**Tetris Game Dart**



Dokumentation für das Modul “Webtechnologie Projekt”

Florian Jeger

Christoph Jeger

Matthias Steffen

SoSe 2017

**Inhaltsverzeichnis**

[**1.** **Einleitung** 4](#_Toc482194342)

[**2.** **Anforderungen und abgeleitetes Spielkonzept** 5](#_Toc482194343)

[2.1 Anforderungen 5](#_Toc482194344)

[2.2 Spielkonzept des Tetris Games 7](#_Toc482194345)

[**3.** **Architektur und Implementierung** 8](#_Toc482194346)

[3.1 Modell 8](#_Toc482194347)

[3.2 View 8](#_Toc482194348)

[3.2.1 HTML-Dokument 8](#_Toc482194349)

[3.2.2 TetrisView als Schnittstelle zum HTML-Dokument 8](#_Toc482194350)

[3.3 Controller 8](#_Toc482194351)

[**4.** **Level- und Parametrisierungskonzept** 9](#_Toc482194352)

[**5.** **Nachweis der Anforderungen** 10](#_Toc482194353)

[5.1 Nachweis der funktionalen Anforderungen 10](#_Toc482194354)

[5.2 Nachweis der Dokumentationsanforderungen 10](#_Toc482194355)

[5.3 Nachweis der Einhaltung technischer Randbedingungen 11](#_Toc482194356)

[5.4 Verantwortlichkeiten im Projekt 12](#_Toc482194357)

**Abbildungsverzeichnis**

[Abbildung 1: Die Tetris-Bausteine I, J, L, O, S, T und Z 7](#_Toc479859577)

[Abbildung 2: Spielprinzip von Tetris Game 7](#_Toc479859578)

**Tabellenverzeichnis**

[Tabelle 1: Anforderungen 6](#_Toc482102586)

[Tabelle 2: Punktesystem für Tetris 9](#_Toc482102587)

[Tabelle 3: Nachweis der funktionalen Anforderungen 10](#_Toc482102588)

[Tabelle 4: Nachweis der Dokumentationsanforderungen 10](#_Toc482102589)

[Tabelle 5: Nachweis der technischen Randbedingungen 11](#_Toc482102590)

[Tabelle 6: Projektverantwortlichkeiten 12](#_Toc482102591)

**Programm-Listings**

[Listing 1: HTML Basisdokument des Spiels 9](#_Toc486499552)

# **Einleitung**

Diese Dokumentation erläutert den Aufbau des Tetris-Games. Im Kapitel 2, ist das Spielkonzept und die zu erfüllenden Anforderungen zu finden. Die Umsetzung des Spielkonzepts basiert auf eine Model-View-Controller Architektur und wird im Kapitel 3 dargestellt. Das Kapitel 4 befasst sich mit dem Spiellevel und den zu definierenden Spielparametern. Abschließend in Kapitel 5 geht es um den Nachweis der Anforderungen.

# **Anforderungen und abgeleitetes Spielkonzept**

## **Anforderungen**

Das Tetris Game soll folgende in Tabelle 1 aufgeführten funktionalen Anforderungen,

Dokumentationsanforderungen und technischen Randbedingungen erfüllen.

