

FREIE UNIVERSITÄT BERLIN

Anwendung von Algorithmen

Graphenzeichnen: Game of Thrones

Teilnehmer:

Adrian Defèr,
Timo Haffner,
Lidia Krus,
Cedric Laier,
Florian Mercks,
Markus Alexander Mies,
Peter Strümpel,
Othman Watad

bei
Prof. Dr. Günter Rote

Inhaltsverzeichnis

1 Einleitung	2
1.1 Aufgabenstellung und Zielsetzung	2
1.2 Vorgehensweise und Organisation	3
1.3 Meilensteine	4
1.4 Rollenverteilung	5
2 Analyse des Graphen	6
2.1 Datenquellen	6
2.2 Statistische Auswertung der Knoteneigenschaften	6
2.2.1 Knoteneigenschaften	6
2.2.2 Familienhäuser	7
2.2.3 Gruppen	8
2.2.4 Status	8
2.3 Statistische Auswertung der Kanteneigenschaften	9
3 Evaluierung von Frameworks	10
3.1 PyGraphML	11
3.2 D3JS	12
3.3 Gephi	12
4 Entwicklung der finalen Lösung	14
4.1 Iteration 1	14
4.2 Iteration 2	15
4.3 Iteration 3	16
4.4 Iteration 4	17
4.5 Iterationen 5 und 6	18
4.6 Iterationen 7 und 8	18
5 Zusammenfassung und Ausblick	18

1 Einleitung

Dieses Dokument ist der Abschlussbericht zu dem Softwareprojekt “Graphenzeichnen: Game of Thrones”. Die Arbeit am Projekt wurde von den Teilnehmern der Lehrveranstaltung “Softwareprojekt: Anwendung der Algorithmen” im Sommersemester 2018 am Fachbereich für Informatik der Freien Universität Berlin durchgeführt. Der Projektbericht beschreibt die Aufgabenstellung, die Planung, das Umfeld sowie den Ablauf des Projekts. Des Weiteren werden die erzielten Ergebnisse ausführlich beschrieben. Schließlich werden der Nutzen und die Verwertbarkeit der Ergebnisse diskutiert.

Im Rahmen des Softwareprojekts “Anwendung von Algorithmen” sollte eine Software zum Zeichnen eines bestimmten Graphs entwickelt werden. Als Aufgabenstellung dient der Zeichenwettbewerb des International Symposium on Graph Drawing and Network Visualization. Bei erfolgreicher Teilnahme an diesem Softwareprojekt soll an diesem Wettbewerb teilgenommen werden.

Nach Auswahl der zwei vorgegebenen Aufgabenstellungen haben wir uns für die Erstellung eines möglichst übersichtlichen Graphen entschieden, der die Beziehungen zwischen den Charakteren der bekannten Serie “Game of Thrones” darstellt. Vorgegeben ist ein Datensatz in Form einer GraphML-Datei, welcher mit einer Programmiersprache unserer Wahl grafisch dargestellt werden soll.

Unser Team bestand anfangs aus 10 Teilnehmern und hat sich im Verlauf der Projektphase auf 9 reduziert.

1.1 Aufgabenstellung und Zielsetzung

Für den erfolgreichen Abschluss des Softwareprojekts „Anwendung von Algorithmen: Graphenvisualisierung“ haben wir uns einen Graphen erarbeitet, der die folgenden Anforderungen erfüllt:

- Eine Visualisierung der Grafik in einem Format Ihrer Wahl.
- Eine kurze Beschreibung, wie die Grafik und das Layout erstellt wurden.
- Ein webfreundliches Bild Ihrer Visualisierung. Maximale Auflösung 1000 x 1000 Pixel.

- Eine PDF-Version Ihrer Visualisierung, geeignet für den Druck auf einem großen A0-Poster.
- Eine Beschreibung wie der Graph erstellt wurde
- Abgabe eines Projektberichts bis zum Semesterende (21.07.2018)

1.2 Vorgehensweise und Organisation

Im Rahmen des Softwareprojekts haben wir versucht einer agilen Arbeitsweise zu folgen, die sich an Scrum orientiert. In gemeinsamen Brainstorming-Sessions haben wir notwendige Issues in unserem Gitlab-Repository erstellt, die daraufhin den einzelnen Bearbeitern zugewiesen wurden. Über das Git-interne Kanban-Board haben wir die Issues visualisiert und einen Überblick über deren Dringlichkeit und Bearbeitungsstatus erhalten. Unsere Projektfortschritte haben wir in wöchentlichen Sprints angestrebt und diese daraufhin in Meetings evaluiert. Dabei haben wir uns folgender Organisations- und Kommunikations-Tools bedient, um einen ortsunabhängigen Workflow und Meeting-Zyklus zu gewährleisten:

- Slack zur textbasierten Kommunikation in Gruppenchats mit Abstimmungsfunktion
- Google Hangouts für wöchentliche Meetings (audiovisuell)
- Google-Drive als Ablageort für Projektdokumente
- Gitlab als Code Repository und Projektmanagement Tool unter Verwendung der weiteren integrierten Tools:
 - Issue Tracker
 - Kanban Board

In Abbildung 1 ist das von uns verwendete Kanban-Board dargestellt. Der Workflow ist in die vier Zustände Backlog, In Progress, Code Review und Closed unterteilt. In Backlog werden neue Aufgaben gesammelt. Die Spalte In Progress beinhaltet die Aufgaben, die aktuell bearbeitet werden. Alle Tickets die bearbeitet wurden, sind zunächst der Spalte Code Review untergeordnet, bis der Code mit einem anderen Projektteilnehmer durchgesehen und mit dem Master-Branch gemerged wurde.

Anwendung von Algorithmen

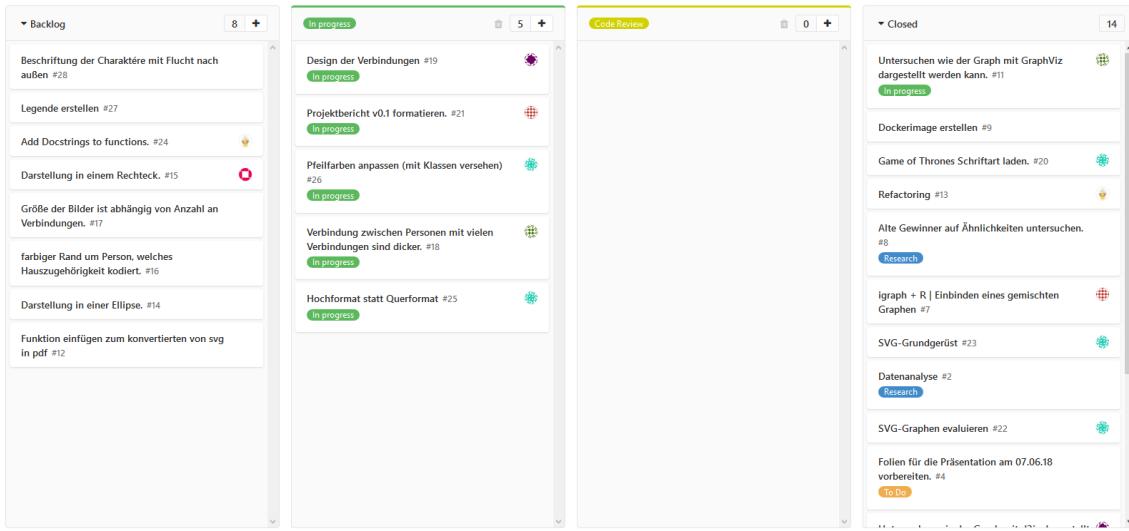


Abbildung 1: Screenshot des von uns verwendeten Kanaban-Boards, das in GitLab integriert ist. Zu sehen sind die vier Spalten Backlog, In Progress, Code Review und Closed.

