Projekt Texting Spree

Der nachfolgende Beschrieb dient zum besseren Verständnis des Projektes sowie der Definition der Schwerpunkte und der Kernsysteme. Darüber hinaus werden Systeme definiert, die für das Projekt von Vorteil sein könnten allerdings nicht unbedingt notwendig sind, um ein gutes Erlebnis der Software zu garantieren.

## **Ausgangslage**

In der heutigen Zeit gibt es leider immer weniger gute Geschichten. Weder werden diese in guten Filmen gezeigt, noch werden diese in guten Spielen dargestellt. Die Geschichten sind oft langweilig und ohne jede Änderung bzw. Entwicklung in den Charakteren, noch in der Handlung selbst. Darüber hinaus lesen die Menschen nicht mehr gerne, da ihre Interaktivität nicht gefördert wird und sie denken: «Wenn an seiner/ihrer Stelle gewesen wäre, hätte ich es so gemacht!»

Das Projekt mit dem Codenamen «Texting Spree» ist als spielbare Erzählung in Format eines Buches bzw. Textes gedacht. Es soll eine spannende Geschichte erzählen und den Spieler dazu motivieren, einerseits die Geschichte zu lesen, andererseits sich darin zu investieren und eigene Entscheidungen zu treffen, die den Verlauf der Geschichte massgebend beeinflussen.

Aufgrund der Beeinflussung durch den Spieler, wird sich das Geschehen der Geschichte ändern und es werden verschiedene Optionen freigeschalten. Darüber hinaus gibt es Systeme wie ein Inventar, in dem der Spiele verschiedene Items aufbewahren kann und die für den Verlauf des Spiels massgebend sind.

## **Involvierte Personen**

Nachfolgend werden die Personen der in Zukunft als «Three Troll Games» bezeichneten Firma, aufgelistet, die in diesem Projekt mitbeteiligt sind und deren Einfluss bzw. deren Arbeit sich auf den weiteren Verlauf des Projektes auswirkt. Darüber hinaus wird ihre Funktion sowie die damit zusammenhängende Arbeit genauer definiert.

|  |  |  |
| --- | --- | --- |
| **Person** | **Titel** | **Bezeichnung/Arbeitsgebiet** |
| Bärfuss Frederic | Lead Story Designer | Der Lead Story Designer ist dafür verantwortlich, die Geschichte des Projektes zu erstellen. In diesem Sinne ist er auch dafür verantwortlich, dass die Story kohärent und ohne Logiklöcher ist. |
| Dulla Kastriot | Game Director | Der Game Director ist der Projektleiter. Er hat die Verantwortung über das ganze Projekt und trägt zu dessen Erfüllung bestmöglich bei. Bei Entscheidungen ist es der Game Director, der Features bestätigt oder ablehnt. Er hat ebenfalls die Aufsicht über die Geschichte und kann Änderungen nach seinem Gutdünken machen. |
| Dulla Kastriot | Programer | Programmierer sind für die technische Umsetzung des Projektes verantwortlich. Primär ist das die Entwicklung des Codes und aller dazugehörigen Elemente. |
| Dulla Kastriot | Story Designer | Untersteht dem Lead Story Designer und ist ihm bei allen Aufgaben behilflich. |
| von Ballmoos Kevin | Chief Programer | Der Chief Programer ist der führende Programmierer des Projektes. Er ist für die technische Umsetzung des Projektes verantwortlich und berät sich primär mit dem Game Director. |
| von Ballmoos Kevin | Lead Game Designer | Der Lead Game Designer beschäftigt sich mit dem Design der Spielelemente. Er ist dafür verantwortlich das Dinge wie Usability, Benutzerfreundlichkeit und Design des Projekt stimmen. |

Tabelle : Jobbeschreibungen

Im Verlauf des Projektes können weitere Stellen hinzukommen, die besetzt werden müssen. Daher besteht die Option für die oben genannte Tabelle sich zu erweitern.

## **Entwicklungsumgebung und Engine**

Die Entwicklung des Projektes wird ausschliesslich mit der Unity Engine Version (2021.3.5f1 oder höher) gemacht. Die zu entwickelnden Komponenten werden in einem ersten Schritt vom Unity Asset Store heruntergeladen und in einem späteren Schritt durch eigens kreierte Assets ersetzt.

Der Code und die damit zusammenhängende Entwicklung der C# Skripte wird mit der Entwicklungsumgebung (IDE) Rider Version 2022.2.2 (oder höher) durchgeführt. Auf Wunsch des jeweiligen Programmierers oder auf Anweisung des Chefentwicklers kann eine andere IDE verwendet werden.

