Universität Leipzig - Softwaretechnik Praktikum 2014/2015 Entwurfsbeschreibung zum Projekt: Ein kartenbasiertes "Multiplayer"-Spiel

Gruppe: SWT15-GKP

31. Mai 2015

Inhaltsverzeichnis

1 Allgemeines

Im Rahmen des SWT-Praktikums wird ein kartenbasiertes Multiplayerspiel entwickelt, das es ermöglicht, ein an das alte Pacman angelehnte Computerspiel auf einem realen Kartenausschnitt zu spielen. Die entstehenden Highscores werden dann zu den jeweiligen Karten gespeichert, um sich mit anderen Spielern messen zu können.

2 Produktübersicht

Der Nutzer kann online über die Spiele Website auf das Programm zugreifen. Der Spieler kann durch Scrollen über die Karte einen Spieleort auswählen. Es sind nur bestimmte Orte als Spielort zugelassen, um eine Vergleichbarkeit zu gewährleisten. Die spielbaren Spieleorte sind durch bestimmte Marker auf der Karte erkennbar. Hat man den Ort ausgewählt, indem man auf einen der Marker geklickt hat, wird das Spielfeld erzeugt und man kann ein neues Spiel beginnen. Das Spiel wird vorerst Pausiert, um dem Spieler eine kurze Eingewöhnungszeit zu geben, klickt man irgendwo auf das Spielfeld, beginnt das Spiel. Man kann den Pucman nun mit Hilfe der Pfeiltasten steuern. Ziel des Spiels ist es den Pucman über die Karte zu steuern und die Kekse, die auf der Karte verteilt liegen, zu fressen.

An dieser Aufgabe will den Spieler der Geist hindern, der sich auch über das Spielfeld bewegt und Pucman fressen kann. Frisst ein Geist den Spieler, so wird die Anzahl seiner Leben wird um eins verringert. Falls keine Leben mehr übrig sind, wird das Spiel beendet und die erreichte Highscore zusammen mit der Karten URI und dem Namen des Spielers, der eingegeben werden muss, abgespeichert. Während des Spielens werden die Möglichkeiten zur Veränderung der Kartenansicht deaktiviert. Das Spielgeschehen ist durch einen Soundtrack unterlegt, der eine funktionale und inhaltliche Verbindung zwischen Bild und Ton generiert. Desweiteren werden für das Spiel wichtige Informationen wie aktueller Highscore, die verbleibende Anzahl an Leben und die Möglichkeit, direkt auf die Website des Spiels zugreifen zu können, angezeigt.

3 Grundsätzliche Struktur- und Entwurfsprinzipien

Als grundsätzliche Programmiersprache haben wir uns für Java-Script entschieden, wobei der Java-Scriptcode von einer HTML-Seite aufgerufen wird. Desweiteren benutzen wir Phaser als Gameframework. Ein Teil der Software wurde mithilfe von Funktionen aus der Java-Script Bibliothek JQuery geschrieben, da diese eine intuitive Herangehensweise bietet und für uns verständlicher als klassiches JavaScript ist. Die von uns umgesetzte Web-Anwendung basiert auf einer Client-Server Architektur. Wir arbeiten mit einem HTML-Server auf dem die Daten liegen, die vom Client abgefragt und an diesen übertragen werden. Diese Daten beinhalten auch den Gameblock, der dann vom Client ausgeführt wird. Außerdem gibt es noch ein Map-Modul, welches das Level aus den Geo-Daten erstellt, die mithilfe der Overpass API von OpenStreetMaps bezogen werden. Dazu kommt das Highscore-Modul, welches für die Verwaltung der Highscores verantwortlich ist. Desweiteren kommt noch ein Datenserver hinzu, der die Highscores unter Verwendung eines Triplestores speichert.

4 Struktur- und Entwurfsprinzipien der einzelnen Pakete

4.1 Server

Vom Server werden die benötigten Spieldateien zur Verfügung gestellt und automatisch geladen. Desweiteren werden Highscoreanfragen und sonstiger Datenverkehr zwischen dem Dataserver und dem Client verarbeitet und anschließend weitergeleitet.

