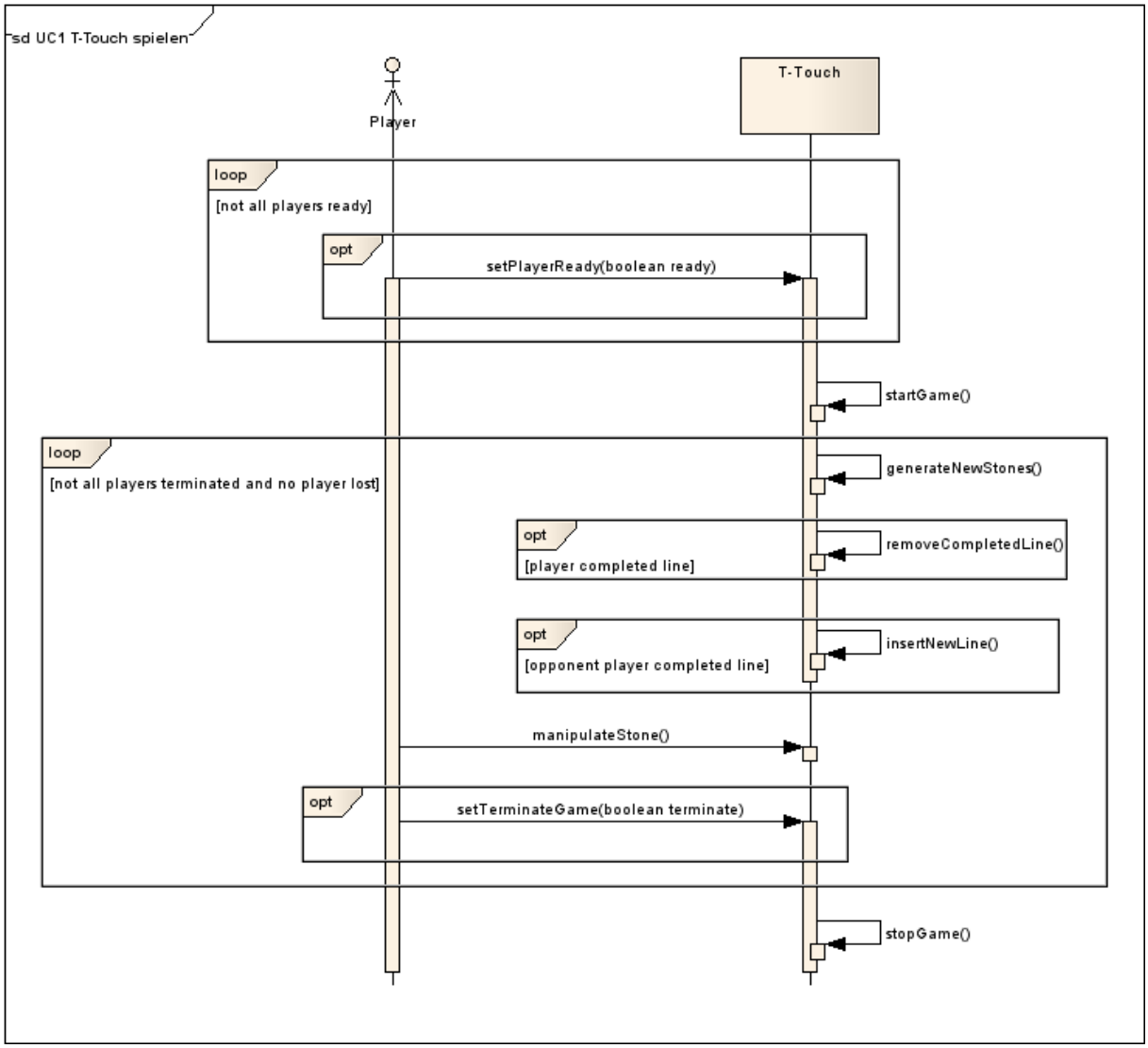


Abbildung 7.2: UC1 T-Touch spielen

7.5 Systemoperationen

7.5.1 Contracts UC1: T-Touch spielen

Im Folgenden sind die Contracts zu *UC1: T-Touch spielen* definiert.

CO1: setPlayerReady

Operation:	setPlayerReady(boolean ready)
Vorbedingung:	keine
Nachbedingung:	- Richtiges Attribut der *Game*-Klasse (playerOneReady oder playerTwoReady, entsprechend Actor) auf Wert von Parameter ready gesetzt.

CO2: startGame

Operation:	startGame()
Vorbedingung:	Attribute playerOneReady und playerTwoReady der *Game*-Klasse müssen true sein.
Nachbedingung:	- Eine *BattleZone*-Instanz bzi wurde erstellt. - bzi wurde an die *Game*-Klasse angeknüpft. - Eine *StoneGenerator*-Instanz sgi wurde erstellt. - sgi wurde mit bzi verknüpft. - sgi wurde an die *Game*-Klasse angeknüpft. - Zwei *PlayerGrid*-Instanzen pgi1 und pgi2 wurden erstellt. - pgi1 und pgi2 wurden an die *Game*-Klasse angeknüpft.

CO3: generateNewStones

Operation:	generateNewStones()
Vorbedingung:	Das Spiel wurde gestartet. (siehe *Kapitel 7.5.1: "CO2: startGame"*)
Nachbedingung:	- Es wurden keine oder mehrere (gemäss frequency *TetrisStone*-Instanzen tsi erstellt. - Die Instanzen tsi wurden an die *BattleZone* angeknüpft.

CO4: removeCompletedLine

Operation:	removeCompletedLine()
Vorbedingung:	Steine wurden ins *PlayerGrid* gesetzt. (siehe *Kapitel 7.5.1: "CO6: manipulateStone"*)
Nachbedingung:	- Die Verknüpfung zwischen den *Field*'s der completed *Line* und deren *BaseStone*'s wurde entfernt. - Sämtliche *BaseStone*'s die mit einem Field oberhalb der completed *Line* verknüpft sind, wurden nachgerückt. Das heisst die Verknüpfung mit ihrem aktuellen *Field* wurde gelöscht und sie wurden mit dem gleichwertigen *Field* der *Line* eins tiefer verknüpft.

CO5: insertNewLine

Operation:	insertNewLine()
Vorbedingung:	Eine komplette *Line* wurde abgebaut. (siehe *Kapitel 7.5.1: "CO4: removeCompletedLine"*)
Nachbedingung:	- Die Verknüpfung jedes BaseStone's mit dem *Field* das er besetzt wurde gelöscht. - Jeder *BaseStone* wurde an das gleichwertige *Field* der darüberliegenden *Line* verknüpft. - Es wurden 9 *BaseStone*-Instanzen bsi erstellt. - Die bsi's wurden mit den *Field*'s der untersten *Line* verknüpft.

CO6: manipulateStone

Operation:	manipulateStone()
Vorbedingung:	Ein freier, noch nicht festgesetzter *TetrisStone* befindet sich im Spiel. (siehe *Kapitel 7.5.1: "CO3: generateNewStones"*)
Nachbedingung:	- Die *3DOrientation* des betroffenen *TetrisStone* wurde angepasst. - Der *TetrisStone* wurde inkl. allfälligen Verknüpfungen zu *PlayerGrid* oder *BattleZone* zerstört falls: - a) Der *TetrisStone* über einem festgesetzten *TetrisStone* losgelassen wurde. - b) Der *TetrisStone* nicht auf einem *PlayerGrid* oder der *BattleZone* losgelassen wurde.

CO7: setTerminateGame

Operation:	setTerminateGame(boolean terminate)
Vorbedingung:	Das Spiel wurde gestartet. (siehe *Kapitel 7.5.1: "CO2: startGame"*)
Nachbedingung:	- Richtiges Attribut der *Game*-Klasse (playerOneTerminated oder playerTwoTerminated, entsprechend Actor) auf Wert von Parameter terminate gesetzt.

CO8: stopGame

Operation:	stopGame()
Vorbedingung:	Attribute playerOneTerminated und playerTwoTerminated der *Game*-Klasse müssen true sein oder ein Spieler hat das Spiel verloren.
Nachbedingung:	- Die Attribute der *Game*-Klasse playerOneReady, playerTwoReady, playerOneTerminated und playerTwoTerminated wurden auf false gesetzt. - Sämtliche in *CO2: startGame* erstellten Instanzen und Verknüpfungen wurden gelöscht.

8 Externes Design

8.1 Änderungshistorie

Rev.	Datum	Wer	Was wurde geändert
0	12.12.2009	mtre	Neues Dokument
1	13.12.2009	mtre	Erste vollständige Version

8.2 Einführung

8.2.1 Zweck

Dieses Dokument beschreibt das externe Design für das Projekt T-Touch.

8.2.2 Definitionen und Abkürzungen

Definitionen und Abkürzungen sind im Glossar am Ende dieses Dokuments zu finden.

8.2.3 Referenzen

Die Entwicklung der Benutzeroberfläche ist im *Entwicklungsprotokoll* beschrieben.

8.2.4 Übersicht

Dieses Dokument gibt einen Überblick über die visuellen und interaktiven Elemente sowie die Benutzerführung von T-Touch.

8.3 GUI Navigation Map

Das folgende Diagramm beschreibt die Benutzerführung des Spiels. Die Anwendung befindet sich zunächst im StartScreen. Durch eine Berührung der Oberfläche wechselt die Applikation zum GetReadyScreen. Nachdem beide Spieler ihre Bereitschaft signalisiert haben geht es weiter zum GameScreen und das Spiel beginnt. Hier können beide Spieler das laufende Spiel abbrechen oder sie spielen bzw. manipulieren Tetris-Steine bis einer von Ihnen gewinnt. In beiden Fällen findet ein Übergang zum StartScreen und somit zum Urzustand statt. Dieser Zyklus kann beliebig oft wiederholt werden.

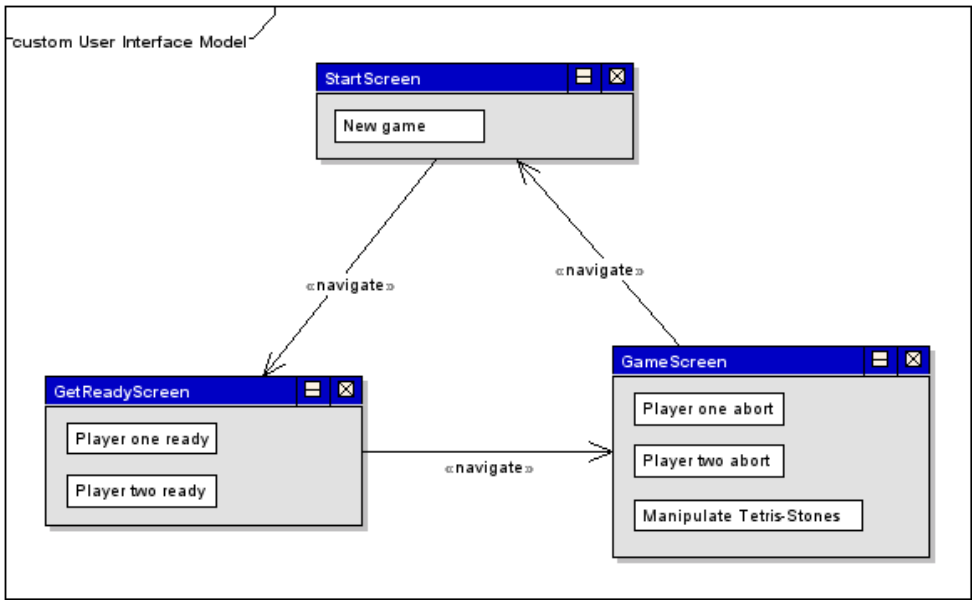


Abbildung 8.1: GUI Navigation Map

8.4 Prototyping

Zur Entwicklung der grafischen Elemente wurden verschiedene Prototypen gebaut. Diese sind im *Entwicklungsprotokoll* beschrieben.

8.5 Screens

8.5.1 StartScreen

Die Anwendung startet in diesem Zustand. Der Spieler kann einen beliebigen Punkt berühren um den Übergang zum GetReadyScreen auszulösen.

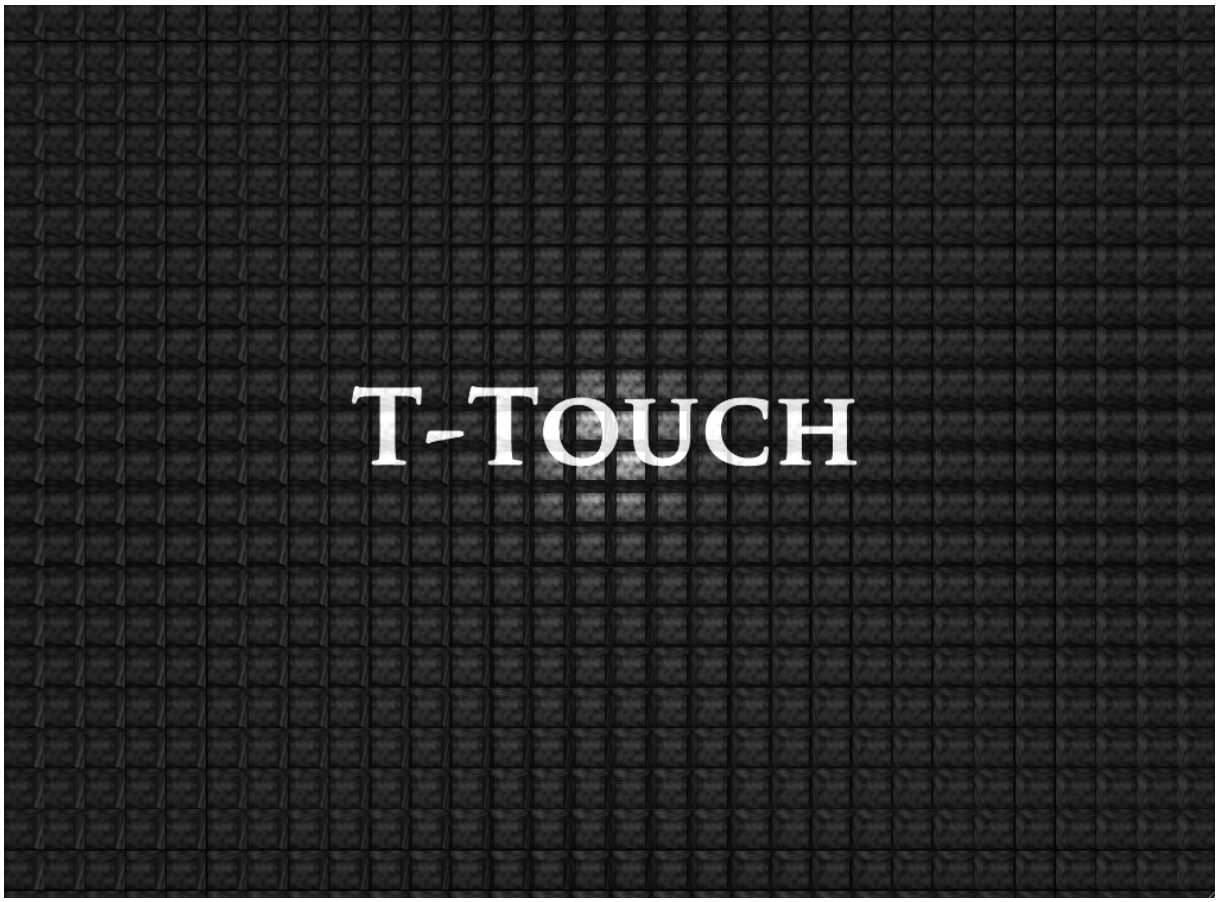


Abbildung 8.2: StartScreen

8.5.2 GetReadyScreen

Auf den folgenden zwei Abbildungen ist ersichtlich wie die Spieler ihre Bereitschaft signalisieren. Sie berühren dazu den vor ihnen liegenden Tetris-Stein. Ist ein Spieler bereit, leuchtet sein Stein weiss und wird vom jeweiligen Scheinwerfer statisch beleuchtet. Sind beide Spieler bereit startet das Spiel und die Anwendung wechselt in den GameScreen.

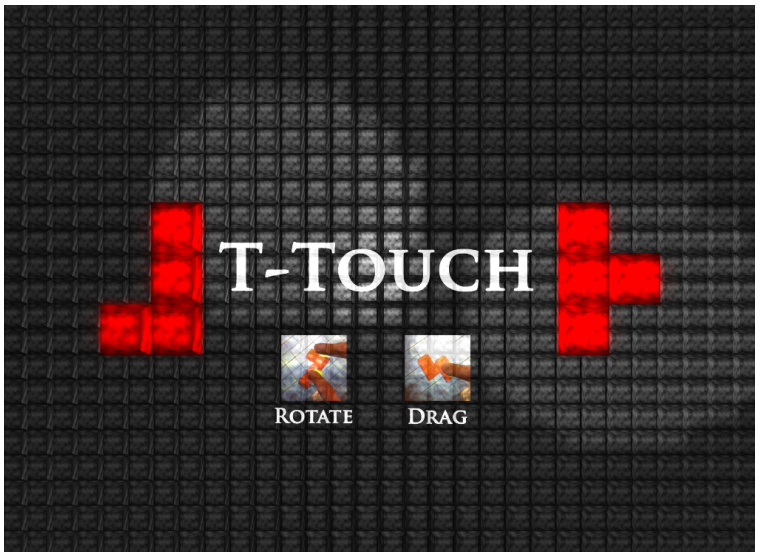


Abbildung 8.3: GetReadyScreen

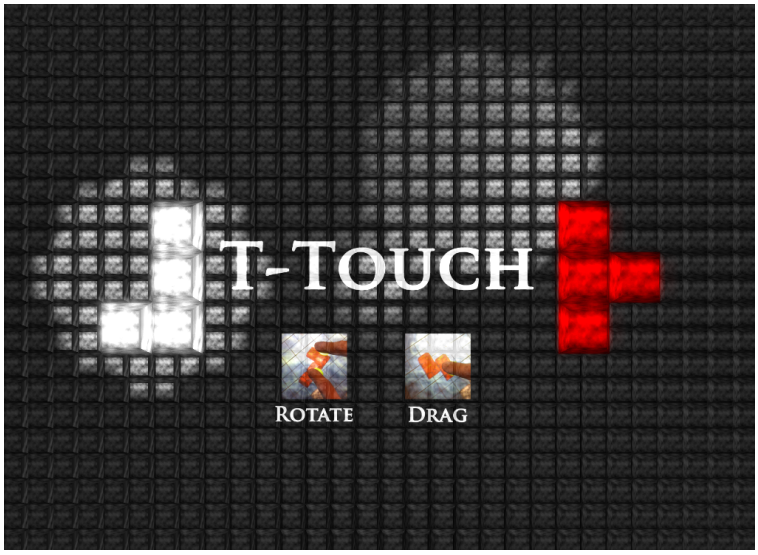


Abbildung 8.4: GetReadyScreen, Spieler links ist bereit

8.5.3 GameScreen

Hier findet das eigentliche Spiel statt. Die Spieler duellieren sich im Multiplayer-Tetris. Gewinnt ein Spieler oder brechen beide Spieler das laufende Spiel per Abort-Button ab, wechselt die Anwendung wieder in den StartScreen.

