**Game Design Document – Binding of Tank**

**Autoren: René Kraus, Franz Mörike, Alexander Sigmund, Jan Apsel, Dennis Braunmüller**

**Kurs: CMN5201gpr-0322**

**Abgabedatum: 08.04.2022 (Pre-Production)**

**Spielkonzept:**

In einem industriellem *cyberpunk* Setting muss sich ein einsamer Soldat, gestrandet und umzingelt vom Feind mit seinem Panzer durch Wellen an Gegnern kämpfen… Es gibt kein Entkommen!

Der Spieler spielt in der Top-Down Perspektive. Die Gegner sind insektenartige Aliens, die auch Panzer steuern und versuchen den Spieler zu töten. Natürlich sind diese zahlenmäßig weit überlegen und verfügen über high-tech Alientechnologie. Die Level sind offen gestaltet und rendern in Reichweite des Spielers.

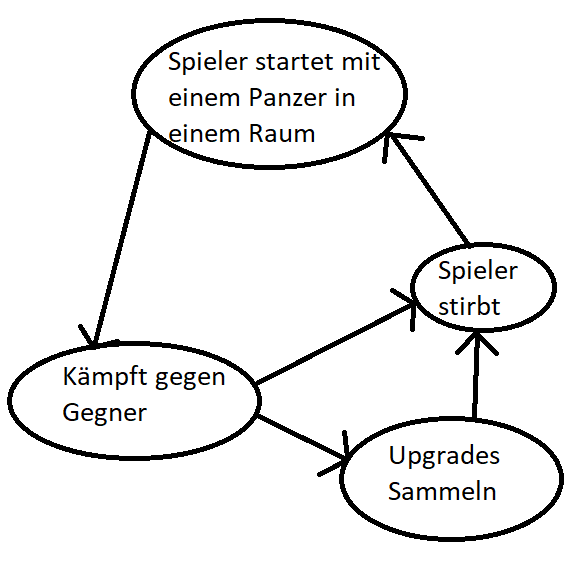
**USP:**

Ein *sci-fi bullethell dungeon crawler* mit *Heatmeter*.

**Gameplay Tags:**

* Dungeon crawler
* Roguelike
* Indie
* Bullethell

**Grundlegender Gameloop:**



**Platform:**

Vorerst nur auf PC, vielleicht spätere Portierung auf Android / IOS.

**Zielgruppe:**

Panzerliebhaber und Dungeon-Crawler Fans (Spiele wie *Binding of Isaac*, *Hades*, ...). Für Spieler die gerne versuchen, bis zum Endboss zu kommen und zu besiegen. Nach dem Arcade-prinzip *plug&play*. Der Spieler startet prinzipiell immer am selben Punkt, nur die Räume Gegner ändern sich.

**Funktionen des Team isFine-Entertainment:**

**Lead Project Manager:** Franz Mörike

**Lead Artist:** Jan Apsel

**Lead Designer:** Jan Apsel

**Lead Coder:** René Kraus

**Lead UI Designer:** Dennis Braunmüller

**Lead Audio Engineer:** Franz Mörike

**Lead QA Engineer:** Alexander Sigmund



**Steuerung:**

**Spieler-Steuerung:**

* Der Spieler steuert einen Hover-Panzer der sich auch seitwärts bewegen kann.
* Dragmechanik bei der die Geschwindigkeit langsam verringert wird wenn nicht aktiv beschleunigt wird.
* Kollision mit Gegenständen.

**Zielen:**

* Drehen des Panzers in die Richtung in die mit der Maus gezielt wird.
* Die Waffen haben einen Spielraum und können sich beim Zielen zum Beispiel 30 Grad in beide Richtungen drehen.

