# 1. Motivation

Dialoge sind in fast jeder Art Spiel zu finden, das in irgendeiner Weise einer Handlung folgt. Ob ich hierbei an die kleinen Monologe von NPCs in Inscryption (2021, Daniel Mullins Games) denke oder die komplexen Dialogzweige aus Disco Elysium (2019, ZA/UM), – beim Spielen von Videospielen begegnen mir Dialoge täglich, denn es ist generell eher schwer, eine Geschichte ohne Dialog zu erzählen. Als begeisterte Rollenspielerin, die Rollenspiele im digitalen und im analogen Raum spielt, habe ich allerdings das Gefühl, dass sich Dialoge als System in der digitalen Spielwelt in den letzten zwanzig Jahren sehr wenig weiterentwickelt haben. Während ich die großen Rollenspiele der letzten Jahre gespielt habe, musste ich bemerken, dass sich Unterhaltungen dort nur wenig von Unterhaltungen in Spielen wie Fallout (1997, Interplay Entertainment) oder The Elder Scrolls III: Morrowind (2002, Bethesda Softworks) unterschieden. So bieten beispielsweise Genshin Impact (2020, miHoYo) und Cyberpunk 2077 (2020, CD Projekt RED), zwei größere Rollenspiele aus den letzten Jahren, noch immer nur die Möglichkeit, dem Dialog zu folgen, ohne selbst viel Input zu geben. Dieser Zustand wird ab und an unterbrochen, damit Spielende einen eigenen Gedanken einwerfen können, der in Form von zwei bis vier Antwort-Auswahlmöglichkeiten dargestellt wird, die die Spielentwickler den Spielenden zur Auswahl stellen. Kommunikation passiert in Videosequenzen oder langen Dialogen, in denen Spielende sehr wenig Kontrolle über ihre Spielfigur haben und so ist das Rollenspielerlebnis, wenn es um Eigeninitiative in Dialogen geht, noch immer ein eher passives.  
Die Entscheidung, Spielende meist nur aus einigen vorgegebenen Antwortmöglichkeiten wählen zu lassen, hat natürlich viele Vorteile: so zeigt das Spiel deutlich, welche Optionen den Spielenden zur Verfügung stehen, und macht es den Entwickelnden einfacher, für jede mögliche Entscheidung der Spielenden tatsächliche Inhalte zu kreieren. Auch bedarf das System nur wenig Erklärung und selbst Spielende, die noch nie ein Videospiel gespielt haben, können einfach verstehen, wie es funktioniert.  
Trotzdem finde ich, dass sich dieses System sehr limitierend anfühlen kann. Oft, wenn ich ein Spiel mit solch einem System gespielt habe, hatte ich das Gefühl, dass die Antwortmöglichkeiten mich in eine Bahn drängen würden, und mir vorgeben würden, was für eine Art von Charakter ich spielen könnte. Generell haben diese Systeme für mich sehr wenig von der Entscheidungsfreiheit und Bandbreite an Möglichkeiten, den eigenen Charakter zu definieren, den Spielende in analogen Rollenspielen, wie Dungeons and Dragons (1974, Wizards Of The Coast) zur Verfügung stehen, in denen Spielende jede Art von Charakter spielen können, solange sie die Erlaubnis von der spielleitenden Person bekommen. Ein weiteres Problem, welches ich mit dem aktuellen Dialogsystem großer Spiele sehe, ist, dass Spielende nur in den nötigsten Momenten nach einer Eingabe gefragt werden. So geben Spielende eher wenig über sich selbst preis und sie können die Charakterzüge, die Backstory und die Beziehungen ihrer Spielfigur nicht wirklich ausleben und ausgestalten. Ein Developer des Spiels Fallout: New Vegas (2010, Obsidian Entertainment), Josh Sawyer, beschreibt in seinem GDC Talk „Choice Architecture, Player Expression, and Narrative Design in Fallout: New Vegas“ aus dem Jahr 2012 das momentane Input-System als eine Art „Interview-Simulator“. Spielende können NPCs zu jeder Art Thema ausfragen und müssen dabei lange Dialogzweige „abarbeiten“, um jede Information, die das Spiel einem geben möchte, zu erhalten. Dabei fehlt meiner Meinung nach auch die Kommunikation, Spielenden zu zeigen, dass sie nicht jede Antwortmöglichkeit in einem Dialog wirklich ansprechen müssen, da sich Spielende oft gezwungen sehen, alle zusätzlichen Informationen, die ein NPC hat, zu erfragen.  
Wenn Spielenden jede Antwort von vornherein präsentiert wird, fehlt ebenso die Möglichkeit, eine kreative oder clevere Antwort auf ein Problem zu finden, da die einzige Beteiligung, die die Spielenden in einem Gespräch haben, ist, eine von mehrere gegebenen Optionen zu wählen.  
Während einige Spiele versuchen, Dialoge in unkonventionellen Wegen anzugehen, bieten diese Systeme meist nicht das Potenzial, außerhalb des Kontexts ihres eigenen Spiels zu funktionieren oder sind für große Rollenspiele zu komplex.  
Als begeisterte Rollenspielerin, die sich gleichermaßen für den analogen und digitalen Raum der Rollenspielwelt interessiert, wollte ich herausfinden, ob und wie Dialoge auch anders dargestellt werden können, ohne, dass das System zu komplex wird und zu viel Erklärung benötigt. Ebenso wollte ich sehen, ob das System, welches sich aus meiner Arbeit ergeben würde, möglicherweise auch in einem Rollenspiel genutzt werden könnte, dass dafür nicht spezifisch den Kontext einer Welt bieten müsste. Diese Arbeit führt die Lesenden durch das finale Werkstück, meine Gedanken und zusätzliche Ideen für die Erweiterung meines Systems und endet schlussendlich damit, dass ich mein finales Werkstück und meine Arbeitsweise evaluiere.

# 2. Vorbereitung

Vor dem Entwickeln meines Systems war es mir wichtig zu wissen, in welchen Arten und Weisen Antwortmöglichkeiten normalerweise in Rollenspielen genutzt werden und welchen Zweck sie für Entwickelnde haben. Dafür habe ich mir die Dialogsysteme aus verschiedenen Rollenspielen angesehen, die ich gespielt habe und diese auf die Verwendungen von ihren Dialogsystemen analysiert. Für meine Analyse betrachtete ich die Spiele *80 Days* (2014, Inkle / Cape Guy Ltd), *Coffee Talk* (2020, Toge Productions), *Cyberpunk* (2020, CD Projekt RED), *Disco Elysium* (2019, ZA/UM), *Genshin Impact* (2020, miHoYo), *Greedfall* (2019, Spiders), *Firewatch* (2016, Campo Santo), *Pyre* (2017, Supergigant Games), *The Council* (2018, Big Bad Wolf Studio), *The Long Dark* (2014, Hinterland Studios), *The Outer Worlds* (2019, Obsidian Entertainment), *The Witcher 3* (2015, CD Projekt RED), *Outer Wilds* (2019, Mobius Digital), *Undertale* (2015, Toby Fox), ebsenso wie die Gesamtheit der Spiele der Spielereihen *Assassin’s Creed* (Ubisoft), *Dragon Age* (Bioware), *Fallout* (Bethesda Softworks / Obsidian Entertainment / Interplay Entertainment), *Harvest Moon* (Rising Star Games / Marvelous Entertainment), *Life Is Strange* (Dontnod Entertainment / Deck Nine Games), *Pathologic* (Ice-Pick Lodge) und *The* *Elder Scrolls* (Bethesda Softworks). Meine Erkenntnisse wollte ich bei der Entwicklung meines Systems beachten und sichergehen, dass mein System diese Funktionalitäten zumindest im Teil erfüllt.

Bei meiner Betrachtung von diesen Rollenspielen, bin ich zuallererst dem Schluss gekommen, dass Antwortmöglichkeiten meist dann genutzt werden, wenn Spielende eine Entscheidung zu treffen haben, die den Spielverlauf in irgendeiner Art und Weise beeinflusst – das können große, weitreichende Entscheidungen sein, aber auch kleinere Entscheidungen, wie, ob eine Situation diplomatisch gelöst werden soll oder welcher Weg genommen werden soll. Vor allem die *Life Is Strange* Spiele demonstrieren, dass eine Entscheidung wichtig ist, in dem sie die Szene anhalten, und den Spielenden Zeit geben, eine von mehreren Entscheidungen auszuwählen, die ihnen groß auf dem Bildschirm angezeigt werden.

