

# Monster Kampf-Simulator

## GDD Monster Fighter

### 1. Projektübersicht

**Titel:** Monsterkampf-Simulator

**Genre:** Rundenbasiertes 1v1 Konsolen-Kampfspiel

**Zielplattform:** PC (Konsole)

**Ziel:** Uni-Projekt

**USP:**

- Vier unterschiedliche Monster-Klassen
  - Aktive & passive Skills
  - Buffs, Debuffs, DOT, Regeneration
  - Damage Pipeline (mehrschrittige Berechnung)
- 

### 2. Core Gameplay Loop

1. Spieler wählt ein Monster
  2. Gegner-Monster wird generiert
  3. Beide Monster spawnen & passive Fähigkeiten werden aktiviert
  4. Runde → Spieler wählt eine Aktion (Skill / Attack)
  5. Angriff → Damage Pipeline → Status Effekte
  6. Gegner führt AI-Entscheidung aus
  7. Cooldowns & DOTs werden abgearbeitet
  8. Ende der Runde → Sieg/Niederlage prüfen
-

### 3. Spielmechaniken

- **Rundenbasiertes Kampfsystem**
  - **AP-basierter Schaden:** Skills multiplizieren AP
  - **Cooldown-System:** Skills sind nicht immer verfügbar
  - **Status-Effekte:** Buffs/DeBuffs/DOTs
  - **Turn-Order:** Via Speed
  - **StatPoint-System:** Spieler kann Monster verbessern
  - **Level-Up-System:** Werte skalieren pro Stufe
- 

### 4. Monstersystem (Player + Enemy)

#### Monsterklassen:

- **Goblin** – schnell, aggressiv, Poison & Stone
- **Orc** – sehr starke AP, Buff & Fear-Passive
- **Troll** – Tank, Regeneration-Passive
- **Slime** – Elementar-Angriffe, Damage Absorb-Passive

#### Monsterdaten:

- **Meta-Werte:**
  - MaxHP
  - CurrentHP
  - AP
  - DP
  - Speed
- **Resistenzen:**
  - Fire
  - Water
  - Physical
  - Poison

- **SkillPackage:**
    - PassiveSkill
    - ActiveSkills
    - EventPassives
- 

## 5. Skills & Status Effects

### Skill-Arten

- **Aktive Skills**
  - Basic Attack
  - Throw Stone
  - Poison Dagger
  - Power Smash
  - Fireball
  - Waterball
  - Tribe Scream (Buff)
- **Passive Skills**
  - Regeneration (Troll)
  - Fear (Orc)
  - Absorb (Slime)
  - Greed (Event)

### Status Effects

- **Poison** – DOT auf MaxHP
  - **Fear** – Speed halbieren
  - **Absorb** – Damage Reduction
  - **Regeneration** – Heal per Turn
  - **Tribe Scream Effect** – AP Buff
  - **Permanent Effects** – unendlich (Absorb, Regeneration, Fear)
-

## 6. Kampfsystem / Damage Pipeline

### Ablauf:

1. Raw Damage ( $AP * Skill.Power$ )
  2. Resistenzberechnung (je nach DamageType)
  3. StatusEffect.ModifyFinalDamage()
  4. Garantierter Mindestschaden: **min 1**
  5. Schaden zufügen
  6. Skill setzt Cooldown
- 

## 7. Rundenablauf

### Start der Runde

- Start-of-Turn Effekte (Regen, Buff Apply)

### Spielerphase

- Input → Skill wählen → Attack

### Enemyphase

- AI wählt Skill:
  - stärkster Skill ready → bevorzugt
  - sonst Basic Attack

### Ende der Runde

- DOT Effekte (Poison etc.)
  - Cooldown Tick
  - StatusEffect Duration Tick
  - Prüfen auf Tod
-

## 8. UI / Konsolen-Interface

- ASCII-Sprites für Player & Enemy
  - Player-Box (Name, HP, AP, DP, Speed)
  - Enemy-Box
  - Skill-Auswahl (Liste mit Cursor)
  - Message-Box (Schaden, DOT, Buffs, Logs)
- 

## 9. Technische Struktur (Design-Ebene)

*(ohne Code, nur Gameplay-Sicht)*

- **Game Flow Manager**
    - Rundensteuerung
    - Kampfstart & Siegbedingungen
  - **Monster Factory**
    - Erzeugt Monster + Skills + Startwerte
  - **Balancing Data**
    - Basiswerte, Scaling, StatPoint-Werte
  - **Damage Pipeline (Designsicht)**
    - Raw → Resist → StatusMods → Final
- 

