Hector Seminar

Karlsruher Institut für Technologie

Gamification zur Annotation von Torahzeichen



Abschlussbericht der Kooperationsphase 2025/26

Durchgeführt am Scientific Computing Center (KIT)

Betreut durch

Volker Hartmann (KIT), Maximilian Inckmann (KIT), Lilly Osburg (KIT)

Erik Dan Șenchiu und Frederik Haase

Kurs KA 20

Inhaltsverzeichnis

1. Hintergrund **2**
1.1. Torahrollen . . . 2
1.2. Dekorationen bei Torahzeichen . . . 2
1.3. Metadaten . . . 2
1.4. Annotationsguideline . . . 3
1.5. Citizen-Science . . . 3
1.6. Gamification . . . 3
1.7. Herausforderungen . . . 4
1.8. Chancen . . . 4

2. Grundlagen **4**
2.1. Unity . . . 4
2.2. Java, Spring Boot und Gradle . . . 5
2.3. Docker . . . 6
2.4. Git und GitHub . . . 6
2.5. Datensatz . . . 7

3. Design **8**
3.1. Game Design . . . 8
3.1.1. Point-and-Click-Adventure . . . 8
3.1.2. Story und Aufgaben . . . 9
3.1.3. Annotieren . . . 9
3.1.4. Annotationsguideline . . . 12
3.2. Softwarearchitektur . . . 12
3.2.1. Client-Server-Architektur . . . 12
3.2.2. Anforderungen an den Server . . . 13
3.2.3. Anforderungen an den Client . . . 13
3.3. Assets und Art . . . 13

4. Implementierung **15**
4.1. Spiel und Client . . . 15
4.1.1. Implementierung mit wiederkehrenden Mustern . . . 15
4.2. Server . . . 16
4.2.1. Datenbank . . . 16
4.2.2. Programmstruktur . . . 17

4.2.3. API 17
4.3. Datenausgabe 18
4.3.1. Konfidenz-Berechnung 18
4.3.2. Struktur der Ausgabe 19

5. Diskussion **19**

6. Fazit **22**

7. Ausblick **23**
7.1. Verfeinerung der Algorithmen bei der Konfidenz-Berechnung 23
7.2. Erweiterung des Spiels 24

8. Danksagung **28**

9. Selbstständigkeitserklärung **29**

10. Literatur **30**

A. Annotationsguideline **31**

B. Links **33**

C. Prompt zur Bildgenerierung **33**

Abbildungsverzeichnis

1. Verschiedene Vorkommen des Buchstaben Shin im Datensatz 7
2. Struktur des Spiels (Szenenkarte) 10
3. In-Game Annotationsfenster 11
4. Komponentendiagramm der Anwendung 12
5. KI-generierte Hintergrundbilder aus dem Spiel 14
6. Struktur der Datenbank 16

Abstract

Some letters in Torah scrolls feature decorative elements, such as tagin, which are the focus of this project. In modern Torah scrolls, there are many established rules on which letters should carry decorations and which should not. Historically, different rule sets and intentions behind these decorations have existed. To learn how decorations changed over time, scientists have to annotate large amounts of letters. To effectively scale up this process, this project built a system which enables large groups of volunteers to support the scientific research. Additionally, gamification was used to increase the amount of time said volunteers spend annotating letters. The developed system combines a web-based game, in which users annotate letters, with a server application that manages users and their annotations. By statistically weighting the annotations from different users, the collected data can be made suitable for scientific analysis despite being gathered by volunteers without specialized expertise.

1. Hintergrund

1.1. Torahrollen

Die Torah ist der zentrale Text des Judentums. Sie enthält die fünf Bücher Mose und umfasst über 300 000 Buchstaben. Für die gottesdienstliche Nutzung liegt sie als handgeschriebene Torahrolle (Sefer Torah) vor. Diese werden von einem ausgebildeten Schreiber, dem Sofer, mit Feder und Tinte auf Pergament übertragen.[1] Dabei bestimmen strenge Regeln jedes Detail, von der Form der Buchstaben zu der Anordnung der Spalten. Trotz strenger Vorgaben unterscheiden sich Torahrollen verschiedener Epochen und Regionen in feinen Details.

1.2. Dekorationen bei Torahzeichen

Während der Text der Torah selbst über alle Rollen hinweg identisch ist, sind die Verzierungen (siehe Beispiele in Abb. 1) der Buchstaben in historischen Rollen ein Bereich, in dem sich individuelle und regionale Schreibtraditionen und Regelwerke einfinden. Eine systematische Untersuchung dieser Dekorationen könnte darüber Aufschluss geben, wie sich diese Regeln im Laufe der Zeit entwickelt haben, wie sich verschiedene Entwicklungen gegenseitig beeinflusst haben und woher die heutigen, sehr genauen Vorgaben stammen. Um solche Muster zu erkennen, reicht die Betrachtung einzelner Rollenabschnitte jedoch nicht aus. Erst der Vergleich vieler Rollen mit sehr vielen Buchstaben lässt statistisch belastbare Aussagen zu. Dafür müssen jedoch zunächst die Buchstaben klassifiziert werden, die überhaupt eine Dekoration tragen, was von Hand wegen der schieren Menge an Daten so gut wie unmöglich wäre . Deswegen wird hier die Annotationsarbeit auf viele Freiwillige verteilt (siehe 1.5 und 1.6). Sobald ausreichend Bilder manuell annotiert wurden, sollen diese als Trainingsdatensatz für ein KI-Modell genutzt werden, um noch deutlich größere Mengen an Buchstaben auf Dekorationen untersuchen zu können.

1.3. Metadaten

Metadaten sind Daten, die andere Daten beschreiben, strukturieren oder identifizieren. Sie enthalten nicht den eigentlichen Inhalt eines Objektes, sondern Informationen über dieses Objekt. Sie können dabei helfen, Daten leichter zu finden, zu verstehen und zu verarbeiten. Als Beispiel kann die Photographie genannt werden. Das Bild selbst zeigt ein Motiv, während die zugehörigen Metadaten zum Beispiel festhalten, wann und wo die Aufnahme entstanden ist oder mit welcher Kamera und welchen Einstellungen geschossen wurde. So können diese zum Beispiel systematisch nach Aufnahmedatum geordnet werden.
Unsere Daten erhalten ihren wissenschaftlichen Wert erst durch Metadaten. Die Position des

Buchstaben in den Torahrollen und die gesammelten Informationen über die Annotation, etwa wie viele unabhängige Einschätzungen für denselben Buchstaben vorliegen, sind essentiell für die Auswertung der Daten. Ohne diese könnten die Torahzeichen selbst nicht ausgewertet werden, da diese keine eigenen Informationen enthalten.

1.4. Annotationsguideline

Da bei der Annotation reproduzierbare Daten gesammelt werden sollen, ist es wichtig, einen Standard festzulegen, an dem sich die Annotatoren und Annotatorinnen orientieren können. Dieser Standard wird in Form einer Annotationsguideline festgehalten. Die Annotationsguideline enthält Informationen darüber, nach welchen Kategorien annotiert werden soll. So wird verhindert, dass verschiedene subjektive Einschätzungen der Annotierenden starke Variationen in den Annotationen nach sich ziehen.[2]

1.5. Citizen-Science

Citizen-Science bezeichnet die aktive Beteiligung von Laien bzw. Freiwilligen an wissenschaftlichen Forschungsprozessen, meist beim Sammeln, Klassifizieren oder Auswerten von Daten. Hierdurch wird eine Datenerhebung in einem Umfang möglich, der für einzelne Forschende oder Forschungsgruppen aus räumlichen, zeitlichen und personellen Gründen normalerweise unerreichbar wäre.
Eine große Herausforderung ist hierbei die Validierung der Ergebnisse, da die Daten nicht von geschulten Fachleuten, sondern von Laien mit nur geringem Vorwissen, annotiert werden, und somit leichter Fehler passieren können. In der Bevölkerung muss man oft erst das Interesse wecken, die Daten gründlich annotieren zu wollen. Hierfür bietet sich beispielsweise die Methode Gamification an.

1.6. Gamification

Gamification ist eine Methode, um zum Beispiel im Rahmen von Citizen-Science bei einer breiten Menschenmenge Interesse zu wecken. Gamification kommt von dem englischen Wort Game (=Spiel) und steht für die Verwendung spielerischer Elemente in einem spielfremden Kontext. Hierbei kann es sich um vieles Handeln: z. B. eine Rangliste, eine Abenteuer-Reise oder eine Belohnung. Dieses Prinzip wird oft bei Aufgaben genutzt, die für die meisten Personen eher unattraktiv sind. Die intrinsischen Motivation diese Aktion durchzuführen soll gesteigert werden, indem man den Spielenden ein Ziel gibt, das sie gerne erreichen möchten. Dieses verbindet

man mit einer ursprünglich eher langweiligen Aufgabe. Das Verlangen, das Ziel zu erreichen, verdrängt dabei die Langeweile, die Spielenden werden also motiviert, die Aufgabe zu erfüllen. Gamification bedeutet zusammengefasst das Anreichern von Nicht-Spiel-Anwendungen mit Spielelementen.

1.7. Herausforderungen

Wie bereits erwähnt, müssen sehr viele Torahzeichen ausgewertet werden, was von einzelnen Personen nur sehr schwer umsetzbar ist. Die deshalb notwendige Verteilung des Annotationsprozesses auf eine größere Gruppe führt allerdings zu neuen Herausforderungen. Während sich einzelne Personen einer geschlossenen Forschergruppe sehr gut auf einen gemeinsamen Standard festlegen können, muss bei der Ausweitung auf größere Gruppen ein besonderer Fokus darauf gelegt werden, dass alle Bearbeiterinnen und Bearbeiter nach dem gleichen System annotieren. Eine weitere Herausforderung ist die Organisation der verschiedenen Annotationen. Während kleine Gruppen noch einfach gemeinsam Tabellen bearbeiten können, ist eine solche Vorgehensweise bei großen Gruppen nicht denkbar. Hier ist es nötig, ein zusätzliches Programm zu nutzen, um die Annotationen zusammenzufassen und zu sammeln, während gleichzeitig verhindert wird, dass Personen, die nicht gewissenhaft annotieren, große Schäden in den gesammelten Daten verursachen. Da nicht nur mit Fachleuten gearbeitet wird, ist außerdem eine regelmäßige Prüfung der Qualität nötig. Verstärkt wird diese Notwendigkeit dadurch, dass selbst sehr genaue Annotationsrichtlinien in schwierigen Fällen unterschiedlich ausgelegt werden können.

