

Dokumentation

Draconis occisio

Paul Fuchs

54584

MI20w2-B

Dozenten/Prüfer: Prof. Dr. Marc Ritter, Manuel Heinzig

24.08.2022

Gliederung

- Spielkonzept
- Herausforderungen
 - Bewältigung
- Software-Architektur
- Implementierungsdetails
 - Schild
 - Damage Over Time
 - Healthpackage
- 2D-Art
- Menüstruktur
- Spielanleitung
 - Bewegung
 - Fähigkeiten
 - Cam-Lock
 - Menü
- Installationsanleitung
- Ausblick/Fazit
- Quellen

Spielkonzept

Dein Ziel ist es, durch das Besiegen mehrerer Drachen, Punkte für einen Highscore zu sammeln. Dabei ist jedem Drachen ein fester Wert und ein zufälliger Bonuswert zugeordnet, dies wird im Abschnitt [Herausforderungen](#) beschrieben. Deine drei besten Highscores werden lokal auf deinem PC gespeichert. Du startest mit einem Zeitpuffer von 5 Minuten und pro Kampf hast du 3 Minuten und 30 Sekunden Zeit einen Drachen zu besiegen. Solltest du erfolgreich sein, wird die restliche Zeit zu deinem Zeitpuffer hinzugaddiert. Wenn du es nicht schaffst, den Drachen zu besiegen, werden dir 3 Minuten und 30 Sekunden von deinem Zeitpuffer abgezogen. Du hast das Spiel verloren, wenn dein Zeitpuffer aufgebraucht ist oder deine Lebenspunkte auf 0 gesunken sind. Für den Kampf stehen dir 2 Zauber zur Verfügung: ein Schild zur Verteidigung und der Feuertornado für den Angriff

Herausforderungen

Im "Play"-Menü kannst du zwischen vier unterschiedlichen Drachen oder einer zufälligen Reihenfolge auswählen. Dabei hat jeder Drache unterschiedliche Mechaniken.

Name	Souleater	Nightmare	Usurper	Terrorbringer
Bild	A dark purple, multi-headed dog-like creature with glowing purple eyes and a large, jagged mouth.	A small, brown, horned creature with a wide, toothy grin and large, bulging eyes.	A large, brown, bat-like creature with long, spiky wings and a long, forked tail.	A green, scorpion-like creature with multiple legs, a segmented body, and a large, stinger-tipped tail.
Eigenschaften	Bewegung am Boden Schießt einen kleinen Feuerball mit geringem Schaden	Bewegung am Boden Schießt einen Tornado mit mittlerem Schaden	Bewegung in der Luft Schießt einen Tornado mit mittlerem Schaden	Bewegung in der Luft Schießt drei Tornados mit jeweils mittlerem Schaden
Punkteberechnung	$90 + 100 * (0 - 0.3)$	$120 + 100 * (0 - 0.45)$	$240 + 100 * (0 - 0.75)$	$160 + 100 * (0 - 0.45)$

Wenn dich ein Drache mit einer Fernkampf-Attacke trifft, werden dir Lebenspunkte abgezogen. Genauso verlierst du Lebenspunkte, wenn du einen Drachen berührst.

Bewältigung

Verlorene Lebenspunkte kannst du durch Einsammeln von Heilungspaketen wiederherstellen. Auf der Tribüne sind fünf Heilungspakete verteilt, diese erscheinen, 45 Sekunden nachdem sie eingesammelt wurden, wieder.

Software Architektur

Zum UML-Diagramm:

[Klassendiagramm](#)

Zur Dokumentation:

[Docs](#)

Ich habe mich dazu entschieden, die Klassen / Monobehaviour basierend auf der Verwendung im Spiel zu ordnen, so ist bspw.:

- der *PlayerTornado* im *Player* Ordner
und
- der *EnemyTornado* im *Enemy* Ordner

statt einer Differenzierung in der Funktionalität, wie es

- der *PlayerTornado* im *Projectile* Ordner
- der *EnemyTornado* im *Projectile* Ordner

wäre.