1.3 Meilensteine

Um den Projektaufwand in sinnvolle Zeitabschnitte aufzuteilen haben wir den Projektverlauf in mehrere Meilensteine aufgeteilt. Somit waren wir in der Lage unseren Arbeitsfortschritt mit der zeitlichen Begrenzung des Projektes abzuwegen.

- M1: Analyse verfügbarer Bibliotheken und Frameworks und Wahl einer Programmiersprache.
- M2: Evaluierung von Frameworks / Libraries
- M3: Projektvorstellung
- M4: Datenanalyse
- M5: Zwischenpräsentation - Erster Prototyp
- M6: Finetuning / Optimization
- M7: Finale Version
- M8: Abschlusspräsentation

Anwendung von Algorithmen

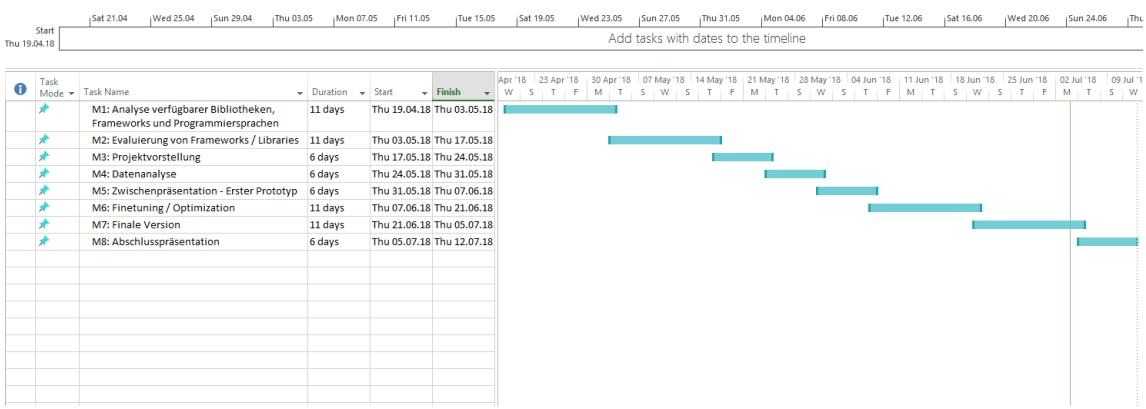


Abbildung 2: Meilensteinplanung in Form eines Gantt-Diagramms mit festgelegten Zeiträumen zur Fertigstellung

1.4 Rollenverteilung

Die Rollenverteilung innerhalb des Projektteams wurde in Folge der ein- bis zweiwöchentlichen Meetings bestimmt. Anfangs sind wir einem explorativen Ansatz gefolgt, in dem sich jedes Teammitglied in eine vielversprechende Technologie eingearbeitet hat. Nachdem wir in der Zwischenpräsentation erste Erfolge unter Verwendung eines geeigneten Tools erlangt haben sind einige, als Entwickler erfahrene Mitglieder, dazu übergegangen einen Prototypen zu entwerfen und weitere Aufgaben zusammen mit den restlichen Teammitgliedern aufzuteilen.

Dabei haben Timo Haffner und Lidia Krus eine tragende Rolle in der Planung, Aufteilung, Durchführung und Evaluation richtungsweisender Projektentscheidungen übernommen.

Die technischen Entwicklungsschritte zur Erzeugung der vorliegenden Codebasis wurde hauptsächlich von Timo Haffner, Peter Strümpel, Adrian Defèr und Lidia Krus vorgenommen, wobei die restlichen Teammitglieder bei Recherchearbeiten, der Auswahl, wie auch Anwendung von Tools und technischen Planungen beteiligt waren.

Die Erstellung des Projektberichts wurde von Lidia Krus initiiert und im weiteren Projektverlauf von Florian Mercks fortgeführt, wobei sich alle Teammitglieder an der Fertigstellung des Abschlussberichts mit Verbesserungsvorschlägen und deren Durchführung beteiligt haben.

Für die graphische Ausgestaltung waren Adrian Defèr und Cedric Laier verantwortlich.

Das Aufsetzen und die Konfiguration von Slack/gitlab zur Kommunikation wurde durch Peter Strümpel durchgeführt. Der Review-Prozess wurde von Othman Watad in Verbindung mit weiteren Teammitgliedern unterstützt.

2 Analyse des Graphen

In diesem Abschnitt wird auf die Analyse der Daten des Graphen eingegangen. Zudem wird darauf eingegangen welche weiteren Daten beschafft wurden.

2.1 Datenquellen

Der GraphML-Datensatz, welcher die gesamten Knoten- und Kantenverbindungen zwischen den Charakteren beschreibt ist [hier](#) verfügbar.

Um den uns vorliegenden Datensatz in seiner Zusammensetzung besser verstehen zu können, wird in den folgenden Abschnitten auf die Datenanalyse mit den Eigenschaften der Knoten und Kanten genauer eingegangen.

Zudem wurden die Character-Portraits von [hier](#) beschafft, um diese in einem späteren Iterationsschritt für die Knoten unseres Graphen zu verwenden.

2.2 Statistische Auswertung der Knoteneigenschaften

2.2.1 Knoteneigenschaften

Die Knoten des Datensatzes werden durch folgende Eigenschaften beschrieben:

Property	Description
name	Required
status	Alive Deceased Uncertain
house-birth	Affiliation to a house by birth
house-marriage	Affiliation to a house by marriage
group	Affiliation to a group that is not a house

Tabelle 1: Knoteneigenschaften des Graphen mit einer Beschreibung der möglichen Inhalte.

2.2.2 Familienhäuser

Die insgesamt 22 Familienhäuser mit ihrem jeweiligen Haus (house name) bestehen aus Mitgliedern, die entweder in das Familienhaus hineingeboren worden sind (house-birth) oder die, die in das Familienhaus eingeheiratet haben (house-marriage) (vgl. Tabelle 2).

