Projektbericht zum Modul Information Retrieval und Visualisierung Sommersemester 2022

Marktanalyse des Videospielemarktes

Analyse und Visualsierung der Verkaufszahlen der Videospiele auf der Platform XBoxOne

Lena Arloth

13. November 2022

Inhaltsverzeichnis:

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Der Spielemarkt ist groß, es steckt viel Geld dahinter und für die Entwicklung von neuen Spielen sind viele finanzielle Mittel nötig. Es braucht also eine gewisse Sicherheit, dass sich die Investitionen lohnen. Dazu braucht es eine Anaylse der verganenen Daten. XBox als Platform sehr beliebt, wenn auch hinter Playstation. Vergleichbarkeit der Spiele pro Platform, da zB XBox Spieler nicht mit Playstation-Spielern spielen können und Konsole nicht mit PC.

1.2 Zielproblem

Marktanalyse der Spieleindustrie der Platform XBox One aus Sicht der Publisher zur Ermittlung der Konkurrenz, der Erlangung von Kenntnissen über den Markt und über die eigenen Erfolge/den eigenen Standpunkt im Markt zur besseren Ausrichtung und Kontrolle des Unternehmens/Publishers anhand eines Marktüberblicks und genaueren Verkaufsfakten.

1.3 Relevanz

Jedes Unternehmen braucht eine Marktanalyse, so auch Spieleunternehmen/Publisher. Ein Publisher muss wissen, was er zukünftig mit guten Verkaufschancen entwickeln kann und in welchem Genre. Wo liegt der Schwerpunkt des Unternehmens.

1.4 Fragestellungen

1.4.1 Überblick

Welche Genres bedient ein Publisher aus seiner Sichtweise und wieviele Spiele bietet er in diesem Genre an? Wo liegt der Fokus/Schwerpunkt des Publishers? Wer ist die Konkurrenz in den verschiedenen Genres, der zB mehr Spiele in dem Genre anbietet?

1.4.2 tiefere Marktanalyse

Wie gut verkauften sich Spiele im Einzelnen eines Publishers eines Genres im Vergleich zu anderen Spielen des Genres? Wo gibt es Verbesserungspotenzial, wo gibt es Marktlücken, wo gibt es viel Konkurrenz? Gibt es Spiele in einem Genre eines Publishers, die sich nicht mehr lohnen?

1.4.3 konkreter Blick auf jedes Spiel einzeln

Wie verkauften sich Spiele des Publishers im einzelnen im direkten Vergleich auf den verschiedenen Märkten/Regionen der Welt?

1.4.4 allgemein

Welche Spiele waren gut, welche schlecht? Welche Spiele soll der Publisher anbieten in welchem Genre? Wo liegt Verbesserungspotenzial?

1.5 Anwendungshintergrund

Sinn der Marktanalyse in Unternehmen Sinn der Vergleiche von Genres Fakten zu Größe und Volumen des Spielemarkts ingesamt und in verschiedenen Regionen Fakten XBoxOne und Plattformen Fakten zu Anzahl Publisher und wieviele Spiele ein Publisher ungefähr bietet Was die Anwendung/Visualisierung bringen soll

1.6 Zielgruppen

Publisher und dort oberes und mittleres Management/die Entscheider Vorwissen: Kein detailliertes Vorwissen zu speziellen Visualisierungstechniken. Aber Vorwissen zu BWL-Analysen, da sie im Grunde Analysten sind. Kennen also aus ihrer täglichen Arbeit Diagramme, Scatterplots, Koordinatensysteme, Baumdiagramme, Zeitreihendiagramme und parallele Koordinaten.

Eventuell auch Forces und Recursive Pattern. Kennen also nicht die ganz spezifischen Visualisierungstechniken/haben kein ganz spezifisches Vorwissen, sodass hier viel erklärt werden müsste, was nicht zielführend wäre. Das was sie kennen, vor allem Diagramme, Scatterplots, Bäume und Zeitreihendiagramme verstehen sie schnell, da sie es gut kennen und intuitiv damit arbeiten können.

Zusätzlich müssen sie die Visualisierungen u.U. auch Business Partnern zeigen oder im eigenen Unternehmen dem weiteren Management. D.h. sie müssen immer wieder auch schnell auf den ersten Blick sehen können, was gezeigt ist und die Visualisierung sagen soll und es entsprechend schnell und intuitiv anderen erklären können. Informationsbedürnis: Harte Fakten. Was bietet der Publisher an? Denn großer Publisher = viele Entwickeler = viele Projekte/Spiele = eventuell braucht das Management erstmal eine Übersicht, was sie überhaupt alles anbieten und in welchen Genres die Spiele eingeordnet sind zur Erinnerung. Zusätzlich zum Überblick über die Konkurrenz in gleicher Weise. Wie sind die Zahlen der Spiele im Detail in den verschiedenen Märkten? Wie sind die Zahlen der Spiele des Genres im Vergleich zur Konkurrenz in dem Genre? Wo liegt das Verbesserungspotenzial des Publishers in seiner Ausrichtung der Spiele? Wo gibt es Marktlücken und wo liegen die Stärken?

1.7 Überblick und Beiträge

Überblick Daten: XBoxOne Datensätze aus 2016 mit den Attributen Position in der Tabelle, Publisher, Jahr der Veröffentlichung, Genre, Verkaufszahlen global, VK NA, VK EU, VK Asien, VK andere von der Platform Kaggel. Info Datenherkunft auf Kaggle hinzufügen. relevant: alle außer Jahr der Veröffentlichung Überblick Visualsierungen: Visualisierung 1: explizites Baumdiagramm zur Übersicht über Publisher und den von ihnen jeweils bedienten Genres und den Spielen in den Genres jeweils zur Beantwortung Fragengruppe 1 Visualierung 2: Scatterplot zur Beantwortung von Fragegruppe 2, der tieferen Marktanalyse, also wie sich die Spiele in einem Genre im Vergleich verkauften mit Zusammenhängen zwischen globalen Verkäufen und Verkäufen in den auswählbaren Regionen. So kann ein Publisher vergleichen, wie seine Spiele eines Genres im Vergelich zur Konkurrenz verkauft wurden bzw wie erfolgreich seine Spiele waren. Visualisierung 3: parallele Koordinaten zur Beantwortung von Fragegruppe 3, damit Publisher auf einen Blick vergleichen können, wie ein konkretes Spiel von ihnen in den verschienden Regionen/Märkten verkauft werden konnte. Auf welchen Märkten war es am erfolgreichsten, welche Märkte sollten stärker beworben werden, auf welche sollte der Fokus gelegt werden? Spiele auswählbar je nach Genre. Beitrag allgemein der Visualisierungen: Lösung Problemstellung und Fragegruppe 4, also der allgemeinen. schnellerer Überlick über Erfolg der Spiele des Publishers in den Märkten und in Konkurrenz zu den anderen Spielen eines Genres zur zukünftigen Ausrichtung des Publishers. kurze Vorstellung des Vorgehens im Projektbericht: 1. Daten, 2. Visualisierungsbeschreibung + Interaktion, 3. Konkrete Umsetzung/Implementierung, 4. Konkreter Anwendungsfall, also ein Publisher aussuchen und durch die Visualisierungen gehen und dabei die oben beschriebenen Fragen beantworten und Problem lösen, 5. Verwandte Arbeiten vorstellen am besten in Kombi

BWL und Visual Analytics und Spielemarkt, 6. Zusammenfassung und ausblick. Für Ausblick kann integriert werden, dass die analyse erweitert werden kann mit adäquaten intern erhobnen Daten der Publisher zu Umsätzen und Gewinnen insgesamt und pro Spiel + zu Kosten pro Spiel für die Entwicklung/Marktreife + Kosten Spiel auf dem Mart zur tieferen Analyse der Verkaufszahlen

2 Daten

Der für das vorliegende Visualisierungsprojekt verwendete Datensatz stammt von der Plattform *Kaggle* von dem Nutzer *SID_TWR* mit dem Titel *"Video Games Sales Dataset"*.[1] Die Daten entstanden laut Nutzer durch Erweiterung der Daten eines Web Scrapes von *VGChartz Video Games Sales* motiviert durch Gregpry Smith um weitere Attribute aus einem Web Scrape von *Metacrtitic*.

Der Nutzer stellt drei Datensätze zur Verfügung, von denen für dieses Projekt der Datensatz *XBoxOne_GameSales* ausgewählt wurde. Hierbei ist zu beachten, dass die zuvor erwähnten Erweiterungen um Attribute stammend von *Metacrtitics* in diesem Datensatz nicht enthalten sind. Der Originaldatensatz liegt als CSV-Datei vor und beinhaltet zehn Spalten mit 613 einzelnen Positionen, also Videospielen.

Zunächst ist eine eindeutige Identifikationsnummern gegeben, hier als *Positionen* bezeichnet. Weiterhin sind der Videospielnamen sowie das jeweilige Jahr der Veröffentlichung der Spiele aufgelistet. Das Attribut *Genre* kategorisiert die Videospiele in verschiedene Genre und *Publisher* ordnet jedem Videospiel seinen Verleger zu. Zuletzt sind die Attribute *North America*, *Europe*, *Japan*, *Rest of World* und *Global* aufgeführt. Sie stellen die Verkäufe der Videospiele in den Regeionen Nordamerika, Europa, Japan, dem Rest der Welt und auf globaler Ebene dar. Die Einheit ist hier bei allen *millions of units*, also Millionen Stück verkaufter Kopien, was durch eine Recherche auf *VGChartz Video Games Sales* deutlich wird. Hochgeladen wurden die Daten circa vor drei Jahren. Der für das Projekt ausgewählte Datensatz bildet Videospiele von 2013 bis ungefähr 2020 ab, wobei seit dem Veröffentlichungsjahr 2019 keine Verkaufszahlen mehr eingetragen sind. Auch auf *VGChartz Video Games Sales* finden sich schon im Laufe des Veröffentlichungsjahres 2018 weniger bis keine Verkaufszahlen. Zwar wurden die Videospiele bis heute weiter eingetragen, jedoch ohne Aktualisierung der Verkaufszahlen. So ist davon auszugehen, dass die für das Projekt verwendeten Daten Verkaufszahlen ab 2013 bis circa 2018 abbilden. Der Stand der Zahlen hat sich auch laut *VGChartz Video Games Sales* aufgrund unbekannter Gründe nicht geändert. Es werden die kumulierten Verkaufszahlen in Millionen Stück in den Jahren seit Erscheinungsdatum bis circa Ende 2018 dargestellt.

