Videospiel-Releases über Ort & Zeit

Paul Hufnagel, 11145561, Medieninformatik-Master
https://git-ce.rwth-aachen.de/paul.hufnagel/visualization-project-game-releases

In diesem Dokument werden nun die Änderungen und Abweichungen zum Project Proposal von Anfang Juni erläutert. Den Beginn macht das Szenario, welches nun in seiner aktuellen Version in diesem Dokument präsentiert wird, anschließend geht das Dokument auf die Änderungen zu den Visualisierungskomponenten ein, vergleicht die Sketches mit dem finalen User Interface und im letzten Abschnitt werden dann noch weitere Änderungen aufgelistet, die in die vorherigen Kapitel nicht gepasst haben, aber dennoch stattgefunden haben.

Ein mögliches Szenario:

Peter ist ein Videospiel-Entwickler, der die aktuellen Trends in seiner Branche analysieren möchte, um zu verstehen, welches Projekt er als nächstes angehen könnte. Sobald Peter die Game-Release-Visualisierungsapp öffnet, sieht er anhand des Heats auf der Weltkarte, wie die Verteilung der Videospielentwickler aussieht, und die Zeitachse zeigt ihm auf, wann welche und wie viele Spiele erschienen sind. Neben einer Heatmap hat er dennoch auch die Möglichkeit sich stattdessen Pins anzeigen zu lassen, welche den exakten Standort eines Entwicklers darstellen können. Mit den Pins lässt sich über die Verteilung ein genaueres Bild zeichnen. Er kann sowohl die Karte als auch die Zeitachse nach seinen Bedürfnissen [filtern] und kann anhand der Verteilung die Veröffentlichungen [vergleichen], um zu verstehen, welche Genres aktuell am meisten herausgebracht werden, um daraus etwaige Trends abzuleiten. Die Zeitachse bietet zudem die Möglichkeit die Entwicklungen in seiner Branche zu [beobachten], um daraus ebenfalls Schlüsse ziehen zu können. Wenn Peter an einem einzelnen Spiel oder Entwickler interessiert ist, kann er wahlweise über die Karten-Filter oder über die Spieleauswahl auf ein einzelnes Spiel zugreifen, Informationen darüber [abrufen] und zeitgleich auch das Herkunftsland des Entwicklers sehen.

Die beiden als Zentralkomponenten beschriebenen Views sind vorhanden. Es gibt eine Weltkarte, welche die Verteilung von Entwicklerstandorten darstellt. Anders als ursprünglich geplant, besteht diese Karte jedoch nicht ausschließlich aus Pins. Stattdessen verwendet die Karte eine Heatmap, welche basierend auf der Dichte der Entwicklerdaten Heat über die Kontinente verteilt. Wie der Heat genau verteilt wird, lässt sich einer Legende entnehmen, die sich per Button ein- und ausblenden lässt. Die Idee von Pins wurde jedoch nicht gänzlich verworfen. Der Nutzer kann wahlweise auch auf eine Pin-basierte Weltkarte wechseln, welche dann für jeden Entwicklerstandort einen grünen „Pin“ darstellt, welcher im Projekt durch einen grünen kleinen Kreis visualisiert wird. Die Pins lassen detailliertere Rückschlüsse auf einzelne Standorte zu, aufgrund ihrer Menge ist diese Visualisierungsart jedoch auch rechenintensiver. Um zu verhindern, dass Karte versehentlich bewegt wird – etwa beim Scrollen – ist diese zunächst gelocked und wird erst bewegbar, wenn der Nutzer vor einmal mit Linksklick in die Karte geklickt hat. Daneben gibt es zudem einen Filter mit dem sich die auf der Karte befindlichen Entwicklerstudios filtern lassen, indem ein Studio per Dropdown-Menü ausgewählt werden kann, was alle anderen Studios ausblendet. Im Project Proposal war zudem ein Filter nach Ländern zu sehen, welcher jedoch nicht umgesetzt worden ist. Aufgrund der bidirektionalen Verbindungen zwischen Karte und Zeitachse, wenden sich die Filter der Zeitachse jedoch auch auf die Karte an, genauso wie der Studio-Filter sich auch auf die Zeitachse anwendet. Näheres dazu, sobald der Zustand der Zeitachse beschrieben worden ist. Schlussendlich gibt es für die Karte noch einen Button, mit dem sich die Karte in den Ursprungszustand zurücksetzen lässt. Dies ist beispielsweise dann sinnvoll, wenn Filter auf die Karte angewendet worden sind.

