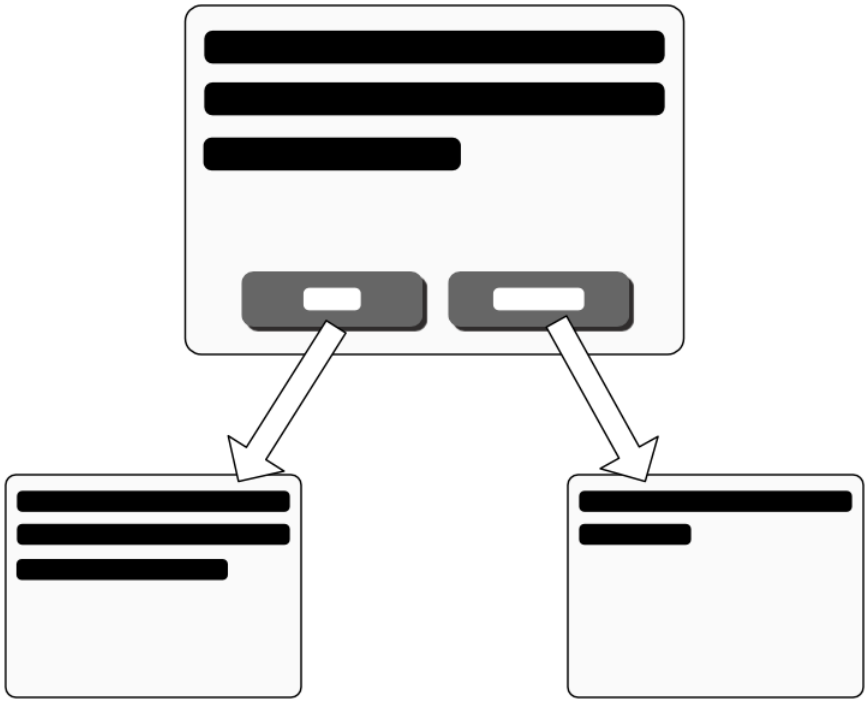
Ökumenisches Gymnasium
zu Bremen
Oberneulander Landstr. 143a
28355 Bremen

Entwicklung eines Autorensystems für interaktive Fiktion

Seminarfacharbeit Informatik, Deutsch



Verfasser: Fabian Keßler
Fachlehrer: Christian Panse, Katja Eggert

Abgegeben am:

1. Einleitung

1.1 Was ist interaktive Fiktion?

Interaktive Fiktion (eher bekannt unter dem englischen Begriff „Interactive Fiction") beschreibt ein textbasiertes Computerspielgenre, bei dem der Spieler durch sein Handeln den Verlauf der Geschichte beeinflussen kann.[1] Die bekanntesten Vertreter dieses Genres sind Textadventures, wie das 1976 veröffentlichte „Adventure". Dies war das erste veröffentlichte Spiel des Genres und baute bereits auf denselben Spielmechaniken auf wie seine Nachfolger: Mit Hilfe von einfachem Text wird eine Geschichte erzählt, bei der der Spieler durch ausgeschriebene Befehle in das Geschehen eingreifen kann. Das Spiel gab durch seine Erkundungsaspekte auch der neu entstandenen Kategorie „Adventure" ihren Namen.[2] Allerdings sind nicht alle interaktiven Fiktionen Adventure-Spiele. Die im Gegensatz zu den rein textbasierten Textadventures auch noch heute beliebten japanischen „Visual-Novels" weisen beispielsweise keine Erkundungselemente auf. Zudem erzählen diese Bildromane ihre Geschichten neben dem Text auch mit gezeichneten Standbildern. Auch die Interaktion mit der Geschichte findet nicht über Texteingaben, sondern durch Mausklicks statt.[3]

Anhand der beiden Subgenrebeispiele Textadventures und Visual-Novels lässt sich gut erkennen, dass „Interaktive Fiktion" ein weiter Oberbegriff ist. Zwischen den Subgenres unterscheiden sich sowohl die Art der Erzählung als auch die Art der Interaktion sehr.

[1] Vgl. Keller, Bernd. „Was bedeutet Interactive Fiction?". interactive-fiction.de.
[2] Vgl. Wikipedia. „Adventure". de.wikipedia.org/wiki/Adventure.
[3] Vgl. Wikipedia. „Visual Novel". de.wikipedia.org/wiki/Visual_Novel.

1.2 Was sind Autorensysteme?

Autorensysteme sind Programme, die zur Entwicklung von interaktiven Inhalten genutzt werden. Dabei liegt der Schwerpunkt oft auf der mühelosen Bedienung des Nutzers ohne weitere Vorkenntnisse wie zum Beispiel im Programmieren.[4] Ein bekanntes Beispiel wäre die Präsentationssoftware PowerPoint von Microsoft. Das Programm bietet viele Möglichkeiten, den Präsentationsinhalt audiovisuell aufzubereiten und Folien interaktiv zu gestalten sowie einen klaren Ablauf festzulegen.[5]

Häufig werden Autorensysteme auch im Bereich „E-Learning" für die Entwicklung von interaktiven Kursen verwendet. Dabei gestalten die Lehrkräfte digitale Lernmodule, welche von den Schülern oder Auszubildenden bearbeitet werden.[6]

Aber auch für interaktive Fiktion werden Autorensysteme entwickelt. Anfangs entwickelten Computerspielfirmen wie Infocom und Telarium die Systeme nur für die interne Nutzung der angestellten Autoren. So konnte sich auf die Spielgestaltung konzentriert werden und die Autoren brauchten keine Vorkenntnisse im Programmieren. Im Jahr 1983 wurde schließlich das erste öffentlich zugängliche Autorensystem „The Quill" veröffentlicht. Es ermöglichte bereits die Entwicklung einer Spielwelt mithilfe einer grafischen Oberfläche.[7]

Heutzutage werden neben Visual-Novels keine kommerziellen textbasierten Spiele mehr entwickelt.[8] Neben vielen eher nutzerunfreundlichen Skriptsprachen und Programmbibliotheken für bereits existierende Programmiersprachen gibt es aktuell nur drei bekannte Autorensysteme: „Twine", „ADRIFT" und „Quest". Alle drei Autorensysteme werden kostenlos angeboten und setzen auf eine visuelle Benutzeroberfläche, die es möglichst einfach gestalten

[4] Vgl. Wikibooks. „Multimedia im Überblick/ Gestaltung/ Inhalte/ Autorensysteme". de.wikibooks.org/wiki/Multimedia_im_%C3%9Cberblick/_Gestaltung/_Inhalte/ _Autorensysteme.
[5] Vgl. Microsoft Office-Support. „Was ist Powerpoint?". support.microsoft.com/de-de/office/was-ist-powerpoint-5f9cc860-d199-4d85-ad1b-4b74018acf5b.
[6] Vgl. BILD. „Autorensysteme – für jeden Bedarf das Richtige?". bild.de/infos/e-learning/e-learning/autorensysteme-10752772.bild.html.
[7] Vgl. Wikipedia. „Interactive Fiction". de.wikipedia.org/wiki/Interactive_Fiction.
[8] Vgl. Keller, Bernd. „Werden noch neue Spiele entwickelt?". interactive-fiction.de.