Insgesamt werden die vorhandenen Daten als gut geeignet für die anvisierte Zielgruppe des oberen und mittleren Managements der Verleger von Videospielen für die Platform XBox One und das zuvor eingeleitete Zielproblem eingeschätzt. Die Daten ermöglichen eine grobe Übersicht über die verschiedenen Verleger und ihre Angebote in Form von Videospielen und bedienten

Genres. Weiterhin schlüsseln sie detailliert auf, wie sich ein Videospiel seit seiner Veröffentlichung global, aber auch in den einzelnen Weltregionen verkaufte. Auf Basis dieser Daten ist eine Marktanalyse zur Lösung des Problems der fehlenden Übersicht und Kenntnis der Leistung der Videospiele eines Publishers am Markt sowie seiner direkten Konkurrenz im Genre möglich. Denkbar ist jedoch für eine weitergehende und tiefere Analyse die Einbeziehung von Daten zu Kritiken, Entwicklungskosten, Kosten pro Videospielkopie am Markt sowie die durch den Verkauf eines Spiels erzielten Umsätze. Diese Daten sind jedoch größtenteils nicht öffentlich zugänglich, unvollständig oder müssten mit sehr hohem Zeitaufwand aus diversen Quellen zusammengetragen werden. Die Visualisierung dieser Daten würde das hier gesetze Zielproblem ergänzen, jedoch die anvisierte überblicksartige Marktanalyse überschreiten. In einer unabhängigen, weiteren Martkanalyse und den zugehörigen Visualisierungen wäre dies jedoch denkbar.

Der Datensatz enthält jeodch auch fehlende Werte bei den Verlegern, fortan Publisher genannt, und den Verkaufszahlen jeder Region. Die Daten werden also um jene Videospiele bereinigt, denen kein Publisher zugeordnet ist oder deren Publisher unbekannt ist. Weiterhin werden jene Videospieldaten gelöscht, deren Verkäufe in allen Regionen bzw. global einen Wert von Null aufweisen. Zuletzt wird die Zahl der Videospiele um jene reduziert, deren Publisher nur ein Spiel herausgibt. Die Spalte des Erscheinungsjahres des jeweiligen Videospiels wird aus der hier durchgeführten Marktanalyse und ihrer Visualsierung ausgegliedert. Hieraus können weder Daten für ein Zeitreihendiagramm extrahiert werden, noch macht ein spezifischer Vergleich der Verkaufszahlen in Relation zum Erscheinungsjahr für die Problemstellung und deren Lösung in einem ersten Überblick Sinn. Eine Analyse dieser Daten gliedert sich in mögliche weiterführende Visualisierungen, wie im obigen Abschnitt beschrieben, ein.

2.1 Bereitstellung der Daten

Wie zuvor beschrieben, werden die originalen Daten über *Kaggle* zur Verfügung gestellt. Diese werden als CSV-Datei heruntergeladen und zusätzlich zur Bearbeitung in das ODS-Format überführt. Alle Daten sowie das Projekt und dieser Bericht werden in einem öffentlichen GitHub Repository bereitgestellt. Die Datendateien sind im Ordner *Daten* in den Unterordnern *CSV*, *JSON* und *Tabelle* je nach Format zu finden. Die Dateien mit den Originaldaten enthalten die Endung _*Original*, Dateien mit Testdaten die Endung _test und finale für das Projekt nutzbare Dateien die Endung _*Projekt*. Die in Tabellenform überführten Originaldaten werden im ODS-Format bearbeitet, gelöscht, in einer neuen Tabellendatei gespeichert und wieder in CSV konvertiert. Die Datenvorverarbeitung im Detail ist im folgenden Unterkapitel einsehbar. Die CSV-Datei mit der Endung _*JSON* bildet die Grundlage für die Erstellung der JSON-Datei.

Um die Visualisierung mittels explizitem Baumdiagramm zu erstellen, wird eine JSON-Datei benötigt, in der die Beziehungen der Publisher, Genre und Videospiele zueinander beschrieben sind. Videospiele sind hier Kinder der Genres, die wiederum Kinder der Publisher sind und unter einem Wurzelknoten zusammengefasst werden. Für jeden Publisher sind die Genres als Kinder beschrieben, die er bedient. Unter jedem Genres, das ein Publisher bedient werden dann

die Videospiele kategorisiert. Die Datei findet sich im Ordner *Daten/JSON* mit der Bezeichung *XBoxOne_GamesSales_Projekt.json*.

Für die Realisierung der Visualisierungstechniken des Scatterplots und der parallelen Koordinaten wird dieselbe CSV-Datei benötigt. Sie ist im Ordner *Daten/CSV* mit der Bezeichnung *XBoxOne_GamesSales_Projekt* zu finden. In dieser Datei sind die nachfolgend detailliert beschriebenen Modifikationen enthalten. Sie beinhaltet entsprechend die alle Positionen, Videospielnamen, Genres, Publisher, Verkäufe in Nordamerika, Europa, Japan, dem Rest der Welt und die globalen Verkäufe. Dezimaltrennzeichen sind als Punkt realisiert und die einzelnen Werte in der Datei sind mit Kommata voneinander getrennt.

2.2 Datenvorverarbeitung

Zur besseren Nachvollziehbarkeit der Datenvorberarbeitung wird im GitHub Repository im Ordner *Daten* eine zweite README.md-Datei erstellt, die den Fortlauf der Vorverarbeitung dokumentiert. Da der originale Datensatz einige Anpassungen verlangt, um für die Visualisierung sinnvoll nutzbar zu sein, ist eine Datenvorberarbeitung und Modifikation unausweichlich. Die benötigte Datengrundlage wird in sieben Schritten erreicht.

Zuerst werden die Daten als CSV-Datei heruntergeladen und unter der Bezeichnung *XBoxOne_GamesSales_Original.csv* im Ordner *Daten/CSV* gespeichert. Außerdem wird die CSV-Datei in ein ODS-Format konvertiert, um mit dem Programm *Open Office Calc* besser bearbeitet werden zu können. Dies Datei ist unter *XBoxOne_GamesSales_Original.ods* im Ordner *Daten/Tabelle* zu finden.

In einem zweiten Schritt werden Testversionen der Originaldaten mit den ersten 20 Positionen erstellt im CSV- und ODS-Format erstellt. Mit ihnen können die Visualisierungen zunächst übersichtlich auf Funktionaltität getestet werden. Erst in einem späteren Schritt werden dann nur die Quellen für die Daten im Programm ausgetauscht. Die Testdaten werden zudem um die Spalte des Erscheinungsjahres reduziert. Eine Erklärung folgt in Schritt drei. Die Testdateien werden unter *XBoxOne_GamesSales_test.csv* und *XBoxOne_GamesSales_Original.ods* in den Ordnern *Daten/CSV* und *Daten/Tabelle* gespeichert. Im dritten Schritt erfolgt die eigentliche Modifikation in den Originaldaten in originaler Größe. Die Originaldatei im ODS-Format wird zusätzlich als *XBoxOne_GamesSales_Projekt.csv* im Ordner *Daten/Tabelle* gespeichert, sodass hier ohne Veränderung der Originaldaten modifiziert werden kann. Zunächst wird die Spalte *Year* gelöscht. Diese Daten sind für die Visualisierungen nicht nötig. Die Daten geben keine Grundlage für eine Visualisierung als Zeitreihe. Das Erscheinungsjahr könnte jedoch in Relation zu den Verkaufszahlen betrachtet werden, da diese im ersten Moment durch die Kumulation der Verkaufszahlen über die Jahre größer sein können, je länger das Spiel auf dem Markt ist. Jedoch kann auch ein bspw. erst 2017 erschienenes Videospiel höhere Verkaufszahlen aufweisen als eines, das bspw. im Jahr 2013. Zuverlässige, eindeutige und schnell erkennbare Schlüsse, die für eine hier angestrebte überblicksartige Marktanalyse nötig sind, können somit nicht ermöglicht werden. Es werden zusätzlich jene Positionen, also Videospiele, eliminiert, die einen Wert von

Null in allen Regionen der Verkäufe enthalten. Auch unter Anbetracht der Tatsache, dass hier zumindest geringe Verkäufe unter 100.000 verkauften Stück erzielt worden sein könnten, ist ein Vergleich und eine Visualisierung sinnlos. Weiterhin werden die Positionen gelöscht, die keinen Publisher oder den Wert *Unknown* an dieser Stelle enthalten. Sie sind für die Zielgruppe und das Zielproblem irrelevant. Zusätzlich wird der Datensatz um die Positionenen minimiert, die von einem Publisher stammen, der nur dieses eine Videospiel verlegt. Ein Vergleich der Spiele des Publishers untereinander und mit Konkurrenten in dem Genre ist nicht sinnvoll. Zudem ist damit zu rechnen, dass sich jene Publisher sehr spezifisch ausgerichtet haben und keine hier visualisierte überischtsartige Marktanalyse benötigen. Die entstehende Datei wird zuletzt wieder in das CSV-Format konvertiert und unter *XBoxOne_GamesSales_Projekt.csv* im Ordner *Daten/CSV* gespeichert.

Im vierten Schritt wird der Name des Publishers *Namco Bandai Games* in *Bandai Namco Games* vereinheitlicht, da beide denselben Verleger darstellen. Dieser wurde 2014 in letzteren umbenannt

Als Vorbereitung für die Erstellung der JSON-Datei wird im fünften Schritt eine weitere CSV-Datei erstellt, die die Abhängigkeiten für die gewünschte Baumstruktur als Spaltenüberschriften enthält. Die benötigten Daten des Publishers, Genres und Videospielnamens werden entsprechend der Struktur übertragen. Die Datei ist unter *XBoxOne_GamesSales_JSON.csv* im Ordner *Daten/CSV* zu finden.

Der fünfte Schritt beinhaltet die Konvertiertung der erstellten CSV-Datei mit den Abhängigkeiten in eine JSON-Datei. Diese wurde mittels des Online-Tools *convertcsv.com* vorgenommen. Der entstandene Code wird im Editor *Visual Studio Code* in eine neue Datei kopiert und zunächst unter *Daten/JSON* gespeichert. Zusätzlich wird eine Testdatei im JSON-Format mit nur 20 Videospielen erstellt, um schneller und übersichtlicher entwickeln und das Entwickelte testen zu können. Auch diese Datei wird im entsprechenden Ordner gespeichert. Die weiteren Schritte werden sowohl für die Testdatei, als auch für die große, später genutzte JSON-Datei durchgeführt.