|  |  |  |
| --- | --- | --- |
| **Id** | **Kurztitel** | **Anforderung** |
| AF-1 | Einplayer  Game | Das Spiel soll ein Einplayer Game sein (Mehrplayer Konzepte können als Einplayer Game realisiert werden, wenn Spieler durch “künstliche Intelligenzen” gesteuert werden. Beachten Sie dabei bitte, dass abhängig vom Spielkonzept die Komplexität des Spiels erheblich steigt, denken sie bspw. an Schach. Es bietet sich an, sich von alten Arcade Klassikern inspieren zu lassen.) |
| AF-2 | 2D Game | Das Spiel soll konzeptionell auf einem 2D-Raster basieren. |
| AF-3 | Levelkonzept | Das Spiel sollte ein Levelkonzept vorsehen. |
| AF-4 | Parametrisierungs-  konzept | Das Spiel sollte ein Parameterisierungskonzept für relevante Spielparameter vorsehen. |
| AF-5 | Mobile Browser | Das Spiel muss auf SmartPhone Browsern spielbar sein. |
| **Dokumentationsanforderungen** | | |
| D-1 | Dokumentations-  vorlage | Die Dokumentation soll sich an vorliegender Vorlage orientieren. |
| D-2 | Projekt-dokumentation | Das Spiel muss geeignet dokumentiert sein, so dass es von projektfremden Personen fortgeführt werden könnte. |
| D-3 | Quelltext-dokumentation | Der Quelltext des Spiels muss geeignet dokumentiert sein und mittels schriftlicher Dokumentation erschließbar und  verständlich sein. |
| D-4 | Libraries | Alle verwendeten Libraries sind aufzuführen und deren Notwendigkeit zu begründen. |
| **Technische Randbedingungen** | | |
| TF-1 | No Canvas | Die Darstellung des Spielfeldes sollte ausschließlich mittels DOM-Tree Techniken erfolgen. Die Nutzung von  Canvas-basierten Darstellungstechniken ist **explizit** untersagt. |
| TF-2 | Levelformat | Level sollten sich mittels deskriptiver Textdateien definieren lassen (z.B. mittels CSV, JSON, XML, etc.), so dass  Level-Änderungen ohne Sourcecode-Änderungen des Spiels realisierbar sind. |
| TF-3 | Parameterformat | Spielparameter sollten sich mittels deskriptiver Textdateien definieren lassen (z.B. mittels CSV, JSON, XML, etc.), so dass Parameter-Änderungen ohne Sourcecode-Änderungen des Spiels realisierbar sind. |
| TF-4 | HTML + CSS | Der View des Spiels darf ausschließlich mittels HTML und CSS realisiert werden. |
| TF-5 | Gamelogic in  Dart | Die Logik des Spiels muss mittels der Programmiersprache Dart realisiert werden. |
| TF-6 | Browser  Support | Das Spiel muss im Browser Chromium/Dartium (native Dart Engine) funktionieren. Das Spiel muss ferner in allen anderen Browserrn (JavaScript Engines) ebenfalls in der JavaScript kompilierten Form funktionieren (geprüft wird ggf. mit Safari, Chrome und Firefox). |
| TF-7 | MVC  Architektur | Das Spiel sollte einer MVC-Architektur folgen. |
| TF-8 | Erlaubte  Pakete | Erlaubt sind alle dart:\* packages, sowie das Webframework start. |
| TF-9 | Verbotene  Pakete | Verboten sind Libraries wie Polymer oder Angular. (Sollten Sie Pakete verwenden wollen, die außerhalb der erlaubten  Pakete liegen, holen Sie sich das Go ab, begründen sie bitte, wieso sie das Paket benötigen). |
| TF-10 | No Sound | Das Spiel muss keine Sounds unterstützen. |

Tabelle 1: Anforderungen

## **Spielkonzept des Tetris Games**



Abbildung 1: Die Tetris-Bausteine I, J, L, O, S, T und Z

In Abbildung 1 sind die Tetris-Bausteine zu sehen, die für das Spiel vorgesehen sind. Die Tetris-Bausteine werden zufällig generiert, d.h. es gibt eine 1/7-Wahrscheinlichkeit, das z.B. der I-Baustein vorkommt.



Abbildung 2: Spielprinzip von Tetris Game

Abbildung 2 zeigt das Spielprinzip des Tetris-Games. Der Spieler muss vom oberen Rand des rechteckigen Spielfeldes herunterfallende Tetris-Bausteine so platzieren, dass sie am unteren Rand horizontal möglichst lückenlose Reihen bilden. Dazu kann der Spieler die Tetrominoes (Tetris-Bausteine) in 90-Grad-Schritten drehen und oder sie links und rechts verschieben. Sobald eine Reihe komplett ist, verschwindet diese. Alle Reihen, die darüber liegen, rücken nach unten und geben einen Teil des Spielfeldes wieder frei. Sollte es dem Spieler gelingen gleichzeitig mehrere Reihen zu tilgen, erhält der Spieler eine höhere Punktezahl pro Reihe, als für das Tilgen einer einzelnen Reihe. Wenn eine bestimmte Anzahl entfernter Reihen erreicht worden ist, steigt der Spieler im Level auf und die Fallgeschwindigkeit der Tetrominoes wird erhöht. Das Spiel endet, sobald sich die nicht abgebauten Reihen, also jene mit Lücken, bis zum oberen Spielfeldrand aufgetürmt haben.

# **Architektur und Implementierung**

Die Architektur von Tetris Game folgt dem Model-View-Controller Prinzip. Softwaretechnisch gliedert sich die Spiellogik in mehrere Komponenten (Klassen) mit spezifischen funktionalen Verantwortlichkeiten. Eine zentrale Rolle für die Spielsteuerung hat der Controller (Klasse TetrisController).

Der Controller kann

• Nutzerinteraktionen (insbesondere Betätigen von Buttons) sowie

• Zeitsteuerung (fallen des Tetrominoes)

erkennen und in entsprechende Modelinteraktionen umsetzen.