Auch werden Dialoge oft genutzt, um Spielenden die Chance zu geben, vorformulierte Nachfragen anzustellen, die ihnen erlauben, optionales, tieferes Wissen zu einem momentanen Thema zu erhalten. Vor allem die Rollenspiele von CD Projekt RED geben Spielenden oft die Möglichkeit, optionale Nachfragen auswählen zu können, ehe sie im Dialog weitergehen müssen. Diese können zusätzliche Informationen zu einer Mission oder Situation geben. Auch in anderen Spielen, wie der *Fallout* und *Dragon Age* Reihe tauchen Nachfragen als Dialogoption auf, mit der Besonderheit, dass hier zusätzlich Begleiter-NPCs jederzeit angesprochen werden können, um ihnen viele zusätzliche Fragen zu ihrer Vergangenheit, ihren Gedanken und Ansichten zu stellen. Einen weiteren Ansatz, um zusätzliche Informationen zu erhalten, bietet das Spiel *Pyre*, das es Spielenden erlaubt, mit der Computermaus über markierte Begriffe in Dialogen zu halten, die sie nicht verstehen, um in einer kleinen Info-Box eine kurze Erklärung zum Thema zu erhalten.

Die meisten anderen Nutzungen von Dialogoptionen finden sich nicht in jedem Spiel wieder oder werden nur in sehr spezifischen Situationen genutzt. Beispielsweise gibt es die Möglichkeit, in Dialogen spezifisches Wissen und spezifische Skills abzufragen, die ein Charakter besitzt oder nicht besitzt, um so verschiedene Lösungen für ein Problem finden zu können. Dieser Ansatz findet sich beispielsweise in einigen *Fallout* Spielen, *Disco Elysium* oder in *The Outer Wilds*.

Auch bieten einige Spiele die Möglichkeit, Diskussionen zu führen. Hierbei unterscheidet sich stark, wie Spiele eine Diskussion interaktiv gestalten – das kann sein, die richtige Antwort aus mehreren Optionen zu wählen, um jemanden zu überzeugen oder aber, wie in *Life Is Strange: Before The* *Storm* (2017, Deck Nine Games), eine Diskussion in einem schnellen Minispiel zu lösen. Auch ist es öfters der Fall, dass Spielende, um eine Diskussion zu gewinnen, über spezifisches Wissen verfügen müssen. Das kann sein, eine Fähigkeitsprobe zu schaffen, aber auch, etwas herauszufinden, was den Spielenden im Gespräch weiterhelfen kann. Beispielsweise kann der Antagonist des Spiels *Fallout* (1997, Interplay Interactive) davon überzeugt werden, dass sein Plan nicht funktionieren kann, wenn Spielende genug Informationen in den alten Laboren der Welt gesammelt haben, um zu diesem Schluss zu kommen. Auch einen Preis zu verhandeln ist eine immer wieder auftretende Mechanik, die meist sehr unterschiedlich dargestellt wird, aber meist Spielende in irgendeiner Weise auffordert, einen Preis zu nennen, bis der Gegenüber einwilligt. In *Genshin Impact* wird dies beispielsweise durch einen Slider gelöst, in dem ein Wert gewählt werden kann, während in anderen Spielen schlicht bei einer Bezahlung nach mehr Geld gefragt werden kann oder sogar für etwas Geld verlangt werden kann, wofür Spielende normalerweise kein Geld bekommen würden.

## 2.1. Dialogsituationen

Trotzdem sollte mein System sich nicht nur von bereits existierende Dialogsystemen in Rollenspielen definieren lassen, sondern ich wollte auch neue Möglichkeiten finden, Dialoge zu führen. Deshalb betrachtete ich nicht nur bereits existierende Rollenspiele, sondern versuchte auch, mir Gedanken dazu zu machen, in welchen Situationen Menschen Sprache nutzen, was sie damit bezwecken wollen und wie verschiedene Arten der Konversation aufgebaut sind.

Zunächst legte ich eine Liste an, die verschiedenen Arten von Gesprächen auflistete, die aus einem eigenem Brainstorming und dem Feedback von Kommilitonen entstand. Die Liste reichte von *eine Rede halten,* zu *eine Geschichte erzählen*, zu *über seinen Tag reden*, bis hin zu *flirten*. Ich versuchte, die Begriffe so wenig, wie möglich zu werten oder weiter zu gruppieren. Nachdem ich für mich selbst genug Konversationstypen gefunden hatte, beschloss ich, meine Liste in Hinsicht auf verschiedene Kriterien zu analysieren: hierbei betrachtete ich, wie oft eine Dialogsituation tatsächlich im alltäglichen Leben auftrat, ob es sich bei der Situation eher um einen Monolog oder Dialog handelte, wie lang sie in der Regel war und welche Dialogtypen sich ähnelten. Diese Analyse half mir dabei, herauszufinden, wie häufig eine Dialogsituation in einem Spiel ungefähr gebraucht werden könnte, welche Situationen gleiche oder ähnliche Mechaniken nutzen könnten und wie viel Informationseingabe bei jedem Dialogtyp vonseiten der Spielenden gebraucht werden würde. Diese Erkenntnisse verglich ich dann miteinander, um eine Hierarchie zu erstellen, die mir zeigte, welche „Gesprächsarten“ die praktischsten und sinnvollsten waren, um sie in ein Dialogsystem einzubauen.

## 2.2. Wahl des Systems

Nachdem ich erarbeitet hatte, welche Dialogsituationen ich möglicherweise umsetzen könnte und analysiert hatte, welche Relevanz jede davon in einem Dialogsystem hätte, begann ich mit dem Prozess, mir zu überlegen, wie diese systemisch umgesetzt werden könnten. Zunächst versuchte ich, die erarbeiteten Situationen einzeln zu betrachten und mir vorzustellen, welche Informationen ich gerne an das Spiel weitergeben wollen würde, wenn ich beispielsweise eine Diskussion oder einen Flirt in einem Spiel spielen würde. Ich versuchte dabei auch Fokus darauf zu legen, wie ich diese Eingaben steuern könnte, welche Systeme zu komplex werden könnten und welche User Interfaces ich benötigen würde, um dieses System darzustellen.

In diesem Schritt kamen mir viele Konzepte in den Sinn, von Dialogen, die über ein Natural-Language-Processing-Programm liefen, zu Dialogsituationen, die als verschiedene Minispiele dargestellt werden. Schnell kam mir die Idee, Worte aus dem Dialog nehmen zu können und diese zu benutzen, um Nachfragen anzustellen oder Antworten zu geben. Als ich im Folgeschritt analysierte, welche Systeme viele Dialogsituationen decken würden und gleichzeitig intuitiv und simpel zu bedienen wären, blieb mir dieses System als vielversprechendstes System im Kopf. Auch, weil ich es persönlich für den spannendsten Ansatz hielt, wollte ich das System unbedingt tiefer erkunden. Da bereits erste konzeptionelle Ansätze sehr vielversprechend wirkten, beschloss ich schnell, die anderen gefundenen Ansätze nicht ebenfalls prototypisch umzusetzen. Diese Entscheidung traf ich auch aus Zeitgründen, da das Aufbauen einer Dialogstruktur mich viel Zeit kosten würde und jedes der geplanten Systeme fundamental anders funktionierte.

## 2.3. Arbeitsprozess

Generell probierte ich, zunächst über alle Möglichkeiten des Systems nachzudenken und legte für mich verschiedene Features fest, die ich gerne in der Arbeit sehen wollen würde. Für jedes davon definierte ich, was es können sollte, und versuchte mögliche Probleme zu finden, ehe ich mir eine Liste an Aufgaben zusammenstelle, die die Programmierarbeit in kleine Teile teilen sollte.

Auf der Plattform Miro führte ich eine Liste, auf der ich jede anstehende Aufgabe aufschrieb, die ich mir bereits überlegt habe, damit ich keine Aufgaben vergaß, die ich mir vorgenommen habe. Sobald ich einen Bug oder ein Problem fand, schrieb ich sofort eine Notiz dafür, um immer im Blick zu haben, was noch zu tun war. Diese aufgeteilte Arbeit machte es einfacher, Systeme kürzen oder streichen zu können, als ich bemerkte, dass die Zeit nicht reichen würde, um alle meiner Vorhaben umzusetzen. Zudem half Miro mir dabei, meinen Fortschritt schnell fotografieren und dokumentieren zu können, und gab mir einen zentralen Punkt, an dem ich alle Daten in Bezug auf mein Projekt speichern konnte. Ich nutzte Miro auch, um schnell Gedanken festzuhalten oder ein System genauer unter die Lupe zu nehmen.