soll, interaktive Geschichten zu entwickeln. Dabei sind allerdings die Anpassungsmöglichkeiten eher gering. Zwar scheinen alle drei Systeme Funktionen wie Bildanzeige und Tonwiedergabe anzubieten, jedoch sind die Art der Bildpräsentation, Animationen oder Interaktionsweise nicht konfigurierbar.[9]

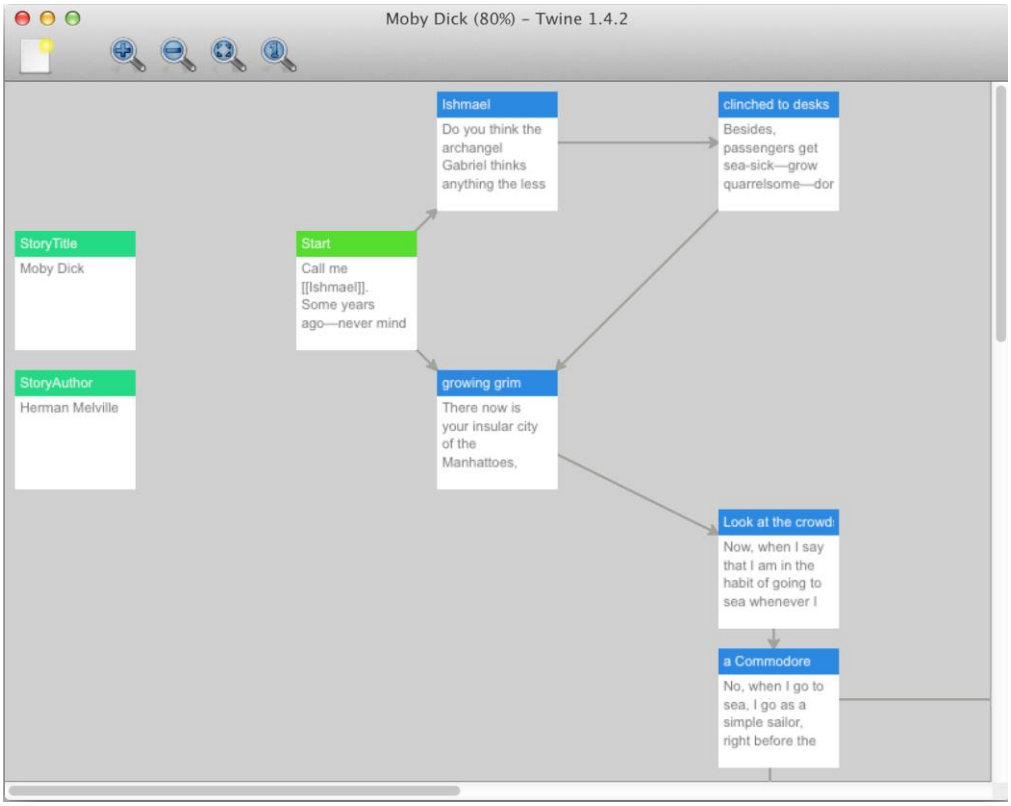


Abbildung 1: „Editing a Story in Twine 1.4" - twinery.org

1.3 Das Ziel der Arbeit

Die im vorigen Kapitel genannten Autorensysteme „Twine", „ADRIFT" und „Quest" bieten zwar alle gute Möglichkeiten der Gliederung von interaktiven Geschichten, weisen aber große Mängel in der gestalterischen Freiheit von Darstellung und Interaktion auf.[9]

Ziel der Arbeit ist es daher, ein eigenes Autorensystem zu entwickeln, das durch eigene Skripte, Erweiterungen („Plugins") und Designs („Themes") gestalterische Freiheit in allen wesentlichen Aspekten einer interaktiven Geschichte ermöglicht. Um die Entwicklung dieser Aspekte möglichst einfach

[9] Vgl. Ephesos Software. „3 Tools zum Erstellen eigener Textabenteuerspiele". ephesossoftware.com/articles/gaming/3-tools-to-create-your-own-text-adventure-games.html.

zu halten, wird versucht, eine Entwicklungsumgebung zu gestalten, welche Werkzeuge für das Entwickeln von Skripten, Designs, Erweiterungen sowie Geschichtenstruktur bereitstellt. Dabei sollen alle Aspekte separat voneinander entwickelt, exportiert sowie importiert werden können, um so die Anpassung für das finale Produkt einer interaktiven Fiktion möglichst frei zu gestalten. So kann beispielsweise von einer Person ausschließlich die Geschichte und ihr Ablauf gestaltet werden und für die Anpassung der visuellen und interaktiven Elemente Designs und Erweiterungen anderer Nutzer verwendet werden.

Mit einer solchen Freiheit wären nicht nur wie bei „Twine", „ADRIFT" und „Quest" die Entwicklung von simplen Textadventures möglich, sondern auch die von jeglicher anderen Art interaktiver Fiktion (wie z. B. Visual-Novels).

2 Wahl der zu nutzenden Technologien

2.1 Tech-Stacks

Zur Planung der Entwicklung von Software ist die Wahl der zu nutzenden Technologien ein essenzieller Bestandteil. Gerade wenn ein Software-Projekt wie das eines Autorensystems viele verschiedene Arten von Prozessen verarbeiten können soll, wird oft ein sogenannter „Tech-Stack" erstellt. Dieser gibt an, welche Technologien (wie z. B. Programmiersprachen, Datenspeicherungstechnologien oder Programmbibliotheken) ein Projekt benötigt.[10]

Um sich für einen Tech-Stack zur Entwicklung des Autorensystems entscheiden zu können, muss zuerst geklärt werden, welche Fähigkeiten das Projekt beansprucht. Dafür teilt man die Ziel-Software vorerst in kleine Softwaremodule auf, die nur für das Lösen bestimmter Problemstellungen zuständig sind.

2.2 Äußere Module

Angefangen habe ich beim Anwender (hier der Autor), der von außen nur mit drei separaten Modulen interagieren wird. Hauptsächlich muss er mit Hilfe eines Programms in der Lage sein, seine Ressourcen wie Texte, Bilder oder Ton organisieren zu können und in ein von einem anderen Softwaremodul lesbares Format zu bringen. Dieses Softwaremodul würde das gesamte Autorensystem beschreiben.

Zudem ist es üblich den Anwendern eines Softwaremoduls eine Anleitung zur Verfügung zu stellen. Diese sogenannte „Dokumentation" hat die Aufgabe, dem Anwender die Nutzung der Software möglichst verständlich und umfangreich zu erklären. Eine solche Dokumentation wäre ebenfalls notwendig, um nicht nur den potentiellen Autoren das Autorensystem näherzubringen, sondern vor allem um Theme- und Plugin-Entwickler über nutzbare Schnittstellen zu informieren, über die sie bestimmte Module ansprechen können.[11]

Das dritte Modul beschreibt die Anwendung, mit der schlussendlich auch die Spieler der interaktiven Fiktionen interagieren. Sie stellt das Computerprogramm oder auch die mobile Anwendung (Handy-Spiel) dar, in der die von den Autoren exportierten und veröffentlichten Geschichten regelrecht „abgespielt" werden. Zusätzlich kann dieses Modul auch von Theme- und Plugin-Entwicklern sowie Autoren während des

[10] Vgl. Teufel, Yvonne. Mixpanel. „Was ist ein Technologie-Stack?". https://mixpanel.com/de/blog/was-ist-ein-technologie-stack/.
[11] Vgl. Augsten, Stephan. Dev-insider. „Was ist eine Softwaredokumentation?". https://www.dev-insider.de/was-ist-eine-softwaredokumentation-a-744924/.

