**Destiny’s Reminder**

**Eine interaktive Geschichte**



Ein Projekt von: Carole Meier  
Abteilung: G4C  
Datum: 31.8.23  
Schule: Kantonsschule Wettingen (KSWE)  
Betreuende Lehrperson: Oliver Probst   
Gegenleserin: Sara Nyffenegger

**Zusammenfassung**

In dieser Arbeit wurde die Videospielart Visual-Novel genauer untersucht um ein eigenes Spiel dieser Art herzustellen. Das Ziel war das Schreiben und Gestalten einer interessanten, interaktiven Geschichte und diese dann durch das Programmieren eines Visual-Novels spielbar zu machen. Aus welchen Bestandteilen ein solches Spiel aufgebaut ist und wie man eine ansprechende, interaktive Geschichte mit verschiedenen Enden schreibt, wurde zu Beginn durch Recherche im Internet in Erfahrung gebracht. Für die Kreation des Spiels wurde sich dann mit der Programmier-Engine Ren‘ Py auseinandergesetzt. Mit den erhaltenen Erkenntnissen konnte so ein vollständiges und spielbares Skript programmiert werden, zu welchem später die passenden visuellen Elemente angefertigt und angefügt wurden.

**Vorwort**

Ein eigenes Spiel wie dieses zu schreiben und gestalten war schon lange eine meiner Ambitionen. Seit meiner Kindheit befasse ich mich in meiner Freizeit gerne mit Videospielen und Geschichten jeglicher Art. Mich in Erzählungen zu vertiefen aber auch eigene zu kreieren war schon immer eine meiner Lieblingsbeschäftigungen. Neben dem Lesen und Spielen dienen mir auch das Malen und Schreiben, um mich kreativ auszudrücken. Gemalt habe ich immer schon gerne, doch eigene Geschichten zu entwickeln und zu illustrieren begann ich erst so wirklich in den letzten Jahren. Während meiner Zeit an der Kantonsschule konnte ich dank dem breiten Freikursangebot bereits Erfahrungen im Bereich des literarischen Schreibens sammeln und meine Leidenschaft Geschichten zu erzählen weiter ausleben. Doch für mich wurde während dieser Zeit auch klar, dass ich meine Geschichten gerne auch visuell darstellen möchte. Es begann mit Comics und einzelnen Illustrationen zu eigenen Geschichten, doch durch mein Interesse an Informatik und Gamedesign entstand schlussendlich die Idee aus einer meiner Geschichten ein Spiel zu programmieren. Als die Wahl eines Themas für die Maturaarbeit anstand griff ich diese Idee wieder auf und entschied mich eine eigene interaktive Geschichte in Form eines Spiels programmieren.

**Inhaltsverzeichnis**

[1. Einleitung 1](#_Toc149197623)

[1.1 Themeneingrenzung 1](#_Toc149197624)

[1.2 Ziele 1](#_Toc149197625)

[1.3 Fragestellungen 1](#_Toc149197626)

[1.4 Methoden 1](#_Toc149197627)

[2. Theorie 2](#_Toc149197628)

[2.1 GitHub 2](#_Toc149197629)

[2.2 Interaktive Geschichten und Visual-Novels 2](#_Toc149197630)

[2.3 Ren.py 4](#_Toc149197631)

[2.4 Welche Bildelemente werden benötigt? 4](#_Toc149197632)

[3. Arbeitsprozess 6](#_Toc149197633)

[3.1 Konzipierung der Geschichte 6](#_Toc149197634)

[3.2 Planung der Handlungsstränge 8](#_Toc149197635)

[3.3 Schreibprozess des Skripts in Ren.py 8](#_Toc149197636)

[3.4 Illustrationsprozess der Spielgrafiken 11](#_Toc149197640)

[3.5 Verknüpfen von Skript und Grafik 14](#_Toc149197644)

[4. Schluss 16](#_Toc149197648)

[4.1 Beantwortung der Fragestellung 16](#_Toc149197649)

[5. Reflexion 19](#_Toc149197655)

[5.1 Inhalt 19](#_Toc149197656)

[5.2 Methoden 19](#_Toc149197657)

[7. Anhang 22](#_Toc149197658)

# **1. Einleitung**

## 1.1 Themeneingrenzung

Bla

## **1.2 Ziele**

Das Ziel dieser Arbeit war es, eine ansprechende interaktive Geschichte zu gestalten und diese durch das Programmieren eines Videospiels in spielbarer Form umzusetzen. Die Geschichte soll mehrere interessante Pfade und Enden besitzen, welche der Spieler bzw. die Spielerin mit seinen bzw. ihren Entscheidungen beeinflussen kann. Die Geschichte soll hauptsächlich aus natürlich und spannend gestaltetem Dialog der verschiedenen Charaktere bestehen und soll durch passende Grafiken ergänzt werden. Das Endprodukt soll ein in sich abschließendes und visuell ansprechendes Spielerlebnis werden.

## **1.3 Fragestellungen**

Eine interessante und interaktive Geschichte in Form eines Videospiels erstellen zu können erforderte eine ausführliche Auseinandersetzung mit folgenden Fragen:

* Wie schreibt man eine interaktive Geschichte?
* Welche Methoden gibt es, um eine Geschichte in spielbarer Form umzusetzen?
* Was braucht es, um einen „Visual-Novel“ zu kreieren?
* Wie verstärken visuelle Elemente den Eindruck einer Geschichte?
* Wie kann ich meinen Code strukturiert notieren und verwalten?

## **1.4 Methoden**

Da sich die Umsetzung einer Geschichte in eine spielbare Form aus einer Kombination vieler verschiedenen Bereichen zusammensetzt, waren Recherche und Arbeit in den Bereichen Literatur, Kunst und Informatik nötig. Zur Recherche diente hier hauptsächlich das Internet. Bezogen wurden verschiedene Dokumentationen der genutzten Programme, wie auch Anleitungsvideos zu spezifischen Problemen, welche während des Arbeitsprozesses auftraten. Literarische Recherche war nur minimal nötig, da bereits eine gute Grundlage und einiges an Erfahrung bezüglich literarischer Arbeit bestand. Als Vorbereitung auf das Schreiben der Geschichte wurden verschiedene Pläne und Mindmaps der möglichen Handlungen gezeichnet und auch ein übersichtliches Flussdiagramm der vollständigen Handlung erstellt . Geschrieben wurde die Geschichte und der Code anschließend direkt mit der Visual-Novel-Engine Ren’ Py. Als Hilfsmittel zur Planung und Illustration der Hintergründe diente die Bau-Funktion des Lebenssimulators „Sims 4“. Letztlich wurden dann die notwendigen Grafiken digital mit der App Procreate auf dem iPad gemalt und angefügt. Der Fortschritt der ganzen Arbeit wurde mittels GitHub festgehalten und abgespeichert.

# **2. Theorie**

## **2.1 GitHub**

A black cat in a circle

Description automatically generatedSchon von Beginn an war klar, dass in diesem Projekt selbst geschriebener Code involviert sein würde, welcher zu verwalten war. Als Werkzeug zum Speichern und Teilen des Codes wurde deshalb von der betreuenden Lehrperson GitHub1 empfohlen.

Abbildung (!)

«GitHub ist ein Onlinedienst zur Versionsverwaltung für Software-Entwicklungsprojekte»2.

Als eines der weitverbreiteten, kollaborativen Versionsverwaltung Tools dient GitHub heutzutage vielen Softwareentwicklern ihren Code über das Internet zu speichern und mit anderen zu Teilen.

Es ist jedoch auch für Einzelpersonen und jede Art von Projekt ein nützliches Hilfsmittel. Es ermöglicht kontinuierlich neue Versionen von Code-Dateien und ganzen Projekten festzuhalten und archivartig abzuspeichern.

Dank ihrem Cloud-Storage System kann mit einem GitHub Account reibungslos von verschiedenen Geräten und Personen an einem Projekt gearbeitet werden.

