Gruppe: Paul; Valentin; Felix; Georg

Spiel: Etintrof

**Engine: Pygame**

Battle Royale spiel 2-4 Spieler

Map ähnlich wie Brawlstars mit blöcken

Kamera bewegt sich mit und der Spieler ist immer Zentriert.

UI elemente

**Dimensionierung:** 1 Spieler 1 Feld**,** Map 50\*50

**Mauer:** Wasser kann man drüberschießen aber nicht laufen; Mauer kann man garnichts; und normalen Boden kann man schießen und und auch Laufen

Server: wissen wo Spieler sind, Wissen ob Schüsse getroffen hat,

**PC-> Server:** Schussrichtung, Bewegrichtung, Server muss verarbeiten alles

**Server -> PC:** PC macht die ganze Visualisierung, Am Anfang Schickt der Server die Maps, Position von den Spielern, Projektile inklusive Richtung, getroffen oder nicht , Leben und Schüsse,

**2 Leute Programmieren PC seite: Anmelde UI, Input an server senden, Visualisierung**

**2 Leute Programmieren Server: Input verarbeiten, positionen und leben der spieler und projectile zurücksenden**

**Aufgabenstellung: Entwicklung des Battle Royale Spiels "Etintrof"**

**1. Projektüberblick**

Entwickelt wird ein 2-4 Spieler Battle Royale Spiel namens **"Etintrof"** unter Verwendung der **Pygame Engine**. Das Spiel setzt auf ein dynamisches Gameplay mit Echtzeit-Interaktion zwischen Client (PC) und Server. Der Server übernimmt die Spiel-Logik und verarbeitet Spieleraktionen, während die Clients die Visualisierung und Benutzereingaben übernehmen.

**2. Spielbeschreibung**

* **Spielmodus:** Battle Royale (2-4 Spieler)
* **Kartenlayout:**
  + Map-Größe: 50 x 50 Felder
  + Jedes Feld entspricht einem Spielerbereich, d.h. 1 Spieler = 1 Feld.
  + Die Karte ist ähnlich aufgebaut wie in Brawlstars, mit verschiedenen Blocktypen:
    - **Normaler Boden:** Betretbar und beschießbar.
    - **Mauer:** Kann weder betreten noch beschossen werden.
    - **Wasser:** Kann nicht betreten werden, aber Schüsse können darüber hinweg abgefeuert werden.
* **Kamera & UI:**
  + Die Kamera folgt dem Spieler und hält ihn stets zentriert.
  + UI-Elemente zur Anzeige von Spielerstatus (z.B. Leben, Schüsse, etc.) und weiteren wichtigen Informationen.

**3. Technische Architektur & Kommunikation**

**Client (PC) Seite:**

* **Aufgaben:**
  + **Anmelde-UI:** Implementierung einer Benutzeroberfläche zur Anmeldung.
  + **Input-Handling:** Erfassen der Eingaben (Bewegungsrichtung, Schussrichtung) und Versand dieser Daten an den Server.
  + **Visualisierung:** Darstellung der Spielwelt, Spieler, Projektile, UI-Elemente und Animationen.
  + **Initiale Datenübernahme:** Beim Spielstart werden vom Server die Map, Spielernummer(Farbe),Spielerpositionen, Projektile (inkl. Richtung und Trefferstatus), Leben und Schüsse übermittelt.

**Server Seite:**

* **Aufgaben:**
  + **Input-Verarbeitung:** Empfang und Verarbeitung der Daten von den Clients (Bewegungsrichtung, Schussrichtung).
  + **Spielzustandsverwaltung:** Aktualisierung und Verwaltung der Positionen und Lebenszustände der Spieler.
  + **Projektil-Management:** Berechnung der Bewegung und Kollision von Projektile, Ermittlung, ob Schüsse getroffen haben.
  + **Kommunikation:** Versand aktueller Spielinformationen (Spielerpositionen, Projektil-Daten, Trefferstatus, Leben, Schüsse) an alle Clients.

**4. Aufteilung der Aufgaben im Team**

**Teamzusammensetzung:**

* **Gruppe:** Paul, Valentin, Felix, Georg

**Zuständigkeiten:**

* **PC-Seite (Client) – 2 Personen: Ziege**
  1. **UI & Anmeldung:**
     + Entwicklung der Anmelde-UI und Implementierung der Verbindungslogik zum Server.
     + Gestaltung und Umsetzung der Benutzeroberfläche.
  2. **Input-Handling & Visualisierung:**
     + Implementierung der Steuerung (Bewegungs- und Schussrichtungseingabe).
     + Darstellung der Spielwelt (Karte, Spieler, Projektilbewegungen) sowie dynamischer UI-Elemente.
     + Integration der empfangenen Daten vom Server in die Visualisierung.
* **Server-Seite – 2 Personen: Löwe**
  1. **Server-Logik & Input-Verarbeitung:**
     + Implementierung der Logik zur Verarbeitung von Client-Eingaben.
     + Berechnung und Aktualisierung der Spielerpositionen sowie Verwaltung der Spiellogik.
  2. **Projektil- & Kollisionsmanagement:**
     + Entwicklung der Mechanismen zur Berechnung der Flugbahnen von Projektile.
     + Implementierung von Kollisionserkennung (Unterscheidung zwischen normalem Boden, Wasser und Mauer) und Bestimmung des Trefferstatus.
     + Verwaltung der Lebenszustände und Schusszählungen.
  3. **Kommunikationsschnittstelle:**
     + Implementierung der Netzwerkkommunikation (Client-Server-Architektur).
     + Versand initialer Spieldaten (Map, Spielerpositionen, etc.) und kontinuierliche Aktualisierung des Spielzustands an die Clients.

