Programmieren eines 2D Spiels

Inhaltsverzeichnis

1 Projektidee

1.1 Unser Projektauftrag

1.2 Ziele:

1.3 Mögliche Probleme:

1.4 Begründung / Motivation für Themenwahl

2 SMART Ziele

2.1 Spezifisch

2.2 Messbar

2.3 Ausführbar

2.4 Realistisch

2.5 Terminiert

3 Informieren

3.1 Unsere Ziele

3.2 Tools zum Erstellen vom Game

- **Game Engines**: Game Engines sind Software-Plattformen, die es Entwicklern ermöglichen, Spiele zu erstellen und zu testen. Beispiele für Game Engines sind Unity, Unreal Engine und CryEngine.
- **2D-Modellierungstools**: 2D-Modellierungstools werden verwendet, um 2D-Modelle für Spiele zu erstellen. Beispiele für 2D-Modellierungstools sind Blender, Maya und 3ds Max.
- **Grafik-Editoren**: Grafik-Editoren werden verwendet, um Grafiken und Texturen für Spiele zu erstellen. Beispiele für Grafik-Editoren sind Adobe Photoshop und GIMP.
- **Audio-Editoren**: Audio-Editoren werden verwendet, um Audio-Dateien für Spiele zu bearbeiten und zu erstellen. Beispiele für Audio-Editoren sind Audacity und Adobe Audition.
- **Level-Design-Tools**: Level-Design-Tools werden verwendet, um die Levels in einem Spiel zu entwerfen und zu gestalten. Beispiele für Level-Design-Tools sind Tiled und Mappy.
- **Debugging-Tools**: Debugging-Tools werden verwendet, um Fehler in Spielen zu finden und zu beheben. Beispiele für Debugging-Tools sind GDB und Visual Studio Debugger.

3.3 Kenntnisse, die wir können müssen

Wir müssen uns bei diesen Kenntnissen besser informieren, um sie besser anzuwenden können:

Level-Design: Um die Levels in Ihrem Spiel zu entwerfen, um in der Lage sein, die Levels zu gestalten, Objekte zu platzieren und das Gameplay zu entwerfen.

2D-Modellierung: Um 2D-Modelle für Ihr Spiel zu erstellen und zu bearbeiten, Texturen zu erstellen und Materialien anzuwenden.

Audio-Editoren: Audio-Editoren werden verwendet, um Audiodateien zu bearbeiten und zu verändern. Mit einem Audio-Editor können Sie beispielsweise Musiktitel schneiden, Lautstärkeanpassungen vornehmen, Hintergrundgeräusche entfernen oder Audioeffekte hinzufügen.

Debugging-Tools: Debugging-Tools werden verwendet, um Fehler in Computerprogrammen zu finden und zu beheben. Wenn ein Programm nicht wie erwartet funktioniert oder abstürzt, können Debugging-Tools verwendet werden, um herauszufinden, warum das Programm nicht korrekt ausgeführt wird. Debugging-Tools bieten in der Regel Funktionen wie die Möglichkeit, den Code Schritt für Schritt auszuführen und Variablenwerte während der Ausführung anzuzeigen, um den Fehler zu identifizieren und zu beheben.