Um die Änderungen eines Dokuments oder eines Ordners aufzeichnen zu können, muss auf GitHub lediglich ein Repository generiert werden. Dieser agiert als Speichermedium aller Dateien, welche von GitHub für dieses spezifische Projekt erfasst werden sollen. Das Programm merkt, sobald eine Änderung innerhalb dieses Ordners vorgenommen wird und zeichnet diese kontinuierlich auf. Werden die Änderungen dann abgeschlossen, kann durch einen Commit-Befehl die nun neue Version des Projekts wieder mit der Cloud synchronisiert werden. Die alte Version wird dann ins Archiv abgespeichert und durch die neue Version ersetzt. Diese Chronologische Aufzeichnung kann sowohl als Protokoll wie auch als Backup verwendet werden. Falls ein Fehler entsteht oder Code aus älteren Versionen benötigt wird, können durch GitHub diese alten Versionen wiederhergestellt werden. Mit jedem Commit wird das Datum und die genaue Uhrzeit aufgezeichnet und der Nutzer bzw. die Nutzerin wird aufgefordert eine kurze Beschreibung der Änderungen anzufügen. Dies erleichtert es später bestimmte Versionen des veränderten Codes wiederzufinden. Abbildung (!) zeigt einen Ausschnitt des Commit-Archivs dieses Projektes.

A white paper with black lines

Description automatically generated

Abbildung(!) Github Commits vom 14. Oktober

Für dieses Projekt bedeutete die Nutzung von GitHub, dass kein zusätzliches Journal geführt werden musste. Den Code und die weiteren Spieldateien fortlaufend in die Cloud hochzuladen, ersetzte ein traditionelles Journal, da jede Änderung bereits auf GitHub aufgezeichnet wurde. Durch das Teilen des Projekt Repositorys mit der betreuenden Lehrperson konnte so auch der Fortschritt der Arbeit dokumentiert werden.

## **2.2 Interaktive Geschichten und Visual-Novels**

Interaktive Geschichten gibt es in vielen verschiedenen Formen und für jederlei Altersgruppen. «Choose your own Adventure» Bücher wie zum Beispiel die 1000 Gefahren Serie des Ravensburger Verlags3, sind zwar in Stil und Inhalt eher an Kinder gerichtet, zeigen jedoch erfolgreich wie rein Text basierenden Medien ein spielerisches Leseerlebnis bieten können. Auch bei digitalen Medien, wie Filmen und Videogames gibt es Methoden die Entscheidungen der Konsumenten bzw. Konsumentinnen in die Geschichte mit einzubeziehen. Die Folge «Bandersnatch»4 aus der Serie «Black Mirror» ist eines der bekanntesten Beispiele einer selbst gewählten Geschichte in einem Film. Inspiration für dieses Projekt bildete jedoch der Bereich der Videospiele. Aus den zuvor genannten Optionen bilden sie eine gute Mischung der kreativen Freiheit von Texten und den ansprechenden visuellen Elementen von Filmen. Als Ziel der Arbeit wurde deshalb ein kurzer selbst illustrierter Visual-Novel gesetzt.

**Visual-Novel:**

Wie der Begriff bereits impliziert, besteht ein Visual-Novel aus einer Kombination von Grafik, deshalb der Begriff «Visual», und Text, oder eben ein «Novel». Diese Elemente vereint bilden in einem Visual-Novel eine spielbare Geschichte. Oft wird das Genre deshalb als Unterklasse der Videospiele gesehen.

A video game of a cartoon character

Description automatically generatedIhren Ursprung nahm diese Spielart in Japan, wo sie auch heute noch sehr beliebt ist. Obwohl sie mittlerweile auch in anderen Ländern bekannt sind, ist der Einfluss japanischer Kultur, wie auch der von Manga und Anime, oft noch gut erkennbar. Beispiele bekannter Visual-Novels sind die «Persona»5 Serie von ATLUS und SEGA oder «Doki Doki Literature Club»6 welches von einem kleineren Indie-Studio namens Salvato programmiert wurde. Dank des eher simplen Aufbau eines Visual-Novel eignen diese sich gut als Projekt für kleinere Studios oder sogar Einzelpersonen. Auch als Einstiegsprojekt sind sie hilfreich, um sich mit dem Designen eines Spiels bekannt zu machen. Mit ihrem starken Fokus auf eine zentrale Erzählung und dem sehr limitierten, dialogbasierten Gameplay sind die Anforderungen um einen Visual-Novel zu kreieren deutlich geringer als bei anderen, komplizierteren Videospielen.

Abbildung (!)

Anders als bei den meisten Videospielen interagiert der Spieler bzw. die Spielerin in einem Visual-Novel über eine direkte Ich-Perspektive mit der Geschichte. Generell wird mit der Hilfe von Klicks durch die vielen Gespräche navigiert, welche teils auch durch eigene Entscheidungen beeinflusst werden können. Auf Grund des sehr einseitigen Gameplays, welches primär aus Lesen besteht, werden Visual-Novels jedoch zum Teil gänzlich von Videospielen getrennt.7 Sie besitzen viele Merkmale verschiedener, erzählerischer Medien und können deshalb nur schwer einer Gruppe zugeordnet werden. Visual-Novels mischen und nutzen die unterschiedlichen Stärken dieser Medien, um eine neue Form der Erzählung zu generieren.

Durch ihre limitierten, vorgefertigten Hintergründe und Charaktere wie auch der dialogbasierten Erzählungsweise sind Visual-Novels Dramen ebenfalls sehr ähnlich. Im Gegensatz zu anderen literarischen Erzählungsformen beinhalten sie beide in ihrer vorgeführten oder gespielten Form keine beschreibenden Textstellen und Begleitsätze. Stattdessen verlassen sie sich auf ihre Visuellen Elemente, um Gefühle und Atmosphären zu vermitteln. Was jedoch in einem Drama gesprochen wird, muss in einem Visual-Novel hingegen vom Spieler bzw. der Spielerin gelesen werden. Sie mischen so das visuelle Erlebnis eines Films oder Theaters mit leserischen Aspekten analoger Medien wie Büchern oder Comics.

All diese Aspekte vereint und kombiniert mit den interaktiven Elementen, welche meist in Videospielen zu finden sind, verleihen dem Genre des Visual-Novels sein sehr einzigartiges Spielerlebnis.

## **2.3 Ren.py**

A cartoon of a child holding a yellow duck

Description automatically generatedFrüh im Rechercheprozess dieser Arbeit fiel die Entscheidung, die Visual-Novel Engine Ren’ Py8 zu nutzen. Ren’ Py ist eine auf Python basierende Programmier-Engine, welche eine gute Grundlage bietet, um eigene interaktive Geschichte in Form eines Videospiels zu erstellen. Über 4'000 Spiele wurden bereits mit ihr erstellt und viele davon sind zurzeit im Internet erhältlich.9 Spiele wie das zuvor genannte „Doki Doki Literature Club“ entstanden ebenfalls mit der Hilfe der Ren’ Py-Engine. Da diese Spiele mit denselben Mitteln programmiert wurden, dienten sie auch als Inspiration für mögliche Themen und Spielmechaniken in Destiny’s Reminder.

Abbildung (!)

Ebenfalls schnell ersichtlich wurde, dass die Kombination des Python-Codes mit den bereits vordefinierten Befehlen durch Ren’ Py, einen verständlichen und äusserst intuitiven Schreibprozess ermöglichen würde. Das Programm eignete sich somit perfekt für dieses Projekt.

Ren’ Py stellt für jedes Spiel eine Vorlage zur Verfügung, welche zu belieben verändert und auf das geplante Spiel angepasst werden kann. Es erlaubt eigene Bilder einzufügen und macht es einfach diese später mit dem Code zu verknüpfen. Um auf alle notwendigen Dateien zugreifen zu können, erstellt das Programm für jedes Projekt einen Ordner. In diesem werden jegliche Spieldateien und Bilder abgelegt, um später vom Code genutzt zu werden. Da dieser Ordner bereits alle nötigen Dateien enthält, konnte er einfach zur Aufzeichnung des Fortschrittes und Backups in einem GitHub Repository platziert werden.