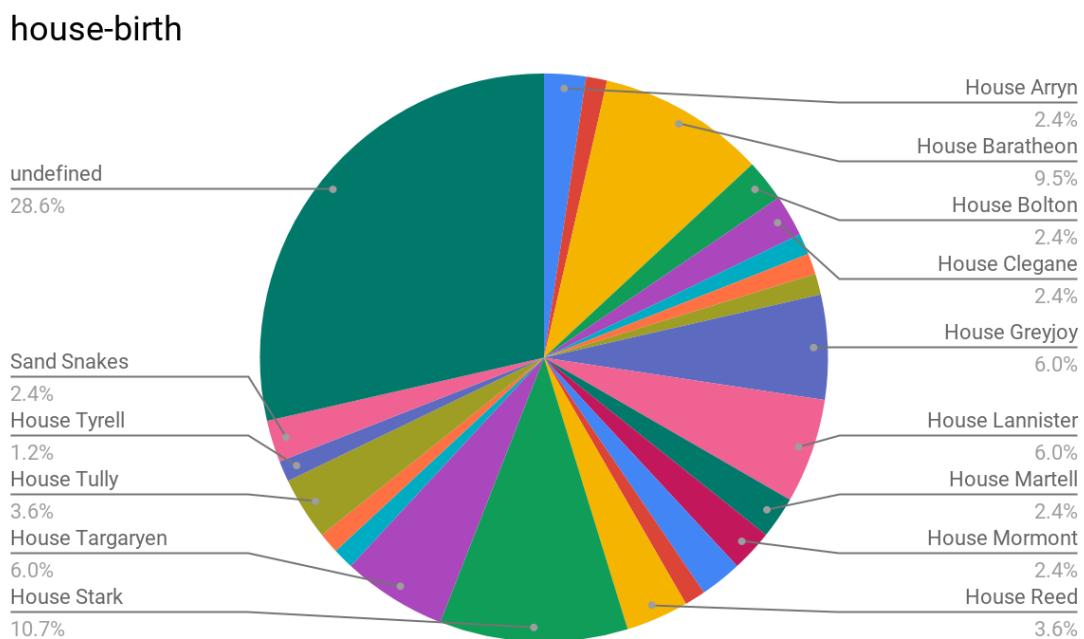


Abbildung 3: Prozentuale Verteilung der Hauszugehörigkeit aufgrund der direkten Verwandschaft.

house name	house-birth	house-marriage
House Arryn	2	2
House Baelish	1	
House Baratheon	8	1
House Bolton	2	
House Clegane	2	
House Dayne	1	
House Dondarrion	1	
House Frey	1	
House Greyjoy	5	
House Lannister	5	
House Martell	2	
House Mormont	2	
House Payne	2	
House Redwyne	1	
House Reed	3	
House Stark	9	1
House Targaryen	5	3
House Tarth	1	
House Thorne	1	
House Tully	3	
House Tyrell	1	1
Sand Snakes	2	
undefined	24	76

Tabelle 2: Gruppierung der Charaktere nach Häusern.

2.2.3 Gruppen

Insgesamt setzen sich alle Personen aus fünf (Interessens-) Gruppen zusammen, wie auch einem Rest undefinierter Mitglieder, die offiziell keiner Gruppe angehören. Die Informationen, die sich daraus schließen lassen, sind weniger interessant, da sie unvollständig zu sein scheinen und nur aus je 1 bis 3 Personen bestehen.

2.2.4 Status

Der Status einer Person beschreibt ihren Daseinszustand. So kann eine Person entweder am Leben, verstorben oder ihr Zustand kann aktuell unbekannt sein. In Ab-

Abbildung 4 ist die Häufigkeitsverteilung diesbezüglich dargestellt.

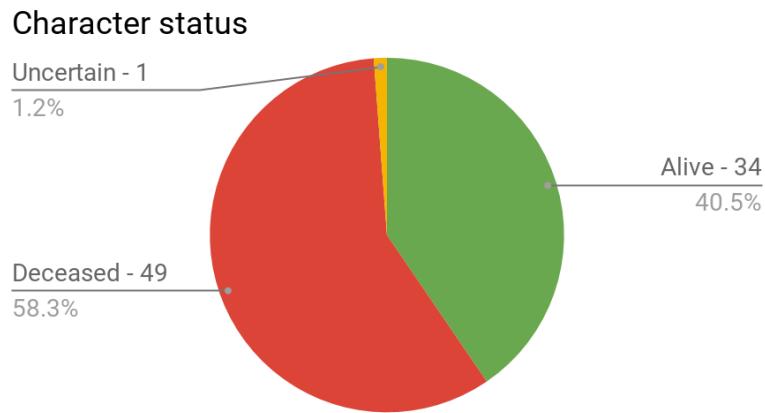


Abbildung 4: Gegenüberstellung der Anzahl verstorbener und lebender Charaktere mit der Angabe der absoluten Personenzahl über der Linie und der prozentualen Angabe unter der Linie.

2.3 Statistische Auswertung der Kanteneigenschaften

Die Kanten des Datensatzes werden durch folgende Eigenschaften beschrieben:

textbf{Property}	textbf{Description}
sibling	biological
allegiance	Direwolf Ward Kingsguard Queensguard ... Pledge biological
killed	killed
father	biological legal
mother	biological
spouse	spouse
lover	lover

Tabelle 3: Mögliche Kanteneigenschaften des Graphen mit Beschreibung der Inhalte.

Die Relationen Allegiance und Father lassen sich in Typen unterteilen. Die restlichen Relationen sind einfacher Natur ohne komplexe Untertypen.

In Abbildung 5 ist die Mächtigkeit aller Kantenbeziehungen im Graphen dargestellt.

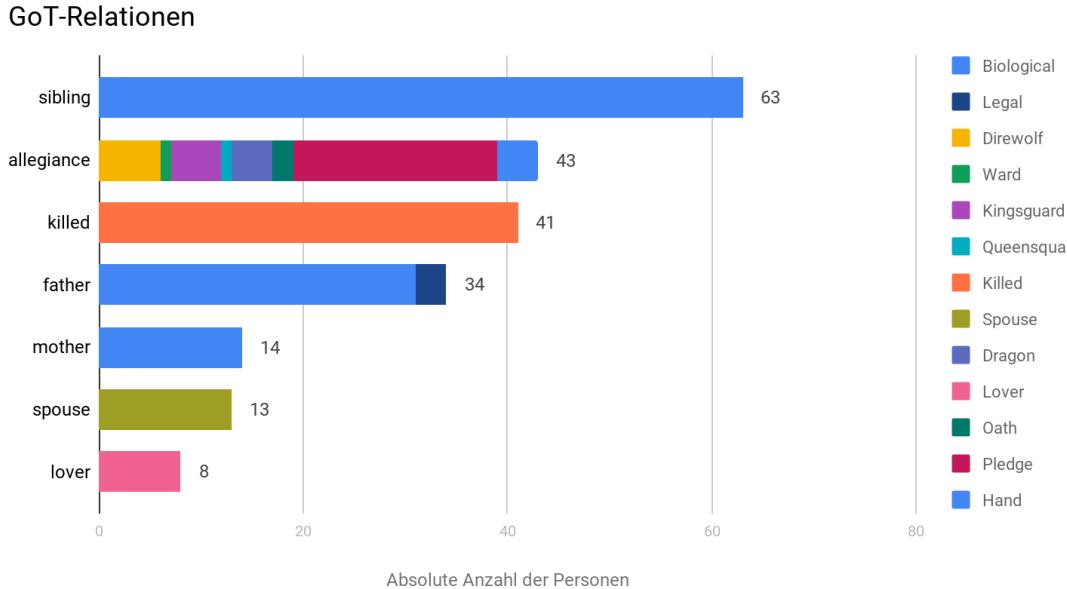


Abbildung 5: Anzahl der Entitäten, der zwischen Charakteren vorkommenden Relationen.