Zuletzt wird der JSON-Code auf Fehler in den Abhängigkeiten konrolliert und verbessert. Dasselbe wird respektive in der Datei *XBoxOne_GamesSales_JSOn.csv* vorgenommen. In der JSON-Datei wird ein Wurzelknoten hinzugefügt und die leeren Felder, die im expliziten Baumdiagramm leere Blattknoten darstellen würden, gelöscht. Zusätzlich wird pro Genre der jeweilige Eltern-Publisher wenn möglich abgekürzt in Klammern hinzugefügt. Somit können die Genres eindeutig identifiziert werden und es kommt nicht zu Überschneidungen der Pfade und mehreren eingehenden Pfaden pro Knoten respektive mehreren Elternknoten pro Kindknoten.

Für das in diesem Projekt anvisierte Ziel und den beschriebenen Anwendungsfall sind weitere Berechnungen nicht nötig. Desweiteren ist eine Umrechnung der Einheiten der Verkäufe nicht ratsam, da sie in der vorliegenden Form sehr übersichtlich sind. Bei einer Umrechnung in bspw. Tausend Stück würde die Zahl schon zu unübversichtlich und nicht auf den ersten Blick erkennbar sein, wie es für die Zielgruppe notwendig ist. Zur Erläuterung der Einheiten wird ein

Informationstext erstellt und die Achsenbeschriftungen entsprechend angepasst.

Zunächst wurde eine Filterung der Daten auf Null-Werte in den Verkäufen jeder Region in Betracht gezogen, sodass diese Videospiele, sollten sie auch nur in einer Region einen Null-Wert enthalten, nicht angezeigt würden. Dies ergibt jedoch wenig Sinn im Zusammenhang einer übersichtsartigen Marktanalyse. Hierfür sind auch Informationen über keine erfolgten Verkäufe und die Region, in der nichts verkauft werden konnte, wichtig. Zu beachten ist jedoch, dass als Null-Werte auch solche aufgeführt sind, deren Verkaufszahlen bei unter 0.01 Millionen Stück verkaufter Videospielkopien liegt. In Anbetracht der Größe der Märkte sind diese sehr geringen Verkaufszahlen jedoch als Null und entsprechend verbesserungswürdig anzusehen. Entsprechende Erklärtexte sind notwendig.

3 Visualisierungen

Im Folgenden Kapitel wird zunächst eine Analyse der Anwendung zur Lösung des Zielproblems einer überblicksartigen Marktanalyse der Verkaufszahlen von Videospielen der Plattform XBoxOne durchgeführt anhand derer Anforderungen an die Visualisierungen abgeleitet werden. Schließlich werden die Visualisierungen und Interaktionen präsentiert.

3.1 Analyse der Anwendungsaufgaben

Das Zielproblem dieses Projekts ist eine überblicksartige Marktanalyse des Videospielemarktes der Plattform XBoxOne anhand der Verkaufszahlen der Videospiele global und in einzelnen Regionen der Welt. Ziel der Marktanalyse aus Sicht der Publisher ist eine Übersicht über das eigene Verlagshaus sowie die Konkurrenz, die eigenen Videospielerfolge und Kenntnisse über den Markt zur Erfolgskontrolle vergangener Jahre und informierteren Ausrichtung des Publishers in der Zukunft. Hieraus ergeben sich drei Unterziele zur Lösung des Gesamtproblems, die nachfolgend analysiert werden.

Zunächst soll die Frage nach den angebotenen Videospielen der Publisher und ihrer Konkurrenz in einem ersten, eher qualitativen Überblick beantwortet werden. Hierbei liegt der Fokus aufgrund der gegebenen Daten und deren guten Übersichtsmöglichkeiten auf der Identifikation der von einem Publisher angebotenen Genres und der Videospiele, die jenen zugeordnet werden können. So lässt sich erkennen, welche Schwerpunkte der Publisher in der Vergangenheit bzgl. des Genres legte und wieviele Spiele er in welchem Genre anbot. Besonders für größere Publisher, die viele Videospiele und/oder viele Genres anbieten, ist dieser Überlick sinnvoll. In Kombination mit einer Übersicht über die konkurrierenden Publisher in derselben Art, kann ermittelt werden, welcher Publisher eine starke Konkurrenz in welchem Genres darstellen kann, welche Genres stark oder weniger stark angeboten werden und wieviele Videospiele ihnen zugeordnet werden können. Somit können erste potenzielle Marktlücken, aber auch Sättigungen identifiziert werden, die für eine zukünftige Ausrichtung des Publishers eine Rolle spielen können. Aufgrund der Zielgruppe des oberen und mittleren Managements der einzelnen Publisher, kann

es durchaus der Fall sein, dass u.a. besonders die Darstellung zur Lösung des ersten Unterziels auch Geschäftspartnern und anderen Stakeholdern vorgestellt werden muss, die weniger stark in die Einzelheiten und Strukturen des Verlags eingebunden sind. Weiterhin wird die Annahme getroffen, dass die Zielgruppe häufig und in unregelmäßigen Abständen den Überblick über ihre Videospiele, deren Genres und ihre Konkurrenz auf den ersten Blick benötigt, um sie diese Daten ins Gedächtnis zu rufen. Mental soll bei den Personen der Zielgruppe ein Modell der Hierarchie und Zugehörigkeit der Videospiele zu Genres und dieser Genres zu den Publishern entstehen, um eine Übersicht zu ermöglichen. Dieses mentale Modell soll sich auch in der Visualisierung zeigen. Durch eine Visualisierung der Eltern-Kind-Beziehungen, also der Hierarchie, ist eine Lösung der Fragestellung des ersten Unterziels möglich.

Das zweite Unterziel beschäftigt sich mit einem konkreteren Blick auf die einzelnen Videospiele innerhalb eines Genres in den verschiedenen Regionen der Welt im Vergleich. Hierbei soll die Frage beantwortet werden, wie sich die Videospiele eines Genres im direkten Vergleich in den Regionen der Welt verkauften. Gleichzeitig soll verglichen werden können, wie performant konkurrierende Videospiele des Genres waren. Für Publisher X soll erkennbar sein, welche seiner Spiele in dem ausgewählten Genre in welchen Regionen erfolgreich waren und welche nicht. Durch den Vergleich mit der Konkurrenz kann er möglich Gründe für die Verkaufszahlen seiner Spiele feststellen. Wenn bspw. eines seiner Spiele in allen oder nur einzelnen Regionen nicht gut verkauft werden konnte, ein Spiel eines Konkurrenten aber in diesen Regionen aber gut, dann deutet das auf einen starken Konkurrenten hin. Sind alle oder die meisten Spiele der Konkurrenz in allen oder einzelnen Regionen besser verkauft worden, als das Videospiel von Publisher X, dann deutet dies auf einen gesättigten Markt des Genres mit starker Konkurrenz hin, in das sich möglicherweise weitere Investititonen nicht lohnen. Andersherum können sich auch Marktlücken pro Genre identifizieren lassen. Bei der Bearbeitung dieser Probleme helfen vor allem Übersichten und Vergleiche als mentale Modelle mit Visualisierungen von Daten auf Positionen auf Achsen und der Verbindung der Daten eines jeden Videospiels. So kann sich ein zusammenhängendes Bild über die Verkaufszahlen in den Regionen ergeben ähnlich zu einem Motiv, dass sich durch das Verbinden von Zahlen in einem Zahlenbild für Kinder ergibt. Werden verschiedene solcher Bilder übereinandergelegt, hier entsprechend der konkurrierenden Videospiele, lassen sich Unterschiede erkennen. Sie sind mitunter nicht zwingend zur Lösung nötig, verbessern das Verständnis gerade in Bezug auf Schnelligkeit und Überischtlichkeit aber ungemein.

Zuletzt stellt sich die Frage, wie gut sich die Videospiele im Vergleich zu anderen Spielen des Genres im genaueren Vergleich verkauften. Hierbei legt sich der Fokus auf die möglichen Korrelationen zwischen den einzelnen Regionen und den globalen Verkäufen, aber auch auf konkreter sichtbare Verhältnisse von Verkaufszahlen eines Videospiels eines Genres auch im Vergleich zur Konkurrenz. So lassen sich zum einen anhand konkreter sichtbarer Zahlen als in der zweiten Visualisierung der Erfolg eines Videospiel im Genre kontrollieren, aber zum anderen auch Kenntnisse zu neuen Positionierungen des Publishers für neue Videospiele ermöglichen. Gibt es bspw. viele Videospiele eines Genre, die in der Region Europa schlecht verkauft werden konnten,

aber mittelmäßige bis gute globale Verkaufszahlen zeigen, so kann das auf eine Präferenz der Region für andere Genres deuten. Sollte es zudem einen oder weitere wenige Ausreißer geben, die besonders gute Verkaufszahlen in dieser Region und global haben, dann deutet das auf eine Bedarfsdeckung dieser Region durch das entsprechende Videospiel hin. Eine weitere Investition in diese Region in dem Genre wäre entsprechend zu überdenken. Ist aber bswp. ein Videospiel des Publishers X in dieser Region der Ausreißer und besitzt auch global gute Verkaufszahlen, so ist eine Investition in weitere Videospiele dieser Reihe eine sinnvolle Option. In einem mentalen Modell können diese Analysen durch räumliche Abbildungen bzw. Abbildungen der Daten auf Positionen in einem Kontext dargestellt werden. Zur Unterstützung der einfachen Erkennung von Korrelationen ist ein Modell mit nur zwei Dimensionen sinnvoll. Die Visualisierung soll auch die im mentalen Modell gewünschten, möglicherweise entstehenden Cluster abbilden können. Für die Bearbeitung dieses Unterziels ist das entstehende räumliche mentalte Modell der Abbildung der Daten auf Positionen sehr wichtig und notwendig, um vor allem die Korrelationen und Cluster zwischen den Regionen erkennen zu können.

Insgesamt gliedern sich die Unterziele von einer groben, überblicksartigen Analyse in und zwischen Publishern in und zwischen Genres zu einer konkreteren, aber dennoch überblickartigen Analyse der Verkaufszahlen innerhalb der Genres zur Erfolgskontrolle und Vergleich mit der Konkurrenz bis hin zu einer detaillierten Analyse der Verkaufszahlen im Vergleich von nur zwei Regionen und auch auf Korrelationen und Clustern zwischen den Regionen.