Entwicklungsprozesses verwendet werden, um ihre Kreationen in Aktion zu sehen und mögliche Fehler zu finden („Debugging“). Dieses Modul ist die sogenannte „Game-Engine”, also das Programm, das letztendlich alle von dem Autorensystem bereitgestellten Ressourcen in die Form eines ausführbaren Spiels bringt.

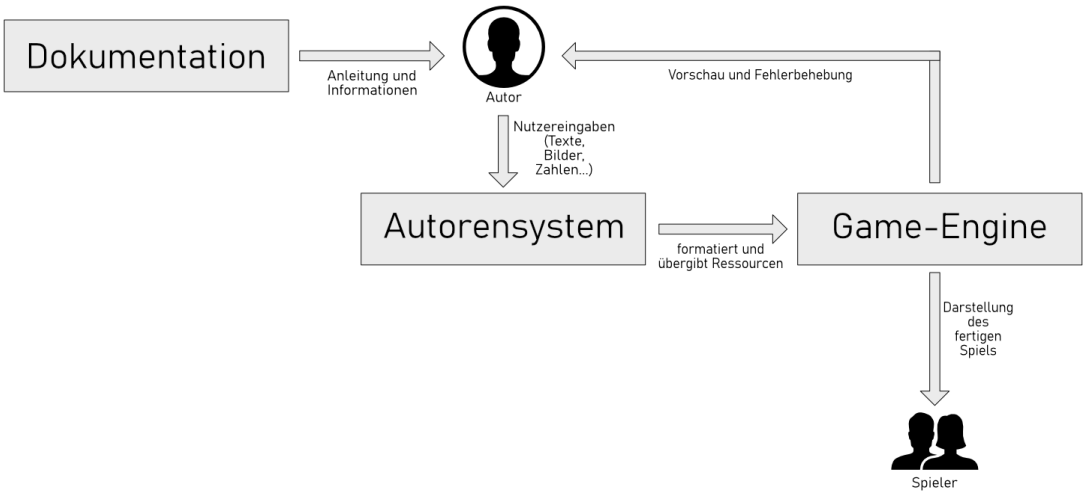


Abbildung 2: Schematische Darstellung der drei äußeren Module

2.3 Innere Module

Die inneren Module beschreiben jene, die Teil der drei äußeren Module sind und somit weder mit dem Autor noch mit den Spielern direkt interagieren.

Das Autorensystem beinhaltet neben der umfassenden grafischen Oberfläche zur Eingabe durch die Autoren beispielsweise ein Modul zur Umwandlung der Projektdaten in ein von der Game-Engine lesbares Format.

Die Game-Engine bietet ein Modul für die grafische Benutzeroberfläche zur Auswahl der zu spielenden Werke und eines in Form einer sogenannten „API” (Application Programming Interface). Dies ist eine „Programmierschnittstelle”, durch die Plugins und Themes mit der Game-Engine über Programmcode kommunizieren können.[12] Der wesentliche Bestandteil der Engine ist allerdings das Interpretations-Modul. Jede auszuführende Projektdatei wird zuerst ausgelesen und verarbeitet, bevor sie als Spiel dargestellt werden kann. Dabei werden die von dem Autorensystem formatierten Daten strukturiert abgerufen und Stück für Stück in interaktive Elemente umgewandelt.

[12] Vgl. Talend. „Was ist eine API? Einfach erklärt!”. https://www.talend.com/de/resources/was-ist-eine-api/.

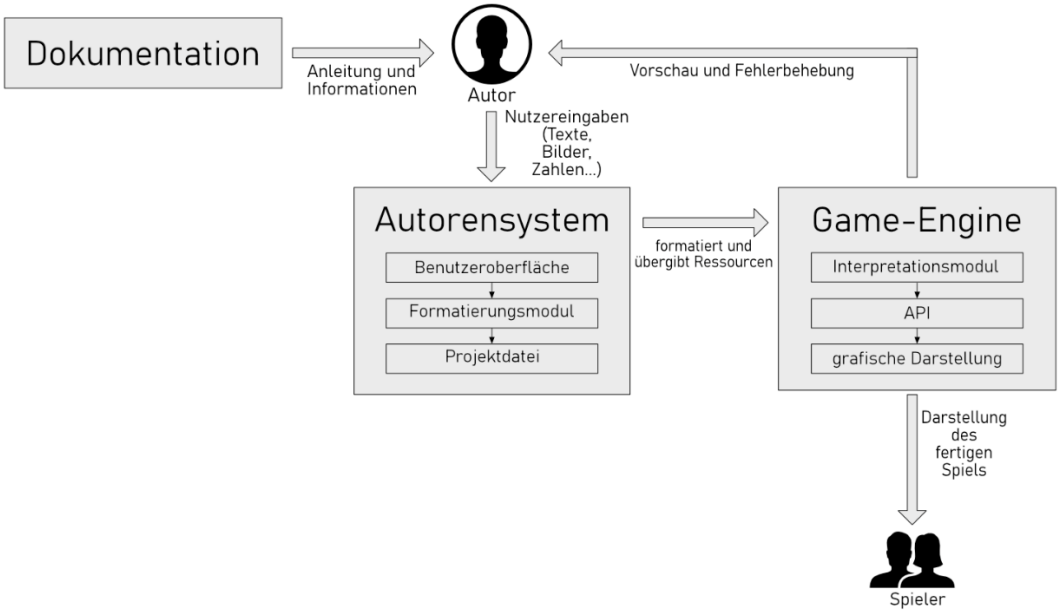


Abbildung 3: Schematische Darstellung der äußeren und inneren Module

2.4 Gewählte Technologien

Unter dem Begriff „Game-Engine" versteht man normalerweise eine Software, welche bei der Entwicklung von Spielen unterstützt, indem sie meist komplizierte und systemnahe Operationen wie das Rendern[13] von Bildern, das Abspielen von Klängen oder das Berechnen der Spiele-Physik[14] vorwegnimmt oder durch gewisse Programmschnittstellen dem Entwickler leicht zugänglich macht.[15] Die in dieser Arbeit entwickelte Game-Engine ist allerdings nur in der Lage, die Projektdateien des Autorensystems zu interpretieren. Die finale Darstellung erfolgt durch eine weitere nicht selbst entwickelte Game-Engine. Diese Game-Engine musste in der Lage sein, verschiedenste Arten von Ressourcen während der Laufzeit zu importieren und im Spiel zu implementieren. Zudem sollte sie auch möglichst entwicklerfreundlich sein, um die Theme- und Plugin Erstellung möglichst angenehm zu gestalten. Letztendlich wurde im Rahmen dieser Arbeit die Open Source[16] Game-Engine „Godot" verwendet. Vor allem die verzweigte Szenenstruktur ist für die Entwicklung von Plugins und Themes von großer Hilfe (siehe Abbildung 4).