Die Gruppe der Geschwister stellt mit 63 Relationen die Größte dar. Wir haben 16 abgeschlossene Geschwistergruppen ausfindig gemacht, die meistens aus 2-3 Mitgliedern bestehen. Zwei größere Gruppen in diesem Zusammenhang verfügen über 6 bzw. 7 Mitglieder. Dies sind die Geschwister Stark und Tiere (Werewolves). Die Familie Stark tritt hier als Sonderfall auf, da in dieser Familie nicht alle Geschwister miteinander in sich eine geschlossene Kantenbeziehung aufweisen.

3 Evaluierung von Frameworks

In der zweiten Phase unseres Projekts haben wir verschiedene Frameworks für unseren Gebrauch evaluiert. Auf jedes dieser Frameworks gehen wir nachfolgend ein und zeigen die Vor-, wie auch Nachteile für unser spezielles Problem auf.

Bei der Wahl eines hilfreichen Frameworks, wie auch der passenden Programmiersprache fiel unsere Wahl schnell auf Python und JavaScript. Plotly (Python) erfreut sich allgemeiner Beliebtheit, ist gut dokumentiert und effektiv, erfordert jedoch das Überwinden anfänglicher Einarbeitungshürden. Aus diesem Grund bot sich zuerst

Pygraphml, ebenfalls in Python, in Verbindung mit den auch ansonsten oft genutzten Frameworks NetworkX und Matplotlib an. Ein alternativer Ansatz war es d3js (JavaScript) zu nutzen. Im Folgenden sind die Links zu den von uns ins Auge gefassten Frameworks aufgeführt:

- <https://plot.ly/> - Plotly (Python)
- <http://networkx.github.io/> - NetworkX (Python)
- <https://d3js.org/> - D3.js (JavaScript)
- <http://visjs.org/> - vis.js (JavaScript)
- <https://matplotlib.org/> - matplotlib (Python)
- <http://hadim.fr/pygraphml/index.html> - PyGraphML

3.1 PyGraphML

In diesem ersten Versuch verwendeten wir PyGraphML zusammen mit NetworkX und Matplotlib und erhielten direkt einen ersten, jedoch nur sehr simplen Plot (Abbildung 6). Bei der weiteren Recherche und Betrachtung der Gallerie-Bilder auf den Webseiten der hier verwendeten Frameworks wurde uns schnell klar, dass diese Technologie unseren Ansprüchen an Komplexität und Ästhetik nicht gerecht werden würde.

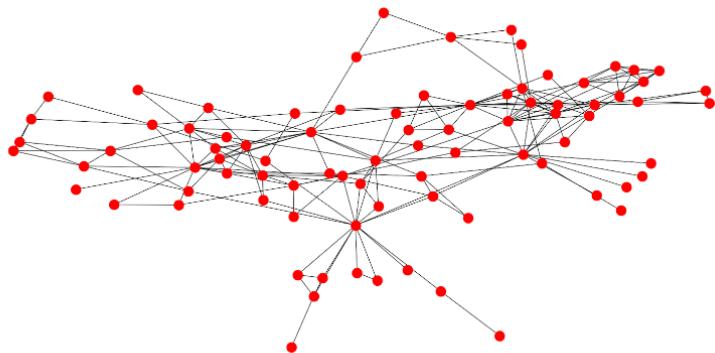


Abbildung 6: Mithilfe von NetworkX und Matplotlib erstellter Testplot des Datensatzes.

3.2 D3JS

Ein motivierender Zwischenschritt in der Projektentwicklung war der Versuch den Graphen mit D3JS (Javascript) darzustellen. Hier konnten wir erstmals die vorher gescrapten Character-Portraits anstelle der einfarbigen Knoten verwenden, wie auch farbige Kantenbeziehungen. Negativ aufgefallen ist uns hierbei jedoch die Abwesenheit der out-of-the-box Funktionalität, insofern es uns nicht möglich war die Browseransicht des Graphen zu exportieren. Beim Versuch einen PDF-Export vorzunehmen wurden leider alle Bilder in der linken oberen Ecke aufeinander gestapelt, während der eigentliche Graph hingegen gänzlich ohne Bilder verblieb.

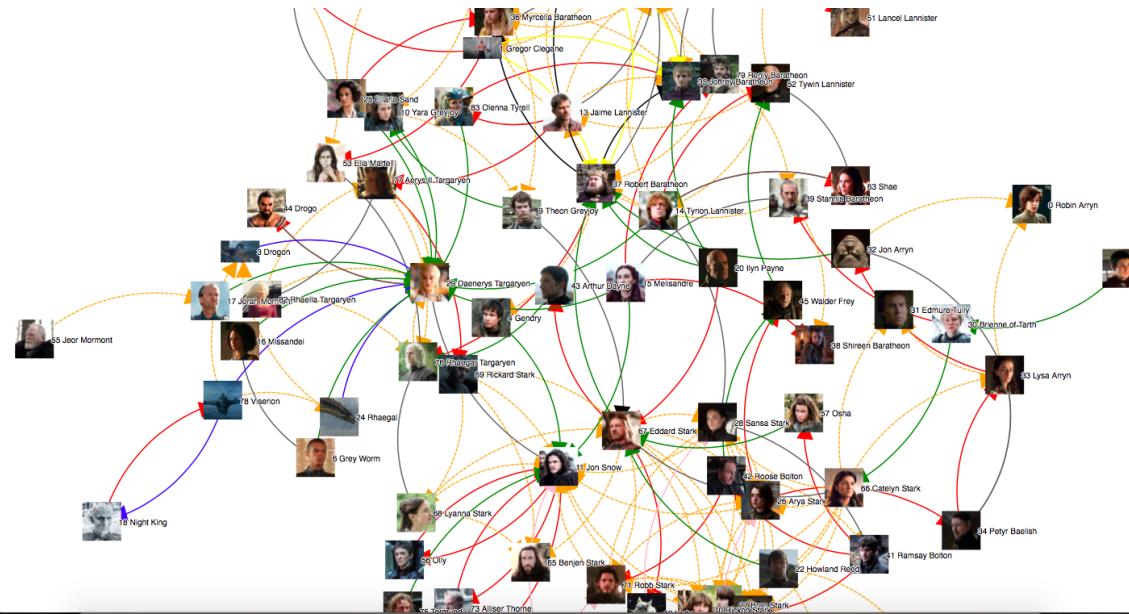


Abbildung 7: Personengraph mit farblich markierten gerichteten Kanten, erstellt mit D3JS.

3.3 Gephi

Kurz vor der Zwischenpräsentation des Projektstands entdeckten wir das Tool „Gephi“. Dieses kann mit sehr großen Datenmengen umgehen und eignet sich unter anderem sehr gut für die Analyse und Visualisierung von sozialen Netzwerken. Außerdem kann es mit Graphen umgehen, welche sowohl gerichtete als auch ungerichtete Kantenbeziehungen haben. Letztendlich war das Design der Plots für unsere Zwecke nicht umfangreich genug anpassbar (Abbildung 8). Jedoch stellte sich die Nutzung dieses Tools als gute Orientierungsmöglichkeit für uns heraus. Durch die

Anwendung von Algorithmen

Möglichkeit die Knoten per Drag & Drop zu manipulieren konnten wir Gephi dazu nutzen mit der Anordnung der Knoten zu experimentieren und verschiedene Layouts auszuprobieren.