3.2 Anforderungen an die Visualisierungen

Aus den zuvor analysierten Anwendungsaufgaben und Unterzielen leiten sich diverse Anforderungen an die Visualisierungen ab. Allgemein sollen alle Visualsierungen effizient, expressiv und angemessen sein. So ist bedingt durch die Zielgruppe vorrangig wichtig, dass die Inhalte intuitiv und ohne viele Erklärungen erkennbar werden. Weiterhin muss die Aussagekraft eindeutig sein, um unbeabsichtigte Suggestionen zu vermeiden. Zuletzt sollte besonders im Kontext einer betriebswirtschaftlich orientierten Analyse auch die Angemessenheit gegeben sein. Zwar ist eine hier anvisierte Marktanalyse essentiell für ein Verlagshaus und entsprechend auch bezüglich des Ressourchenverbrauchs wertvoll, jedoch werden besonders hier die Kosten in Bezug zur Effektivität gestellt. Es ergibt also wenig Sinn, besonders komplexe, tiefgründige Visualsierungen zu nutzen, die viele Ressourcen verbrauchen, wenn es gerade in Bezug auf die Zielgruppe sinnvollere, eventuell weniger komplexe, aber dafür leichter und schneller verständliche Visualisierungstechniken gibt, die entsprechend weniger Ressourcen in der Erstellung benötigen.

Zur Beantwortung des ersten Unterziels, welches sich mit den angebotenen Videospielen und Genres der Publisher und ihrer Konkurrenz beschäftigt, ist eine Darstellung eben dieser Daten notwendig. Es müssen alle Publisher, sowie die jeweils von ihnen abgedeckten Genres und angebotenen Videospiele überblicksartig dargestellt werden. Wichtig ist Darstellung, die von den jeweiligen Publishern ausgeht und dann spezifischer wird. Um die eigenen Angebote schnell erkennen zu können, sollte nicht nur die Anzahl und Art der Genres je Publisher dargestellt wer-

den. Zur mentalen Gewichtung der abgedeckten Genres ist auch die Darstellung der Anzahl der ihnen zugeordneten Videospiele je Publisher nötig. Aufgrund des geforderten Überblicks und der konkreten Darstellung der Videospiele, müssen auch die Titel der Spiele abgebildet werden. Weiterhin ist es zur Ermittlung der möglichen Konkurrenz nötig, alle Publisher in einer Übersicht in derselben Art und Weise darzustellen. Neben den oben beschriebenen Anforderungen ist eine der wichtigsten jedoch die schnelle Übersicht, vor allem über das eigene Verlagshaus. Es muss ohne viel Nachdenken und viele Erklärungen sowohl für die Zielgruppe, als auch für Stakeholder des Verlages ersichtlich werden, welche Genres der Publisher anbietet und welche und wieviele Spiele unter diesen Genres katgorisiert sind. Auf dieselbe Art muss entsprechend auch die Konkurrenz leicht und schnell ablesbar sein. Die Hauptaufgabe der ersten Visualisierung liegt in der Präsentation der schon klar bestimmten Daten des Sachverhaltes der Publisher, der von ihnen abgedeckten Genres und den ihnen zugeordneten Videospielen in einer schnell verständlichen, hierarchischen Darstellung.

Aus der Analyse des zweiten Unterziels ergibt sich die übergeordnete Anforderung der visuellen Analyse der Datenmenge für die zweite Visualisierung. Es muss in der Visualisierung in einem Vergleich auf einen Blick erkennbar werden, in welchen Regionen der Welt sich welche Spiele eines Genres gut verkauften und wo entsprechend nicht. Weiterhin muss in derselben Darstellung sichtbar werden, wie die konkurrierenden Spiele abschnitten. Somit ist es von Nöten, alle in den Daten gegebenen Regionen in einer Darstellung abzutragen, in der sie miteinander verglichen werden können. Für eine bessere Vergleichbarkeit der Daten soll nach Genre kategorierisiert werden und dieses je nach Betrachtungswunsch ausgewählt werden. In der Visualisierung soll hauptsächlich ein Spiel in verschiedenen Regionen verglichen werden, jedoch auch Spiele untereinander. Entsprechend ist es für das Design wichtig, dass Werte eines Videospiels miteinander verbunden werden. Für ein einheitliches Verständnis soll dies auf dieselbe Weise mit den konkurrierenden Videospielen passieren. Es braucht deshalb aber eine deutliche Unterscheidung der Spiele untereinander, bspw. mittels Farbe, sowie eine Beschriftung mit den Titeln der Videospiele. Die Ausprägung der Verkaufszahlen muss zumindest grob ablesbar sein, wobei der visuelle Vergleich zur Unterstützung des mentalen Modells des übersichtsartigen Vergleichens vor expliziten, genauen Zahlen Vorrang hat.

Die übergeordnete Anforderung an die dritte Visualisierung ist wieder die visuelle Analyse der Datenmenge. Hierbei soll es der Zielgruppe möglich sein, Korrelationen zwischen den Regionen der Videospielverkäufe und den globalen Verkäufen zu erkennen. So kann erkannt werden, wie die Zusammenhänge zwischen bspw. sehr guten Verkaufszahlen einer Region zu den Verkaufszahlen global gegeben sind. Weiterhin sollen noch einmal präziser die Verkaufszahlen der Videospiele der Publisher in einem Genre im Kontext dargestellt werden. Hierzu sollen nur die globalen Verkaufszahlen und die einer auswählbaren weiteren Region aufgezeigt werden. Auch hier besteht die Anforderung, die Analyse und Visualisierung im Kontext des Genres vorzunehmen, damit die Zielgruppen die einzelnen Videospiele konkreter vergleichen können. In dieser Visualisierung muss die Relation von zwei Regionen zueinander dargestellt werden, sowie der

Titel des entsprechendes Videospiel ersichtlich werden. Zum präzisen Vergleich von Verkaufszahlen in den Regionen müssen auch die exakten Zahlen zu finden sein und möglichst nicht nur rein visuell abgetragen werden. Für eine Visualisierung der möglichen Cluster und Zusammenhänge zwischen den Verkaufszahlen konkurrierender Spiele, müssen alle Titel aller Publisher des Genres vertreten sein. Durch die gewünschte detaillierte Analyse ist eine intuitiv verständliche Visualisierung sowie ein gut verständliches Design dieser wichtig.

Allen spezifischen Anforderungen voran steht die Anforderung der leichten, möglichst intuitiven Erkennbarkeit der Zusammenhänge und der schnellen, überischtsartigen Vergleichbarkeit, die für die eingangs beschriebende Zielgruppe notwendig ist.

3.3 Präsentation der Visualisierungen

Zur Erfüllung der Anaylsen und Anforderungen werden die Visualisierungstechniken *explizites Baumdiagramm*, *parallele Koordinate* und *Scatterplot* gewählt. Nachfolgend werden die ausgesuchten Visualisierungen vorgestellt und diskutiert.

3.3.1 Visualisierung Eins

Zur Erfüllung der Anforderungen und in der Analyse gefundenen Problemstellungen wird für die erste Visualisierung ein explizites Baumdiagramm gewählt.

In der Visualisierung gibt es einen Wurzelknoten aus dem hierarchisch die Kindknoten der Publisher abgehen. Für jeden Publisher gibt es wiederum Kindknoten, die aus den vom Publisher abgedeckten Genres bestehen. Zur besseren Übersicht und Vergleichbarkeit werden nur diejenigen Genres pro Publisher abgebildet, die dieser Publisher anbietet. Die Genreknoten werden für jeden Publisher in derselben alphabetischen Reihenfolge aufgeführt. Wichtig dabei ist, dass keine sich überkreuzenden Pfade entstehen respektive es keinen Kindknoten gibt, der mehrere Elternknoten hat. Dies dient der Erfüllung der Anforderungen und der Übersichtlichkeit der Visualisierung. Die Genreknoten besitzen noch einmal Kindknoten, die die Titel der Videospiele, die dem entsprechenden Genre zugeordnet sind, tragen. So entsteht eine hierarchische Übersicht über das Angebot jedes Publishers, sodass dieser sowohl sich selbst, als auch die Konkurrenz in wenigen Blicken erfassen kann. Die Anforderungen an eine schnelle, intuitive und für jeden verständliche Übersicht werden größtenteils erfüllt. Jedoch besteht der Nachteil der Größe des Baumes durch die vielen Publisher- und Genreknoten. Trotzdem kann der Baum durch seine Nutzung der Hierarchie, also Abbbildung der nominalen Daten auf eine Position im Raum, sowie durch die Nutzung von Verbindungslinien zur Unterstützung der Positionierung und damit Erstellung des mentalen Modells der gewünschten Hierarchie die Anforderungen an Effektivität gut erfüllen. Entsprechende visuelle Variablen werden als sehr effektive Abbildungsmethodik für nominale Daten eingestuft. Expressiv und damit ohne viele Erklärungen und Missverständnisse ist diese Technik zudem. Jedoch leidet die Expressivität etwas unter der Menge an Knoten und damit der Größe des Baumes. Auch angemessen ist eine Visualsierung als explizites Baumdia-

gramm, da es wenig Ressourcen fordert.

Es gibt verschiedene Alternativen zu dieser Darstellung bestehend aus impliziten Baumdiagrammen, radialen Baumdarstellungen und hyperbolischen Bäumen. Implizite Baumdarstellungen können ebenso wie explizite die Hierarchien zwischen Eltern- und Kind-Knoten darstellen. Radiale und hyperbolische Bäume sind gute Alternativen für Darstellungen mit vielen Knoten wie es in dieser Visualisierung auch der Fall ist. Vor allem letztere Alternativen bieten in Bezug auf Expressivität, Effektivität und Angemessenheit nahezu dieselben Vorteile wie die expliziten Bäume. Letztlich wird die Entscheidung für ein explizites Baumdiagramm trotz des Nachteils der vielen Knoten, die durch hyperbolische und radiale Bäume übersichtlicher gestaltet werden könnten, aufgrund der Übersichtlichkeit durch Intuitivität der expliziten Baumdarstellungen getroffen. Dies mag paradox wirken, jedoch sind explizite Bäume durch ihr häufigeres Vorkommen in allen Lebenssituationen noch ein wenig intuitiver und damit effektiver verständlich als die Alternativen. Da die Anforderung vor allem auf schnellem Erkennen und Wiedererkennen insbesondere des eingenen Verlagshauses der Zielgruppe und deren Stakeholdern liegt, wird diesem Aspekt eine hohe Gewichtung gegeben.