Der Controller wird in Abschnitt 3.3, die View in Abschnitt 3.2 und das Model in Abschnitt 3.3 erläutert.

## **Modell**

Aus dem Spielkonzept des Abschnitts 2.2 wurden Tetrominoes abgeleitet. Ein Spiel (Tetris Game) besteht dabei aus mindestens einem Tetromino. Das Klassendiagramm des Models wird in Abbildung Klassendiagram Nummer vom Model gezeigt.

## **View**

Die View ist für die Darstellung des Spiels zuständig. Im Kern besteht die View aus einem HTML-Dokument (siehe Abschnitt 3.2.1) und einer clientseitigen Logik, die den DOM-Tree des HTML-Dokuments manipuliert (siehe Abschnitt 3.2.2).

### **HTML-Dokument**

Die View wird im Browser durch folgendes HTML-Dokument erzeugt. Der DOM-Tree dieses HTML-Dokuments wird im Verlaufe des Spiels durch die Klasse TetrisView manipuliert (siehe Abbildung **NUMMER DER KLASSENDIAGRAM ABBILDUNG HIER EINFÜGEN**), um den Spielzustand darzustellen und Nutzerinteraktionen zu ermöglichen. Die Klasse TetrisView wird dabei durch das Script tetrisclient.dart als clientseitige Logik geladen.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88 | <!DOCTYPE html> <**html** lang=**"de"**> <**head**>  <**meta** charset=**"utf-8"**>  <**meta** name=**"viewport"** content=**"user-scalable=no, width=device-width, initial-scale=1.0"**>  <**title**>Tetris</**title**>  <**link** href=**"favicon.ico"** rel=**"shortcut icon"** type=**"images/x-icon"**>  <**link** rel=**"stylesheet"** type=**"text/css"** href=**"styles.css"**> </**head**> <**body**> *<!-- Container für die Startseite -->* <**div** class=**"container\_start"**>  <**div** id=**"title"**></**div**>  <**p** id=**"start"**>START</**p**>  <**p** id=**"copy"**>**&copy;** Florian Jeger, Christoph Jeger **&amp;** Matthias Steffen</**p**> </**div**>  <**div** id=**"overlay"**> </**div**>  *<!-- Container für die Nachrichten -->* <**div** class=**"container\_message"**>  <**img** id=**"logo"** src=**'img/tetris\_menu\_logo.png'**>  <**img** id=**"portrait"** src=**'img/portrait.png'**>  <**p** id=**"message"**></**p**>  <**p** id=**"continue"**>FORTSETZEN</**p**>  <**p** id=**"newGame"**>NEUES SPIEL</**p**> </**div**>  <**div** class=**"container\_game"**>  *<!-- Container für die Level Ziele -->* <**div** class=**"container\_goal"**>  <**b**>Ziel:</**b**> <**span** id=**"goalDescription"**></**span**> <**span** id=**"goalProgress"**>0</**span**>/<**span** id=**"goal"**>0</**span**>  <**br**/>  <**b**>Belonung: </**b**> <**span** id=**"bonusPoints"**>0</**span**> Punkte  </**div**>   *<!-- Tabelle für das Spielfeld -->* <**table** id=**"field"**>  </**table**>   <**div** class=**"container\_sidebox"**>  *<!-- Container für die Seitenbox -->* <**div** class=**"container\_nextstone"**>  <**h3**>Nächster Stein</**h3**>  *<!-- Tabelle für den nächsten Stein -->* <**table** id=**"nextstone"**>  </**table**>  </**div**>   <**div** class=**"container\_holdstone"**>  <**h3**>Stein halten</**h3**>  *<!-- Tabelle für den nächsten Stein -->* <**table** id=**"holdstone"**>  </**table**>  </**div**>   *<!-- Container für die Spielinformationen -->* <**div** class=**"container\_info"**>  <**h3**>Level</**h3**>  <**p** id=**"level"**>0</**p**>  <**h3**>Punkte</**h3**>  <**p** id=**"points"**>0</**p**>  </**div**>  </**div**>    *<!-- Container für die Steuerung -->* <**div** class=**"container\_control"**>  <**div** id=**"menu"**><**img** src=**'img/menu.png'**></**div**>  <**div** id=**"hard\_drop"**><**img** src=**'img/hard\_drop.png'**></**div**>  <**div** id=**"left\_rotation"**><**img** src=**'img/left\_rotation.png'**></**div**>  <**div** id=**"right\_rotation"**><**img** src=**'img/right\_rotation.png'**></**div**>  <**div** id=**"hold"**><**img** src=**'img/hold.png'**></**div**>  <**div** id=**"left"**><**img** src=**'img/left.png'**></**div**>  <**div** id=**"down"**><**img** src=**'img/down.png'**></**div**>  <**div** id=**"right"**><**img** src=**'img/right.png'**></**div**>  </**div**>   <**script** type=**"application/dart"** src=**"tetrisclient.dart"**></**script**>  <**script** data-pub-inline src=**"packages/browser/dart.js"**></**script**> </**div**> </**body**> </**html**> |

Listing 1: HTML Basisdokument des Spiels

Um das Spiel einzublenden wird diese HTML-Dokument genutzt.