1.8. Chancen

Neben dem bloßen Annotieren großer Mengen an Daten hat die Ausweitung auf eine große Gruppe im Rahmen eines Citizen-Science-Projekts (siehe auch 1.5) auch weitere Vorteile. So haben derartige Projekte in der Bevölkerung das Potenzial, die Forschung erlebbarer zu machen und dadurch das Interesse zu steigern. Auch für die Forschung bietet eine Ausweitung der Gruppe die Möglichkeit, die Annotationen neutral zu halten und subjektive Ansichten durch die große Menge an Annotierenden auszugleichen.

2. Grundlagen

2.1. Unity

Bei der Realisierung des Spiels als gamifiziertes Annotationstool kommt Unity als zentrale Entwicklungsumgebung zum Einsatz. Unity ist eine Game-Engine, die für die Erstellung und

Optimierung von Videospielen ausgelegt ist.
Als Programmiersprache verwendet Unity primär C#. Klassen in C# definieren das Verhalten einzelner Spielobjekte, wobei sie folgendermaßen aufgebaut sind: Methoden wie `Start()` laufen einmalig bei der Initialisierung ab, während `Update()` in jedem Frame ausgeführt wird. Durch dieses Modell wird die Spiellogik strukturierter und erleichtert die Umsetzung interaktiver Abläufe.
Ein wichtiges Merkmal von Unity ist die komponentenbasierte Architektur. Eine Szene besteht aus sogenannten GameObjects, die ihre Funktionalität durch das Hinzufügen von Komponenten erhalten, wie zum Beispiel Renderer für die graphische Darstellung, Collider für die Kollisionserkennung, Rigidbodies zur Simulation physikalischer Gesetze oder C#-Skripts für die Spiellogik. Dieses modulare Prinzip verbessert die Wiederverwendbarkeit von Code und Klassen. Prefabs sind Vorlagen von Spielobjekten und können beliebig oft instanziiert (erzeugt) werden. Die von Unity mitgelieferten Bibliotheken und Systeme, darunter zum Beispiel eine integrierte Physik-Engine, ein System, das die physikalischen Eigenschaften und Bewegungen von Objekten simuliert, und ein integriertes System für die Erstellung von Benutzeroberflächen, eignen sich besonders gut für das Programmieren von Spielobjekten und deren Interaktionen untereinander. Das Echtzeit-Rendering der Engine unterstützt zudem eine interaktive Arbeitsweise: Jegliche Änderung am Code oder an Objekten zeigt sich unmittelbar im laufenden Spiel, da es sich jederzeit direkt im Editor starten und testen lässt. Der Unity-Editor selbst stellt viele Möglichkeiten zur Verfügung, aktive Anpassungen am Spiel vorzunehmen, ohne den Code verändern zu müssen. Öffentliche Variablen eines Skripts erscheinen automatisch im Inspector, einem Editor-Fenster in Unity, und lassen sich dort auch zur Laufzeit des Spiels modifizieren, wodurch beispielsweise Spielparameter wie die ID der nächsten Szene interaktiv geändert werden können. Diese Eigenschaft der Entwicklungsumgebung beschleunigt den Entwicklungsprozess erheblich. Ein weiterer entscheidender Vorteil liegt in der Plattformunabhängigkeit der Engine. Ein einmal erstelltes Projekt lässt sich ohne grundlegende Anpassungen für eine Vielzahl an Plattformen exportieren, darunter Windows, macOS und Linux, aber auch für mobile Systeme oder über WebGL (einer Plattform für das Bauen von Web-Spielen) in einem Browser. Diese einfache Bereitstellung macht Unity für das Projekt besonders attraktiv, da das fertige Spiel so ohne Installation im Browser ausgeführt werden kann und somit für viele Menschen deutlich einfacher erreichbar ist.

2.2. Java, Spring Boot und Gradle

Java ist eine objektorientierte Programmiersprache, die durch die Java Virtual Machine auf vielen Betriebssystemen lauffähig ist. Sie gehört zu den weltweit populärsten Programmiersprachen.

Zur Entwicklung des Servers wird Spring Boot genutzt, das ein 2014 veröffentlichtes Framework für Java ist. Es ermöglicht, deutlich schneller und einfacher Anwendungen zu entwickeln. Dabei konfiguriert und richtet es viele Komponenten automatisch ein, sodass sich in der Entwicklung auf die Geschäftslogik konzentrieren werden kann.[3] Es baut dazu auf Spring Framework auf, das auf dem Inversion-of-Control (IoC) Prinzip basiert. Das heißt, dass nicht jede Komponente selbstständig ihre Abhängigkeiten instanziiert, sondern diese vom sogenannten Spring IoC Container verwaltet werden. Das hat den Vorteil, dass Komponenten stärker voneinander getrennt sind und somit einzeln getestet und weiter entwickelt werden können. Zusätzlich wird es einfacher, Komponenten mehrfach zu verwenden. So können verschiedene Klassen z. B. auf denselben Dienst zugreifen.[4] Ein weiterer Vorteil von Spring Boot ist, dass es Bibliotheken, die häufig zusammen genutzt werden, in vorgefertigten Modulen zur Verfügung stellt. Dadurch kann sichergestellt werden, dass alle Bibliotheken in ihren Versionen zueinander passen. Zusätzlich können bei einem Update von Spring Boot alle Pakete passend mit aktualisiert werden.[5] Zur Verwaltung der benötigten Pakete wird Gradle genutzt. Es sorgt dafür, dass alle Abhängigkeiten in zueinander passenden Versionen vorhanden sind. Letztendlich wird über das Spring Boot Plugin ein ausführbares JAR erstellt.[6]

2.3. Docker

Docker ist eine Plattform, die es ermöglicht, verschiedene Prozesse in Containern zu verpacken und zu nutzen. Durch die Nutzung einfacher Konfigurationsdateien kann beschrieben werden, wie Docker verschiedene Container zu einer Anwendung verknüpfen soll. Auf verschiedenen Systemen wird so sichergestellt, dass sich die vorhandene Software gleicht. Es wird somit verhindert, dass verschiedene Personen unterschiedliches Verhalten der Software beobachten. Gleichzeitig wird durch die Nutzung von vorgefertigten Containern keine zusätzliche Konfiguration im Host-System nötig. Alle Abhängigkeiten und Konfigurationen, die zum Ausführen der Anwendung gebraucht werden, können direkt in Docker eingebunden werden. Im Vergleich zu herkömmlichen virtuellen Maschinen erhält allerdings nicht jeder Container ein eigenes, vollwertiges Betriebssystem. Stattdessen laufen alle Container im Kernel des Gastgebersystems. Separiert werden sie dabei mit verschiedenen Funktionalitäten im Linux-Kernel. Durch diese weniger starke Trennung hat das System potenziell nicht die Sicherheit einer herkömmlichen virtuellen Maschine, dafür werden die benötigten Ressourcen und die Zeit, die das Hochfahren benötigt, massiv reduziert.[7]

2.4. Git und GitHub

Zur Versionskontrolle wird Git in Kombination mit GitHub genutzt. Git ermöglicht es dabei, immer wieder Snapshots der aktuellen Dateiversion zu speichern. Das macht es möglich, beim

plötzlichen Auftreten von Fehlern, wieder zu einer früheren Version zurückzukehren, wodurch ein versehentliches Zerstören des Projekts verhindert wird. Indem Git genutzt wird kann außerdem von mehreren Personen an verschiedenen Stellen etwas geändert werden und diese Änderungen können automatisch wieder zusammengefügt werden.[8] GitHub dient in diesem Prozess als Server, der die aktuelle Version des Projekts zur Verfügung stellt und auf dem die einzelnen Änderungen gesammelt werden, was die lokalen Repositories synchron hält.

2.5. Datensatz

Der Datensatz, der annotiert werden soll, stammt aus der Arbeit “Towards a Dataset for Paleographic Details in Historical Torah Scrolls” [9]. Dort wurden zunächst bestimmte Scans historischer Torahrollen ausgewählt, die in ihrer Qualität ausreichend sind und unter einer freien Lizenz verfügbar sind. Diese Scans wurden anschließend automatisch in einzelne Buchstaben zerlegt und transkribiert. Ein kleiner Teil der Buchstaben wurde anschließend bereits annotiert.



(a) Ms. or. fol. 1216 (b) Ms. or. fol. 133 (c) BSB Cod.hebr. 501 (d) BSB Cod.hebr. 501

Abbildung 1: Verschiedene Vorkommen des Buchstaben Shin im Datensatz
Quellen: Digitalisate der Staatsbibliothek zu Berlin (*Ms. or. fol. 1216*, *Ms. or. fol. 133*) und der Bayerischen Staatsbibliothek (*BSB Cod.hebr. 501*).

In vielen Fällen, wie in (a) und (b) aus Abbildung 1 ist die Annotation sehr klar. Teilweise kann es aber wie in (c) und (d) durch eine niedrige Qualität der Digitalisierung schwieriger sein, zu erkennen, ob ein Buchstabe dekoriert ist. In diesem Fall ist (c) dekoriert, während (d) undekoriert ist. Durch die automatisierte Erstellung des Datensatzes wurden in manchen Fällen die Zuschnittsmasken schlecht gesetzt, was zu nicht auswertbaren Bildern führt. Daher kann den Annotierenden nicht nur die Auswahl zwischen dekoriert und undekoriert gegeben werden, sondern es wird eine weitere Auswahlmöglichkeit eingebaut, um das Bild als Fehler zu markieren. So kann mit der Zeit auch der Datensatz selbst verbessert werden, indem häufig als fehlerhaft markierte Bilder nach Überprüfung entfernt werden können.

