Projektdokumentation – Textbasiertes Rollenspiel in C

1. Einleitung

Im Rahmen der Fallstudie im Modul **Programmiertechnik** entwickelten wir ein textbasiertes Rollenspiel in der Programmiersprache **C**. Ziel war es, ein einfaches Konsolenspiel zu programmieren, das verschiedene Funktionen eines Rollenspiels abbildet. Unsere Gruppe bestand aus **Sven**, **Gian** und **Cristian**. Die Fallstudie diente als praktische Übung zur Vorbereitung auf die spätere Praxisarbeit und zur Vertiefung zentraler Programmierkonzepte.

2. Zielsetzung

Das Ziel des Projekts war es, ein modular aufgebautes Spiel zu entwickeln, bei dem der Spieler mit einem benutzerdefinierten Charakter gegen verschiedene Monster kämpft. Dabei sollten folgende Kernfunktionen umgesetzt werden:

- Erstellung eines Charakters mit festgelegter Gesundheit und Angriffswert
- Auswahl eines Monsters aus einer Liste
- Auswahl der Angriffsart (physisch, magisch, Fernkampf, Gift)
- Rundenbasierter Kampf zwischen Spieler und Monster
- Spielende bei Tod des Spielers oder nach dem Besiegen aller Monster

3. Projektstruktur

Das Projekt wurde in mehrere Module aufgeteilt, die jeweils in .c- und .h-Dateien unterteilt sind:

Modul	Beschreibung
main.c	Einstiegspunkt, Spielstart mit Aufruf von startGame()
character.c/h	Erstellung und Anzeige des Spielercharakters
monster.c/h	Definition und Anzeige von Monstern
attack.c/h	Definition der Angriffstypen und deren Auswirkung auf Gegner
game.c/h	Spiellogik mit Monsterauswahl, Angriffssteuerung und Ablaufkontrolle

4. Spiellogik und Umsetzung

Beim Spielstart gibt der Spieler seinen Namen ein. Der Charakter startet mit **100 Lebenspunkten** und **20 Angriffspunkten**.

Die Monster heißen **Spinne**, **Golem** und **Drache**. Jedes Monster hat eigene Lebens- und Angriffswerte und kann individuell ausgewählt werden.

Der Spieler kann zwischen vier Angriffsarten wählen:

- **Physisch**: Schaden = Angriffskraft
- **Magisch**: Schaden = Angriffskraft × 1.5
- **Fernkampf**: Schaden = Angriffskraft × 0.9
- **Gift**: Schaden = Angriffskraft × 0.2 – bei zweimaligem Einsatz wird das Monster betäubt (Betäubt: Das Monster kann nicht angreifen für 1 Runde)

Die **Giftmechanik** wurde erweitert: Nach zwei erfolgreichen Giftangriffen wird das Monster für eine Runde betäubt.

Der Spieler wählt ein Monster und eine Angriffsart. Danach greift er an. Solange das Monster lebt, greift es zurück an. Der Kampf endet, wenn entweder das Monster oder der Spieler besiegt ist. Danach kann ein neues Monster gewählt werden, bis entweder alle Monster tot sind oder der Spieler keine Lebenspunkte mehr besitzt.

5. Technische Herausforderungen

Die vom Dozenten bereitgestellten Dateien waren teilweise unvollständig oder fehlerhaft. Diese mussten zuerst überarbeitet werden. Das Zusammenspiel der verschiedenen .c- und .h-Dateien erforderte sorgfältige Organisation und ein sauberes Header-Management.

Die Umsetzung der Zustände, insbesondere für die Giftmechanik mit Betäubung, war technisch anspruchsvoll und erforderte zusätzliche Kontrollstrukturen sowie das Arbeiten mit Bitfeldern.

Die Auswahlfunktionen für Monster und Angriffe mussten benutzerfreundlich gestaltet werden, um Verwirrung bei der Bedienung zu vermeiden.