4 Planung

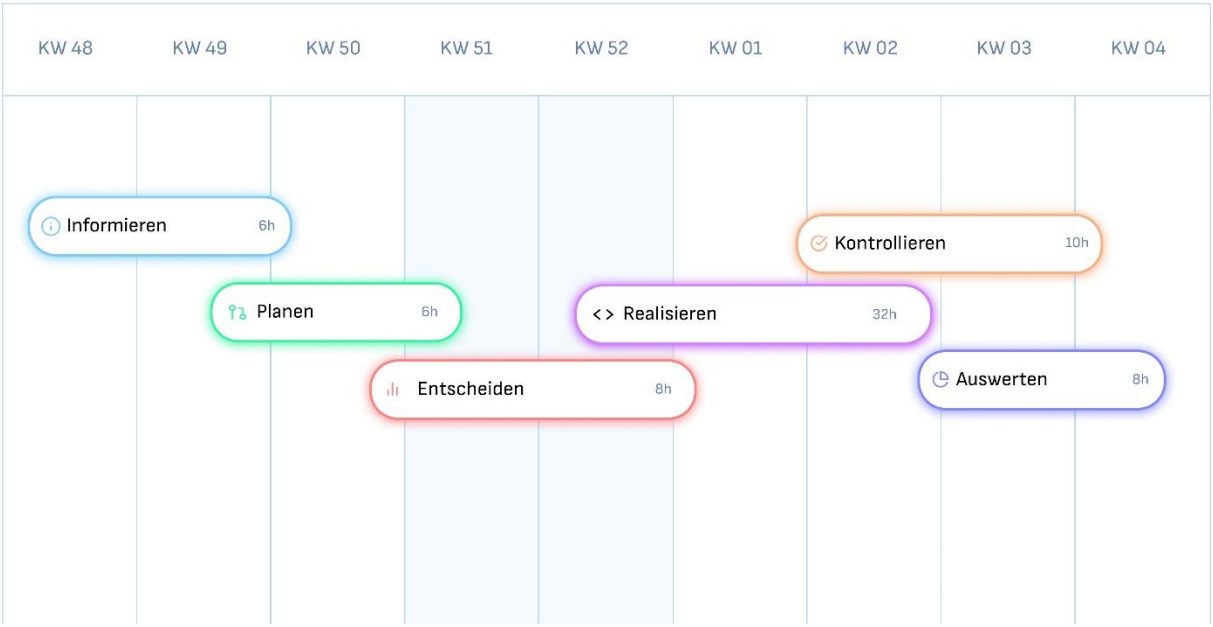
4.1 Zeit-Planung

Wir haben eine Excel Tabelle erstellt, damit wir unser Projekt erfolgreich meistern können.

Was?	Zeit?	Wer?	Start?	Ende?
Projekt definieren	30min	Jann und Nicolaj	02.12.2022	02.12.2022
Ziele definieren	45min	Jann und Nicolaj	02.12.2022	02.12.2022
Projektauftrag schreiben	2std	Jann	09.12.2022	16.12.2022
Informieren	4std	Jann und Nicolaj	09.12.2022	16.12.2022
Informieren schreiben	2std	Jann	16.12.2022	16.12.2022
Planen	4std	Jann und Nicolaj	09.12.2022	16.12.2022
Planung schreiben	2std	Jann	16.12.2022	23.12.2022
Entscheiden	6std	Nicolaj	23.12.2022	23.12.2022
Entscheidung schreiben	2std	Jann	13.01.2023	13.01.2023
Realisieren	32std	70% Nicolaj / 30% Jann	16.12.2022	27.01.2023
Kontrollieren	8std	Nicolaj	20.01.2023	20.01.2023
Kontrollieren festhalten	2std	Jann und Nicolaj	20.01.2023	27.01.2023
Auswerten	4std	Jann und Nicolaj	27.01.2023	27.01.2023
Auswerten festhalten	4std	Jann und Nicolaj	27.01.2023	03.02.2023
Dokumentation abschliessen	2std	Jann	20.01.2023	03.02.2023
Präsentation erstellen	2std	Jann und Nicolaj	27.01.2023	03.02.2023

4.2 Roadmap

Das ist unsere Roadmap zu unserer Zeitplanung.



4.3 Programmiersprachen

1. **C#** ist eine objektorientierte Sprache, die hauptsächlich für die Entwicklung von Windows-Anwendungen und Spiele mit der Unity-Engine verwendet wird.

```
using System;

namespace HelloWorld
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Hello World!");
    }
  }
}
```

4.4 Planung zum Realisieren

5 Entscheidung

5.1 Paarvergleiche

5.1.1 Kriterien Programmiersprachen

Für die Nutzwertanalyse der Programmiersprachen vergleichen wir erstmals 5 Kriterien.

0 = keinen Wert

1 = mittel Wert

2 = überlegen

Paarvergleich	Leistungs fähigkeit	Verbreitung	Skalierbar keit	Anpassbar keit	Lernkurve	Bewertung
Leistungs fähigkeit		2	1	1	1	25%
Verbreitung	0		0	0	0	0%
Skalierbar keit	1	2		0	1	20%
Anpassbar keit	1	2	2		2	35%
Lernkurve	1	2	1	0		20%

1. **Leistungsfähigkeit**: Diese Kategorie würde die Fähigkeit der Sprache untersuchen, Aufgaben schnell und effizient auszuführen, einschliesslich Faktoren wie Speicherverwaltung und CPU-Auslastung.

2. **Verbreitung**: Diese Kategorie würde die Beliebtheit und Verbreitung der Sprache in der Entwicklergemeinschaft untersuchen, einschliesslich der Verfügbarkeit von Ressourcen und der Anzahl der Entwickler, die die Sprache beherrschen.