Die Zeitachse ist genauso umgesetzt worden wie geplant. Es gibt einen Zeitstrahl, der zunächst einen Ausschnitt an Jahren anzeigt. Mit 2 Buttons lässt sich je ein Jahr vorwärts gehen, sowie ein Jahr zurück. Per Hovering über die Balken in einem Jahr lässt sich die Menge an Releases in dem Jahr sehen und eine Liste unterhalb der Zeitachse stellt alle Spiele dar, die in dem Jahr rausgekommen sind, auf den der Nutzer geklickt hat. Wenn der Nutzer anschließend in der Liste über einen Spieletitel hovert, werden zu dem jeweiligen Spiel weitere Infos per Tooltip eingeblendet. Alternativ kann das Spiel angeklickt werden und es öffnet sich ein Popup, das dieselben Informationen bereithält. Mit diesem Klick wird in der Weltkarte zudem ein Pin gesetzt, der die Position des Entwicklers anzeigt, sofern dazu Daten vorliegen. An Einstellungsmöglichkeiten gibt es für die Zeitachse ebenfalls ein paar Optionen. Wie im Proposal vorgesehen lässt sich die Zeitachse entweder nach einzelnen Jahren oder nach Intervallen filtern. Auch die ursprünglich als optional angesehenen Filter für Genres und Plattformen sind vorhanden. Neu hinzugekommen ist zudem eine Spieltitel-Suche, welche den Nutzer nach einem expliziten Spiel suchen lässt. Allerdings ist die Suche case-sensitive, wodurch teilweise ein paar Anläufe braucht, bis man das Spiel findet was, man sucht. Wie bereits im vorherigen Absatz erwähnt, wirken sich die Filter der Zeitachse auch auf die Weltkarte aus. Wenn beispielsweise ein kleineres Intervall gewählt wird, wird auch automatisch der Heat mit reduziert, da dadurch weniger Spiele und somit weniger Entwickler vorhanden sind.

Alle Daten werden über lokal hinterlegte CSV-Datensätze bezogen, welche wiederum von Kaggle stammen.

Die innovative Komponente des Projekts stellt die Zeitachse dar. Dafür bietet D3 keine Vorlagen und somit muss diese von Grund auf entwickelt werden. Des Weiteren stellt die Zeitachse über eine Bar die Anzahl der Veröffentlichungen pro Jahr dar. Sie wurde wie gesagt, genauso umgesetzt, wie geplant.