[13] Erzeugen eines Bildes aus Rohdaten
[14] Vgl. Wikipedia. „Game physics". https://en.wikipedia.org/wiki/Game_physics.
[15] Vgl. Funes, Antonio. PCGames. „Spiele-Engines: Was ist eine Engine?". https://www.pcgames.de/Unreal-Engine-Software-239301/Specials/spiele-games-grafik-3d-historie-allgemein-epic-vorschau-1350751/2/.
[16] Software mit frei zugänglichem Quelltext

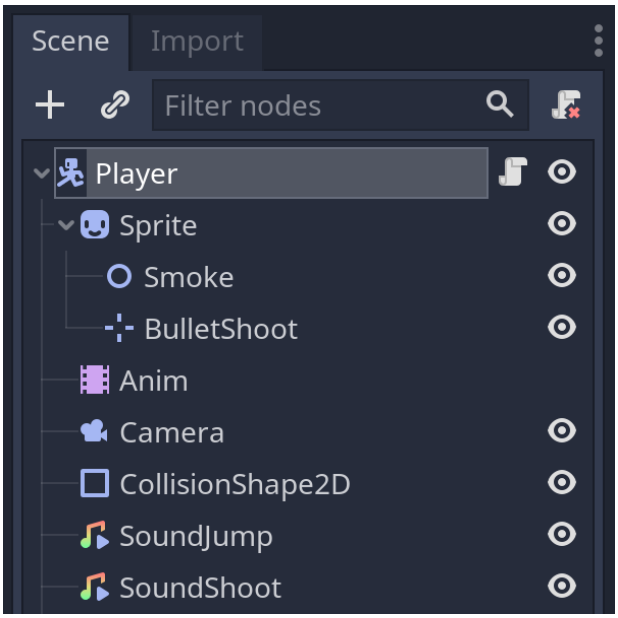


Abbildung 4: „Big or small ideas adapt seamlessly to Godot's node-based architecture, making your life easier." - godotengine.org

Für die Technologie des Autorensystems mussten ganz andere Kriterien erfüllt sein. Sie sollte das Erstellen der umfangreichen Benutzeroberfläche möglichst einfach gestalten und die Möglichkeit bieten relativ große Datenmengen effizient zu verarbeiten. Zudem war es wichtig, in der Lage zu sein, umfangreiche Operationen im Dateisystem zur Erstellung der Projektdateien vorzunehmen. Am vertrautesten für mich war dafür die Programmiersprache „C#" in Kombination mit „Windows Forms". Auch wenn letzteres den Nachteil hatte, nicht auf anderen Betriebssystemen als Windows ausgeführt werden zu können, bot es mir einen schnellen und unkomplizierten Weg die Benutzeroberfläche des Autorensystems zu entwerfen.[17]

Auch für die Dokumentation entschied ich mich für zwei sehr hilfreiche Technologien. Um die Dokumentation möglichst plattformunabhängig und teilweise automatisiert anbieten zu können, entschied ich mich für die Verwendung von „Vuepress", das mit Fokus auf Softwaredokumentationen entwickelt wurde. Es ermöglicht das Schreiben von Websites mit der einfach anzuwendenden Auszeichnungssprache[18] „Markdown".[19] Markdown ist beim Schreiben einer Dokumentation besonders hilfreich, da ein in Markdown geschriebener Text auch in seiner Quelltextfassung noch gut lesbar, und die Formatierung ebenfalls erkennbar ist.

[17] Vgl. Adegeo. Microsoft Docs. „Desktop Guide (Windows Forms .NET)". https://docs.microsoft.com/de-de/dotnet/desktop/winforms/overview/?view=netdesktop-6.0.
[18] maschinenlesbare Sprache für die Gliederung und Formatierung von Texten und anderen Daten
[19] Vgl. Vuepress Guide. „How It Works". https://vuepress.vuejs.org/guide/#how-it-works.

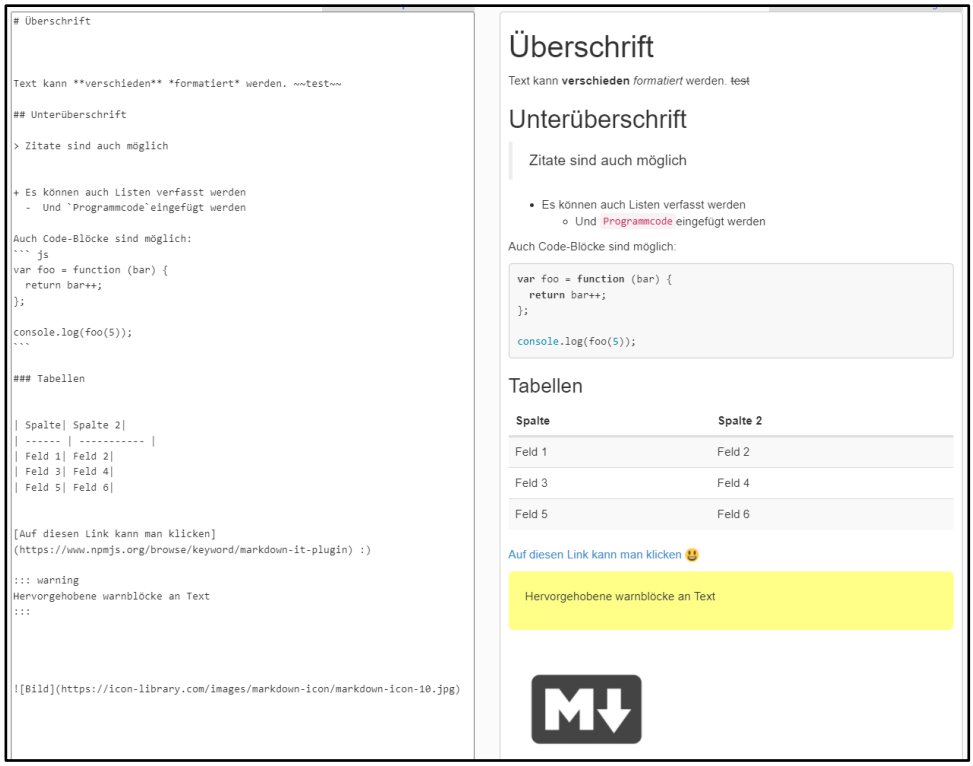


Abbildung 5: Eigens auf der Seite markdown-it.github.io kreiertes Beispiel von Markdown, auf der linken Seite ist der Quelltext zu lesen, auf der rechten das Resultat

Ein weiterer Vorteil von Markdown ist, dass die zweite gewählte Technologie für die Dokumentation, der ebenfalls quelloffene „GDScript-Docs-Maker", seine Resultate nach Markdown exportieren kann.[20] Wie der Name bereits vermuten lässt, kann man mit Hilfe dieses Tools automatisch eine Dokumentation erzeugen lassen. Allerdings besteht dieser Teil der Dokumentation dann nur aus einer sogenannten „Code Reference", einer automatisch mit den Informationen und Kommentaren aus dem Code generierten Übersicht aller Klassen und Methoden.[21] Diese wird besonders für alle Plugin- und Theme-Entwickler von Bedeutung sein. Der zweite Teil, also die ausformulierten Anleitungen für das Autorensystem und die sonstigen Artikel, muss manuell in Markdown verfasst werden.