Der Code, welcher schlussendlich dann das Spiel bildet, wird von Ren’ Py in vier verschiedene rpy-Dateien unterteilt. Diese beinhalten jeweils einen der Aspekte des Gamedesign Prozesses und werden separat bearbeitet. Die „Skript“-Datei enthält, wie es der Name schon sagt, das ganze Skript des Visual-Novels. Hier wird der Ablauf der Handlungen des Spiels programmiert und alle dazu notwendigen Grafiken dargestellt. Die „GUI“-, „Screens“- und „Options“-Dateien enthalten den Code, welcher die Hintergrundprozesse des Spiels regelt. In der „GUI“-Datei kann das Aussehen der Benutzeroberfläche und jeglicher Menus angepasst werden. Die Screens-Datei enthält Befehle, welche definieren welche Elemente innerhalb dieser Menus angezeigt werden und wann diese aufgerufen werden. Letztlich beinhaltet die Options-Datei weitere Optionen, um die allgemeine Darstellungsweise des Spiels zu bearbeiten. Hier werden beispielsweise der Name des Spiels, die Wiedergabe-Geschwindigkeit des Texts oder die Übergänge beim Starten und Schliessen des Spiels definiert. Ren’ Py lässt seinen Nutzern bzw. Nutzerinnen die komplette Freiheit, wenn es um die Gestaltung der einzelnen Komponenten des Spiels geht.

Ist der Designprozess abgeschlossen kann Ren‘ Py sogar genutzt werden, eine App- oder Exe-Version des eigenen Projektes zu generieren.

## **2.4 Welche Bildelemente werden benötigt?**

Wie in Abschnitt 2.2 erwähnt, sind Bilder und Animationen ein ausschlaggebender Teil eines Visual-Novels. Herauszufinden wie diese zu gestalten waren, war deshalb ein wichtiger Schritt im Designprozess von Destiny’s Reminder.

Das Darstellen von Bildelementen ist durch Ren’ Py grundsätzlich keineswegs limitiert. Es können beliebig viele Grafiken und Animationen dargestellt werden und sogar die Benutzeroberfläche kann selbst angepasst werden. Welche Form die visuellen Elemente für dieses spezifische Projekt annehmen mussten, klärte sich erst nach einem genaueren Blick auf die Funktionen und Prozesse der Ren’ Py-Engine. Um eine genauere Idee zu erhalten welche Illustrationen für einen Visual-Novel benötigt werden, wurden die Grafiken gemäss ihrer Darstellungsweisen in vier Klassen unterteilt.

**Einfache Objekte**

Dies ist die einfachste der vier Darstellungsweisen und somit die freiste der Illustrationsklassen. Die für diese Grafiken verwendete Darstellungsweise wird benutzt, um jegliche nicht interagierbare Objekte darzustellen. Um ein solches Objekt darzustellen, muss es lediglich im Skript mit einem show-Befehl aufgerufen werden. Folgende Grafik wird dann im Format, in welchem sie im Projekt Ordner hinterlegt ist, angezeigt. Man findet hier keine Limitationen bezüglich Form und Grösse.

Charaktere wie auch andere einfache Objekte können so mit einer Vielzahl verschiedener Darstellungseinstellungen, wie zum Beispiel Position, Grösse oder Transparenz, zu einem beliebigen Zeitpunkt durch den show-Befehl aufgerufen und durch einen hide-Befehl wieder versteckt werden. Diese Art des Darstellens limitiert nicht die Anzahl der sichtbaren Objekte und es kann frei bestimmt werden, welche Grafiken Andere überdecken.

**Benutzeroberfläche**

Diese Klasse der Grafiken ist sehr ähnlich wie die der einfachen Objekte. Von Ren’ Py selbst werden beide gleichbehandelt, unterscheiden sich jedoch in Speicherort und Code, von welchem sie dargestellt werden. Um das Aussehen von Elementen wie Text-Boxen, Menus oder Icons anzupassen, müssen die jeweiligen Grafiken im Projektordner ausgetauscht werden. Man kann so komplett selbst entscheiden, wie beispielsweise der Dialog im Spiel dargestellt wird und welche Form, Grösse etc. die Text-Boxen und Menus haben. Diese Darstellungsweise wurde jedoch für dieses Projekt in eine separate Klasse eingeteilt, da sie nicht selbst im Skript-Code aufgerufen werden müssen. Sie werden durch einen separaten Teil des Codes, welcher, wie in Abschnitt 2.3 erwähnt, unter GUI gespeichert ist, aufgerufen. Das Verändern der Benutzeroberfläche erfolgte deshalb lediglich durch das ersetzten von bereits gegebenen Grafiken. Diese mussten jedoch auf die gegebenen Vorlagen abgestimmt werden.

Da der Fokus der Arbeit auf der Geschichte und dem Umsetzen des eigentlichen Spielerlebnis lag, wurde die Benutzeroberfläche nur minimal angepasst. Die meisten Elemente wurden übernommen oder nur leicht in Farbe oder Helligkeit angepasst. Bei die wenigen neu illustrierten Grafiken musste jedoch auf striktere Formatvorgaben geachtet werden, um den Vorgaben der bereits vorhandenen Elemente zu entsprechen.

**Hintergründe**

Wie auch die Elemente der Benutzeroberfläche, ähnelt die Klasse der Hintergründe den einfachen Objekten in vielen Aspekten. Sie werden gleich wie die Einfachen Objekte direkt durch das Skript aufgerufen, haben jedoch ein paar spezielle Eigenschaften. Diese entstehen durch den Befehl mit welchem sie dargestellt werden. Wenn ein Hintergrund durch den Befehl «Scene» aufgerufen wird, sorgt dies dafür, dass jegliche zuvor dargestellte visuellen Objekte verschwinden. So kann durch einen Scene-Befehl sowohl ein neuer Hintergrund gesetzt werden, wie auch direkt eine neue, leere Szene begonnen werden. Hintergründe haben ebenfalls eine Formatanforderung. Um korrekt vom Spiel dargestellt werden zu können, müssen die Hintergrund-Illustrationen bereits in derselben Grösse und Auflösung wie das Spiel selbst gegeben sein.

**Animationen und Videosequenzen**

Die letzte und speziellste Klasse bilden die Animationen und Videosequenzen. Ren’ Py kann nicht nur Bilder, sondern auch ganze Videodateien darstellen und während des Spielens abspielen. Da diese Art der Darstellungen für dieses Projekt jedoch nicht vorgesehen war, wurde sich mit den genauen Methoden, solche darzustellen, nicht befasst.

# **3. Arbeitsprozess**

## **3.1 Konzipierung der Geschichte**

**Handlung**

Bevor mit dem eigentlichen Schreibprozess begonnen werden konnte, brauchte es eine packende Idee, welche die Handlungen des Spiels später tragen würde. Diese Idee kam in Form einer bereits begonnenen Geschichte. Diese spielte schon mit dem Konzept, jeden Tag fünf Errungenschaften zu erfahren welche im Verlauf des Tages geschehen würden.

Um den Umfang des Projekts etwas zu limitieren wurde der Fokus der Geschichte auf fünf Hauptcharaktere gesetzt, von denen einer die spielbare Figur und somit den Spieler bzw. die Spielerin repräsentiert. Die Errungenschaften, in diesem Fall Achievements genannt, erscheinen eines Morgens auf mysteriöse Weise und beginnen so die Geschichte. Während dem Verlauf des Spiels erfährt der Hauptcharakter Destiny, auf einem für andere unsichtbaren Textfeld, was am kommenden Tag geschehen würde. Die zu Beginn eher skeptische Destiny lernt im Verlauf der fünf Tage wie sie mit diesen Prophezeiungen umgehen muss und was sie auslöst. Um dem Spieler bzw. der Spielerin ein Ziel zu geben auf welches hingearbeitet werden kann, wurde die Geschichte zu einem Krimi gemacht. Im Laufe der Geschichte erfährt die Hauptfigur Destiny von einer Entführung. Eine der anderen Figuren, Kai, scheint zuerst lediglich untergetaucht, beginnt jedoch in Destiny’s Träumen aufzutauchen und liefert Hinweise zu ihrem Verschwinden. Destiny, nun in die ganze Situation verwickelt ist versucht dieser Entführung auf den Grund zu gehen. Im Verlaufe des Spiels muss sich dann der Spieler bzw. die Spielerin zwischen den drei restlichen Figuren entschieden und den Täter finden. Basierend auf Hinweisen, welche man in den fünf Tagen erhält, zeigt sich welche der drei in der Entführung involviert ist und wie Destiny Kai helfen kann. Das Ziel soll es sein Kai zu retten und den Täter zu schnappen. Die Geschichte soll zu Beginn sehr fröhlich und nichts ahnend erscheinen, doch dann sich weiter und weiter verwickeln bis zum Teil sogar eine düstere Atmosphäre entsteht.

