Zeitplanung

Programmierprojekt (DarkIT)

DataKeeper

Ein 2D-Survival-Spiel mit 3 Levels und Bosskampf

Student:	Abdulkarim Bashir Termanini
Matrikelnummer:	1813326
Team:	6 Personen
Studiengang:	Angewandte Informatik
Fakultät:	Fakultät IV - Wirtschaft und Informatik
Hochschule:	Hochschule Hannover

Hannover, 31. Oktober 2025

Inhaltsverzeichnis

1 Projektübersicht

1.1 Projektziele

DataKeeper ist ein Java-basiertes 2D-Survival-Spiel. Ziel ist es, den Spieler durch drei Standard-Levels bis zu einem Boss-Level zu führen. Der Fokus liegt auf flüssigem Gameplay (60 FPS), klarer Progression, guter Spielbalance und stabilen Builds (JAR-Release).

1.2 Projektumfang

- Gameplay: Überlebe bis Timerende; Gegner-Spawning, Portal zum Fortschritt
- Level-System: 3 Levels + Boss-Level, steigende Schwierigkeit
- Kampf: Boden- und Luftangriffe, Slide/Dash, kurze iFrames
- UI/UX: HUD, Hauptmenü, Levelabschluss-, Sieg- und GameOver-Screens
- Audio: Hintergrundmusik (Menu/Level/Boss), SFX, Combat-Layer
- Persistenz: SaveManager für Sitzungsstatistiken

1.3 Technologie-Stack

- Programmiersprache: Java 17+
- GUI-Framework: Swing
- Build/Run: Shell-Skripte, JAR Packaging
- Versionskontrolle: Git/GitHub
- UML/Docs: PlantUML, Javadoc

1.4 Zeitrahmen

12 Wochen (ca. 3 Monate).

2 Schritte zur Erstellung des Zeitplans

2.1 1. Projektziele definieren

Ziele und Anforderungen wurden für Gameplay, Levelprogression, UI und Audio festgelegt.

2.2 2. Aufgaben identifizieren

Aufgaben in Phasen gegliedert: Initiierung, Analyse, Design, Implementierung (3 Phasen), Testing/Optimierung, Dokumentation, Abschluss.

2.3 3. Aufgaben priorisieren

Design vor Implementierung; Kernfunktionen vor Nice-to-Haves; Testing parallel; Dokumentation kontinuierlich.

2.4 4. Ressourcen zuweisen

Team (6 Personen) mit Rollen: Gameplay, Entities, Level, UI/UX, Audio, Projektleitung.

2.5 5. Zeitaufwand schätzen

Eine Woche pro Phase; Pufferzeit pro Woche ca. 10%.

2.6 6. Meilensteine festlegen

Meilensteine decken die Pflichttermine (20.10, 27.10, 31.10, 19.12) sowie wöchentliche Ziele ab.

3 Detaillierter Zeitplan

Woche	Aktivität	Aufgaben	Verantwortlich
1	Projektinitiierung	Kick-off, Ziele fixieren; Technologien auswählen; Repository/Build-Skripte anlegen	Projektleitung
2	Anforderungsanalyse	Use-Case-Recherche, Spielregeln und Progression definieren; GUI-Entwürfe skizzieren	Gameplay, UI
3	Anwendungsfälle definieren	Use-Case-Diagramme; Szenarien: Kämpfen, Spawning, Portal; UI-Flows	Gameplay, UI
4	Klassenmodell entwerfen	Klassendiagramm (Player, Enemy, Boss, LevelManager, SpawnManager, HUD, SoundManager, Portal)	Entities, Level
5	Architekturentwurf	Package-Struktur, Game-Loop, Event-Flows; Asset-Pfade; Audio-Konzept	Projektleitung, Audio
6	Implementierung (Phase 1)	GameCharacter/Player/Constants; GamePanel Grundgerüst; Input-System	Gameplay, Projektleitung
7	Implementierung (Phase 2)	Enemy, SpawnManager, LevelManager; HUD, Hauptmenü, Game States	Entities, Level, UI
8	Implementierung (Phase 3)	Boss-Klasse, Portalfluss, LevelConfig; AnimationManager; SoundManager	Entities, Level, Audio
9	Sound & UI Polish	Musik/SFX, Combat-Layer; UI-Finishing, Hintergründe; kleine Effekte	Audio, UI
10	Testing & Bugfixing	Unit-/Integrationstests; Performance-Optimierung; Speicherlecks prüfen	Alle
11	Dokumentation	Javadoc, README, UML finalisieren; Nutzerhinweise	Alle
12	Projektabschluss	JAR erstellen; finaler Test; Abgabe und Präsentationsvorbereitung	Projektleitung

4 Benötigte Ressourcen

4.1 Software-Ressourcen

IntelliJ/VS Code, Java 17+, Git/GitHub, PlantUML, Image/Sound Tools, JDK/JRE.