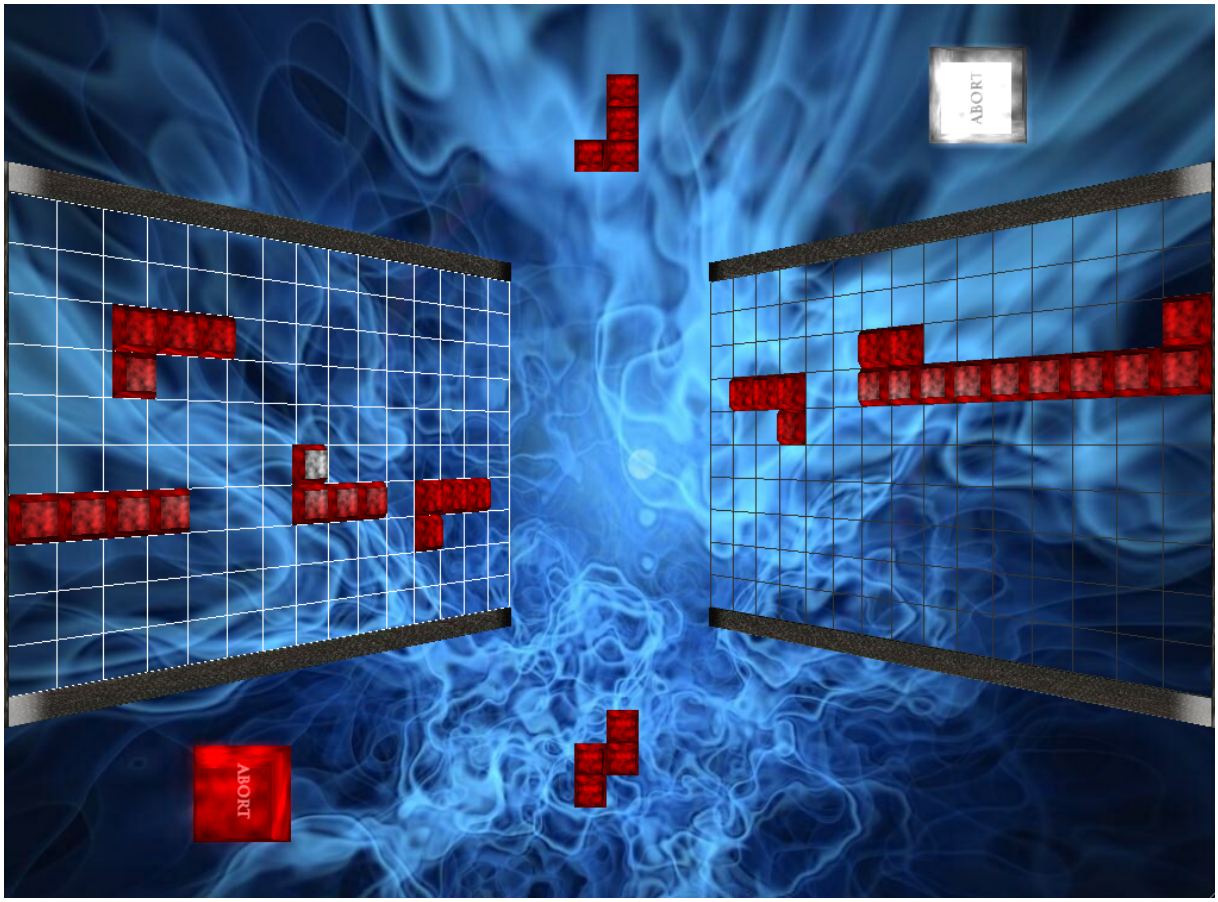


Abbildung 8.5: GameScreen

8.6 Übergänge zwischen Screens

8.6.1 StartScreen zu GetReadyScreen

Zwei zufällige Tetris-Steine fallen aus Sicht der Kamera auf den Hintergrund. Die zwei Scheinwerfer werden eingeschaltet und beginnen sich zufällig zu bewegen.

8.6.2 GetReadyScreen zu GameScreen

Die Scheinwerfer stellen ab und die Steine im Hintergrund sowie die zwei Tetris-Steine fallen in den Raum. Das Spielfeld erscheint.

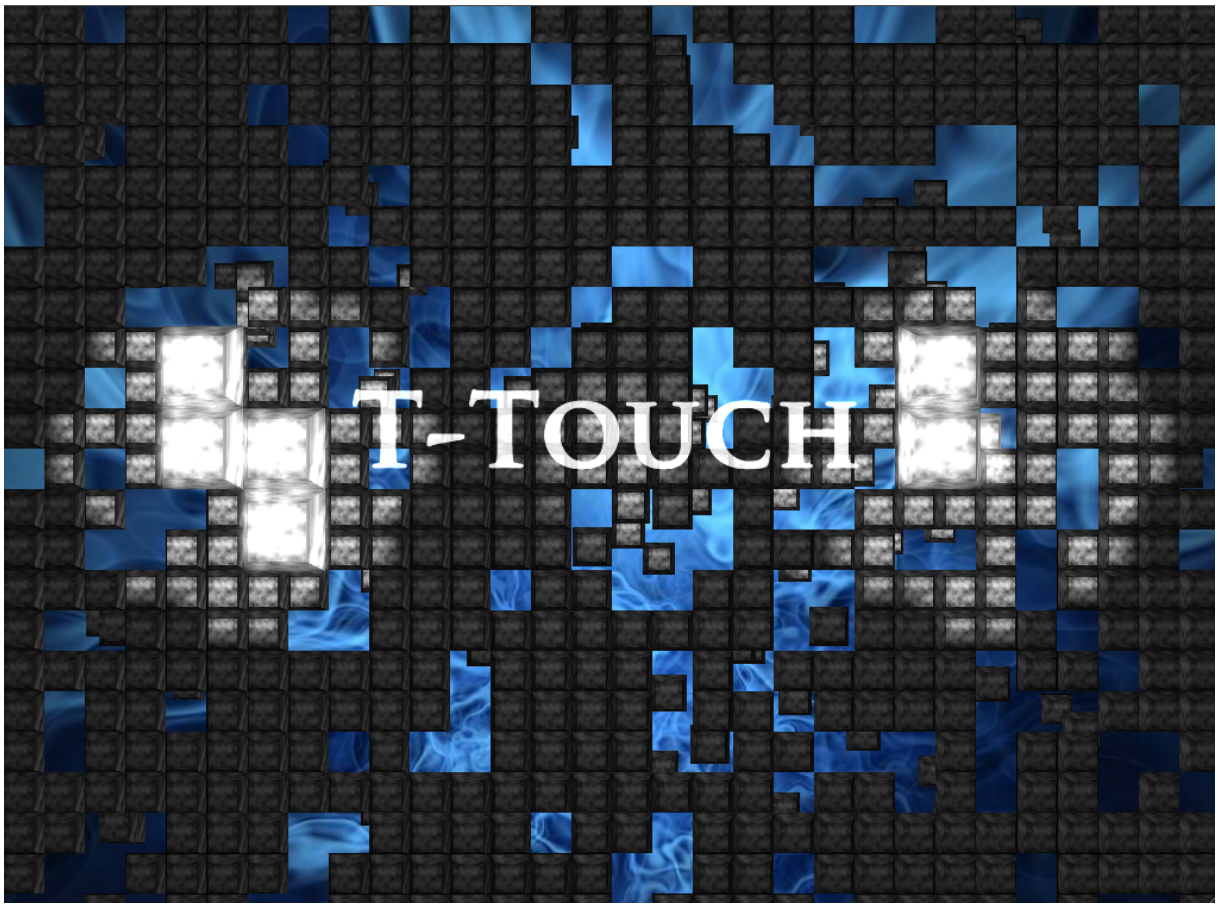


Abbildung 8.6: Übergang GetReadyScreen zu GameScreen

8.6.3 GameScreen zu StartScreen

Auf jeder Spielerseite erscheint die Nachricht warum das Spiel beendet ist. Danach wartet die Anwendung einige Sekunden und wechselt anschliessend direkt in den StartScreen. Mögliche Nachrichten sind “Game canceled”, “You are victorious!” oder “’You loose.. revenge!’

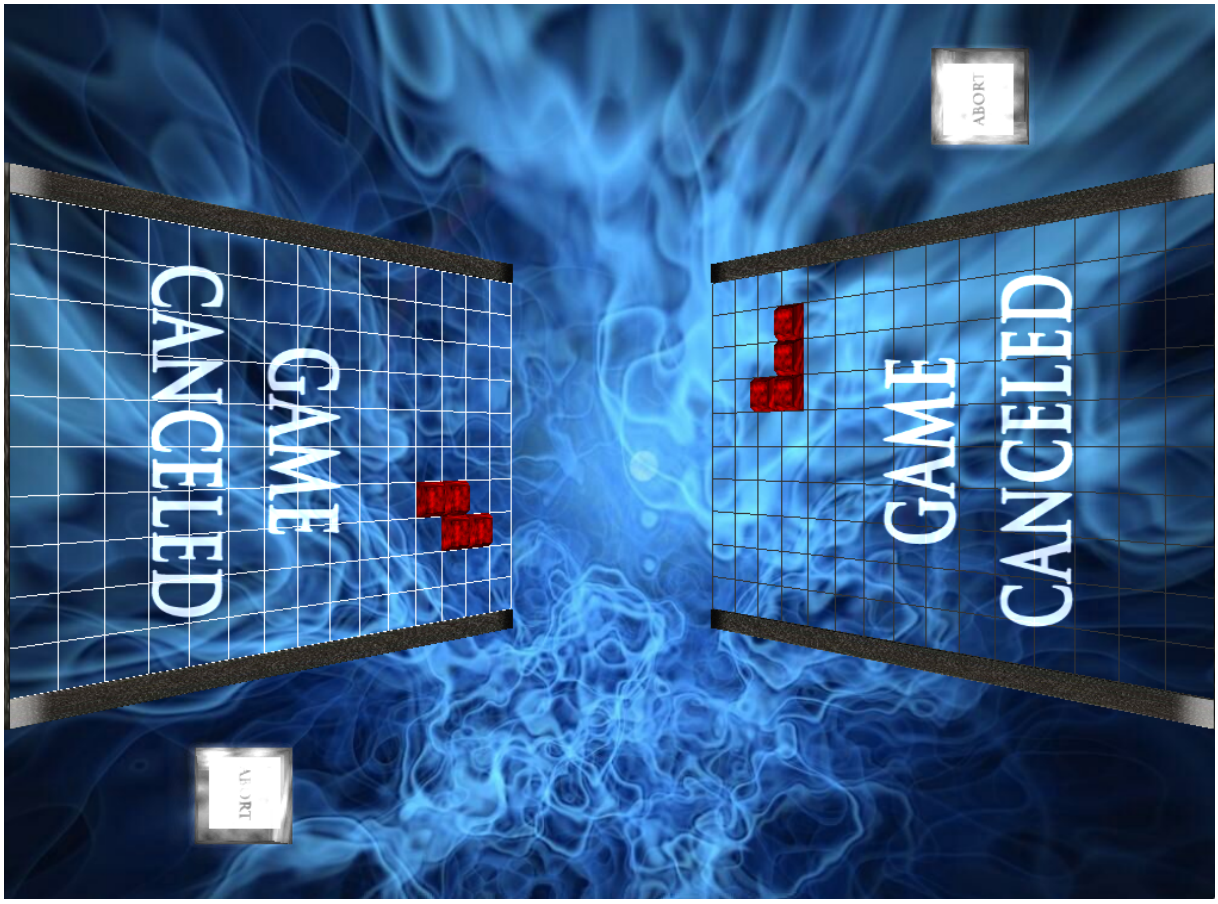


Abbildung 8.7: Übergang GameScreen zu StartScreen bei Abbruch durch Spieler

8.7 Tetris-Steine

Tetris-Steine bestehen aus einem bis vier quadratischen Basis-Steinen mit schrägen Seitenwänden. Die Anordnungen dieser Basis-Steine entsprechen den klassischen Tetris-Steinen.

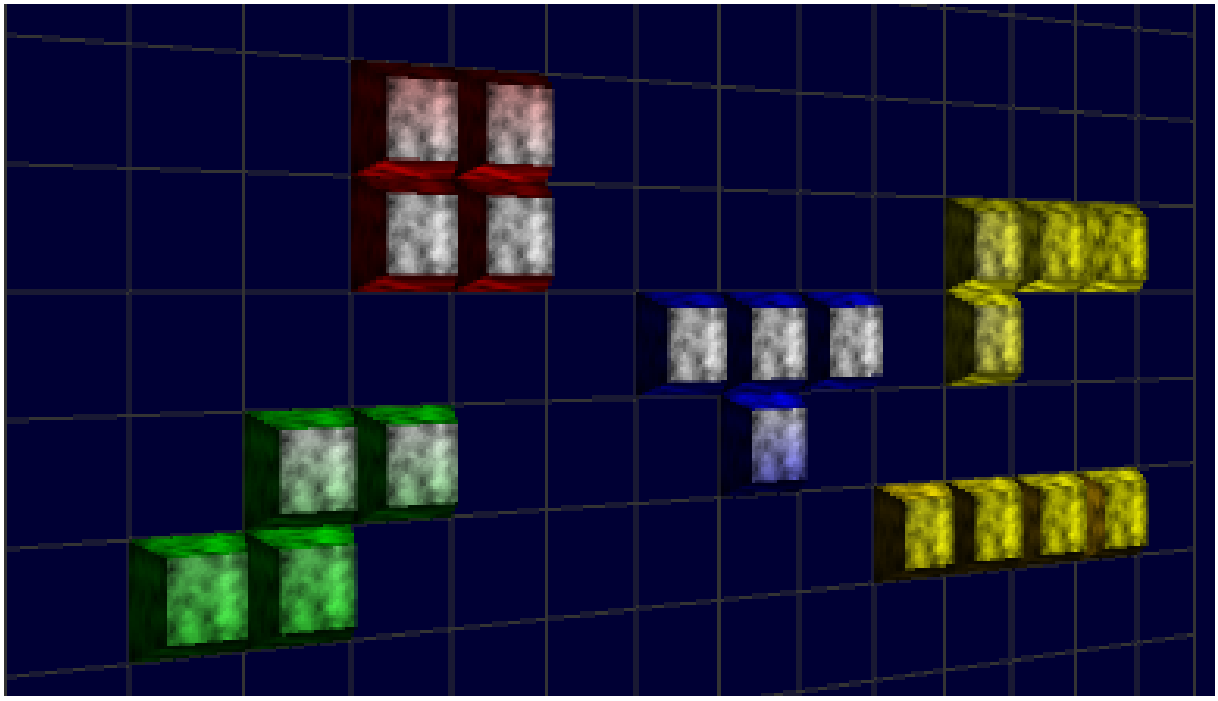


Abbildung 8.8: Verschiedene Tetris-Steine

9 System Architektur Dokument

9.1 Änderungshistorie

Rev.	Datum	Wer	Was wurde geändert
0	12.12.2009	ral	Neues Dokument
1	14.12.2009	ral	Umgebung und Entscheide
2	15.12.2009	ral	Erste vollständige Version
3	16.12.2009	mtr	Qualitätskontrolle

9.2 Einführung

9.2.1 Zweck

Dieses Dokument beschreibt die Software Architektur für das Projekt T-Touch.

9.2.2 Definitionen und Abkürzungen

Definitionen und Abkürzungen sind im Glossar zu finden.

9.2.3 Übersicht

In diesem Dokument werden die verschiedenen Aspekte der Architektur und grundlegende Konzepte, welche in diesem Projekt verwendet wurden, beschrieben und erläutert.

9.3 Umgebung

9.3.1 to-fuse Multi-Touch Platform

Für das Projekt stand dem Team die von der Firma to-fuse entwickelte *to-fuse Multi-Touch Platform* zur Verfügung. Die Oberfläche des Tisches wird von allen Seiten mit Infrarot-Licht abgedeckt und Reflexionen, wie sie durch Fingerberührung entstehen, durch eine Infrarot-Kamera und entsprechende Software erkannt. Die Plattform macht in der Anzahl gleichzeitiger Berührungspunkte keine Einschränkungen im Gegensatz zu anderen Modellen wie z.B. dem HP TouchSmart600 PC, welcher lediglich 2 Touch-Punkte erkennt.