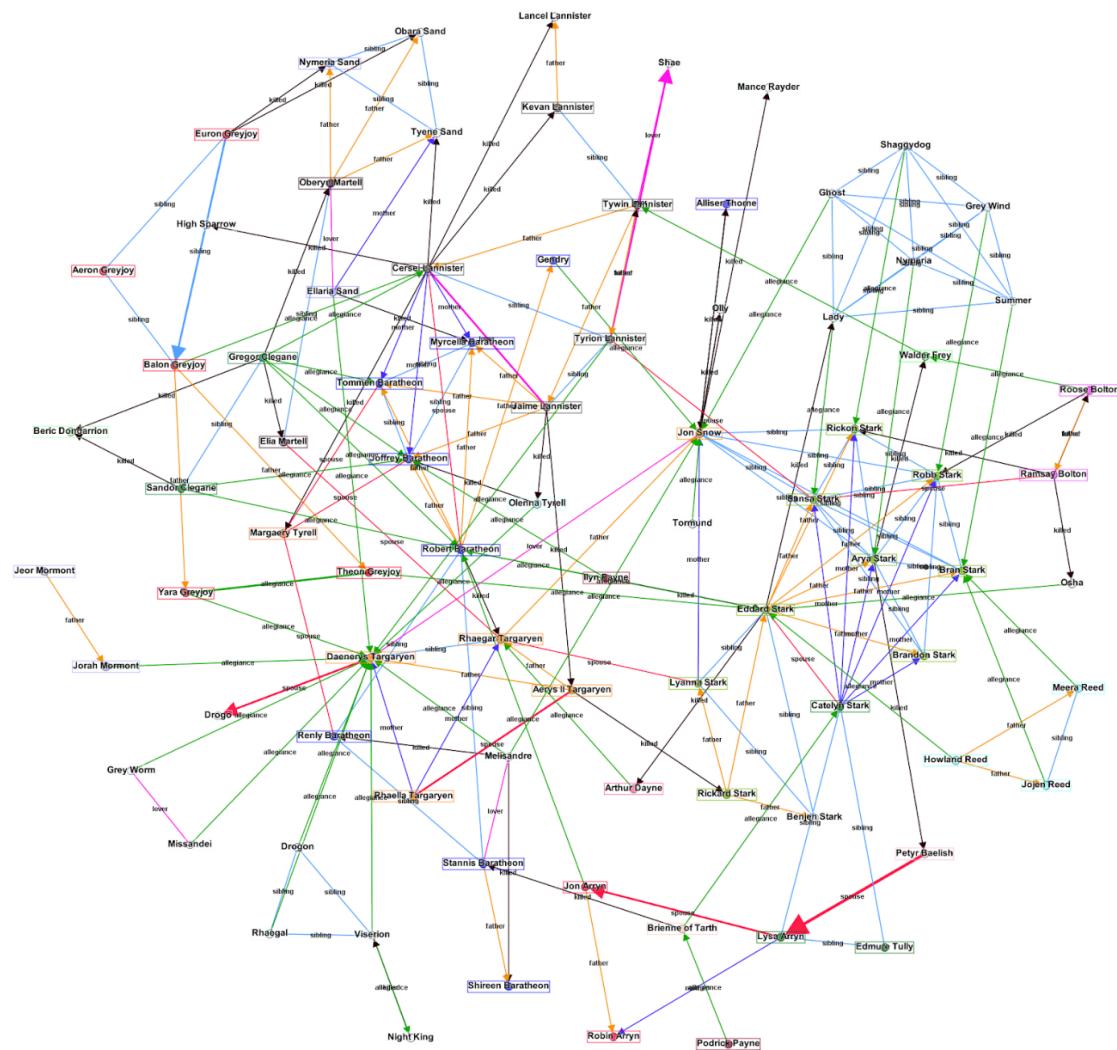


Abbildung 8: Plot eines Teildatensatzes (einige Kanten wurden weggelassen) mit Gephi nach Anwendung des Fruchterman Reingold Algorithmus zur Anordnung der Knoten.

4 Entwicklung der finalen Lösung

Bis Meilenstein 5 wurde an einer Lösung zur Erzeugung von einem Graphen als SVG-Datei mithilfe von Python gearbeitet. Es sollte als Alternative für die eingesetzten Frameworks dienen. Hierbei wurde besonderen Wert darauf gelegt, dass sich der gezeichnete Graph sehr flexibel anpassen lässt. Das erklärt auch die Technologiewahl: Scalable Vector Graphics (SVG) in Verbindung mit Cascading Style Sheets (CSS). Zudem ist es ursprünglich angedacht gewesen, die Knotenpositionen aus anderen Frameworks zu importieren und die Sortierung mit anderen Werkzeugen vorzunehmen. Aus diesem Ansatz ist im weiteren Verlauf des Projekts eine eigenständige Lösung entstanden. Im folgenden wird auf die einzelnen Iterationen und damit auf die Entwicklung dieser Lösung genauer eingegangen.

4.1 Iteration 1

Zunächst wurden die Knoten importiert und zufällig angeordnet. Diese erste Iteration enthält bereits Kanten mit Pfeilspitzen und die Knoten als Kreise. Abbildung 9 enthält den Stand, mit dem er dem gesamten Team erstmals vorgestellt wurde.

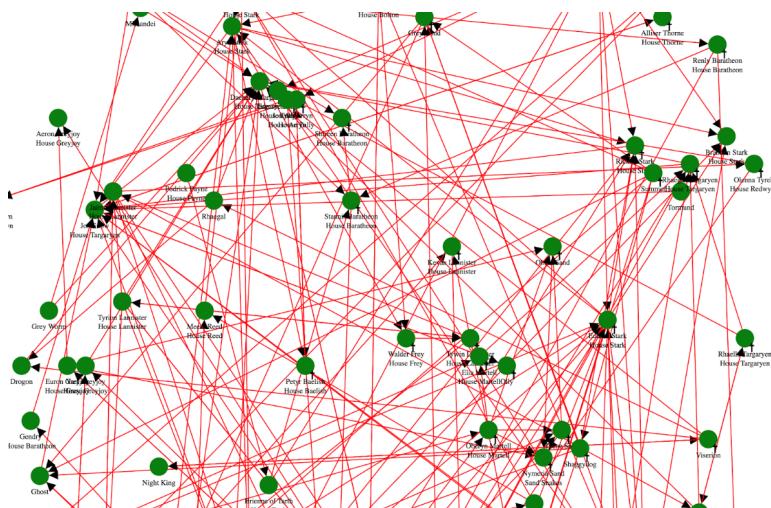


Abbildung 9: Die erste Iteration des SVG-Projekts. Die Knoten sind hier zufällig angeordnet.

4.2 Iteration 2

In der zweiten Iteration wurde der Graph optisch angepasst. Insbesondere wurden die Knoten in dieser Iteration auf einem Kreis angeordnet, die Farben der Grafik angepasst und die Charakterportraits importiert. Zusätzlich wurden verstorbene Charaktere mit einem Symbol gekennzeichnet. Zudem wurde im Team besprochen, welche Strategien angewandt werden könnten, um die Knoten besser anzutragen. Die zuvor durchgeführte Datenanalyse (siehe Abschnitt 2) erwies sich dafür als sehr nützlich, da große Gruppen und die Arten der Beziehungen schnell erkannt werden konnten.