Sketches

Abbildung 1: UI-Übersicht

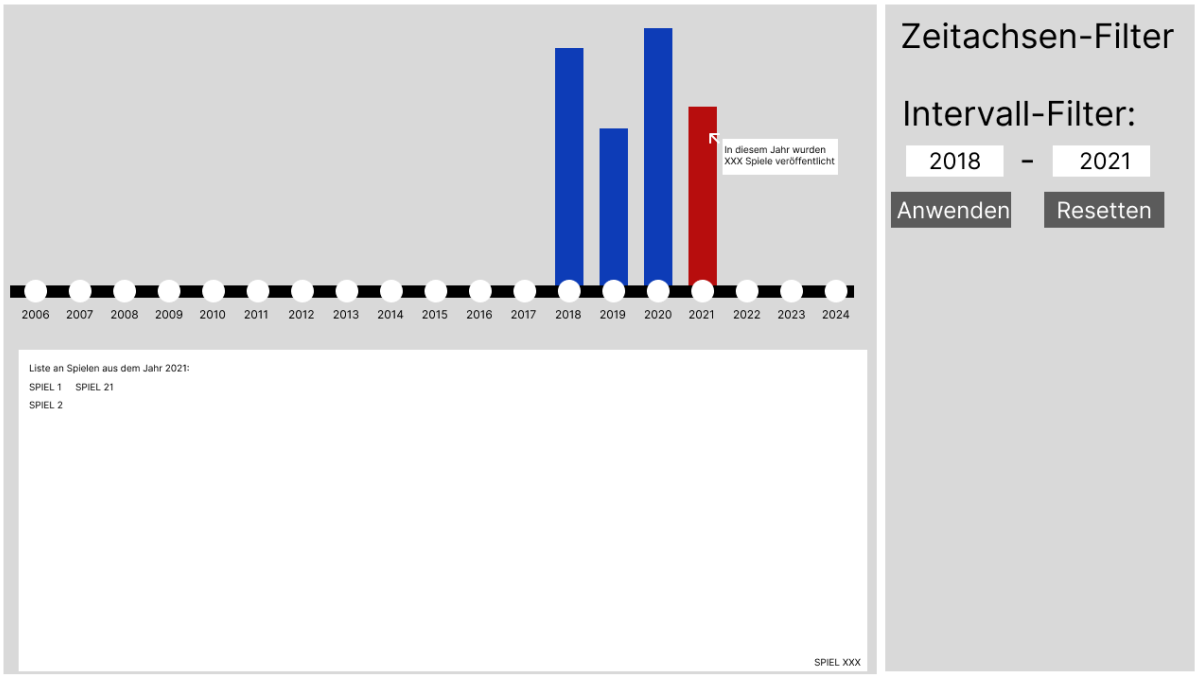

Abbildung 2: Zeitachsen-Sketch

Abbildung 3: Weltkarte

(Karte von Freepik.com)

Die ersten 3 Abbildungen stellen die Sketches zum Zeitpunkt des Project Proposals dar, welche nun mit dem finalen User Interfaces verglichen werden können.

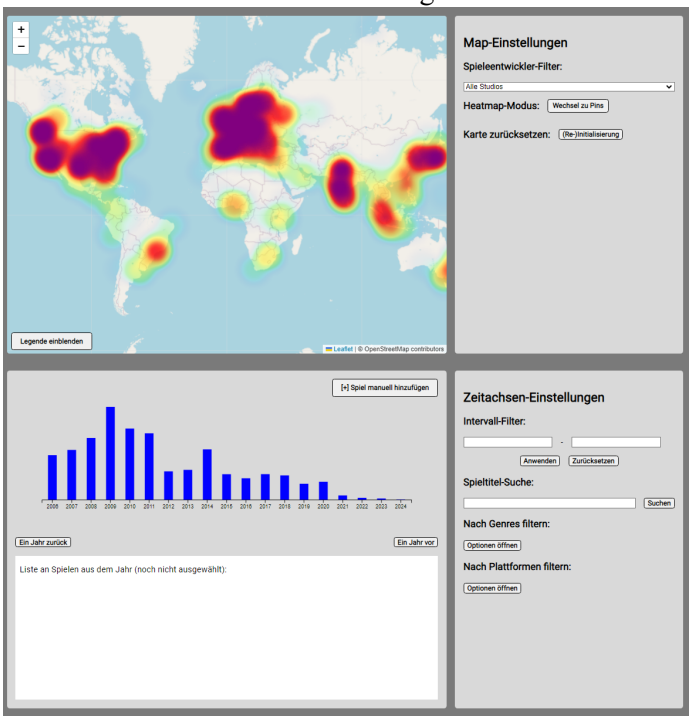

Abbildung 4: Finales UI

Wie in Abbildung 4 zu sehen, wurde die Grundstruktur, die in Abbildung 1 festgehalten worden ist, nicht verändert. Links sind immer die View-Komponenten, während direkt rechts von ihnen ihre jeweiligen Einstellungsboxen sind. Die meisten Elemente des User Interfaces sind bereits im Abschnitt zuvor erläutert worden. Die verbliebenden Elemente werden im kommenden Abschnitt beschrieben.

Weitere Änderungen

In diesem Abschnitt finden nun Änderungen Platz, die laut Project Report Write-Up zwar nicht ausdrücklich gefordert waren, aber dennoch stattgefunden haben.