Ich versuchte so oft, wie ich konnte, Feedback einzuholen und mein System testen zu lassen, um zu sehen, ob die Steuerung intuitiv war, und wo noch Feedback und Kommunikation fehlte. Nach Weihnachten machte ich mir einen Plan, in dem ich genau aufschrieb, was noch zu tun war, und definierte eine genaue Liste an Aufgaben, die ich bis zur Abgabe abarbeitete. Ab Januar nahm ich mir meist einen Tag in der Woche vor, an dem ich am Textteil arbeiten würde, um mit der Arbeit nicht zu spät anzufangen.

Da mein Fokus auf System Design und Programmierung lag, waren Sound-Design, die Gestaltung der Szene und Narrative für mich eher zweitrangig. Daher arbeitete ich hier vor allem mit gratis 3D Assets, Animationen, Rigs und Sounds, die ich online fand und in meiner Szene zusammenbaute. Die Szene sollte keinen Fokus auf die Handlung in den Dialogen legen, sondern zeigen, wie mein erarbeitetes System in verschiedenen Dialogsituationen genutzt werden kann, deshalb versuchte ich, eine möglichst unaufwändige Geschichte zu finden, die aus modularen Teilen besteht, anstatt aus einem langen narrativen Strang.

# 3. Werkstück

## 3.1. Die Szene

Für das Werkstück habe ich mich für eine Szene entschieden, in der die Spielenden auf einer Hochzeitsfeier umherlaufen und die Gäste dort in beliebiger Reihenfolge ansprechen können. In der Szene befinden sich acht ansprechbare Charaktere, sowie ein Tutorialcharakter, der den Spielenden zunächst erklärt, wie das Spiel gespielt wird. Nach der Begegnung mit dem Tutorialcharakter ist die Szene frei begehbar. Dabei sollen kleine Aufgabenstellungen dabei helfen, die Spielenden zu motivieren, das System auf verschiedene Arten und Weisen auszuprobieren.

## 3.2. Die Handlung

Die Szene, die ich in meiner Arbeit präsentiere, spiegelt einen Abend auf einer Hochzeitsfeier in einem Wald in einem fiktiven Land namens *Nocturne* wider. Hier haben sich Gäste zusammengefunden, um nach der Hochzeitszeremonie zu feiern, zu essen und zu tanzen. Die Szene zeigt den frühen Abend der Feier, zu dem erst wenige Gäste den Schauplatz erreicht haben, was unter anderem an der komplizierten Lage des Schauplatzes liegt. So werden Charaktere, wie beispielsweise die Eltern des Bräutigams und Jordan und Katie, zwei gute Freunde des Brautpaars, erwähnt, aber befinden sich nicht zusätzlich in der Szene.

Die Spielenden treffen insgesamt auf acht Charaktere, die sich mit ihnen unterhalten und ihnen kleine Aufträge und unterschiedliche Dialogoptionen bieten. In der Szene befinden sich die folgenden Charaktere:

Riley, welcher demonstriert, wie Spielende mit einem Begleiter-NPC in einem größeren Rollenspiel interagieren könnten, der ihnen konstant durch die Welt folgt. Riley ist ein verunsicherter junger Mann, der auf der Suche nach der Person ist, die sich mit ihm über eine Dating-App verabredet hat und ihn zur Hochzeit eingeladen hat.

Heleen, die Tante der Braut, die auf der Suche nach ihrem Sohn ist, der auf der Feier verschwunden ist. Ihre Aufgabe soll Spielende ermutigen, den Charakteren in der Szene Fragen zu stellen, in dem sie umhergehen und die Gäste der Feier nach Heleen’s Sohn, Matheo, befragen.

Mickey, Heleen’s Ehemann, der die verschiedenen Gerichte, die auf dem Buffet der Feier präsentiert werden, probiert und auf Nachfrage detaillierte Beschreibungen und genauere Erklärungen zu ihnen geben kann. Der Dialog mit Mickey soll Spielenden zeigen, dass sie zu Themen, die sie interessieren, weitere Nachfragen stellen können und soll sie animieren, so viel über das Buffet zu erfahren, wie sie können.

Stubi, ein Charakter, der darauf aus zu sein scheint, die Geschenke zu stehlen und die Spielenden dazu anleitet, herauszufinden, was es für Geschenke gibt. Seine Aufgabe soll, ähnlich, wie Heleen’s Aufgabe, dafür sorgen, dass Spielende animiert werden, Fragen zu stellen.

Der DJ, der scheinbar nach einem der Gäste zu suchen scheint. Wenn die Spielenden ihn anlügen, und behaupten, die Person zu sein, nach der der DJ sucht, können sie herausfinden, was den DJ so nervös macht. Diese Aufgabe soll dabei helfen, Spielenden zu zeigen, dass sie mit ihren Antworten nicht immer die Wahrheit sagen müssen und soll den Spielenden ein weiteres Ziel zum Erfragen geben.

Die Braut, Renna, die schnell einen Ersatzredner für den Abend braucht – die Spielenden können dann in einem Dialog mit ihr eine Rede vor ihr halten. Ihre Gespräche sollen zeigen, dass das System auch für längere Monologe, wie dem Halten von Reden, benutzt werden könnte.

James, der Bräutigam, der den Spielenden den Aufbau des Schauplatzes genauer erklärt und ansonsten auch dafür da ist, um Fragen zu beantworten.

Matheo, Heleen’s Sohn, der sich in den Geschenken versteckt. Matheo hat einiges auf der Feier mitbekommen und kann in einigen Aufgaben, die die Spielenden lösen müssen, helfen.

## 3.3. Design

Für das Design des Werkstücks lag mein Fokus vor allem auf dem Design der UI, da sie der Teil des Spiels ist, mit dem die Spielenden am meisten interagieren würden. Trotzdem sollte die Welt, in der die finale Szene spielt, trotzdem nicht zu prototypisch wirken.

Es war mir wichtig, dass das UI-Design nicht zu unübersichtlich wirken sollte und ich wollte etwas, was das Sammeln von Worten intuitiv repräsentiert und kommuniziert. Ich entschied mich für eine Art Journal-Design, was klar mit dem Notieren von Worten assoziiert wurde. Inspiration für die Farbpalette kam hier von den Farbschemen von Klebezetteln und buntem Papier, die ich wegen ihrer klar unterscheidbaren Farben wählte, die intuitiv mit Kategorien und Organisation verbunden werden.

Die Wortkarten sollten bewegbar aussehen, weshalb ich mich für kleinere, lange Zettel entschied. Um zu kommunizieren, dass längere Worte immer noch zusammengehören, sind diese Zettel mit Klebeband verbunden. Ich versuchte einen Stil zu wählen, den ich schnell produzieren konnte und der einfach zu lesen war, weshalb die meisten UI-Elemente nur aus zwei oder drei Farbtönen unterschieden. Die starken schwarzen Umrisse wurden später hinzugefügt, da es schwer war, viele Teile voneinander zu unterscheiden und sich auch vom Hintergrund abzugrenzen.

Um die Farbschemen schnell anpassen zu können, erstelle ich eine Vorlage, in welcher ich Farbschemen speichern konnte und schnell mit einer anderen Vorlage ersetzen konnte. Ich bemerkte, dass die Farben generell sehr entsättigt sein mussten, damit die vielen verschiedenen Farbtöne ein stimmiges Farbbild ergaben.

In der Szene selbst, probierte ich mit möglichst wenig Assets auszukommen. Da ich für meine Arbeit generell 3D Modelle aus dem Internet benutzen wollte, musste ich mit dem Aufbau der Szene etwas kreativ werden. In vielerlei Hinsicht hat das Angebot an verfügbaren 3D Modellen meine Szene und Handlung stark beeinflusst – so wäre eine Szene in der Stadt oder auf dem Land sehr schwer aufzubauen gewesen und ich entschied mich stattdessen für einen Wald. Auch musste ich beispielsweise mit Hüten arbeiten, um die NPCs unterscheidbar zu machen, da ich nur zwei geriggte Charaktere online gefunden habe und die Dialoge über das Buffet sind stark davon beeinflusst worden, dass die einzigen 3D modellierten Assets für das Buffet, die ich gefunden habe, modellierte Versionen von chinesischen Dim Sum Gerichten waren.