Das to-fuse Application Framework stellt die Schnittstelle zwischen der Hardware und unserer Applikation dar. Es liefert den von der Kamera erkannten Input der Benutzer und stellt die Grundstruktur unserer Architektur. Das Application Framework ist so aufgebaut, dass die

Abbildung 9.1: to-fuse Multi-Touch Platform

Objekte, welche auf dem Tisch dargestellt werden, von der Klasse SceneGraphObject erben. So wird die benötigte Funktionalität zum Darstellen und Interagieren bereitgestellt. Alle diese Objekte werden in einer Baumstruktur verwaltet, damit die Update- und Zeichenroutine auch wirklich durch alle Objekte geführt wird.

Das Framework war bisher allerdings nur für 2D Applikationen verwendet worden und die Funktionalität für 3D war nicht vollumfänglich implementiert. Nur die Grundfunktionen zur Darstellung von 3D Szenen standen uns zur Verfügung. Dies führte zu den im nächsten Abschnitt beschriebenen Erweiterungen.

9.3.2 Erweiterungen am to-fuse Application Framework

Generelle Erweiterungen

Die folgenden Funktionen stellen kleine Erweiterungen des Frameworks dar, welche den Umgang mit den SceneGraphObjects, also den zu zeichnenden und der Interaktion dienenden Objekten, betreffen.

afterChildShapesDraw

Diese Funktion wird jeweils nach der draw() Methode, welche die Objekte zeichnet, ausgeführt. Sie kann dazu verwendet werden, Effekte zu zeichnen, die auf das bereits gezeichnete Objekt wirken. In unserem Fall haben wir damit den Blur-Effekt auf die Tetris-Steine gezeichnet.

changeParentTo

Die SceneGraphObjects werden in einer Baumstruktur verwaltet, welche als eine verkettete Liste mit Referenzen auf das nächste Objekt aufgebaut ist. Beim Wechsel des Parents auf einen Neuen muss deshalb die hinterlassene Lücke auch wieder korrekt gschlossen werden.

removeFromParent

Will man nur ein Kindobjekt aus dem Baum entfernen, hat man dasselbe Problem wie beim oben beschriebenen Wechsel des Parents. Diese Funktion nutzt entsprechend auch indirekt die changeParentTo Methode, um das Kindobjekt aus der Kette zu entfernen.

hasActiveStrokes

Diese Methode überprüft, ob aktive Strokes (getrackter Touch-Punkt) auf dem Objekt vorhanden sind. So kann ermittelt werden, ob ein Spieler gerade mit dem Objekt interagiert.

releaseStroke

Mit dieser Funktion kann die Interaktion eines Benutzers mit dem Objekt unterbrochen werden. Das heisst er muss danach das Objekt auf dem Bildschirm neu aufgreifen.

Erweiterungen für 3D

Wie oben bereits angedeutet war die Funktionalität für eine 3D Applikation nicht vollumfänglich im Framework enthalten. Eine erste Änderung benötigte das Koordinatensystem der Input-Koordinaten. Für 2D ist der Ursprung des Koordinatensystems in der oberen linken Ecke, bei 3D hingegen im Zentrum des Bildschirms. Neben dem anders platzierten Ursprung ist die Y-Achse bei 2D Anwendungen invertiert. Das Resultat daraus war, dass sich Objekte in entgegengesetzter Y-Richtung zu der vom Benutzer gewollten Richtung bewegten. Bei der Anpassung der vom Framework erhaltenen Koordinaten eines Berührungspunktes hat uns to-fuse geholfen, da an diversen Stellen im Framework tiefgreifende Änderungen nötig waren. Abgesehen davon erweiterten wir das InteractiveSceneGraphObject, welches Basis-Verhalten wie Rotation, Translation und Skalierung bereits für 2D kennt, um zusätzliche Funktionalität sodass sich dieses auch in 3D korrekt verhält.

initializeState

In dieser Methode wird festgestellt, ob das Objekt durch einen oder zwei Berührungspunkte vom Benutzer bedient wird. Entsprechend werden die Abstände von der Objektposition zu den Eingabepunkten berechnet. Diese werden benötigt, damit das Objekt nicht unter der Berührung des Benutzers springt. Bei der Berechnung des Abstandes ist die korrekte Input Koordinate entscheidend und wird durch unten beschriebene Funktionen für 3D nun anders berechnet.

update

Diese Funktion berechnet die Translation und Transformation eines Objektes aufgrund von Benutzereingaben. An dieser Stelle brauchen wir die exakte Position auf dem Objekt ohne perspektivische Verzerrung.

calculateIntersectionPoint

Die zweidimensionalen Input Koordinaten rechnen wir mit folgendem Algorithmus in die dreidimensionalen Koordinaten der Spielwelt um:

- Die Input Koordinaten werden so umgerechnet, dass der Ursprung in der unteren linken Ecke ist

- Berechne den ersten Punkt der Geraden mit gluUnProject und z=0
- Berechne den zweiten Punkt mit gluUnProject und z=1
- Definiere eine Ebene im Raum durch drei Punkte
- Berechne den Schnittpunkt zwischen Ebene und Gerade mit Hilfe der Funktion pointPlaneIntersectionPoint

pointPlaneIntersectionPoint

Diese Hilfsfunktion berechnet den Schnittpunkt einer Geraden mit einer Ebene. Dabei wenden wir folgende Formeln für die Berechnung an:

$$P_{hit} = A + c * t_{hit} \tag{9.1}$$

$$t_{hit} = \frac{n * (B - A)}{n * c} \tag{9.2}$$

A ist ein Punkt auf der Geraden, B ein Punkt auf der Ebene, n die Normale der Ebene und c die Normale der Geraden.

addRotation

In 3D gibt es nicht wie in 2D nur eine Drehachse, sondern beliebig viele. Allerdings ist es schwierig direkt die resultierende Achse und Winkel einzugeben. Nun ist es möglich eine weitere Rotation hinzuzufügen und diese Methode berechnet dann die resultierende Rotation. Für die Berechnung nutzen wir Quaternionen, welche ein Tupel aus vier Zahlen sind und ähnlichen Rechengesetzen unterliegen wie Vektoren. So lassen sich elegant Rotationen addieren ohne komplizierte Matrizenmultiplikationen zu benötigen.

9.4 Architektonische Entscheidungen

9.4.1 3D-Bewegung

Wie werden die Positionen der Berührungspunkte, welche in zweidimensionalen Koordinaten vorliegen, in Koordinaten im dreidimensionalen Raum umgerechnet?

Faktoren

- Direkte Umsetzung des 2D Punktes kann perspektivische Verschiebung ergeben.
- Die Umrechnung soll möglichst einfach sein.
- Es muss berücksichtigt werden, dass Objekte windschief im Raum liegen können.

Lösung

Wir berechnen den Inputvektor mit Hilfe verschiedener OpenGL Funktionen. Dabei ist wichtig, dass die Berechnung ohne Transformation durch die perspektivische Projektion durchgeführt. Damit können wir den Schnittpunkt mit dem Objekt berechnen und erhalten so den genauen Punkt im dreidimensionalen Raum.

Ungelöste Probleme

keine

Erwogene Alternativen

Es wurde überlegt, mittels zurückgelegtem Weg der Berührungspunkte den Korrekturfaktor für die Verschiebung des Objekts zu errechnen. Allerdings würde das einen grossen Mehraufwand bedeuten, da die perspektivischen Korrekturen bei der von uns bevorzugten Lösung durch OpenGL Funktionen ausgeführt werden. Der Ausgleich mit Hilfe eines Faktors ist dann nicht genügend Einfacher, um diese Alternative interessant zu machen.

9.4.2 Spielflusskoordination

Wie wird der Wechsel vom Startbildschirm zur Spieldarstellung und bei Spielende wieder zurück realisiert? Wie können beim Abbau von Linien dem anderen Spieler Strafsteine eingeschoben werden?

Faktoren

- Der Wechsel von Startscreen zu Spiel und zurück soll mehrmals möglich sein, da er vor und nach jedem Spiel nötig ist.
- Die Koordination zwischen den Playergrids beim Linienabbau und -einschub soll ohne zusätzliche Beziehungen zwischen den Klassen möglich sein.

Lösung

Durch Anwendung des Singleton-Patterns wird eine zentrale Koordinationsstelle geschaffen. So wird sichergestellt, dass keine Nebeneffekte auftreten.

Ungelöste Probleme

keine

Erwogene Alternativen

keine

9.4.3 Modellierung des Spielerfelds

Wie werden Reihen erkannt? Wie wird sichergestellt, dass nicht zwei Steine in derselben Zelle sind?

Faktoren

- Ein Feld auf dem Playergrid kann nur durch einen Stein besetzt sein.
- Ein durch den Spieler gesetzter Stein darf nicht auf einem anderen Stein zu liegen kommen.
- Die Position jedes Steines muss bekannt sein für die Erkennung von Linien.

Lösung

Pro Playergrid wird eine Matrix als Modell verwendet. In diese wird eingetragen, welcher Stein sich in welchem Feld befindet. So können wir ermitteln, ob Steine platziert werden können und ob Linien komplett sind. Beim Abbauen von kompletten Linien oder Hinzufügen von Neuen wird entsprechend das Modell angepasst und die grafischen Elemente dementsprechend aktualisiert.

Ungelöste Probleme

keine

Erwogene Alternativen

keine

9.4.4 Animationen

Wie werden die Animationen implementiert?

Faktoren

- Animationen müssen flüssig laufen.
- Animationen müssen zeitlich mit den Frames synchronisiert sein.
- Animationen müssen kombinierbar sein.

Lösung

Wir benützen die Animator-Klasse des Frameworks, welche mit eigenen Timern die Zeit misst und je Interval das Objekt entsprechend anpasst. Das Ende der Animation wird mittels Event gemeldet, so dass darauf reagiert werden kann.

Ungelöste Probleme

Werden viele Animationen gestartet, kann es zu Engpässen bei der Performance führen.

Erwogene Alternativen

Eine andere Möglichkeit Animationen zu steuern haben wir im XNA-Framework von Microsoft kennengelernt. Dabei wird die vergangene Zeit per Frame in der Update()-Methode mitgegeben und anhand dieser die entsprechenden Veränderungen vorgenommen. Für diese Lösung hätten wir aber im Framework viele tiefgreifende Änderungen machen müssen.

9.5 Architekturkonzepte

9.5.1 GameCoordinator

Konzept

Der GameCoordinator hat zwei Aufgaben. Als erste Aufgabe koordiniert er den Wechsel zwischen StartScreen und Spiel. Dafür wird er vom StartScreen informiert, wenn beide Spieler ihre Bereitschaft erklärt haben zu spielen, so dass er das Spiel in Gang setzen kann. Beim Spielende wird der GameCoordinator ebenfalls informiert, so dass die nötigen Funktionen ausgeführt werden um den Sieger zu benennen und danach in den StartScreen zu wechseln. Die zweite Aufgabe ist das Koordinieren des Linienabbaus. Dabei meldet das eine Playergrid, dass eine komplette Linie abgebaut wurde. Anschliessend löst der GameCoordinator auf dem anderen Playergrid das Hinzufügen einer Straflinie aus. Dieser Vorgang wird im folgenden Sequenzdiagramm noch veranschaulicht.

Sequenzdiagramm

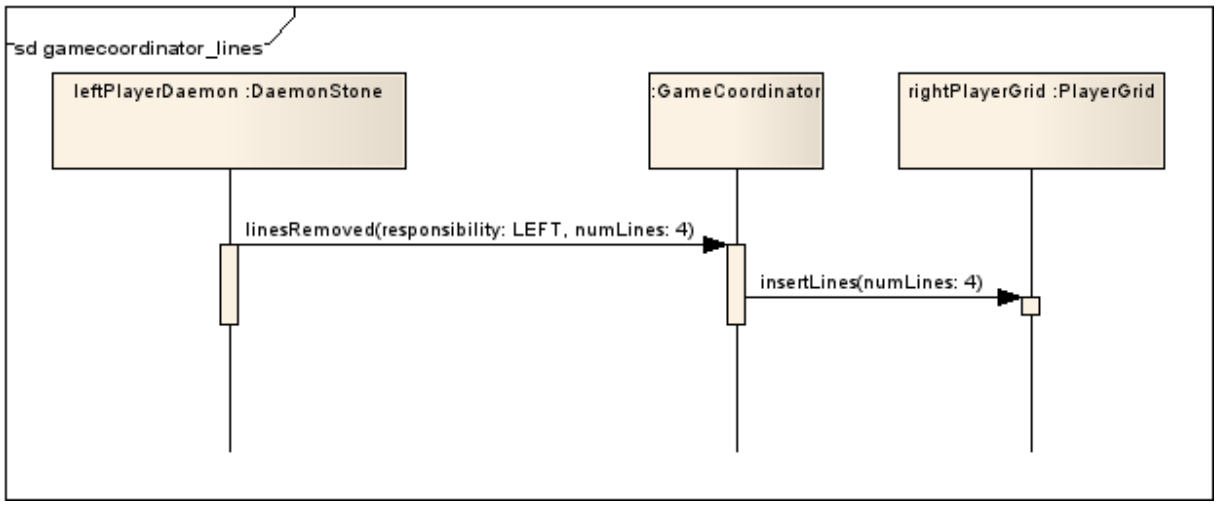


Abbildung 9.2: SSD des GameCoordinators beim Linienabbau

9.5.2 Bewegung der Steine

Konzept

Das to-fuse Application Framework bietet die Möglichkeit, Objekte die von SceneGraphObject ableiten als Kindobjekt eines anderen SceneGraphObjects einzutragen. Der grosse Vorteil davon ist, dass Position und Rotation des Kindobjekts relativ zu seinem Vaterobjekt gesetzt werden. Beim Zeichnungsvorgang wird dann die korrekte Weltkoordinate anhand der Objekte welche auf dem Pfad zum Baum-Root liegen berechnet.

Dieses Konzept konnten wir für das Bewegen der Tetris-Steine ausnutzen. Die Steine werden als Kindobjekte dem jeweiligen Playergrid angehängt. Da die Playergrids Ebenen sind, können wir die Steine auf dem zweidimensionalen Koordinatensystem des Playergrids bewegen, was sehr viel einfacher ist als direkt dreidimensionale Positionen zu berechnen.

Das selbe Konzept haben wir auch auf die Tetris-Steine selber angewandt. Ein Tetris-Stein besteht aus vier Basis-Steinen, welche relativ zur Position des Tetris-Steins angeordnet sind. Jede Positions- und Rotationsänderung des Tetris-Steins wird folglich automatisch auf die einzelnen Basis-Steine übertragen. Dies vereinfacht den Umgang mit kombinierten Objekten wie dem Tetris-Stein enorm.

9.5.3 Animationen

Konzept

Um Animationen im Spiel zu realisieren verwenden wir die im to-fuse Framework bereits verwendete Animator Klasse. Beim Starten einer Animation wird ein neuer Animator instanziert und gestartet. Läuft dieser, informiert er zuvor registrierte TimingTargets in unregelmässigen Abständen über die bereits verstrichene Zeit. Jedes TimingTarget transformiert oder versetzt das entsprechende Objekt dann jeweils um den Anteil, der gemäss verstrichener Zeit vorgesehen ist.

Sequenzdiagramm

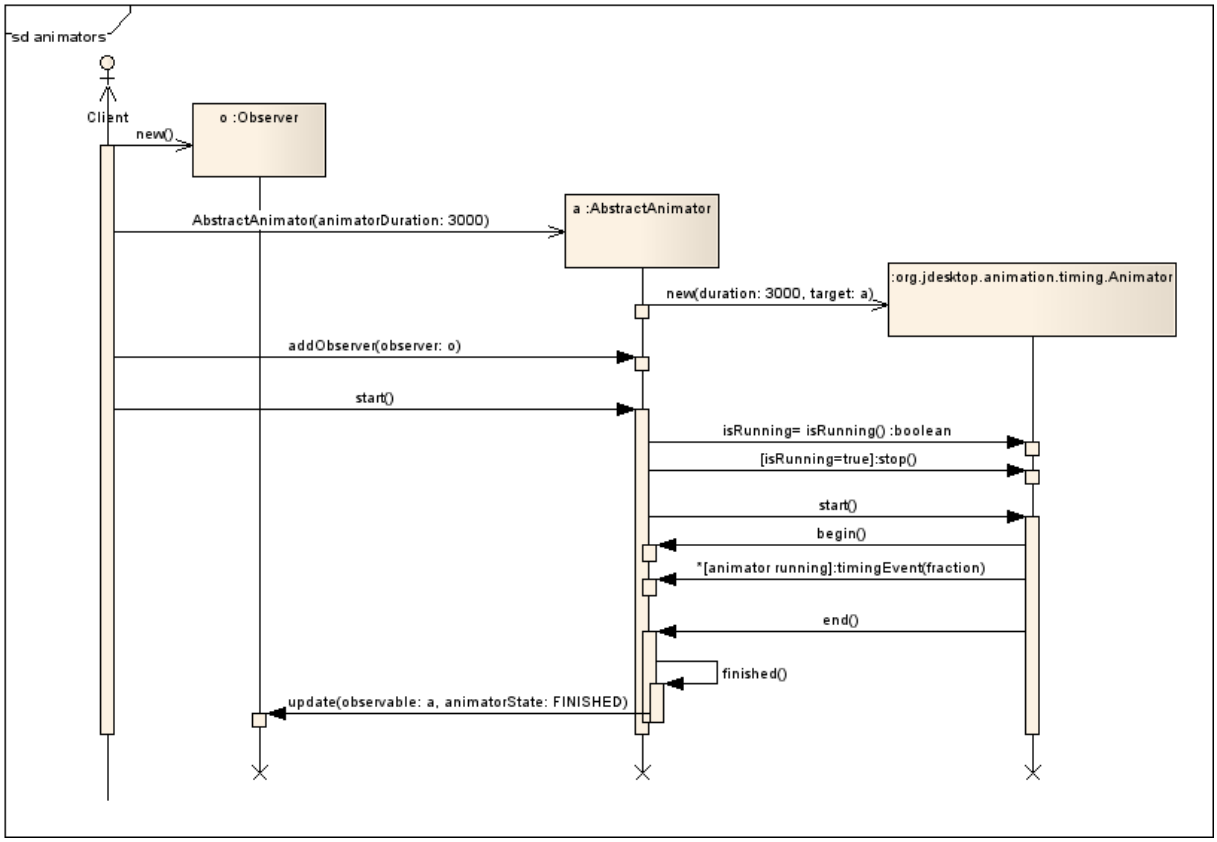


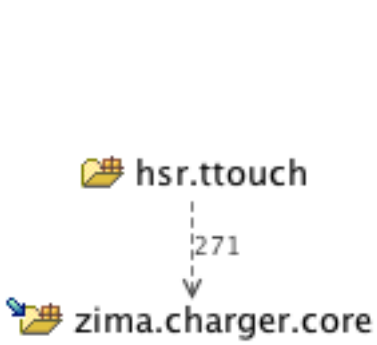
Abbildung 9.3: SSD der Animator Klassen

9.6 Logische Architektur

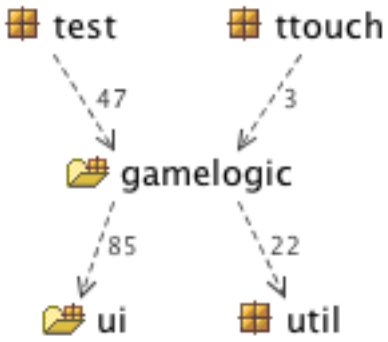
T-Touch ist in einer Zweischichtenarchitektur aufgebaut, die GameLogic- und die UI-Schicht. Eine Persistenz-Schicht ist nicht vorhanden weil ein Spiel jeweils in sich abgeschlossen ist und keine Daten gespeichert werden.