3. Design

3.1. Game Design

3.1.1. Point-and-Click-Adventure

Das Spiel wird als ein Point-and-Click-Adventure entwickelt. Spiele dieses Genres bestehen aus einer Abfolge größtenteils statischer Szenen, durch die sich die Spielerinnen und Spieler allein per Mausklick bewegen. Ein Klick auf z. B. einen Pfeil wechselt die Szene, in der man sich befindet, wobei ein Klick auf ein Objekt eine Interaktion auslöst. Der Spielfortschritt entsteht hierbei nicht durch Geschick oder Reaktionszeit, sondern durch das Erkunden, Sammeln von Gegenständen, und am wichtigsten durch das Annotieren von Torahzeichen.
Aus mehreren Gründen wird dieses Spielgenre genutzt: Der wichtigste ist die natürliche Integration der Annotationsaufgaben. In einem Spiel, in dem man einen Charakter kontrolliert und ihn hin und her bewegt gibt es eher wenig Möglichkeiten, das Annotieren einzubauen, ohne dass dies sehr aufgezwungen wirken würde und das Spielerlebnis stört, das sonst eher dynamisch und flüssig ist. In einem Point-and-Click-Adventure jedoch gibt es so gut wie keine Bewegung von Charakteren oder Objekten. Alles ist statisch, was jedoch dem Spielerlebnis keinen Abbruch macht, da sich dies eher auf die Szenerie, die Story und das Weiterkommen fokussiert als auf aktive Spieländerungen. Wenn sich also in diesem Spielgenre bei dem Klick einer Tür oder einer Truhe ein Menü öffnet, um Torahsymbole zu annotieren, fühlt sich das passender an und der Spielende wird nicht aus seinem Rhythmus gerissen. Dies führt zu einem besseren Spielerlebnis. Auch gibt es in einem Point-and-Click-Adventure sehr viele verschiedene Ansatzpunkte, an denen man die Annotation einbauen kann: z. B. beim Öffnen von Truhen und Kisten oder beim Aufheben eines schweren Gegenstandes. Dazu diktiert dieses Genre kein Tempo, wodurch gelassener gespielt werden kann. Das kann auch dazu führen, dass mit größerer Sorgfalt annotiert wird, was für die Spielenden heißt, dass sie das Spiel in ihrem eigenen Tempo durchspielen können. Daher ist ein besser angepasstes Spielerlebnis vorzuzeigen und die Annotation fällt eher sorgfältig statt hastig aus.
Ein weiterer Grund ist die niedrige Einstiegshürde. Die gesamte Steuerung besteht aus dem Bewegen und Klicken der Maus, es gibt also so gut wie keine Tastenkombinationen oder Reflexanforderungen zu erlernen. Damit ist das Spiel für ein möglichst breites Publikum zugänglich, was im Citizen-Science-Kontext sehr der Datenerhebung zugutekommt, da viele Menschen eher weniger Erfahrung mit Videospielen haben.
Schließlich ist die Realisierbarkeit eines Point-and-Click-Adventures viel höher als die eines bewegten z. B. 2D-Adventure-Spiels. Statische Szenen benötigen so gut wie keine Animationen und verlangen weder Bewegungslogik noch Physiksimulationen. Die Spielwelt besteht primär

aus Hintergrundbildern und darauf platzierten, klickbaren Objekten. Das passt zum einen zu einer KI-gestützten Asset-Erstellung (siehe Kapitel 3.3) und zum anderen zur datengetriebenen Architektur des Spiels (siehe Kapitel 4.1.1). Genre, Grafikproduktion und technische Umsetzung des Spiels greifen so ineinander.

3.1.2. Story und Aufgaben

Um Spielerinnen und Spieler bis zum Ende des Spiels durchhalten zu lassen, ist es notwendig, eine attraktive Hintergrundhandlung zu kreieren: So setzt die Handlung nach einem Fluch an, der das Land befallen und alle Tiere verschwinden lassen hat, mit der einzigen Ausnahme des Spielenden. Dieser wandert nun als letztes verbleibendes Wesen durch die Welt auf der Suche nach einem Weg, diesen Fluch zu brechen. Diese Ausgangslage erfüllt zwei Hauptzwecke: Zum einen erklärt sie auf verständliche Weise, warum in der Spielwelt keine anderen Menschen wie Dorfbewohner vorhanden sind und führt dazu, dass das Spiel keine bewegten oder animierten Charaktere benötigt. Zum anderen soll sie den Wunsch auslösen, diesen Fluch aufzuheben und damit die Welt zu retten. Wenn das Spiel so eher durchgespielt wird, können dadurch mehr Daten gesammelt werden.
Das Ziel der Spielenden ist hierbei das Erreichen der Szene GRD3, in der sich der Legende nach eine mysteriöse Kugel befindet, die den Fluch brechen kann. Von dieser Kugel erfährt man jedoch nicht direkt, sondern über verstreute Schriftstücke, die im Laufe der Erkundung in Kisten zu finden sind und auf das Artefakt und seinen Aufbewahrungsort hindeuten. Die Story wird also nicht in Zwischensequenzen erzählt, sondern durch Erkundung und Lesen der Schriftstücke zusammengesetzt.
Abbildung 2 zeigt die vollständige Struktur des Spiels in seiner derzeitigen Version: alle Szenen sind mitsamt ihren Verbindungen und allen Items dargestellt. Hierbei ist es wichtig, dass man ohne ein Item nicht weiter vordringen kann, da der Weg versperrt ist. So muss zum Beispiel der Weg von P2 zu der Höhle in C1 erstmal mit einer Axt freigemacht werden, da der Höhleneingang mit dicken Planken zugenagelt ist. In Violett ist der Ort dargestellt, an dem die spielende Person ein jeweiliges Item bekommen kann, und in Orange ist der Ort dargestellt, an dem dieses Item benötigt wird. Fund- und Verwendungsorte eines Items liegen ganz bewusst auseinander, sodass man die Welt erkunden muss.

3.1.3. Annotieren

Das Annotieren der Torahzeichen ist als zentrale Spielmechanik in das Spiel eingebettet. Jede bedeutsame Interaktion in der Spielwelt ist durch eine Annotationsrunde verdeckt: Wird auf eine geschlossene Truhe, eine verriegelte Tür oder ein Interactable geklickt, öffnet sich zunächst das

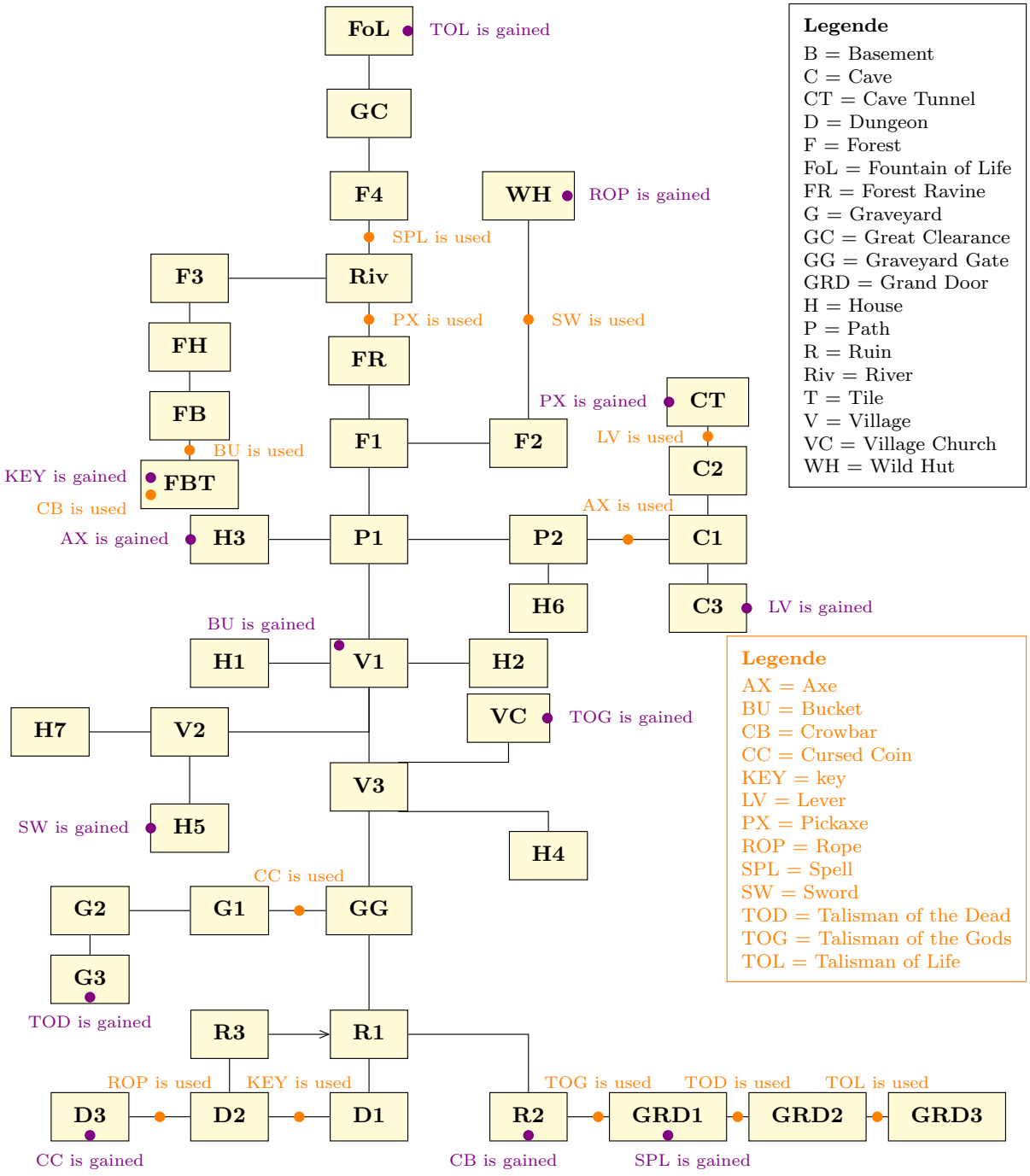


Abbildung 2: Struktur des Spiels (Szenenkarte)

Annotationsmenü (siehe Abbildung 3). In diesem werden nacheinander 18 verschiedene Fotos von Torahzeichen angezeigt, bei denen man jeweils entscheiden muss, ob das entsprechende Zeichen im Bild dekoriert oder undekoriert ist. Zusätzlich werden weitere Informationen bereitgestellt: Zum einen wird der Buchstabe in seiner normalen Form angezeigt, damit auch bei Fotos, auf denen mehrere Buchstaben vollständig sichtbar sind, klar ist, welcher Buchstaben annotieren werden soll. Zum anderen wird die Option angeboten, „Bad Data“ auszuwählen, wenn der Buchstabe im Bild entweder nicht annotierbar oder nicht zu sehen ist.



Abbildung 3: In-Game Annotationsfenster

Die Balance zwischen dem Sammeln möglichst vieler Daten und dem Verhindern von Langeweile ist ein zentrales Problem im Spiel. Für jeden Fortschritt im Spiel müssen die Annotatoren und Annotatorinnen mehrere Bilder auswerten. Eine sehr geringe Anzahl an Bildern pro Schritt führt dazu, dass insgesamt weniger Bilder pro Spieldurchgang annotiert werden. Eine zu große Anzahl an Bildern kann dazu führen, dass die Unterbrechungen frustrierend werden, wodurch gar keine Daten mehr gesammelt werden, oder das Annotationsfenster nur möglichst schnell durchgeklickt wird, was die Datenqualität massiv verringert. Als Mittelweg müssen 18 Bilder pro Schritt annotiert werden. Da die meisten Bilder sehr deutlich einer Kategorie zugeordnet werden können, dauert eine Runde so in der Regel weniger als 30 Sekunden, was kurz genug ist

um die Story nicht deutlich zu stören.