# 4. Das System

## 4.1. Was kann das System?

Das in meinem Werkstück vorliegende finale System besteht aus verschiedenen UI-Elementen, mit denen jeweils eine Interaktion möglich ist. Es ist möglich, sich umherzubewegen und die verschiedenen NPCs, die in der Szene zu finden sind, anzusprechen. Hier kann ein kurzes Gespräch begonnen werden, bei welchem der Text ausgeschrieben in einem Dialogfeld angezeigt wird. Einzelne Worte aus Dialogen oder auch aus anderen Stellen in der Szene können eingesammelt werden und in einer Wortsammlung gespeichert werden. Aus dieser Sammlung können sie ebenfalls wieder herausgelöscht werden. In der Wortsammlung bestehen Möglichkeiten, gesammelte Worte zu Favoriten hinzuzufügen oder nach einer Wortkategorie zu suchen. Gesammelte Worte können genutzt werden, um auf Fragen von NPCs zu antworten oder selbst eine Frage zu stellen. Um den Spielenden ein Ziel zu geben, beinhaltet mein Werkstück einige kleinere Aufgaben, die erledigt werden können und dabei helfen, das System zu präsentieren.

## 4.2. Mechanics

Im Folgenden werde ich genauer erläutern, welche Mechaniken in meinem System existieren und wie sie zusammenwirken. Hierbei gehe ich darauf ein, wie ein Wort ausgewählt wird, wie es gespeichert werden kann und wie die Worte in Antworten und Nachfragen genutzt werden kann.

### 4.2.1. Worte auswählen

In Dialogen, und auch an anderen Stellen im Spiel, können Spielende immer wieder farblich markierten Worten begegnen. Halten Spielende den Maus-Cursor über ein solches Wort, wird das Wort farblich hinterlegt und kann angeklickt werden. Wird ein Wort ausgewählt, kann es mit der Maus bewegt werden und an verschiedenen Stellen im User Interface platziert werden. Worte können in der Wortsammlung gespeichert werden oder als Antworten und Nachfragen verwendet werden.

Worte werden verschieden farblich markiert, abhängig davon, ob ein Wort eingesammelt werden kann oder nicht, was das Sammeln von Worten vereinfachen soll. Es ist nicht möglich, ein Wort, welches bereits eingesammelt wurde, noch einmal in die Wortsammlung zu ziehen, aber das Wort kann außerhalb der Wortsammlung benutzt werden, um beispielsweise auf eine Frage zu antworten.

Das Einsammeln von Worten und das Antworten mit ihnen wird durch die Implementierung eines Doppelklicks vereinfacht. Ein doppelt geklicktes Wort wird dorthin bewegt, wo es für das Wort momentan am meisten Sinn ergeben würde, sich hinzubewegen. So würde sich ein Wort, dass im Dialog doppelt geklickt wurde in die Wortsammlung bewegen und ein Wort aus der Wortsammlung würde sich (wenn vorhanden) in eine Antwort bewegen. Ist kein erkenntliches Ziel vorhanden oder befindet sich das geklickte Wort bereits in der Wortsammlung, wird Spielenden das Feedback gegeben, dass ihre Aktion nicht möglich ist.

### 4.2.2. Worte speichern

Ist ein Wort aus dem Dialog ausgewählt worden, kann es nun in der Wortsammlung gespeichert werden. Hierzu wird es einfach in das dazugehörige UI-Feld gezogen. Das UI-Feld reagiert darauf, welcher Wortkategorie das momentan ausgewählte Wort angehört und speichert Worte abhängig von ihrer Kategorie. Die momentan implementierten Kategorien sind *Name*, *Ort*, *Adjektiv*, *Objekt, Konzept* und *Ja und Nein,* ebenso wie die Sonderkategorien *Alle* und *Favoriten*. Jede Kategorie wird durch eine Farbe repräsentiert.

Wird ein Wort ausgewählt, wird die Kategorie des Wortes ausgelesen und die jeweils richtige Seite in der Wortsammlung geöffnet. So würde das Wort *Stift* beispielsweise als *Objekt* eingestuft werden, während das Wort *Gut* Teil der Kategorie *Adjektiv* wäre. Eine momentan inaktive Kategorie ist die Wortgruppe *Andere*, die Worte speichert, die nicht wirklich in eine Kategorie. Wird die Sonderkategorie *Alle* geöffnet, werden alle gespeicherten Worte angezeigt. Ähnlich werden in der Sonderkategorie *Favoriten*, alle Worte angezeigt, die von den Spielenden mit einem Stern markiert wurden.

Die Organisation der Worte nach Kategorien und Favoriten soll dabei helfen, ein gesuchtes Wort so schnell wie möglich in der Wortsammlung zu finden. Worte, die nicht mehr benötigt werden, können auf das Mülleimer-Icon gezogen werden und somit weggeworfen werden.

Ich musste bemerken, dass es oft schwer war, ein Wort gut einordnen zu können, ohne zu viele Kategorien zu erstellen, weshalb sich die Kategorien über die Zeit viel geändert haben oder noch einmal unterteilt werden mussten. Auch musste ich schnell Unterkategorien einführen, was bedeutet, dass jedes Wort, das sich in einer Kategorie befindet, immer auch eine Unterkategorie besitzt. So wäre das Wort *China* beispielsweise ein Ort mit der Unterkategorie *Land*, während *London* ein Ort mit der Unterkategorie *Stadt* wäre. Diese Kategorisierungen halfen mir vor allem dabei, zu limitieren, was auf Fragen geantwortet werden konnte, und was nicht.

### 4.2.3. Antworten

Wird den Spielenden eine Frage gestellt, öffnet sich ein kleines Fenster im User Interface, in dem sich eine Box befindet, in die Spielende ein Wort einfügen können. Diese Box nimmt jeweils eine Farbe an, die Spielenden zeigt, welche Kategorie (oder welche Kategorien) als Antwort zugelassen sind. Wird ein Wort eingefügt, reagiert der NPC abhängig von der gegebenen Antwort auf die Frage. Nach der Auswahl einer Antwort, die in das Feld passt, muss die Auswahl bestätigt werden, ehe der Dialog fortgesetzt wird.

Hierbei unterscheiden sich Ja-Nein-Fragen und Fragen, die ein Wort als Antwort verlangen. Da erwartet wird, dass Spielende entweder die Worte *Ja* und *Nein* zu jeder Zeit in der Wortsammlung haben, können Ja-Nein-Fragen nicht übersprungen werden. Fragen, die erwarten, dass ein Wort einer gewissen Kategorie angegeben wird, werden jedoch von einem Knopf begleitet, der den Spielenden erlaubt, die Frage je nach Kontext zu verlassen oder zu überspringen. Da Spielende möglicherweise ein anderes Wort nutzen möchten, als sie zurzeit im Inventar haben oder schlichtweg kein Wort der Kategorie im Inventar haben, wird hier eine Möglichkeit benötigt, die Frage unbeantwortet zu lassen.

In einer Antwort auf eine Frage kann auch eine falsche Behauptung angestellt werden. Ob der angesprochene NPC bemerkt, ob es sich hierbei um eine Lüge handelt, hängt davon ab, ob das System an dieser Stelle prüfen möchte, ob gelogen wurde oder nicht. Während es wenig Sinn ergeben würde, die Wahrheit der gegebenen Antwort auf die Frage „Wie geht es dir?“ zu überprüfen, kann es Sinn ergeben zu überprüfen, ob die Antwort auf die Frage „Wo hast du Matheo gesehen?“ der Wahrheit entspricht.