4.2 Fachliche Ressourcen

Java/Swing, OOP, Basis Audio (JavaSound), 2D-Animation/Sprites, einfache KI/Physik.

4.3 Materialien

Sprites (Player/Enemy/Boss), Hintergründe, Audio-Dateien, UI-Icons.

5 Risikomanagement

5.1 Kurz und verständlich (Stand: 29.10.2025)

Wir halten die Risiken bewusst einfach. Zu jedem Risiko steht kurz, was passieren kann und was wir konkret tun.

- **Boss zu schwer oder buggy** — Auswirkung: Spielspaß leidet, Zeitverzug. Maßnahme: Mit einfachen Mustern starten, jede Änderung testen, notfalls Muster vereinfachen.
- **Leistung (FPS) fällt** — Auswirkung: Ruckeln. Maßnahme: Gegneranzahl begrenzen, leichte Effekte nutzen, früh testen und nachbessern.
- **Zeitplan rutscht** — Auswirkung: Manche Features fehlen. Maßnahme: An MVP festhalten, Extras verschieben, wöchentliche Puffer nutzen.
- **Merge-/Integrationskonflikte** — Auswirkung: Build bricht. Maßnahme: Kleine Änderungen, täglich mergen, Feature-Flags nutzen.
- **Assets/Lizenzen unklar** — Auswirkung: Austausch nötig. Maßnahme: Nur eigene oder frei nutzbare (z. B. CC0) verwenden, Quellen dokumentieren.
- **Kein Sound im Release** — Auswirkung: Spiel wirkt leer. Maßnahme: Ressourcenpfade im JAR testen, einheitliches Laden der Dateien.
- **Speicherverbrauch steigt** — Auswirkung: Lags oder Absturz. Maßnahme: Ressourcen freigeben, wiederverwenden (Pooling), einfache Leak-Checks.
- **Träge Eingaben** — Auswirkung: Schlechte Steuerung. Maßnahme: Eingaben vom Rendern trennen, kurze Tests mit Spielern.

5.2 Puffer und Eskalation

Wöchentliche Kontrolle: Wenn ein Risiko auftritt, vereinfachen wir sofort auf MVP und priorisieren Stabilität.

6 Meilensteine und Erfolgskriterien

6.1 Haupt-Meilensteine (Status Stand: 29.10.2025)

- **20.10: Analyse + GUI-Design** — erledigt
- **27.10: Klassenmodell** — erledigt
- **31.10: Zeitplanung** — in Arbeit (Abnahme ausstehend)
- **19.12: Implementierung/Tests** — geplant

6.2 Erfolgskriterien (Zielzustand, noch nicht erfüllt)

Das Projekt gilt als erfolgreich, wenn die folgenden Ziele erreicht sind. Hinweis: zum Stichtag 29.10.2025 sind diese *geplant*, nicht abgeschlossen.

- Spiel läuft stabil bei angestrebten 60 FPS auf Standard-Hardware — *geplant*
- 4 spielbare Level (3 + Boss) inkl. Portalsystem — *geplant*
- Audio vollständig integriert (Musik, SFX, Combat-Layer) — *geplant*
- JAR-Release lauffähig (Assets via getResource) — *geplant*
- Vollständige Dokumentation (Javadoc, README, UML) — *geplant*
- Termingerechte Abgabe — *geplant*

7 Überwachung und Berichtsmethoden

Wöchentliche Fortschrittskontrolle, Git-Commits mit sinnvollen Messages, Issue-Board; Anpassungen bei technischen Risiken; MVP-Ansatz bei Abweichungen.

8 Fazit

Der Plan bietet eine klare Struktur bis zum 19. Dezember. Mit MVP-Fokus, regelmäßiger Integration und gezielter Optimierung ist eine termingerechte Abgabe realistisch.