### **TetrisView als Schnittstelle zum HTML-Dokument**

Alle notwendigen CSS-Gestaltungen werden in der style.css vorgenommen. Im tetrisclient.dart Script wird die Applikationslogik geladen. Für Browser die nicht Dart-fähig sind, wird gemäß den Dart-Konventionen die dart.js geladen, damit wird die Dart Logic in Javascript Engines zu Ausführung gebracht.

## **Controller**

Für die Ablaufsteuerung des Spiels ist der Controller zuständig.

# **Level- und Parametrisierungskonzept**

Die folgende Tabelle zeigt wie unsere Punkte berechnet werden:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Level** | **Punkte für 1 Reihe** | **Punkte für 2 Reihen** | **Punkte für 3 Reihen** | **Punkte für 4 Reihen** |
| 1 | 40 | 100 | 300 | 1200 |
| 2 | 80 | 200 | 600 | 2400 |
| 3 | 120 | 300 | 900 | 3600 |
| … | … | … | … | … |
| 10 | 400 | 900 | 3000 | 12000 |
| n | 40\*(n+1) | 100\*(n+1) | 300\*(n+1) | 1200\*(n+1) |

Tabelle 2: Punktesystem für Tetris

Des Weiteren bekommt der Spieler pro erreichtes Level Bonuspunkte. Der Spieler bekommt beim Abschluss von Level 1 1000 Bonuspunkte und für jedes weitere Level werden 250 Punkte aufaddiert. Somit bekommt der Spieler beim Abschluss von Level 2 1250 Punkte und für Level 3 1500 extra Punkte und so weiter.

Um ein Level aufzusteigen muss der Spieler bestimmte Ziele erfüllen. Hierbei gibt es zwei verschiedene Zielen die erfüllt werden müssen. Hierbei handelt es sich um eine bestimmte Anzahl von Reihen tilgen oder es muss eine bestimmte Anzahl an Tetrominoes gespielt werden.

Die Bonuspunkte und die Ziele zum Erreichen eines Levels werden im Spiel angezeigt.

Die Fallgeschwindigkeit erhöht sich bei jedem Level, damit das Spiel schwerer wird. Die Start Fallgeschwindigkeit liegt bei 1000ms, d.h. jede Sekunde fallen die Tetrominoes um eine Einheit. Jedes Level reduziert sich die Fallgeschwindigkeit um 200ms und die Tetrominoes fallen dementsprechend schneller.

Alle Parameter lassen sich in der game-config.json Datei anpassen bzw. modifizieren.

# **Nachweis der Anforderungen**

Nachfolgend wird erklärt wie die im Kapitel 2 aufgeführten funktionalen Anforderungen eingehalten bzw. erfüllt werden. Dies wird nachfolgend argumentativ erfolgen und wird nicht durch Testfall-getriebene Nachweisführung erfolgen. Abschließend wird angegeben, wer im Team welche Verantwortlichkeiten hatte.

## **Nachweis der funktionalen Anforderungen**

Nachfolgend erfolgt der Nachweis der Einhaltung funktionaler Anforderungen.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Id** | **Kurztitel** | **Erfüllt** | **Teilw. erfüllt** | **Nicht erfüllt** | **Erläuterung** |
| AF-1 | Einplayer Game | x |  |  | Das Tetris Game ist ein Einpersonen Spiel, wie aus dem in Abschnitt 2.2 dargestellten Spielkonzepts hervorgeht. |
| AF-2 | 2D Game | x |  |  | Das Tetris Game wird auf einem 2D-Raster gespielt, wie aus dem in Abschnitt 2.2 dargestellten Spielkonzepts hervorgeht. |
| AF-3 | Levelkonzept |  |  |  |  |
| AF-4 | Parametrisierungs-  konzept |  |  |  |  |
| AF-5 | Mobile Browser | x |  |  | Das Spiel ist in Mobile Browsern spielbar. Es wurde in Chrome, Safari und Firefox erfolgreich zur Ausführung gebracht. |