**5. Anforderungen & Dimensionierung**

* **Map:** 50 x 50 Felder, wobei jedes Feld als Einheit für den Spieler fungiert.
* **Gameplay:**
  + Der Spieler ist immer in der Bildschirmmitte.
  + Unterschiedliche Blocktypen (normaler Boden, Wasser, Mauer) beeinflussen die Bewegung und Schusswege.
* **Netzwerk:**
  + Echtzeitkommunikation zwischen Clients und Server.
  + Der Server trägt die zentrale Verantwortung für die Berechnung und Verteilung des Spielzustands.

**6. Datenpakete**

* Map (50x50 Array) 0=Boden; 1=Wand; 2=Wasser;
* Eingabe PC->Server: Richtungsvektor Laufen (WASD), Richtungsvektor Schauen (Maus), LMB gedrückt?
* Visualisierung Server->PC: Position und Orientation aller Spieler und Projektile; Leben jeder Spieler

**7. Projektmeilensteine**

1. **Planungsphase:**
   * Definition der Spielmechanik, UI-Design, und Netzwerkinfrastruktur.
   * Aufgabenverteilung und Erstellung eines detaillierten Zeitplans.
2. **Implementierungsphase – Server-Seite:**
   * Aufbau der Serverarchitektur.
   * Implementierung der Input-Verarbeitung, Kollisionslogik und Projektil-Management.
   * Testen der Netzwerkkommunikation (Simulation der Clients).
3. **Implementierungsphase – Client-Seite:**
   * Entwicklung der Anmelde-UI und Integration der Steuerungslogik.
   * Umsetzung der Visualisierung (Kamera, Spieler, Map, UI-Elemente).
   * Integration und Test der Kommunikation mit dem Server.
4. **Integration & Testphase:**
   * End-to-End Tests der Client-Server-Interaktion.
   * Fehlerbehebung, Performance-Optimierung und finaler Feinschliff.
5. **Abschlussphase:**
   * Dokumentation des Projekts.
   * Präsentation und Übergabe des fertigen Spiels.

**7. Abschlussbemerkungen**

Das Projekt "Etintrof" erfordert eine enge Zusammenarbeit zwischen den Teams, da die Client- und Server-Komponenten eng miteinander verzahnt sind. Es wird erwartet, dass alle Teammitglieder regelmäßig kommunizieren und ihre Fortschritte abstimmen, um ein reibungsloses, synchronisiertes Spielerlebnis zu gewährleisten.

# Backlog

|  |  |  |
| --- | --- | --- |
| **Datum** | Ziege (Pc-Seite) | Löwe (Server-Seite) |
| 7.2.2025 | Gedanken über das Spiel | -//- |
| 14.2.2025 | UI Map angefangen (Wand, Boden, Wasser) Runder Spieler Erstellt | Socket Ausprobieren (nicht erfolgreich) |
| 21.2.2025 | Player Zentrieren, Player Zoom | Socket erfolgreich einseitige Kommunikation |
| 28.2.2025 | Gun Visualisierung, Drehung der Spieler | Socket erfolgreiche zweiseitige Kommunikation |
| 7.3.2025 | Ausrichtung der Waffe, Bullet Visualesiert | Anfangen mit der Gamelogik (Bewegen) |
| 14.3.2025 | Login Screen, Helf Bar | Gamelogik und Schussrichtung Fertig |
| 21.3.2025 | -//- | Testung der Pc-Server Kommunikation |
| 28.3.2025 | -//- | Problemlösung der Lags zwischen Server und Client |
| 4.4.2025 |  | Funktionierende kommunikation |
| 11.4.2025 |  | Spiel Logik: Leben/Tod |
| 18.4.2025 |  | Spiel Logik: Game Over screen, Schüsse |
| 25.4.2025 |  | Bugs Beheben |
| 2.5.2025 |  | Bugs Beheben |
| 9.5.2025 |  | Testphase |
| 16.5.2025 |  | Testphase |
| 23.5.2025 |  | Vorstellung des Spieles |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Ziele:

### Team Ziege:

1. Map erstellen (Wand, Boden, Wasser)
2. Spieler erstellen
3. Spieler Zentrieren und Waffe hinzufügen
4. Drehung und Bewegung des Spielers
5. Bullet Visualisierung
6. Login Screen, helf Bar

### Team Löwe:

1. Erste Kommunikation mit Socket
2. Kommunikation mit Socket in beide Richtungen
3. Gamelogik: Spieler Bewegen
4. Gamelogik: Bullets und richtung des Spielers
5. Komplettes Frame verarbeiten und neues schicken

### Gemeinsam:

1. Beide Codes (Client und Server) miteinander verbinden
2. Einwandfreie Kommunikation
3. Spiel Logik fertigstellen: Leben/Tod, Game Over screen, Schüsse
4. Letzten Bugs beheben
5. Testphase (2h Gamen)
6. Finish (Vorstellung)