[20] Vgl. Lovato, Nathan. GitHub. „GDScript Docs Maker". https://github.com/GDQuest/gdscript-docs-maker.
[21] Vgl. Microsoft Docs. „Reference documentation" https://docs.microsoft.com/en-us/style-guide/developer-content/reference-documentation.

3 Entwicklung

3.1 Vorgehen

Angefangen habe ich mit der Gestaltung des Autorensystems, welches ich „MysteryMaker“ nannte. Für jedes der möglichen Datenfelder einer Projektdatei musste eine Eingabemöglichkeit geschaffen werden. So habe ich eine Baumansicht entwickelt, die eine gute Übersicht über die Spielelemente bietet und es ermöglicht, neue Spielelemente per Rechtsklick hinzuzufügen oder zu entfernen (siehe Abb. 6). Auch kann durch das Selektieren von Einträgen der jeweilige Eintrag bearbeitet werden.

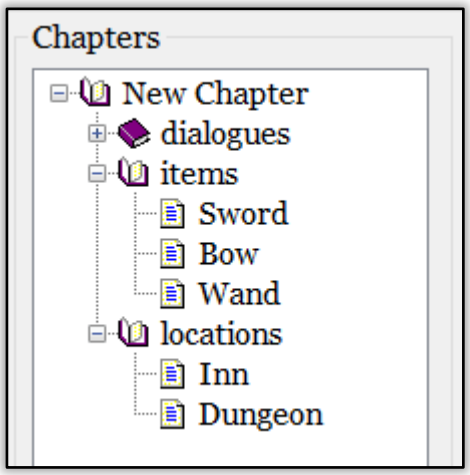


Abbildung 6: Baumförmige Kapitelansicht in MysteryMaker

Für jede Art von Eintrag (Dialog, Gegenstand oder Ort) mussten nun sogenannte „GUI-Panels“[22] gebaut werden. Dies sind Gruppierungen von Interface-Elementen, welche gemeinsam angezeigt und ausgeblendet werden können. Für jede der Eigenschaften eines Spielobjekts wurde ein möglichst passendes Eingabeelement gewählt. So bekommt nun der Anwender des Programms immer eine passende Eingabeoberfläche für das derzeit zu bearbeitende Element.

[22] GUI: Graphical User Interface

Beispiel der Nutzung von „GUI-Panels“:

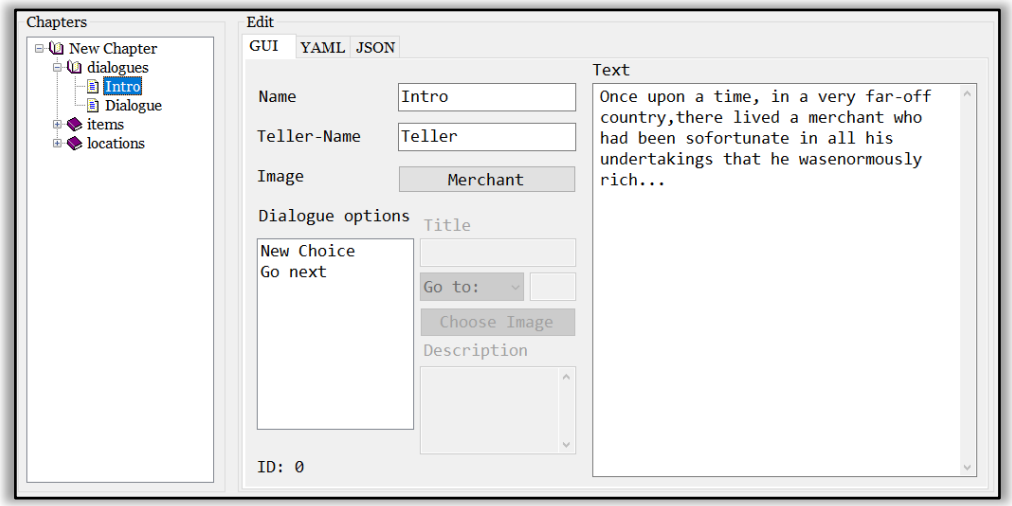


Abbildung 7: Die Auswahl eines Dialog-Elements führt zur Anzeige der zur Eingabe von Dialogen behilflichen Interface-Elementen.

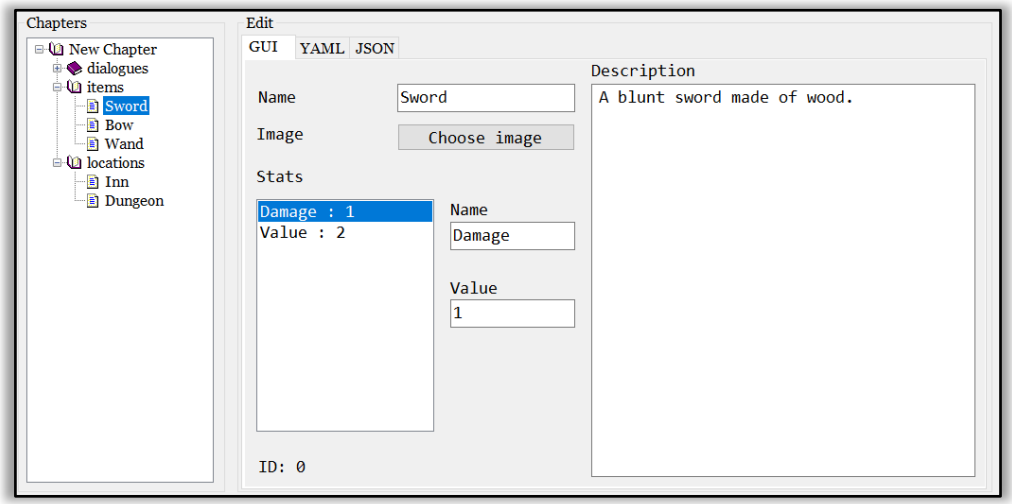


Abbildung 8: Die Auswahl eines Gegenstand-Elements führt zur Anzeige der zur Eingabe von Gegenständen behilflichen Interface-Elementen.

Um dem Nutzer zu ermöglichen, seine Eingaben speichern zu können, implementierte ich anschließend auch ein Projekt-Management-System. Mit dessen Hilfe konnten Projektordner neu erstellt, gespeichert und geladen werden.