Implementierungsdetails

Code

Das Grundprinzip der Animationhandler für Spieler und Drachen habe ich aus dem folgenden Video übernommen:

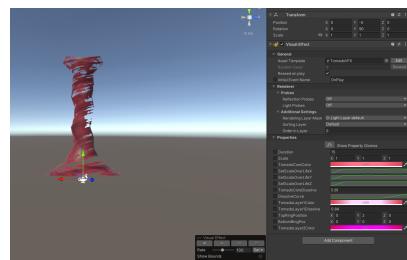
[Animate-like-a-programmer](#)

An dieser Stelle möchte ich ein besonderes Augenmerk auf die [ReferenceTable](#) legen. Diese hält die statischen Referenzen auf den [GameManager](#), den Spieler und den momentan lebenden Drachen und ist somit verantwortlich für die Target-locked Camera.

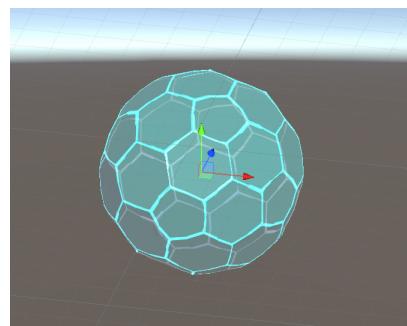
VFX

Folgende VFX sind nach Tutorials entstanden:

- Tornado



- Schild



Schild

Damit das Schild seinen Zweck erfüllt, habe ich Layer verwendet. Das Monobehaviour des Drachentornado überprüft zunächst, ob das kollidierte Objekt auf dem Schild Layer liegt und wenn dies der Fall ist, zerstört sich der Drachentornado selbst.

```
public void OnTriggerEnter(Collider other) {  
    if (other.gameObject.layer ==  
        LayerMask.NameToLayer("Shield"))  
        StartCoroutine(DestroyThis());
```

```

    }

    public IEnumerator DestroyThis(){
        Destroy(transform.parent.gameObject);
        yield return new WaitForSeconds(.3f);
    }

```

Damage over Time Effekt

Da Feuer brennt, hat der Tornado einen Damage over Time Effekt. Der Algorithmus dazu liegt im Health beschrieben:

```

private IEnumerator GetDamaged(int value, float time, int
tickRate) {
    var tickTime = time / tickRate;
    var actualTickRate = tickRate;

    while (tickRate > 0) {
        yield return new WaitForSeconds(tickTime);
        GetDamagedInstantly(value / actualTickRate);
        tickRate--;
    }
    GetDamagedInstantly(value % actualTickRate);
}

```

Beweis:

$(x // 3) * 3 + x \bmod 3 = x$
 $\rightarrow // - \text{ Floor Division}$

- $50 // 3 = 16$

- $16 * 3 = 48$

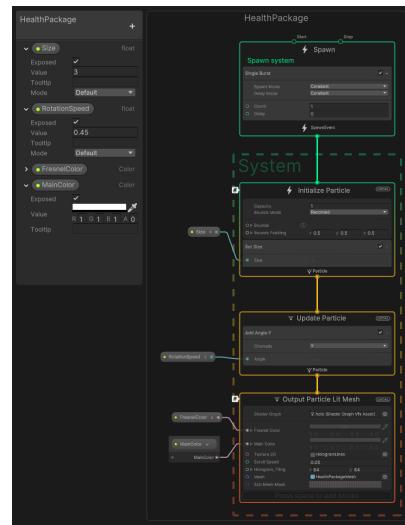
- $50 \bmod 3 = 2$

- $48 + 2 = 50$ q.e.d.

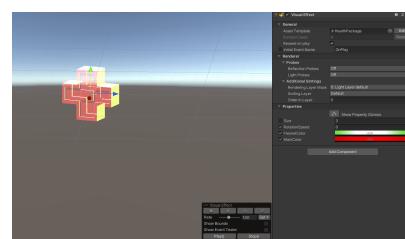
Healthpackage

Der Hologram Shader ist nach einem Tutorial entwickelt. Nachdem ich die VFX für das Feuer und den Tornado anhand von Tutorials erstellt habe, wollte ich den VFX für das Healthpackage selbst erstellen, um mein Erlerntes anzuwenden.