6. Projektverlauf (Journal)

Tag	Tätigkeit
Tag 1	Spielstruktur analysiert, fehlerhafte Dateien überarbeitet
Tag 2	Einzelne C-Dateien zusammengeführt, Spiel erstmals lauffähig getestet
Tag 3	Spielmechanik verbessert, sprechende Monsternamen eingeführt
Tag 4	Giftmechanik und Status-Effekte erweitert, Spielabschluss umgesetzt

7. Fazit und Reflexion

Das Projekt zur Entwicklung eines textbasierten Rollenspiels in C war für unser gesamtes Team eine intensive, aber sehr lehrreiche Erfahrung. Im Laufe der vier Projekttage konnten wir nicht nur unser Wissen in der Programmiersprache C deutlich erweitern, sondern auch wichtige Kompetenzen im Bereich der Zusammenarbeit, Problemlösung und Projektstrukturierung festigen. Obwohl einige Grundlagen durch die Vorgaben des Dozenten bereits vorgegeben waren, mussten viele Teile zunächst angepasst, erweitert oder komplett überarbeitet werden, bevor das Spiel vollständig funktionierte.

Im Mittelpunkt des Projekts stand die saubere Strukturierung eines C-Programms in mehrere Module mit klar definierten Verantwortlichkeiten. Dabei haben wir viel über die Verwendung von Header-Dateien, über die Trennung von Spiellogik und Darstellung sowie über die Kommunikation zwischen den Modulen gelernt. Zusätzlich wurden wir mit Konzepten wie Enums, Bitfeldern und der Implementierung von Spielmechaniken wie Zustandsveränderungen (z. B. Vergiftung) konfrontiert, die wir erfolgreich umsetzen konnten.

Cristian

Für mich war dieses Projekt eine spannende Gelegenheit, das bisher Gelernte aus dem Unterricht in die Praxis umzusetzen. Besonders herausfordernd, aber auch sehr interessant, war die Steuerung der Kampflogik – insbesondere die Schleifen und Bedingungen, mit denen der Spielverlauf koordiniert wurde. Ich habe durch dieses Projekt ein viel besseres Verständnis für verschachtelte Logik und die Bedeutung von sauberer Struktur in größeren Programmen gewonnen. Ein Highlight für mich war die Umsetzung der Giftmechanik mit dem Stun-Effekt nach zwei Treffern, da hier nicht nur mathematische Logik, sondern auch Zustandsspeicherung über Runden hinweg gefragt war. Ich habe erkannt, wie komplex schon einfache Spiellogik sein kann, wenn man sie robust umsetzen will.

Gian

Ich konnte durch dieses Projekt mein Grundverständnis der Programmiersprache C deutlich vertiefen. Am Anfang fiel es mir noch schwer, die Struktur eines mehrteiligen Programms zu überblicken – besonders, wenn mehrere Dateien miteinander kommunizieren müssen. Durch die Arbeit an den Modulen, insbesondere an der Monsterauswahl und den Benutzereingaben, habe ich gelernt, wie wichtig eine durchdachte Funktionsaufteilung und ein klarer Datenfluss im Programm sind. Auch der Umgang mit Header-Dateien und Funktionsprototypen war zu Beginn ungewohnt, wurde mir mit der Zeit aber immer klarer. Besonders stolz bin ich darauf, dass wir es geschafft haben, das Spiel vollständig zum Laufen zu bringen – inklusive Monsterauswahl mit echten Namen und einer benutzerfreundlichen Menüführung.

Sven

Da ich bereits Erfahrung mit der Programmiersprache C mitbrachte, konnte ich mein Wissen gezielt in das Projekt einbringen. Besonders bei der technischen Umsetzung der komplexeren Teile wie der Angriffsfunktionen, der Schadensberechnungen und der Modularisierung des Codes konnte ich Verantwortung übernehmen. Für mich war es wichtig, dass der Code gut strukturiert, klar kommentiert und einfach erweiterbar bleibt – denn gerade in C kann ein unübersichtlicher Aufbau schnell zu Fehlern führen, die nur schwer zu finden sind. Eine besondere Herausforderung war die saubere Umsetzung der Giftmechanik mit Bitfeldern und internen Statusabfragen. Auch das Zusammenspiel von attack.c, game.c und der Benutzerinteraktion erforderte ein gutes Verständnis für Schnittstellen zwischen Modulen. Obwohl ich viele Konzepte schon kannte, konnte ich durch die Anwendung im Spielkontext mein Verständnis für Zustandsverwaltung, Fehlerbehandlung und Benutzerführung nochmals vertiefen.