3.3.2 Visualisierung Zwei

Als zweite Visualisierungstechnik zur Lösung des in der Analyse gestellten Unterziels zwei und seinen Anforderungen werden die parallelen Koordinaten gewählt.

Für diese Visualisierung werden die Datenpunkte als Abfolge von Liniensegmenten dargestellt, wobei der Attributwert der jeweilige Endpunkt des Liniensegments ist. Die Wertebereiche der Attribute sind die Liniensegmente. Die Liniensegmente sind im hier vorliegenden Fall die Wertebereiche der Attribute *North America*, *Europe*, *Japan*, *Rest of World* und *Global*, welche die Verkaufszahlen in Millionen Einheiten der Videospiele in den verschiedenen Regionen darstellen. Durch die Verbindung der Liniensegmente und den jeweiligen Endpunkten auf den parallelen Liniensegmenten bzw. Achsen kann die Zielgruppe die Attributwerte dieser fünf Attribute ablesen. Durch die Darstellung aller Videospiele als eine solche Sequenz von Liniensegmenten, aber entsprechend immer anderen Attributwerten, können Unterschiede erkannt werden.

Wie in den Anforderungen beschrieben, soll es zur besseren Vergleichbarkeit der Datenpunkte und damit Videospiele eine Filterung nach Genres geben, die mittels eines Drop-Down-Menüs umgesetzt wird. Da es bei parallelen Koordinaten zu Problemen mit dem Überzeichenen von Liniensegmenten kommen kann, wenn es zu viele Datenpunkte gibt, schafft die Filterung hierbei zudem Abhilfe.

Zur Erkennung des Videospieltitels und des Publishers, wird beim Hovern über ein Liniensegment beides eingeblendet. Zudem wird das entsprechende Segment mittels Farbe hervorgehoben, um das Segment eindeutig erkennbar zu machen. Durch dieses Designentscheidungen werden die allgemeinen Anforderungen an Expressivität und Effektivität erfüllt.

Quantitative Daten werden zudem am besten mit den visuellen Variablen Position, aber auch Orientierung dargestellt. Beides wird in den parallelen Koordinaten umgesetzt, die Orientierung

vor allem durch die Sequenz an verbundenen Liniensegmenten, die parallelen Achsen und den entstehenden Kontext der Daten.

Die Expressivität wird zusätzlich durch die eindeutige Beschriftung der Liniensegmente beim Hovern, die Einfärbung und die eindeutige Beschriftung der Achsen, die jeweils dieselbe Einheit besitzen erreicht. Auch die weiteren Anforderungen vor allem an die Vergleichbarkeit und allgemeine Übersicht über die Verkaufszahlen der Videospiele in einer leicht verständlichen Art und Weise wird erfüllt.

Zusätzlich können leicht Cluster erkannt werden, wenn bestimmte Videospiele, bspw. eine Reihe in einem Genre, alle in allen Attributen, sprich Regionen, gute Werte aufzeigen. Auch dies bietet Aufschlüsse über den Markt und entsprechende Konkurrenz.

Die konkreten Zahlen sind anhand der Achsenbeschriftung zumindest ungefähr ablesbar, jedoch überwiegt wie gewünscht das visuelle Erkennen und Vergleichen. Aus der Anforderung nach einer Vergleichbarkeit mit zumindest groben Datenwerten leitet sich zudem ab, dass keine Techniken der Datentinte verwendet werden. Dies wird vor allem durch dieselben Einheiten jeder parallelen Achse, der Vergleichbarkeit dieser dadurch und damit der intuitiven Einordnung von *hoch* und *niedrig* entlang der Achsen möglich.

Auch zu den parallelen Koordinaten gibt es Alternativen. Eine dieser Alternativen sind die Stickfigure Plots. Sie werden jedoch als weniger gut im Vergleich zu den parallelen Koordinaten eingeschätzt. Auch Section und Prosection wäre eine legitime Alternative, da auch hier mehrdimensionale Daten dargestellt werden können. Das Problem liegt Zuletzt bieten Sternkoordinaten und auch Polar Plots alternative Möglichkeiten der Darstellung mehrdimensionaler Daten. Allerdings

3.3.3 Visualisierung Drei

Zuletzt wird das dritte Unterziel und die abgeleiteten Anforderungen durch die Visualisierungtechnik Scatterplot umgesetzt.

In diesem Scatterplot wird auf die X-Achse je nach Auswahlwunsch des Betrachters je der Wertebereich der Attribute *North America*, *Europe*, *Japan*, *Rest of World* und *Global* abgebildet. Die Abbildung der Wertebereiche auf die Y-Achse funktioniert auf dieselbe Art. So kann der Anwender frei zwischen zwei Attributen seiner Wahl wählen, die er untersuchen möchte. Auch hier sind die Einheiten wieder gleich und somit gut vergleichbar. Es werden je die Datenpunkte visualisiert, die dem per Drop-Down-Menü ausgewählten Genre entsprechen.

Die Datenpunkte werden klassisch als ungefüllte Kreise dargestellt, wobei sich ihre Position aus den Werten der oben beschriebenen, momentan ausgewählten Atribute bestimmt. Beim Hovern über die Punkte werden diese eingefärbt und ein weiteres Attribut, der Titel des Videospiels angezeigt. Zur exakten Erkennung der Attributwerte der Verkaufszahlen werden diese hinter dem Titel in Klammern aufgelistet. Es wird zuerst der Wert für die X-Achse, dann der Wert für die Y-Achse dargestellt.

Durch diese Designentscheidungen werden die zuvor formulierten Anforderungen erfüllt. Die

Kodierung der Daten auf die Punkte im Scatterplot, also Koordinatensystem ermöglichen ein Wahrnehmen und damit eine Orientierung im Raum, der mental entsteht und das Verständnis für die Relation der Datenpunkte zueinander verbessert. Weiterhin ist dadurch die visuelle Darstellung von möglichen Korrelationen zwischen zwei Attributen erkennbar.

Durch die zweidimensionale Darstellung der Attribute eine genauere Betrachtung dieser erfolgen. Entsprechend kann auch die Konkurrenz noch einmal anders und möglichlerweise genauer je nach Regionseinstellung wahrgenommen werden. Durch die Einblendung der exakten Attributwerte ist auch diese Anforderung erfüllt.

Die eher klassische Darstellung als Punkte erleichtert das Erkennen von Punktwolken respektive Cluster durch Intuition, insbesondere sollte es Korrelationen zwischen einer Region und den globalen Verkaufszahlen geben. Ähnlich wie in Visualisierung zwei werden auch in Scatterplots die für quantitative Daten gut geeigneten visuellen Variablen Position, Orientierung und Gebiet genutzt.

Weiterhin ist diese Darstellungsweise allgemein bekannt, sodass eine schnelle Analyse einzelner Wertpaare, aber auch Korrelationen und Ausreißer möglich ist. Dadurch und durch die oben genannten Aspekte ist die Effektivität gegeben. Auch expressiv ist diese Visualisierung, da die Darstellung der Datenpunkte als Kreise auch ohne Legende schnell erkannt wird. Durch die Einblendung der exakten Werte beim Hovern wird eine mögliche Verwirrung durch Überlagerung verschiedener Symbole, Farben oder Größen der Punkte verhindert. Dank der Filterung in Genre und der recht unkomplizierten Darstellung ist auch die Angemessenheit gegeben. Wie in allen drei Visualisierungen gilt hier auch die Angemessenheit durch Effektivität.

Alternativ bestehen in diesem Kontext vor allem drei weitere Visualisierungsmöglichkeiten. Diese sind Q- und QQ-Plots, Histogramme und Korrelationstabellen. QQ-Plots eignen sich zwar auch zur Darstellung von paarweisen Vergleichen und Entdeckung von Korrelationen und Ausreißern, jedoch sind sie weniger intuitiv, benötigen ein gewissen Vorwissen und sind nicht schnell erkennbar. Histogramme wiederum Eine weitere Alternative sind Korrelationstabellen.

3.4 Interaktion

Interaktion in Visualisierung 1 Tree: Highlight Knoten + Anzeigen Beschriftung Hovern -> Übersicht, Details ohne zu Überfordern; Klick auf Link zu detaillierten Plots Link zu detaillierten Plots (Grund: Übersicht zuerst und dann etwas getrennter den Rest) Interaktion in Visualisierung 2 Parallele Koordinaten: Auswahl der Regionen pro Achse -> kein Switchen, da durch so viele Möglichkeiten durch so viele Achsen unübersichtlich, aber trzd sowas sinnvol weil deshalb freie Auswahl sodass Anwender schneller und einfacher das auswählen kann was er sehen will ohne alles einmal durchzuswitchen highlight der Datenpunkte durch Farbe beim Hovern + ANzeigen Titel, Publisher, konkrete Werte beim Hovern Interaktion in Visualisierung 3 Scatterplot: Auswahl der x und y Achse mit Regionen, die verglichen werden sollen highlight Datenpunkte durch Farbe + titel, publisher, konkrete werte beim hovern (einheitlich mit visualisierung 2) Interaktion Verknüpfung 2 und 3: Auswahl der Plots mit Dropdown -> wenn Plot gewechselt

wird und dann wieder zurück, bleiben die Achseneinstellungen erhalten (Bei tree Text + Link) in jedem Plot Möglichkeit unten direkt zurücl zum Tree zu kommmen da dies ausgangsseite und Übersicht Auswahl Genre für jede Visualisierung möglich -> bei Klick auf anderen Plot wird diese auswahl auch dort gleich übernommen bzw führt dort zu einer Änderung durch filterung nach genre -> weniger einstellen, schnelleres Vergleichen

4 Implementierung

Im foglenden Kapitel wird die Implementierung der zuvor beschriebenen Visualisierungen und Interaktionen beschrieben. Der Code wird durch Kommentare ergänzt, um eine Übersichtlichkeit und Erklärung zu einzelnen Funktionen zu ermöglichen. Der Code wird in fünf verschiedene Elm-Module unterteilt, um die vollen Möglichkeiten der Sprache zu nutzen, das Programm performanter zu machen und den Code für Außenstehende übersichtlicher zu gestalten. Es existiert je ein Modul für die drei einzelnen Visualisierungen (TreeHierarchy, ParallelPlot, Scatterplot), eines für Datenstrukturen (Data) und eines zur Zusammensetzung und Ausführung des Programms (MainScatterParallel). Innerhalb der Module, insbesondere der Visualisierungsmodule, wird versucht, den Code einheitlich zu strukturieren. Hier werden zuerst kleinere Funktionen definiert, die im späteren Verlauf genutzt werden bis zuletzt allgemeine Einstellungen der Plots vorgenommen werden.

