Gewerbeschule Lörrach

Projektarbeit

C. Schmidt

Zeitraum  
23.11.2023 – 08.04.2025



**Dungeon Adventure  
Escape JOSCHUA**

Vorgelegt von:

Nico Brauer & Joschua Stammherr

[BrauerNic@mail.gws-loerrach.de](mailto:BrauerNic@mail.gws-loerrach.de)  
[StammherrJos@mail.gws-loerrach.de](mailto:StammherrJos@mail.gws-loerrach.de)

Abgabedatum: 08.04.2025

**Inhaltsverzeichnis**

1. Vorwort…………………………………………………………………………... 2

2. Konzeptionierung……………………………………………………………….. 3

2.1. Spieldetails…………………………………………………………………. 3

2.2. Aufgabenteilung……………………………………………………………. 3

2.3. Programmierung…………………………………………………………… 3

2.4. Grafikdesign………………………………………………………………... 3

3. Spielgeschehen…………………………………………………………………. 4

3.1. Allgemein…………………………………………………………………… 4

3.2. Levelsystem………………………………………………………………... 4

3.3. Krüge………………………………………………………………………... 4

3.4. Darkroom…………………………………………………………………… 5

4. Komplikationen & Lösungen…………………………………………………… 7

4.1 Programmiertechnisch…………………………………………………….. 7

4.2 Graphisch…………………………………………………………………… 8

5. Fazit

6. Quellenverzeichnis

7. Anlagen

**Vorwort**

Die vorliegende Dokumentation entstand im Rahmen der Abschlussprüfung für die Fachhochschulreife am Berufskolleg für Informationstechnik der Gewerbeschule Lörrach.

Die gesamte Dokumentation wurde in der dritten Person verfasst, um es dem Leser zu vereinfachen, die Aufgaben und Perspektiven beider Teammitglieder zu verstehen.

Bevor die Entscheidung zwischen einer Cloud mittels eines Raspberry Pi und einem in Java programmierten Videospiel getroffen wurde, war es notwendig zwischen Einzel- oder Partnerarbeit zu wählen.  
Aufgrund von Nicos und Joschuas bisheriger Programmiererfahrung und der mangelnden Kenntnisse in Datenbank- und Serversystemen, entschieden sie sich für die Programmierung eines Spieles, entwickelt mit der objektorientierten Programmiersprache Java.  
Zu der Zusammenarbeit, anstatt der Einzelarbeit, entschieden sie sich, da sie einerseits sehr gut miteinander auskommen, aber sich auch gegenseitig zurechtweisen und unterordnen können.

Das Projekt ist ein 2D Dungeon Adventure, in dem der Spieler zwei Rätsel lösen muss. Beide Rätsel finden in eigenen Räumen statt, aus denen, durch Lösen des Rätsels, ein Signal im Hauptraum aktiviert wird. Die Grafiken dieses Spieles wurden per Pixel Art Technik selbst gezeichnet. Durch das Lösen aller Rätsel, kann das Spiel beendet werden.

**Konzeptionierung**

**Spieldetails**

Anfangs trafen sich Nico und Joschua ein bis zwei Mal pro Woche, um die Einzelheiten des Spieles auszuarbeiten. Sie schrieben vieles auf von dem einiges dann doch wieder verworfen wurde, bis ein Spielkonzept zustande kam, welches beiden gefiel. Auch außerhalb der Treffen tauschten sie sich mittels WhatsApp aus, wodurch manche Ideen sofortig verworfen oder weiterentwickelt wurden, um sie bei den Treffen zu Ende zu führen. Als die Details fertig besprochen wurden, legten sie die Aufgabenteilung fest.

**Aufgabenteilung**

Wird sich für eine Partnerarbeit entschieden, ist es gerade bei der Entwicklung eines Spieles wichtig, eine sinnvolle Aufgabenteilung festzulegen. Dies vermeidet die gegenseitige Arbeitsbehinderung, durch das Arbeiten an derselben Aufgabe. Da sich schon in der Ideenfindung herausstellte, dass Joschua sehr kreativ und Nico eher programmierbegabt ist, entschieden sie, dass Joschua sich um das Erstellen der Grafiken und Nico um das Programmieren kümmern wird.