4.2 Client

4.2.1 Map-Modul

Die Overpass API extrahiert die Geo-Daten aus OpenStreetMap, die zum Hervorheben der benutzbaren Straßenzüge und zur Erstellung des Levels benutzt werden.
Die Datei **Graph.js** sorgt dafür, dass die Geo-Daten von OpenSteetMap weiterverarbeitet werden. Die Erstellung des Spielgraphen ist in dieser Software, der aufwendigste Teil.
Durch eine Vorselektierung durch eine angepasste Query für die Overpass API werden die ersten Straßenzüge entfernt, wenn diese bestimmte Kriterien erfüllen, so wurden z.B. Fußwege, Fahrradwege, aber auch Treppen entfernt. Die Herangehensweise dieser Anfrage sit umgangsprachlich gesagt:„so wenig

wie möglich, so viel wie nötig“. Das führt unter anderem dazu, dass die erstellten Graphen nicht zu groß werden, da wir ansonsten Probleme mit der Komplexität bekommen hätten. Es werden nur Straßen berücksichtigt, die in dem sichtbaren Bereich des Browser Fensters liegen.
Die Anfrage liefert als Rückgabe eine Datei im GeoJson Format, welche wir mittels einem Parser durchlaufen und die relevanten Informationen in einen Graphen der Cytoscape Bibliothek übergeben, wir haben also nun einen ungerichteten Graphen. Knoten mit Geo-Koordinaten, und Kanten zwischen den Knoten.
Als nächsten Schritt haben wir die Koordinaten in Pixel auf dem Bildschirm umgewandelt. In einer Iteration über alle Knoten werden nun Sackgassen gelöscht, da diese für uns unbrauchbar sind. Wir haben auch Straßen aus dem Graphen entfernt, die in einer Kreuzung mit mehr als fünf Straßen münden. Wir wollen nicht zufallsgesteuert eine Straße auswählen müssen, wenn die Spielfigur an einer Sternkreuzung steht und zwei Straßen in die selbe Himmelsrichtung führen.
Da wir am jetzigen Zeitpunkt eventuell mehrere Graphen durch die Selektion der Straßenzüge erhalten haben, lassen wir einen Algorithmus über den Graphen laufen, der uns den größten zusammenhängenden Teilgraphen ausgibt.
Als letzten Schritt müssen wir nun noch die Karte für uns spielbar machen. Wir haben eine Knoten- und eine Kantenliste, um jedoch eine flüssige Bewegung zu gewährleisten, braucht man Pixel in bestimmten Abständen. Es wird eine Interpolation gestartet, die zwischen den Knoten des Graphens die fehlenden Pixel setzt, so dass die Spielfigur sich über ein Netzt aus Pixeln bewegen kann, die eine bestimmte Dichte ausweisen. Die flüssige Bewegung ist leider nicht immer gegeben, da die ausgeführten Funktionen teilweise sehr Rechenaufwändig sind. Bei alten CPUs (z.B. einem Doppelkernprozessor) läuft das Spiel langsamer, als auf einem neuwertigen i7-Prozessor.

4.2.2 Game-Modul

Der Gameblock entspricht dem Model-View-Controller Prinzip, welches durch die Nutzung von Phaser umgesetzt wird. Phaser übernimmt für uns die Eingabe und Ausgabeverarbeitung und stellt den Gameloop zur Verfügung, dies ist eine Schleife, die währen des Spielablaufs immer wieder durchlaufen wird. Dabei werden im wesentlichen zwei Dinge ausgeführt: Die Update-Methode enthält: Tastatur- und Mauseingabe, Bewegung, Kollisionserkennung, etc.
Die Render-Methode enthält: Anzeige aller Sprites an den neuen Positionen, Hintergrund animieren, etc.
Die Game Loop wird abgebrochen, wenn das Spiel beendet wird.

Interface.js
In diesem Modul wird die Schnittstelle zum Benutzer definiert. So werden

hier z.B. Knöpfe und Anzeigen, die dem Benutzer Informationen über den aktuellen Spielstand liefern, bzw. diese abrufbar machen, erstellt. Die Schnittstelle ist der Teil eines Systems, welcher der Kommunikation dient.

Load.js
Zeigt einen neuen Ladescreen, während die Karte lädt.

Game.js
Hier ist der Gameloop implementiert.

Character.js
In diesem Modul wird die Spielfigur, also der Pucman definiert.