Die klare fünf Tage-Struktur eignete sich gut für ein Spiel dieser Art, da die kapitelartigen Etappen dem Spieler bzw. der Spielerin eine klare Struktur mit Abwechslung bietet und gleichzeitig auch genutzt werden konnte, um einen klassischen Spannungsbogen aufzubauen. Den Tagen wurde, wie in einem klassischen 5 Akt Drama10, jeweils die Exposition, die steigende Handlung, der Höhepunkt, die fallende Handlung und die Lösung des Konflikts zugeteilt.

**Charaktere**

Da ein erheblicher Teil der Geschichte auf den Gesprächen zwischen Destiny und den drei erwähnten Charakteren beruht, wurde großen Wert darauf gelegt jeden einzelnen einzigartig zu gestalten und ihnen eine individuelle Persönlichkeit zu verleihen. Hierbei wurden die Persönlichkeit der entführten Figur, Kai, und die von Destiny, dem Hauptcharakter, absichtlich eher neutral gehalten. Man erhält so etwas mehr Freiraum sich selbst in diese Figuren hineinzuversetzen. Eine neutralere und somit formbarere Persönlichkeit bedeutete, dass sich die Entscheidungen später natürlicher anfühlen würden, sowohl für den Spieler bzw. die Spielerin wie aber auch für Destiny als Figur. Um jedoch eine solide Grundlage zu legen auf welche die Geschichte aufbauen konnte, wurden dennoch einige ihrer Charakterzüge und Eigenarten eingefügt. Diese helfen ihre Interaktionen mit den anderen Figuren prägen und eine realistischere Dynamik zwischen ihnen herstellen.

Destiny

Destiny ist eine eher schüchterne, aber talentierte junge Frau. Manchmal etwas unordentlich und unorganisiert. Diese eher unvorteilhaften Charaktereigenschaften zu zeigen, diente dazu sie dem Spieler bzw. der Spielerin sympathischer zu machen und gleichzeitig ihre Unbeholfenheit in einer neuen Situation zu Illustrieren. Viele der Entscheidungen würden auf kleine Fehler aufbauen, welche ihr in ihrem Alltag geschehen. Während dem Verlauf der Geschichte erfährt man, dass Destiny vor etwa einem Jahr einen neuen Lebensabschnitt bei einer neuen Firma und einem neuen Wohnort begonnen hatte. Ihre Unerfahrenheit und folglich auch Naivität werden ihr jedoch schlussendlich zum Verhängnis.

Gabriel, Matthew und Simon, die drei Figuren, mit welchen sich Destiny in den fünf Tagen am meisten unterhält, wurden jeweils gemäss einem der vielen Klischees eines potenziellen Antagonisten geschrieben. Da der Spieler bzw. die Spielerin am Ende der Geschichte entscheiden muss, wer der wahre Täter ist, mussten alle möglich suspekt bleiben. Das Klischee des „Falschen besten Freunds“, des „Tyrann“ und des „Wolf im Schafspelz“ können alle erfolgreich genutzt werden, um sowohl den wahren Antagonisten zu verkörpern oder als falsche Fährte zu dienen. Sie auf diese sehr oberflächlichen und bekannten Klischees zu reduzieren, machte es einfacher starke und klar motivierte Persönlichkeiten zu kreieren. Diese Stereotypen helfen ebenfalls dem Spieler bzw. der Spielerin, bereits früh die verschiedenen Charaktere zu verstehen und ihre weiteren Handlungen abzuschätzen.

A cartoon of a person waving

Description automatically generated

**Gabriel**

Gabriel ist der „beste Freund“ oder „falsche beste Freund“. Er ist sehr fürsorglich und immer nett zu Destiny. Er versucht oft das Beste aus einer Situation zu machen und rettet sie aus unangenehmen Situationen. Er und Destiny sind bereits zu beginn der Geschichte befreundend und verbringen so natürlich am meisten Zeit miteinander. Je nach Route, welche der Spieler bzw. die Spielerin einschlägt, hilft Gabriel den Täter zu schnappen. Jedoch kann er auch verdächtigt und von Destiny genauer verfolgt werden.

Abbildung (!)

A person wearing a tie

Description automatically generated

**Matthew**

Matthew füllt die Rolle des „Tyrann“. Als Destiny’s Chef und Leiter der Abteilung passt seine eher sehr kalte und egoistische Art zu seiner Position. Er nutzt oft seine Autorität, um andere einzuschüchtern und dadurch eigene Vorteile zu erzielen. Er ist ernster als die beiden anderen und zeigt offen, dass er Destiny nicht leiden kann. Seine narzistische Art und einige Auseinandersetzungen zur Entführung können Verdacht erwecken. Oberflächlich passt er am besten zum stereotypischen Bild eines Bösewichts, letztendlich hat er jedoch nichts mit der Entführung zu tun.

Abbildung (!)

A cartoon of a person

Description automatically generated

**Simon**

Simon ist der wahre Täter und stellt einen „Wolf im Schafspelz“ dar. Er ist zu Beginn sehr freundlich zu Destiny und hilft ihr sich in ein neues Projekt zu vertiefen. Er scheint stets unbesorgt und bleibt hilfsbereit. Was sich jedoch später als Fassade herausstellt. Er manipuliert viele der Situationen, um harmlos und unwissend zu wirken, zieht im Hintergrund jedoch die Fäden.

Abbildung (!)

## **3.2 Planung der Handlungsstränge**

Als die Charaktere konzipiert und die grobe Handlung des Spiels entschieden war, galt es die genaueren Details und Verzweigungen der Geschichte festzulegen. Hierbei war für das Planen der Interaktiven Momente zu beachten, dass durch stärkere Auswirkungen auf die Handlung oder vermehrtes Auftreten solcher Sektionen ein persönlicheres und somit spannenderes Spielerlebnis erzielt werden konnte. Gleichzeitig bedeutete jedoch jede neue Verzweigung erneuter Schreibaufwand und neuen Bedarf an Illustrationen. Folglich musste auch bei der Planung darauf geachtet werden die Anzahl und die Konsequenzen der Entscheidungen in einem umsetzbaren Rahmen zu halten. Eine gute Planung der Handlungsstränge half später auch Fehler zu vermeiden und den Überblick über die verschiedenen Szenen zu halten. Mit einem genauen Plan konnte während dem Schreiben des Skript-Codes der Überblick behalten werden.

Um die Handlung zu planen, eigneten sich zuerst Mindmaps und später Flussdiagramme, welche die verschiedenen Entscheidungen und die darauffolgenden Pfade klar visuell darstellen konnten. Ein solches wurde auf „Draw.io“11 generiert.

A diagram of a work flow

Description automatically generated

Abbildung1

Abbildung 1 zeigt einen Ausschnitt des detaillierten Flussdiagramms, welches alle möglichen Pfade und handlungsrelevanten Entscheidungen enthält. Das vollständige Diagramm ist im Anhang zu finden. Der Spieler bzw. Die Spielerin kann zu (!) Zeitpunkten, während dem Verlauf des Spiels zwischen zwei oder mehr Optionen wählen und so den weiteren Verlauf der Geschichte beeinflussen. Neben den handlungsändernden gibt es auch mehrere kleinere Entscheidungen, welche keinen oder minimalen Einfluss auf den allgemeinen Verlauf der Geschichte nehmen. Diese sind zwar eher insignifikant, ermöglichen jedoch ein individuelleres und interaktiveres Spielerlebnis, ohne den Zeitaufwand relevant zu erhöhen.

Im Ganzen besitzt die Geschichte (!) verschiedene Wege das Spiel zu beenden und vier Endsituationen, welche so erreicht werden können. Je nach Entscheidungen, welche gefällt wurden, tritt ein sehr schlechtes, schlechtes, gutes oder sehr gutes Ende ein. Das Ziel des Spielers bzw. der Spielerin soll sein, zu einem möglichst guten Ende zu gelangen.

Die Eingrenzung auf vier Hauptsituationen half eine fokussiertere Geschichte zu erzählen und das Projekt in einem umsetzbaren Rahmen zu halten. Die vier Situationen unterscheiden sich jeweils in zwei Aspekten voneinander. Einerseits beeinflussen die Entscheidungen, ob die entführte Person gefunden wird, andererseits bestimmen sie, ob der Täter gefangen wird.