**Items:**

**Waffen:**

*Heatmeter? Wenn der Spieler eine Waffe abfeuert, wird eine Leiste aufgeladen, wenn diese das maximum erreicht kann der Spieler nicht mehr feuern. Hört der Spieler auf zu Feuern verringert sich diese. Ist das Meter auf max verlängert sich die Entladungs Zeit*

* Laser: Ein Beam der mehr Damage macht, desto länger man diesen aktiv hält
* Heavy Bulltet: Langsamer cast, höher Schaden, hohe Armor Penetration
  + Light Bullet (Normal Weapon): Durchschnittliche Feuerrate und Schaden
  + Shot Gun : Mehrfach Schuss der ausstreut, Fragmente machen wenig Schaden auf höherer Distanz
  + Machine Gun: Sehr hohe Feuerrate, wenig Schaden einzelner Bullets
  + Rocket Launcher: Feuert eine Rakete, welche bei Aufprall explodiert

**Armor Upgrades:**

* Damage Reduction (reduziert eingehenden Schaden)
* Deflection (reflektiert/blockt Schaden komplett)
* Armor Penetration Reduction (reduziert die durchschlagskraft von Bullets)

**Upgrade Slots:**

Sollen wie Gems oder Kristalle funktionieren. Diese geben spezielle Effekte und können den Spielstil maßgeblich ändern.

* Schilde
* Passive Regeneration
* Andere Bullets (Explosive Bullets)
* Multishot

**Schadensmodell:**

**Visuell:**

* Panzer und *eventuell auch die Spielumgebung* sind zerstörbar.
* Dazu werden verschiedene Grafiken übereinandergelegt, also z.B. eine Grafik vom gesunden Panzer und eine oder mehrere Grafiken von einem zerstörtem Panzer. Die Schadensgrafiken werden mithilfe einer sich erweiternden Maske dort eingeblendet wo der Panzer getroffen wurde.

**Gameplay-technisch:**

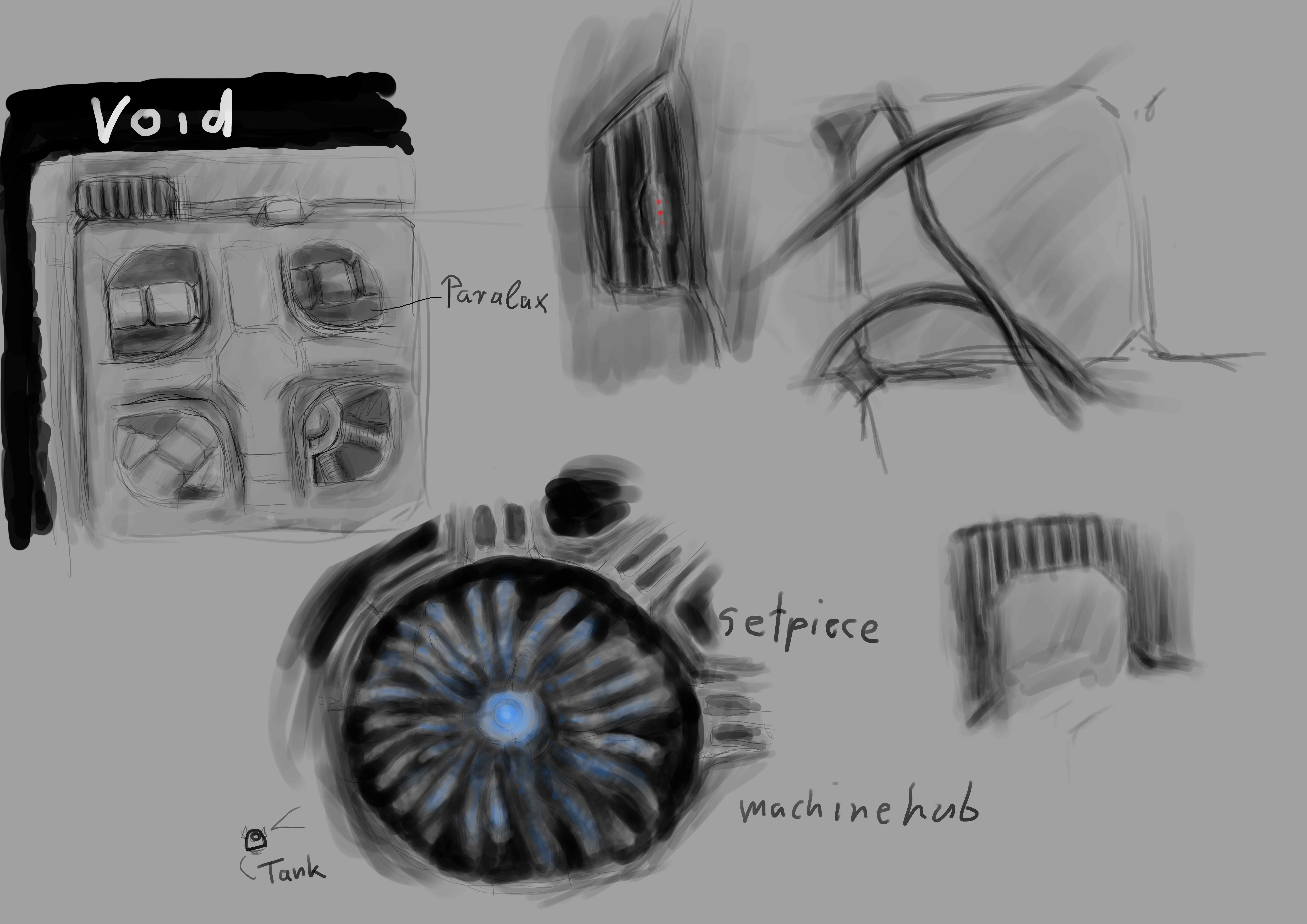
* Die Panzer haben an der Außenseite Panzerung. Diese wird bei Beschuss abgetragen wodurch die innere Struktur des Panzers freigelegt wird welche den generellen Lebenspunkte-Pool des Panzers darstellt. Wird diese beschossen können *interne Systeme kurzfristig ausfallen* und der Panzer zerstört werden.
* *Waffen sind oft spezialisiert und verursachen mehr Schaden gegen Panzerung, innere Struktur oder Schilde.*