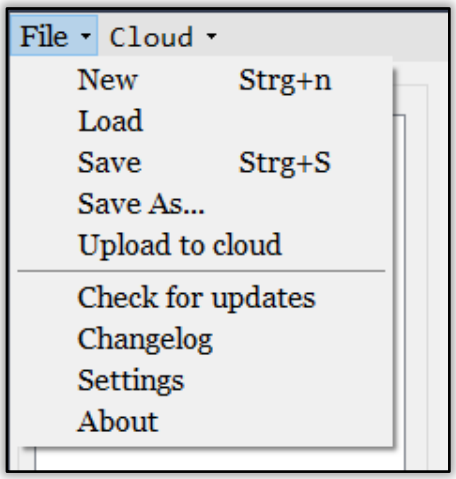


Abbildung 9: Menüleiste von MysteryMaker

Nachdem MysteryMaker-Projekte exportiert werden konnten, begann die Entwicklung der Game-Engine zu deren Interpretation. Diese Engine bekam den Namen „Velius Engine“ und wurde innerhalb der Godot Game-Engine entwickelt. Neben der Interpretation ist ein wichtiger Bestandteil der Velius Engine auch die Verwaltung der importierten Projekte sowie eine spielergerechte Navigation.

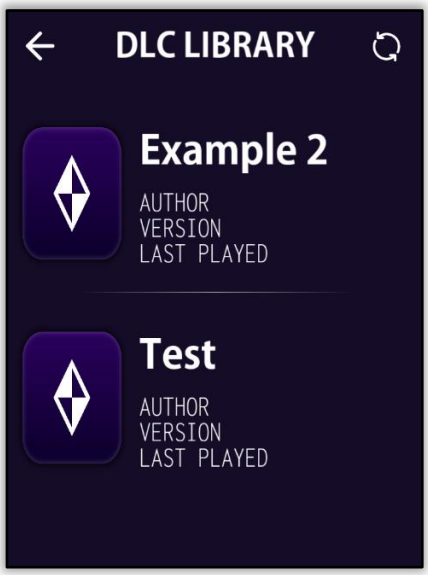


Abbildung 10: Projekt Auswahlmenü der Velius-Engine

Um eine möglichst reibungslose Arbeitsroutine bieten zu können, wurden anschließend auch noch minimalistische „Debugging-Tools“, also Hilfsmittel zur Fehlerbehebung wie einen „Play“-Knopf und eine Ausgabekonsole im MysteryMaker implementiert. So lässt sich nun per Knopfdruck das geöffnete Projekt direkt von der Velius Engine ausführen und Fehler sowie jegliche Werte-Ausgaben über die Konsole einsehen. Das spart viel Zeit und Arbeit sowohl bei der Entwicklung der interaktiven Fiktionen, Themes und Plugins, als auch bei der Weiterentwicklung der Velius Engine und des MysteryMakers selbst.

Da zur Darstellung des fertigen Spiels von der Velius Engine immer ein „Theme“ benötigt wird, musste noch ein solches entwickelt werden. Ich entschied mich für eine eher umfangreiche Darstellung, um so die große Freiheit bei der Entwicklung von Themes zu betonen. Ein eher minimalistisches Theme, das gut als Vorlage für von Nutzern erstellte Themes verwendet werden kann, ist ebenfalls in Planung.

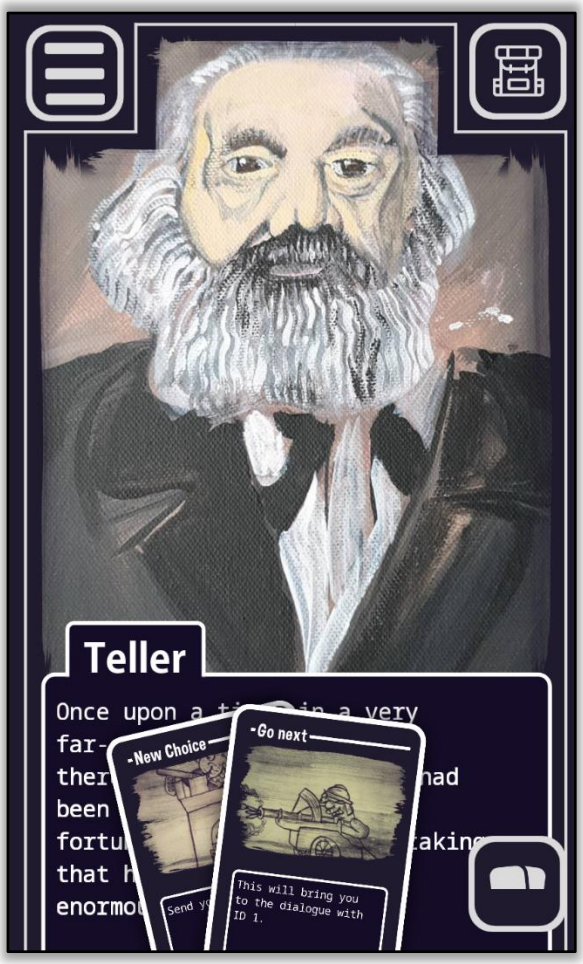


Abbildung 11: Das von mir erstellte „Card-Theme“ in Aktion, Bilder von Milan Ilic

3.2 Probleme

Während der Entwicklung der GUI für den MysteryMaker fiel mir ein fundamentaler Nachteil grafischer Oberflächen auf. Sollten von Dritten erstellte Themes beispielsweise mehr Werte entgegennehmen wollen als vom Programm vorgegeben (wie z. B. ein zweites Bild für einen Dialog), wäre dies nicht über das GUI möglich. Aus diesem Grund fügte ich zwei weitere Bearbeitungsmöglichkeiten von Spielelementen ein: YAML und JSON. Beide sind wie Markdown Auszeichnungssprachen, die allerdings zur Datenserialisierung, also zum organisierten Speichern von Daten genutzt werden.[23] So konnte man nun durch das Wechseln zwischen den jeweiligen Registerkarten flexibel zwischen verschiedenen Anzeigeformen wechseln und auch Datenfelder, welche in der grafischen Oberfläche nicht verfügbar sind, erstellen und befüllen. Der große Vorteil von YAML ist, dass alle Informationen kompakt in einem Textfeld angezeigt werden können:

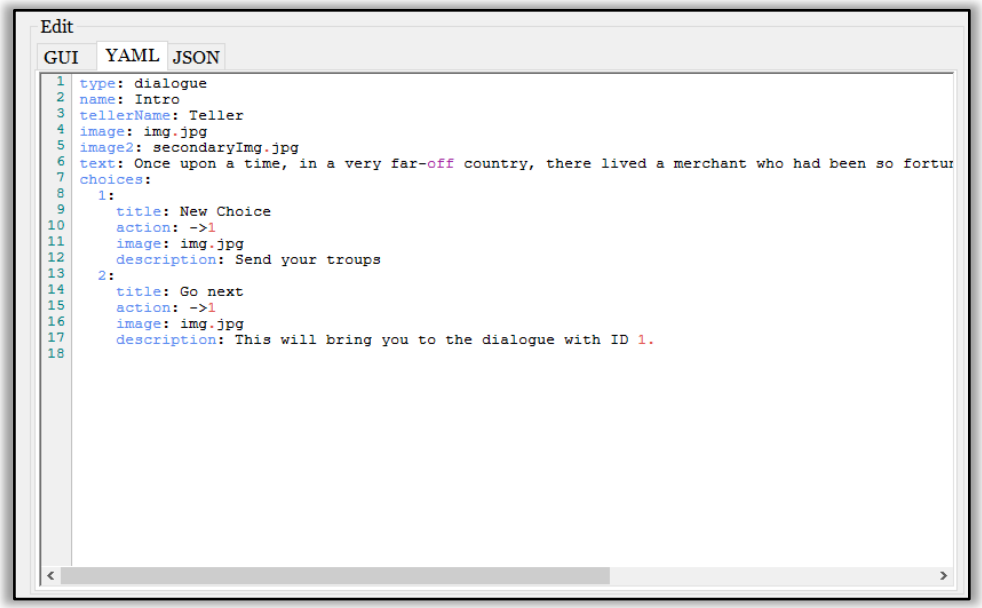


Abbildung 12: Bearbeitung eines Dialoges in MysteryMaker mit YAML

[23] Vgl. Keshavarzi, Kaveh und Bayer, Thomas. predic8. „JSON, XML und YAML im Vergleich“. https://www.predic8.de/xml-json-yaml.htm.