## **Kernsysteme**

Nachfolgend werden die Kernsysteme, die als Must-Have-Funktionen definiert sind, aufgelistet. Darüber hinaus folgt eine Erklärung, was genau diese Systeme im fertigen Projekt bzw. im ersten Prototyp beinhalten müssen.

|  |  |
| --- | --- |
| **Namensgebung** | **Funktionsbeschreibung** |
| Dialogsystem | Es können verschiedene Dialoge in das Spiel erstellt werden. Der Spieler kann diverse Entscheidungen in diesen Dialogen treffen, die den weiteren Spielverlauf, je nach Entscheidung, beeinflussen. Der Spieler kann ausserdem an einer Stelle des Spiels nach hinten Springen, um die vorangegangenen Dialogsequenzen und Entscheidungen zu sehen (kann diese aber nicht mehr beeinflussen). |
| Speichersystem | Der Spieler ist in der Lage das Spiel an einem beliebigen Punkt zu Speichern. Das bedeutet, alle Entscheidungen sowie der angegebene Zeitpunkt des Spiels wird festgehalten.  Eventuell eine Autosave Funktion einbauen, bei der das Spiel nach einem Kapitel gespeichert wird. |
| Ladesystem | Der Spieler ist in der Lage einen vorangegangenen Spielstand, den er durch das Speichersystem erstellt hat, zu laden und von dem Zeitpunkt an weiterzuspielen. Das beinhaltet auch sämtliche Entscheidungen. |
| Titelbildschirm / Menübildschirm | Es soll eine Szene mit einem Titelbildschirm erstellt werden. In diesem Bild sollen die gängigen Menüpunkte wie z.B. «Neues Spiel», «Fortfahren», «Laden», «Optionen» und «Beenden» sichtbar sein. |
| Inventarsystem | Ein Inventarsystem, das es dem Spieler erlaubt Gegenstände während der Geschichte aufzusammeln und die er in einem separaten Menü einsehen kann.  Diese Gegenstände beeinflussen die Handlung der Geschichte insofern, als das diverse Optionen angezeigt oder ebenen nicht angezeigt werden, je nach dem ob man einen bestimmten Gegenstand besitzt oder nicht. |
| Checkpoint System / Kapitel | Während des Spiels soll der Spieler mehrere Checkpoints erhalten, die es ihm erlauben, beim Tod oder aber bei einem «Stuck» im Spiel an besagtem Checkpoint das Spiel zu laden.  Diese Checkpoints sollen über ein Kapitel verteilt, eventuell bei wichtigen Entscheidungen, sein sodass bei einer Fehlentscheidung das Spiel geladen werden kann. |
| Pausenmenu | Durch Drücken der «Esc»-Taste soll dem Spieler ein «Pausenmenü» zur Verfügung gestellt werden, damit das Spiel gespeichert, geladen, beendet, oder zurück zum Titelbildschirm geführt werden kann. |
| Kartenmenu | Es müssen mehrere Karten eingepflegt werden, die dem Spieler je nach Kapitel aufzeigen, wo er sich befindet (z.B. in einer Stadt, auf dem Umland der Stadt, in einer Burg etc.).  Diese Karte soll ausserdem über wichtige Informationen verfügen, wie z.B. den Standort des lokalen Schmieds oder eines Händlers. |
| Protagonisten | Der Spieler muss bei Beginn des Spiels einen von mehreren Protagonisten wählen. Je nach Wahl wird ein anderer Dialog getriggert. |
| Musikstücke | Es sollen ein bis drei verschiedene Musikstücke in das Spiel integriert werden. Diese können Eigenkompositionen sein oder aber es können diverse Stücke aus dem Unity Asset Store sein, deren Gebrauch gratis ist. |

Tabelle : Kernfunktionen des Projektes

## **Erweiterte Systeme**

Im folgenden Kapitel definieren wir die Systeme und Funktionen, deren Entwicklung massgeblich zur Erhöhung der Qualität des Projektes beiträgt, allerdings nicht funktional zwingend sind. Diese Funktionen können, müssen aber nicht eingebaut werden. Sie werden daher als Nice-To-Have-Funktionen definiert.