3. **Skalierbarkeit**: Diese Kategorie würde die Fähigkeit der Sprache untersuchen, grosse und komplexe Projekte zu verwalten und zu skalieren.

4. **Anpassbarkeit**: Diese Kategorie würde die Flexibilität der Sprache untersuchen und wie leicht es ist, sie für verschiedene Anwendungen und Projekte anzupassen.

5. **Lernkurve**: Diese Kategorie würde die Lernkurve der Sprache untersuchen und wie leicht es ist, die Sprache für Entwickler zu erlernen und zu meistern.

5.1.2 Ansicht

Für die Nutzwertanalyse der Ansicht vergleichen wir erstmals 2 Kriterien.

0 = keinen Wert

1 = mittel Wert

2 = überlegen

Paarvergleich	Erlebnis	passend	Bewertung
Erlebnis		1	50%
passend	1		50%

5.1.3 Perspektive

Für die Nutzwertanalyse der Perspektive vergleichen wir erstmals 3 Kriterien.

0 = keinen Wert

1 = mittel Wert

2 = überlegen

Paarvergleich	Sichtbarkeit	Steuerbarkeit	Atmosphäre	Bewertung
Sichtbarkeit		0	1	16.6%
Steuerbarkeit	2		1	50%
Atmosphäre	1	1		33.4%

1. **Sichtbarkeit**: Dieser Faktor bezieht sich darauf, wie gut die Perspektive des Spiels die Spielumgebung und Hindernisse darstellt und wie leicht es für den Spieler ist, sich im Spiel zurechtzufinden.
2. **Steuerbarkeit**: Dieser Faktor bezieht sich darauf, wie gut die Perspektive des Spiels die Steuerung des Charakters ermöglicht und wie präzise und reaktionsschnell die Steuerung ist.
3. **Atmosphäre**: Dieser Faktor bezieht sich darauf, wie gut die Perspektive des Spiels die Atmosphäre des Spiels unterstützt und wie gut sie dazu beiträgt, das Spielerlebnis zu verbessern.

5.1.4 Programmierumgebung

Für die Nutzwertanalyse der Programmierumgebung vergleichen wir erstmals 3 Kriterien.

0 = keinen Wert

1 = mittel Wert

2 = überlegen

Paarvergleich	Performance	Benutzerfreundlichkeit	Kompatibilität	Bewertung

Performance		0	2	33.3%
Benutzerfreundlichkeit	2		2	66.6%
Kompatibilität	0	0		0%

1. **Performance**: Die Applikation sollte schnell und effizient sein, um eine reibungslose Spielerfahrung zu gewährleisten.
2. **Benutzerfreundlichkeit**: Die Applikation sollte einfach zu verwenden und intuitiv sein, damit die Spieler schnell und einfach Paare vergleichen können.
3. **Kompatibilität**: Die Applikation sollte mit verschiedenen Plattformen und Geräten kompatibel sein, um eine breite Anwenderbasis anzusprechen.

5.2 Assets

Die Assets → Art, Audio, 3d models, Animations, Scripts, Textures und Shaders Scenes haben wir entschieden, dass wir die vorgegebenen von Unity nehmen. Wichtig war uns, dass wir einen Player haben und dass es 3d Models beinhaltet.

5.3 Scenes

Die Gameplay Scenes sind vorgegeben → wir können entscheiden, was die Scenes beinhalten, jedoch nicht was es für Scenes gibt. Jedes Spiel soll → Main Menu, Gameplay Scene, Pause Menu und Game Over Menu beinhalten.

5.4 Allgemeine Entscheidung

- Programmiersprache → C#
- Ansicht → 3d
- Programmierumgebung → Unity
- Dokumentation → Microsoft Word
- Perspektive → Ego Person
- Assets → Art, Audio, 3d models, Animations, Scripts, Textures, Shaders
- Scenes → Main Menu, Gameplay Scene, Pause Menu, Game Over Menu

5.5 Nutzwertanalyse Programmiersprachen

		Javascript	Java	C#	Python
Bewertung 1-5	**Gewichtung**	**Bewertung Nutzwert**	**Bewertung Nutzwert**	**Bewertung Nutzwert**	**Bewertung Nutzwert**
Leistungsfähigkeit	**25%**	**4 | 1**	**3 | 0.75**	**4 | 1**	**3 | 0.15**
Skalierbarkeit	**20%**	**5 | 1**	**5 | 1**	**4 | 0.8**	**3 | 0.6**
Anpassbarkeit	**35%**	**4 | 1.4**	**4 | 1.4**	**4 | 1.4**	**3 | 1**
Lernkurve	**20%**	**4 | 0.8**	**3 | 0.6**	**5 | 1**	**3 | 0.6**
Gesamtnutzwert		**4.2☆**	**3.75☆**	**4.2☆**	**2.35☆**
Ranking		**1**	**2**	**1**	**3**

Unsere Nutzwertanalyse zeigte uns, dass wir mit der Programmiersprache C# programmieren werden. Wir hätten auch JavaScript nehmen können, jedoch wollten wir wieder etwas Neues ausprobieren mit C#.