4.2.3 Highscore-Modul

Das Highscore Module verwaltet die Highscores der Spieler, d.h. es stellt Anfragen an den Server zum Triplestore und kann sie über den Server auch abfragen, falls sie im Spiel angezeigt werden sollen. Die Highscores werden in einem Tripelstore gespeichert, welche mithilfe von SPARQL Queries abgefragt werden, bzw. durch SPARQL Update Methoden in die Datenbank eingepflegt werden. Die Realisierung dieser Methoden wird von den Modulen **Highscore.js**, **Dbupdate.js** und **Dbquery.js** übernommen.

4.3 Bibliotheken

Cytoscape.js
Diese Bibliothek wird von dem Map Modul benötigt um die Graphalgorithmen auszuführen. In der Bibiothek sind bereits viele Algorithmen enthalten.

JQuery.js
Diese Bibliothek vereinfacht den Umgang mit verschiedenen aufgetretenen Problem.

Phaser.js
Wie betreits erwähnt, stellt Phaser als Gameframework einen sehr wichtigen Teil unseres Projektes dar. Die Bibliothek stellt alle Funktionen zur Verfügung die man als Spieleentwickler benötigt, um einen flüssigen Spielfluß, eine Eingabe- und Ausgabeverarbeitung etc. zu haben.

5 Datenmodell

Siehe Abbildung 1.

Beschreibung zu Abbildung 1:
$A \xrightarrow{\text{Datenfluß}} B$ beschreibt den Datenfluss von A nach B, wobei A und B Module der Software sind.

6 Testkonzept

Am Ende des Sprints wird die aktuelle Version der Software auf den Webserver geladen. Diese wird anhand eines Test-Workflows geprüft. Dabei werden die angegebenen Eingaben getätigt und die Reaktion der Software mit der erwarteten Ausgabe verglichen. Sollten dabei Abweichungen auftreten, gilt der Test als nicht bestanden.

6.1 Testumgebungen

1. OS: Ubuntu 15.04
   Kernel: Linux version 3.19.0-18-generic
   CPU: Intel(R) Core(TM) i7-4712MQ CPU @ 2.30GHz
   RAM: 8GiB
   Firefox 30.0
   Resultat: Läuft flüssig.

2. OS: Ubuntu 15.04
   Kernel: Linux version 3.19.0-18-generic
   CPU: Intel Core i5-2520M CPU @ 2.50GHz x 4
   RAM: 7,7 GiB
   Chrome 42.0.2311.90 (64-bit)
   Resultat: Läuft flüssig bei 30% Auslastung mit 15% Idle.

3. OS: Arch Linux
   Kernel: x86_64 Linux 4.0.3-1-ARCH
   CPU: Intel Core2 Duo CPU T7300 @ 2.001GHz
   RAM: 3883MB
   Firefox 38.0.1
   Resultat: Läuft eher langsam, eine Kern voll ausgelastet.

7 Erweiterbarkeit

Die entwickelte Software ist in verschiedene Module zerlegt, so dass eine Erweiterbarkeit an diversen Punkten denkbar ist. Wir benutzen Mapstraction um die Hintergrundkarte anzuzeigen. Mithilfe von Mapstraction kann man mit nur wenigen Zeilen Codeänderung die Karten API austauschen, wenn z.B. andere Features benötigt werden. Der Geist ist in der **ghost.js** definiert, was es ermöglicht diese Datei zu editieren um die Bewegung des Geistes intelligenter zu gestalten. Es ist auch denkbar weitere semantische Daten in das Programm einfließen zu lassen. Etwa Powerups einzuführen, die an bestimmten Orten auf der Karte erscheinen, wie z.B. Kraftpillen an Krankenhäusern , oder die Geister an bestimmten Orten starten zu lassen, z.B. an einer Polizeistation.