3.1.4. Annotationsguideline

Damit die von vielen verschiedenen Freiwilligen erstellten Annotationen vergleichbar und wissenschaftlich verwertbar sind, braucht es eine gemeinsame Grundlage: die Annotationsguideline (siehe A). Sie legt fest, was annotiert wird, welche Kategorien zur Verfügung stehen und wie mit schwierigen Fällen umzugehen ist. Schwierige Fälle entstehen dabei in unserem Fall durch verschiedene Auslöser. Erstens können durch variierende Handschriften insbesondere durch Serifen Verwechselungen mit Dekorationen entstehen. Verschmutzungen und unterschiedliche Digitalisierungsmethoden und Qualitäten können ebenfalls zu Artefakten führen, die verschiedene Einschätzungen zur Folge haben können. Ein weiteres Problem beim Annotieren sind Fehler in der automatisierten Erstellung des Datensatzes. Diese treten z. B. dadurch in Erscheinung, dass das Bild nur weißen Hintergrund oder Buchstaben zeigt, die nicht mit der Transkription überein stimmen. Für solche schwierigen Fälle werden in der Annotationsguideline Beispiele gezeigt, um eine einheitliche Herangehensweise sicherzustellen.

3.2. Softwarearchitektur

3.2.1. Client-Server-Architektur

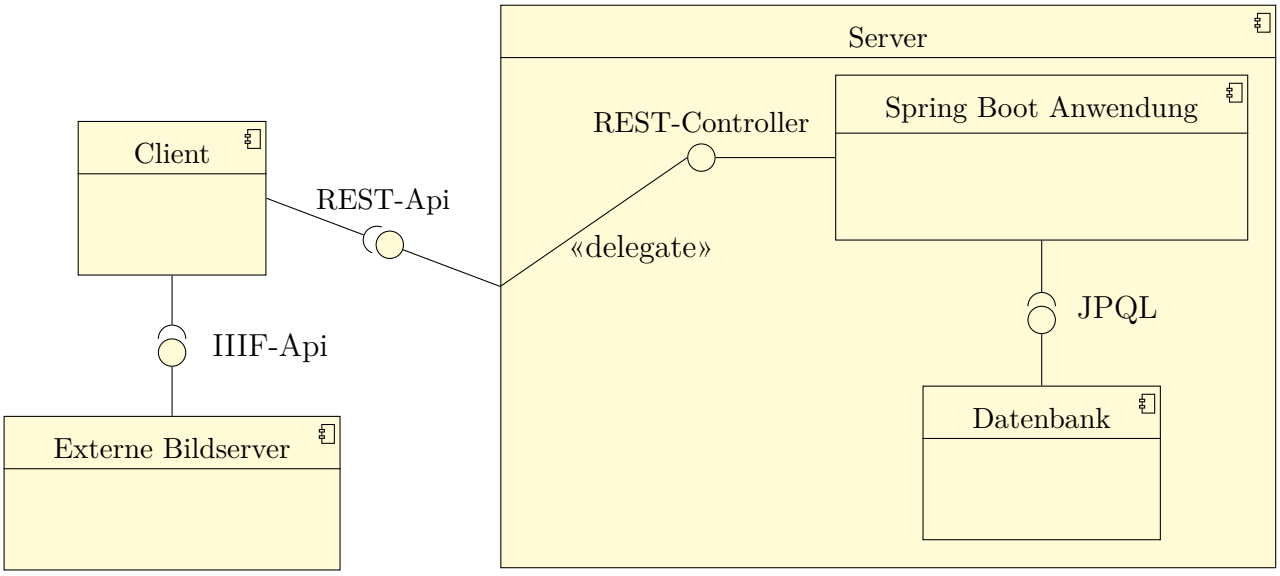


Abbildung 4: Komponentendiagramm der Anwendung

Die Software ist in einer Client-Server-Architektur aufgebaut. Die Annotatoren und Annotatorinnen interagieren dabei in ihrem Browser mit dem Client. Dieser enthält das Spiel und die Annotationsoberfläche. Er erhält dazu vom Server die URLs unter denen er die Bilder der Tora-Buchstaben laden kann. Sobald diese Buchstaben dann annotiert sind, werden die Ergebnisse wieder zurück an den Server gesendet, der sie in der Datenbank speichert. Server und Client kommunizieren dabei per HTTP über eine REST-API.

3.2.2. Anforderungen an den Server

Der Server ist nach einem Schichtenmodell aufgebaut. Die äußerste Schicht enthält dabei Controller, die die HTTP-Anfragen entgegennehmen. Die darin enthaltenen Informationen werden dann an die zweite Schicht weitergeleitet, die verschiedene Services enthält. Diese tragen die Geschäftslogik in sich und kommunizieren über eine dritte Schicht, die die verschiedenen Datenstrukturen enthält, mit der Datenbank. Anschließend senden die Services die Antworten zurück an die Controller, die diese an das Spiel weiterleiten. Die einzelnen Komponenten können so unabhängig voneinander entwickelt werden.
Für die Annotation muss der Server also in der Lage sein, dem Client die zur Darstellung nötigen Informationen zu senden. Die Annotationen muss der Server dann korrekt zuordnen und werten können. Anschließend muss der Server die Annotationen verschiedener Personen dann vergleichen und auswerten und die gesammelten Informationen benutzerspezifisch ausgeben können. So benötigen Forschende Statistiken zu einzelnen Buchstaben, während Annotierende eher ein Interesse an ihren eigenen Leistungen haben.

3.2.3. Anforderungen an den Client

Der Client stellt insbesondere die Oberfläche dar, mit der die Annotatoren und Annotatorinnen interagieren. Er muss dafür im Kontext von Gamification und Citizen-Science einfach und intuitiv verwendbar sein. Dazu muss er das Bild anzeigen und Annotationen über entsprechende Bedienelemente entgegennehmen können. Anschließend muss er in der Lage sein, die Klassifikationen zurück an den Server zu senden. Gleichzeitig sollte er ein interessantes Spiel zur Verfügung stellen.

3.3. Assets und Art

Sämtliche visuellen Assets und Szenenhintergründe des Spiels entstehen mithilfe generativer KI, genauer gesagt mit einer Mischung aus ChatGPT, Claude Fable und UseAI. Diese Entscheidung hat zwei Hauptgründe: Zum einen verfügt keiner der Autoren über nennenswerte Erfahrung

im Erstellen digitaler Grafiken, zum anderen würde sowohl das eigenhändige Zeichnen als auch das Zusammensuchen passender, frei lizenzierter Assets aus verschiedenen Quellen einen Zeitaufwand bedeuten, der im begrenzten Projektrahmen nicht zu rechtfertigen wäre. Zudem ergeben Assets aus unterschiedlichen Quellen kaum einen einheitlichen visuellen Stil, während die KI-Generierung eine stilistisch kohärente Spielwelt ermöglicht. Auch ist hierdurch garantiert, dass jede Szene, die man sich vorstellen konnte, auch generiert werden kann, während bei einem Zusammensuchen von Assets nie alles möglich wäre.



(a) Church Scene (VC) (b) River Scene (RIV)

Abbildung 5: KI-generierte Hintergrundbilder aus dem Spiel

Die Erstellung der Bilder ist dabei keineswegs einfach. Jedes Asset beginnt mit einem ausführlichen Prompt, der die gewünschte Szene, Perspektive und Stimmung detailliert beschreibt. Nur selten liefert bereits der erste Versuch ein verwendbares Ergebnis: Häufig enthalten die generierten Bilder logische Fehler, wie etwa Wege, die ins Nichts führen, Objekte an unplausiblen Positionen oder das Ergebnis entspricht schlicht nicht der Vorstellung. In solchen Fällen muss der Prompt angepasst und die Generierung wiederholt werden, teilweise mit gezielten Korrekturanweisungen für einzelne Bildbereiche. So benötigt ein einzelnes finales Bild zwischen einem und dreißig Generierungsversuche. Die KI-Nutzung ersetzt den kreativen Aufwand also nicht vollständig, sondern verlagert ihn: Von der handwerklichen Bilderstellung hin zum präzisen Formulieren, Bewerten und schrittweisen Verfeinern der Prompts. Ein Beispiel hierfür liegt im Anhang C.

4. Implementierung

4.1. Spiel und Client

4.1.1. Implementierung mit wiederkehrenden Mustern

Zuerst werden nicht sichtbare Spielinhalte wie konkrete Szenen oder Items, sondern wiederverwendbare Strukturen entwickelt. Das Ziel ist dabei, dass sich neue Inhalte wie Szenen, Truhen, Türen oder Interactables ohne Änderungen im Code hinzufügen lassen sollten, alleine durch das Anlegen neuer Daten. Diese Vorbereitung spart im weiteren Projektverlauf tatsächlich mehrfach Zeit, da die Szenen hierdurch schnell aus den immer gleichen Grundbausteinen zusammengesetzt werden können.
Das zentrale Muster hierfür ist ein datengetriebener Aufbau der Spielwelt. Jede Szene des Spiels existiert nicht als eigene Unity-Szene, sondern als Instanz der Klasse „SceneData“, genauer gesagt als ein ScriptableObject, das ausschließlich Daten enthält: den Hintergrund, die Position der Pfeile und eine Liste aller Türen, Truhen und Interactables mitsamt ihrer Eigenschaften wie Position, Skalierung, Zustand und Inhalt. Ein anderes Skript (SceneManager) liest bei jedem Szenenwechsel das passende Objekt aus und ersetzt die gerade angezeigte Szene, indem es die neue Szene anhand dieser Daten vollständig aufbaut. Eine neue Szene zu erstellen bedeutet hierdurch also lediglich, ein neues Datenobjekt im Unity-Editor zu kreieren und dessen Daten auszufüllen, ohne in den Code zu müssen.
Beim Erzeugen der Objekte folgt jeder Objekttyp demselben Spawner-Muster: Spawner, wie z. B. ArrowSpawner, DoorSpawner oder ChestSpawner, sind jeweils dafür zuständig, aus den gegebenen Daten konkrete Objekte zu instanziieren und beim Szenenwechsel wieder zu entfernen. Zudem gibt es zentrale Manager-Skripts wie den SceneManager oder QuestManager, die von jedem Objekt aus über eine einzige Instanz erreichbar sind. Der SceneManager speichert die Zustände aller Objekte (etwa welche Türen schon geöffnet, welche Truhen schon gelootet und welche Interactables schon benutzt wurden) in Dictionaries, die Objekte über eindeutige IDs referenzieren. Ohne dies wären jegliche Zustandsänderungen von Objekten sonst bei jedem Szenen-Wechsel verloren, da die Objekte selbst bei jedem Wechsel unwiderruflich gelöscht und neu erzeugt werden, sie selbst die Zustände also nicht dauerhaft speichern können. Durch die zentrale Speicherung im SceneManager bleibt der Fortschritt des Spielenden in den verschiedenen Szenen vorhanden.
Zusammengenommen führen diese Muster dazu, dass der Aufwand für neue Inhalte fast ausschließlich in deren Gestaltung liegt statt in deren Programmierung, was die Erweiterung des Spiels deutlich erleichtert (siehe Kapitel 7.2).