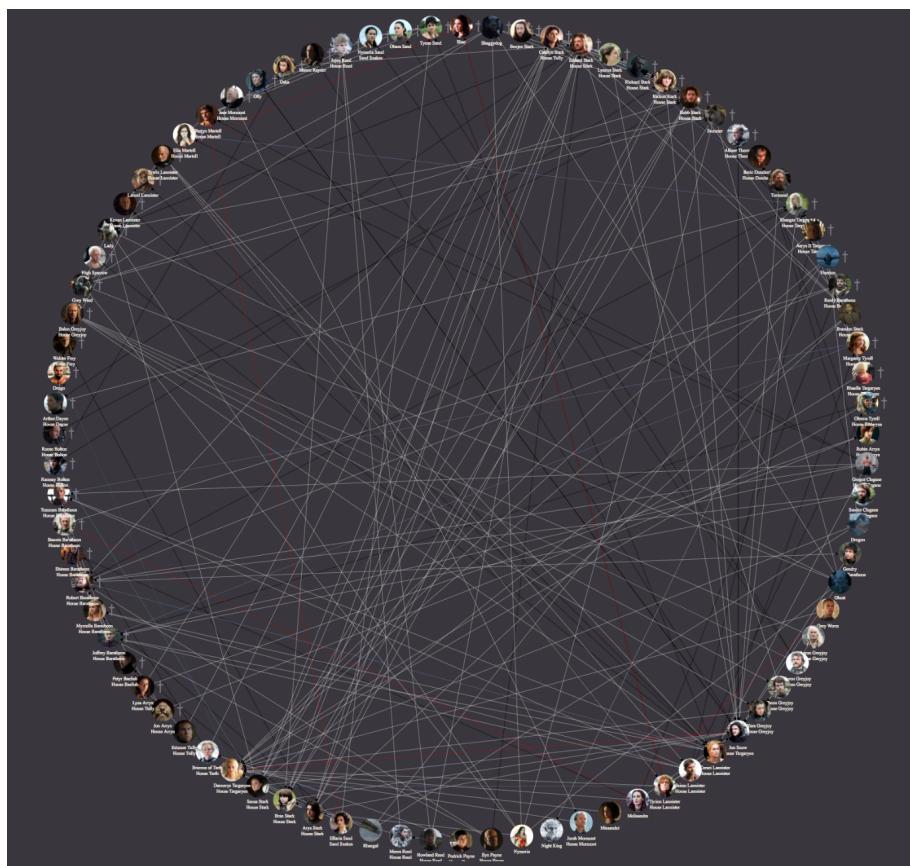


Abbildung 10: Zweite Iteration des SVG-Projekts. Anordnung der Knoten auf einem Kreis. Das Design des Graphen wurde verbessert.

4.3 Iteration 3

Als Ergebnis der Teambesprechung über Verbesserungsmöglichkeiten wurden die Knoten in der nächsten Iteration auf zwei Kreise aufgeteilt und diese nach Haus-Zugehörigkeit sortiert. Das Ergebnis ist in Abbildung 11 ersichtlich. Der Vorschlag, für die Kanten der Geschwister-Beziehungen eine andere Darstellung zu suchen erwies sich als schwierig. Die Idee dahinter ist, dass ein Großteil der Kanten für diese Beziehungen verwendet wurden.

Das Ergebnis dieser Iteration hatte den Vorteil einer effektiveren Platznutzung, aber den Nachteil, dass einige Kantenverläufe nur noch schwer nachvollzogen werden konnten, da sie durch andere Knoten hindurch verliefen.

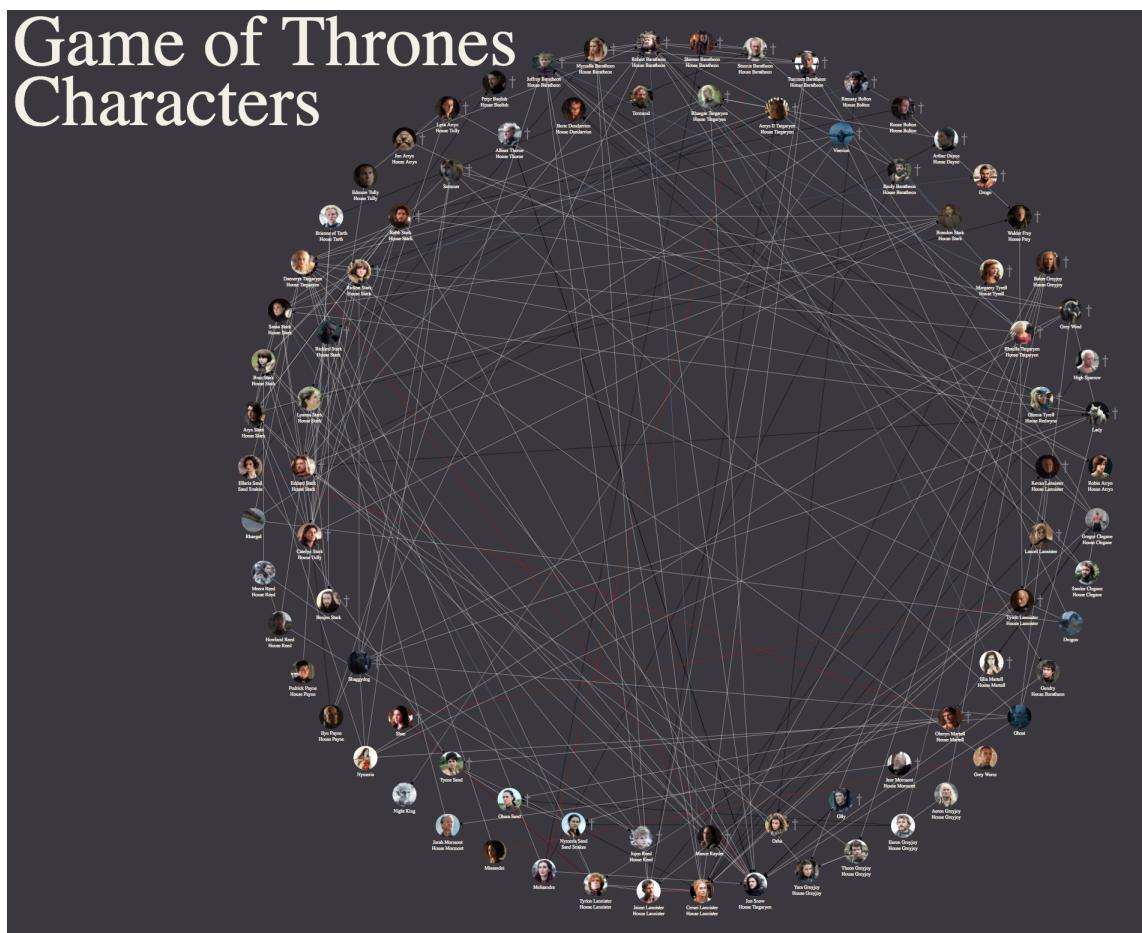


Abbildung 11: Dritte Iteration des SVG-Projekts. Anordnung der Knoten auf zwei Kreisen verschiedener Größe um den Platz besser auszunutzen.