Über alle Module hinweg werden diverse Bibliotheken genutzt. Die Bilbliotheken *elm/core* und *elm/browser* werden nur für MainScatterParallel importiert. *elm/http* wird von MainScatterParallel, Data und TreeHierarchy verwendet, *elm/html* von MainScatterParallel, TreeHierarchy und Scatterplot. Zur Decodierung von JSON-Dateien für TreeHierarchy wird die Bibliothek *elm/json* in Data genutzt, für die Decodierung von CSV-Dateien für Scatterplot und ParallelPlot in Data *lovasoa/elm-csv* und *ericgj/elm-csv-decode*. Die Bibliothek *elm-community/typed-svg* wird für das Zeichnen der Plots in Scatterplot, ParallelPlot und TreeHierarchy gebraucht ebenso wie die Bibliothek *gampleman/elm-visualization*. *avh4/elm-color* wird nur in ParallelPlot und TreeHierarchy genutzt. *elm-community/list-extra* und *folkertdev/one-true-path-experiment* sind nur für das Zeichnen des ParallelPlots wichtig. Zur Berechnung des Layouts von Bäumen wird *wasdacraic/elm-tree-layout* in TreeHierarchy genutzt und *zwilias/elm-rosetree* zum Speichern der Bäume in einer Datenstruktur in TreeHierarchy und Data importiert.

Insgesamt war die Implementierung der Anwendung sehr zeitaufwändig, da vor allem die Routine im Programmieren mit Elm trotz der umfangreichen Übungsaufgaben fehlte. Die Übungsaufgaben waren eine große Stütze. Jedoch musste auch hier noch einmal nachvollzogen werden, was dort gemacht wurde. Teils kamen so noch neue Erkenntnisse und neues Verständnis für den Code, sodass sich durch die Implementierung der Projektanwendung ein zusätzlicher großer Lerneffekt einstellte. Es gab trotz der guten Vorlagen der Abgaben der Autorin zu den Übungsaufgaben einige Schwierigkeiten, die in den folgenden Unterkapiteln Erwähnung finden.

4.1 Data.elm

In diesem Modul sind nahezu alle verwendeten Datentypendeklarationen enthalten. Einzig der *type Model* und *type Msg* aus MainScatterParallel sind dort nicht vertreten. Dies ist begründet in der sinkenden Überischtlichkeit und Verständlichkeit des Codes in MainScatterParallel, sollten eben genannte, recht umfangreiche Typen in Data deklariert sein und in MainScatterParallel referenziert werden müssen.

In Data sind zunächst alle für dieses Modul benötigten Importe zu erkennen, bevor die Decodierung der CSV-Dateien mit den Funktionen *decodeGameSales*, *csvString_to_data* und *gamesSalesList* geschieht. Hier werden die einzelnen Felder der CSV-Datei als String decodiert und dann in eine Liste vom Typ GameSales geschrieben. Darauf folgt die Decodierung der JSON-Datei mittels der Funktion *treeDecoder*. Beide Decoder basieren auf denen für die Übungen sieben und neun verwendeten und wurden an die in diesem Projekt verwendeten Daten angepasst.

Es folgt die Definition der Datentypendeklarationen, die für alle Visualisierungen außer die TreeHierarchy wichtig sind. In *type alias GameSales* sind die Daten aus der CSV-Datei gespeichert, welche dann im Model in MainScatterParallel weiterverwendet werden. Die Deklarationen *type RegionType* und *type PlotType* als CustomType werden für die Auswahl der Achsen in Visualisierung zwei und drei sowie die Auswahl der anzuzeigenden Visualisierung benötigt. Sie werden im *type Model* und im *type Msg* in MainScatterParallel, im *regionFilter* in Scatterplot und im *multiDimenData* in ParallelPlot verwendet. Als nächstes werden die Datentypendeklarationen für den Scatterplot, den ParallelPlot und die TreeHierarchy definiert. Somit sind alle drei Visualisierungen modular zusammengesetzt.

Zuletzt werden für die Anwendung mit Buttons wichtige Funktionen zur Datentypkonversion von einem String zu PlotType, String zu RegionType, RegionType zu String und String zu ein einem Tupel aus RegionType und String beschrieben. Dies ist nicht nur zur Darstellung von Buttons, die nur Strings erlauben, also der Implementierung von Interaktionen nötig, sondern auch zur beispielsweise korrekten Anzeige von aktuell ausgewählten Regionen. Weiterhin ist noch eine Konversion von einem RegionType zu einem Tupel aus GameSales und Float für die Funktion *multiDimenData* in ParallelPlot notwendig.

4.2 TreeHierarchy.elm

Im Modul TreeHierarchy werden wie immer in Elm die Importe definiert. Aus Data werden nur die für die Baumdarstellung benötigten Funktionen importiert. Danach folgt die Funktion *main*, die die Bestandteile der Anwendung definiert.

Entsprechend dieser werden nun die Funktionen *init* und *update* beschrieben. Ersteres initialisiert die Anwendung und lädt die Daten unter Anwendung des Decoders für JSON-Dateien. Im *update* wird deklariert, dass im Falle eines korrekten Ladens von Daten der neue Baum angezeigt wird, im Falle eines inkorrekten Ladens eine Error-Nachricht. Es folgt eine Helferfunktionen *convert* zur Konvertierung des Baumes basierend auf den Vorgaben zu Übung neun.

Die Funktionen *line* und *point* werden zum Zeichnen der Verbindungen von Eltern zu Kindknoten beziehungsweise zur Darstellung der Knoten an sich benötigt und in *treePlot* und *treePlot2* angewandt. Sie definieren die Linie beziehungsweise den Knoten als Kreis mit Beschriftung mit einer Rotation von 75° und einer fett gedruckten Serifenschriftart.

Als nächstes sind zwei Funktionen zum Zeichen des Baumes beschrieben. Zur Ansicht des Baumes ohne Scrollen als Übersicht wird *treePlot2* verwendet, zur näheren Ansicht des Baumes wird der *treePlot* verwendet. Der einzige Unterschied zwischen den Plots ist die prozentuale Vergrößerung der global definierten Höhe und Breite um 200 bzw. 150 Prozent im SVG des *treePlot*. Die Plots an sich sind stark basierend auf dem Code der Autorin zu Übung neun. Mittels lokaler Funktionen wird zunächst das Layout des Baumes berechnet, bevor lokal die Abhängigkeiten des Baumes berechnet werden, um die Pfade zwischen Eltern- und Kindknoten zeichnen zu können. Dazu werden die X- und Y-Werte der Eltern- und Kindknoten mittels Dict.get herausgezogen. Weiterhin wird eine Verbindung des Wurzelknotens mit einem nicht existierenden Elternknoten durch die Funktionen *checkRootNegative* und *nodeValuesPath* verhindert. Zum Zeichnen der Verbindungen wird die zuvor beschriebene Funktion *line* genutzt, auf die die lokale Funktion *nodeValuesPath* angewandt wird. Für die Punkte wird *point* verwendet, auf das die lokale Funktion *nodeValues* angewandt wird. Die Definition von Kreis-, Linien- und Textfarben, Umrandungen derselben und entsprechenden Änderungen beim Hovern mit der Maus wird im global definierten CSS *cssTree* vorgenommen.

In der *view*-Funktion wird zunächst lokal der konvertierte Baum sowie das Layout berechnet. In einem Html.div werden geschachtelt durch weitere divs Überschriften, Erklärungen, Links und Buttons zur Ausgabe implementiert. Die Anordnung sowie der Designstil ist an MainScatterParallel angepasst. An den entsprechenden Stellen in der Abfolge der gewünschten Texte werden die beiden Baumhierarchien gezeichnet, wobei *treePlot* einen Balken zum horizontalen Scrollen erhält.

Zuletzt werden global allgemeine konstante Einstellungen zum Plot definiert. Hierzu zählen Breite, Höhe, Innenabstand, Radius, Skalierungen, Default-Einstellungen sowie Einstellungen zum Bereich der Datenwerte. Diese Einstellungen basieren auf den Vorgaben in der Vorlage zum Arbeiten mit Bäumen aus Übung neun und sind auch in den Abgaben zu Übung neun der Autorin zu finden.

Die Implementierung der expliziten Baumhierarchie bereitete wenig Schwierigkeit, da sehr viel auf den abgegebenen Codes der Autorin zu Übung neun basiert. Ledgilich der Decoder in Data musste an die Datenfelder angepasst, der Link zum Laden der Daten geändert sowie die besten Design- und Darstellungsweisen, inklusive der zweifachen Darstellung des Baumes, gefunden werden. Weiterhin wurde das CSS global definiert, das allgemeine Design der darzustellenden Seite im *view* an jenes aus MainScatterParallel angepasst und Erklärungstexte für die Anwender hinzugefügt.

4.3 ParallelPlot.elm

Im Modul ParallelPlot werden wieder zuerst die Importe definiert, wobei aus Data nur die für die Parallelen Koordinaten benötigten Funktionen importiert werden.

Die folgenden drei Funktionen dienen dem Mapping und der Reduzierung der Datensätze um solche, die Null-Values enthalten. Dies ist nach der manuellen Vorverarbeitung der Daten zwar nicht zwingend notwendig, jedoch werden somit mögliche Fehler in dieser berücksichtigt und entfernt. Weiterhin ist der Code damit universeller einsetzbar, sollten andere Datengrundlagen genutzt werden wollen. Die erste Funktion ist eine Hilfsfunktion, die mittels Maybe.map2 beim Pipen hilft. Sie basiert auf *https://package.elm-lang.org/packages/elm/core/latest/Maybe* Die zweite Hilfsfunktion basiert auf einer normalen Maybe.map5 Funktion, die um einen Parameter erweitert wird *https://package.elm-lang.org/packages/elm/core/latest/Maybe*. Da es kein Maybe.map6 gibt, wird hier die vorherige Hilfsfunktion angewandt, um dieses Problem zu umgehen. So kann transformiert werden wie es eine Maybe.map6 Funktion machen würde, wenn es sie gäbe. *AssignmentAndReduce* wiederum mapped schließlich lokal mithilfe der *helpMapBig* die Daten vom Typ GameSales zu Maybe GameSales und reduziert diese dann um die Datensätze mit fehlenden Werten mittels *List.filterMap*. Diese Funktion basiert auf den Abgaben der Autorin zu Übung sechs.