4.2. Server

4.2.1. Datenbank

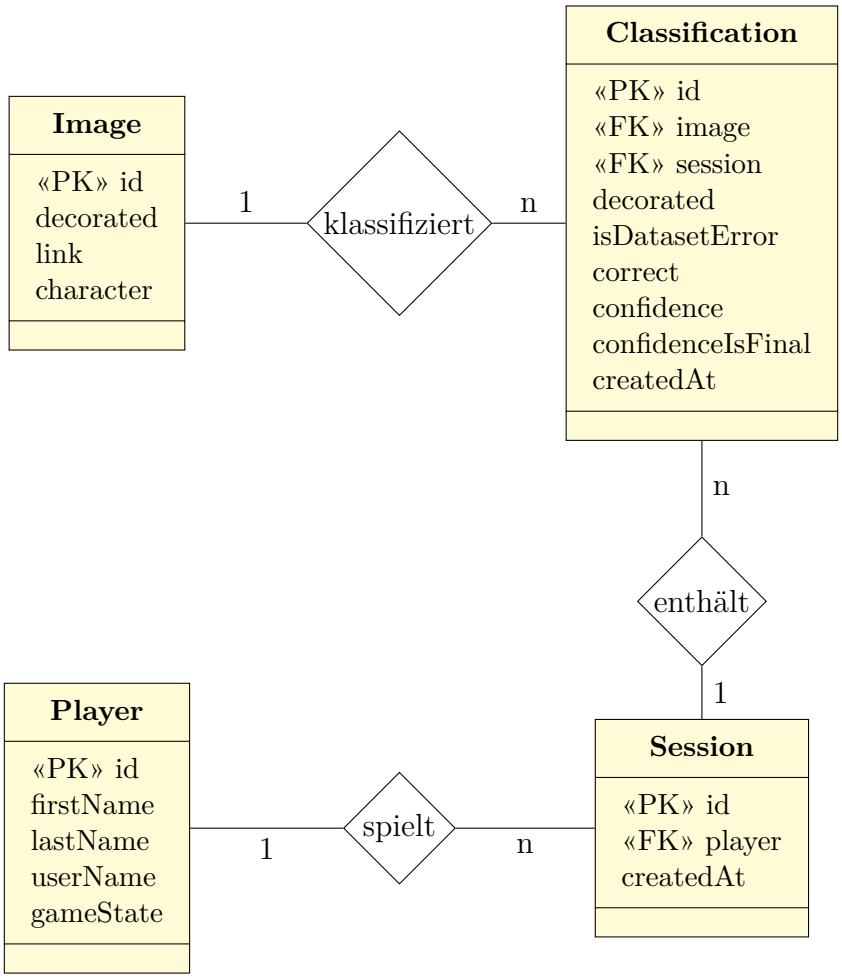


Abbildung 6: Struktur der Datenbank

Die Datenbank speichert Profile, Bilder und Annotationen in vier Tabellen. Dabei enthält die Tabelle *Image* zu allen Bildern des Datensatzes die URL unter der sie geladen werden können, den darauf dargestellten Buchstaben und, wenn sie im Datensatz schon annotiert sind, die tatsächliche Klassifikation in dekoriert und undekoriert. Die Annotationen der Spielenden sind in der Tabelle *Classification* abgespeichert und enthalten jeweils die zurückgemeldete Annotation, einen Zeitstempel und den berechneten Konfidenzwert. Wenn die Annotation ein Bild betrifft, das im ursprünglichen Datensatz bereits von Fachleuten annotiert wurde, wird außerdem gespeichert, ob die neue Annotation mit der Referenzklassifikation übereinstimmt. Jede Annotation ist der Nutzersitzung, in der sie erstellt wurde, zugeordnet. Diese verweist

wiederum auf die Spielenden selbst. Der oder die jeweilige Spieler oder Spielerin ist mit Vorname, Nachname und dem Benutzernamen, der beispielsweise in Ranglisten dargestellt wird, in der Tabelle *Player* gespeichert. Diese Namen werden auch zum Anmelden genutzt. Zusätzlich enthält jeder der Spielenden ein Feld, in das das Spiel einen Spielfortschritt abspeichern kann.

4.2.2. Programmstruktur

Der Server ist aus drei Schichten aufgebaut. Die oberste Schicht nimmt dabei die API-Anfragen des Spiels an, verarbeitet sie zum Teil vor und leitet sie an die zweite Schicht weiter. In dieser Schicht wird mit den Services der größte Teil der Geschäftslogik berechnet. Die Services lösen dabei komplexere Aufgaben und können mehrere Data-Access-Objects (DAOs) der datenbanknächsten Schicht nutzen. Die DAOs sind dabei die Schnittstelle zwischen Geschäftslogik und Datenbank und stellen verschiedene einfache JPQL-Anfragen zur Verfügung.
Die Services und DAOs sind den entsprechenden Tabellen in der Datenbank zugeordnet, wobei sie jeweils die Funktionen enthalten, die primär die Objekte ihrer Tabelle betreffen. Bei den Services gibt es zusätzlich noch eine Klasse, die verschiedene Methoden zur statistischen Auswertung enthält.
Wenn nun z. B. im Spiel neue Personen registriert werden sollen, geht die Anfrage zuerst an den Player-Controller. Hier wird geprüft, ob alle drei Namen angegebenen sind und ob sie die zulässige Länge einhalten. Die Anfrage wird dann weitergeleitet an den Player-Service, in dem mithilfe einer Anfrage an das Player-DAO geprüft wird, ob bereits Spielende mit dieser Namenskombination vorhanden sind. Wenn nicht, wird die neue Person in der Datenbank gespeichert und der Controller sendet den HTTP-Statuscode 200 OK zurück an das Spiel. Wenn bereits Spielende vorhanden sind oder die Namen zu lang sind, wird vom Controller entsprechend ein Fehlercode zurückgesendet.

4.2.3. API

Der Server stellt eine REST-API zur Verfügung, über die er mit dem Client kommuniziert. Die API ist dabei in drei Endpunkte gegliedert, die jeweils in einem eigenen Controller implementiert sind. Der erste Endpunkt ist für die Spielenden an sich zuständig. Er ist unter `/players` erreichbar und stellt Funktionen zum Erstellen eines neuen Spielenden, sowie zum Bearbeiten, das heißt zum Ändern der Namen bereit. Zusätzlich kontaktiert das Spiel diesen Endpunkt um einen neuen Spielstand zu speichern und sich anzumelden. Dazu sendet das Spiel die Namen an den Server, woraufhin eine neue Sitzung generiert wird. Das Spiel erhält daraufhin neben seinem gespeicherten Spielstand die ID der Session zurück und nutzt diese für alle weiteren Interaktionen mit dem Server während dieser Sitzung zur Identifizierung.

Der zweite Endpunkt der API ist wohl der am meisten kontaktierte und zentral im Annotationsprozess. Er ist unter `/image` zu erreichen, gibt Bilder zum annotieren aus und nimmt auch die Annotationen an. Normalerweise erhält das Spiel dabei keine Rückmeldung darüber, wie viele Bilder korrekt annotiert wurden. Der Server ermöglicht es aber durch Angabe des entsprechenden Parameters in der Anfrage, nur Bilder zu erhalten, die bereits annotiert sind, sowie eine Rückmeldung zu den Annotationen der Spielenden zu erhalten. Annotationen, zu denen eine Rückmeldung gesendet wird, werden allerdings nicht gespeichert, um Verfälschungen in den gesammelten Daten zu verhindern.
Der dritte Endpunkt ist unter `/Statistics` erreichbar und ermöglicht es den Annotierenden, Informationen über die Anzahl ihrer Annotationen und über den für sie berechneten Konfidenzwert (siehe 4.3.1) zu erhalten. Diese erhalten sie als tägliche Auswertung für jeden Tag des abgefragten Jahres. Alle Spielenden können sich außerdem hier die Ranglisten ausgeben lassen. Diese enthalten für alle Profile jeweils den Nutzernamen und die gesamte Anzahl an Annotationen, bzw. den durchschnittlichen Konfidenzwert. Die insgesamt im Server gesammelten Annotationen können hier ebenfalls abgerufen werden.
Durch die Nutzung von OpenAPI ist für die REST-API außerdem eine automatisch generierte Swagger-Dokumentation verfügbar.

4.3. Datenausgabe

4.3.1. Konfidenz-Berechnung

Durch den Ansatz als Citizen-Science-Projekt kann nicht darauf vertraut werden, dass alle Annotierenden tatsächlich korrekte Ergebnisse liefern. Um die einzelnen Annotationen der verschiedenen Personen zu einem möglichst korrekten Ergebnis zusammenzufassen ist es deshalb essenziell, Annotationen danach zu gewichten, wie wahrscheinlich sie korrekt sind. Dazu werden den Nutzenden in zufälligen Abständen neben zu annotierenden Bildern auch bereits bekannte Bilder als Tests ausgegeben und es wird geprüft, ob die Annotation der Nutzenden mit der tatsächlichen übereinstimmt. Für jede Annotation wird eine Konfidenz danach berechnet, wie gut er in den Tests abschneidet. Dazu werden für jede Annotation über die Tests in einem 60-Tage-Zeitraum der Anteil an korrekten Tests berechnet. Die Tests werden dabei mithilfe einer exponentiellen Abnahme anhand ihres zeitlichen Abstands von der Annotation gewichtet. Wenn in der letzten Zeit nicht genügend Tests vorliegen werden außerdem zusätzliche Tests ausgegeben, das geschieht z. B. auch immer am Anfang des Spiels. Wenn der 60-Tage-Zeitraum bereits vorüber ist und die Konfidenzen dann neu berechnet werden, wird bei den Annotationen, an deren Konfidenzen sich somit nichts mehr ändern kann, außerdem ein Flag gesetzt, durch den diese Annotationen bei zukünftigen Berechnungen nicht mehr beachtet werden, was Rechenaufwand

spart.