Die folgenden Bilder illustrieren die Package-Abhängigkeiten des Projekts.

Abbildung 9.4: Übersicht der externen Abhingigkeiten



(a) Abhängigkeitsgraph zum to-fuse Application Framework



(b) Übersicht der internen Abhingigkeiten

Abbildung 9.5: Die Package-Abhängigkeiten des Projekts

9.6.1 Übersicht

Das nachfolgende Diagramm zeigt die Package- und Klassenstruktur der Applikation.

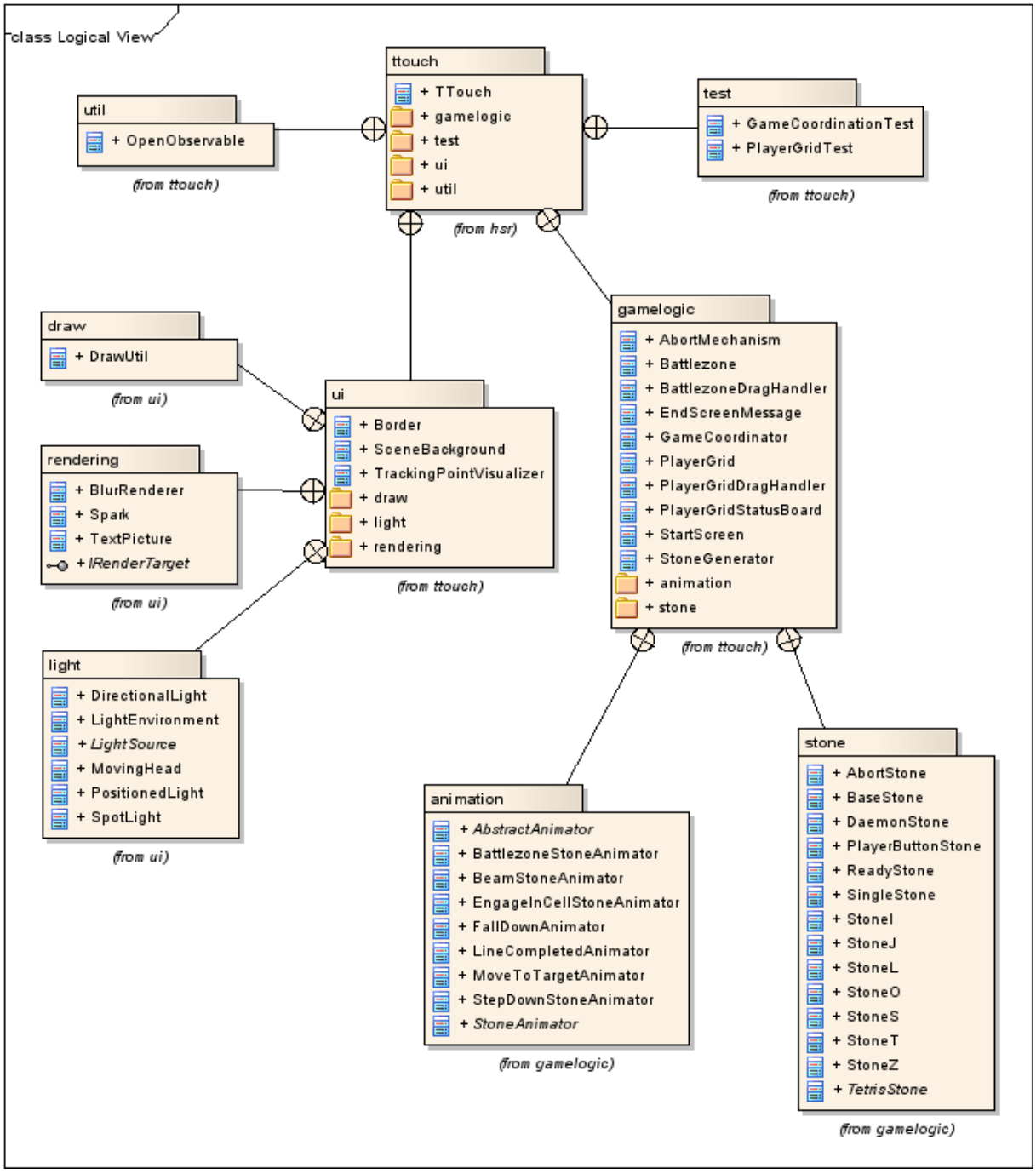
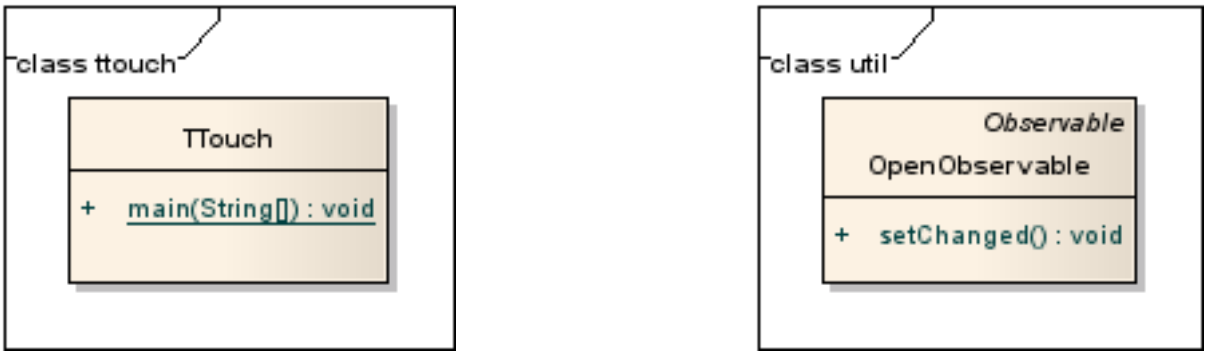


Abbildung 9.6: Package- und Klassenstruktur von T-Touch

9.6.2 Design Pakete

Package ch.hsr.ttouch

Das Package ch.hsr.ttouch bildet das Basispackage und enthält die Main-Klasse zum Starten des Spiels sowie die Subpackages gamelogic, ui und util und test. Auf gamelogic und ui wird weiter unten eingegangen. Im Subpackage util ist die Klasse PublicObservable enthalten welche ermöglicht, das Observer-Pattern und die dazu vorhandene Implementierung in Java als Klassen-Field nutzen zu können.



(a) Klassendiagramm ttouch

(b) Klassendiagramm ttouch.util

Abbildung 9.7: Klassendiagramme der Packages ch.hsr.ttouch und ttouch.util

Package test

Hier sind sämtliche Unit-Test enthalten.

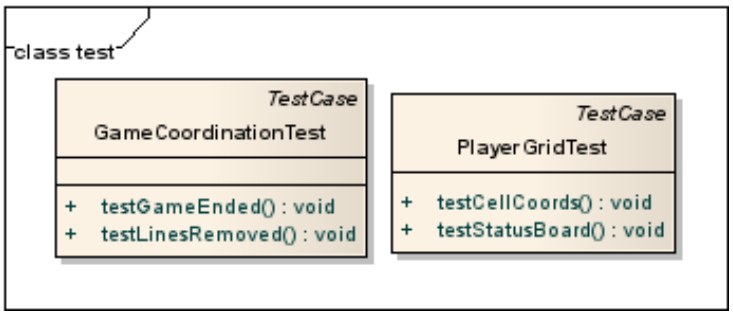


Abbildung 9.8: Klassendiagramm des Package test

Package gamelogic

In diesem Paket befinden sich die Klassen, welche die Spiellogik enthalten. Das heisst, sie koordinieren den Spielfluss, stellen die korrekte Behandlung der Benutzereingaben sicher und halten die Zustände des laufenden Spieles fest. Des weiteren befinden sich auch die Subpackages animation und stones in diesem Package, da auch diese teil der Spiellogik sind.

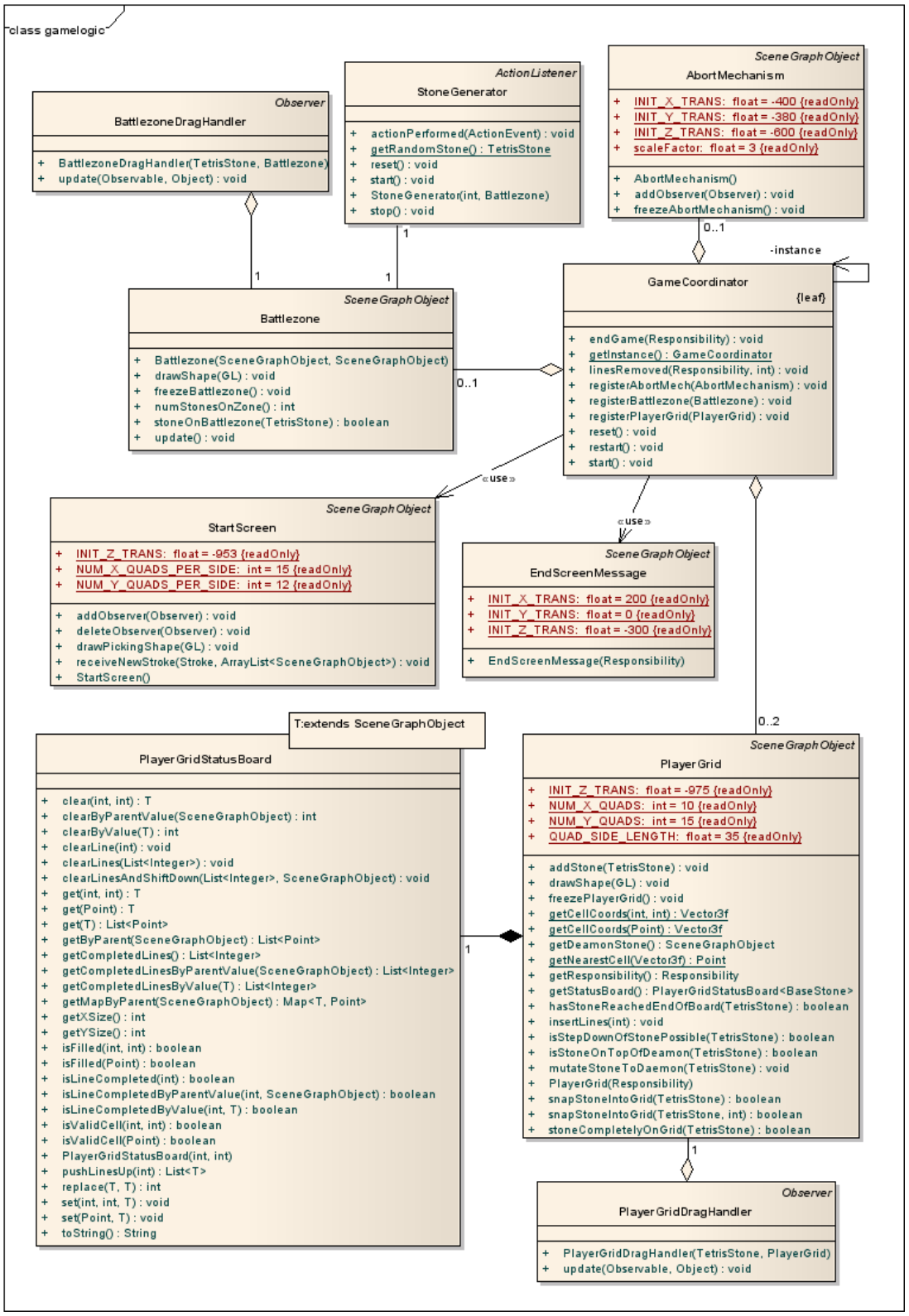


Abbildung 9.9: Klassendiagramm des Package gamelogic

Klassen

BattlezoneDragHandler
Steuert das Verhalten eines Tetris-Steins während sich dieser auf der Battlezone aufhält.

StoneGenerator
Generiert neue Tetris-Steine und hängt diese in die Battlezone ein.

AbortMechanism
Leitet den Spielabbruch ein wenn beide Spieler das Spiel abbrechen wollen.

Battlezone
Steuert den Übergang eines Tetris-Steins zu den entsprechenden Spielfeld-Ebenen.

GameCoordinator
Koordiniert den Spielfluss. Dieser beinhaltet das Spielende, den Spielabbruch sowie die Koordination zwischen den PlayerGrid's wenn Linien abgebaut wurden und Straflinien eingefügt werden müssen.

StartScreen
Löst den Spielstart aus und stellt sich dass beide Spieler zuvor ihre Bereitschaft signalisiert haben.

EndScreenMessage
Ist verantwortlich dafür die korrekte Meldung bei Spielende auf den Spielfeldern anzuzeigen.

PlayerGridStatusBoard
Speichert welche Steine auf welchen Feldern liegen als Matrix und bietet Algorithmen an um komplette Linien abzubauen oder neue einzufügen.

PlayerGrid
Repräsentiert das persönliche Feld eines Spielers.

PlayerGridDragHandler
Steuert das Verhalten eines Tetris-Steins während sich dieser auf dem PlayerGrid aufhält.

Package animation

Dieses Paket enthält alle Animatorklassen. Diese führen animierte Bewegungen der Objekte aus und koordinieren diese wo nötig.

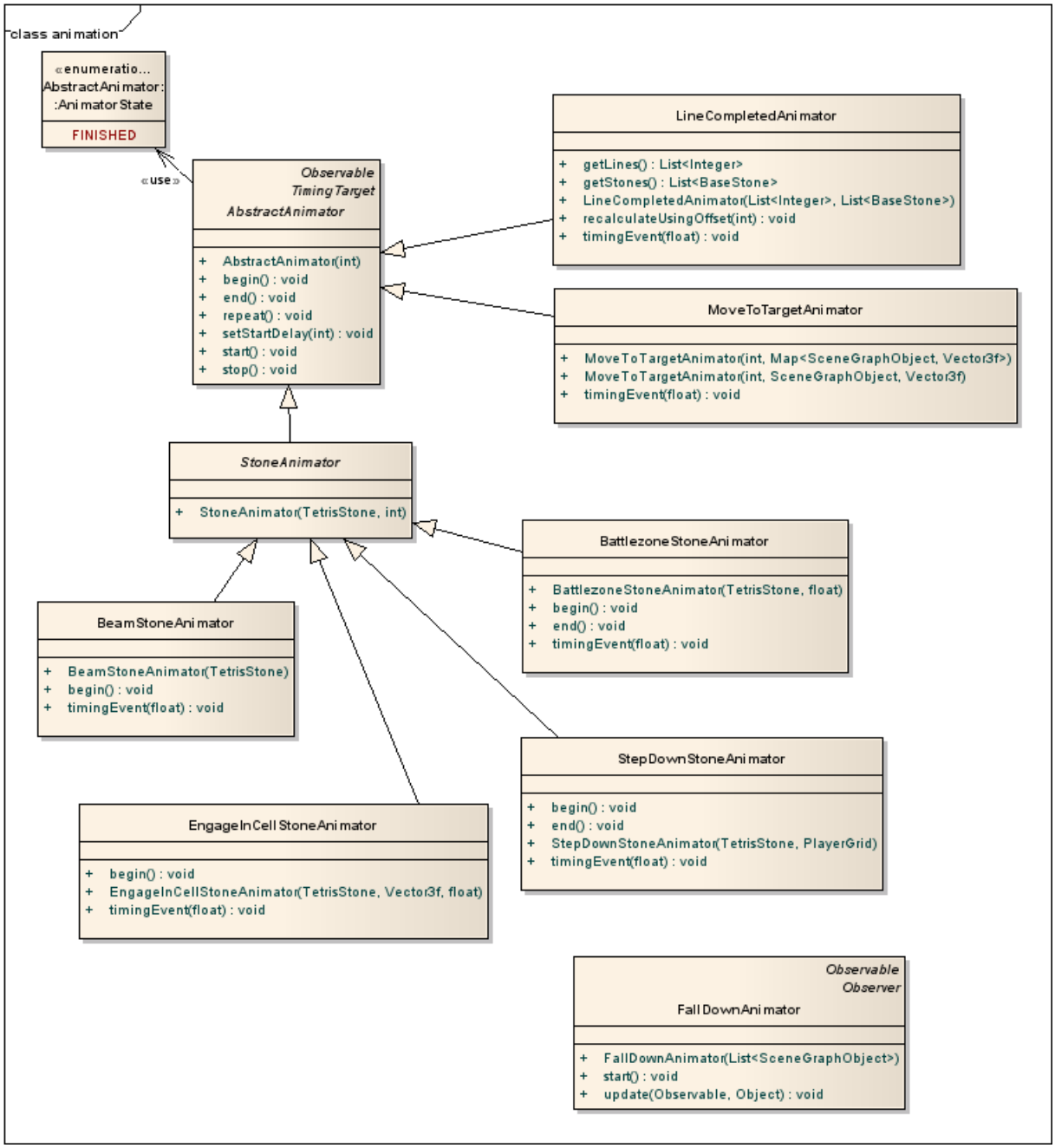


Abbildung 9.10: Klassendiagramm des Package animation

Klassen

LineCompletedAnimator
Animation: Blinkende Linien.