5.6 Nutzwertanalyse Ansicht

		2d	3d
Bewertung 1-5	**Gewichtung**	**Bewertung | Nutzwert**	**Bewertung | Nutzwert**
Erlebnis	**50%**	**3 | 1.5**	**5 | 2.5**
passend	**50%**	**3 | 1.5**	**4 | 2**
Gesamtnutzwert		**3☆**	**4.5☆**
Ranking		**2**	**1**

Unsere Nutzwertanalyse zeigte uns, dass wir mit einer 3d Ansicht arbeiten werden.

5.7 Nutzwertanalyse Perspektive

		Isometrisch	Ego	Third person
Bewertung 1-5	**Gewichtung**	**Bewertung | Nutzwert**	**Bewertung | Nutzwert**	**Bewertung | Nutzwert**
Sichtbarkeit	**16%**	**3 | 0.5**	**4 | 0.7**	**3 | 0.5**
Steuerbarkeit	**50%**	**3 | 1.5**	**4 | 2**	**5 | 2.5**
Atmosphäre	**33%**	**2 | 0.7**	**5 | 1.7**	**3 | 1**
Gesamtnutzwert		**2.7☆**	**4.4☆**	**4☆**
Ranking		**3**	**1**	**2**

Unsere Nutzwertanalyse zeigte uns, dass wir mit der Ego Perspektive unser Spiel realisieren werden

5.8 Nutzwertanalyse Programmierumgebung

		VSC	Unity	Intellij DIE
Bewertung 1-5	**Gewichtung**	**Bewertung | Nutzwert**	**Bewertung | Nutzwert**	**Bewertung | Nutzwert**
Performance	**33%**	**4 | 1.3**	**4 | 1.3**	**3 | 1**
Benutzerfreundlichkeit	**66%**	**5 | 3.3**	**5 | 3.3**	**4 | 2.6**
Gesamtnutzwert		**4.6☆**	**4.6☆**	**3.6☆**
Ranking		**1**	**1**	**2**

Unsere Nutzwertanalyse zeigte uns, dass wir mit Unity programmieren werden.

6 Realisation

6.1 Entwicklerumgebung Unity

Unity ist eine plattformübergreifende Spiel-Engine und -Entwicklungsumgebung, die von Unity Technologies entwickelt wurde. Sie wird hauptsächlich von Spielentwicklern verwendet, um 2D- und 3D-Spiele für PC, Konsolen, mobile Geräte und Web-Browser zu entwickeln und zu veröffentlichen.

1 2D-Spiel in Unity

2 3D-Spiel in Unity

Unity bietet eine Vielzahl von Werkzeugen und Funktionen, die es Entwicklern ermöglichen, Spiele schnell und effektiv zu erstellen. Dazu gehören eine visuelle Editor-Umgebung, ein integrierter Skript-Editor, ein Animationssystem und Unterstützung für eine Vielzahl von Grafik- und Audio-Formaten.

Unity wird auch von vielen anderen Arten von Entwicklern verwendet, um interaktive Inhalte wie Simulationen, Prototypen und Prototypen zu erstellen. Es hat auch eine grosse und aktive Community, die hilfreiche Ressourcen und Support bietet.

6.2 Entwicklersprache C# («C Sharp»)

C# (gesprochen «C Sharp») ist eine objektorientierte Programmiersprache, die von Microsoft entwickelt wurde. Sie wird häufig für die Entwicklung von Anwendungen und Spielen verwendet, insbesondere für Windows- und Xbox-Plattformen.