**Programmierung**

Bevor der Programmierprozess begann, stand die Idee im Raum, mit  
UML-Klassendiagrammen zu starten, um strukturierter arbeiten zu können. Allerdings äußerte sich Nico dagegen, da er bezüglich der Attribute und Operationen eher spontan arbeite. Die genannten Klassendiagramme sind dennoch in den Anlagen vorzufinden.  
Um immer mit dem Fortschritt der letzten Sitzung vertraut zu sein, führte Nico ein Ergebnisprotokoll zu jeder Arbeit am Projekt, welches ebenfalls Beschwernisse und offene Fragen umfasst.

**Grafikdesign**

Joschua arbeitet mittels eines iPad und eines Apple Pencil. Er brachte sich selbst die Pixel Art Technik bei, um die Grafiken für das Spiel zu entwickeln. Parallel notierte sich auch Joschua auf welche Komplikationen er stieß.  
Um die Grafiken in das Spiel zu implementieren, sendete Joschua diese nach Fertigstellung per AirDrop (Apples Drahtlose Übertragungstechnik) an Nicos iPhone, welche Nico dann per Mail an sich selbst sendete, um sie Schlussendlich am Rechner in das Spiel zu implementieren.

**Spielgeschehen**

**Allgemein**

Klassisch für ein Dungeon Adventure, spielt sich das Geschehen in unterirdischen Höhlen ab. Der Spieler startet im Hauptraum, von dem er die Möglichkeit hat, jeden weiteren Raum zu betreten. Es existieren zwei weitere Räume, welche jeweils ein Rätsel enthalten. Ein Raum enthält die Aufgabe durch ein Labyrinth zu finden, während nur ein kleiner Radius um den Spieler sichtbar ist. Innerhalb dieses Labyrinths befindet sich ein Schalter, welcher das Rätsel löst und damit dem Raum wieder sichtbar macht. Im zweiten Raum stellt der Spieler zwei von vier Krügen auf zwei Druckplatten. Sind beide Druckplatten aktiviert, ist das Rätsel gelöst. Allerdings wird ihm hierbei der Weg von einem NPC (=Non-Playable Character) erschwert, der sich beständig in eine zufällige Richtung bewegt. Kommt der Spieler mit dem NPC in Berührung, wird der Spieler wieder an seine Startposition, innerhalb des Raumes, gesetzt. Der NPC wird mittels eines Bildes von Joschua dargestellt, woraus sich ebenfalls der Spieltitel ableitet. Jeder Raum kann ohne Voraussetzung betreten werden, was in einer flexiblen Rheinfolgenauswahl resultiert. Jedes abgeschlossene Rätsel lässt ein Signal im Hauptraum aufleuchten. Durch Lösen aller Rätsel wird eine Tür entriegelt, die in die Freiheit führt.

**Levelsystem**

Neben dem Bild, welches die Map zeigt, beinhaltet jedes Level auch ein zweidimensionales Array, dass unbegehbare Bereiche, aber auch Türen kennzeichnet. Nach jeder Spielerbewegung prüft das Programm die aktuelle Position des Spielers. Ist diese Position gleich der Position einer gekennzeichneten Tür im Map-Array, wird jeder graphische Inhalt gelöscht, ein neues Bild so wie Array hinterlegt, neue Spielerkoordinaten festgelegt und anschließend das neue Level gezeichnet.

**Krüge**

Die Krüge im Druckplattenraum sind kein fester Bestandteil der Map. Sie sind einzelne Objekte, die über die Map gezeichnet werden und, wie beispielsweise die Spielfigur, eigene Koordinaten und ID’s besitzen. Befindet sich der Spieler nun auf dem Feld eines Kruges, was ebenfalls mittels des Map-Arrays ermittelt wird, bekommt er die Möglichkeit diesen, durch betätigen der „E“ Tasten, aufzuheben. Der Krug wird graphisch entfernt und die Spielfigur ändert ihr Bild, zu einer krugtragenden Figur. Wird die „E“ Taste nun erneut betätigt, wird der Krug an der Position gezeichnet, an der sich der Spieler in diesem Moment befindet und das Bild der Spielfigur wird zurückgesetzt. Auch im Map-Array wird beim Aufheben und Ablegen eines Kruges der Array-Wert an der entsprechenden Position angepasst.

