2D-Spiel "FORLORN HOPE" (MS Projekt B)

Team: S. Morlok, M. Laudon, T. Lamping

Inhaltsverzeichnis

Grundidee

Die Grundidee war die Entwicklung eines 2D Plattformers im Stil der alten Metroid-Spiele. So sollen die Level nicht in sich abgeschlossen sein, sondern eher so, dass der Spieler zu einem späteren Zeitpunkt zurückkehren muss, um neue Wege zu erkunden. Dies soll den Spieler motivieren, die einzelnen Levelabschnitte genauer zu erkunden.
Eine weitere Motivation ist das Lösen von Rätseln, um voran zu kommen. Der Spieler findet dazu ein Schwerkraftgerät, mit dessen Hilfe er um sich herum eine Art Blase erschaffen kann, in die er schwerelos ist. So kann der Spieler auch Orte erreichen, die normalerweise unerreichbar für ihn wären. Auch Objekte werden angezogen, sobald sie sich in dieser Blase befinden. Somit kann der Spieler ebenfalls Objekte erreichen, die sonst unerreichbar wären.

Die Hauptmotivation liegt darin, ein bestimmtes Ziel zu erreichen. In diesem Fall bedeutet das, dass der Hauptcharakter eine geliebte Person sucht, die sich auf einem verlassenen Raumschiff befinden soll.

Das verlassene Raumschiff ist dabei der Hauptschauplatz des gesamten Spiels und beinhaltet die verschiedenen Levelebenen, die der Spieler erkunden kann.

Konzept

Pattform: PC

Genre: 2D Plattformer

Zielgruppe: Gewohnheitsspieler

Spielwelt: Science-Fiction- Weltraum

Gameplay / Motivation: Schwerkraft ein- und ausschalten, Rätsel lösen, Sammeln

- Grundlegende Motivation: Finde deine geliebte Person
- Auf dem Weg dorthin Rätsel lösen, um voran zu kommen
- Sammeln von Informationen für die Story

Bedienung: Hauptfokus liegt zunächst auf Controller

Engine: Unity

Umsetzung und verwendete Software

Für die Erstellung eines 2D Spiels gibt es verschiedene Tools/Engines, wie zum Beispiel den **Game Maker** oder **Unity.** Letztendlich fiel die Entscheidung auf Unity. Eine der Hauptaspekte der Entscheidung war die bereits gesammelte Erfahrung mit dieser Engine und die unterstützte Programmiersprache C#, sodass nicht extra Zeit in dem Erlernen einer neuen Engine gesteckt werden musste.

Da wir nicht so viel Erfahrungen im Erstellen von Grafiken haben, haben wir uns dazu entschieden, das grundlegende Tileset für unser Spiel im Asset Store zu kaufen, sodass die Konzentration mehr auf die Erstellung der Mechaniken, Sounds und der Story liegen kann. Zur Versionskontrolle haben wir github verwendet.

Hier die Auflistung der verwendeten Software:

- Unity
- Github
- Discord
- Trello
- Adobe Creative Suite (Photoshop, Audition)
- Ableton Live 9
- Audacity

Playthrough

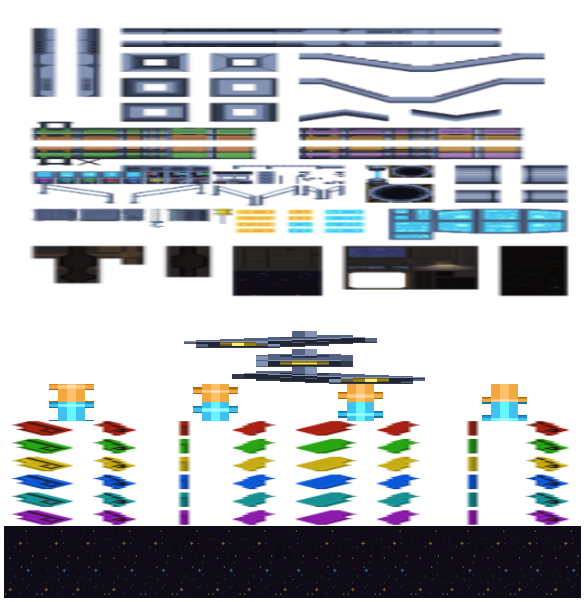
Hier ein Playthrough des ersten Levels

Grafiken

Der Großteil der Grafiken kommt entweder aus dem in Unity integrierten Asset Store oder von der Webseite https://craftpix.net/ , welche die Grafiken teilweise kostenlos anbieten. Die Icons für die Controller Buttons stammen von https://opengameart.org/content/free-keyboard-and-controllers-prompts-pack

Die Schriftart stammt von der Seite: www.1001freefonts.com.