So wurde auf den Einsatz einer zweiten Geocoding-API verzichtet. Die Applikation verwendet die Google Places API. Diese wurde nun statt der Google Geocoding API verwendet. In der Ausführung macht dies keinen Unterschied, da dadurch dennoch Geokoordinaten vorliegen, allerdings unterscheidet sich hier der Input. Die Google Geocoding API fordert Adressdaten als Input, der Google Places API reicht ein Stichwort – in diesem Fall der Name eines Entwicklerstudios. Die OpenCage API wird hingegen gar nicht mehr verwendet. Der Einsatz der Google API findet jedoch auch nicht wie ursprünglich statt. Statt auf Livedaten zu basieren, basiert die Applikation nun auf zwei CSV-Datensätzen. Der erste enthält die Spieledaten (Titel, Entwickler, Plattform, Genre usw.). Der zweite enthält – basierend auf den Entwicklern aus dem ersten CSV-Datensatz – eine Liste an Geokoordinaten für besagte Entwickler. Dadurch muss nicht bei jeder Benutzung der App erneut eine Anfrage für die Daten gestellt werden. Die Erstellung des CSV-Datensatzes mit den Geokoordinaten wurde über ein Fetching-Skript realisiert, welches sich ebenfalls im Projekt-Repository finden lässt.

Dennoch wurde die API-Nutzung nicht gänzlich über Bord geworfen. Der Nutzer hat die Möglichkeit manuell ein Spiel in die Zeitleiste einzufügen. Dazu kann er ein entsprechendes Popup im User Interface triggern, dort sein gewünschtes Spiel per Titel eingeben und die App versucht dann das passende Spiel einzufügen. Hierfür wird der Button „[+] Spiel manuell hinzufügen“ verwendet, welcer sich im Bereich der Zeitachsen-Komponente befindet (siehe Abbildung 4). Dazu wird live eine Anfrage an die RAWG API (respektive RAWG.io Datenbank) gesendet, wo über den Titel weitere Daten ermittelt werden. Wenn die notwendigen Daten gefunden worden sind, wird das Spiel der

Zeitleiste hinzugefügt. Ein Hinzufügen zur Karte findet jedoch nicht statt. Es war ursprünglich geplant mit den vorhandenen Entwicklerdaten dann noch eine Anfrage an die Google API zu stellen, womit der Entwicklerstandort hätte ermittelt werden sollen, allerdings ließ die Google CORS Policy dies nicht zu und hat jegliche Anfragen verweigert. Hier wäre zum Beispiel der Einsatz einer Proxy nötig gewesen, was jedoch nicht gewünscht gewesen ist. Der Vollständigkeit halber befindet sich der Code dazu, dennoch noch in den Projektfiles, obwohl er auskommentiert ist und nicht zum Einsatz kommt.

Wie bereits in diesem Kapitel erläutert, basiert das Projekt nun auf lokal vorhandenen CSV-Datensätzen. Diese wurden im weiteren Prozess noch weiterverarbeitet. So hat sich etwa herausgestellt, dass der ursprüngliche Kaggle-Datensatz zwar der komplexeste war, jedoch keine Genre-Daten enthielt. Es konnte dann dennoch ein weiterer Datensatz gefunden werden, der die Genres bereithielt, allerdings hatte dieser zum Teil andere Daten. Um keine Daten zu verlieren – und um die Geokoordinaten nicht erneut ermitteln zu müssen, kam 2 weitere Skripte zum Einsatz: Ein Merger und ein Unifier, welche die beiden vorhandenen Kaggle-Datensätze zusammengesetzt haben. Auch deren Code befindet sich, wie schon beim Fetching-Skript im Projekt-Repository.

Im Project Proposal gab es noch ein weiteres Kapitel mit dem Namen „Optionales“. Darin enthalten waren 2 Erweiterungsideen für das Projekt, die hätten umgesetzt werden sollen, wenn während der Entwicklung genüg Zeit dafür gewesen wäre. Der erste Vorschlag die Einbindung von weiteren Attributen aus der RAWG-Datenbank. Diese Erweiterungsmöglichkeit konnte nicht mehr umgesetzt werden, da das Projekt seine Daten eben nicht mehr live (über RAWG) bezieht, sondern per lokalem Kaggle-Datensatz. In der aktuellen Version, werden aber beim manuellen Hinzufügen von Spielen einige Attribute über RAWG abgefragt. Allerdings werden dabei vor allem Eigenschaften abgefragt, die auch bei den bereits vorhandenen Daten verwendet werden.
Die zweite Erweiterungsmöglichkeit wäre die Umsetzung eines kleinen Quiz-Moduses gewesen. Diese Idee wurden hingegen nicht umgesetzt, da der Zeitplan dies nicht zugelassen hat, da an dem Hauptprojekt länger als erwartet gearbeitet wurde.