4.3.2. Struktur der Ausgabe

Die Daten werden über eine API-Anfrage an `/Statistics/completeStatistics` abgerufen. Daraufhin aktualisiert der Server zuerst alle Konfidenzwerte der einzelnen Annotationen. Anschließend werden die Annotationen für jedes Bild geladen und jeder Konfidenzwert wird in ein Gewicht übertragen, wobei ein Konfidenzwert von 0 einem Gewicht von -1, ein Konfidenzwert von 0.5 einem Gewicht von 0 und ein Konfidenzwert von 1 einem Gewicht von 1 entspricht. Diese Gewichte werden dann jeweils auf das entsprechende Feld in der akkumulierten Annotation addiert. Diese Ergebnisse werden dann noch auf einen Wert zwischen 0 und 1 normiert. Die finale Ausgabe erfolgt dann in Form einer JSON-Liste, die für jedes Bild ein Element enthält. Diese Bilder enthalten jeweils wiederum ihre ID, den Buchstaben der dargestellt ist, die Anzahl an Annotationen, die zu diesem Bild abgegeben wurden, sowie die Ergebnisse in Form des Anteils der jeder möglichen Klassifizierung gegeben wurde.

5. Diskussion

Ziel des Projekts war, ein Spiel zu entwickeln, in das die Annotation von Torahzeichen eingebettet ist, wodurch Freiwillige dazu motiviert werden sollen, wissenschaftlich nutzbare Daten zu sammeln. Dies erfolgt, indem jede wichtige Interaktion mit dem Spiel eine Runde des Annotierens vorraussetzt. Durch weitere Methoden wie die Annotationsguideline und ein statistisches Auswerten der Annotationen durch Tests wird ein wissenschaftlicher Wert der Daten garantiert. Da bisher allerdings noch keine Tests mit großen Zahlen an Annotierenden stattgefunden haben, kann noch nicht abschließend beurteilt werden, ob das Spiel tatsächlich für langfristige Motivation sorgt.

Aussagekraft der Daten

Die im Spiel gesammelten Daten sind umso aussagekräftiger, je konsequenter die Annotierenden sind. Durch die Konfidenzwerte können dauerhaft ungenaue und dauerhaft sehr genaue Annotatoren und Annotatorinnen gut gegeneinander gewichtet werden. Je größer allerdings die Varianzen bei deren Genauigkeiten sind, desto schlechter sind die vorhandenen Algorithmen geeignet, um daraus noch nutzbare Daten zu gewinnen. Außerdem kann der spielerische Charakter dazu führen, dass ein gewissenhaftes Annotieren für einige Anwender und Anwenderinnen in den Hintergrund tritt. Insbesondere Mechanismen wie Ranglisten und das Verstecken der Handlung hinter Annotationen könnten bewirken, dass die Annotationsrunden nur möglichst

schnell weggeklickt werden, wodurch keine nutzbaren Daten entstehen und korrekte Daten im Rauschen untergehen können. Auch der vorliegende Datensatz kann durch die ungleiche Verteilung der als Tests genutzten Annotationen zu Verschiebungen in den Konfidenzwerten führen. Diese möglichen Effekte können aber erst überprüft werden, nachdem das Spiel bereits in größerem Umfang genutzt und getestet wurde.

Funktionalität des Spiels

Das Spiel erfüllt seine grundlegende Funktion: Es stellt eine vollständig spielbare Point-and-Click-Umgebung bereit, in der die Annotation von Torahzeichen nicht als separate wissenschaftliche Aufgabe, sondern als zentrale Voraussetzung für den Spielfortschritt erscheint. Die Spielerinnen und Spieler können die verschiedenen Szenen erkunden, Gegenstände finden und verwenden, Hindernisse überwinden und schließlich das Ende der Handlung erreichen. Damit liegt ein abgeschlossenes Spielerlebnis mit einem klaren Ziel vor.

Da wichtige Interaktionen erst nach einer Annotationsrunde ausgeführt werden, sammelt jede Person, die im Spiel vorankommen möchte, zwangsläufig Daten für das Forschungsprojekt. Die Annotationsaufgabe wirkt innerhalb des eher ruhigen Point-and-Click-Spielprinzips weniger störend, als dies bei einem Spiel mit schnellen und kontinuierlichen Bewegungsabläufen der Fall wäre. Gleichzeitig erhalten die Annotationen durch das Öffnen von Truhen, das Aufheben von Gegenständen oder dem Freischalten neuer Wege einen unmittelbar erkennbaren Zweck innerhalb des Spiels.

Die Bedienung ist durch die Beschränkung auf wenige Interaktionsformen grundsätzlich leicht verständlich, auch die Navigation zwischen den Szenen und die Interaktion mit Objekten erfolgen größtenteils durch Mausklicks. Dies entspricht der Anforderung, das Spiel für eine möglichst große Gruppe von Freiwilligen zugänglich zu machen. Einschränkend ist jedoch, dass interagierbare Objekte nicht immer eindeutig als solche erkennbar sind. Teilweise müssen die Spielenden daher verschiedene Bereiche einer Szene ausprobieren, was zu Frustration führen kann.

Sicherheit

Da es sich um ein Citizen-Science-Projekt handelt, werden keine sensiblen Daten verarbeitet und die Sicherheitsanforderungen sind somit geringer als bei vielen anderen Webanwendungen. Dennoch ist die Sicherheit ein wichtiger Aspekt. Die Nutzenden melden sich mit einer Kombination aus Vorname, Nachname und Benutzernamen an, die als einfacher Text gesendet und verarbeitet werden. Das reicht für den Einsatzzweck aus, da keine personenbezogenen Daten gespeichert werden. Sollten in Zukunft aber weitere Inhalte hinzukommen, die solche Daten

verarbeiten, müsste eine sicherere Anmeldemethode eingesetzt werden. Die API ist derzeit nur über HTTP erreichbar. Auch das reicht im aktuellen Rahmen aus, da vor allem einfache Nutzung und Entwicklung im Zentrum steht. Für breitere Nutzung wäre es allerdings sinnvoll, auf HTTPS zu wechseln.

6. Fazit

Im Rahmen dieses Kooperationsprojekts wurde ein Programm entwickelt, das die Methoden Citizen-Science und Gamification kombiniert, um Bilder von Buchstaben aus historischen Torahrollen anhand von Dekorationen zu annotieren. Dazu wurde ein als Point-and-Click-Adventure konzipiertes webbasiertes Spiel als Nutzeroberfläche erstellt, in der die Annotationen abgegeben werden. Zur Verwaltung von Spielenden und Annotationen wurde mit Spring Boot ein Server programmiert, der verschiedene Statistiken anhand der abgegebenen Annotationen berechnet. Durch diese Statistiken können die Annotationen verschiedener Personen gewichtet werden und so, obwohl sie von Personen ohne Fachwissen stammen, für wissenschaftliche Forschung genutzt werden.
Neben der Funktionalität wurde durch einen modularen Aufbau von Server und Client eine Struktur entwickelt, die einfach um weitere Inhalte erweitert werden kann. Insbesondere die Spielwelt ist durch einen Aufbau mit verschiedenen Prefabs ohne Programmieraufwand leicht erweiterbar, was auch in Zukunft ein großes Interesse am Spiel und damit am Annotieren ermöglicht. Zusätzlich wurde eine Annotationsguideline geschrieben, die eine einheitliche Bewertung von Dekorationen unterstützt und dadurch die Vergleichbarkeit der Annotationen verbessert.
Insgesamt wurde das Ziel des Projekts erreicht, ein Spiel zu entwickeln, das durch Citizen-Science große Mengen an Annotationen von Torahzeichen sammeln kann.

7. Ausblick

7.1. Verfeinerung der Algorithmen bei der Konfidenz-Berechnung

Im Moment sind die meisten Algorithmen, die die Konfidenzen berechnen und die Spielenden untereinander abwägen, sehr einfach berechnete Durchschnitte. Das hat den Nachteil, dass die Ergebnisse womöglich nur schlecht zu verwenden sind, da die Ausgaben durch unerfahrene Annotatoren und Annotatorinnen stark schwanken könnten und somit womöglich nur wenig aussagekräftig sind. Sobald Daten über solche Schwankungen gesammelt sind sollten die Algorithmen deshalb entsprechend verbessert werden.
Besonders beim finalen Zusammenfassen werden im Moment alle Klassifikationen linear anhand ihrer Konfidenz-Werte gewichtet. Das hat allerdings den Nachteil, dass die Menge an unzuverlässigen Annotationen, die nötig ist um sehr gute Annotationen zu übertreffen nicht besonders hoch ist. Zusätzlich ist eventuell die Abgrenzung zwischen sehr zuverlässigen und eher zuverlässigen Annotierenden nicht stark genug. Dadurch ist die Annotation einer Person, die in allen Tests richtig antwortet, nur geringfügig stärker gewichtet als die Annotation einer Person, die in 90% der Fälle richtig lag. Um diese Probleme zu beheben wären verschiedene Ansätze denkbar. Eine Möglichkeit wäre, z. B. alle Annotationen, deren Konfidenzwerte unter 70% liegen komplett rauszulassen. Das würde die Qualität der Daten erhöhen, indem ein Einfluss durch Spielende, die nahezu zufällig tippen, verhindert werden würde. Gleichzeitig führt eine solche Grenze aber dazu, dass insgesamt weniger nutzbare Daten gesammelt werden. Es könnte außerdem sinnvoll sein, die Annotationen nicht linear zu gewichten, sondern sehr genaue Annotationen überproportional einzubeziehen. Das kann ebenfalls den Einfluss von ungenauen Annotationen verringern, in Fällen in denen sich auch sehr starke Annotierende mal irren oder schlicht aus Versehen das falsche Feld anklicken, würde eine solche Gewichtung aber auch den Fehler vergrößern.
Ein weiterer Bereich, in dem Verbesserungen möglich wären, ist die Berechnung der Konfidenzen. Es wäre hierbei denkbar, weitere Muster auszunutzen, um mit den vorhandenen Tests genauer an die tatsächlichen Sachverhalte zu kommen. Hier wäre es möglich, die Tests, die in der gleichen Sitzung gemacht wurden noch stärker zu gewichten. So könnten Schwankungen über verschiedene Tage abgefangen werden. Mit weiteren gesammelten Daten wären auch größere Muster denkbar. So könnte es sein, dass Annotierende an verschiedenen Wochentagen oder zu verschiedenen Uhrzeiten genauer sind als zu anderen. Auch systematische Änderungen über die Zeit einer Sitzung könnten vorliegen, das heißt Annotierende könnten zu Beginn einer Sitzung noch sehr genau arbeiten und dann mit der Zeit den Fokus verlieren. Diese Änderungen könnten die Konfidenzwerte wesentlich genauer machen, sie sind aber nur nach Auswerten sehr vieler gesammelter Daten denkbar.