Der Graph:

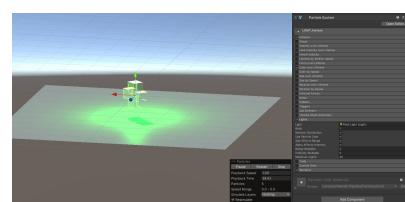


Output:



Einstellbar sind die Größe des angezeigten Objekts, die Geschwindigkeit mit der sich das Paket um die eigenen vertikale Achse dreht, die Fresnel-Color, also die Farbe mit der Hologram-Linien dargestellt werden und die Main-Color.

Lighting:



Um dem Spieler intuitiv zu zeigen, dass es sich um ein Objekt handelt, das eingesammelt werden kann und ein kleines Highlighting zu geben, da die

Heilungspakete von der Ausgangsposition des Spielers nicht zu sehen sind, habe ich das Partikel-System aus dem Feuer-Tutorial, welches das Lighting verursacht, auf die Farben des Heilungspakets abgestimmt und wiederverwendet.

Sound

Die Sound-Effekte beim Treffen eines Gegners, selbst getroffen werden und Einsammeln eines Healthpackages habe ich mit Audacity erstellt.

2D Art

Da ich den Großteil der 3D-Modelle aus dem Unity Assetstore verwendet habe, wollte ich die 2D Art selbst erstellen. So sind die Zeichnungen und Bilder entweder mit Sketchbook gezeichnet oder mittels Screenshot aufgenommen und später mit GIMP nachbearbeitet worden.

Lebensanzeige

Nach einigen Skizzen für eine Lebensanzeige und Feedback von Freunden, ist aus dem Balken

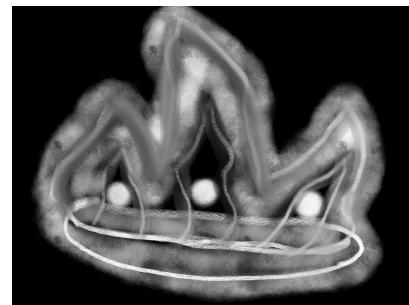


eine Herz-Anzeige geworden.



Flammentextur

Um die Alpha Details des Torch-Shader auszunutzen, habe ich eine schwarz-weiß-graue Textur für die Flamme gezeichnet.

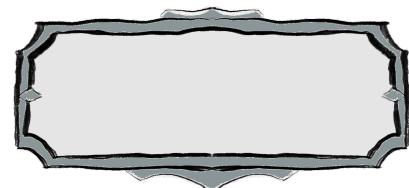


Button

Nachdem ich mir eine Menüstruktur und -layout überlegt hatte, habe ich die Sprites für die Buttons gezeichnet. Nach einfachen, symmetrischen Rechtecken

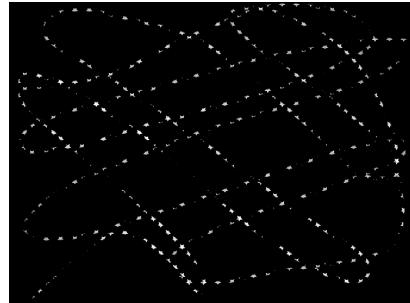


habe ich ein paar Verzierungen hinzugefügt, um dem mittelalterlichen Setting des Spiels gerecht zu werden.



Skybox

Letztlich habe ich noch eine Textur für die Skybox gezeichnet.



Menüstruktur

Ich habe mich dazu entschieden, die Buttons im Startbildschirm des Spiels auf der linken Seite des Bildschirms zu platzieren, da das Auswahl-Menü des *Play*-Buttons dazu gehört, ist die Platzierung hier einheitlich. Um den Platz auf der rechten Seite zu füllen, habe ich eine Collage aus Zauberer gegen Drache erstellt.



Im Pausemenü gibt es nur drei Auswahlmöglichkeiten: Weiterspielen, Optionen und zurück zum Hauptmenü. Da der Spieler den Blick während des Spiels in der Mitte des Bildschirms hat, sind auch die Buttons im Pausemenü in der Mitte angeordnet.



Im Optionsmenü gibt es nur eine geringe Anzahl von Einstellungen, weshalb die Slider in der Mitte sind. Somit sieht der Spieler direkt, was er einstellen kann.