4.4 Iteration 4

In den nachfolgenden Iterationen wurde ein Hauptkreis zur Positionierung der Knoten verwendet. Auf einem zweiten, äußeren Kreis wurden dann Knoten positioniert, die nur eine Kante zu einem anderen Knoten haben. Um Kanten zu vermeiden, die durch andere Knoten auf dem Kreis führen würden dann Bezier-Kurven eingesetzt, die zunächst in Richtung Kreismitte verliefen.

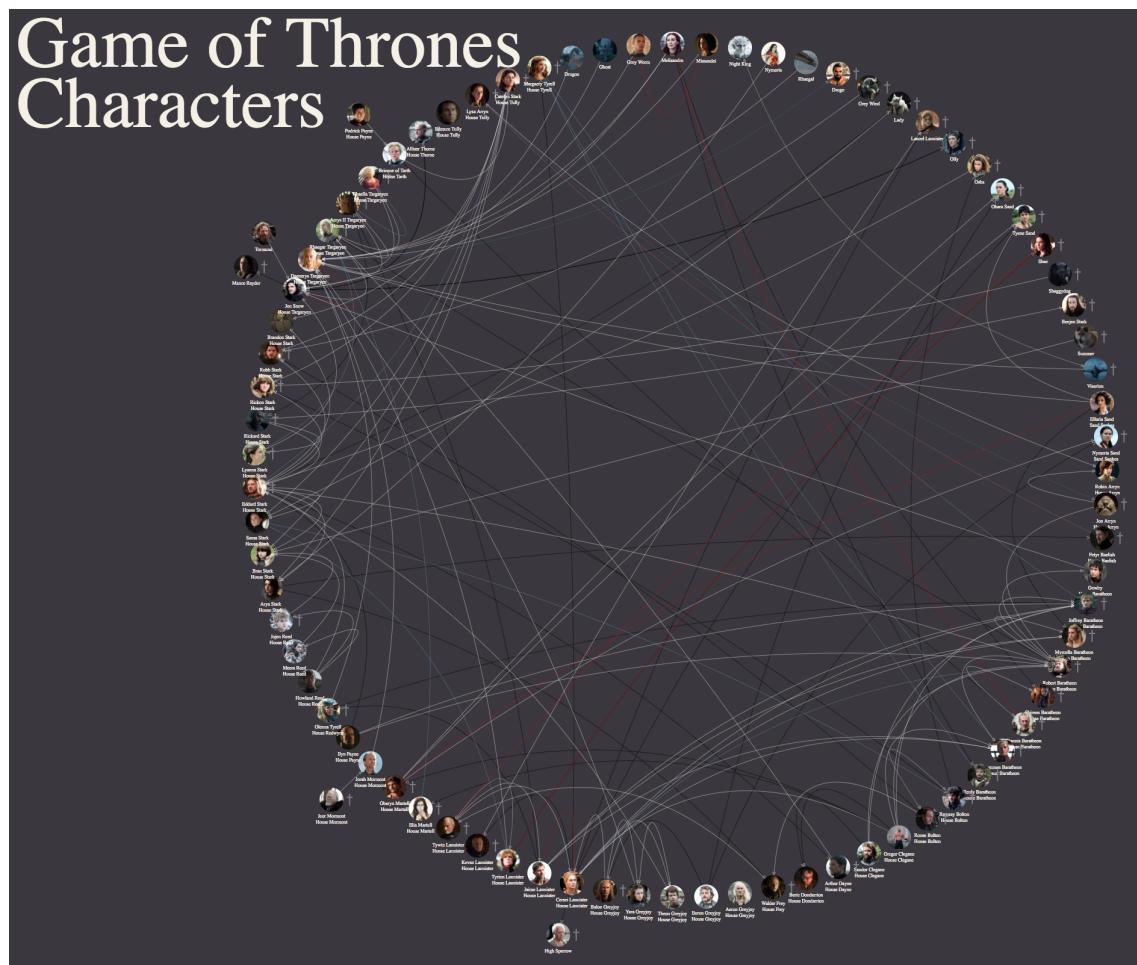


Abbildung 12: Vierte Iteration des SVG-Projekts. Die zuvor geraden Kanten wurden durch Bezier-Kurven ersetzt, um ein Nachverfolgen der Kanten zu ermöglichen.

4.5 Iterationen 5 und 6

In den Iterationen fünf und sechs wurde eine neue Schriftart in dem Stil der Serie verwendet und in die SVG-Datei eingebettet. Andere Änderungen sind nachfolgend zusammengefasst:

- die Graphik wurde vom Querformat in das Hochformat überführt
- die Kanten wurden je nach Art der Beziehung unterschiedlich gefärbt
- die Kanten von oder zu wichtigen Personen wurde vergrößert
- einige weitere Knoten wurden manuell nach außen ausgelagert

4.6 Iterationen 7 und 8

In der letzten Iteration wurde zunächst ein Gradient als Hintergrund eingefügt. Zudem haben wir uns entschieden Kanten, die die Beziehungen von Geschwistern darstellen, nach außen zu richten, wenn sie sich in unmittelbarer Nähe befinden.

In einem letzten Schritt wurde die per Software erzeugte SVG-Grafik manuell überarbeitet. Das Endergebnis ist in Abbildung 13 dargestellt.

5 Zusammenfassung und Ausblick

Das Softwareprojekt „Anwendungen von Algorithmen SS18: Graphenalgorithmen“ startete mit einer Orientierungsphase, in der jeder Beteiligte darauf bedacht war eine geeignete Programmiersprache, wie auch geeignete Bibliotheken und Frameworks zu identifizieren.

In der anfänglichen Projektphase stellten sich Python und JavaScript als Favoriten bei der Technologiewahl heraus. Zur Zwischenpräsentation wurde ein ansprechender Prototyp mit Hilfe von Gephi realisiert. Durch diesen Zwischenschritt konnten sich alle Beteiligten eine Vorstellung des Endergebnisses machen, ohne dass eine lange Einarbeitungszeit in ein Framework notwendig gewesen ist.

Während Lidia Krus unter Einsatz des D3-Frameworks einen weiteren Prototypen entwickelte und sich daraus zwischenzeitlich erneut eine Tendenz zu JavaScript er-