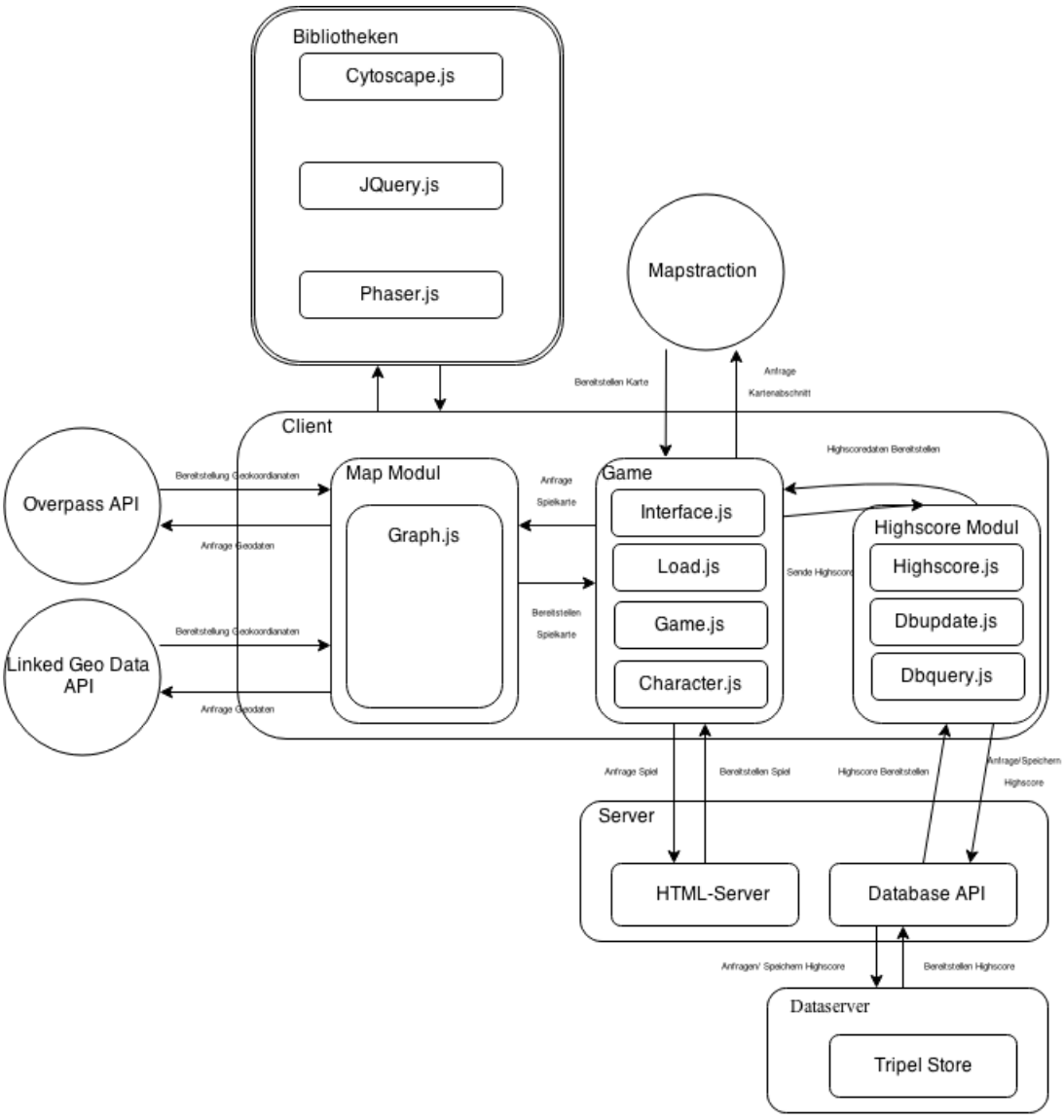

Abbildung 1: Architekur

8 Das Projekt im Netz

Die Projekt Website findet man unter
`http://pcai042.informatik.uni-leipzig.de/~swp15-gkp/`,
hier liegen alle wichtigen Dokumente zu dem Projekt.
Der Source-Code ist unter `https://github.com/GKP15/pucman` abrufbar.
Eine Einführung in GitHub gibt es unter `http://lmgtfy.com/?q=github+tutorial`.

Struktur des Source-Codes:

/**lib**
Hier liegen die Bibliotheken, die wir benötigt haben, wie Phaser, Cytoscape, etc.

/**rdf**
Daten für den Triplestore.

/**resources**
Bilder, Musik, etc.

/**src**
Die einzelnen Module für das Spiel, als Java-Script Code.

/
Die Html-Dateien, Config-Dateien, etc.

9 Glossar

Das Glossar wurde überarbeitet und als externes Dokument angefügt.