## 10. Datenstruktur / Balancing-Werte

### Monster-Basiswerte (aus Balancing)

- HP, AP, DP, Speed pro Monster
- Resistenzen pro Monsterklasse

### Skill-Konstanten

- Multipliers (z. B. ThrowStone 1.3f)
- DOT-Prozentwerte (z. B. Poison 10%)
- Cooldowns

## **Status-Effect-Dauer**

- TribeScream: 5 Runden
  - Poison: 2 Runden
  - Fear: Permanent
  - Absorb: Permanent
- 

## **11. Projektumfang & Grenzen (Scope)**

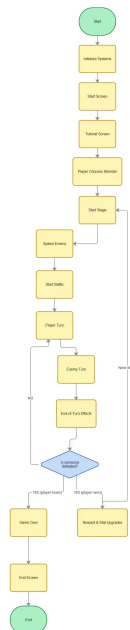
### **Enthalten**

- 4 Monsterklassen
- Aktive & Passive Skills
- Status Effekte
- Damage Pipeline
- Konsolen-UI
- Turn-Based System
- AI-Battle-System

### **Nicht enthalten**

- Animationen
- Audio
- Items
- Multiplayer
- Komplexe Story
- Open World / Progression außerhalb Kämpfe

## 12. Programm Ablauf Diagramm



Der Ablauf des Spiels beginnt mit einem **Start Screen**.

Anschließend folgt ein **Tutorial Screen**, der die Steuerung erklärt und die wichtigsten Grundlagen des Monsterkampf-Simulators vermittelt.

Danach gelangt der Spieler zum **Monster-Auswahlbildschirm**, in dem er eines von vier Monstern auswählen kann. Jedes Monster besitzt eine eigene Beschreibung sowie unterschiedliche Werte.

Im nächsten Schritt wechselt das Spiel in den **Kampfbildschirm**.

Hier werden das Monster-Sprite und die zugehörigen Informationsboxen angezeigt.

Zu Beginn des Kampfes wird anhand des **Speed-Wertes** entschieden, welches Monster die erste Runde erhält.

- Beginnt der Gegner, führt er sofort einen Angriff aus und es erfolgt ein entsprechender Text-Output.
- Beginnt der Spieler, kann er einen Skill auswählen und den Angriff mit *Enter* bestätigen.

Nach jeder Aktion wird die UI aktualisiert und ein Text-Output ausgegeben.

Am Ende einer Runde wird geprüft, ob eines der Monster besiegt wurde.

- **Falls ja**, wird das Ergebnis angezeigt und zum **End Screen** gewechselt.
- **Falls der Spieler gewonnen hat**, erhält er zusätzlich eine kurze Kampfstatistik und kann anschließend durch Drücken von *Enter* zum **Reward-Screen** fortfahren, wo er ein Level-Up erhält und Stat-Punkte verteilen kann.

Nach der Belohnungsphase beginnt die Schleife erneut:

Ein neuer Gegner wird generiert und mit dem passenden Level zugewiesen.

Dieser Ablauf wiederholt sich so lange, bis der Spieler verliert.

In diesem Fall gelangt er ebenfalls zum End Screen, wie im Flussdiagramm dargestellt.

## PROJEKTABLAUFPLAN (PHASENPLAN)

### Phase 1 – Analyse & Konzept

- Ideendefinition
- Mini-GDD
- Projektstrukturplan
- Datenmodell & Klassendiagramm
- Skill- & StatusEffect-Konzept

### Phase 2 – Grundarchitektur

- Program.cs
- GameManager
- BattleManager
- Dependencies
- InputSystem (IPlayerInput / KeyboardInputManager)

### Phase 3 – Gameplay-Logik

- MonsterBase + Meta

- MonsterFactory
- Balancing-System
- Skills (Active, Passive)
- StatusEffectBase + Effekte
- DamagePipeline

## **Phase 4 – UI / Screen-System**

- UIManager
- ScreenManager
- PrintManager
- SymbolManager

## **Phase 5 – Test & Debug**

- DiagnosticsManager integrieren
- Rundenflow testen
- Monster/Skill-Balancing testen
- Edge-Cases testen (DOT, Buffs, Speed-Probleme)

## **Phase 6 – Feinschliff**

- Kommentare vervollständigen
- Refactoren
- Cleaning
- Finaler Build

## **Phase 7 – Dokumentation / Abgabe**

- Mini-GDD
- Projektstrukturplan
- Zeitplan
- Readme

# Zeitplan Gantt-Diagramm

## Monster\_Kampf\_Simulator

Read-only view, generated on 03 Dec 2025