Einige der wichtigsten Merkmale von C# sind:

- **Typisierung:** C# ist eine typisierte Sprache, was bedeutet, dass jeder Variablen ein bestimmter Datentyp zugewiesen wird. Dies hilft, Fehler zu vermeiden und die Lesbarkeit des Codes zu verbessern.
- **Objektorientierung:** C# unterstützt die objektorientierte Programmierung, bei der der Code in "Klassen" organisiert wird, die Eigenschaften und Verhaltensweisen definieren. Dies ermöglicht es, komplexe Systeme zu bauen und wiederverwendbar zu machen.
- **Ereignisse und Delegaten:** C# unterstützt das Konzept von Ereignissen und Delegaten, die es ermöglichen, dass sich verschiedene Teile des Codes miteinander verbinden und auf bestimmte Ereignisse reagieren können.

C# ist eine bekannte und vielseitige Programmiersprache, die in vielen Bereichen eingesetzt wird, von der Entwicklung von Anwendungen und Spielen bis hin zur Steuerung von Industrieprozessen und der Datenanalyse. Sie bietet Entwicklern zahlreiche Werkzeuge und Funktionen, um leistungsstarke und effektive Software zu erstellen.

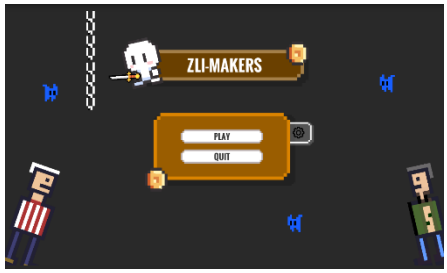
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    public float speed = 5.0f;
    public float turnSpeed;
    public float horizontalInput;
    public float forwardInput;

    void Start()
    {

    }



    void Update()
    {
        horizontalInput = Input.GetAxis("Horizontal");
        forwardInput = Input.GetAxis("Vertical");