### 4.2.4. Nachfragen

Zu jeder Zeit, in der sich Spielende in einem Dialog befinden, können sie auf einen UI-Knopf in der oberen Ecke des Dialogfensters klicken, der ihnen erlaubt, Nachfragen anzustellen. Vom Aufbau her, funktionieren Fragen genauso, wie Antworten: Ein Fenster öffnet sich, in das ein Wort gezogen werden kann, und wenn die Auswahl bestätigt wird, beginnt der Charakter, auf die Frage zu reagieren. Sei es, dass sie im Dialog gerade ein Wort gehört haben, das sie nicht verstehen, oder dass sie dem Charakter eine persönliche Frage stellen möchten – ihnen stehen die meisten Worte aus ihrer Wortsammlung zur Verfügung, um eine Frage zu stellen. Die Ausnahme sind hierbei Worte, die generell in einer Frage keinen Sinn ergeben würden. So könnte gefragt werden *„Was kannst du mir über die Hochzeit erzählen?“*, aber nicht beispielsweise *„Was kannst du mir über ‚perfekt‘ erzählen?“*. Da trotzdem nicht jeder NPC jedes Wort verstehen kann, weil ihm möglicherweise die Kenntnisse oder der Kontext zu der gestellten Frage fehlen, können Charaktere auf solche Fragen mit einer Rückgriff-Antwort der Art *„Das verstehe ich nicht“*, antworten. Trotzdem habe ich versucht, diese Antwort so selten, wie möglich zu nutzen und Spielenden so oft, wie ich konnte irgendeine eine Art von Antwort zu geben. Denn, auch wenn ihnen die Antwort an dieser Stelle nicht weiterhelfen würde, wollte ich Spielende ermutigen, auch weiterhin Fragen zu stellen, anstatt aufzugeben, nachdem sie auf gestellte Fragen keine Antworten erhalten haben.

Nachfragen werden in meiner Arbeit an verschiedenen Stellen benutzt: einerseits können Spielende Fragen über spezifische Gesprächsthemen stellen, andererseits können Spielende auch Nachfragen zu persönlichen Themen stellen und so Charaktere nach Familie, Wohnort oder ihrer Meinung zu anderen Charakteren fragen. Zuletzt können Nachfragen auch genutzt werden, um Aufgaben in der Szene zu erfüllen. Das könnte in einem größeren Spiel auch eine Wegbeschreibung in eine andere Stadt oder ein Hinweis zu einer längeren Aufgabe sein.

## 4.3. Systemkonzepte

Neben den Mechaniken, die dazu benötigt werden, um das Spiel zu steuern, habe ich ebenfalls einige Konzepte eingebaut, die für das System nicht notwendig sind, aber generell das Spiel bedienbarer machen oder einen Ausblick darauf geben, wie das System erweitert werden könnte und wie es in einem Spiel aussehen könnte.

### 4.3.1. Spielstart

Da es schwer einzuschätzen ist, in welcher Reihenfolge Spielende mit NPCs interagieren werden, war es wichtig, dass sie einige Worte bereits vor Beginn der Szene besitzen – so zum Beispiel *Ja* und *Nein*, aber auch den Namen ihres Charakters und Adjektive, um auf Fragen wie *„Wie geht es dir?“,* zu antworten. Obwohl es natürlich möglich ist, einen Dialog jederzeit zu verlassen, falls den Spielenden eine Antwort fehlen sollte, sollten die ersten Fragen und Dialoge nicht abschreckend wirken. Ich wollte vermeiden, dass Spielende das Gefühl bekommen, dass sie etwas falsch gemacht haben und sich entmutigt fühlen, Gespräche zu beginnen, da sie auf keine der gestellten Fragen antworten können. Ebenso wollte ich vermeiden, dass ich Spielenden zu oft eine Antwort vorgab: So habe ich einige Fragen zu Beginn so gestellt, dass sich mögliche Antwortmöglichkeiten bereits in der Fragestellung befinden, damit die Spielenden nicht in der Frage stecken bleiben. So gibt beispielsweise die Frage *„Ja oder Nein: Sind wir hier verabredet?“,* den Spielenden die Chance, *Ja* und *Nein* einzusammeln, sollten sie sich noch nicht in der Wortsammlung der Spielenden befinden. Da es allerdings schwer ist, vorherzusehen, in welcher Reihenfolge Dialoge gestartet werden, müssten diese Wortvorschläge vor fast allen Fragen in der Szene kommen. Das wollte ich auf jeden Fall vermeiden, da es einerseits den Sinn der freien Wahl einer Antwortmöglichkeit relativiert und andererseits an vielen Stellen eher unsinnig klingen würde. Während die oben genannte Frage möglicherweise wirklich gestellt werden könnte, macht eine Frage wie *„Wie geht es dir? Gut, normal oder eher schlecht?“*, einen eher merkwürdigen Eindruck.

Um diese Probleme zu lösen, experimentierte ich mit verschiedenen Ideen für den Beginn des Spiels. Zunächst erschien zu Spielstart ein Menü, in welchem den Spielenden verschiedene Fragen gestellt wurden, wie *„Was ist dein Name?“*, und sie bat, mehrere Charakterzüge auszuwählen. Aus den gegebenen Antwortmöglichkeiten wurde dann ein Pool an Worten generiert, mit dem Spielende das Spiel starten konnten. Das Beantworten dieser Fragen und das Erhalten der ersten Worte hatte allerdings sehr wenig damit zu tun, wie der Rest des Spiels mit Worten umging. Ich bemerkte in Tests sogar, dass es Spielenden eine falsche Erwartung davon gab, wie das Spiel funktionierte. Deshalb beschloss ich schlussendlich ich mich, den ersten Abschnitt des Spiels noch einmal zu verändern.

In der aktuellen Version des Spiels beginnt die Szene mit einem Tutorialcharakter, der den Spielenden die groben Grundlagen des Spiels erklärt und diesen als Teil des Tutorials Fragen beantworten lässt. Je nachdem, wie sie auf die Fragen antworten, beginnt der Tutorialcharakter, ihnen ihre Situation zu erklären. Antworten sie also beispielsweise, dass sie mit der Braut verwandt sind, erfahren sie im Tutorial einige der Namen von anderen Verwandten, die auf der Feier sein werden. Diese Worte können in der Wortsammlung gespeichert werden und geben Spielenden schnell die Möglichkeit, eine Rolle in der Szene anzunehmen, da sie nun über anderes Wissen verfügen, als Spielende, die beispielsweise als ein guter Freund des Bräutigams in die Szene starten. Auch starten sie das Spiel durch das Tutorial jetzt bereits mit Worten wie *Ja* und *Nein* und einigen anderen Worten, die sie dort für relevant gehalten haben.

### 4.3.2. Weltkarte

Im oberen Teil der Szene kann eine Karte der Region entdeckt werden, die als eine Art Touristeninformation oder Wanderwegkarte die Region erklärt. Was ich hierbei zeigen wollte, war, dass die Wortsammlung nicht nur mit Worten aus einem Dialog gefüllt werden müssen, sondern Wissen auch an anderen Stellen gefunden werden kann. Spielende können die Namen der Orte, die auf der Karte eingetragen sind, genauso aufheben, wie sie sonst auch Worte aus dem Dialog aufheben können. Auch wenn diese Karte nur ein kleiner Einblick in dieses Konzept ist, gäbe es viele Möglichkeiten, Worte außerhalb von Dialogen einzusammeln, sei es auf Karten, in Logtexten oder Bucheinträgen oder in der tatsächlichen Spielewelt.

### 4.3.3. Begleiter

In der Szene kann Riley, der erste NPC, den Spielende in der Welt treffen, dazu überredet werden, den Spielenden während der Feier zu folgen. Riley soll generell darstellen, wie ein Charakter funktionieren könnte, der den Spielenden in einem Rollenspiel für längere Zeit folgt. Er kann mehrfach angesprochen werden, kann zu vielen Themen befragt werden und offenbart in diesen Befragungen auch mehr über sich selbst und sein Leben. Stellen Spielende Riley genug Fragen, erfahren sie so von seiner Arbeit, von seiner Familie, seinem Bruder und der Stadt, in der er groß geworden ist. Natürlich könnten größere Spiele hier noch mehr in die Tiefe gehen, was einen Begleiter angeht – von einem Ziel, welches er verfolgt zu seiner Meinung zu einer aktuellen Aufgabe.

# 5. Erweiterungsmöglichkeiten des Systems

In den achtzehn Wochen, an denen ich an dieser Bachelorarbeit gearbeitet habe, hatte ich viel Zeit, mir Konzepte zu überlegen, die ich gerne im System ausprobieren wollte. Nicht jedes dieser Konzepte konnte ich in dieser Zeit in mein Projekt einarbeiten und andere musste ich nach einiger Bearbeitungszeit wieder aus dem Projekt entfernen. Auf den folgenden Seiten findet sich die Sammlung an Konzepten für neue Mechaniken und Ideen für die Erweiterung von bereits existierenden Systemen.