Die Funktion *multiDimenData* dient zusammen mit der lokalen Funktion *multiDimFunction* aus MainScatterParallel der freien Auswahl und Belegung der Achsen sowie der Anwendung der Filterung nach Genre. Das Grundgerüst beider Funktionen basiert auf den Abgaben der Autorin zu Übung sechs. Hier wurden die Achsen jedoch getauscht, was aufgrund flexiblerer und schnellerer Auswahl der Achsen in diesem Projekt geändert wurde. *MultiDimenData* benötigt als Eingabe eine Liste von GameSales sowie fünf RegionTypes. Diese sind die auszuwählenden Regionen für die Achsen. Weiterhin braucht es den Namen des Spieles und den Publisher des Spieles zur Anzeige dieser im späteren Plot. Der Typ begründet sich aus dem *type alias GameSales*. Zusätzlich sind fünf Namen, also Beschriftungen als String nötig. All dies begründet sich in der Definition der Recordfelder in *type Model* sowie dem für das *update* benötigten Datentypen *(RegionType, String)*. Als Output muss die Funktion den Typ *MultiDimData* ausgeben, der benötigt wird, um die Daten dem *scatterplotParallel* zu übergeben und den Plot zeichnen zu können. Eine besondere Schwierigkeit bei der Konstruktion der Funktion lag im Umfang dieser und im Verständis welcher Datentyp benötigt wird. So muss der RegionType innerhalb der Funktion mittels *regionTypeToAxisAnnotation* nochmals umgeschrieben werden. Die Funktion wird in MainScatterParallel in der lokalen Funktion *multiDimFunction* angewandt und bekommt dort die von ihr benötigten Daten übergeben.

Es folgt der *scatterplotParallel*, der nahezu unverändert aus den Abgaben der Autorin zu Übung sechs übernommen werden kann. Hier werden unter anderem mittels lokaler Funktionen die Achsen in X-Richtung positioniert. Weiterhin wird das umgebende Rechteck festgelegt, sowie die Achsen und ihre Beschreibung gezeichnet. Die Zeichnung der Datenpunkte wird mittels Shape.line und Shape.linaerCurve realisiert. Zudem wird der beim Hovern über einen Datenpunkt

anzuzeigende Text lokal beschrieben bevor die Funktion dann zuletzt mit den entsprechenden Daten angewandt wird.

Wie auch in der TreeHierarchy schon geschehen, wird auch hier das CSS global in *cssParallel* definiert. Ein besonderes Augenmerk liegt dabei auf der Definition der Opacity mit 0.5. So entsteht trotz weiß gefärbtem Umgebungsrechteck eine Art Röntgeneffekt. Mögliche Cluster und Aufälligkeiten in den Datenpunkten sind durch Überlagerungen so besser erkennbar. Durch das Einfärben in einem stärkeren Grünton beim Hovern und dem Einblenden der Details zum Datenpunkt lassen sich die einzelnen Videospiele besser nachvollziehen.

Auch hier schließt die Implementierung mit den generellen Einstellungen für den Plot. Diese basieren auf den Abgaben der Autorin zu Übung sechs.

Die Erstellung des ParallelPlot basierte in weiten Bereichen auf den Abgaben zu Übung sechs. So bereitete das Zeichnen des Plots wenig Schwierigkeiten. Etwas komplizierter war zunächst die Implementierung des Mappings und der Filterung nach Null-Werten durch das fehlende List.map6. Schwieriger gestaltete sich jedoch die Umsetzung der freien Auswahl der Achsen wie oben schon beschrieben. Dies war mit viel Ausprobieren und Verständnis, welcher Datentyp für welche weitere Funktion benötigt wird und welche durch die vorherige Definition der Datentypen vorhanden waren, verbunden.

4.4 Scatterplot.elm

Das Modul Scatterplot beginnt mit der Definition der Importe. Aus Data werden nur die für den Scatterplot benötigten Funktionen importiert.

Folgende drei Funktionen verfolgen denselben Zweck wie im ParallelPlot. Die ersten beiden Funktionen sind dieselben wie im ParallelPlot. *FilterAndReduceGames* ähnelt *AssignmentAndReduce* stark, basiert aber auf den Abgaben der Autorin zu Übung 1. Hier werden die Daten zunächst auch mithilfe von *helpMapBig* gemapped, jedoch vom Typ GameSales zum Typ Maybe Point. Dieser wird später für die Funktion *point* und diese für den Plot an sich benötigt. Weiterhin wird wieder mithilfe vom List.filterMap nach Null-Values gefiltert. Die gemappten und gefilterten Daten in *filter* werden dann unter anderem in XyData geschrieben, dass als Output der Funktion *FilterAndReduceGames* definiert ist. XyData werden benötigt für das Zeichnen des Scatterplots, da sie unter anderem eine Liste Daten des Typs *Point* enthalten.

Die Funktion *regionFilter* filtert mit Eingabe einer Liste GameSales und des RegionTypes die Datenpunkte nach den je gewünschten Regionsauswahlen für die Achsen. Ausgegeben wird eine Liste Floats. Diese Funktion ist nötig zur Interaktion innerhalb des Plots, um die Achsen frei wählen zu können und damit auch die richtigen Datenpunkte anzuzeigen. In MainScatterParallel wird *regionFilter* lokal für *valuesX* und *valuesY* aufgerufen und auf die nach Genre gefilterten Daten sowie die X- bzw. Y-Achse angewandt. Die Funktion basiert auf den Abgaben der Autorin zu Übung vier.

In der Funktion *point* wird die Positionierung der Punkte sowie deren Beschriftung festgelegt und als SVG Nachricht ausgegeben. So kann *point* im *scatterplot* aufgerufen werden.

Der *scatterplot* ist für das Zeichnen des Plots mit den Datenpunkten zuständig. Mit der lokalen Funktion *pointsXY* wird die Variabilität der Achsen und damit X- und Y-Werte der Punkte ermöglicht, die für die Interaktivität der Achsenauswahl nötig ist. Dies ist in vielen Übungsabgaben der Autorin nicht umgesetzt, jedoch in Übung vier, sodass diese als Grundlage für die lokale Funktion dient. In weiteren lokalen Funktionen wird die Abbildung bzw- Umrechnung der Daten auf SVG berechnet. Im SVG des *scatterplot* wird die Positionierung der X- und Y-Achse, ihrer Beschriftungen und der Punkte festglegt. Letzteres wird durch die lokalen Funktionen *xScaleLocal*, *yScaleLocal* und *pointsXY* sowie der zuvor beschriebenen globalen Funktion *point*. Der *scatterplot* basiert auf den Abgaben der Autorin zu Übung 1.4 sowie Übung vier.

Als letztes werden allgemeine Einstellungen, die konstant für den Plot gelten sollen, global definiert. Diese basieren auf den Abgaben der Autorin zu Übung eins bis vier.

Die Implementierung des Scatterplots verlief in weiten Teilen problemlos, wenn auch nicht schnell durch ständiges Überprüfen der richtigen Datentypen. Schwieriger war wie beim ParallelPlot auch die Umsetzung des Mappings und Filterns nach Null-Werten durch das fehlende List.map6. Weiterhin musste beachtet werden, dass die Achsen des Plots frei wählbar, also variabel definiert sind und nicht strikt festgelegt wie in den meisten Übungsserien. Mit entsprechender Zeit- und Denkinvestition war auch dies überwindbar.

4.5 MainScatterParallel.elm

Auch das Modul MainScatterParallel definiert zunächst die Importe. Aufgrund der besseren Erkenntlichkeit aus welchen Modulen welche Funktionen stammen, werden keine Funktionen dezidiert importiert, sondern nur das allgemeine Modul.

Für dieses Modul wird wie in TreeHierarchy die gesamte Elm-Architektur verwendet, da es den Scatterplot und die Parallelen Koordinaten ausführen und alle drei Visualisierungen verbinden soll. Hierzu wird das Programm zunächst wieder in der *main*-Funktion definiert, sowie die *subcritpions*- und die *init*-Funktion implementiert.

Danach folgt die Definition der Datentypendeklarationen *type Model* und *type Msg*. Diese könnten zwar auch in Data definiert werden, verweilen aber zugunsten der Übersichtlichkeit und schnelleren Verständnisses des Codes in diesem Modul. In *type Model* werden die drei Varianten Error, Loading und Success beschrieben, die in den entsprechenden Fällen angezeigt werden. Die Variante Success ist gleichzeitig ein Record aus den dort beschriebenen Feldern. Wichtig zu beachten ist, dass die initialen Werte erst in der *update*-Funktion hier hinein geschrieben werden. Im Feld data können die geladenen Daten als Liste vom Typ GameSales gespeichert. Zur Ermöglichung der Interaktionen gibt es weiterhin Felder für das Genre als String, die Achsen eins bis fünf als RegionType, die Namen der Achsen eins bis fünf als String, die X- und Y-Achse als RegionType und den gewünschten Plot als PlotType. Der *type Msg* definiert die Varianten für die Nutzung in *update* zur Überschreibung des Models. Es gibt eine Variante zum erfolgreichen Laden der Daten und damit der Initialisierung des Plots sowie für jeden möglichen Wechsel von Achsen in den Plots sowie den Wechsel des Plots an sich.

In der *update*-Funktion werden zunächst die initial anzuzeigenden Attributwerte definiert und in Success gespeichert. Hierbei muss auf die entsprechenden Datentypen geachtet werden, die in *type Model* unter Success gespeichert werden. Außerdem wird hier die Decodierung der geladenen Daten vorgenommen. Weiterhin werden für alle möglichen Interaktionen mit dem Model die Varianten von *Msg* eingefügt und beschrieben, wie bei einer Änderung dieser Variante durch Klick auf deinen Button das Model überschrieben wird.

Als nächstes werden alle benötigten Buttons als Funktionen implementiert, sodass das Model bei Klick auf eine Variante überschrieben werden kann. Da die Buttons nur Strings akzeptieren, die Varianten von *Msg* jedoch die CustomTypes bzw. die Tupel aus CustomType und String, muss hier bei allen Buttons außer dem *buttonGenreType* die entsprechende in Data definierte Umwandlungsfunktion genutzt werden. Bei dem *buttonGenreType* ist dies nicht nötig, da die Variante von *Msg* auch einen String benötigt. Die Änderungen werden dann in der Variante des *Msg*-Typen gespeichert und durch die *update*-Funktion das Model überschrieben. Die Buttons basieren auf den Abgaben der Autorin aus Übung vier.