        transform.Translate(Vector3.forward * Time.deltaTime * speed * forwardInput);
        transform.Rotate(Vector3.up, turnSpeed * horizontalInput * Time.deltaTime);
    }
}
```

3 Beispiel von C# Code

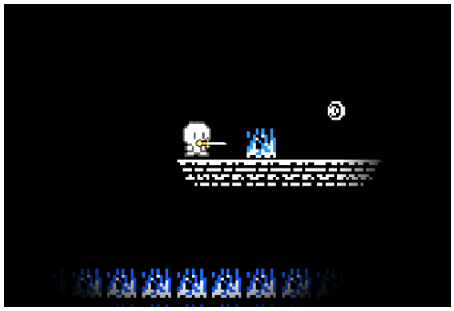
6.3 Scenes

Scenes werden in Videospielen verwendet, um das Gameplay, die Handlung und die Atmosphäre des Spiels zu strukturieren und zu gestalten. Sie helfen dabei, das Spielerlebnis für den Spieler abwechslungsreicher und interessanter zu gestalten, indem sie verschiedene Aspekte des Spiels hervorheben und unterschiedliche Gameplay-Elemente und Erfahrungen bieten.

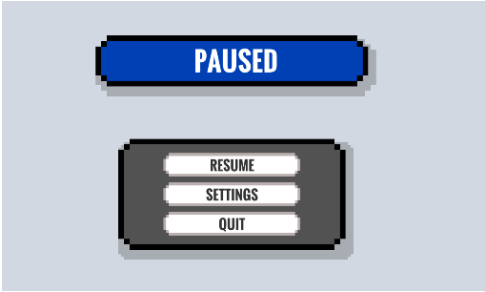
1. **Das Hauptmenü** ist in der Regel das erste, was der Spieler sieht, wenn er ein Videospiel startet. Es bietet dem Spieler verschiedene Optionen wie das Starten eines neuen Spiels, das Laden eines gespeicherten Spiels, das Anpassen von Einstellungen und das Verwalten von Inventar.



2. **Die Gameplay-Szene** ist der Abschnitt des Spiels, in dem der Spieler tatsächlich spielt und die Kontrolle über den Spielcharakter hat. In dieser Szene werden die Gameplay-Elemente wie Kämpfe, Rätsel und Exploration dargestellt.



3. **Das Pausenmenü** ist eine Szene, die angezeigt wird, wenn der Spieler das Spiel pausiert. In diesem Menü kann der Spieler Optionen wie das Speichern des Spiels, das Verlassen des Spiels oder das Fortsetzen des Spiels auswählen.



4. **Das Game Over-Menü**
   ist eine Szene, die angezeigt wird, wenn der Spieler das Spiel verloren hat oder das Spiel beendet wurde. In diesem Menü kann der Spieler Optionen wie das Neustarten des Spiels oder das Verlassen des Spiels auswählen.

6.4 Assets

Wir wollen ein Probuilder Asset Package verwenden. Nun stellt sich die Frage, was es uns ermöglicht und wo es uns hilft. ProBuilder ist ein Unity-Asset, welches uns ermöglicht**, 3D-Geometrie** schnell und einfach in Unity zu erstellen und zu bearbeiten. Es bietet **Werkzeuge zum Erstellen von Formen**, zum Bearbeiten von **Mesh-Modellen** und zum **Erstellen von UV-Koordinaten**. Es ermöglicht uns auch, ihre Modelle in Unity zu importieren und zu exportieren, ohne dass ein externes Modellierungsprogramm erforderlich ist. ProBuilder ist ein flexibles Werkzeug, das die Entwicklung von 3D-Spielen und Anwendungen in Unity erheblich beschleunigen kann.

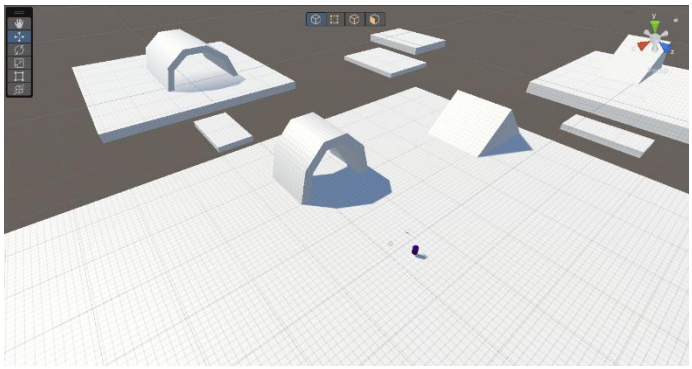
1. **2D-Geometrie**

3D-Geometrie in Unity bezieht sich auf die Erstellung und Verwendung von 3D-Modellen und Formen innerhalb des Unity-Spiel-Engines. Es ermöglicht Entwicklern, 3D-Umgebungen und -Objekte zu erstellen, die in ihren Spielen und Anwendungen verwendet werden können.

2. **Mesh-Modelle**

Ein Mesh-Modell in Unity ist eine Sammlung von Punkten, auch als Vertices bezeichnet, die miteinander verbunden sind, um eine 3D-Form darzustellen. Mesh-Modelle können aus vielen verschiedenen Quellen importiert werden, wie z.B. von 3D-Modellierungssoftware wie Blender oder Maya. Sie können verwendet werden, um 3D-Objekte und Umgebungen in Unity-Spielen und Anwendungen darzustellen.

Mit ProBuilder können wir Mesh-Modelle erstellen, indem wir Formen wie **Würfel**, **Zylinder** und **Kugeln** erzeugen und diese **bearbeiten**, indem wir die **Vertices**, **Kanten** und **Polygone hinzufügen oder entfernen**.



4 Unsere Testmap mit einem Mesh-Modell

6.5 Montag

Zusammen haben wir ein Videospiel entwickelt, indem jeder von uns einen wichtigen Beitrag geleistet hat. Philip hat dafür gesorgt, dass der Hintergrund ohne Unterbrechung läuft. Jann hat die Haupt- und Game Over-Szenen erstellt und auch einen Testteil für das erste Level programmiert. Eris hat eine komplexe KI für die Gegner entwickelt. Patrick hat sich um die Bewegung des Spielers gekümmert und auch dafür gesorgt, dass das Game Over Fenster eingeblendet wird.

Es ist bemerkenswert, wie gut wir zusammengearbeitet haben und jeder von uns seine Stärken einbringen konnte, um ein erfolgreiches Spiel zu schaffen. Unsere Zusammenarbeit hat uns ermöglicht, ein gut durchdachtes und spaßiges Spiel zu entwickeln. Wir haben gezeigt, dass wir als Team stärker sind als als Einzelpersonen.

was haben wir (eris, patrick, philip und jann) gemacht?