## 5.1. Erweiterung der Nachfragen

Neben Sinnfragen oder dem Nachfragen nach der Meinung eines Charakters zu einem Thema, könnten Nachfragen auch genutzt werden, um einen Charakter nach einer bestimmten Aktion zu fragen. So müsste es beispielsweise bei einem Händler keinen spezifischen *Handeln*-Knopf geben, sondern Spielende können einen Händler auf *Handeln* ansprechen, um das Handelsmenü zu öffnen. Dieses System könnte auch für andere Aktionen funktionieren, wie zum Beispiel einen NPC zu bitten, mit einem ein Kartenspiel zu spielen oder einem Begleiter eine spezifische Anweisung zu geben.

## 5.2. Einbau eines Inventarsystems

Sollte das System in ein Spiel eingebaut werden, in dem Spielenden ein Inventar zur Verfügung steht, könnte das Inventar ebenfalls genutzt werden, um aus besessenen Gegenständen Worte zu erschaffen. Das würde die Organisation in der Wortsammlung erleichtern, da somit nicht jedes Item, was sich im Inventar der Spielenden befindet, in der Wortsammlung gespeichert werden muss. Mit diesem Ansatz könnten Items aus dem Inventar ebenfalls als Worte genutzt werden, nach denen gefragt oder mit denen geantwortet werden kann.

Auch wenn Spielende in einem Laden Items kaufen oder verkaufen, könnte so der Handel über einen Dialog stattfinden, da jedes Wort, welches sich im eigenen Besitz oder im Besitz des Gegenübers befindet, so zu einem Wort gemacht werden kann. Das vereinfacht es auch, nach einem Item zu fragen, welches Spielende kaufen möchten, sollte der Bedarf einer Erklärung bestehen.

Probleme, die sich hierbei ergeben könnten, sind, dass wenn ein Item verbraucht wird, das dazugehörige Wort in diesem Fall nicht mehr zur Verfügung stehen würde, obwohl Spielende es möglicherweise weiterhin nutzen wollen. Somit müssten sie das Wort vorher in ihrer Wortsammlung speichern, wenn sie es nicht verlieren wollen.

## 5.3. Einbau eines Tausch- und Handelsystems

Handel kann in zwei verschiedenen Kontexten betrachtet werden: Zum einen könnten Spielende einen Handel durchführen, indem sie mit einem Händler einen Dialog führen, in dem sie Worte anbieten, die Items beschreiben, die sie in ihrem Besitz haben und dafür im Dialog Items fordern, die der Händler in seinem Besitz hat. Zum anderen könnten Spielende mit den tatsächlichen Worten in ihrer Wortsammlung handeln, unabhängig davon, ob das Wort ein physisches Objekt beschreibt, welches sie besitzen.

Die erste Situation lässt sich vor allem in Kombination mit dem Konzept eines Inventarsystems gut umsetzen, da somit garantiert ist, dass Spielenden Worte für jedes Item ihrem Besitz zur Verfügung stehen, ebenso, wie für alle Items, die ihnen beim Handel angeboten werden.

Die zweite Situation könnte eine Möglichkeit bieten, Worte, die Spielende im Moment nicht brauchen einzutauschen, um relevantere Worte zu erhalten. Möglicherweise könnten die Worte auch später zurückgetauscht werden. Hierbei könnte jedes Wort eine Art Wert besitzen, je nachdem wie selten das Wort ist bzw. wie schwer es ist, das Wort zu bekommen. Das würde auch Worten, die im Moment wenig Nutzen haben, eine größere Bedeutung geben, weil sie immer noch für einen gewissen Wert eingetauscht werden können. Dieser Wert könnte manuell in die Datenbanken eingetragen werden oder es könnte ein Tool geschrieben werden, welches überprüft, wie oft jedes Wort in Dialogen vorkommt.

## 5.4. Speichermöglichkeiten für Worte

Ähnlich wie eine Truhe, in die Gegenstände gelegt werden können, die Spielende behalten möchten, aber im Moment nicht brauchen, könnte es eine Art Ablage geben, in der Worte gespeichert werden können, die momentan nicht gebraucht werden, aber generell nicht weggeschmissen oder verkauft werden sollen. Eine Ablage könnte auch dafür sorgen, dass es sich weniger bestrafend anfühlt, dass die Wortsammlung weniger als zwanzig Worte in jeder Kategorie zulässt, ohne dass die Wortsammlung so zu groß und unübersichtlich wird. Möglicherweise können Worte, die sich nicht in der Wortsammlung befinden, aber bereits in der Ablage liegen, eine andere Farbe im Text haben, als bereits eingesammelte Worte, um zusätzliches Feedback zu geben.

## 5.5. Einbau eines Auftrags- und Questsystems

Aufträge waren für lange Zeit Teil des Systems. Da sie allerdings über die Zeit immer komplexer wurden, oft für Bugs gesorgt haben und schließlich nur noch wenig Nutzen hatten, habe ich sie im Januar entfernt. Da mein fertiges Werkstück zu klein war, um Aufträge wirklich darstellen zu können und es generell sehr schwer war, zu erklären, wie sie zu benutzen waren, beschloss ich, sie zunächst bei einem Konzept zu belassen.

Möglicherweise können weiterentwickelte Versionen meines Systems ein Auftragssystem beinhalten, in welchem die Spielenden Aufträge, ähnlich wie Worte, in ein Auftragsmenü ziehen können, um sich diese zu speichern. Aus diesen gespeicherten Aufträgen können einzelne Worte herausgenommen werden: So kann jederzeit aus dem Auftrag *„Finde Matheo“*, der Name *Matheo* genommen werden, was es Spielenden einfacher macht, Aufträge zu managen. Um den Fortschritt, der in einem Auftrag gemacht wurde, überblicken zu können, sollen Aufträge als ein weiterer Speicherort für gefundene Worte dienen. Sollten Spielende also herausfinden, dass Matheo in der Nähe des DJs gesehen wurde, können sie das Wort „DJ“ zum Auftrag hinzufügen. Diese Methode, Worte zu speichern, soll auch dabei helfen, in einem Auftrag Fortschritte zu machen. Fragen Spielende den DJ nach dem Auftrag, ohne dieses Zusatzwissen zu haben, erhalten sie möglicherweise andere Informationen, als wenn sie nach dem Auftrag mit dieser Information fragen.

Generell mache ich mir bei einem Auftragssystem Sorgen, dass es womöglich zu wenig Nutzen hat, um die sehr komplexe Nutzung zu rechtfertigen. Möglicherweise kann eine vereinfachte Lösung gefunden werden, in der beispielsweise Aufträge gespeichert werden können, aber sie nicht ebenfalls Worte abspeichern können.

## 5.6. Erweiterung des Redehaltens

Da das Halten einer Rede nicht allzu oft für ein Spiel relevant ist, könnte es helfen, die Mechanik auch auf andere Situationen zu übertragen. So könnte langes Reden benutzt werden, um einen Charakter hinzuhalten, eine Geschichte zu erzählen, etwas, was passiert ist, zusammenzufassen oder möglicherweise sogar einen Plan zu erklären oder Anweisungen zu geben.

Für einige dieser Situationen könnte das System noch erweitert werden, in dem die Zeit, die den Spielenden zum Antworten bleibt, begrenzt wird. Während eine Zeitbegrenzung für eine Rede vor einem Publikum vielleicht weniger relevant wäre, könnte sie beispielsweise eine Situation interessanter machen, in welcher Spielende einen Charakter möglichst lange hinhalten müssen, um diesen abzulenken.

## 5.7. „*Wer-Kennt-Wen?“-*System

Da das System momentan gegebene Antworten speichert, um sie später noch einmal verwenden zu können, könnte ein System eingebaut werden, was überprüft, ob ein Charakter möglicherweise eine Information haben könnte, obwohl Spielende sie dem Charakter nicht gegeben haben. So könnte das System eine Datenbank enthalten, die speichert, welche Charaktere sich kennen oder miteinander befreundet sind, und dann, wenn es darum geht, Informationen abzurufen, nicht nur die eigenen Informationen, sondern auch die Informationen der Freunde und Bekannten des Charakters abzurufen. Hierbei muss natürlich beachtet werden, dass Freunde nicht jede Art von Information miteinander austauschen. Daher wäre es sinnvoll, wichtige gegebene Informationen zu markieren, damit das System weiß, welche Informationen ein Charakter weitergegeben hätte und welche nicht. So würden andere NPCs vielleicht wissen, wie der Spielercharakter heißt oder als was dieser arbeitet, aber sie wüssten eher weniger, was Spielende auf die Frage *„Wie geht es dir?“*, geantwortet hätten.