Die Funktion *filterGenre* beruht auf den Abgaben der Autorin zu Übung vier. Mit dieser Funktion kann der Anwender nach dem Attribut Genre filtern und ein für ihn sehenswertes Genre auswählen. Im Prinzip sorgt diese Funktion, wenn später im *view* angewendet, dafür, dass ausgewähltes Genre und angezeigtes Genre übereinstimmen, also Gleichheit besteht.

In der *view*-Funktion werden nun die verschiedenen Fälle des Models behandelt. Für diese Erklärung wird nur auf den Fall des Erfolges beim Laden eingegangen. Zunächst wird eine *let-in*-Konstruktion auf erster Ebene erstellt. In lokalen Funktionen wird hier *gameSalesData* sowie die Anzahl der Spiele insgesamt beschrieben. Zusätzlich wird auf dieser oberen Ebene der globale *filterGenre* angewandt, mit dem die Daten so gefiltert werden, dass dem Anwender nur sein gewünschtes Genre angezeigt wird. Weiterhin wird die Länge dieser neuen Liste, also die Anzahl der Spiele im ausgewählten Genre berechnet. Damit nun zwischen den Plots gewechselt werden kann, werden die verschiedenen Varianten bzw. Fälle des Datentyps *PlotType* im *in*-Teil angelegt. Deshalb befindet sich auch das Feld *plot* im Record der Variante *Success* des Models. Je nachdem, welcher Plot mittels Button nun ausgewählt wurde und wie das Model durch das *update* überschrieben wurde, wird nun einer der drei Fälle angezeigt. Der Filter bzw. diese Interaktionsmöglichkeit nach Genre soll nun in allen Plots übernommen und angewandt werden, ohne dass bei jedem Plotwechsel das Genre neu ausgewählt werden muss. Der Zustand des Models mit dem gefilterten Genre soll also bestehen bleiben. Dazu dient zum einen die zuvor beschrieben Variantendefinition im *in*-Teil der äußeren *let-in*-Kosntruktion sowie jeweils eine weitere *let-in*-Konstruktion in den jeweiligen Falldefinitionen. Diese befindet sind nun auf zweiter bzw. innerer Ebene, sodass die in der äußeren Konstruktion berechneten Zustände, also Genreauswahlen, auch hier gelten und auch bei einem Wechsel des Plots übernommen werden. Um dieses Ziel zu erreichen müssen in den lokalen Funktionen der inneren Konstruktion der Plots die lokalen Funktionen der äußeren Ebene auch angewandt werden. Würde in jeder der inneren Ebenen für jeden Plot die Filter nach Genre definiert werden und lokal angewandt statt

in der übergeordneten Ebene, würden die Einstellungen repsektive der Zustand des Models nicht übernommen werden.

Im ersten Case wird der ParallelPlot behandelt. Lokal werden die gesamten Daten nun mittels der *assignmentAndreduce*-Funktion gemapped und reduziert, sowie die entstehende Listenlänge berechnet. Auch die in der äußeren Ebene nach Genre gefilterten Daten werden auf dieselbe Art behandelt. In *multiDimFunction* wird nun *multiDimenData* letztlich angewandt und die entsprechenden Daten dafür übergeben. Als erste Liste vom Type GameSales wird die von Null-Werten bereinigte und nach Genre gefilterte *clearedGameSalesData* genutzt. Im *in*-Teil der inneren Ebene wird per Html und vielen verschachtelten divs und Attributeeinstellungen das Aussehen der Seite festgelegt. Hierbei wird darauf geachtet, dass zur angenehmeren und weniger ablenkenden Ansicht die Farben den Grüntönen in den Plots angepasst werden. Zur Direktion des Auges über die Seite und verbesserten Erkennung von wichtigen Abschnitten wie der Auswahl der Plots, des Genres und der Achsen werden Ränder und verschiedene Grüntonabstufungen genutzt. Wichtiges wird mit deutlichen Änderungen gekennzeichnet, alles andere zur impliziten Lenkung mit sogenannten *Just noticable differences* in der Farbe und im Rand. Neben den Informationstexten und Überschriften werden hier auch die Buttons als Dropdown zum Wechsel des Plots, zur Änderung des Genrefilters sowie der Anpassung der Achsen eingefügt. Zudem wird transparent angezeigt, nach welchem Schritt wieviele Videospiele in der Auswahl verbleiben und welche Einstellungen gerade aktuell getroffen sind. Nach Anzeige der ausgewählten Einstellungen wird der Parallele Koordinaten Plot implementiert. Hierzu wird dem *scatterplotParallel* das *cssParallel* übergeben sowie zwei Konstanten für die Höhe und die Aspect Ratio. Schließlich wird die zuvor lokal berechnete *multiDimFunction* vom Typ MultiDimData übergeben. Zuletzt wird noch ein Text mit Link direkt zurück zur TreeHierarchy anbgeboten.

Im zweiten Case wird der Scatterplot behandelt. Das Vorgehen mit den lokalen Funktionen zur Anwendung der Filter und des Mappings ist ähnlich zum ParallelPlot. Durch die lokalen Funktionen *valuesX* und *valuesY* werden dem *regionFilter* die nach Genre gefilterten Daten als Liste von GameSales sowie die RegionTypes der X und Y-Achse übergeben. Somit wird der Filter nach Regionen hier angewandt und die Interaktion mit freier Auswahl der Achsen kann funktionieren. Der *in*-Teil der Plotvariante Scatterplot ähnelt dem des ParallelPlots stark und unterscheidet sich nur in einigen Informationstexten sowie der Auswahl der Achsen. Weiterhin wird hier dem *scatterplot* der *cssPoint* als String, die zuvor berechneten *gameSalesDataCleared* als XYData sowie die *valuesX* und *valuesY* als Liste von Floats übergeben. Um die Beschriftung der Achsen zu ermöglichen, werden diese noch mittels der Umrechnung von RegionType zu String als String hinzugefügt.

Da die TreeHierarchy in *TreeHierarchy.elm* implementiert ist, muss für diese Plotvariante nur das Html definiert werden. Auch dies geschieht wieder mit denselben Designs wie zuvor. Weiterhin besteht hier die Möglichkeit durch einen Link zum eigentlichen Plot des Baumdiagramms zu gelangen. Da für das Baumdiagramm der Genrefilter irrelevant ist, muss er weder hier noch in TreeHierarchy angewandt werden.

Die Implementierung dieses Moduls war einerseits leicht, andererseits kompliziert. Die zuvor jeweils in eigenen Anwendungen getrennten Visualisierungen mussten zusammengefügt werden. Dabei wurden alle Teile, die nur für den jeweiligen Plot genutzt werden, in den entsprechenden Modulen belassen. Alles konnte stückweise kopiert und in einheitlichen Funktionen zusammengesetzt werden. Besonders die Erstellung einer einheitlichen *update*-Funktion war zeitlich durch die Menge an Interaktionsmöglichkeiten aufwändig. Zudem war es anstrengend, alles gleichzeitig im Blick zu behalten. Weiterhin schwieriger war die Implementierung der Buttons für den Parallel-Plot. Zunächst waren diese Buttons je einzeln und nicht als Dropdown-Menü wie gewünscht möglich. Durch einige Denkarbeit, Ausprobieren vor allem auch mit der Funktion *multiDimenData* und verschiedenen Konvertierungsfunktionen konnten sie doch wie gewünscht realisiert werden. Auch hier musste zudem zur Überprüfung immer wieder zeitaufwendig das *update* angepasst werden. Die Implementierung der Interaktion über die Plots hinweg durch die Verschachtelung des *view* mit Definition und erster Anwendung des Genrefilters auf äußerer Ebene, der Einführung des Datentypen *PlotType* und der Anwendung im *update* und *view* war ein Zufallsfund. Zunächst wurden beide Plots so implementiert, dass sie untereinander gezeichnet wurden und es noch keinen Selektor für den Plot gab, entsprechend nur eine *let-in*-Kontruktion nötig war. Hier wurde die Filterung des Genres logischerweise für beide Plots übernommen, jedoch entsprach diese Darstellung nicht den Wünschen der Autorin. Nach schrittweisem Hinzufügen der Auswahl des Plots und allen zugehörigen Codeteilen, wurde durch Ausprobieren entdeckt, dass eine Übernahme des Modelzustands durch die Schachtelung der *let-in*-Konstruktionen und der Anwendung des Variantenwechsels für den Plot möglich war. Hier war es im Verlauf kompliziert, den Überblick über die Schachtelungen sowie außen und innen definierten lokalen Funktionen zu behalten.

5 Anwendungsfälle

Präsentieren sie für jede der drei Visualisierungen einen sinnvollen Anwendungsfall in dem ein bestimmter Fakt, ein Muster oder die Abwesenheit eines Musters visuell festgestellt wird. Begründen sie warum dieser Anwendungsfall wichtig für die Zielgruppe der Anwenderinnen ist. Diskutieren sie weiterhin, ob die oben beschriebene Information auch mit anderen Visualisierungstechniken hätte gefunden werden können. Falls dies möglich wäre, vergleichen sie die den Aufwand und die Schwierigkeiten ihres Ansatzes und der Alternativen.

5.1 Anwendung Visualisierung Eins

5.2 Anwendung Visualisierung Zwei

5.3 Anwendung Visualisierung Drei

6 Verwandte Arbeiten

Führen sie eine kurze Literatursuche in der wissenschaftlichen Literatur zu Informationsvisualisierung und Visual Analytics nach ähnlichen Anwendungen durch. Diskutieren sie mindestens zwei Artikel. Stellen sie Gemeinsamkeiten und Unterschiede dar.

7 Zusammenfassung und Ausblick

Fassen sie die Beiträge ihre Visualisierungsanwendung zusammen. Wo bietet sie für die Personen der Zielgruppe einen echten Mehrwert.

Was wären mögliche sinnvolle Erweiterungen, entweder auf der Ebene der Visualisierungen und/oder auf der Datenebene?

Anhang: Git-Historie

Literatur

[1] SID_TWR. *Video Games Sales Dataset: Video Games Sales & Game Ratings Data Scraped from VzCharts*. URL: `https://www.kaggle.com/datasets/sidtwr/videogames-sales-dataset?select=XboxOne_GameSales.csv`.