Grafik 3: Screenshots aus dem Spiel Starsector. Zu sehen ist ein Raumschiff mit verschieden starker Beschädigung.

**Artdesign:**

* Die Gestaltung des Schadensmodells orientiert sich an dem Spiel Starsector. (zu sehen in Grafik 3)
* Die Gestaltung der Panzer orientiert sich ebenfalls an diesem Spiel.
* Die Umgebung ist industriell und futuristisch gestaltet zum Beispiel mit viel Metall, Rohren, Schläuchen und Konsolen.



Grafik 4: Umgebungsskizze Grafik 5: Panzermodell

**Animationen:**

* Waffen/Schuss Animationen
* Objekte im Level
* UI
* Buttons
* Panels
* (Bilder)

**UI:**

Das Spiel soll eine benutzerfreundliche Oberfläche aufweisen, die leicht zu navigieren und bedienen ist.

Die einzelnen Komponenten wie z.B. Buttons, Panels, Schriftarten, Bilder, … sollen das Spiel grafisch noch einen letzten “Touch” geben.

**Game-Sound:**

* Arcade-like
* geloopte Spielmelodie
* Main Menu Hintergrundmusik
* *Button-Click-sounds*
* Soundeffekte bei Bewegung des Spielers / der Gegener.
* Explosionssounds bei Raketenaufprall o.Ä.