MoveToTargetAnimator
Animation: Bewegung von SceneGraphObject's zu beliebigen Zielkoordinaten.

BattlezoneStoneAnimator
Animation: Fahrende Bewegung von Tetris-Steinen auf der Battlezone.

StepDownStoneAnimator
Animation: "Klettern" von Tetris-Steinen auf dem PlayerGrid.

EngageInCellStoneAnimator
Animation: Tetris-Stein in die nächstgelegene Zelle verschieben/rotieren.

BeamStoneAnimator
Animation: Zerstören eines Tetris-Steins.

FallDownAnimator
Animation: SceneGraphObject's in die Tiefe fallen lassen.

Package stone

Das Package stone enthält die konkreten Tetris-Steine, welche beim Spielen verwendet werden sowie auch den BaseStone und spezielle Steine, die als Buttons Verwendung finden.

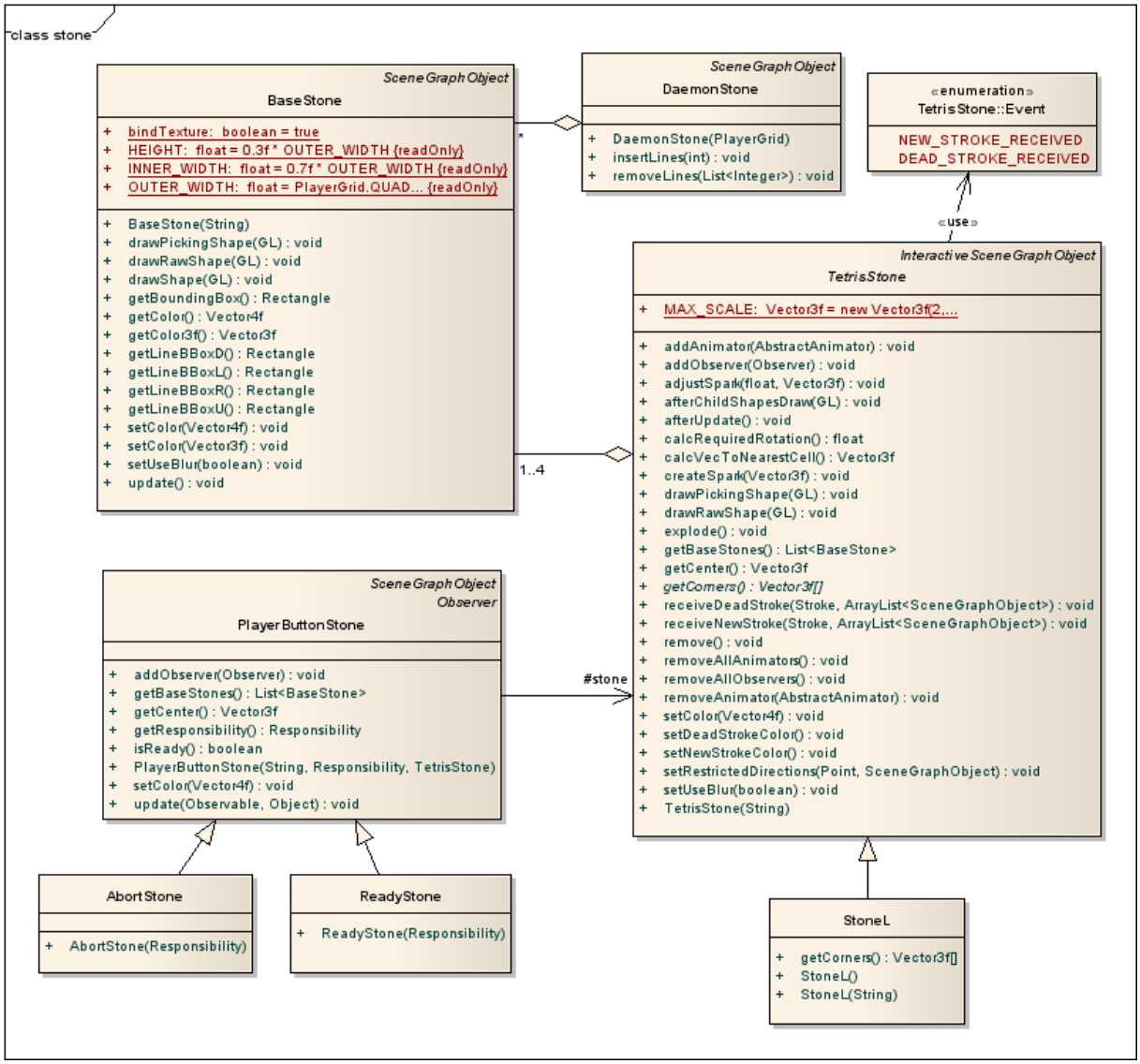


Abbildung 9.11: Klassendiagramm des Package stone

Klassen

BaseStone

Der Basis-Stein aus welchem die Tetris-Steine bestehen.

DaemonStone

Verwaltet alle, auf dem PlayerGrid festgesetzten Basis-Steine. Ein Basis-Stein wird dem

DaemonStone übergeben sobald der vorher zuständige Tetris-Stein nicht mehr bewegt werden kann. Der Tetris-Stein wird bei der Übergabe zerstört.

PlayerButtonStone
Stellt Button Funktionalität zur Verfügung wie sie beim Abort- und ReadyStone benötigt werden.

AbortStone
Button der es dem Spieler erlaubt den Spielabbruch zu signalisieren.

ReadyStone
Button der es dem Spieler erlaubt die Bereitschaft für ein neues Spiel zu signalisieren.

TetrisStone
Repräsentiert einen Tetris-Stein und bietet Möglichkeiten um diverse Animatoren anzuhängen und auf Berührungsereignisse reagieren zu können.

StoneL
Eine konkreter Tetris-Stein. Erzeugt die entsprechenden BaseStone's und ordnet diese visuell "L" - ähnlich an. Es existieren noch weitere konkrete Tetris-Steine welche die restlichen Formen abdecken.

Package ui

Die Klassen für die Darstellung auf dem Bildschirm sind in diesem Package enthalten. Wie in der Packages-Übersicht zu sehen ist, haben wir alle Klassen die mit Licht und Effekten zu tun haben, je in einem separate Package gekapselt und werden diese weiter unten erläutern.

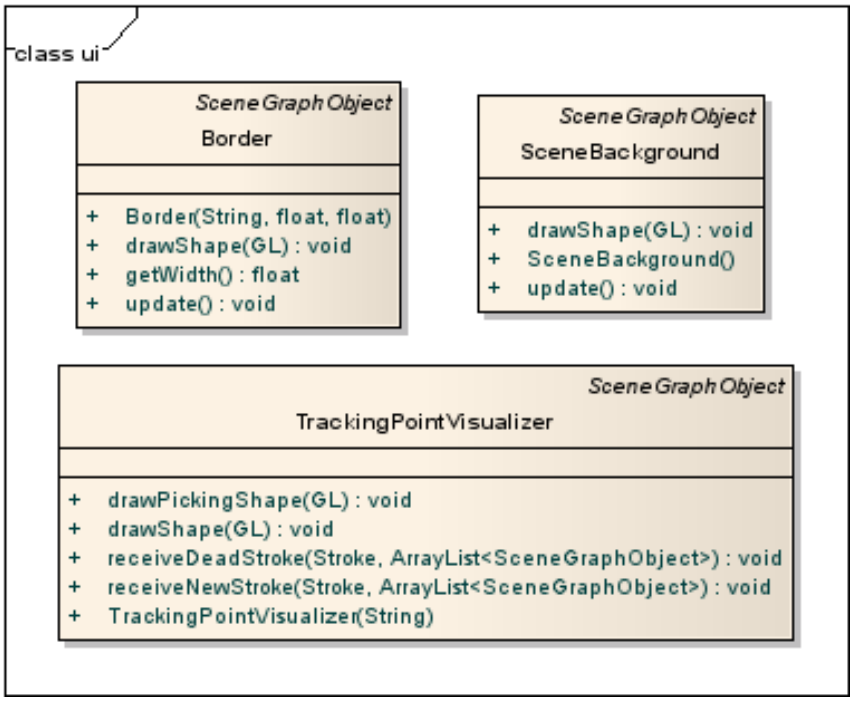


Abbildung 9.12: Klassendiagramm des Package ui

Klassen

TrackingPointVisualizer

Verantwortlich für das visuelle Feedback (weisser Kreis) bei Berührung der Tisch-Oberfläche.

Package draw

In diesem Package ist nur eine Klasse enthalten, die alle von uns benötigten Zeichenfunktionen für Steine und Bilder enthält.

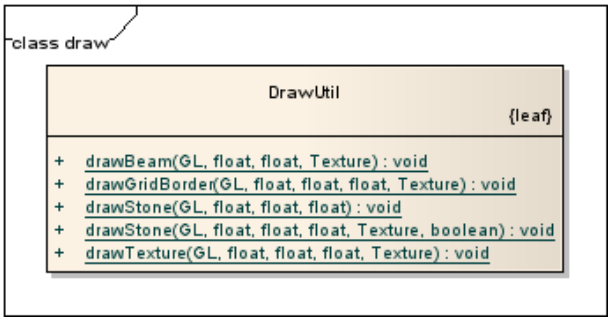


Abbildung 9.13: Klassendiagramm des Package draw

Package light

Dieses Paket enthält alle Klassen welche für die Beleuchtung der Szene benötigt werden.

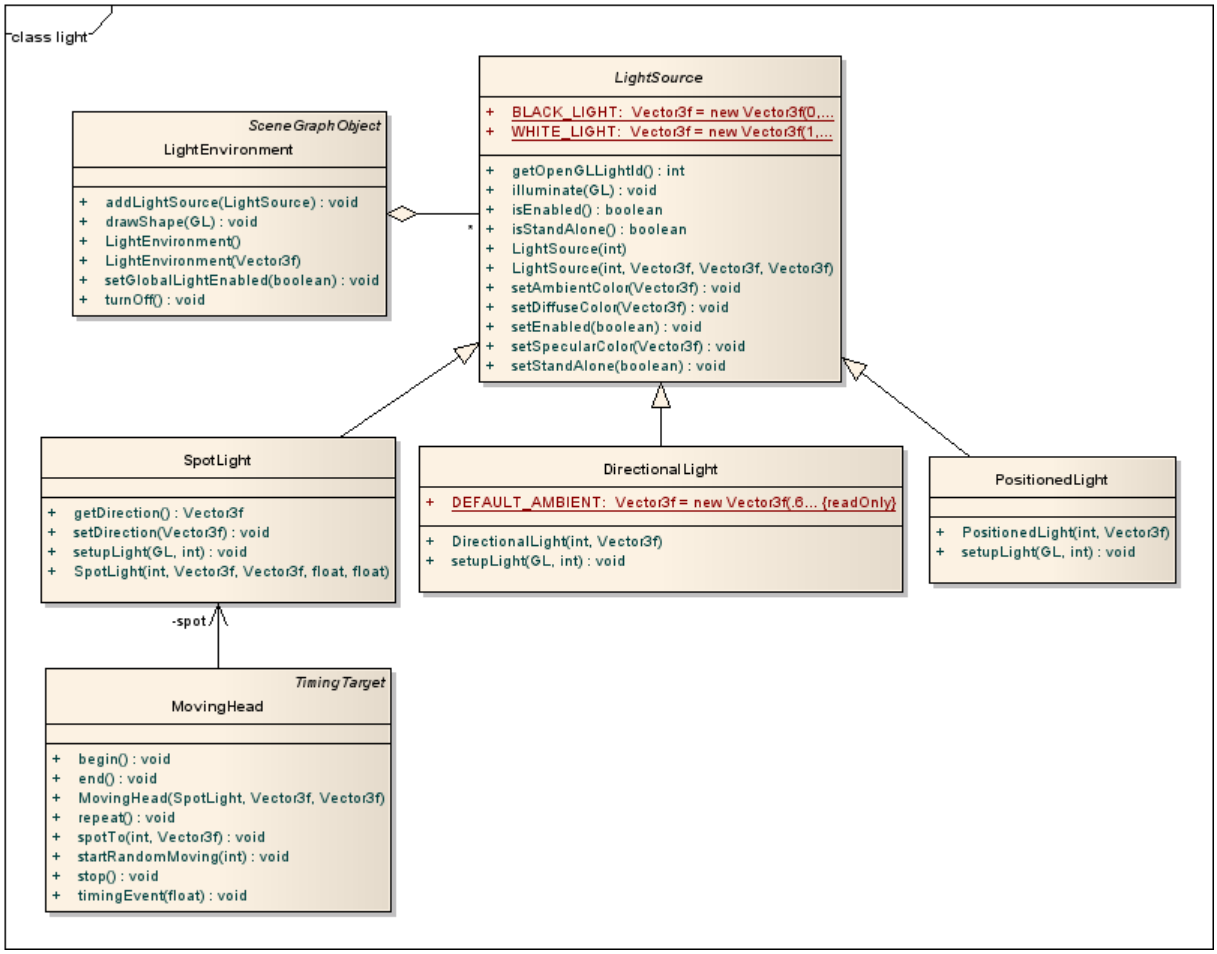


Abbildung 9.14: Klassendiagramm des Package light

Klassen

LightEnvironment

Verantwortlich für Beleuchtung der Szene.

LightSource

Repräsentiert eine OpenGL-Lichtquelle.

SpotLight

Implementation einer statischen Scheinwerfer-Lichtquelle.

DirectionalLight

Unendlich weit entfernte, gerichtete Lichtquelle.

PositionedLight
Lichtquelle die im Raum positioniert ist und in alle Richtungen strahlt.

MovingHead
Bietet die Funktionalität eines schwenkbaren Schweinwerfers (SpotLight).

Package rendering

Alle Klassen, die für spezielle Effekte wie Blur auf den Tetris-Steinen verantwortlich sind, befinden in diesem Package.

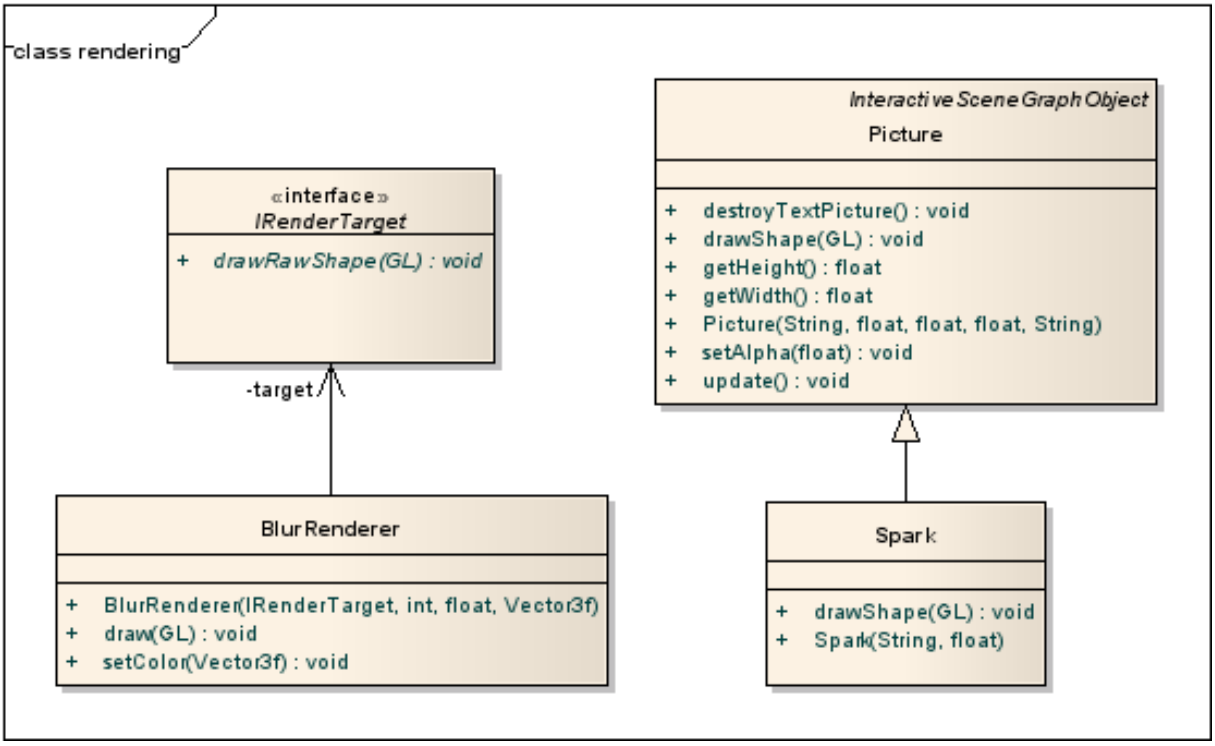


Abbildung 9.15: Klassendiagramm des Package rendering

Klassen

BlurRenderer

Kann durch mehrfaches Zeichnen des angegebenen IRenderTarget's einen Blur-Effekt erzeugen.

Spark

Zeichnet die Stein-Explosion Funkenähnlich.