- philip hat den Hintergrund zum Laufen gebracht (endless loop)

- jann hat mainscene und gameoverscene gemacht

- jann hat einen testteil zum ersten level gemacht

- eris hat eine enemy ai programmiert

- patrick hat spieler moving gemacht

- patrick hat das gameover fenster einblenden lassen

6.6 Dienstag

Als Team haben wir an einem Videospiel gearbeitet, bei dem jeder von uns einen wichtigen Beitrag geleistet hat. Eris hat die KI für die Gegner abgeschlossen, während Jann die Gegner zeichnete und für die Navigation im Main Menu und Game Over Bereich verantwortlich war. Patrick hat sich um die Bewegung des Spielers gekümmert und das Leveldesign verbessert. Philip hat Partikeleffekte und den Hintergrund erstellt und nach passenden Assets gesucht. Insgesamt haben wir gezeigt, dass wir

als Team erfolgreich arbeiten können und jeder von uns seine Stärken einbringt, um ein unterhaltsames Spiel zu schaffen.

was haben wir (eris, patrick, philip und jann) gemacht?

- eris hat die enemy ai abgeschlossen

- jann hat die gegner (Jörg und Fabian) gezeichnet

- jann hat die mainmenu navigation zum "play" erstellt, dass man auf das erste level kommt

- jann hat die gameover navigation zum "return" gemacht, dass man auf das mainmenu kommt

- patrick hat movement gemacht

- patrick hat das leveldesign ein wenig verbessert

- philip hat particles erstellt

- philip hat den background erstellt

- eris hat ein leveldesign angefangen

- philip hat nach passenden assets gesucht

6.7 Mittwoch

Heute konnte Eris erfolgreich den Gegner im Spiel so programmieren, sodass er stirbt, wenn er von dem Spieler berührt wird. Darüber hinaus hat Eris dem Gegner eine Animation hinterlegt. Unser Teamkollege Jann hat die Buttons und Menüs aktualisiert, Patrick hat die Movements erweitert und Philipp hat die Beleuchtung eingerichtet. Alles in allem haben wir heute große Fortschritte gemacht und alles verlief wie geplant. Jetzt müssen wir nur noch die KI und das Balancing des Spiels optimieren, damit es ein reibungsloses Spielerlebnis bietet. Wir haben uns auch bereits Gedanken gemacht, wie wir neue Levels und Waffen hinzufügen können, um das Spiel noch spannender zu gestalten. Es ist aufregend zu sehen, wie unser Spiel Tag für Tag Fortschritte macht und bald bereit für die Vorstellung.

6.8 Donnerstag

Heute waren wir etwas chaotischer als gewöhnlich. Unser aktueller Auftrag und Fokus liegt darin, unsere bereits programmierten Level miteinander zu verbinden. Dafür sind Jann und Patrick zuständig. Patrick hat es heute geschafft, unserem Spieler ein Schwert hinzuzufügen, welches man mit der Taste X nutzen kann. Phillip arbeitet derzeit an unserem Gegner und Eris hat ein drittes Level auf die Beine gestellt.

6.9 Freitag

Heute waren wir leider nur zu zweit im Team. Phillip war krank, und Jann war an einem Betriebsevent. Trotz all dem, haben der Eris und Patrick versucht das Beste daraus zu machen. Dementsprechend lag heute der Fokus hauptsächlich die Endgegner zu programmieren, respektive den Jörg und Fabian. Wichtig war, dass wir das Skript der beiden Gegner geschrieben hatten, und erst danach die Szenen bzw. Umgebung erstellen. Zusätzlich haben wir nebendran dokumentiert und dementsprechend Dokumentation geschrieben.

Ziele vom Montag und Dienstag:

- Testlevel erstelle
- Mainmenu erstellen
- Movement implementieren
- Github-Integration mit Unity
- AI für Gegner implementieren
- Grobes Level fertigstellen
- Navigation zwischen Szenen implementieren
- Game Over-Bildschirm bei Tod einblenden
- Partikeleffekte erstellen
- Gegner gestalten
- AI für Gegner abschließen

7 Kontrolle

7.1 Blackbox Testing

7.2 Whitebox Testing

7.3 Personen Testing

7.3.1 Feedback von den Eltern von Jann Fanzun

7.3.2 Feedback von der Schwester von Nicolaj Haueter

8 Auswertung

Wo entstanden Probleme?