**Effekte:**

* Rauch
* Explosionen
* Feuer
* Kabelbrand
* Laser

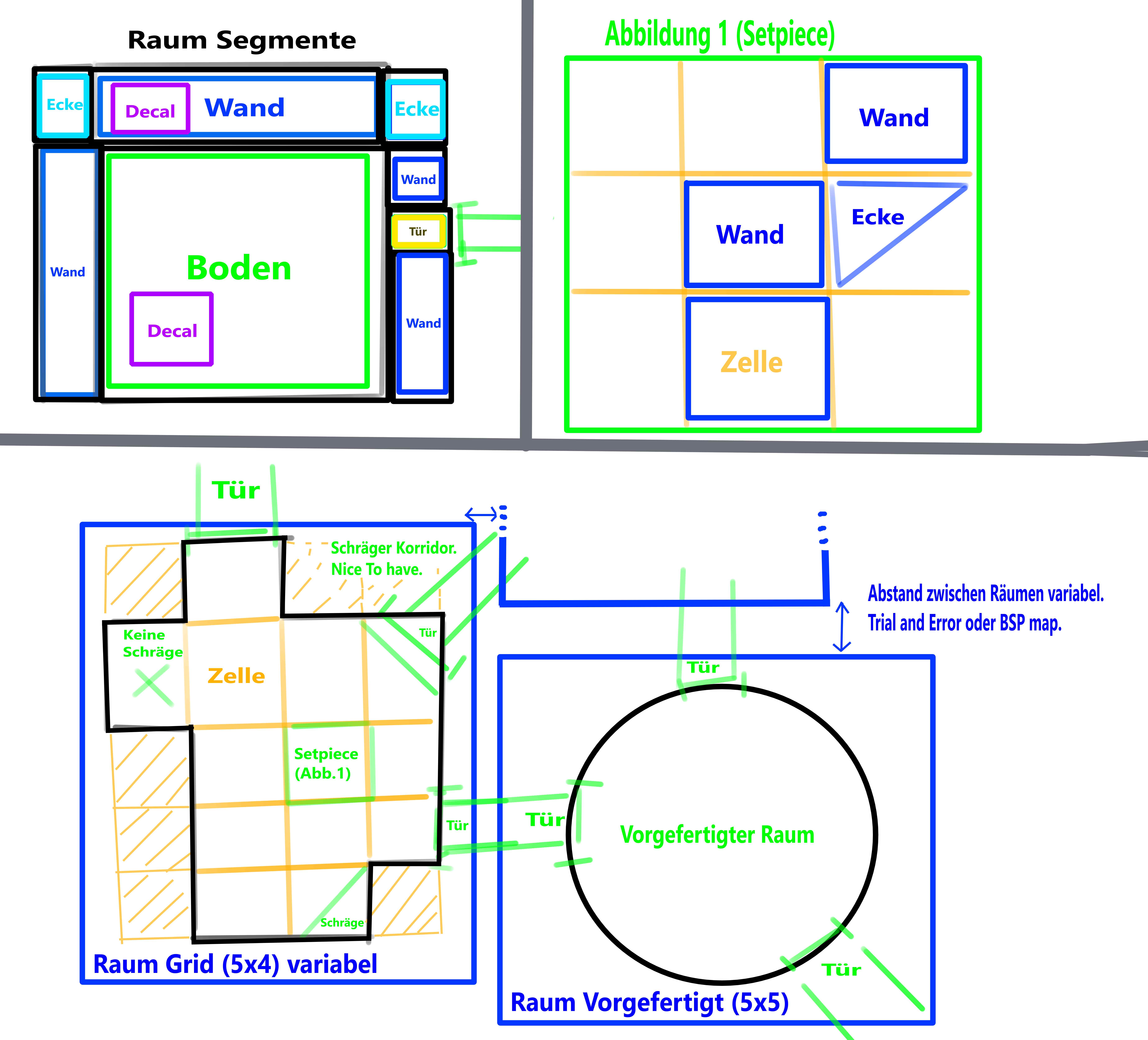
**Taskliste:**

* Generische Levelgenerierung (Franz)
* Steuerung / Player / Schadensmodell (Jan)
* Leveldesign (Jan / Dennis)
* Items / Upgrades (Alex)
* KI (Rene)
* Effekte (Franz)
* Animationen (Dennis)
* UI (Dennis)
* Sound (Franz)

Die Einteilung in speziellere Teilbereiche und Zeitmanagement werden mithilfe des Tools *hacknplan* verwaltet.

**Generische Levelgenerierung:**

Das Spiel wird vorerst innerhalb einer Szene bzw. Einem Level stattfinden, in der es mehrere Räume gibt, die prozedural pseudorandom generiert werden sollen. Am Ende soll ein Bossraum sein. Danach könnte man ggf. das nächste Level aufziehen mit schwierigeren und mehr Gegnern, anderer Umgebung und bessere Upgrades / Items.



Grafik 6: Konzept Level Generierung

**Assetliste:**

**Player:**

* Panzermodell (Jan)
* Waffen (Jan)
* Bewegungsanimationen (Dennis)
* Waffensounds (Franz)
* Bewegungssounds (Franz)

**Gegner:**

(Gegnerassets sollen ähnlich wie die des Players sein, allerdings mit verändertem Aussehen, Effekten, Funktionen...)