# 6. Code

Der folgende Teil soll sich damit befassen, wie das System meiner Bachelorarbeit tatsächlich im Code aussieht und aufgebaut ist. Hierbei werde ich vor allem die Aspekte eingehen, die für mich selbst technische Herausforderungen waren und werde erklären, wie sie funktionieren. Mein Fokus liegt hier auch vor allem auf Codestrukturen, die tatsächlich mit dem Dialogsystem zu tun haben und weniger auf nebensächlichen Details, wie beispielsweise der Steuerung der Spielfigur.

## 6.1. Textprojektion

Um die Dialoge zu schreiben und in der Szene erscheinen zu lassen habe ich mit dem *Unity*-Plugin *Yarn Spinner* gearbeitet, was es einfach gemacht hat, Texte zu verfassen und die richtigen Dialoge in den richtigen Momenten abzuspielen. Hierbei laufen in der Szene zwei Instanzen des *DialogueRunners*, den Yarn Spinner nutzt, um Dialoge in der Szene erscheinen zu lassen. Einer der beiden Runner kümmert sich hierbei um den normalen Dialog, der beim Ansprechen eines Charakters angezeigt wird, während der andere Runner sich um die Antworten auf Nachfragen kümmert. Wird eine Nachfrage gestellt, wird die Antwort über dem aktuellen Dialog angezeigt und gerendert. Erst, wenn die Antwort gegeben wurde, wird dieses Dialogfenster wieder ausgeblendet und das Gespräch, welches vor der Frage stattgefunden hat, wird fortgesetzt.

Ein Wort über dem Text erscheinen zu lassen, macht sich die Daten, die *TextMeshPro* zur Verfügung stellt, zunutze. TextMeshPro ist ein vorinstalliertes Unity Plugin, welches die genaue Position und die Begrenzungen von einzelnen Buchstaben oder Worten in einem Text bestimmen kann. Das half mir dabei, ein Wort exakt über dem existierenden Text instanziieren zu können. Beim Aufsammeln eines Wortes wird ein Wort erst über dem Text erstellt, von wo aus es aufgehoben und verschoben werden kann. Wird das Wort nun bewegt, wird es visuell verändert, um eher wie eine Wortkarte auszusehen, dabei wird aber kein neues Wort instanziiert.

## 6.2. Wortinformationen

Alle Worte, die für mich von Relevanz waren, schrieb ich in eine von mehreren Excel-Tabellen, die zu Beginn des Spiels ausgelesen werden und dann genutzt werden, um den Worten ihre korrekten Eigenschaften zuzuweisen. Ein Eintrag in die Tabelle speichert den tatsächlichen Namen des Wortes, die Kategorie, zu der es gehört, mindestens eine Unterkategorie, und in einigen Fällen eine Spezifikation. So ist beispielsweise bei Worten, der Kategorie *Gegenstände* spezifiziert, wie teuer sie ungefähr sind und bei Worten, der Kategorie *Orte* ist spezifiziert, ob es sich um eine Stadt, ein Gebäude oder ein Land handelt und wo sich dieser Ort befindet. Diese ausgelesenen Daten werden zunächst als Datensatz gespeichert und werden bei Erscheinen eines Texts mit den Worten, die sich im Text befinden, verglichen. Findet sich dabei ein Wort aus dem Datensatz, wird es farblich markiert. Erstellte Worte übernehmen dann einen gespeicherten Datensatz. Je nachdem, ob dieser Datensatz behauptet, das Wort wäre als Favorit markiert oder es gehöre zu einer gewissen Kategorie, wird es visuell anders dargestellt. Je nach Kategorie wird ein Wort mit einer anderen Farbe dargestellt. Um Unterkategorien visuell voneinander zu unterscheiden, generiert der Name der Unterkategorie jeweils einen Hash-Wert, aus dem eine Farbe generiert wird, die dann in die Farbe der Kategorie gemischt wird. Die Daten der momentan gespeicherten Worte werden in einer Liste festgehalten und können dort verändert werden. Aus diesen Daten werden beim Laden der Wortsammlung die Wortkarten erstellt.

## 6.3. Antworten und Nachfragen

Mit Yarn Spinner war es einfach, selbstgeschriebene Befehle in die Dialogdateien einzubauen. Das hat es ermöglicht, sehr effizient ein Frage- oder Antwortfenster zu öffnen. Um die richtige Frage oder Antwort zu öffnen, legte ich eine Tabelle mit IDs an, die an bereits vordefinierte Fragen geknüpft waren. Diese Tabelle speicherte ebenfalls, welche Kategorien jeweils als Eingabe erlaubt waren und welche nicht. Die gegebene Antwort kann dann durch einen anderen selbstgeschriebenen Befehl in den Dialog zurückgeholt werden, und ausgewertet werden. Es kann hierbei nicht nur der Name angefragt werden, sondern auch die Kategorie, Unterkategorie oder eine andere Spezifikation. So kann beispielsweise auf die Frage *„Wie geht es dir?“,* überprüft werden, ob das als Antwort gegebene Adjektiv als positiv, neutral oder negativ spezifiziert wurde. Antworten können auch langfristig gespeichert werden, wobei beim Speichern ein String beigegeben werden muss, über den die Antwort erneut aufgerufen werden kann.

Eine Nachfrage funktioniert dadurch, dass jeder Charakter zu verschiedensten Themen Antwortdialoge besitzt. Um den richtigen Antwortdialog für eine Situation zu finden, ist jede Antwort unter einem Namen im Stil *Charaktername*.*Gesprächsthema* gespeichert. Ist das System nun auf der Suche nach der korrekten Reaktion auf eine Frage, wird das System nach dem Namen des angesprochenen Charakters und dem aktuellen Thema durchsucht. Würden Spielende also James nach der Hochzeit fragen, wäre diese Antwort unter *James*.*Hochzeit* zu finden. Findet das System keine übereinstimmende Antwort, wählt es eine allgemeine Antwort – so wie *„Tut mir leid, ich verstehe nicht, was du meinst.“*

## 6.4. Adaptives System

Mir war es wichtig, ein besonders veränderbares System zu bauen, da viele Entscheidungen erst ausprobiert werden mussten, um zu sehen, ob sie funktionierten. Daher versuchte ich, so viele Variablen, und Daten einstellbar zu machen und Features schnell ein- und ausschaltbar zu machen. In meinem Projekt kann in kürzester Zeit eine Kategorie hinzugefügt oder gelöscht werden, ohne einen neuen UI-Knopf oder ähnliches erstellen zu müssen. Durch eine Excel-Tabelle werden spezifische Begriffe deklariert, die in der Wortsuche genutzt werden. So ist nirgendwo im Code deklariert, wie die Datensätze ausgelesen werden sollen, sondern stattdessen findet sich der Schlüssel zu den Daten schnell veränderbar in einer Tabelle.

Es war mir wichtig, dass UI Elemente problemlos verschoben werden können, ohne dass Probleme auftreten, Farben schnell verändert werden können und Dialoge schnell angepasst werden können.

Ein Datentyp, bei dem ich es nicht geschafft habe, ihn schnell und einfach ersetzbar zu machen, sind die Namen der NPCs gewesen. Da ich nicht ohne Weiteres auf die Dialoge zugreifen konnte, hätte der Name in jedem Dialogskript geändert werden müssen, ebenso wie die Namen der Dialogskripte und der Name jeder Node in der der Charaktername erwähnt wird.

## 6.5. Die Kategorie „Andere“

Die Kategorie *Andere* war für eine lange Zeit Teil meines Systems gewesen. Da die Worte momentan allerdings in keiner Situation im Werkstück einen Nutzen haben, sind sie momentan ausgeschalten. Die Kategorie *Andere*, soll für mehr Interaktion sorgen, da mehr Worte im Dialog farbig markiert sind. In späteren Systemen, die in das Werkstück eingeführt werden könnten, könnte die Kategorie eine interessante Rolle spielen, da dort beispielsweise mit diesen Worten gehandelt werden könnte.