## **3.3 Schreibprozess des Skripts in Ren.py**

### **3.3.1 Definieren der Charaktere**

Um mit dem Skript zu beginnen, mussten als aller erstes die Charaktere und die Darstellungsweise ihres Textes definiert werden. Um im Dialog zu erkennen wer gerade spricht, werden in einem Visual-Novel meist für jeden Charakter neben eigenen Grafiken auch unterschiedliche Text-Box-Designs oder Textformatierungen verwendet. So auch in Destiny’s Reminder. Die Formatierungen, welche in diesem Schritt definiert wurden, werden später genutzt, um im Spiel den Namen und den gesprochenen Text des Charakters darzustellen. Das Speichern all dieser Einstellungen erfolgt durch das Erstellen einer Variable, die später im Code aufgerufen werden muss wenn die durch sie definierte Figur spricht.

In Abbildung 2 ist ein Beispiel zu sehen in welchem der Charakter Gabriel definiert wurde. Als erstes wurde wie gemäss den zuvor erwähnt eine Variable gewählt, welche die Formatierungseinstellungen dieses spezifischen Charakters speichert. Durch das Abkürzen des Namens „Gabriel“ zu „g“ bleibt der Code verständlich, konnte jedoch später schneller geschrieben werden, ohne jedes Mal den ganzen Namen schreiben zu müssen wenn dieser Charakter etwas Neues sagt.

Der erste Wert, der auf das Definieren des Charakters folgt, speichert den Namen. Dieser muss als String eingegeben werden und wird in der Text-Box angezeigt, wenn Gabriel spricht. Mit Hilfe der nachfolgenden „Who“ und „What“ Parameter wurden dann die Formatierungen, welche für den Text dieses Charakters zutreffen festgelegt. „Who“ Aussagen beziehen sich auf das Aussehen des Namens, während „What“ Aussagen das Aussehen des gesprochenen Textes verändern.

Um mehr Persönlichkeit zum Ausdruck zu bringen, können diese in Ren’ Py für jeden Charakter individuell angepasst werden. So können Textgrösse, Schriftart, Farbe, Transparenz und viele weitere Parameter von sowohl dem Namen wie auch dem gesprochenem Text auf jeden Charakter persönlich abgestimmt werden. Da Ren’ Py in der „GUI“-Datei bereits einige Grundeinstellungen besitzt, welche bei nicht definierten Attributen eines Charakters verwendet werden, war zum Beispiel das Definieren einer Schriftart für den Namen hier nicht nötig.

A black background with colorful text

Description automatically generated

Abbildung 2: Definierung des Charakters Gabriel

A white rectangular object with black border

Description automatically generated

Abbildung 3: Textausgabe des Charakters Gabriel

Die Abbildung 3 zeigt, wie ein Text durch den nun definierten Charakter „Gabriel“ dargestellt wird.

In Destiny’s Reminder erhielt jeder Charakter eine eigene Farbe und für den gesprochenen Text auch eine eigene Schriftart. Dies soll helfen die Persönlichkeit eines jeweiligen Charakters auch auf visueller Ebene klar darzustellen. Für gelassenere und chaotischere Figuren wie Destiny und Gabriel wurde jeweils eine handschriftlichere Schriftart gewählt. Diese vermitteln bereits auf den ersten Blick eine gewisse Unordentlichkeit und Menschlichkeit. Diese Ausstrahlung steht wiederum in starkem Kontrast zu der der anderen zwei Hauptfiguren. Für die beiden formelleren und strikteren Charaktere Simon und Matthew wurde eine typografische Schriftart gewählt. Sie verkörpern deren genauere und professionellere Persönlichkeiten. Die Wahl unterschiedlicher Schriftarten dient neben ihrer stilistischen Anwendung jedoch gleichzeitig auch dazu, einen Charakter schneller und einfacher erkennen zu können.

Alle verwendeten Schriftarten wurden von „1001 Fonts“12 und „Dafont“13 heruntergeladen. Mit der Auswahl von „Minako-Regular“14, „Hey Comic15, „DraftingMono-Bold“16, „Coolvetica“17, „KGRedHands“18, „gabriele-bad“19 und „texgyrecursor-bold“20 konnte ein breites Spektrum an handgeschriebenen und typografischen Schriftarten abgedeckt werden.

### **3.3.2 Schreiben der Szenen mittels Labels**

Um das Navigieren des Codes und das Wechseln zwischen Szenen einfacher zu machen, arbeitet Ren’ Py mit einer eigenen Label-Funktion. Wie in Abbildung 4 zu sehen ist, wird jeder neue Textabschnitt von einem Label-Statement initiiert. Der Name dieses Labels kann frei gewählt werden und hilft danach diesen Teil des Codes wieder aufzurufen. Labels helfen das Skript zu strukturieren, sind jedoch auch nützlich, um von einer Szene zur nächsten zu springen.

Durch den Befehl „Jump“ kann zwischen den Labels beliebig hin und her gesprungen werden. Labels sind grundsätzlich nicht nötig um eine Geschichte mit Hilfe der Ren’ Py Engine zu schreiben, werden jedoch zu einem der wichtigsten Hilfsmittel, sobald interaktive Elemente eingebaut werden. So kann zwischen den verschiedenen Handlungssträngen gewechselt und wieder zu gemeinsamen Szene gesprungen werden, ohne den Text mehrfach im Skript vorhanden haben zu müssen.

A screenshot of a computer

Description automatically generated

Abbildung 4: Anwendung einer Label-Funktion

Innerhalb dieser Labels konnten nun die verschiedenen Szenen wie in einem gewöhnlichen Skript geschrieben werden. Jede neue Zeile enthält den Text der zu einem Zeitpunkt auf dem Bildschirm angezeigt wird und per Klick wird im Spiel dann zur nächsten Zeile gewechselt.

### **3.3.3 Einfügen Interaktiver Elemente**

Ren‘ Py in diesem Projekt wurden zwei verschiedene Methoden angewandt, um interaktive Elemente in die Geschichte einzubauen.

**Menus**

Die Wahl einer Handlung wird durch das Einbauen von Menüs in das Skript ermöglicht. Diese funktionieren grundsätzlich sehr ähnlich wie Labels, jedoch mit mehreren von dem Spieler bzw. der Spielerin zu wählenden Optionen.

Durch das Aufrufen eines Menüs wird im Spiel eine Auswahl von anklickbaren Optionen angezeigt. Diese können in beliebiger Menge und mit zuvor definiertem Text angezeigt werden. Um eine Option eines Menus zu generieren, wird dem Code ein String gegeben, welcher den Text enthält, der für diese Option angezeigt werden soll. In Abbildung 5 wurden die Texte „I jammed the printer“ und „Someone jammed the printer“ gegeben. Wird nun eine dieser Optionen angeklickt, springt das Programm zum Code, welcher unter dieser Option eingerückt steht und springt nach dessen Ausführung, falls keine andere Anweisung gegeben wurde, wieder zum Code ausserhalb des Menus. In Abbildungen 5 und 6 sind Beispiele eines Menu-Befehls und dessen Darstellungsweise zu sehen.

Durch solche Menus können direkt kurze Texte ausgegeben werden oder mit Hilfe von Labels zu ganz neuen Handlungsstränge gesprungen werden. In Destiny’s Reminder wurden so alle der in Abbildung 1 aufgezeichneten interaktiven Stationen in das Spiel eingebaut.

A black background with white text

Description automatically generated

Abbildung 5: Anwendung einer Menu-Funktion

**Spieler-Input**

Eine etwas kompliziertere Weise den Spieler bzw. die Spielerin in das Spielerlebnis mit einzubeziehen, kam in der Form des Benennens einer Katze in einem der Abschnitte. Dies konnte direkt mittels Python-Befehlen getan werden. Um dies zu tun musste erst im code angegeben werden, dass folgende befehle nicht als Ren‘ py- sondern als Python-Code auszuführen waren. Mit einem Input-Statement kann das Spiel nach einem Text fragen und diesen in einer Variable speichern. Wie die Abbildung 7 zeigt, wurde dieser in Destiny’s Reminder in die Variable „catname“ untergebracht. Später im Spiel kann dann zum Beispiel bei der Definition des Namens der Katze auf diesen Text zugegriffen werden.

Der Parameter „length“ begrenzt den Text welcher akzeptiert wird. Für dieses Beispiel wurde die Länge auf 32 Zeichen limitiert. Der strip-Befehl entfernt lediglich Leerzeichen vom Beginn des gegebenen Texts.