Schnittstellen Das Package definiert die Schnittstelle IRenderTarget. Dieses Interface kann verwendet werden um Effekte wie Blur auf einem Target-Objekt erzeugen zu können.

9.7 Datenspeicherung

T-Touch ist auf kurze, in sich abgeschlossene Spiele ausgelegt. Die Benutzer können kein Profil erstellen und es gibt keine Bestenliste. Einziges Ziel ist das Besiegen des Gegners im gerade laufenden Spiel. Somit besteht kein Bedarf an einer Datenspeicherung.

10 Testdokumentation

10.1 Änderungshistorie

Rev.	Datum	Wer	Was wurde geändert
0	11.12.2009	ral	Neues Dokument
0	16.12.2009	ral	Vollständige Version
0	17.12.2009	mtr	Qualitätskontrolle

10.2 Einführung

10.2.1 Zweck

Dieses Dokument dient zur Protokollierung der System und Usability Tests.

10.2.2 Definitionen und Abkürzungen

Definitionen und Abkürzungen sind im *Glossar* zu finden.

10.3 Systemtest

10.3.1 Voraussetzung

Das Spiel wurde speziell für die *to-fuse Multi-Touch Platform* entwickelt und läuft nur auf dieser, da es auf Basis des entsprechenden Frameworks entwickelt wurde. Darum konnten wir es auch nur auf der *to-fuse Multi-Touch Platform* testen.

10.3.2 Testbeschreibung

Der nachfolgende Test wurde auf dem finalen Prototypen gemacht und stellt die grundlegende Funktionalität des Spieles sicher. Getestet werden die Funktionalitäten gemäss Anforderungsspezifikationen der Use Cases.

Systemtest UC1: T-Touch spielen

Dieser Systemtest wurde mit der komplett implementierten Beta-Version des Spieles gemacht.

Startscreen
Der Startscreen funktioniert einwandfrei. Das Spiel wird gestartet wenn beide Kontrahenten ihren Startknopf gedrückt haben.

Erscheinen der Steine
Die Steine erscheinen an der gewünschten Position und mit der gewünschten Häufigkeit.

Steine bewegen sich in Richtung Playergrid
Die Bewegung der Steine in der Battlezone läuft ohne Probleme und die Steine gehen zum richtigen Playergrid.

Übergang der Steine aufs Playergrid
Die Steine werden nicht immer korrekt übergeben von der Battlezone an das entsprechende Playergrid. Teilweise wird das Zerstören des Steines ausgelöst, aber der Stein fährt dann trotzdem auf dem Grid weiter. Dabei befindet er sich noch halb in der Explosionsanimation und ist nicht mehr durch den Spieler manipulierbar.

Tetris-Stein manipulieren
Steine ziehen und rotieren funktioniert.

Tetrsistein positionieren
Positionierung der Steine funktioniert.

Reihen erkennen und abbauen
Reihen werden korrekt erkannt und abgebaut.

Strafsteine dem Gegner hinzufügen
Straflinien werden richtig hinzugefügt.

Sieger ermitteln und anzeigen
Das Spielende wird nicht immer korrekt erfasst.

Rückkehr zum Startscreen
Die Rückkehr zum Startscreen verläuft problemlos.

Spiel abbrechen
Beim Spielabbruch treten keine Fehler auf.

10.3.3 Verbesserungsmöglichkeiten

Den Fehler beim Übergang aufs Playergrid gilt es noch zu beheben. Leider fehlt aber die Zeit, die Fehlerquelle zu finden.

10.4 Usability Test

10.4.1 Voraussetzung

Die Testpersonen für den Usabilitytest wurden auschliesslich aus Studenten der HSR Rapperswil ausgesucht, da T-Touch nur auf der *to-fuse Multi-Touch Platform* gespielt werden kann. So konnten wir keine sehr breitgefächerte Auswahl an Meinungen zu Bedienung und Eindruck gewinnen.

Viele der Kommentare und Anregungen zur Usability des Spiels haben wir von Personen bekommen, die kurze Tests des Spiels gemacht haben. Diese Anregungen sind dann jeweils direkt eingeflossen. Deshalb haben wir hier nur einen repräsentativen Testbogen abgebildet.

10.4.2 Fragebogen

Optischer Eindruck

Wie ist der optische Eindruck des Spiels? M: Gut. Es hat coole Effekte und die 3D Grafik gefällt.

Verbesserungsvorschläge M: Das Licht auf die Steine auf dem Spielfeld irritiert, weil es die Farbe der Steine verändert, aber die Funktion die selbe bleibt.

Bedienung des Spiels

Wie ist der generelle Eindruck zur Bedienung des Spiels? M: Gut.

Ist die Steuerung intuitiv? M: Ja.

Lassen sich die Steine gut bewegen und drehen? M: Die Steine lassen gut bewegen und auch die Drehung funktioniert normalerweise gut. Allerdings gibt es zwei Probleme. Wenn die Steine in den Feldern platziert werden drehen sie sich manchmal zur falschen Zelle und nach dem ersten Mal drehen werden die Steine in wenigen Fällen bei erneutem drehen komisch versetzt.

Wie schwer ist es, Tetris Steine zu platzieren? M: Ohne die Steine zu drehung geht es gut, mit Drehung ist es etwas schwieriger.

Verbesserungsvorschläge M: Bis auf die Fehler ist die Bedienung gut.

Spielfluss und Schwierigkeitsgrad

Ist der Spielfluss angenehm? M: Ja

Wie ist die Spielgeschwindigkeit? M: Angenehm.

Verbesserungsvorschläge M: Bei längeren, ausgeglichenen Spielen wäre es interessant, wenn die Steine mit der Zeit schneller erscheinen würden.

11 Persönliche Berichte

11.1 Persönlicher Bericht von Ricardo Alvarez

11.1.1 Erfahrungsbericht

Ziel dieses Projekts war es, ein Spiel für die Multi-Touch Plattform von to-fuse zu entwickeln, deren Framework auf Java und OpenGL basiert. Dies stellte insofern eine grosse Herausforderung dar, dass sowohl OpenGL wie auch die Multi-Touch Technologie Neuland für uns war. Wir hatten zwar bereits im SE2-Projekt ein Spiel entwickelt, allerdings mit dem XNA.NET Framework. Somit war ich gespannt auf die Unterschiede der Technologien.

Da die Projektvorgabe darauf beschränkt war, dass es ein Spiel werden soll, musste erst eine Spielidee gefunden werden. Zu diesem Zweck haben wir die Ausstellung im Technorama von Microsoft besucht, welche einige Multi-Touch Applikationen zeigte, unter anderem auch ein Spiel für den Microsoft Surface. So konnten wir erste Eindrücke gewinnen zu Bedienung und Eindruck von Multi-Touch Applikationen. Wir haben uns dann dazu entschieden ein Multiplayer Tetris zu entwickeln, weil die Spielidee zwar schon älter ist, aber nach wie vor viel Spielspass verspricht. Vor allem als Multiplayerspiel sahen wir Potenzial für interessante Duelle. Zudem ist es graphisch verhältnismässig einfach umzusetzen, zumal keiner von uns in diesem Bereich erfahren ist.

Da kam dann bereits die erste grössere Herausforderung, nämlich das Erstellen der Spezifikation. Da wir mit der Multi-Touch Technologie nicht vertraut waren, wussten wir zu jenem Zeitpunkt nicht, welche Ideen funktionieren und was nicht. Wir entschieden uns, mittels Prototypen zu ermitteln, was wir umsetzen wollen. In dieser Phase entschieden wir uns dann dazu, das Spiel in 3D zu gestalten, da wir so dank der perspektivischen Sicht mehr Platz zur Verfügung haben. Diese Entscheidung machte das Projekt zwar noch interessanter, weil wir so Erfahrungen mit 3D Programmierung sammeln konnten, aber die Implementaiton gestaltete sich auch viel schwieriger, da Berechnungen einiges komplexer sind in 3D.

Natürlich gab es auch Sackgassen in der prototyping Phase. Zum Beispiel haben wir einige Zeit in Versuche mit Kollisionserkennung gesteckt und auch relativ einfach implementieren können. Allerdings war die Reaktion auf Kollisionen technisch komplex und führte vom spielerischen Aspekt für für den Benutzer zu unangenehmen Situationen. Zum Beispiel wenn der Finger weiterfahren kann auf der Tischoberfläche, aber das Objekt hängen bleibt. Deshalb wurde dieser Ansatz wieder verworfen zu gunsten alternativer Modelle.

Eine weitere Schwierigkeit dieser Umstellung auf 3D war, dass das Framework von to-fuse

bisher nur für 2D Applikationen genutzt worden war und somit einiges an Funktionalität fehlte für unser Spiel. Gerade im Bereich Tracking und Bewegen der Objekte haben wir einige Erweiterungen dem Framework hinzugefügt. Gute unterstützt haben uns Christian Iten und Emanuel Zgraggen von to-fuse, herzlichen Dank nochmal an dieser Stelle.

Nachdem wir dann endlich die Spezifikation für das Spiel erstellt hatten, ging die Programmierung des Spiels relativ schnell, da wir viel Code aus den Prototypen übernehmen konnten.

Auch beim Projektmanagement haben wir für uns neue Wege beschritten. Wir haben im SE2-Projekt RUP kennengelernt, fanden es aber für dieses Projekt zu wenig Agil. Zumal sich die anfängliche Planung schwierig gestaltet ohne Spezifikation. Deshalb haben wir SCRUM als Richtlinie genommen. Dies stellte sich als gute Entscheidung heraus, da wir zu Anfang nur die Grobplanung machen mussten und die jeweilige Detailplanung dann erst für die Sprints gemacht werden. So konnten wir jederzeit auf neue Erkenntnisse in der Entwicklungsphase reagieren.

11.1.2 Fazit

Das Projekt stellt eine interessante Erfahrung dar und hat mir in vielen Punkten einiges gebracht. Ich habe mit der Multi-Touch Technologie und der Programmierung in 3D zwei faszinierende Gebiete kennengelernt. Und dass wir das Framework nicht nur benutzen konnten, sondern es offen zur Verfügung stand und bei Bedarf erweitert werden musste war eine ungewohnte, aber interessante Erfahrung. Und am Schluss ist es sehr erfreulich und motivierend, wenn sich eine Gruppe um das Spiel sammelt und es begeistert ausprobiert.

11.2 Persönlicher Bericht von Mischa Trecco

11.2.1 Erfahrungsbericht

Dieses Projekt fing gleich sehr interessant an. Nur wenige Tage nachdem wir die Zusage für die ausgeschriebene Arbeit “Table Computing Multi-Player Game” hatten, fand ich mich bereits im Technorama in Winterthur wieder. An der Ausstellung “The Magic of Software”, die anlässlich des 20 jährigen Bestehens von Microsoft Schweiz stattfand, liessen wir uns noch vor Semesterbeginn von Anwendungen auf dem Microsoft Surface, HP TouchSmart und anderen innovativen Produkten für eine Spiel-Idee inspirieren.

Einige kreative Sitzungen später stand fest, dass wir für die to-fuse Multi-Touch Platform ein Multi-Player Spiel basierend auf dem Spieleklassiker “Tetris” entwickeln wollen. Die Idee eignete sich nicht nur weil die grafischen Elemente einfache symmetrische Figuren waren und wir nicht zwingend einen Grafiker benötigten, sondern vorallem weil das Spielprinzip stark erweiterbar war. Wir mussten uns ja erst noch in das to-fuse Framework und vorallem in OpenGL einarbeiten was Aufwandschätzungen zu diesem Zeitpunkt erschwerte und uns darin bestärkte eine flexibles Spielkonzept zu wählen um agil auf Fehlschätzungen reagieren zu können.

Wir hatten schnell einige 2D-Prototypen zusammen um zuvor definierte Risiken wie z.B. die Reaktionszeit des to-fuse Tisches abzuklären. Die wirklich technische Herausforderung kam aber noch auf uns zu. Die 2D-Prototypen zeigten uns dass die Auflösung und Grösse der Oberfläche nicht für vier Spielfelder reichte und wir entschlossen uns die 3D-Möglichkeit zu nutzen um so mehr visuellen Platz durch die Z-Komponente zu erhalten.

Da unser Spiel aber die erste 3D-Anwendung auf der to-fuse Multi-Touch Platform darstellte und die 3D-Grundstrukturen im Framework noch nicht getestet waren, verloren wir viel Zeit. Die Unerfahrenheit mit OpenGL allgemein und spezifisch der 3D-Welt waren dabei ein nicht unwesentlicher Faktor. Teilweise war es sehr Aufwändig herauszufinden ob ein vorhandener Fehler durch einen selbst oder doch durch das Framework verursacht wurde. Ich war froh hatten wir mit to-fuse einen kompetenten und hilfsbereiten Industriepartner auf den wir stets zugehen konnten und mit dem wir gemeinsam Probleme anpacken und vorallem lösen konnten.

Die grösste Herausforderung, das Umsetzen von 2 dimensionalen Berührungskoordinaten auf einen Punkt auf einem Objekt im 3 dimensionalen Raum, konnten wir mit guter Teamarbeit und interessanten Diskussionen mit allen Beteiligten erfolgreich lösen. Ich war überrascht und erfreut wie gut schnell wir das meisterten!

Schwierig war definitiv auch abzuschätzen und festzulegen welche Ideen nun endgültig in das Endprodukt fliessen sollen und welche nicht. In der Abschlussphase hatten wir uns diesbezüglich teilweise etwas verschätzt und mussten dies mit einigen Überstunden kompensieren. Aber daraus lernt man ja schliesslich... :-)

11.2.2 Fazit

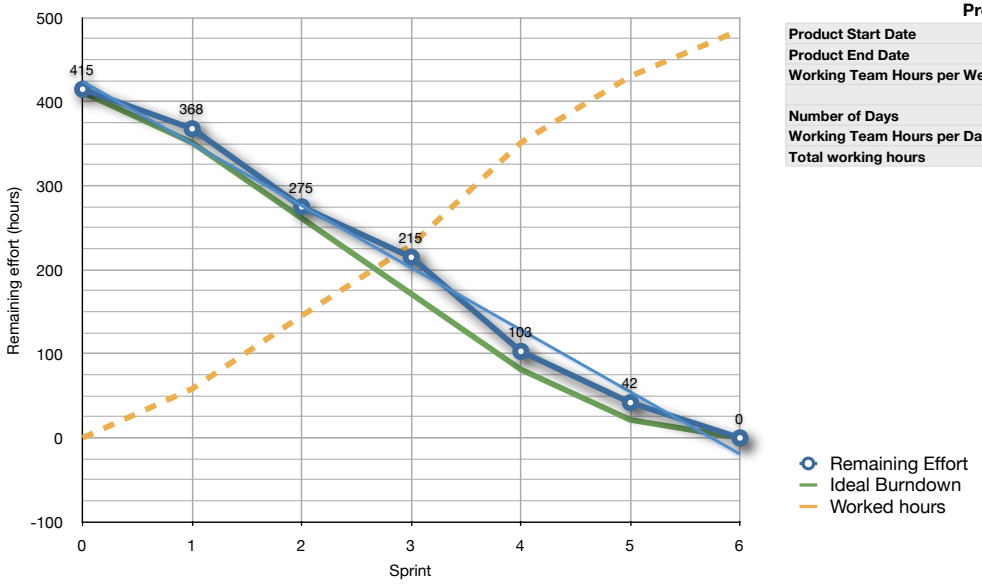
Bei diesem interessanten Projekt konnte ich viele interessante Erfahrungen machen. Nebst dem Kennenlernen der innovativen Multi-Touch Welt, konnte ich mir auch reichlich OpenGL Know-How aneignen und erste Erfahrungen mit SCRUM sammeln. Zudem war das Projekt-Thema an sich von Anfang an motivierend und ein 3D-Spiel entwickelt man so schnell auch nicht wieder. ;-) Zuletzt hat aber sicher auch die gute Zusammenarbeit und Stimmung im Team zu diesen positiven Erfahrungen beigetragen.