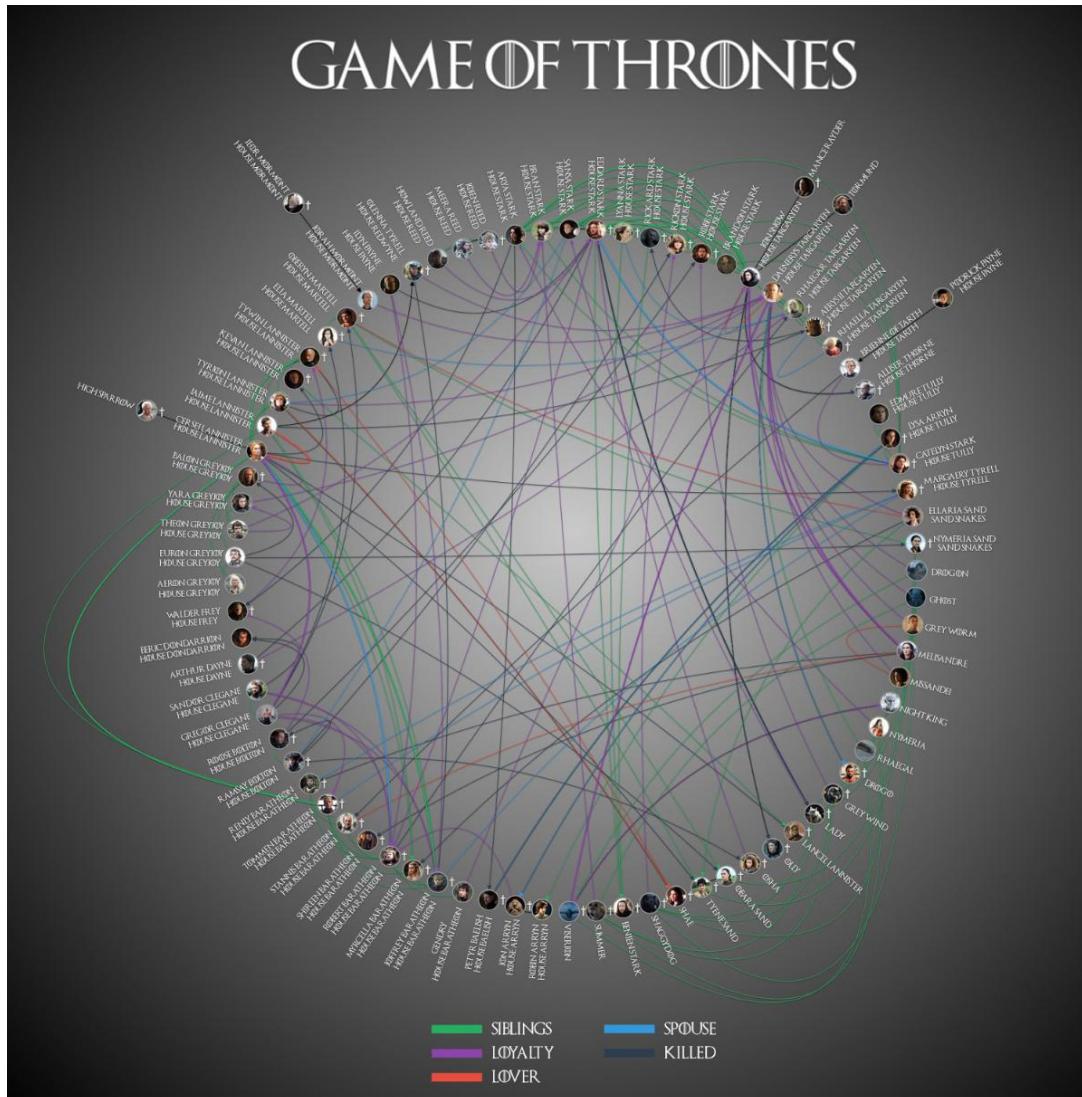


Abbildung 13: Finale Iteration des SVG-Projekts. Die Abbildung ist zugeschnitten.

gab, überraschte Timo Haffner mit seinem Python-SVG-Entwurf und gab damit die finale Richtung vor. Im letzten Drittel der Projektphase wurde dieser bereits umfangreich entwickelte Systementwurf weiter verbessert, indem zunächst offene Probleme und Features festgelegt und als Tickets unter den Projektbeteiligten aufgeteilt wurden. Die Bearbeitung erfolgte sowohl einzeln, als auch in Gruppen um die Kenntnisse der Teilnehmer schnell anzugeleichen und um weitere, wichtige Details zur Veranschaulichung des Datensatzes im Endergebnis zu gewährleisten.

Anwendung von Algorithmen

Entscheidend war in der gesamten Projektphase die Wahl der passenden technischen Hilfsmittel, die wir in [Gitlab](#), [Slack](#), Google Drive und Google Hangouts gefunden haben. Durch diese Tools wurden Kommunikations- und Abstimmungsmöglichkeiten geschaffen. Zudem konnten Dokumente und Quellcode jeweils an einem zentralen Ort abgelegt werden und von mehreren Teilnehmern zeitgleich bearbeitet werden. Verschiedene Zeichentools haben während der Hangouts dazu beigetragen technische Konzepte zwischen Gruppenmitgliedern zu erläutern. Die Verwendung dieser technischen Werkzeuge haben eine effektive und effiziente ortsunabhängige Arbeitsweise ermöglicht und maßgeblich zum Projekterfolg beigetragen.

In besonderem Fokus sei hier die Meilensteinphase M7 erwähnt, in der die Entwicklung der finalen Lösung durch insgesamt acht Iterationen verlief. Anfangs zufällig angeordnete Knoten wurden basierend auf den Erkenntnissen der erzeugten Datenanalyse schrittweise in ihrer Anordnung optimiert. Dabei wurden mehrfach Abwägungen zwischen Knotenanordnungen aufgrund ihrer charakteristischen Kantenbeziehungen getroffen, wie auch einem möglichst optisch ansprechenden und übersichtlichen Designanspruch. Im Endergebnis liegt ein Graph im Hochformat vor dessen Kantenbeziehungen abhängig vom Beziehungsstatus eingefärbt und durch einen im Hintergrund eingefügten Gradienten in seiner Ablesbarkeit optimiert wurden. Die Wahl und Positionierung der Namensschriftzüge neben den Charakterportraits ist gut lesbar und im Stil der Serie designt worden. Charaktere mit einer geringen Vernetzung im Graphen wurden der Übersichtlichkeit halber auf einen weiteren Außenring der Kreisformation positioniert.

Eine erfolgreiche Teilnahme am geplanten Graphen-Wettbewerb in Barcelona war nicht unsere primäre Motivation. Vielmehr wollten wir Erfahrungen in einem agil gestalteten Softwareprojekt sammeln. Trotzdem ist im Laufe dieses Softwareprojekts ein vorzeigbares Endprodukt entstanden.

Literatur

- [1] D3.js - Data-Driven Documents. <https://d3js.org/>, 2018. Accessed: 2018-09-04.
- [2] got-graph.graphml. <http://graphdrawing.de/contest2018/got-graph.graphml>, 2018. Accessed: 2018-09-04.
- [3] Almende B.V. vis.js - A dynamic, browser based visualization library. <http://visjs.org/>, 2018. Accessed: 2018-09-04.
- [4] Hadrien Mary. Welcome to PyGraphML Documentation 8212; PyGraphML 2.2 documentation. <http://hadim.fr/pygraphml/index.html>, 2018. Accessed: 2018-09-04.
- [5] HBO. Game of Thrones Cast Crew | HBO Official Site. <https://www.hbo.com/game-of-thrones/cast-and-crew>, 2018. Accessed: 2018-09-04.
- [6] NetworkX developers. NetworkX. <http://networkx.github.io/>, 2018. Accessed: 2018-09-04.
- [7] Plotly. Modern Visualization for the Data Era - Plotly. <https://plot.ly/>, 2018. Accessed: 2018-09-04.
- [8] The Matplotlib development team. Matplotlib: Python plotting mdash; Matplotlib 2.2.3 documentation. <https://matplotlib.org/>, 2018. Accessed: 2018-09-04.