Um sicher zu stellen, dass schlussendlich ein Name in der Variable „catname“ gespeichert ist, wird diese mit dem Namen „Kitty“ befüllt, falls vom Spieler bzw. der Spielerin kein Name gegeben wird.

A black screen with orange text

Description automatically generated

Abbildung 7: Anwendung einer Spieler-Input-Funktion

## **3.4 Illustrationsprozess der Spielgrafiken**

Mit dem Plan der Handlung und den Anforderungen der 4 Bildklassen, welche während der Recherche-Phase definiert wurden, konnten die nötigen Illustrationen nun passend zur Geschichte angefertigt werden. Alle Illustrationen wurden auf einem iPad mit der Software Procreate21 erstellt.

### **3.4.1 Gestaltung der Charaktere und diverser Spielelemente**

**Charaktere**

Dank der dialogbasierten Spielweise eines Visual-Novel, sind es die Charaktere, welche die Geschichte zum Leben erwecken. Die Geschichte im Mittelpunkt dieser Arbeit, handelt von fünf sehr unterschiedlichen Personen. Um diese fünf Charaktere zum Leben zu erwecken, wurden, neben den unterschiedlichen Sprechweisen, auch für jeden eigene Illustrationen angefertigt. Um die Emotionen der Figuren besser darzustellen und ein abwechslungsreicheres Spielerlebnis zu bieten, werden in Visual-Novels oft mehrere verschiedene Illustrationen oder Animationen eines Charakters erstellt. Je nach Gefühl, welches gerade in einer bestimmten Aussage hervortritt, kann dies dann durch eine passende Grafik unterstrichen werden. Welche Emotion schlussendlich öfter gezeigt wird, kann von Figur zu Figur variieren und somit helfen deren Persönlichkeit zu verstärken. Mit 4-5 Illustrationen pro Charakter konnten die wichtigsten Gefühlslagen und Gesichtsausdrücke abgedeckt werden.

Jede der fünf Figuren erhielt eine zentrale Farbe. Diese Farbe wurde sowohl für die Illustrationen wie auch bereits in Abschnitt 3.3.1 erwähnt für die Textformatierung verwendet. Die Abbildungen 8, 9 und 10 zeigen Beispiele für die verschiedenen Gefühlslagen, welche von einer Figur ausgedrückt werden können.

A cartoon of a person wearing glasses

Description automatically generated A person wearing glasses and a green sweater

Description automatically generated A cartoon of a person wearing glasses

Description automatically generated

Abbildung 8: Gabriel ist besorgt Abbildung 9: Gabriel ist verwirrt Abbildung 10: Gabriel ist glücklich

**Weitere Spielelemente**

Gleich wie die Charaktere mussten auch die Illustrationen für jegliche andere Objekte wie zum Beispiel ein Handy oder ein Computerbildschirm einzeln angefertigt und eingefügt werden. Der Malprozess blieb derselbe wie bei den Charakteren, mit dem einzigen Unterschied, dass diese Objekte meist simpler und somit einfacher zu gestalten waren. Hier war es jedoch wichtig die verschiedenen Illustrationen auf die Hintergründe abzustimmen, auf welchen sie dargestellt werden würden. Als Folge wurden Grafiken wie beispielsweise die Errungenschaftsbildschirme, welche auf allen Hintergründen zu sehen sein würden, in einer sehr neutralen Palette, meist sogar in Graustufen gehalten. Andere Grafiken wie zum Beispiel der Computer, konnten hingegen gemäss der Palette des jeweiligen Hintergrunds gefärbt werden.

### **3.4.2 Gestaltung der Hintergründe**

Die Gestaltung der Hintergründe erfolgte in zwei Schritten. Da für die Geschichte relativ viele Hintergründe benötigt wurden, wurden für alle Illustrationen Bilder als Grundlage verwendet, welche entweder aus dem Internet heruntergeladen oder selbst gemacht wurden. Über diese Bilder konnten dann die verschiedenen Hintergrundillustrationen gemalt werden. Diese Art der Gestaltung vereinfachte das Zeichnen glaubhafter und räumlich wirkenden Szenen und minimierte den Zeitaufwand, da von bereits existierenden Strukturen ausgegangen werden konnte und nicht alle Räume von Hand gemalt werden mussten.

Für allgemeinere Orte, welche nur eine Illustration pro Raum beanspruchten, wie zum Beispiel Destiny’s Büro oder die Strasse, wurden Bilder aus dem Internet als Vorlage verwendet. Für Szenen aus dem Büro wurden Bilder von «Dreamstime»22 und «flickr»23 bezogen. Einer der Strassen-Hintergründe basiert auf eninem 3D Modell von Marteen Hof24. Um jedoch individuellere Orte wie beispielsweise die Cafeteria oder Destiny’s Wohnung von unterschiedlichen Blickwinkeln zeigen zu können, diente Sims425 als nützliches Hilfsmittel um die Räume zu bauen und die verschiedenen Ausschnitte zu fotografieren. In Abbildung 11 ist das Modell Destiny’s Wohnung wie auch der fertige Hintergrund zu sehen.

A cartoon of a kitchen

Description automatically generated

Abbildung 11: Illustration von Destinys Wohnung mit Sims Model

Die Konturen der Aufnahmen wurden nachgezogen und allenfalls mit weiteren Objekten oder Änderungen ergänzt. Um ein einheitliches Aussehen für alle Hintergründe zu haben, wurde für das Färben der Bilder das Werkzeug „Verlaufsumsetzung“ verwendet. Dieses wird genutzt, um jedem hell-dunkel Wert neu auch eine bestimmte Farbe zu verleihen. Dies ermöglichte einen schnelleren Malprozess, da für die eher aufwändigeren Hintergründe nun nur noch auf die Helligkeit eines bestimmten Objekts geachtet werden musste. Zu den Illustrationen in Destiny’s Reminder wurden für die verschiedenen Orte eigene Verläufe erstellt und angewendet. In Abbildung 12 sind die einzelnen Verlaufsumsetzungen mit ihren zugehörigen Illustrationen zu sehen.

A screenshot of a video game

Description automatically generated

Abbildung 12: Verlaufsumsetzungen der Hintergründe

Um mögliche Unstimmigkeiten zwischen Hintergrund und Charakter zu vermeiden, wurden alle Szenen in einem sehr weichen Stil gestaltet. Es wurden hauptsächlich blasse Farben und schwache Kontraste verwendet. Die Farben einer jeweiligen Umgebung wurden gewählt, um eine bestimmte Stimmung zu erzielen. Für Destiny’s Wohnung hiess das ein ruhiges Grün, für das Büro ein kaltes Blau und für die restlichen Orte ein blasses Rot. Die Hintergründe für die Szenen, welche nur mit der Entführung zu tun haben, wurden schwarz-weiss gelassen, um eine bedrückendere und leblosere Atmosphäre zu kreieren.

### A cartoon of a person with a black background Description automatically generated**3.4.3 Gestaltung der Benutzeroberfläche und Icons**

Für die Benutzeroberfläche des Spiels und den Desktop-Icons mussten natürlich auch noch eigene Grafiken angefertigt werden. Da der Fokus der Arbeit auf dem Spiel und seiner Geschichte lag, wurde für die Benutzeroberfläche überwiegend von den bereits von Ren‘ Py gegebenen Formen und Elementen ausgegangen. Mit wenigen Änderungen konnte bereits ein sehr individuelles Design erzielt werden. Die bereits vorhandenen Grafiken wurden kopiert und in der App Procreate angepasst oder abgeändert und später mit der passenden Version ersetzt. Das Layout wurde ebenfalls mehrheitlich übernommen, mit leichten Anpassungen der Farben und dem Einfügen eigener Hintergründe. Die Schriftart des Menü-Texts wurde ebenfalls geändert. Das Wechseln der Grund Schriftart von DejaVuSans26, der von Ren‘ Py als Standard genutzten Schriftart, zu CupCakes27 für in-game und Courier Prime Code28 für out-of-game Text brachten ebenfalls mehr Charakter in das Aussehen des Menus.

Abbildung (!)

Zusammen mit einer Handvoll von Grund auf neuen Illustrationen für das Game-Icon und Startbildschirm wurde so eine stimmige, zum Spiel passende Benutzeroberfläche generiert.