Neben den normalen Listen, die relevante Worte speichern, kann im Unity Editor eine Liste generiert werden, die aus allen Worten besteht, die in Dialogen ignoriert werden sollen. Die Worte in einem Dialog werden dann mit dieser Block-Liste verglichen und wenn sie nicht in der Liste enthalten sind, werden sie farbig markiert. Worte, die so farbig markiert wurden, landen automatisch in der Kategorie *Andere*. Um dieser relativ langen Liste performantere Vergleiche zu erlauben, wird die Liste beim Generieren sortiert und das gesuchte Wort wird dann durch Binary Search mit der Liste verglichen, was die Komplexität des Suchalgorithmus deutlich verringert.

# 7. Evaluation

Nachdem ich achtzehn Wochen am Werkstück gearbeitet habe, habe ich während meiner Arbeit am Projekt und danach Zeit gehabt, um über das Projekt zu reflektieren und zu evaluieren, ob ich die Ziele, die ich mir gesetzt habe, eingehalten habe.

## 7.1. Evaluation der Arbeitsweise

Generell bin ich mit meiner Arbeitsweise sehr zufrieden, da ich im Code sauber gearbeitet habe und den Code auch gegen Ende des Projekts noch gut anpassen und instand halten konnte. Ich habe meine Arbeit gut dokumentiert und konnte schnell Bilder des Prozesses aus jeder Phase des Projekts finden. Trotzdem hätte ich an einigen Stellen eher über Systeme reflektieren sollen, die in meinem System wenig Nutzen hatten – Wie beispielsweise das Aufgabensystem, in welches ich viel Arbeit gesteckt hatte, um es sauber zum Laufen zu bekommen, nur um kurz danach zu bemerken, dass es zu komplex war und den Spielenden im Gegenzug sehr wenig gebracht hat. Gegen Weihnachten habe ich begonnen, das Buch *Clean* *Code* zu lesen, was meine Sichtweise auf viele Aspekte meines Codes drastisch änderte. Da ich meinen gesamten Code nicht in den letzten drei Monaten des Projektes umarbeiten wollte, bin ich mit neueren Teilen meines Codes mehr zufrieden, als mit alten. Trotzdem denke ich, dass ich viel gelernt habe und bin mit meiner Arbeitsweise im Allgemeinen zufrieden.

## 7.2. Evaluation des fertigen Werkstücks - Positives

Auch bin ich sehr glücklich mit meinem fertigen Werkstück. Vor allem die Möglichkeit, Eigeninitiative zu ergreifen, um Fragen zu stellen und die größere Bandbreite an Antwortmöglichkeiten, die auf Fragen gegeben werden kann, funktionieren in diesem System sehr gut. Spielende haben weniger das Gefühl, dass sie aus vorgegebenen Möglichkeiten auswählen und sind freier darin, auch auf eher charakterbezogene Fragen, wie „Wo kommst du her?“, zu antworten, ohne dass das System jede mögliche Auswahlmöglichkeit bereitstellen muss. Das System macht das Lesen eines Textes interaktiver und das Aufnehmen der Informationen wird zu einem Teil des Spiels, denn nach jedem gelesenen Wort stellt sich die Frage, ob dieses möglicherweise im Spielverlauf helfen könnten. Auch können sich Spielende so clever fühlen, wenn sie eine unkonventionelle Antwort auf eine Frage wählen – besonders, wenn das System tatsächlich darauf reagiert. Ich mag, dass das System nicht übermäßig komplex geworden ist, und auch mit vielen Worten noch relativ überschaubar bleibt. Das System fördert, dass Spielende sich nicht gezwungen fühlen müssen, jede Person nach jeder Sache fragen zu müssen, die angeboten wird, und sich nur mit Charakteren und Themen auseinanderzusetzen, die sie selbst interessant finden. Dadurch, dass sie jeden Charakter zu jedem Thema ausfragen können, fühlt es sich weniger so an, als wären sie gezwungen den Dialogbaum eines Charakters „abzuarbeiten“, sondern sie stellen eher Fragen zu Themen, die sie persönlich tiefer erforschen möchten. Das bietet meiner Meinung nach viele Möglichkeiten, Charaktere interessant zu schreiben, da so sehr einfach viele Details und Informationen in einen Charakter hineingearbeitet werden können.

## 7.3. Evaluation des fertigen Werkstücks - Negatives

Trotzdem hat das System noch viele Probleme. Zum einen ist es noch immer nicht möglich, mehr als simple Antworten auf Fragen zu geben, die den Spielenden gestellt werden. Zum anderen müssen Spielende momentan auf sehr viele Eventualitäten eingestellt sein, damit sie nicht immer wieder zu alten Dialogen zurückkehren müssen, wenn sie eine passende Antwort gefunden haben und so kann das Spiel sich zurzeit ab und zu anfühlen, als sei es ein Puzzle. Vor allem, wenn das Spiel in einem größeren Rahmen existiert, kann es schwer werden, viele Namen, Orte und Objekte zu balancieren und tatsächlich die richtigen Worte in der Sammlung zu haben, wenn sie benötigt werden.

Zudem hatte ich mir die Aufgabe gestellt, das System für Rollenspiele jeder Art benutzbar zu machen. Leider muss ich hier feststellen, dass ich nicht sehe, dass das System so, wie es im Moment ist, in ein größeres Spiel (der Art Dragon Age oder Fallout) integrierbar ist. Das System müsste zumindest mit auswählbaren Antwortmöglichkeiten unterstützt werden, da es schwer ist, komplexe Antworten oder Gedanken zu vermitteln, die häufig benötigt werden.

## 7.4. Fazit

Generell denke ich, dass das System vielversprechend ist, um später noch einmal damit weiterzuarbeiten. Hierbei würde es möglicherweise helfen, genau vor Augen zu haben, welche Arten von Dialogen tatsächlich in einem Spiel benötigt werden, um das System dann auf die benötigten Mechaniken und Systeme anzupassen. Vor allem für größere Spiele müssten hier Entscheidungen getroffen werden, die es einfacher machen, mehr Worte zu organisieren und unter Kontrolle zu behalten. Möglicherweise können für viele der momentanen Probleme noch Lösungen gefunden werden und vor allem für kleine Spiele sehe ich die Möglichkeit, durch dieses System interessante Geschichten zu erzählen.

# 8. Quellen

https://www.youtube.com/watch?v=LR4OxNfzTvU

# 9. Genutzte Software

Unity 2020.3.20f1

Adobe Photoshop CC 2021

Microsoft Excel 2016

Miro

Yarn Spinner 1.2.7

Unity High Definition Render Pipeline 10.6.0

HDRP UI Camera Stacking by Antoine Lelièvre 1.2.1

Visual Studio Editor 2.0.14

Visual Studio Code Editor 1.2.4

# Abbildungsverzeichnis

# Danksagung

Das Schreiben dieser Bachelorarbeit wäre nicht möglich gewesen, ohne die Hilfe von vielen verschiedenen Menschen, denen ich in diesem Abschnitt danken möchte. Ich möchte mich herzlich bedanken bei…

…Prof. Thomas Bremer für sein Feedback und die Möglichkeit, Probleme immer in seiner Sprechstunde zu besprechen.

…bei Prof. Susanne Brandhorst und dem Studiengang Game Design für die Räumlichkeiten, die verfügbare Technik und die Software, die uns zur Arbeit zur Verfügung gestellt wurde.

…bei den Kommilitonen, mit denen ich mir einen Arbeitsplatz geteilt habe, für die regelmäßige Hilfe bei Problemen oder Entscheidungen. Und dafür, dass sie sogar in ihren Mittagspausen bereit waren, sich das ein oder andere Problem anzuhören. Mein besonderer Dank geht hier an Philip Hildebrandt, Daria Pankau und Felix Szczesny.

…und bei allen meinen anderen Kommilitonen, die mir immer zur Verfügung standen, um meine Arbeit zu testen oder anderweitig Feedback zu geben.

# Erklärung der Autorenschaft

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbstständig angefertigt habe und keine anderen als die angegebenen Quellen benutzt habe. Alle wörtlichen oder dem Inhalt nach aus fremden Arbeiten entnommenen Stellen, bildlichen Darstellungen und dergleichen habe ich als solche genau kenntlich gemacht.

Berlin, 21.03.2022

Unterschrift