Der dritte Bereich in dem noch etwas genauere Algorithmen eingesetzt werden könnten ist das Ausspielen von Tests. Momentan ist dabei zufällig jedes zehnte Bild ein Test, es sei denn es sind akut zu wenige Tests für eine Person vorhanden, woraufhin für alle Bilder Tests ausgewählt werden. Dieser Ansatz ist zwar gut in der Lage, ein allgemeines Bild zu erhalten, ihm fehlt aber eine sich ändernde Rate, um sich besser an die Spielperson anpassen zu können. Durch häufigeres Testen bei neuen Annotierenden könnte ein Profil der Genauigkeit erstellt werden, welches dann später nur noch mit wenigen Tests überprüft werden müsste. Das könnte helfen, anfänglich genauere Konfidenz-Werte festzustellen, während später dann mehr nutzbare Daten gesammelt werden könnten. Gleichzeitig würde ein solches System aber dazu führen, dass Annotierende, die mit der Zeit besser werden, weniger erkannt werden. Eine weitere Anpassung, die eventuell von Vorteil sein könnte ist dem Datensatz geschuldet. In ihm liegen bereits Annotationen und damit Tests vor allem für den Buchstaben Shin (ש) vor. Das führt dazu, dass auch für die Nutzenden meist nur Tests für den Buchstaben Shin gezeigt werden. Wenn nun Nutzende verschiedene Buchstaben unterschiedlich gut annotieren, wird das in der Auswertung nicht berücksichtigt. Es könnte dabei helfen, die Wahrscheinlichkeiten für alle Tests manuell anzugleichen, das hätte aber wiederum den Effekt, dass wegen der wenigen Tests, die für manche Buchstaben vorliegen, recht schnell Wiederholungen in den Tests auftreten könnten, was deren Aussagekraft massiv reduzieren würde.

7.2. Erweiterung des Spiels

Obwohl das Spiel in seinem aktuellen Zustand ein abgeschlossenes Spielerlebnis mit Anfang und Ende bietet, existieren zahlreiche Ansatzpunkte für zukünftige Erweiterungen. Die folgenden Abschnitte stellen weitere Ideen vor, geordnet von inhaltlichen Erweiterungen der Spielwelt hin zu Verbesserungen der Benutzererfahrung.
Die naheliegendste Erweiterung besteht im Hinzufügen neuer Spielabschnitte, also neuer Szenen, die die jetzige Story erweitern. Eine Idee, die das Spiel um eine komplett neue Landschaftsart erweitern würde: Ein Gebirgsabschnitt bestehend aus schneebedeckten Gipfeln und steilen, mit Geröll und Eis versperrten Pfaden, die sich um die hohen Gipfel der Berge winden und die einzigen Wege zum Ziel darstellen würden. Dies würde weitere Möglichkeiten für Szenen eröffnen, die damit zusammenhängen und das überwinden von Hindernissen erlauben. Dieses Prinzip, ein Hindernis, dessen Beseitigung an einem anderen Ort der Spielwelt verborgen liegt, entspricht der bereits etablierten Spielstruktur und ließe sich dadurch ohne grundlegende Änderungen am Code umsetzen. Im Wesentlichen wären neue Szenen, Items und Interaktionen als weitere Instanzen der vorhandenen Systeme zu erstellen, und daher die besten Möglichkeiten zur Erweiterung des Spieles.

Viele Spiellandschaften, wie der Dungeon oder der Friedhof, bieten Raum für zusätzliche Räume, Rätsel und Geheimnisse, wodurch sich der Umfang des Spiels erhöhen ließe, ohne neue Umgebungsideen erdenken zu müssen und ohne das Spiel unnötig weit von der Grundidee wegzuführen.
Eng verknüpft hiermit ist das Hinzufügen neuer interagierbarer Elemente in das Spiel. Jede neue Umgebung und jede neue Szene motiviert neue Gegenstände oder Materialien, die man braucht, um das jeweilige Hindernis zu überwinden: Bei den Gebirgsabschnitten könnte man etwa ein Feuerzeug mit oben genanntem Zweck oder einen Schneeumhang, ohne den man nicht in die nächste, stürmische Region vordringen kann, hinzufügen. Auch könnten im Spiel generell weitere Items verteilt werden, die man z. B. nur mit einer Zange aus einem Spalt ziehen kann oder nur mit einer Schaufel ausgraben kann. Die Logik hierfür existiert bereits in vollständiger Form im Spiel mit den Interactables und Items, die man jetzt schon innerhalb der Welt finden kann, also müsste auch hier im Code so gut wie nichts hinzugefügt werden, sondern nur die Grafiken und die Definition der Verwendungsorte der Items würden Arbeit verlangen.
Darüber hinaus ist langfristig ein Shop-System geplant, in dem die Spielenden die bis jetzt noch nicht nutzbaren Münzen verwenden können, um zum Beispiel einen großen Hinweis zum nächsten Item zu bekommen oder einen anderen Abschnitt des Spiels freizuschalten. Dadurch wären sie viel motivierter, alle Kisten, denen sie über den Weg laufen, zu öffnen, und würden dadurch mehr Zeichen annotieren. Auch würde dies den Anreiz steigern, das Spiel weiterzuspielen und die Welt weiter zu erkunden, wodurch ebenfalls die Zahl an annotierten Zeichen steigen würde.
Um bei den Spielerinnen und Spielern für mehr Interesse an der Welt zu sorgen, wäre auch eine Überarbeitung der Erzählung der Geschichte im Spiel denkbar. Aktuell erzählt das Spiel seine Geschichte primär über auffindbare Briefe und Pergamente. Dieses System ließe sich zunächst quantitativ ausbauen: Mehr verstreute Schriftstücke würden die Hintergrundgeschichte dichter erzählen und das Erkunden abgelegener Orte und unwichtig erscheinender Häuser belohnen. Ein qualitativer Ausbau wäre die Einführung ansprechbarer Charaktere, die jedoch nur in einfacher Form als stationäre Figuren ohne Bewegungs- oder Verhaltenslogik existieren würden, da sonst das Spiel unnötig verkompliziert würde. Auch unkonventionelle Gesprächspartner sind denkbar, etwa eine sprechende Tür im Dungeon, die die Spielenden arrogant anspricht und dabei kleine Teile der Geschichte der Welt, in der sich das Spiel befindet, offenbart.
Eine kurze Einführung am Beginn des Spiels, welche dessen Ausgangslage erklärt, würde den bisher eher abrupten Einstieg abmildern, die Motivation der Spielfigur etablieren und einen erzählerischen Rahmen geben. In Zukunft wäre hierfür sogar eine kurze Animation denkbar, wobei der Aufwand einer solchen Produktion in keinem Verhältnis zu den übrigen Erweiterungen steht und sie daher die niedrigste Priorität hat.
Ein weiterer wichtiger Punkt, um das Spiel visuell ansprechender zu gestalten, wäre eine generelle

Überarbeitung der Benutzeroberfläche (UI) des Spiels. Die aktuelle Benutzeroberfläche erfüllt zwar rein technisch ihren Zweck, besteht jedoch größtenteils aus schlichten grauen Rechtecken und Knöpfen. Eine Überarbeitung im Pixel-Art-Stil des Spiels würde die visuelle Kohärenz deutlich erhöhen und die UI als Teil der Spielwelt erscheinen lassen statt als Fremdkörper. Daneben würden eine übersichtlichere Anordnung der Elemente sowie das Hervorheben von Buttons beim Überfahren mit der Maus die Bedienbarkeit verbessern.
Verwandt mit der UI-Überarbeitung ist eine bessere Visualisierung von Spielprozessen. Ladebildschirme zwischen Szenenzuständen würden Übergänge flüssiger wirken lassen und könnten zugleich Raum für Hinweise oder Story-Fragmente bieten, da gerade nach Nutzung von Items die Änderung des Szenenhintergrundes abrupt erfolgt, dies oft sehr sichtbar ist und auf die Spielenden unnatürlich wirken kann. Noch wichtiger ist die Visualisierung möglicher Aktionen: Aktuell ist für die Spielenden nicht immer erkennbar, welche Objekte interagierbar sind, wenn man ein bestimmtes Item besitzt. Abhilfe schaffen könnten ein Glow- oder Outline-Effekt um interagierbare Objekte oder ein Highlighting beim Überfahren mit der Maus, zum Beispiel bei Geröll, das man mit einer Spitzhacke weghauen kann. Solche visuellen Rückmeldungen reduzieren Frustration durch erfolgloses Ausprobieren und machen das Spiel insbesondere für jüngere Personen zugänglicher.
Derzeit existiert das Spiel ausschließlich auf Englisch. Eine Sprachauswahl, zunächst mit Deutsch als zweiter Sprache, würde die Zugänglichkeit erheblich erweitern, gerade im Hinblick auf den Citizen-Science-Kontext des Projekts, in dem eine möglichst große und diverse Teilnehmerbasis wünschenswert ist. Die Umsetzung erfordert die Übersetzung aller Item-Namen, Objektbezeichnungen und insbesondere der Briefe, deren erzählerischer Ton bei der Übersetzung erhalten bleiben muss. Technisch würde sich hierfür ein zentrales Lokalisierungssystem anbieten, das Texte anhand von Schlüsseln aus einer Sprachdatei lädt, statt sie direkt im Code zu hinterlegen, was eine Umstellung wäre, die zugleich zukünftige Sprachen ohne weiteren strukturellen Aufwand ermöglichen, jedoch einiges an Zeit kosten würde.
Auch wenn das Spiel bereits jetzt alle Funktionen bereitstellt, die es zum Interagieren braucht, gibt es einige Punkte, die den Spielenden noch mehr Möglichkeiten geben würden. Eine Erweiterung die noch ins Spiel hinzugefügt werden könnte ist die Möglichkeit, den gespeicherten Spielstand zu löschen, um das Spiel zurückzusetzen und die Story erneut spielen zu können. Ebenfalls denkbar wäre, jedem Account mehrere Speicherslots zur Verfügung zu stellen. Insbesondere wenn das Spiel in seinem Umfang weiter zunimmt, könnte das die Zeit, die Annotatoren und Anotatorinnen im Spiel verbringt, und damit die Anzahl an Annotationen, die vorgenommen werden, deutlich erhöhen.
Eine weitere mögliche Erweiterung ist, das Spiel deutlich individualisierbarer zu machen. Hier ist zum einen die Möglichkeit, den Nutzernamen zu ändern, zu nennen. Zusätzlich wäre es möglich,

verschiedene Stile für die UI anzubieten oder die Anzahl an Annotationen pro Interaktion ändern zu können. Beides würde die Interaktion mit dem Spiel deutlich angenehmer gestalten.
Das Spiel wurde mit Fokus auf eine Webanwendung für PC entwickelt. Einige Funktionen benötigen eine Tastatur und die UI ist mit Blick auf einen Monitor im Querformat designt. Zusätzlich muss das Spiel bei jedem Aufruf der Seite neu vom Server geladen werden. In seinem Aufbau ist das Spiel dabei eigentlich gut geeignet, zwischendurch nochmal einige Bilder zu annotieren, da durch den Aufbau als Point-and-Click-Adenture keine schnellen und konstanten Eingaben nötig sind. Es könnte sich also lohnen, das Spiel auch für mobile Endgeräte zu optimieren.