## **3.5 Verknüpfen von Skript und Grafik**

### **3.5.1 Einfügen eigener Grafiken und Schriftarten**

Wie bereits zu Beginn in Abschnitt 2.3 erwähnt, erstellt Ren’ Py für jedes neue Projekt einen eigenen Ordner. Um eigene Illustrationen oder Fotos im Spiel verwenden zu können müssen diese lediglich sinnvoll benannt und im dafür vorgesehenen Ordner abgelegt werden. Für spielinterne Grafiken erstellt Ren’ py einen eigenen «Images»-Ordner. Dank der bereits vordefinierten Befehle von Ren’ Py können die Illustrationen der Charakter und Objekte also ganz einfach in diesen Ordner gelegt und später im Skript aufgerufen werden.

Die Grafiken der Benutzeroberfläche werden in einem separaten «GUI»-Ordner abgelegt. Da für Destiny’s Reminder wie zuvor in Abschnitt 3.4.3 erwähnt die bereits gegebene Benutzeroberfläche lediglich angepasst wurde, mussten die dort vorhandenen Grafiken mit den selbst angefertigten oder bearbeiteten ersetzt werden. Hierbei war es wichtig ihren Namen nicht zu ändern, um sicher zu stellen, dass der Code später auch korrekt funktionieren würde.

Jegliche Schriftarten welche vom Spiel dargestellt werden sollen müssen ähnlich wie die Grafiken im Spielordner als «ttf» oder «otf» abgelegt werden.

### **3.5.2 Darstellen der Charakter und weiterer Spielelemente**

Sobald die verschiedenen Illustrationen der Charaktere im korrekten Ordner hinterlegt waren, wurde das Aufrufen dieser sehr simpel. Wie bereits in Abschnitt 2.4 erwähnt, können einfache Bildelemente durch einen show-Befehl hervorgerufen und durch einen hide-Befehl wieder versteckt werden. Um nun zum richtigen Zeitpunkt die korrekte Emotion oder den korrekten Gegenstand darzustellen, musste erneut durch das Skript gegangen werden und zu jeder Aussage oder zu jedem Geschehnis die passenden Grafiken aufgerufen werden.

Ren’ Py gibt einem die Möglichkeit durch diesen show-Befehl viele Aspekte der Darstellungsweise anzupassen.

Um eine dynamische und glaubhafte Darstellung der Emotionen der Figuren darstellen zu können wurden diese zum Teil sehr oft geändert. Die Abbildung 13 zeigt eine solche Dialogstelle, in welcher das Wechseln der Emotionen gut zu sehen ist.

A screenshot of a computer screen

Description automatically generated

Abbildung 13: Wechseln zwischen Charakter-Grafiken

### **3.5.3 Übergänge**

Der letzte Feinschliff nach dem Einfügen aller Objekte und hintergründe war das verbinden der Szenen mittels passenden Übergängen. Um einen angenehmen und fliessenden Übergang zwischen Szenen

# **4. Schluss**

## **4.1 Beantwortung der Fragestellung**

### **Wie schreibt man eine Interaktive Geschichte?**

Eine interaktive Geschichte unterscheidet sich von einer linearen Geschichte nur in wenigen Elementen. Im Grunde sind interaktive Geschichten lediglich eine Sammlung normaler linearer Geschichten, welche individuell kombiniert und dann nacheinander gelesen werden. Um eine interaktive Geschichte zu schreiben, muss man deshalb zu Beginn sehr ähnlich vorgehen wie bei einer normalen Geschichte. Man plant eine zentrale Handlung und baut danach auf diesen die verschiedenen Pfade auf. Die simpelste Variante eine solche Geschichte zu schreiben, beginnt mit einem der Pfade bis hin zum ersten Ende. Von diesem spaltet man dann die weiteren verzweigenden Handlungen und Enden ab. Falls wie bei dieser Geschichte eines der Enden bevorzugt werden soll, eignet es sich von diesem Pfad auszugehen. Um für alle Pfade eine sinnreiche und spannende Geschichte zu schreiben, erfordern interaktive Geschichten natürlich etwas mehr Planung als herkömmliche, doch schlussendlich beruhen beide auf denselben literarischen Grundlagen und nutzen dieselben rhetorischen und sprachlichen Mittel.

### **Welche Methoden gibt es, um eine Geschichte in spielbarer Form umzusetzen?**

Um eine Geschichte in eine spielbare Form umzusetzen, gibt es viele Methoden. Nebst der hier gewählten Form eines Videospiels, gibt es auch Bücher und TV-Serien welche Geschichten durch interaktive Elemente spielerisch umsetzen. Doch auch unter Videospielen gibt es eine Vielzahl an unterschiedlichen Wegen eine Geschichte zu erzählen. Zusätzlich zu Visual-Novels, deren Gameplay fast ausschließlich aus Dialog bestehet, gibt es auch Rollenspiele, RPGs genannt, welche den Spieler bzw. der Spielerin direkt in die Geschichte eintauchen. Auch Spiele wie „Dungeons & Dragons“29, welche das Konzept eines Fantasy RPGs bereits in Brettspielform umsetzte, sind sehr beliebt. Alle diese Medien nutzen ihre individuellen Stärken und Möglichkeiten, um ansprechende und fesselnde interaktive Geschichten zu erzählen. Dank den vielen verschiedenen Methoden, welche verwendet werden können eine Geschichte spielerisch darzustellen gibt es nicht den einen „richtigen“ Weg. Letzen Endes bestimmen die Präferenzen der Zielgruppe oder des Herstellers welche Form am geeignetsten ist.

### **Was braucht es, um einen Visual-Novel zu kreieren?**

Das Herz jedes guten Visual-Novels ist eine spannende Geschichte. Sie bildet die Grundlage des ganzen Spiels und leitet den Spieler bzw. die Spielerin durch das ganze Erlebnis. Um einen simplen Visual-Novel zu erstellen, braucht es grundsätzlich nur ein Skript, welches gesprochen wird, einfache Grafiken und ein Programm, welches diese Dinge verknüpfen und darstellen kann. Für diese Arbeit wurde deshalb die Engine Ren‘ Py gewählt da sie genau für diesen Zweck erstellt wurde und bereits viele der nötigen Elemente eines Visual-Novels liefert. In den von größeren Studios gemachten Spielen sieht man meist eine hohe Zahl an Orten und Charakteren, welche die Geschichte auskleiden. Dinge wie wechselnde Hintergründe, verschiedene Gesichtsausdrücke oder Übergänge sind jedoch kein muss. Sie können als Stilmittel eingesetzt werden sind und helfen ein ansprechendes Spielerlebnis zu bieten, sind jedoch nicht in allen Fällen nötig. Auch simple Grafiken können eine gute Geschichte erfolgreich ergänzen. Auch Interaktivität ist keine Bedingung, um einen guten Visual-Novel zu kreieren. Es gibt viele linearer Geschichten, welche durch dieses Medium dargestellt wurden. Entscheidet man sich jedoch für eine Version mit verschiedenen Pfaden ist es wichtig diese so zu gestalten, dass die Geschichte für alle Pfade Sinn ergibt.

### **Wie verstärken visuelle Elemente den Eindruck einer Geschichte?**

Um diese Frage beantworten zu können war es wichtig den Unterschied zwischen einem Drama, beziehungsweise einem Dialog basiertem Text wie dem eines Visual-Novels, und anderen literarischen Texten zu verstehen. Texte der Epik oder Lyrik sind geschrieben, im Wissen, das der Text das vollendete Werk darstellt. Sie beinhalten malerische Beschreibungen der Figuren, Orte und Atmosphären um dem Leser bzw. der Leserin die Möglichkeit zu geben sich die beschriebene Szene selbst vorzustellen und die Emotionen, welche präsent sind aufzufassen. Texte wie diese lassen mehr Raum für Interpretation und können teils auch gänzlich ohne direkte Rede erfolgreich Situationen darstellen, in welchen eine Konversation stattfindet. Hier findet sich auch einen der grössten Unterschiede zu einem Skript. Ein Skript oder ein Drama kann zwar als reiner Text gelesen und verstanden werden, wird jedoch grundsätzlich geschrieben, um vorgeführt oder aufgenommen zu werden.