* Panzermodell (Jan)
* Waffen (Jan)
* Bewegungsanimationen (Dennis)
* Waffensounds (Franz)
* Bewegunssounds (Franz)

**Level / Environment**

* Wände (Jan)
* (zerstörbare Gegenstände) (Jan)
* Obstacles (Jan)
* Explosionen (Franz)
* Explosions-sounds (Franz)
* Generierte Räume (siehe procedural level generation / Franz)

**Anderes:**

* Screenshake (Franz)

**Künstliche Intelligenz**

Es wird eine künstliche Intelligenz entwickelt, die auf dem FSM (Finite-State Machine) Model basiert. Die FSM wechselt je nach Entscheidung von einem Zustand (State) in einen anderen, um auf die Bedingungen zu reagieren. Um das Verhalten der KI möglichst einfach an das gewünschte Situation anpassen zu können, werden die States als Scriptable Objects definiert. Des Weiteren wird es über Unity Events möglich sein, weiteres Verhalten einfach zum State hinzufügen.

**Aufbau der FSM:**

|  |  |
| --- | --- |
| **Komponente** | **Beschreibung** |
| State | States enthalten das Verhalten, dass ausgeführt wird, wenn die FSM den State aufruft. |
| Action | Definiert das zu ausführbare Verhalten des State. |
| Decision | Definiert die Bedingung, die für einen State wechselt, sorgt. |
| Transition | Definiert die Verbindung zwischen den unterschiedlichen States. |
| Unity Event | Ermöglicht es das Verhalten, um weitere Funktionen zu ergänzen. |

**Beispielhafte Zusammensetzung eines KI-Verhaltens:**

**Verhalten:**

|  |  |
| --- | --- |
| **State** | **Beschreibung** |
| Idle | KI steht auf der Stelle und bewegt sich nicht. |
| Chase | KI verfolgt den Spieler. |
| Evade | KI weicht dem Spieler oder verbündete KI aus. |
| Patrol  (Random Patrol) | KI läuft festgelegte Wegpunkte oder zufällig generierte Wegpunkte ab. |
| Attack | KI führt einen Angriff durch, wenn die Kampfreichweite erreicht oder unterschritten wurde. |

Da die Komponenten als Scriptable Objects definiert sind, soll sich das KI-Verhalten wie in einem Baukastensystem zusammensetzen lassen.

**Milestones:**

**Pre-Production:**

- Festigen einer grundlegenden Spielidee.

- Aufgabenverteilung nach Kompetenzen.

- Finden eines USP

- GDD erstellen

**Prototyp:**

Erster Spielbarer Prototyp muss stehen.

- Keine vollständigen Features

- kann Bugs enthalten

- soll grundsätzlichen Gameflow zeigen

**Zwischenbericht „Alpha“:**

- Einschätzung der bisherigen Fortschritte.

- Zeitmanagement für verbliebende Zeit daran anpassen.

- Anpassen von Projektentscheidungen / zielen.

**Zwischenbericht „Beta“:**

Ähnlich wie Alpha:

- Einschätzung der bisherigen Fortschritte.

- Zeitmanagement für verbliebende Zeit daran anpassen.

- Anpassen von Projektentscheidungen / zielen.

- Spiel sollte größtenteils fertiggestellt werden

**Praktische Medienproduktion:**

- Präsentation des finalen Produkts.

- Umsetzen / Berücksichtigung von Feedback

- Einhalten von Qualitätsstandarts der Industrie

- Erreichen der vereinbarten Ziele

**Projektreflexion:**

Erstellen eines Dokumentes, das beinhaltet:

- Zusammenfassung des Projektes.

- Evaluieren des eigenen Produktionsprozesses und Projektentscheidungen

- Kritische Reflexion (was hätte anders / besser laufen können und wie?)

- Wurden kreative Ideen gerecht umgesetzt?

- War das Zeitmanagement / die Aufgabenverteilung passend?

- Wie wurden Problemelösungen angegangen?

- War die Kommunikation im Team zufriedenstellend?

**Vorgaben aus dem Canvas:**

Die Projektreflexion soll folgende Punkte abdecken:

- Kreativer Hintergrund und künstlerische Richtung

- Ursprüngliche Idee und tatsächliche Umsetzung

- Ergebnisse und Erfolge des Projektes

- Kosten- und Ressourcenmanagement

- Zeit- und Projektmanagement

- Kommunikation (mit Dritten)

- Qualitätsbewusstsein

- Gut und schlecht gelaufene Dinge, Probleme und Lösungen

- Zeitaufwand und Engagement

- ca 1500 Wörter

- Formatiert (Blocksatz, Seitenzahlen...)

In *kursiv* marktierte Punkte sind vorerst *nice-to-have-Features*, andere Features haben Vorrang.