**Darkroom**

Der Darkroom wurde realisiert in dem eine Schwarze Fläche mit einer mittigen Aussparung, die doppelt so groß ist wie der Raum, über den Spieler gezeichnet wird. Die Aussparung liegt hierbei über dem Spieler, sodass nur ein kleiner Radius um den Spieler sichtbar und der Rest der Map verdeckt ist. Aufgrund der Größe der schwarzen Fläche bleibt die Map auch dann vollständig bedeckt, wenn der Spieler sich am Rand des Raumes positioniert hat.

**Komplikationen & Lösungen**

**Programmiertechnisch**

Nico hat zwar gute Java Kenntnisse und stieß daher selten auf Syntaxbezogene Hürden, jedoch mangelt es ihm noch an reeller Projekterfahrung, weshalb es ihm nicht leichtfiel, ein funktionierendes Levelsystem zu entwerfen. Zuerst sollte bei Durchschreiten einer Tür eine Variable hochgezählt und je nach Wert das nächste Level geladen werden. Da das vorliegende Projekt allerdings nicht aus klassischen Leveln, sondern Räumen besteht, die immer wieder betreten und verlassen werden können, war eine andere Lösung von Nöten. Nach zwei Wöchigem überlegen und ausprobieren, fand Nico eine Lösung. Die Maps bestehen zwar aus Bildern, doch im Hintergrund liest das Programm ein zweidimensionales Integer-Array aus, dass die Positionen von Mauern, Druckplatten oder ähnlichem vorgibt. Jede Tür hat in diesem Array einen eigenen Wert, der einem Raum zugewiesen ist. So wird, abhängig von der betretenen Tür, der entsprechende Raum geladen.

Weiter trat ein Problem mit dem NPC auf. Da eine zufällige Richtungsauswahl ebenfalls in einer nicht begehbaren Richtung resultieren konnte, zeigte der NPC gelegentlich keine Reaktion. Bevor Nico nach einer Lösung suchte, besprach er die Situation mit Joschua und sie überlegten, ob eine Lösung tatsächlich nötig sei. Nico und Joschua entschieden sich diesen Spielfehler zu beheben, also verschachtelte Nico den Code zur Richtungsauswahl in eine Kopfgesteuerte Schleife, die endet sobald eine verfügbare Richtung gefunden wurde.

Nun wurde im praktischen Programmierunterricht das Thema Vererbung vermittelt. Die Vererbung dient der Codereduzierung und Übersichtlichkeit, indem mehrere Klassen auf Attribute und Operationen der sogenannten Oberklasse zugreifen können. Ein Programm kann somit zeiteffizient fertiggestellt werden. Jedoch befand sich das Projekt bereits in einem fortgeschrittenen Stadium, sodass die Vererbungstechnik, abgesehen von der Übersichtlichkeit, zunächst keinen direkten Nutzen aufgewiesen hätte. Doch aufgrund der verbesserten Übersichtlichkeit und des langfristigen Nutzens, entschied sich Nico, die Vererbung dennoch einzubinden. Hierzu wurde eine neue Oberklasse definiert, und die entsprechenden Unterklassen bestimmt. Etwa 140 Zeilen redundanter Code konnten zu diesem Zeitpunkt entfernt werden, da ihre Funktionalität nun in der Oberklasse bereitgestellt wurde.

Auch verstand Nico nicht auf Anhieb, wie die Dunkelheit des Darkrooms in das Spiel implementiert werden solle. Hier schuf erneut der Austausch mit Joschua und eines weiteren Klassenmitglieds Abhilfe. Ein schwarzes Bild, welches eine mittige Aussparung besitzt, wird über den Spieler gezeichnet und folgt jedem seiner Schritte. Das Bild muss hierfür größer als das Spielfenster sein und die Aussparung über dem Spieler liegen.

**Graphisch**

* Zuerst nicht auf Raster angepasst
* Programm abgestürzt, Fortschritt nicht gespeichert