8. Danksagung

Unser besonderer Dank gilt unseren Betreuern vom SCC (KIT), Volker Hartmann, Maximilian Inckmann und Lilly Osburg, die uns mit ihren fachlichen und praktischen Tipps tatkräftig unterstützt haben. Weiterhin danken wir Thomas Hermann und Anke Richert sowie allen weiteren Kursleitern des Hector-Seminars, die uns über die letzten Jahre immer wieder zum Nachdenken angeregt und herausgefordert haben. Abschließend gilt unser Dank Josephine und Hans-Werner Hector für die Förderung durch das Hector-Seminar.

9. Selbstständigkeitserklärung

Hiermit bestätigen wir, dass wir die vorliegende Arbeit selbständig mit Unterstützung durch unsere Betreuer verfasst und keine anderen als die angegebenen Hilfsmittel benutzt haben. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

(Frederik Haase) (Erik Dan Șenchiu)

10. Literatur

Literatur

[1] *Die Thora*. URL: https://www.ikg-m.de/kultus-und-religion/kreislauf-leben/die-thora/ (besucht am 04. 09. 2026).

[2] Nils Reiter. "Anleitung Zur Erstellung von Annotationsrichtlinien". In: *Reflektierte Algorithmische Textanalyse*. Hrsg. von Nils Reiter, Axel Pichler und Jonas Kuhn. Berlin, Boston: De Gruyter, 2020, S. 193–202. ISBN: 978-3-11-069397-3. DOI: doi:10.1515/9783110693973-009. URL: https://doi.org/10.1515/9783110693973-009 (besucht am 13. 07. 2026).

[3] *Was ist Java Spring Boot? | Microsoft Azure*. URL: https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-java-spring-boot (besucht am 12. 07. 2026).

[4] *The IoC Container :: Spring Framework*. URL: https://docs.spring.io/spring-framework/reference/core/beans.html (besucht am 12. 07. 2026).

[5] *Build Systems :: Spring Boot*. URL: https://docs.spring.io/spring-boot/reference/using/build-systems.html#using.build-systems.starters (besucht am 12. 07. 2026).

[6] *Gradle Plugin :: Spring Boot*. URL: https://docs.spring.io/spring-boot/gradle-plugin/index.html (besucht am 12. 07. 2026).

[7] *Einführung in Docker*. URL: https://www.hsbi.de/elearning/data/FH-Bielefeld/lm_data/lm_1359639/building/docker.html (besucht am 13. 07. 2026).

[8] *Git - What Is Git?* URL: https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F (besucht am 13. 07. 2026).

[9] Laura Frank, Germaine Götzelmann und Danah Tonne. "Towards a Dataset for Paleographic Details in Historical Torah Scrolls". In: *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 2: VISAPP*. 20th International Conference on Computer Vision Theory and Applications (VISAPP 2025), Porto, Portugal, 26.02.2025 – 28.02.2025. 2025, S. 926. ISBN: 978-989-758-728-3. DOI: 10.5220/0013173800003912. URL: https://publikationen.bibliothek.kit.edu/1000179797 (besucht am 13. 07. 2026).

A. Annotationsguideline

Annotationsguideline

Einleitung

Buchstaben in Torahrollen enthalten oftmals Dekorationen, beispielsweise in Form von Kringeln und Krönchen. Diese Dekorationen werden Tagin genannt. Ziel des Annotationsprojektes ist die Erarbeitung eines größeren Datensatzes mit annotierten Buchstaben auf Grundlage der dekorativen Elemente. Der Annotationsprozess soll im Rahmen eines Citizen-Science Projekts stattfinden. Hierbei handelt es sich um einen Crowdsourcing-Ansatz bei dem interessierte Personen am Annotationsprozess teilnehmen können und somit wissenschaftliche Arbeit befördern.
Im Folgenden Guideline findet sich eine Beschreibung der relevanten Annotationseinheiten und -kategorien. Weiterhin werden problematische Fälle zur Illustration der Eigenheiten des Datensatzes bezüglich der zu annotierenden Einheiten vorgestellt.

Annotationseinheiten

Das Annotationsprojekt befasst sich auf Ebene der Annotationseinheiten mit Buchstaben. Diese können dekorative Elemente enthalten. Die Buchstaben sind klar voneinander abgegrenzt. Es handelt sich, wie bei Torarollen vorgegeben, um hebräische Buchstaben. Die Buchstaben zeigen sich in handgeschriebener Ausführung.

Annotationskategorien

Der Fokus des Annotationsprojekts liegt auf der Frage nach der Verteilung und dem Einsatz von Buchstabendekorationen in primär historischen Torarollen.
Dementsprechend sind die zentralen Kategorien *dekoriert* und *undekoriert*. Bei ersterer Kategorie handelt es sich um eine Akkumulation mehrere Phänomene. Beispielsweise gibt es sowohl kringelartige wie auch kronenartige dekorative Elemente, die den Buchstaben hinzugefügt wurden. Da der Fokus auf einer allgemeinen Nutzung und Verteilung von Dekorationen liegt, wird im ersten hier beschriebenen Schritt nur zwischen den beiden Kategorie *dekoriert* und *undekoriert* unterschieden.
Dementsprechend sind folgende Kategorien gegeben

1. Dekoriert: Die Annotationseinheit weist über die Struktur des eigentlichen Buchstabens hinaus weitere Elemente dekorativer Natur auf.
2. Undekoriert: Die Annotationseinheit weist über die Struktur des eigentlichen Buchstabens hinaus keine weiteren Elemente dekorativer Natur auf; besteht also nur aus der Form des entsprechenden hebräischen Buchstabens.

Problematische Fälle

Aufgrund des handschriftlichen Natur des Datensatzes, variierender Digitalisierungsmethoden sowie Fehler bei der automatischen Buchstabenerkennung treten Herausforderungen auf. Diese sind in vier Kategorien zu unterteilen.

1. Schlechte Bildqualität: Durch den Digitalisierungs- und Weiterverarbeitungsprozess liegen einige Scans der Torarollen nur in geringer Qualität gemessen an der Auflösung vor. Dementsprechend sehen sich Annotator*innen mit Ambiguitäten aufgrund von verpixelter oder verschommener Buchstaben konfrontiert. Es können aufgrund des niedrigen Bildqualität Artefakte auftreten, die missinterpretiert werden können und somit fälschlicherweise als Dekoration annotiert werden.



mangelhafter Bildqualität: BSB Cod.hebr. 501. Undekoriert



mangelhafter Bildqualität: BSB Cod.hebr. 501. Dekoriert

2. Handschriftliche Natur und undeutliche Dekorationen: Die handschriftliche Natur der Buchstaben stellt Annotator*innen vor weitere Herausforderungen. Die Identifizierung der Dekorationen gestaltet sich gerade bei sehr dünnen oder sehr dicken Buchstabenteilen als kompliziert. Weiterhin kann die verwendete Schriftart der Gruppe der Serifenschreiften zugeordnet werden. Die entsprechenden Serife innerhalb des Annotationen können leicht mit dekorativen Elementen verwechselt werden.



Handschrift: BSB Cod.hebr. 489. Dekoriert

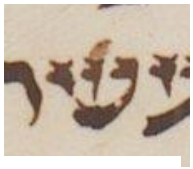


Handschriftt: BSB Cod.hebr. 489. Dekoriert

3. Verschmutzungen: Basierend auf der handschriftlichen Natur sowie Digitalisierungs- und Lagerungsbedingungen finden sich Artefakte aufgrund von Verschmutzungen auf einigen Bilddaten. Diese heben sich jedoch zumeist hinreichend von der Schriffarbe ab. Dementsprechend sind Verschmutzungen klar von Dekorationen zu trennen und treten eine Herausforderung dar, die nur marginal ins Gewicht fällt



Fehler in Daten: Cod. Hebr. 288. Undekoriert



Fehler in Daten: Hs. Or 14091. Dekoriert

4. Datensatzfehler: In seltenen Fällen sind die zu annotierenden Buchstaben nicht im Bildausschnitt sichtbar. Dies entsteht durch den automatisierten Buchstabenerkennungsprozess. Es treten beispielsweise Fälle auf, in denen nur Fragmente mehrerer Buchstaben gezeigt werden.

Praktische Umsetzung

Der Annotationsprozess soll im Rahmen eines Citizen Science Projekts durchgeführt werden. Annotator*innen bekommen mit dem Guideline eine grundlegende Orientierung an die Hand. Dieses Vorgehen wird mindestens in Teilen durch die gegebenen Niederschwelligkeit der einzelnen Kategorien, ermöglicht. Annotiert wird in einem eigens entwickelten Spiel im Sinne eines Gameification-Ansatzes. Das entwickelte Point-and-Click Adventure fordert Nutzer*innen an regelmäßigen Punkten im Spiel zu einer Annotationsaufgabe auf, die zunächst gelöst werden muss, bevor das

eigentliche Spiel fortgesetzt werden kann. Durch den spieleartigen Charakter des Annotationsprozesses soll dieser durch eine ansprechende Gestaltung attraktiver sowie zugänglicher für Laien sein, die Interesse an Forschung zu Torarollen haben-

B. Links

Der Quellcode zu Spiel und Server ist auf GitHub unter `https://github.com/Hector-Torah-Org/Tora-Game-Unity` bzw. `https://github.com/Hector-Torah-Org/Tora-Management-Spring` zu finden.
Das lauffähige Spiel ist unter `https://demo.scc.kit.edu/torah/` verfügbar.

C. Prompt zur Bildgenerierung

Prompt 1: ok, now, create the inside of the church. should be built somewhat like a normal church, with a pult at the far end, but also in the style seen in the image. 2d pixelart 1080p



KI-Antwort 1: Bild:



Prompt 2: very very good looking, but make it from the pov of someone inside the church, like seen in the image i sent attached to this message



Antwort 2: Bild:



Prompt 3: good, remove the door, make light shine through the windows, and add more pillars on the side walls

Antwort 3: Bild:

