Inhaltsverzeichnis

[Einleitung 2](#_Toc169079557)

[Hintergrund des Projekts 2](#_Toc169079558)

[Zweck und Ziel des Projekts 2](#_Toc169079559)

[Team und Thema 2](#_Toc169079560)

[Team 2](#_Toc169079561)

[Thema 2](#_Toc169079562)

[Realisierung der Teilbereiche 2](#_Toc169079563)

[Player 2](#_Toc169079564)

[Collectable 2](#_Toc169079565)

[Character design 3](#_Toc169079566)

[Kartendesign 3](#_Toc169079567)

[Kartengeneration 3](#_Toc169079568)

[Hauptmenü 3](#_Toc169079569)

[Gegner 3](#_Toc169079570)

[Boss 3](#_Toc169079571)

# Einleitung

## Hintergrund des Projekts

Am Anfang konnte sich ein Thema ausgesucht werden, welches mit der Übung oder der Vorlesung zu tun hat. Die wesentlichen Themen der Vorlesung wären zum Beispiel Text, Bild, Audio und Video. Bei den Beispielen in der Übung wurde ein Spiel gezeigt, wodurch die Entscheidung, ein Spiel zu programmieren, getroffen wurde.

## Zweck und Ziel des Projekts

Mit dem Projekt soll das Team in den jeweils favorisierten Bereichen einen Leistungsaufwand von rund 20 Stunden umsetzen und sich dabei neue Fähigkeiten und Techniken aneignen. Im Fall dieses Videospiels sind dies das Design, die Kartengeneration, der Spieler und die Gegner. Um ein lauffähiges Spiel entwickeln zu können, musste das Team gut zusammenarbeiten, da jede dieser 4 Kategorien ineinandergreifen und zum Teil voneinander abhängig sind. Wenn dies nicht der Fall sein sollte, könnten ungewollte Bugs auftreten, das Thema des Spiels nicht zu der eigentlichen Spielerfahrung passen, die Spielerfahrung im Allgemeinen nicht gut sein oder das Spiel erst gar nicht funktionieren.

# Team und Thema

## Team

Unser Projektteam besteht aus vier engagierten Mitgliedern, die jeweils spezifische Bereiche des Projekts übernommen haben, um ein erfolgreiches Ergebnis zu erzielen. Je nach Teammitglied wird verschiedene Programmiererfahrung mitgebracht, jedoch wurde am Anfang des Projektes die Aufgabenteilung so gewählt, dass jedes Teammitglied die Bereiche des Projektes bekommt, die favorisiert wurden.

1. Jan Feldmayer – Player & Item Entwicklung
2. Jonas – Character- & Kartendesign
3. Mathias Kröpfl – Kartengeneration & Menu
4. Wasgen – Gegner & Boss Entwicklung

## Thema und Projektbeschreibung

“Pixel Purge” ist ein spannendes Top-Down 2D-Roguelike-Shooter-Spiel, das in Unity entwickelt wurde. Das Spiel wurde mit einem Singleplayermodus implementiert, in dem der Spieler gegen eine Vielzahl von Feinden kämpfen und versucht sich bis zum Boss durchzuschlagen. Wenn der Boss besiegt wurde, ist das Spiel beendet und man wird zurück ins Hauptmenu geladen. Durch die Tötung der Gegner in den verschiedenen Räumen, können Gegner kleine Health-Items fallen lassen, welche dem Spieler helfen dessen Leben wieder herzustellen. Das Grundprinzip des Spiels ist, dass der Spieler mittels Maus und Tastatur einen Charakter durch ein Level steuert, indem ein Krieg zwischen Kreisen und Polygonen herrscht. Inmitten dieses Krieges ist der Hauptcharakter „Pixel-Bro“, der Anführer der Kreise. Dieser versucht sich durch die Territorien des Feindes zu Kämpfen und die Allianz der Polygone zu zerstören. Um dies zu schaffen, muss der Anführer der Allianz, „Pixel-Dominator“, bezwungen werden. Dieser ist jedoch in seinem Geheimversteck, welches sich nur mithilfe aller Schlüssel aufsperren lässt.

# Realisierung der Teilbereiche

## Player

wad

## Collectable

Aaa

## Character design

A

## Kartendesign

A

## Kartengeneration

A

## Hauptmenü

A

## 

## Gegner

Es gibt eine Vielzahl von Gegnertypen, darunter Nahkampf-, Fernkampf- und Boss-Gegner, die jeweils einzigartige Fähigkeiten, zusätzliche Mechaniken und Verhaltensweisen aufweisen.

**1. Nahkampf-Gegner:** Die Nahkampf-Gegner nutzen die EnemyMovement und EnemyAttack Klassen. Die EnemyMovement Klasse steuert die Bewegung des Gegners. Sie aktualisiert ständig die Richtung zum Spieler und passt die Geschwindigkeit des Gegners entsprechend an. Wenn der Spieler in Reichweite ist (\_attackRange), bewegt sich der Gegner in Richtung des Spielers und greift an. Die EnemyAttack Klasse ermöglicht es dem Gegner, Schaden zu verursachen, wenn er mit dem Spieler kollidiert. Dies stellt eine direkte und unmittelbare Bedrohung für den Spieler dar und erfordert schnelle Reaktionen und Bewegungen, um Schaden zu vermeiden.

**2. Fernkampf-Gegner:** Die Fernkampf-Gegner nutzen auch die EnemyMovement aber auch die RangedEnemyAttack Klasse. Die RangedEnemyAttack Klasse ermöglicht es dem Gegner, Projektile auf den Spieler zu schießen. Die Geschwindigkeit, der Schaden und die Zeit zwischen den Schüssen können individuell eingestellt werden. Die Projektile werden in Richtung des Spielers abgefeuert, wenn der Spieler in Sichtweite ist und der Gegner sich nicht bewegt. Dies stellt eine indirekte Bedrohung für den Spieler dar und erfordert taktisches Bewegen und Ausweichen.

**3. Boss-Gegner:** Die Boss-Gegner nutzen die EnemyMovement und EliteEnemyAttack Klassen. Die EliteEnemyAttack Klasse ähnelt der RangedEnemyAttack Klasse, ermöglicht es dem Boss jedoch, sich gleichzeitig zu bewegen und zu schießen. Dies erhöht die Schwierigkeit und macht den Boss zu einer größeren Herausforderung für den Spieler. Der Boss stellt die ultimative Prüfung der Fähigkeiten des Spielers dar und erfordert eine Kombination aus schnellen Reaktionen, taktischem Bewegen und effektivem Angriff.

**4. Spielerbewusstsein:** Die PlayerAwarenessController Klasse ermöglicht es den Gegnern, sich des Spielers bewusst zu sein, wenn er sich in einer bestimmten Entfernung befindet. Dies wird verwendet, um das Verhalten des Gegners zu steuern, z.B. ob er den Spieler angreift oder ihm folgt. Dies trägt zur Dynamik des Spiels bei und macht das Verhalten der Gegner weniger vorhersehbar und realistischer.

**5. Unverwundbarkeit nach Schaden:** Die EnemyDamagedInvincibility Klasse ermöglicht es den Gegnern, nach dem Erhalt von Schaden für eine bestimmte Dauer unverwundbar zu werden. Während dieser Zeit ändert sich die Farbe des Gegners, um den Spieler visuell darauf hinzuweisen, dass der Gegner vorübergehend unverwundbar ist. Dies fügt eine zusätzliche Ebene der Komplexität hinzu und erfordert, dass der Spieler seine Angriffe sorgfältig timet.

**6. Zerstörung von Gegnern:** Die EnemyDestroyController Klasse ermöglicht es, einen Gegner vollständig aus dem Spiel zu entfernen. Dies wird typischerweise verwendet, wenn der Gegner genug Schaden erlitten hat und besiegt wurde.

**7. Sammelobjekte fallen lassen:** Die EnemyCollectableDrop Klasse ermöglicht es den Gegnern, zufällig Sammelobjekte fallen zu lassen, wenn sie besiegt werden. Die Wahrscheinlichkeit, dass ein Sammelobjekt fällt, kann individuell eingestellt werden. Dies fördert das Besiegen von Gegnern und belohnt den Spieler für seine Bemühungen.

**8. Boss-Gegner Tod:** Die EliteEnemyOnDeath Klasse definiert das Verhalten des Spiels, wenn ein Boss-Gegner besiegt wird. In diesem Fall wird das Spiel zum Hauptmenü zurückgeführt.

# Quellen