

Path Puzzle

Martin Dolacinski und Benjamin Tamke

Inhaltsverzeichnis

[1. Einleitung 3](#_Toc513998693)

[2. Spielkonzept 5](#_Toc513998694)

[2.1. Anforderung 5](#_Toc513998695)

# Einleitung

# Spielkonzept

## Anforderung

Ihr Spiel soll folgende Anforderungen erfüllen. Sie sollen dabei nachvollziehbar dokumentieren, wie diese Anforderungen durch ihre Realisierung umgesetzt werden. Das Spiel soll folgende in Tabelle [1](#page6) aufgeführten funktionalen Anforderungen, Dokumentationsanforderungen und technischen Randbedingungen erfüllen.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | |
|  |  |  |  |  |  |
|  | Id | Kurztitel | Anforderung | | |
|  |  |  |  |  |  |
|  | AF-1 | Einplayer | Das Spiel soll ein Einplayer Game sein (Mehrplayer Konzepte können als Einplayer Game realisiert werden, wenn Spieler | | |
|  |  | Game | 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 | Parametrisier- | Das Spiel sollte ein Parameterisierungskonzept für relevante Spielparameter vorsehen. | | |
|  |  | ungskonzept |  |  |  |
|  |  |  |  |  |  |
|  | AF-5 | REST-basierter | Das Spiel soll auf einen JSON-basierte Storageservice zugreifen können, um Spielzustände (z.B. Highscores und ähnliches) | | |
|  |  | Storage | 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 | Das Spiel muss in Desktop Browsern spielbar sein. | | |
|  |  | Browser |  |  |  |
|  |  |  |  |  |  |
|  | AF-7 | Mobile Browser | Das Spiel muss auf SmartPhone Browsern spielbar sein. | | |
|  |  |  |  |  |  |
|  |  |  | Dokumentationsanforderungen | | |
|  |  |  |  |  |  |
|  | D-1 | Dokumenta- | Die Dokumentation soll sich an vorliegender Vorlage orientieren. | | |
|  |  | tionsvorlage |  |  |  |
|  |  |  |  |  |  |
|  | D-2 | Projekt- | Das Spiel muss geeignet dokumentiert sein, so dass es von projektfremden Personen fortgeführt werden könnte. | | |
|  |  | dokumentation |  |  |  |
|  |  |  |  |  |  |
|  | D-3 | Quelltext- | Der Quelltext des Spiels muss geeignet dokumentiert sein und mittels schriftlicher Dokumentation erschließbar und | | |
|  |  | dokumentation | 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 | Die Logik des Spiels muss mittels der Programmiersprache Dart realisiert werden. | | |
|  |  | Dart |  |  |  |
|  |  |  |  | | |
|  | TF-6 | Storagelogic in | Der Storageservice muss REST-basiert sein und mittels der Programmiersprache Dart realisiert werden. | | |
|  |  | Dart |  |  |  |
|  |  |  |  | | |
|  | TF-7 | Storage | Das Spiel soll gleichermaßen mit der Referenzimplementierung des Storageservice als auch der Eigenrealisierung des | | |
|  |  | Referenz | Storageservice funktionieren. | | |
|  |  |  |  | | |
|  | TF-8 | Storage | Die Eigenrealisierung des Storageservices muss den Referenztest der Referenzimplementierung implementieren. (Nicht | | |
|  |  | Referenztest | erfolgreiche Testfälle sind explizit auszuweisen, zu dokumentieren und zu begründen!) | | |
|  |  |  |  | | |
|  | TF-9 | Browser | Das Spiel muss im Browser Chromium/Dartium (native Dart Engine) funktionieren. Das Spiel muss ferner in allen anderen | | |
|  |  | Support | Browserrn (JavaScript Engines) ebenfalls in der JavaScript kompilierten Form funktionieren (geprüft wird ggf. mit Safari, | | |
|  |  |  | Chrome und Firefox). | | |
|  |  |  |  | | |
|  | TF-10 | MVC | Das Spiel sollte einer MVC-Architektur folgen. | | |
|  |  | Architektur |  |  |  |
|  |  |  |  | | |
|  | TF-11 | Erlaubte | Erlaubt sind alle dart:\* packages, sowie das Webframework start. | | |
|  |  | Pakete |  |  |  |
|  |  |  |  | | |
|  | TF-12 | Verbotene | Verboten sind Libraries wie Polymer oder Angular. (Sollten Sie Pakete verwenden wollen, die außerhalb der erlaubten | | |
|  |  | Pakete | 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. | | |
|  |  |  |  |  |  |

## Spielkonzept

Path Puzzle ist ein Strategiespiel, welches auf einem n x m Spielfeld basiert. Dieses ist in ein 2D Raster gegliedert. Der Spieler blickt aus der Vogelperspektive auf das Spielfeld herab. Ziel des Spieles ist es, einen Weg vom Eingangspunkt zum Ausgangspunkt mit den gegebenen Spielsteinen zu bauen. Die gelingt ihm, indem er die vorgegebenen Puzzelstücke vertauscht. Die Schwierigkeit besteht darin, dass er lediglich eine feste Anzahl hat, je nach Level, um die Stücke zu bewegen, oder einen Timer, der eine Zeitspanne vorgibt, in welcher der Weg gebaut werden muss.

Um die Level anzupassen, sind folgende Erschwerungen vorstellbar:

* Feste, nicht veränderbare Teile
* Puzzelstücke sind zu Beginn verdeckt
* Vergrößerung des Spielfelds
* Nur die genaue Anzahl an Zügen, die es für den am kürzesten zu bauenden Weg braucht
* Hindernisse
* Timer
* Mehrere Ein und Ausgänge

# Architektur und Implementierung

Unsere Architektur ist wie gefordert, gegliedert in einem MVC Konzept. Die Klasse FroggerView kapselt die Schnittstelle zum DOM-Tree und dient zur Visualisierung des Models und stellt dem Controller entsprechende Methoden bereit, um die Spieloberflache¨ zu aktualisieren. Näheres¨ wird im Abschnitt [3.2](#page35) erläutert.

Die Klasse FroggerController ist die Schnittstelle zwischen dem Model und der View. Der Controller kontrolliert das Model und fordert die View auf sich zu aktualisieren. Näheres wird im Abschnitt 3.3 erläutert.

Das Model selbst ist komplex und kann nicht in einer einzelnen Klasse erfasst werden, viel mehr besteht das Model aus einem Zusammenspiel vieler einzelner Entitäten, die Komplexität überschaubar machen. Näheres wird im Abschnitt 3.2 erläutert.

3.1. Spiel- und Levelparameter

Die Spielfeldparameter werden in der JSON Datei „Field“ gehalten. Sie enthält wichtige Parameter wie der Spielfeldgröße und die Festlegungen eindeutiger Kennzeichnungen der Level.

Näheres dazu wird in der exakten Beschreibung der Field Datei erläutert (Abschnitt 3.1.2.2).