Allerdings kann diese Art der Dateneingabe auf unerfahrene Nutzer befremdlich wirken und schnell zu Formatierungsfehlern führen. Selbst ein Leerzeichen zu viel oder zu wenig führt dazu, dass das Projekt nicht einmal gespeichert oder in eine andere Anzeigeform konvertiert werden kann. Um dies zu vermeiden wird bei der JSON-Anzeigeform kein purer JSON-Quelltext angezeigt, sondern durch ein externes Tool verbildlicht:

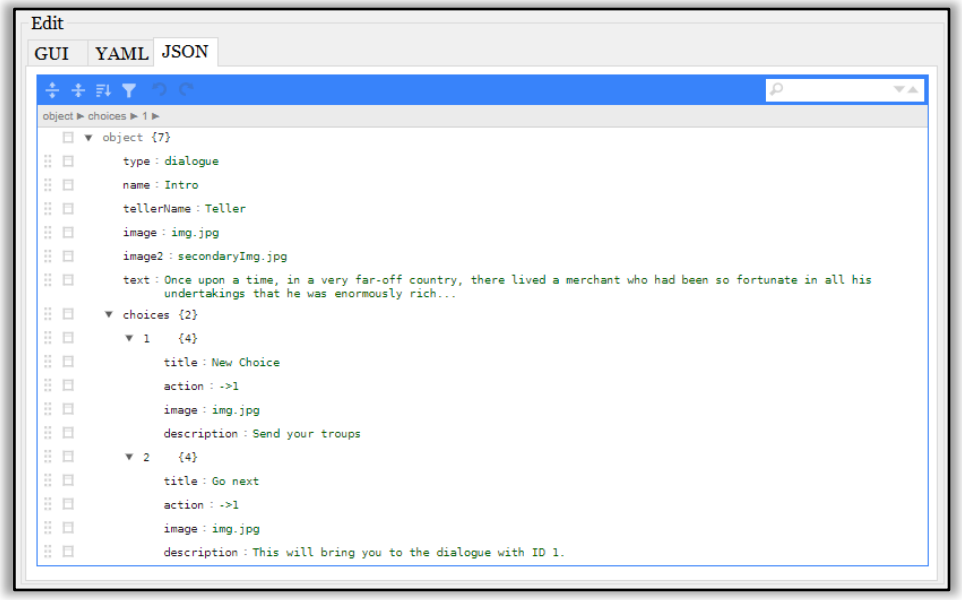


Abbildung 13: Bearbeitung eines Dialoges in MysteryMaker mit JSON-Visualisierung

Die JSON-Anzeige kombiniert daher die Vorteile beider Sprachen: Der Nutzer kann beliebig neue Datenfelder hinzufügen, bearbeiten und überblicken, ohne auf die Syntax der Sprache achten zu müssen.

Ein weiteres Problem dem ich während der Entwicklung begegnet bin, war das Einbinden von Interaktion. Der Autor einer interaktiven Fiktion sollte bestimmen können, was passieren soll, sobald der Spieler eine Dialogoption ausgewählt hat. Anfangs entschied ich mich für eine eigens entwickelte Skriptsprache, um den Nutzern möglich viele Möglichkeiten offen zu lassen. Allerdings ist das Erlernen einer Skriptsprache zeitintensiv und würde viele potentielle Autoren abschrecken, da sie nicht die bereits von ihnen erlernten Kenntnisse anwenden könnten. Aus diesem Grund wechselte ich zu der bereits etablierten und vor allem in der Godot Engine bereits integrierten Skriptsprache GDScript. Dadurch wurde auch die Kommunikation zwischen Skript und Velius Engine, welche ebenfalls in GDScript geschrieben worden war vereinfacht. Allerdings könnte GDScript, auch wenn es als sehr anfängerfreundlich gilt, auf nicht-Programmierer sehr abschreckend, als auch unverhältnismäßig komplex für einfache, vielleicht sogar lineare Geschichten wirken. Daher

habe ich nachträglich noch die Option eingefügt, simple Operationen wie Sprünge zu anderen Dialogen ohne Schreiben eines Skriptes durch ein Dropdownmenü zu ermöglichen:

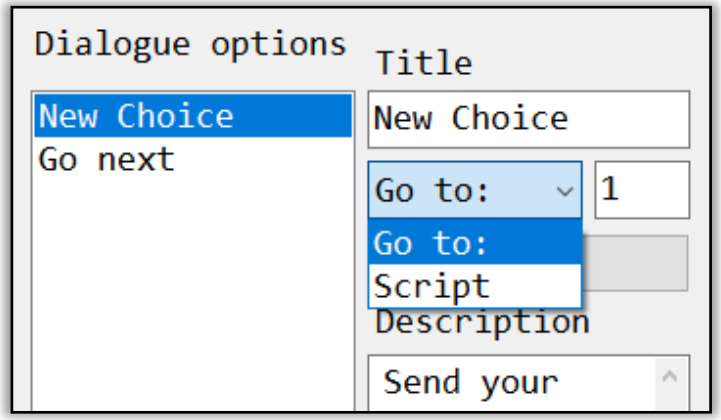


Abbildung 14: Der „Go to"-Shortcut ermöglicht das Wechseln zu einem anderen Dialog durch Angabe der Dialog-ID

3.3 Finaler Stand

Auch wenn der MysteryMaker auf Grund fehlender Optionen wie das Abspielen von Geräuschen oder dem noch nicht implementierten Inventar-System noch kein vollwertiges Autorensystem ist, lassen sich bereits nicht-lineare und somit interaktive Fiktionen erstellen. Auch das Theme-System verleiht Theme-Entwicklern umfangreiche Möglichkeiten zur Gestaltung der Spielererfahrung. Das System ist so offen, dass sogar ganze Spiele im Spiel selbst erstellt werden können. So könnte man die geschriebenen Geschichten noch interaktiver gestalten, indem man nicht nur durch Dialogoptionen, sondern auch durch kleinere Spiele wie „Tic-Tac-Toe" oder Puzzle-Spiele die Geschichte beeinflussen kann.

Ein separates Plugin-System, durch das von anderen Entwicklern einzelne Funktionen nachgeliefert werden können, ist ebenfalls noch nicht vorhanden.

Eine große Einschränkung besteht derzeit noch in der Veröffentlichung der Projekte. Während ähnliche Autorensysteme wie Twine einen Webseiten-Export anbieten, durch den das Spiel unkompliziert als Browserspiel veröffentlicht werden kann[24], müssen MysteryMaker-Projekte manuell als Datei exportiert, verschickt und vom Spieler umständlich importiert werden.

[24] Vgl. Offizielle Website. „Twine / An open-source tool for telling interactive, nonlinear stories". https://twinery.org/.

4 Zukunftspläne und Fazit

Da ich während der Entwicklung des Autorensystems immer wieder auf Einschränkungen gestoßen bin, die durch die Wahl von Windows Forms als grundlegende Technologie entstanden sind, plane ich den MysteryMaker basierend auf einer neueren und etablierteren Technologie neu zu entwickeln. Das ist zwar sehr zeitintensiv, wird aber zu einer besseren Nutzerfahrung führen und die Weiterentwicklung beträchtlich erleichtern.

Zudem werde ich versuchen, das Problem der Veröffentlichung von MysteryMaker-Projekten mit Hilfe eines freien Marktplatzes zu lösen. Dieser wäre dann in der Velius Engine integriert, sodass Entwickler ihre Projekte einheitlich und zentral verwaltet veröffentlichen können und die Spieler ihre interaktiven Fiktionen bequem im Spiel entdecken, herunterladen und spielen können.

Abschließend kann man sagen, dass auch wenn MysteryMaker sowie Velius Engine bereits viele Möglichkeiten zur Entwicklung interaktiver Fiktion bieten, noch viel zur Veröffentlichung einer Vollversion getan werden müsste. Da das Projekt aber OpenSource ist, und es somit jedem frei steht es mitzuentwickeln, ist eine stetige Weiterentwicklung auch in Zukunft möglich.

Quellenverzeichnis

„3 Tools zum Erstellen eigener Textabenteuerspiele / Gaming“. o. J. Ephesos Software Zugegriffen 30. November 2021. https://ephesossoftware.com/articles/gaming/3-tools-to-create-your-own-text-adventure-games.html.

„ADRIFT: Create your own Interactive Fiction“. o. J. Zugegriffen 29. Oktober 2021. https://www.adrift.co/.

„Adventure (1976)“. 2021. In Wikipedia. https://de.wikipedia.org/w/index.php?title=Adventure_(1976)&oldid=213115756.

„Adventure“. 2021. In Wikipedia. https://de.wikipedia.org/w/index.php?title=Adventure&oldid=216090578.

„Alles, was Du über Game Engines wissen musst“. 2020. nobreakpoints® (blog). 14. Juli 2020. https://blog.nobreakpoints.com/game-engines/.

„Godot Engine - Free and Open Source 2D and 3D Game Engine“. Godot Engine. Zugegriffen 29. November 2021. https://godotengine.org/.

„How It Works”. 2021. Vuepress Guide. Zugegriffen 29. November 2021. https://vuepress.vuejs.org/guide/#how-it-works

„Interactive Fiction“. 2021. In Wikipedia. https://de.wikipedia.org/wiki/Interactive_Fiction.

„markdown.de | Markdown Syntax-Dokumentation“. o. J. Zugegriffen 29. November 2021. https://markdown.de/.

„Multimedia im Überblick/ Gestaltung/ Inhalte/ Autorensysteme – Wikibooks, Sammlung freier Lehr-, Sach- und Fachbücher“. o. J. Zugegriffen 29. Oktober 2021. https://de.wikibooks.org/wiki/Multimedia_im_%C3%9Cberblick/_Gestaltung/_Inhalte/_Autorensysteme.

„Quest - Write text adventure games and interactive stories“. o. J. Zugegriffen 29. Oktober 2021. http://textadventures.co.uk/quest.

„Reference documentation” 2020. Microsoft Docs. Zugegriffen 22.01.2022. https://docs.microsoft.com/en-us/style-guide/developer-content/reference-documentation.

„Twine / An open-source tool for telling interactive, nonlinear stories“. o. J. Zugegriffen 29. Oktober 2021. https://twinery.org/.

„Visual Novel“. 2021. In Wikipedia. https://de.wikipedia.org/w/index.php?title=Visual_Novel&oldid=216681037.

„Was genau sind Autorensysteme?“ o. J. bild.de. Zugegriffen 29. Oktober 2021. https://www.bild.de/infos/e-learning/e-learning/autorensysteme-10752772.bild.html.

„Was ist eine API? Einfach erklärt!“ o. J. Talend - A Leader in Data Integration & Data Integrity. Zugegriffen 29. November 2021. https://www.talend.com/de/resources/was-ist-eine-api/.

Adegeo. „Desktop Guide (Windows Forms .NET)“. 2021. Microsoft Docs. Zugegriffen 22.01.2022. https://docs.microsoft.com/de-de/dotnet/desktop/winforms/overview/?view=netdesktop-6.0.

Augsten, Stephan. o. J. „Was ist eine Softwaredokumentation?“ Zugegriffen 29. November 2021. https://www.dev-insider.de/was-ist-eine-softwaredokumentation-a-744924/.

Funes, Antonio. „Spiele-Engines: Was ist eine Engine?”. 2020. PC GAMES. 24. Mai 2020. https://www.pcgames.de/Unreal-Engine-Software-239301/Specials/spiele-games-grafik-3d-historie-allgemein-epic-vorschau-1350751/.

Keller, Bernd. o. J. „interactive-fiction.de“. Text. Bernd Keller. Germany. Zugegriffen 29. Oktober 2021. http://www.interactive-fiction.de/.

Keshavarzi, Kaveh und Bayer, Thomas (2011) „XML, JSON und YAML im Vergleich“. o. J. predic8.com. Zugegriffen 28. Dezember 2021. http://www.predic8.de/xml-json-yaml.htm.

lern.link-Team. o. J. „Autorensystem“. lern.link - eLearning mit Moodle und PowerPoint (blog). Zugegriffen 29. Oktober 2021. https://lern.link/2020/lexikon/autorensystem/.

Lovato, Nathan. „GDScript Docs Maker”. 2021. GitHub. Zugegriffen 22.01.2022. https://github.com/GDQuest/gdscript-docs-maker.

Teufel, Yvonne. „Was Ist Ein Technologie-Stack?“ 2021. Mixpanel. Zugegriffen 23. Juli 2021. https://mixpanel.com/de/blog/was-ist-ein-technologie-stack/.

Abbildungsverzeichnis

Versicherung der selbstständigen Erarbeitung

Hiermit versichere ich, dass ich die Arbeit selbstständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und die Stellen der Facharbeit, die im Wortlaut oder im wesentlichen Inhalt aus anderen Werken entnommen wurden, mit genauer Quellenangabe kenntlich gemacht habe. Verwendete Informationen aus dem Internet sind dem(r) Lehrer/in vollständig im Ausdruck zur Verfügung gestellt worden. Die genutzten Internettexte habe ich alle auf beiliegender CD (ggf. USB-Stick) ordnungsgemäß gespeichert. Ich stelle die Projektarbeit der Schule für unterrichtliche Zwecke zur Verfügung. Die Projektarbeit darf an interessierte Personen ausgeliehen werden.

Ort, Datum:

Unterschrift: