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 3](#_Toc169079564)

[Collectable 3](#_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 Koplinger – 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

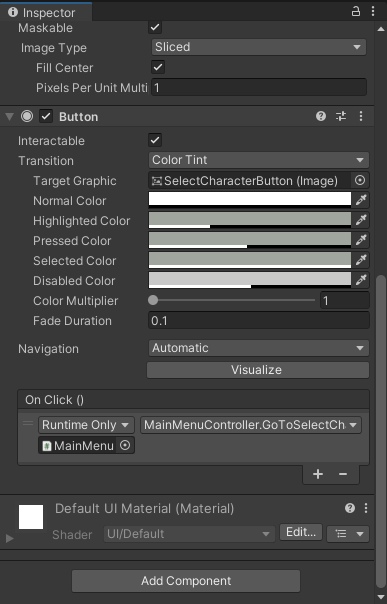
## Player

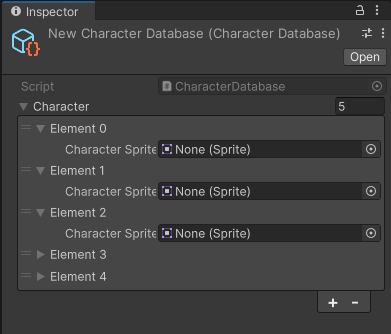
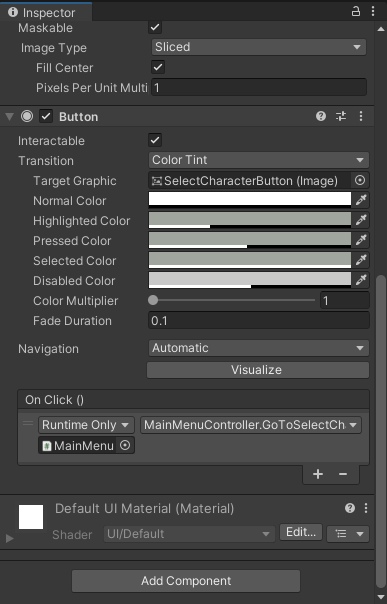
wad

## Collectable

Aaa

## Character-Design

****Beim Character Design ging es hauptsächlich um die Frage „Wie soll unser Spieler aussehen?“.  
Man kennt es in nahezu jedem Spiel, dass es die Möglichkeit gibt, sich das Aussehen des Spielers selbst auszusuchen. Meist muss man Münzen oder ähnliches verdienen, um alle Charaktere freischalten zu können. Diese Option könnte man in weitere Folge in „Pixel Purge“ implementieren, derzeit ist es dem Spieler jedoch frei, zu entscheiden mit welchem Charakter er nun tatsächlich spielen möchte.

**Aber wie implementiert man eine „Select Character“-Funktion nun tatsächlich?**Zuerst wurde eine neue GameScene, namens „Select Character“ erstellt und mit Buttons für das Navigieren durch die Scene erstellt. Danach wurde das Main-Menu um einen Button „Select Character“ erweitert. Dieser erhält im Inspector unter „OnClick()“ das MainMenu-Script, in dem programmiert ist, dass, wenn man auf den Button klickt, dass man zur SelectCharacter-GameScene geleitet wird. Weiters muss man in OneClick() dem Button die Funktion zuweisen, dass auf diese Scene gegangen werden soll, sobald dieser Button gedrückt wird..

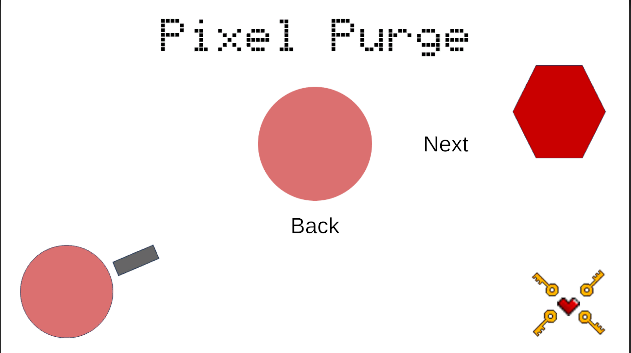
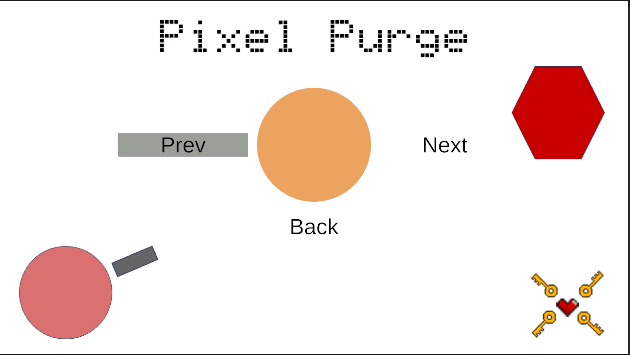
1 Character-Datenbank

**2** Hinzufügen Funktionen Button

Danach erfolgte die Erstellung einer Character-Datenbank, die mit Sprites gefüllt werden kann. In dieser Datenbank kann man im Inspector funktionell eingeben, wie viele Sprites man hinzufügen möchte bzw. die gewünschten Sprites können dann einfach hineingezogen werden. Somit sind diese dann in dieser Datenbank, auf die dann zugegriffen werden kann. Bevor wir uns als Team auf ein einheitliches Design des Spiels festgelegt hatten, war diese Datenbank mit verschiedensten Emoji-Sprites gefüllt, um zu testen, ob die Datenbank auch wirklich funktioniert.

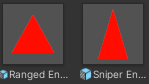
In weiterer Folge wurde in einem CharagterManager-Script die Funktionalitäten des Next-Buttons, des Previous-Buttons bzw. das Speichern und Updaten des ausgewählten Designs des Characters ausprogrammiert, um dem Spieler zu ermöglichen, dass auch wirklich mit dem ausgewählten Design gespielt werden kann. Den Next- bzw. Previous-Buttons mussten, wie dem „Select Character“-Button im Main-Menu, die nötigen Funktionen zugewiesen werden.

Nachdem feststand, wie unser Spiel und unsere Charaktere nun tatsächlich aussehen sollen, wurde die Implementierung der „Select Character“-Funktion nochmals überarbeitet und in einem neuen Script „SelectCharacter“ überarbeitet. In diesem Script wurden zuerst die Farben des Players definiert und in einer Variable gespeichert. Durch Erhöhung/Verringern eines Indizes, das durch die Betätigung der Prev-/Next-Button geschieht, verändert sich in der GameScene ebenfalls die Farbe des Characters bzw. bei Erreichung der Grenzen (Erster/letzter Character) verschwinden oder erscheinen die Prev-/Next-Buttons.

Neben dem Player, der einen Kreis darstellt, entschieden wir uns als Team, den Gegnern eckige Formen zu geben. Somit wurden den verschiedenen Gegnern in SpriteRenderer eine Hexagon-Form (Melee-Enemy) oder Dreiecks-Form (Ranged/Sniper-Enemy) bzw. dem Boss (Elite-Enemy) ein Zehnzackige-Sternenform hinzugefügt.

3 Beispiel "Select Character"-GameScene

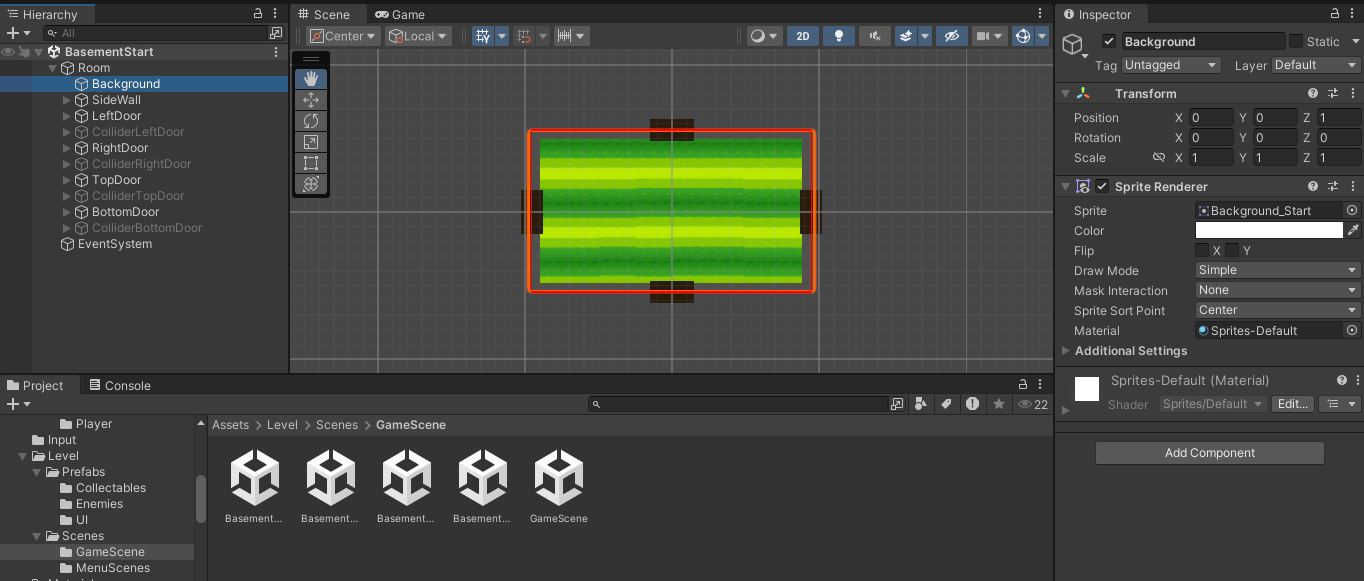
4 Beispiel "Select Character"-GameScene



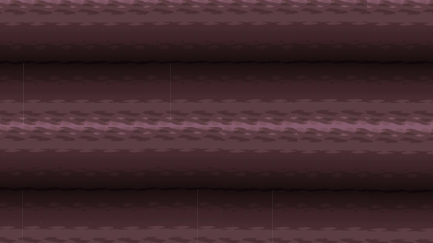
## Kartendesign

Hier ging es darum, dem Spiel in Form von passenden Hintergründen und Designanpassungen Leben einzuhauchen. Wir einigten uns als Team darauf, dass die Hintergründe des Spiels eher einfach gehalten werden sollten, um nicht vom eigentlichen Spielerlebnis abzulenken.

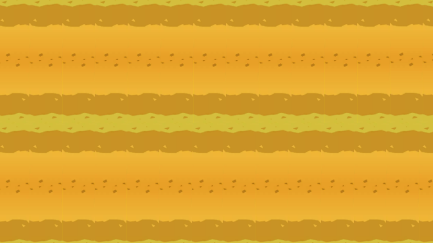
Da in unserem Spiel 3 „Basements“-Gamescenes (BasementStart: Spawnkarte des Players, BasementEmpty: Spawnpunkt der Gegner bzw. allgemeine Map und BasementEnd: Herrschaftsbereich des Endgegners „Pixel-Dominator“) vorhanden waren, die ein neues Design benötigten, mussten herfür 3 Hintergründe gestaltet werden.

Als erster Anhaltspunkt, wie die Hintergründe der 3 Basements aussehen könnten, diente die Internetseite „https://assetstore.unity.com“ (Siehe Quellen). Dort kann man sich etwas durchstöbern bis man findet, nach was man sucht. In diesem Fall ging es um eventuell passende Map-Hintergründe. Eine Filterfunktion, Unterscheidung zwischen kostenlosen und kostenpflichtigen Assets und eine Suchfunktion erleichtern die Suche. Schlussendlich wurden passende Hintergründe gefunden, welche noch etwas angepasst und in das passende Format der Basement-Gamescenes (1920 x 1080 Pixel) formatiert werden mussten. Danach wurden diese zuerst unter Assets -> Models -> Map zum Projekt eingefügt und folgend unter den jeweiligen „Basement“-GameScenes unter „Room“ -> „Background“ im SpriteRenderer als Sprite hinzugefügt.

5 Einfügen Hintergund im SpriteRenderer

Für den BasementStart wurde sich für ein grüner Hintergrund entschieden, welcher eine grasähnliche Atmosphäre und Ruhe ausstrahlen soll, weil der Kampf der „Formen“ noch nicht begonnen hat, da das Spiel hier mit dem Spawnen des Players beginnt. Sobald man durch einen der anliegenden Türen steigt, wird man auf Gegner treffen. Die BasementEmpty beginnt ebenfalls mit Durchschreiten einer beliebigen Tür. Daher wurde ein bräunlicher Hintergrund gewählt, der Wüste und somit einen harten Kampf auf trockenem Boden imitieren soll. Der 3 Raum, der ein Design wünschte, war das „BasementEnd“ – Das Königreich des Allianzanführers der Polygone. Dieser Raum wurde mit einem eher dunkleren, mystischen Hintergrund mit rötlich/violettem Design versehen, um dem Spieler das Gefühl zu geben, in einem höhlen- bzw. höllenähnlichem Lager eingedrungen zu sein.

6 Background\_End

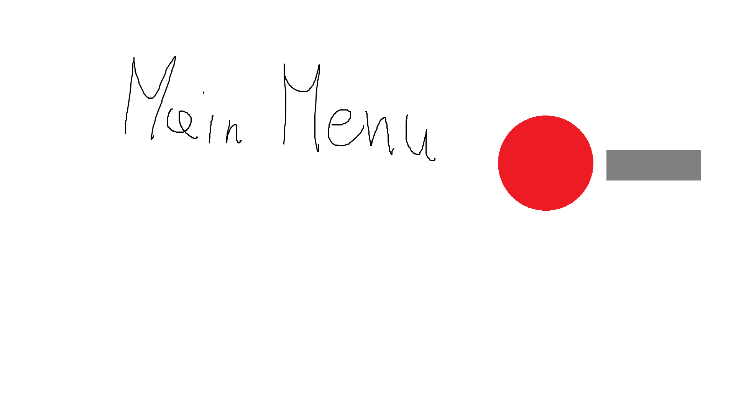
Weiters benötigten auch die Menu-Szenen eine Überarbeitung des Aussehens. Dazu wurde einfach ein weißer Hintergrund hergenommen und in Pixelschrift der Name des Spiels „Pixel-Purge“ und eine Abbildung des „Player-Characters“ der seine Waffe auf einen Gegner zielt, draufgesetzt. Im unteren rechten Eck, wurden die Schlüssel, die der Spieler während des Spiels einsammeln muss, um den Bossraum freizuschalten und ein Herz, welches nach Ableben der Gegner spawnen kann, eingefügt.

7 Background\_Empty

8 Background\_Start

Das Designen dieses Menu-Hintergrundes geschah ganz simpel in PowerPoint.

9 Menu-Design Nachher



10 Menu-Design Vorher

## Kartengeneration

A

## Hauptmenü

A

## Gegner

A

## Boss

A

# Quellen

<https://assetstore.unity.com/>

[Unity Character/Skin Selection Menu - Easy Unity Tutorial (youtube.com)](https://www.youtube.com/watch?v=2PKBChN10us&list=WL&index=3)