12 SCRUM Artefakte

Product Backlog

Nr.	Priorität	Feature	Beschreibung	Initiale Schätzung	Faktor Komplexität	Angepasste Schätzung	Hours Remaining per Sprint					
							0	1	2	3	4	5	6
1	Normal	Projektplan		5	0	5	5	0					
2	Normal	Spezifikation		14	0	14	14	8	8	10	0		
3	Normal	Technologie erarbeiten	OpenGL, to-fuse Application Framework	12	0	12	12	0					
4	Normal	Analyse: Drag & Rotate	Klärung Risiko 1: Berührungserkennung von Objekten auf dem Tisch zu ungenau.	10	0	10	10	10	0				
5	Normal	Analyse: Spielfeld	Klärung Risiko 2: Reaktionszeit Tisch zu langsam. (Spielfluss)	77	0	77	77	77	0				
6	Normal	Domainanalyse		8	0	8	8	8	8	8	0		
7	Normal	Basis Prototyp	Darstellung/Platzierung, Steine bewegen, Reihen bilden	72	0	72	72	72	72	14	0		
8	Normal	Entwicklungsprotokoll		6	0	6	6	6	6	6	0		
9	Normal	Spielbarer Prototyp	Effekte, Animationen, Stone-Handling, Spielfluss, Licht	68	0	68	68	68	68	68	0		
10	Normal	Vollständiger Spiel-Release	Bug fixing, Spielfluss, Start- & End-Screen, Tutorial	30	0	30	30	30	30	30	30	0	
11	Normal	Dokumentation Teil 1	Externes Design, SAD, Testdokumentation, Paper	43	0	43	43	43	43	43	43	16	0
12	Normal	Dokumentation Teil 2	Abstract, Persönliche Berichte, Gesamtbericht	16	0	16	16	16	16	16	16	16	0
13	Normal	Projektabgabe	Abgabe vorbereiten	8	0	8	8	8	8	8	8	8	0
14	Normal	Projektmanagement	Meetings, Projektmonitoring, Planung	46	0	46	46	22	16	12	6	2	0
				Estimated work remaining:			**415**	**368**	**275**	**215**	**103**	**42**	**0**
				Remaining working hours:			**411**	**351**	**261**	**171**	**81.4**	**21.4**	**0**
				Worked hours:			**0**	**58.5**	**146**	**230**	**352**	**431**	**486**

Product Burndown Chart



Product Info

Product Start Date	13.09.2009
Product End Date	18.12.2009
Working Team Hours per Week	30
Number of Days	96
Working Team Hours per Day	4.28571428571429
Total working hours	411.428571428571

Sprints

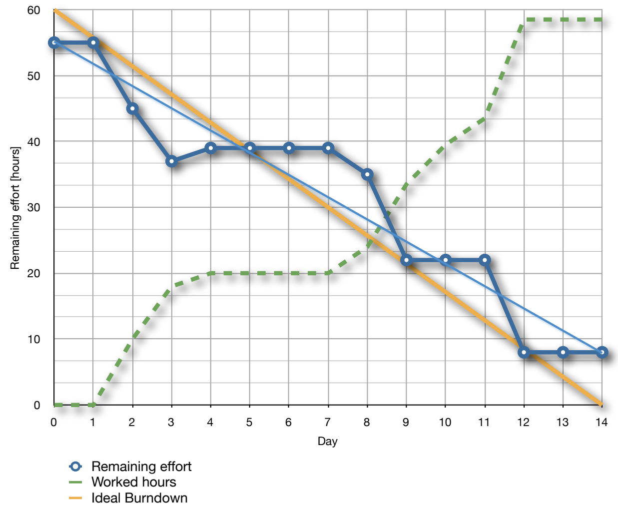
Sprint	Start date	End date	Days	Working team hours	Remaining working hours
0			0	0	411.428571428571
1	13.09.2009	27.09.2009	14	60	351.428571428571
2	27.09.2009	18.10.2009	21	90	261.428571428571
3	18.10.2009	08.11.2009	21	90	171.428571428571
4	08.11.2009	29.11.2009	21	90	81.4285714285714
5	29.11.2009	13.12.2009	14	60	21.4285714285714
6	13.12.2009	18.12.2009	5	21.4285714285714	0
			96	**411.428571428571**	
		Working hours not planned in a sprint:		**0**	

Sprint 1 Backlog

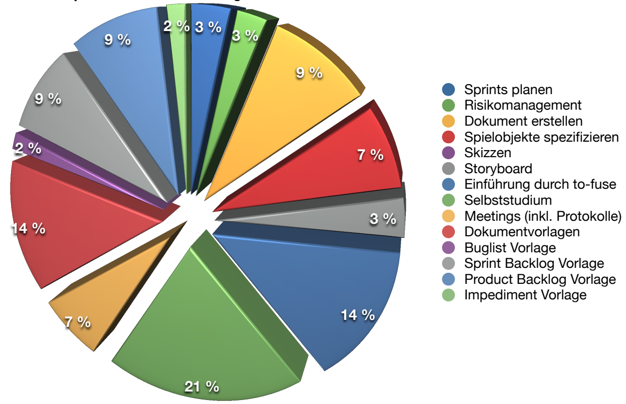
Estimated hours remaining per day | Hours worked on task per day

Nr.	Priority	Status		Task	Verantwortlich	Hours worked	0	1	Mo	2	Di	3	Mi	4	Do	5	Fr	6	Sa	7	So	8	Mo	9	Di	10	Mi	11	Do	12	Fr	13	Sa	14	So
1	High	Completed	**Projektplan**	Sprints planen	Team	2	2	2		2		2		2		2		2		2		2		0	2										
2	High	Completed		Risikomanagement	Mischa Trecco	1.5	1	1		1		1		1		1		1		1		1		0	1.5										
3	Medium	Completed		Dokument erstellen	Ricardo Alvarez	5	2	2		2		2		2		2		2		2		2		2		2	2	2		0	3				
4	Medium	Suspended	**Game Spezifikation**	Spielobjekte spezifizieren	Ricardo Alvarez	4	8	8		4	4	4		4		4		4		4		4		4		4		4		4		4		4	
5	Medium	Suspended		Skizzen	Mischa Trecco	0	2	2		2		2		2		2		2		2		2		2		2		2		2		2		2	
6	Medium	Suspended		Storyboard	Ricardo Alvarez	2	4	4		2	2	2		2		2		2		2		2		2		2		2		2		2		2	
7	Medium	Completed	**Technologie erarbeiten**	Einführung durch to-fuse	Team	8	8	8		8		8		8		8		8		8		8		8		8		8		0	8				
8	Medium	Completed		Selbststudium	Team	12	4	4		4		4		4		4		4		4		4		4		4	4	4	4	0	4				
9	High	Completed	**Organisatorisches**	Meetings (inkl. Protokolle)	Team	4	6	6		4	2	4		4		4		4		4		4		0	2										
10	High	Completed		Dokumentvorlagen	Ricardo Alvarez	8	6	6		6		2	4	4	2	4		4		4		4		0	2										
11	High	Completed		Buglist Vorlage	Mischa Trecco	1	1	1		1		0	1																						
12	High	Completed		Sprint Backlog Vorlage	Mischa Trecco	5	5	5		4	1	3	1	3		3		3		3		1	2	0	1										
13	High	Completed		Product Backlog Vorlage	Mischa Trecco	5	5	5		4	1	3	1	3		3		3		3		1	2	0	1										
14	High	Completed		Impediment Vorlage	Mischa Trecco	1	1	1		1		0	1																						
					Estimated work remaining:		**55**	**55**		**45**		**37**		**39**		**39**		**39**		**39**		**35**		**22**		**22**		**22**		**8**		**8**		**8**	
Completed:		**11/14**			**Remaining working hours:**		**60**	**55.7**		**51.4**		**47.1**		**42.9**		**38.6**		**34.3**		**30**		**25.7**		**21.4**		**17.1**		**12.9**		**8.57**		**4.29**		**0**	
					Hours worked per Day:						**10**		**8**		**2**								**4**		**9.5**		**6**		**4**		**15**				
Sprint finished					**Hours worked total:**	**58.5**		**0**	**0**	**0**	**10**	**10**	**18**	**18**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**24**	**24**	**33.5**	**33.5**	**39.5**	**39.5**	**43.5**	**43.5**	**58.5**	**58.5**	**58.5**	**58.5**	**58.5**

Sprint 1 Burndown Chart



Sprint 1 Aufwandverteilung nach Tasks



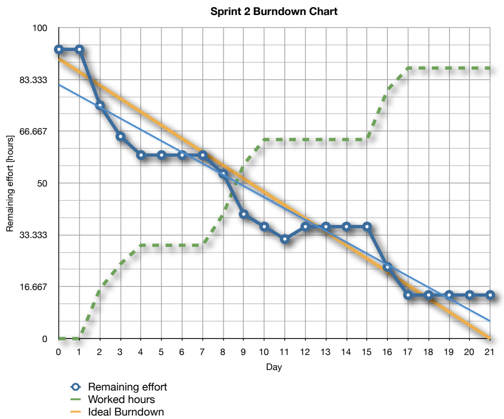
Sprint 2 Backlog

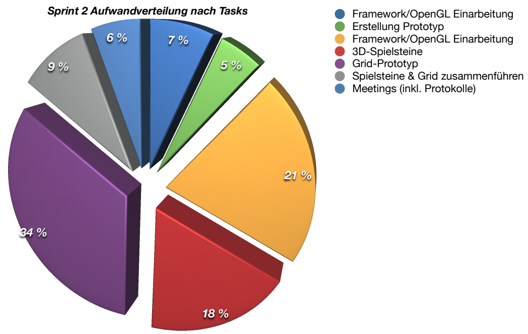
Nr.	Priority	Status		Task	Verantwortlich	Hours worked
1	High	Completed	**Analyse: Drag & Rotate**	Framework/OpenGL Einarbeitung	Team	6
2	High	Completed		Erstellung Prototyp	Mischa Trecco	4
3	Medium	Completed	**Analyse: Spielfeld**	Framework/OpenGL Einarbeitung	Team	18
4	Medium	Completed		Grid-Prototyp	Ricardo Alvarez	30
5	Medium	Completed		3D-Spielsteine	Mischa Trecco	16
6	Medium	Suspended		Spielsteine & Grid zusammenführen	Team	8
7	High	Completed	**Organisatorisches**	Meetings (inkl. Protokolle)	Team	5
					Estimated work remaining:	
Completed:	**6/7**				**Remaining working hours:**	
					Hours worked per Day:	
Sprint finished					**Hours worked total:**	**87**

Estimated hours remaining per day | Hours worked on task per day

Nr.	0	1	Mo	2	Di	3	Mi	4	Do	5	Fr	6	Sa	7	So	8	Mo	9
1	6	6		0	6													
2	4	4		0	4													
3	15	15		9	6	5	4	5		5		5		5		3	2	2
4	30	30		30		28	2	24	4	24		24		24		20	4	10
5	12	12		12		8	2	6	2	6		6		6		6	4	6
6	20	20		20		20		20		20		20		20		20		20
7	6	6		4		4		4		4		4		4		4		2
Estimated work remaining:	**93**	**93**		**75**		**65**		**59**		**59**		**59**		**59**		**53**		**40**
Remaining working hours:	**90**	**85.7**		**81.4**		**77.1**		**72.9**		**68.6**		**64.3**		**60**		**55.7**		**51.4**
Hours worked per Day:					**16**		**8**		**6**								**10**	
Hours worked total:		**0**	**0**	**0**	**16**	**16**	**24**	**24**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**40**	**40**

Nr.	Di	10	Mi	11	Do	12	Fr	13	Sa	14	So	15	Mo	16	Di	17	Mi	18	Do	19	Fr	20	Sa	21	So
1																									
2																									
3	4	2		2		2		2		2		2		0	2										
4	10	8	4	8		8		8		8		8		4	4	0	2								
5		4	4			4		4		4		4		4		0	4								
6		20		20		20		20		20		20		14	8	14		14		14		14		14	
7	2	2		2		2		2		2		2		1	2	0	1								
Estimated work remaining:		**36**		**32**		**36**		**36**		**36**		**36**		**23**		**14**		**14**		**14**		**14**		**14**	
Remaining working hours:		**47.1**		**42.9**		**38.6**		**34.3**		**30**		**25.7**		**21.4**		**17.1**		**12.9**		**8.57**		**4.29**		**0**	
Hours worked per Day:	**16**		**8**												**16**		**7**								
Hours worked total:	**56**	**56**	**64**	**64**	**64**	**64**	**64**	**64**	**64**	**64**	**64**	**64**	**64**	**64**	**80**	**80**	**87**	**87**	**87**	**87**	**87**	**87**	**87**	**87**	**87**



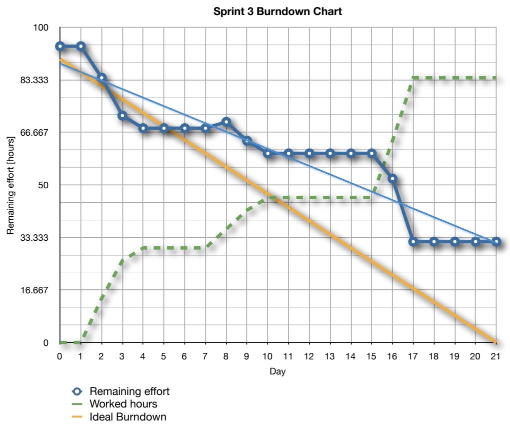


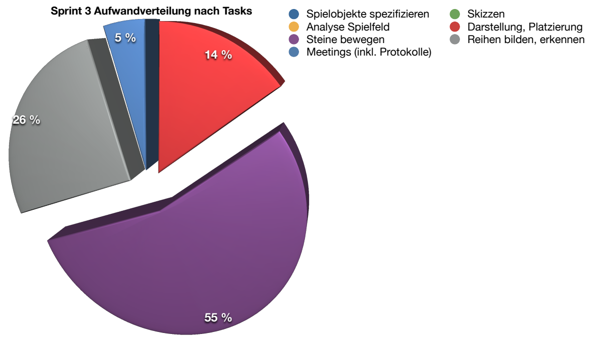
Sprint 3 Backlog

Estimated hours remaining per day | Hours worked on task per day

Nr.	Priority	Status		Task	Verantwortlich	Hours worked	0	1	Mo	2	Di	3	Mi	4	Do	5	Fr	6	Sa
1	Medium	Suspended	**Game Spezifikation**	Spielobjekte spezifizieren	Ricardo Alvarez	0	8	8		8		8		8		8		8	
2	Medium	Suspended		Skizzen	Mischa Trecco	0	2	2		2		2		2		2		2	
3	Low	Suspended	**Domainanalyse**	Analyse Spielfeld	Mischa Trecco	0	8	8		8		8		8		8		8	
4	High	Completed	**Spielfeld**	Darstellung, Platzierung	Team	12	8	8		6	6	4	2	4		4		4	
5	High	Suspended		Steine bewegen	Ricardo Alvarez	46	32	32		26	6	18	8	14	4	14		14	
6	Medium	Suspended		Reihen bilden, erkennen	Mischa Trecco	22	32	32		32		30	2	30		30		30	
7	High	Completed	**Organisatorisches**	Meetings (inkl. Protokolle)	Team	4	4	4		2	2	2		2		2		2	
					Estimated work remaining:		**94**	**94**		**84**		**72**		**68**		**68**		**68**	
Completed:		**2/7**			**Remaining working hours:**		**90**	**85.7**		**81.4**		**77.1**		**72.9**		**68.6**		**64.3**	
					Hours worked per Day:						**14**		**12**		**4**				
Sprint finished					**Hours worked total:**	**84**		**0**	**0**	**0**	**14**	**14**	**26**	**26**	**30**	**30**	**30**	**30**	**30**

Nr.	7	So	8	Mo	9	Di	10	Mi	11	Do	12	Fr	13	Sa	14	So	15	Mo	16	Di	17	Mi	18	Do	19	Fr	20	Sa	21	So
1	8		8		8		8		8		8		8		8		8		8		8		8		8		8		8	
2	2		2		2		2		2		2		2		2		2		2		2		2		2		2		2	
3	8		8		8		8		8		8		8		8		8		8		8		8		8		8		8	
4	4		4		4		4		4		4		4		4		4		4		0	4								
5	14		16	6	10	6	6	4	6		6		6		6		6		10	6	4	6	4		4		4		4	
6	30		30		30		30		30		30		30		30		30		20	10	10	10	10		10		10		10	
7	2		2		2		2		2		2		2		2		2		0	2										
Estimated work remaining:	**68**		**70**		**64**		**60**		**60**		**60**		**60**		**60**		**60**		**52**		**32**		**32**		**32**		**32**		**32**	
Remaining working hours:	**60**		**55.7**		**51.4**		**47.1**		**42.9**		**38.6**		**34.3**		**30**		**25.7**		**21.4**		**17.1**		**12.9**		**8.57**		**4.29**		**0**	
Hours worked per Day:				**6**		**6**		**4**												**18**		**20**								
Hours worked total:	**30**	**30**	**30**	**36**	**36**	**42**	**42**	**46**	**46**	**46**	**46**	**46**	**46**	**46**	**46**	**46**	**46**	**46**	**46**	**64**	**64**	**84**	**84**	**84**	**84**	**84**	**84**	**84**	**84**	**84**



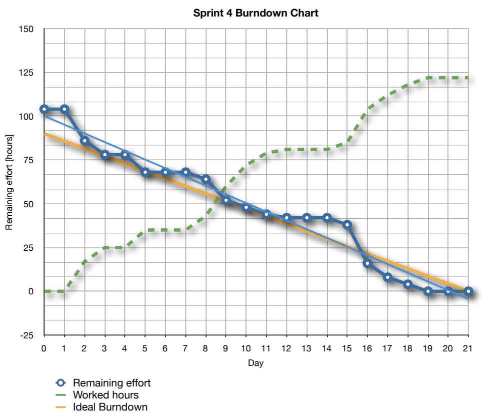


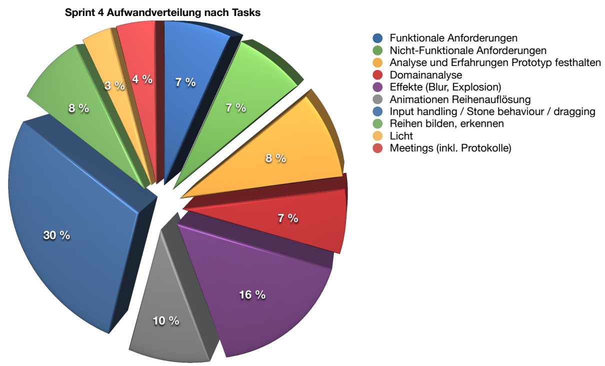
Sprint 4 Backlog

Estimated hours remaining per day | Hours worked on task per day

Nr.	Priority	Status		Task	Verantwortlich	Hours worked	0	1	Mo	2	Di	3	Mi	4	Do	5	Fr	6	Sa	7	So	8	Mo	9	Di
1	Medium	Completed	**Spezifikation**	Funktionale Anforderungen	Ricardo Alvarez	8	8	8		8		8		8		8		8		8		8		6	2
2	Medium	Completed		Nicht-Funktionale Anforderungen	Mischa Trecco	8	8	8		8		8		8		8		8		8		8		6	2
3	Low	Completed	**Entwicklungsprotokoll**	Analyse und Erfahrungen Prototyp festhalten	Ricardo Alvarez	10	6	6		6		6		6		4	2	4		4		4		2	2
4	Low	Completed	**Domainanalyse**	Domainanalyse	Mischa Trecco	8	8	8		8		8		8		8		8		8		8		8	
5	High	Completed	**Spielfeld**	Effekte (Blur, Explosion)	Ricardo Alvarez	20	20	20		20		20		20		20		20		20		20		20	
6	High	Completed		Animationen Reihenauflösung	Mischa Trecco	12	12	12		10	2	8	2	8		4	4	4		4		2	2	0	2
7	High	Completed		Input handling / Stone behaviour / dragging	Ricardo Alvarez	37	20	20		12	8	8	4	8		4	4	4		4		4	4	8	2
8	Medium	Completed		Reihen bilden, erkennen	Mischa Trecco	10	10	10		4	6	2	2	2		2		2		2		0	2		
9	Medium	Completed		Licht	Mischa Trecco	4	6	6		6		6		6		6		6		6		6		0	4
10	High	Completed	**Organisatorisches**	Meetings (inkl. Protokolle)	Team	5	6	6		4	1	4		4		4		4		4		4		2	3
					Estimated work remaining:		**104**	**104**		**86**		**78**		**78**		**68**		**68**		**68**		**64**		**52**	
Completed:		**10/10**			**Remaining working hours:**		**90**	**85.7**		**81.4**		**77.1**		**72.9**		**68.6**		**64.3**		**60**		**55.7**		**51.4**	
					Hours worked per Day:						**17**		**8**				**10**						**8**		**17**
Sprint finished					**Hours worked total:**	**122**		**0**	**0**	**0**	**17**	**17**	**25**	**25**	**25**	**25**	**35**	**35**	**35**	**35**	**35**	**35**	**43**	**43**	**60**