Die Bilder und Grafiken werden eingesetzt, um dem gesprochenen Text mehr Kontext zu verleihen und die Emotionen der Figuren darzustellen. Wie der von einer bestimmten Figur gesprochene Text zu interpretieren ist kann durch die dargestellte Grafik beeinflusst werden. Aussagen können emotional geladen werden oder gar eine komplett andere Bedeutung erhalten je nach Bild welches gleichzeitig angezeigt wird. Auch die Hintergründe verleihen den Szenen weiteren Kontext ohne zusätzliche Texte verwenden zu müssen. Ihre Farben bestimmen die Stimmungen der Szenen und setzen die Charaktere in ein Umfeld mit welchem sie interagieren können.

Was in einem Drama oder Film gesprochen wird, wird in einem Visual-Novel hingegen gelesen. Es kann sich nicht auf den Unterton oder Art und Weise wie ein Satz gesagt wird verlassen werden um die Nuancen der Aussage zu vermitteln. Hier können jedoch Formatierungsoptionen des Text angewandt werden um zum Beispiel mit fettgedrucktem Text eine laute oder mit schiefgedrucktem Text eine leise Stimmlage zu vermitteln.

### **Wie kann ich meinen Code strukturiert notieren und verwalten?**

Die genutzte Programmier-Engine Ren‘ Py gibt seinen Nutzern bzw. Nutzerinnen bereits eine grobe Struktur vor. Der Code ist auf vier Dateien verteilt. Von welchen drei regulieren und definieren wie das Spiel und dessen Benutzeroberfläche angezeigt und abgespielt werden. Diese drei Dateien können trotz der vorgefertigten Struktur beliebig geändert werden, um das Spiel dem Inhalt anzupassen. Dies weist jedoch gewisse Limitierungen auf, da diese auch wichtigen Code welcher nötig ist, um das Spiel starten zu können, beinhaltet. Wird dieser verändert kann es dazu führen dass das Spiel nicht mehr einwandfrei läuft. Ren‘ Py beschreibt deshalb bereits welche Aspekte des Codes zu bearbeiten sind und welche nicht geändert werden dürfen. Da sich die Struktur, des in diesen Dateien vorhandenen Codes, durch das Bearbeiten nicht wesentlich verändert, musste hier keine weitere Strukturierung vorgenommen werden.

Die letzte der vier Dateien beinhaltet das Skript und ist vom Nutzer bzw. der Nutzerin frei zu bearbeiten.

Bei normalem Programmieren ist das Ziel den Code kurz und effizient aber leserlich zu halten. Dies sorgt dafür, dass das geschriebene Programm übersichtlich bleibt und gleichzeitig wenig Speicherplatz beansprucht. Da mit Ren‘ Py jedoch ein Skript geschrieben wird, sieht die Datenstruktur dieser Dokumente etwas anders aus. Um einen neuen Textabschnitt zu beginnen muss dieser auf eine neue Zeile gelegt werden. Da das ganze Skript in einer Datei geschrieben wird und jede neue Aussage eine neue Zeile benötigt, wird das Dokument schlussendlich sehr lang. Die reine grösse des Dokuments kann schnell dazu führen, dass der Code unübersichtlich wird. Wie bei herkömmlichem Programmieren, müssen in Ren‘ Py jegliche Befehle und Variablen, somit auch die Charaktere, vorab definiert werden. Um einen guten Überblick über alle spielrelevanten Variablen und Charaktere zu haben, wurden diese deshalb gerade zu Beginn definiert. So war es einfach die nötige Codestelle zu finden, falls Anpassungen vorgenommen werden mussten.

Das Programm arbeitet, wie bereits in Abschnitt 3.3.2 erwähnt mit einer Label-Funktion. Dies bedeutet, dass die verschiedenen Textstellen theoretisch beliebig angeordnet werden können. Wie die Szenen im Dokument sortiert sind, bleibt also frei zu wählen. Wird zu einem Label gesprungen, dann wird bis ein weiterer Jump-Befehl angetroffen wird, der Code von oben nach unten ausgespielt. Das bedeutet jedoch auch, dass ebenfalls durch korrektes Anordnen von einem Label zum nächsten gewechselt werden kann.

Da diese Geschichte einen klaren, zeitlichen Verlauf besitzt machte es Sinn, die Szenen in chronologischer Reihenfolge anzuordnen. Die verschiedenen Versionen von Szenen, je nach Wahl des Spielers bzw. der Spielerin, wurden untereinandergestellt um Szenen, welche zum selben Zeitpunkt stattfinden am selben Ort im Dokument zu speichern. Eine Anordnung aller Szenen eines Pfades untereinander war eine weiter Möglichkeit den Code zu strukturieren. Da jedoch Szenen vorhanden sind, welche in mehreren Routen vorkommen, wurde eine chronologische Ordnung vorgezogen. So konnten auch diese Szenen sinnvoll untergebracht und mehrfach vorhandener Text vermieden werden.

Die Label-Funktion war zusätzlich zu ihrem Nutzen für das Programm selbst auch ein wichtiges Hilfsmittel während des Schreibens die richtigen Textstellen zu finden. Ren‘ Py erlaubt es den Code labelweise zusammenzuklappen. Neben coderelevanten Labels, welche für den korrekten Ablauf des Programmes nötig sind, wurden deshalb auch rein strukturelle Labels eingebaut. Diese dienten, um den Code während des Schreibprozess in größere Sektionen einzuteilen um diese temporär einklappen zu können.

# **5. Reflexion**

## **5.1 Inhalt**

…

## **5.2 Methoden**

…

**6. Quellenverzeichnis**

1. Microsoft Corporation (2023), GitHub (Version 3.10.2)
2. Wikipedia: <https://de.wikipedia.org/wiki/GitHub> (Abgerufen am 8.10.23)
3. Packard E., Die Insel der 1000 Gefahren, (Ravensburger Verlag, 2007)
4. David Slade, (Netflix, 2018), “Bandersnatch”, Black Mirror
5. P-Studio/Atlus/Sega (2016). Persona5
6. Team Salvato (2017), Doki Doki Literature Club
7. Emmarie.sites.gettysbvurg :<https://emmarie.sites.gettysburg.edu/visual-novels/> (Abgerufen 9.10.23)
8. Tom "PyTom" Rothamel (2023), Ren’ Py (Version 8.0.3)
9. Renpy: <https://www.renpy.org> (Abgerufen 9.10.23)
10. Learnattack: <https://learnattack.de/schuelerlexikon/deutsch/grundformen-des-dramas> (Abgerufen 9.10.23)
11. Draw.io: <https://app.diagrams.net> (Abgerufen 9.10.23)
12. 1001Fonts: <https://www.1001fonts.com> (Abgerufen 9.10.23)
13. Dafont: <https://www.dafont.com> (Abgerufen 9.10.23)
14. Minako: <https://www.dafont.com/minako.font> (Abgerufen 9.10.23)
15. Hey Comic: <https://www.dafont.com/hey-comic.font> (Abgerufen 9.10.23)
16. Drafring\* Mono: <https://www.1001fonts.com/drafting-mono-font.html> (Abgerufen 9.10.23)
17. Coolvetica: https://www.dafont.com/coolvetica.font (Abgerufen 9.10.23)
18. KGRedHands: <https://www.dafont.com/kg-red-hands.font> (Abgerufen 9.10.23)
19. Gabriele-Bad: <https://www.1001fonts.com/gabriele-font.html> (Abgerufen 9.10.23)
20. TeX Gyre Cursor: <https://www.1001fonts.com/tex-gyre-cursor-font.html> (Abgerufen 9.10.23)
21. Savage Interactive (2023), Procreate (Version 5.3)
22. Dreamstime: <https://www.dreamstime.com> (Abgerufen 9.10.23)
23. Flickr: <https://www.flickr.com/photos/markjsebastian/1264424156/> (Abgerufen 9.10.23)
24. Marteen Hof, Artstation: <https://maartenhof.artstation.com/projects/dOGYNK> (Abgerufen 9.10.23)
25. Maxis (2014), The Sims4 (Version 1.101.290.1230)
26. Deja Vu Sans: <https://www.1001fonts.com/dejavu-sans-font.html> (Abgerufen 9.10.23)
27. CupCakes: <https://www.dafont.com/cupcakes.font> (Abgerufen 9.10.23)
28. Courier Prime: <https://www.1001fonts.com/courier-prime-font.html> (Abgerufen 9.10.23)
29. Gary Gygax/Dave Arneson (1974), Dungeons & Dragons

# 7. Anhang

…