Tabelle 3: Nachweis der funktionalen Anforderungen

## **Nachweis der Dokumentationsanforderungen**

Nachfolgend erfolgt der Nachweis der Einhaltung der Dokumentationsanforderungen.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Id** | **Kurztitel** | **Erfüllt** | **Teilw. erfüllt** | **Nicht erfüllt** | **Erläuterung** |
| D-1 | Dokumentations-vorlage | x |  |  | Vorliegende Dokumentation diente als Vorlage für Spieldokumentationen. |
| D-2 | Projektdokumentation | x |  |  | Vorliegende Dokumentation erläutert die übergeordneten Prinzipien und verweist an geeigneten Stellen auf die Quelltextdokumentation. |
| D-3 | Quelltext-dokumentation | x |  |  | Es wurden alle Methoden und Datenfelder, Konstanten durch Inline-Kommentare erläutert. |
| D-4 | Libraries | x |  |  | Alle genutzten Libraries werden in der pubspec.yaml der Implementierung aufgeführt. Da nur die zugelassenen Pakete genutzt wurden, sind darüber hinaus keine weiteren Erläuterungen, warum welche Pakete genutzt wurden, erforderlich. |

Tabelle 4: Nachweis der Dokumentationsanforderungen

## **Nachweis der Einhaltung technischer Randbedingungen**

Nachfolgend erfolgt der Nachweis der Einhaltung der vorgegebenen technischen Randbedingungen.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Id** | **Kurztitel** | **Erfüllt** | **Teilw. erfüllt** | **Nicht erfüllt** | **Erläuterung** |
| TF-1 | No Canvas | x |  |  | Die Klasse TetrisView nutzt keinerlei Canvas basierten DOM-Elemente |
| TF-2 | Levelformat |  |  |  |  |
| TF-3 | Parameterformat |  |  |  |  |
| TF-4 | HTML + CSS | x |  |  | Die View des Spiels beruht ausschließlich auf HTML und CSS |
| TF-5 | Gamelogic in Dart | x |  |  | Die Logik des Spiels ist in der Programmiersprache Dart realisiert worden. |
| TF-6 | Browser Support | x |  |  | Das Spiel funktioniert im Browser Chromium/Dartium. Das Spiel muss ferner in allen anderen Browsern funktionieren. Geprüft wurden Safari, Chrome und Firefox. |
| TF-7 | MVC Architektur | x |  |  | Das Spiel folgt durch Ableitung mehrerer Modell-Klassen, einer View Klasse und dem zentralen Controller einer MVC-Architektur. Der Controller triggert das Modell und die View. Die View greift zudem auf das Model nur lesend und nicht manipulierend zu. |
| TF-8 | Erlaubte Pakete | x |  |  | Es sind nur dart:\* packages, sowie das Webframework start genutzt worden. Siehe pubspec.yaml der Implementierung. |
| TF-9 | Verbotene Pakete | x |  |  | Es sind keine Pakete, außer den erlaubten genutzt worden. Siehe pubspec.yaml der  Implementierung. |
| TF-10 | No Sound | x |  |  | Das Spiel hat keine Soundeffekte. |

Tabelle 5: Nachweis der technischen Randbedingungen

## **Verantwortlichkeiten im Projekt**

Nachfolgend erfolgt eine Übersicht über die Aufgabenteilung.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Komponente** | **Detail** | **Asset** | **Florian Jeger** | **Christoph Jeger** | **Matthias Steffen** | **Anmerkungen** |
| Modell | Spielfeld | /lib/src/model/  Cell.dart |  |  |  |  |
|  | Tetris Spiel | /lib/src/model/  TetrisGame.dart |  |  |  |  |
|  | Tetromino | /lib/src/model/  Tetromino.dart |  |  |  |  |
| View | HTML-Dokument | web/index.html |  |  |  |  |
|  | Gestaltung | web/styles.css |  |  |  |  |
|  | Bilder | web/img/\* |  |  |  |  |
|  | Viewlogik | /lib/src/view/  TetrisView.dart |  |  |  |  |
| Controller | Eventhandling | lib/src/controller/  TetrisController.dart |  |  |  |  |
|  | Parametrisierung |  |  |  |  |  |
|  | Level |  |  |  |  |  |
| Dokumentation | Tetris- Documentation | doc/\*.\* |  |  |  |  |

Tabelle 6: Projektverantwortlichkeiten

V = verantwortlich (hauptdurchführend, kann nur einmal pro Zeile vergeben werden)

U = unterstützend (Übernahme von Teilaufgaben)