Spielanleitung

Du findest dich als Zauberer mit einer Third-Person Perspektive in einer Arena wieder und anhand deiner Auswahl kämpfst du gegen einen oder unterschiedlich wiederkehrende Drachen.

Bewegung

Mit den Tasten **WASD** kannst du dich in alle Kompass-Richtungen bewegen und mit der **Leertaste** kannst du springen.

(Kleiner Tipp: Stehe still und halte die Leertaste gedrückt)

Fähigkeiten

Im unteren mittleren Bereich des Bildschirms, siehst du links neben deiner Spielfigur ein Schild. Dieses kannst du durch Drücken der **Q** Taste aktivieren. Rechts daneben siehst du einen Tornado, diesen kannst du mit der **E** Taste auslösen und mittels **Mausbewegung** in Third-Person steuern. Beide Fähigkeiten haben eine Abklingzeit, also nutze sie weise.

Cam-Lock

Durch Drücken der **F** Taste bleibt die Kamera auf den Drachen fokussiert und deine Ansicht wechselt in eine First-Person Perspektive.

Menü

Die **Esc** Taste pausiert das Spiel und du hast die Möglichkeit die Lautstärke der Hintergrundmusik und der Soundeffekte anzupassen. Solltest du aus dem Pausemenü zum Hauptmenü zurückkehren, gilt der Run als verloren, deine bisher gesammelten Highscore-Punkte, werden aber gespeichert.

Installationsanleitung

1. ZIP-Ordner *Draconis Occisio* entpacken
2. **Draconis occisio** Anwendung ausführen

Fazit

Dies ist das erstes Spiel, welches ich alleine erstellt habe. Natürlich gibt es einige Verbesserungsmöglichkeiten. Das Userinterface kann beispielsweise durch ansprechendes Feedback für den Spieler intuitiver gestaltet werden und im Play-Menü sollte die Coroutine, die beim Hovern auf dem Random - Button läuft, gestoppt werden, sobald das Hovern endet. Eine weitere Verbesserung kann im Aufbau der Software-Architektur gemacht werden, sodass ein modularer Aufbau erreicht wird. Vor allem hinsichtlich abstrakter Klassen und einer Klassenstruktur und -ordnung nach Funktionalität. Eine erste Idee hierfür ist, eine abstrakte Projektil-Klasse zu erstellen, mit der die Bewegung und der Schaden der Tornados eingestellt werden kann und in der Ausimplementierung in Enemy und Player wird dann die "OnTriggerEnter" bzw. "OnCollisionEnter" - Methode überschrieben.

Die nächste Verbesserung sehe ich in den Drachen-Prefabs. Auf jedem Drachen liegen doppelte Collider, zur Kollisionsdetektion mit dem Spieler und zur Triggerdetektion eines Tornados. Ein Collider sollte für die Implementierung der Schadensevaluierung ausreichen. (Ja, doppelte Hits mit dem Spielertornado sind möglich, aber selten).

Die Umsetzung für dieses Spiel ist aus einer stark eingekürzten Version meiner [Game-Design Idee](#) aus dem letzten Semester entstanden. Ich habe die Idee insofern abgeändert, dass das Ziel des Spiels das Erreichen eines Highscores ist und der Spieler mit zwei Fähigkeiten startet. Doch Setting und Grundgefühl sollten dasselbe bleiben.

Abschließend ist zu sagen, dass ich mit dem Ergebnis zufrieden bin.

Quellen

- 3D-Assets:
 - [Zauberer](#)
 - [Arena](#)
 - [Drachen](#)
- VFX:
 - [Tornado](#)
 - [Schild](#)
 - [Feuer](#)
- Shader:
 - [Hologram](#)
- Tools:
 - [Jetbrains-Rider](#)
 - [DocFX](#) zur Erstellung einer Dokumentation anhand der Kommentare
 - [Unity](#)
 - CineMachine
 - Visual Effects Graph
 - [JsonConvert](#)
 - [LucidChart](#)
- [FieldOfView-Drachen](#)

Anhang

Menü Entwurf



Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel

Draconis occisio

selbstständig und ohne unerlaubte fremde Hilfe angefertigt, keine anderen als die angegebenen Quellen verwendet und die verwendeten Quellen als solche kenntlich gemacht habe.

Rochlitz, den 24.08.2022

Paul Fuchs