Zu Beginn unseres Projekts hatte unsere Gruppe Schwierigkeiten mit dem User Interface von Unity, da es einige Zeit dauert, sich mit der Bedienung von Unity vertraut zu machen und auch Vorkenntnisse erfordert. Wir mussten uns zuerst mit den verschiedenen Funktionen und Werkzeugen von Unity auseinandersetzen, bevor wir unsere Ideen umsetzen konnten. Darüber hinaus gab es Probleme beim Verbinden der einzelnen Levels, um sie zu einem zusammenhängenden Spiel zu machen. Wir müssen zugeben, dass die Gruppe in der zweiten Woche beträchtliche Schwierigkeiten bei der Umsetzung unseres Projekts hatte. Wir haben uns jedoch nicht entmutigen lassen und haben hart daran gearbeitet, die Probleme zu lösen und das Projekt zum Abschluss zu bringen. Wir haben uns gegenseitig unterstützt und uns bemüht, unsere individuellen Fähigkeiten zu nutzen, um gemeinsam ein erfolgreiches Projekt zu erstellen. Wir haben uns auf die Herausforderungen konzentriert und haben uns nicht entmutigen lassen, als wir auf Hindernisse gestoßen sind. Durch unsere gemeinsame Arbeit konnten wir unser Wissen und unsere Fähigkeiten im Umgang mit Unity und der Programmierung in C# verbessern. Wir haben gelernt, wie wichtig es ist, ein Teamplayer zu sein und wie man gemeinsam an Problemen arbeitet, um Lösungen zu finden. Insgesamt war das Projekt eine wertvolle Erfahrung, die uns gezeigt hat, wie wichtig es ist, durchzuhalten und hart zu arbeiten, um unsere Ziele zu erreichen. Wir haben unsere Fähigkeiten im Umgang mit Unity und der Programmierung in C# verbessert und sind stolz darauf, was wir gemeinsam erreicht haben.

Was ist uns gut gelungen?

Wir sind stolz darauf, dass uns das Design des Startbildschirms für unser Projekt besonders gut gelungen ist. Jann hat hier eine großartige Arbeit geleistet und ein sehr kreatives und kunstvolles Design entworfen, das perfekt zu unserem Projekt passt. Wir sind auch sehr zufrieden mit unserer Programmierung in C#. Obwohl wir zu Beginn einige Schwierigkeiten hatten, konnten wir uns schnell in die Programmierung einarbeiten und unser Wissen stetig erweitern. Wir haben uns gegenseitig unterstützt und unsere Fähigkeiten in der Programmierung verbessert, um ein erfolgreiches Projekt zu erstellen. Unsere erfolgreiche Umsetzung des Startbildschirms und der Programmierung in C# sind das Ergebnis unseres harten Arbeitens und unseres Einsatzes für das Projekt. Wir haben uns gegenseitig motiviert und uns bemüht, das Beste aus unseren Fähigkeiten herauszuholen, um ein Projekt zu erstellen, auf das wir stolz sein können. Insgesamt waren das Design des Startbildschirms und die Programmierung in C# wichtige Aspekte unseres Projekts, die zum Erfolg unseres Projekts beigetragen haben. Wir haben viel gelernt und unsere Fähigkeiten im Umgang mit Unity und C# verbessert. Wir sind stolz auf unsere Leistungen und werden diese Erfahrungen in zukünftigen Projekten anwenden.

Was haben wir gelernt?

Im Rahmen dieses Projekts haben wir viel über objektorientiertes Programmieren mit C# gelernt. Jeder in unserer Gruppe hat nun ein solides Verständnis von Unity und den Grundlagen von C#. Darüber hinaus haben wir auch wichtige Erfahrungen im Bereich Projektplanung gesammelt und diese erfolgreich in die Tat umgesetzt. Diese Erfahrung wird uns in Zukunft helfen, unsere Projekte noch besser zu planen und effektiver umzusetzen. Während des Projekts haben wir uns auf die Entwicklung einer bestimmten Anwendung konzentriert, die uns die Möglichkeit bot, die verschiedenen Aspekte des Programmierens mit C# und Unity kennenzulernen. Wir haben unsere Programmierkenntnisse erweitert und konnten uns dadurch ein fundiertes Verständnis von objektorientierter Programmierung aneignen. Unsere Teammitglieder haben nun das notwendige Wissen und die Fähigkeiten, um die Programmierung in C# und die Nutzung von Unity eigenständig und effektiv umzusetzen. Darüber hinaus konnten wir unsere Fähigkeiten in der Projektplanung verbessern, indem wir eine gründliche Planung und Durchführung unseres Projekts durchgeführt haben. Dies hat uns nicht nur geholfen, unser Projekt erfolgreich abzuschließen, sondern wird uns auch in zukünftigen Projekten von Vorteil sein. Insgesamt war dieses Projekt eine hervorragende Möglichkeit, um unsere Fähigkeiten im Bereich der objektorientierten Programmierung mit C# und Unity sowie in der Projektplanung zu verbessern. Wir sind stolz auf das, was wir erreicht haben, und sind zuversichtlich, dass wir unsere Fähigkeiten und Erfahrungen in zukünftigen Projekten erfolgreich einsetzen werden.