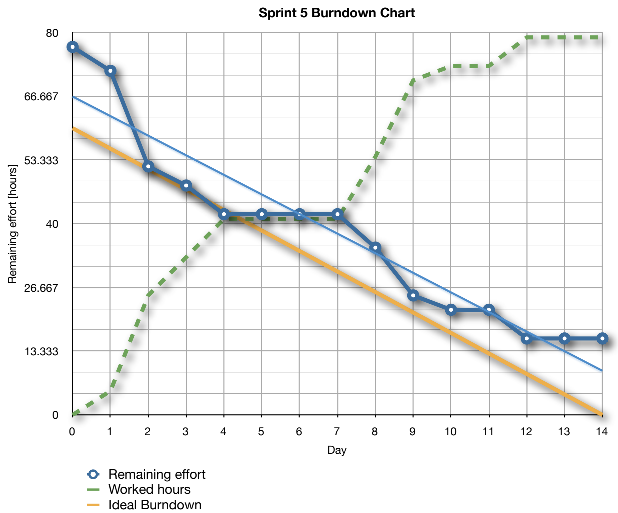
Nr.	10	Mi	11	Do	12	Fr	13	Sa	14	So	15	Mo	16	Di	17	Mi	18	Do	19	Fr	20	Sa	21	So
1	6		6		6		6		6		6		4		4		0	6						
2	6		6		6		6		6		2	4	0	2										
3	2	4	2		0	2																		
4	8		8		8		8		8		8		0	8										
5	20		20		20		20		20		20		12	8	4	8	4		0	4				
6																								
7	4	8	0	7																				
8																								
9																								
10	2		2		2		2		2		2		0	1										
Estimated work remaining:	**48**		**44**		**42**		**42**		**42**		**38**		**16**		**8**		**4**		**0**		**0**		**0**	
Remaining working hours:	**47.1**		**42.9**		**38.6**		**34.3**		**30**		**25.7**		**21.4**		**17.1**		**12.9**		**8.57**		**4.29**		**0**	
Hours worked per Day:		**12**		**7**		**2**						**4**		**19**		**8**		**6**		**4**				
Hours worked total:	**60**	**72**	**72**	**79**	**79**	**81**	**81**	**81**	**81**	**81**	**81**	**85**	**85**	**104**	**104**	**112**	**112**	**118**	**118**	**122**	**122**	**122**	**122**	**122**

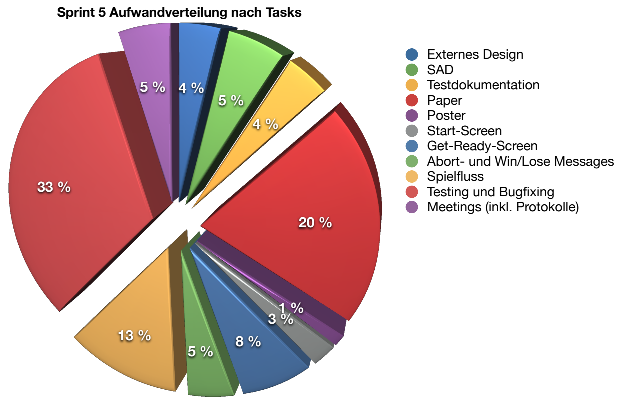




Sprint 5 Backlog

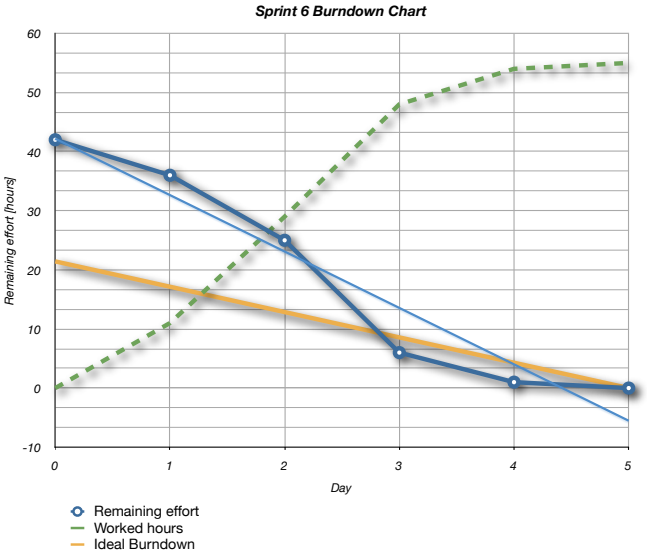
Nr.	Priority	Status		Task	Verantwortlich	Hours worked	0	1	Mo	2	Di	3	Mi	4	Do	5	Fr	6	Sa	7	So	8	Mo	9	Di	10	Mi	11	Do	12	Fr	13	Sa	14	So
							Estimated hours remaining per day | Hours worked on task per day																												
1	Medium	Completed	**Dokumentation**	Externes Design	Mischa Trecco	3	3	3		3		3		3		3		3		3		3		3		0	3								
2	Medium	Suspended		SAD	Team	4	20	20		18	2	18		18		18		18		18		16	2	16		16		16		16		16		16	
3	Medium	Completed		Testdokumentation	Ricardo Alvarez	3	3	2	1	2		2		2		2		2		2		1	1	0	1										
4	Medium	Completed		Paper	Mischa Trecco	16	16	12	4	8	4	8		8		8		8		8		4	4	2	2	2		2		0	2				
5	Medium	Completed		Poster	Ricardo Alvarez	1	1	1		1		1		1		1		1		1		1		0	1										
6	Medium	Completed	**Spiel abschliessen**	Start-Screen	Mischa Trecco	2	2	2		2		0	2																						
7	Medium	Completed		Get-Ready-Screen	Mischa Trecco	6	6	6		2	4	0	2																						
8	Medium	Completed		Abort- und Win/Lose Messages	Ricardo Alvarez	4	4	4		0	4																								
9	Medium	Completed		Spielfluss	Team	10	8	8		4	4	4		2	4	2		2		2		2		0	2										
10	Medium	Completed		Testing und Bugfixing	Team	26	10	10		10		10	4	6	4	6		6		6		6	6	4	8	4		4		0	4				
11	Medium	Completed	**Organisatorisches**	Meetings (inkl. Protokolle)	Team	4	4	4		2	2	2		2		2		2		2		2		0	2										
					Estimated work remaining:		**77**	**72**		**52**		**48**		**42**		**42**		**42**		**42**		**35**		**25**		**22**		**22**		**16**		**16**		**16**	
Completed:	**10/11**				**Remaining working hours:**		**60**	**55.7**		**51.4**		**47.1**		**42.9**		**38.6**		**34.3**		**30**		**25.7**		**21.4**		**17.1**		**12.9**		**8.57**		**4.29**		**0**	
					Hours worked per Day:				**5**		**20**		**8**		**8**								**13**		**16**		**3**				**6**				
Sprint finished					**Hours worked total:**	**79**		**0**	**5**	**5**	**25**	**25**	**33**	**33**	**41**	**41**	**41**	**41**	**41**	**41**	**41**	**41**	**54**	**54**	**70**	**70**	**73**	**73**	**73**	**73**	**79**	**79**	**79**	**79**	**79**

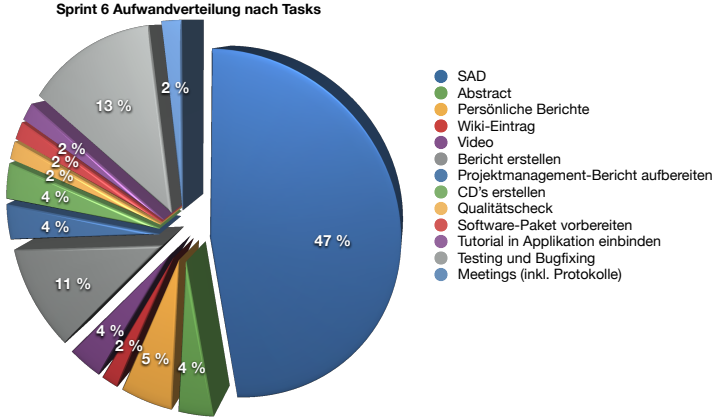




Sprint 6 Backlog

Nr.	Priority	Status		Task	Verantwortlich	Hours worked	Estimated hours remaining per day | Hours worked on task per day									
							0	1	Mo	2	Di	3	Mi	4	Do	5	Fr
1	Medium	Completed	**Dokumentation**	SAD	Mischa Trecco	26	16	14	6	6	12	0	8				
2	Medium	Completed		Abstract	Team	2	1	1		1	1	0	1				
3	Medium	Completed		Persönliche Berichte	Ricardo Alvarez	3	2	2		1	2	0	1				
4	Medium	Completed		Wiki-Eintrag	Mischa Trecco	1	1	1		1		1		1		0	1
5	Medium	Completed		Video	Ricardo Alvarez	2	2	2		2		2		0	2		
6	Medium	Completed		Bericht erstellen	Ricardo Alvarez	6	8	8		8		0	6				
	Medium	Completed		Projektmanagement-Bericht aufbereiten	Team	2	2	1	1	1		0	1				
7	Medium	Completed	**Abgabe**	CD's erstellen	Mischa Trecco	2	1	1		1		1		0	2		
8	Medium	Completed		Qualitätscheck	Mischa Trecco	1	1	1		1		1		0	1		
9	Medium	Completed		Software-Paket vorbereiten	Ricardo Alvarez	1	1	1		1		1		0	1		
10	Medium	Completed		Tutorial in Applikation einbinden	Team	1	1	1		1		0	1				
11	Medium	Completed		Testing und Bugfixing	Team	7	4	1	4	1	2	0	1				
12	Medium	Completed	**Organisatorisches**	Meetings (inkl. Protokolle)	Team	1	2	2		0	1						
					Estimated work remaining:		**42**	**36**		**25**		**6**		**1**		**0**	
Completed:		**13/13**			**Remaining working hours:**		**21.4**	**17.1**		**12.9**		**8.57**		**4.29**		**0**	
					Hours worked per Day:				**11**		**18**		**19**		**6**		**1**
Sprint finished					**Hours worked total:**	**55**		**0**	**11**	**11**	**29**	**29**	**48**	**48**	**54**	**54**	**55**





Riskmanagement

Nr.	Risiko	Beschreibung	Max. Schaden	Max. Schaden [h]	Eintritts-wahrscheinlichkeit	Gewichteter Schaden	Massnahmen zur Vermeidung / Verminderung	Vorgehen bei Eintreffen	Referenzen
1	Berührungserkennung von Objekten auf dem Tisch zu ungenau.	Der Tisch erkennt das Agieren mit immer kleineren Objekten immer ungenauer. Für das Spielprinzip “Tetris” muss das Spielfeld aber genügend Spielsteine zulassen, d.h. die Spielsteine dürfen nicht zu gross sein. Ist die kleinst-akzeptable Grösse der Spielsteine, die vom Tisch noch erkannt wird, zu gross um genügend Spielsteine auf dem Spielfeld darstellen zu können, macht das Spielprinzip “Tetris” keinen Sinn. Agieren bezieht sich auf die User-Aktionen: Skalieren und Rotieren.	Zeitverlust Sprint 2 + neue Spielidee suchen, planen 3 Wochen (3*30h)	90	30.00 %	27	Entwicklung Prototyp 1 in Sprint 2.	Neue Spielidee suchen	
2	Reaktionszeit Tisch zu langsam.	Das Spielprinzip “Tetris” setzt voraus, dass ein gewisses Spieltempo entstehen kann. Bei der Projektidee erscheinen die Steine zwar nicht immer schneller hintereinander wie bei der klassischen Version, ein gewisses Spieltempo muss aber dennoch spielbar sein. Kann dieses Mindest-Spieltempo nicht vom Tisch verarbeitet werden, macht das Spiel keinen Spass und somit keinen Sinn.	Zeitverlust Sprint 2 + Sprint 3 + neue Spielidee suchen, planen 5 Wochen (5*30h)	150	30.00 %	45	Bezüglich Greifgeschwindigkeit: Entwicklung Prototyp 1 in Sprint 2. Bezüglich Ziehgeschwindigkeit: Entwicklung Prototyp 2 in Sprint 3.	Neue Spielidee suchen	
3	Gleichzeitige Berührungen an selber Stelle nicht erkennbar.	Der Tisch kann gleichzeitige Berührungen von mehreren Personen (bzw. Fingern) an selber Oberflächen-Position (nebeneinander) nicht erkennen bzw. erkennt sie als eine einzige Berührung. Dies würde es unmöglich machen einander Spielsteine wegzunehmen also um die Spielsteine zu kämpfen was eine Grundidee des Spiels ist.	Zeitverlust Spielidee abändern 1 Woche (3*30h)	30	30.00 %	9	Entwicklung Prototyp 3 in Sprint 4.	Spielidee abändern	
				270		**81**			

Bug List

Nr.	Bug	Priority	Severity	Status	Resolution	Detected in Sprint	Assigned to Sprint	Beschreibung	Kommentar
1	Tetrissteine teilweise fixiert	Normal	Major	Resolved	Fixed	3	3	Steine lassen sich am unteren Ende des Grids nicht aufnehmen und mehr bewegen	
2	Licht erscheint verzögert	Normal	Major	Resolved	Fixed	4	4	Das Licht wird erst nach erscheinen des ersten Steines gerendert	
3	Ziehen von Steinen nicht realistisch	High	Major	Resolved	Fixed	4	4	Der Stein ist langsamer als die Bewegung, er befindet sich dadurch nicht immer auf der selben Position wie der Finger	Lösungsansatz: Schnittpunkt: PlayerGrid-Ebene = Gerade parallel zur z-Achse
4	Animationen überschreiben sich	High	Major	Resolved	Fixed	4	4	Wenn gleichzeitig mehrere Animatoren laufen die Steine nachrücken bzw. raufschieben kommen sich diese in die Quere und es enstehen komische Animationen.	Lösungsansatz: Wenn ein neuer Animator starten soll wird stattdessen der alte zurückgesetzt, neu initialisiert und wieder gestartet. Er muss dabei die aktuellen Positionen der Steine berücksichtigen.
5	Stein springt teilweise bei aufheben	Normal	Normal	Resolved	Fixed	5	5	Bei Newstroke auf Playergrid springt Tetrisstein auf eine andere Position.	Lösungsansatz: Ein Vektor wird im Charger verändert.

13 Glossar

BaseStone
Grundbaustein eine Tetris-Steins.

Battlezone
Spielfeldbereich zwischen den Playergrids, wo die Steine erscheinen. Beide Spieler können sich die Tetris-Steine in diesem Bereich holen.

Drag & Drop
Ein Objekt auf dem Tisch mit der Hand ziehen und am gewünschten Ort platzieren.

Game
Das Spiel.

Multi-Touch Plattform
Damit sind Eingabegeräte gemeint, welche mehrere Inputpunkte gleichzeitig wahrnehmen können. Sie werden meist mit den Fingern bedient.

Playergrid
Das persönliche Spielfeld, wo die Tetris-Steine zu reihen kombiniert werden müssen.

Tetris-Stein
Die Spielsteine von T-Touch.

to-fuse
Industriepartner des Projekts T-Touch.
Ansprechsperson ist Christian Iten.
Weiterer Mitarbeiter ist Emanuel Zgraggen.

to-fuse Application Framework
Das Framework, welches zum Betrieb des Multi-Touch Tisches benötigt wird. Es bildet die Grundlage einer Entwicklung für die *to-fuse Multi-Touch Platform* .

to-fuse Multi-Touch Platform
Die von to-fuse entwickelte Multi-Touch Plattform.

T-Touch
Der Name des Spiels.

Literaturverzeichnis

[1] F.S. Hill, JR. und Stephen M. Kelley, *Computer Graphics using OpenGL*, 3rd ed. Upper Saddle River, USA: Pearson Prentice Hall, 2006.

[2] R. Khaled, P. Barr, H. Johnston, R. Biddle, *Let's Clean Up This Mess: Exploring Multi-Touch Collaborative Play.* In CHI '09, ACM Press 978-1-60558-247-4/09/04.

[3] S. Hsiao-Heng Chang, L. Stuart, B. Plimmer, B. Wünsche, *Origami Simulator: a Multi-Touch Experience.* In CHI '09, ACM Press 978-1-60558-247-4/09/04.

[4] *to-fuse* http://www.to-fuse.com

[5] *HSR* http://www.hsr.ch

[6] *Carbonated Games* http://www.carbonatedgames.com

[7] *Microsoft Surface* http://www.microsoft.com/surface/

[8] *Firefly-Review* http://sarcasticgamer.com/wp/index.php/2008/02/firefly-the-first-game-for-microsoft-surface.html/1