|  |  |
| --- | --- |
| **Namensgebung** | **Funktionsbeschreibung** |
| Kampfsystem | Es soll ein Kampfsystem eingebaut werden, bei dem der Spieler in der Lage ist, einen Kampf gegen verschiedene Gegner zu führen.  Wie das Kampfsystem auszusehen hat, ist noch nicht entschieden. |
| Interaktive Karte | Eine Karte, die es dem Spieler erlaubt sich nach eigenem Gutdünken auf ihr zu bewegen. Die ihm aber auch wichtige Informationen wie z.B. die nächstgelegene Stadt liefert. |
| Handelssystem | Der Spieler kann sich an diverse Händler wenden, um gefundene Gegenstände zu verkaufen und andere Gegenstände zu kaufen. Es wird daher auch eine Währung in das Spiel integriert. |
| Fähigkeiten | Der Spieler hat je nach Protagonisten Wahl eine Anzahl von Fähigkeiten, die es ihm erlaubt Situationen bzw. Entscheidungen einfacher zu treffen oder es ihm gar erlauben Entscheidungen auf die «Bestmögliche» Weise zu lösen. |
| Charaktererstellung | Der Spieler ist in der Lage sich einen eigenen Charakter zu erstellen inkl. Eigenem Namen und Geschlecht. |
| Charakter Attribute | Der Spieler verfügt über diverse Attribute wie z.B. Stärke, Geschick, Ausdauer und Charisma, die es ihm erlauben diverse Entscheidungen einfacher zu treffen.  Dies erlaubt es dem Spieler, basierend auf einer gewissen Chance, die z.B. mit der Zahl des Attributes multipliziert wird (z.B. Change sich an den Wachen vorbeizuschleichen = 5%, Geschicklichkeit = 10, Resultat = 0.5 X 10 = 5. Somit 5% + 5% = Endresultat) Entscheidungen besser zu bewältigen. |
| Uhrzeitsystem | Ein System das die Tages- und Nachtzeit der Welt festhält. |
| Event Director | Eine unsichtbare Entität, die basierend auf diversen Entscheidungen bzw. Systemen wie dem Uhrzeitsystem oder die Interaktive Karte, Events triggered, die den Spielverlauf beeinflussen. |
| Audio | Audio für das Spiel. Darunter ist Musik sowie diverse Audioeffekte zu verstehen. |
| RNG-System | Diverse Entscheidungen können sich negativ auf die Chance auswirken. Zum Beispiel gibt es eine geringe Chance während einer Scene von einem Dach zu fallen und sich zu verletzen. |
| Hunger/Durst System | Der Spieler erhält während des Spiels Hunger bzw. Durst, je nachdem wie lange er ohne diese Dinge verbracht hat. Essenziel hierfür ist das Uhrzeitsystem. |

## **Zielsetzung**

Das Ziel dieses Dokumentes ist eine solide Grundlage für die Dokumentation des Projektes zu bieten sowie den Mitgliedern einen Einblick in die Zukunft bzw. den momentanen Stand der Entwicklung zu liefern.

Des Weiteren ist es das Ziel der Entwickler bzw. «Three Troll Games» einen lauffähigen Prototypen des Projektes inkl. Aller Must-Have-Funktionen bis am 31.12.2023 fertigzustellen. Darin nicht notwendig sind Animationen sowie Assets, Design und Audio.

## **Code und Versionierung.**

Der Code ist neben den Assets das Herzstück des Projekts. Der Code wird grundsätzlich nur von den Programmierern oder dem Chief Programer bearbeitet. Dabei sind diverse Standards in Bezug auf die Syntax, die Kommentare sowie die Erstellung von Methoden und Klassen anzuwenden.

Der Code wird grundsätzlich mittels Github versioniert. Es gibt neben dem Master-Branch noch den Develop-Branch, sowie den Administration-Branch innerhalb der Versionskontrolle. Für alle weiteren Entwicklungen an dem Projekt, müssen sogenannte Feature-Branches erstellt werden. Folgende Regeln gelten bei der Erstellung der Feature-Branches:

1. Der Feature-Branch wird im lowercase Camel Case geschrieben. **Beispiel**: feature\MainMenu
2. Die Feature-Branches werden mittels Backslasch von der eigentlichen Codestruktur getrennt. Das heisst es wird **feature\<Arbeit>** benannt. **Beispiel**: feature\MainMenu.
3. Alle Branches werden entweder vom Chief Programer oder vom Game Director in den Develop-Branch und später in den Master gemerged. Es darf kein eigener Merge stattfinden.
4. Nach erledigter Arbeit muss ein Pull-Request zum Chief Programmer oder zum Game Director gemacht werden. Erst nach Bestätigung einer der Beiden, darf ein neuer Branch mit einem neuen Feature erstellt werden.

Für eine genauere Dokumentation der Code-Standards sowie der Namenskonventionen während des Projektes kann das Dokument Coding Conventions.docx konsultiert werden.

* 1. **XML-Struktur**

Die Geschichte soll über mehrere Kapitel entstehen. Die Kapitel selbst in die Dialogue-Boxen zu schreiben ist unglaublich mühselig und kann schnell zu Unübersichtlichkeit führen. Dies führt unweigerlich dazu, dass man Fehler innerhalb der Geschichte macht und somit nicht in der Lage ist, die gewünschte Qualität der Geschichte sicherzustellen.

Um diesen Problem entgegenzuwirken, haben wir uns ein XML File erstellt, dass in der Lage ist vom Dialogue-Editor eingelesen zu werden und die einzelnen Story-Brocken und Kapitel in die jeweiligen Nodes zu speichern. Um das im Code zu ermöglichen, muss das File allerdings ein spezielles Formt haben, damit der Code in der Lage ist die Geschichte richtig abzufüllen und schlussendlich in die jeweiligen Nodes zu fügen. Es folgt ein Beispiel der XML-Struktur, wie sie erstellt werden muss, damit das File richtig eingelesen werden kann.

