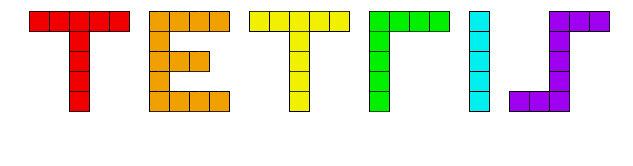
**Tetris Game Dart**



Dokumentation für das Modul “Webtechnologie Projekt”

Florian Jeger

Christoph Jeger

Matthias Steffen

SoSe 2017

**Inhaltsverzeichnis**

[**1.** **Einleitung** 4](#_Toc481400513)

[**2.** **Anforderungen und abgeleitetes Spielkonzept** 5](#_Toc481400514)

[3.1 Anforderungen 5](#_Toc481400515)

[3.2 Spielkonzept des Tetris Games 7](#_Toc481400516)

[**3.** **Architektur und Implementierung** 8](#_Toc481400517)

[3.1 Modell 8](#_Toc481400518)

[3.2 View 8](#_Toc481400519)

[3.2.1 HTML-Dokument 8](#_Toc481400520)

[3.2.2 TetrisView als Schnittstelle zum HTML-Dokument 8](#_Toc481400521)

[3.3 Controller 8](#_Toc481400522)

[**4.** **Level- und Parametrisierungskonzept** 9](#_Toc481400523)

[**5.** **Nachweis der Anforderungen** 10](#_Toc481400524)

**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](#_Toc480728339)

[Tabelle 2: Punktesystem für Tetris 9](#_Toc480728340)

# **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 | REST-basierter  Storage | Das Spiel soll auf einen JSON-basierte Storageservice zugreifen können, um Spielzustände (z.B. Highscores und ähnliches) speichern zu können. Das Spiel muss bei nicht erreichbarem Storage-Service (Gamekey Service) weiterhin spielbar sein. Einzig und allein die Speicherfunktionalitäten dürfen natürlich eingeschränkt sein. (Für den Storage-Service  (Gamekey-Service) gibt es eine Referenz-Implementierung: https://bitbucket.org/nanekratzke/gamekey) |
| AF-6 | Desktop Browser | Das Spiel muss in Desktop Browsern spielbar sein. |
| AF-7 | 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 | Storagelogic in  Dart | Der Storageservice muss REST-basiert sein und mittels der Programmiersprache Dart realisiert werden. |
| TF-7 | Storage Referenz | Das Spiel soll gleichermaßen mit der Referenzimplementierung des Storageservice als auch der Eigenrealisierung des Storageservice funktionieren. |
| TF-8 | Storage  Referenztest | Die Eigenrealisierung des Storageservices muss den Referenztest der Referenzimplementierung implementieren. (Nicht erfolgreiche Testfälle sind explizit auszuweisen, zu dokumentieren und zu begründen!) |
| TF-9 | 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-10 | MVC  Architektur | Das Spiel sollte einer MVC-Architektur folgen. |
| TF-11 | Erlaubte  Pakete | Erlaubt sind alle dart:\* packages, sowie das Webframework start. |
| TF-12 | 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-13 | 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 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 Tetris-Bausteine wird erhöht. Das Spiel endet, sobald sich die nicht abgebauten Reihen, also jene mit Lücken, bis zum oberen Spielfeldrand aufgetürmt haben.

# **Architektur und Implementierung**

## **Modell**

## **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.

**HTML CODE HIER EINFÜGEN**

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** |
| 0 | 40 | 100 | 300 | 1200 |
| 1 | 80 | 200 | 600 | 2400 |
| 2 | 120 | 300 | 900 | 3600 |
| … | … | … | … | … |
| 9 | 400 | 900 | 3000 | 12000 |
| n | 40\*(n+1) | 100\*(n+1) | 300\*(n+1) | 1200\*(n+1) |

Tabelle 2: Punktesystem für Tetris

Nach 10 Reihen, die der Spieler gelöscht hat, gelangt er ins nächste Level, dies erhöht die Fallgeschwindigkeit der Steine. Die Fallgeschwindigkeit erhöht sich um den Faktor 1,3 pro Level. Jedes Level erhöht auch die Punkte, die der Spieler erhalten kann, siehe Tabelle 2.

# **Nachweis der Anforderungen**