Das grundlegende Tileset:



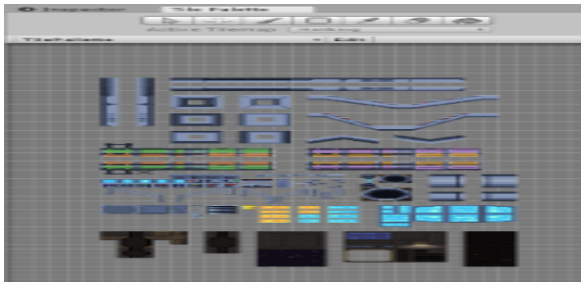
Dieses Tileset stammt aus dem Unity Asset Store.

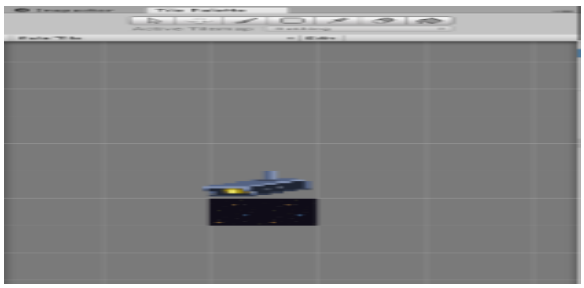
Die UI Elemente:

Diese Icons werden im Startmenü und im Pause Menü eingeblendet, um dem Spieler die Steuerung zu erklären. Der X-Button erscheint auch im Spiel, wenn der Spieler mit einem Objekt interagieren kann. Der Y-Button kommt nur zum Einsatz, wenn der Spieler einen Code eingeben muss. Deswegen wird dieser in der Steuerung nicht erwähnt.

Die Spielwelt

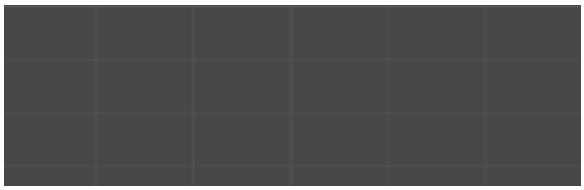
Die Spielwelt wurde als sogenannte *Tilemap* erstellt, welche in Unity integriert ist. Um die Tiles auf die Tilemap zu bekommen, gibt es ebenfalls in Unity integrierten Tile-Palette, von denen man beliebig viele erstellen kann. Hier die zwei Tilepaletten, die für das Projekt erstellt worden sind:

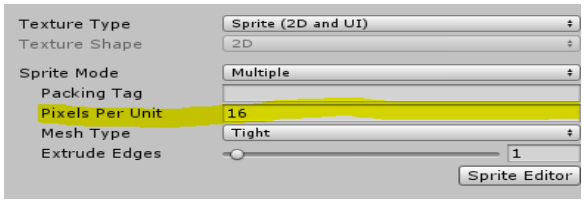




Bei der ersten Tilealette handelt es sich um das normale Tileset. In der zweiten Tilepalette befinden sich sogenannte ***Rule-Tiles,*** welche extern in Unity eingebunden werden mussten, um diese zu erstellen. Mit diesen kann man die Tiles animieren und ebenfalls auf die Tilemap gezeichnet werden. Das hat den Vorteil, dass die animierten Tiles nicht einzeln als Objekte in die Spielwelt plaziert werden müssen.

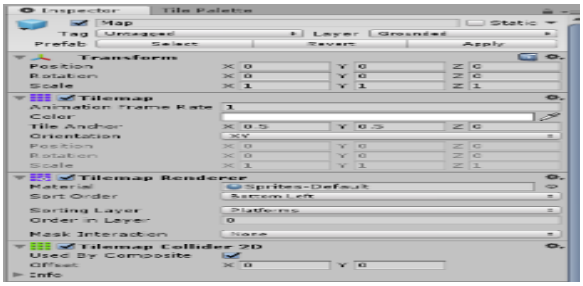
Jedes Tileobjekt ist genau eine Unit groß. Bei einer Unit handelt es sich genau um ein Kästchen

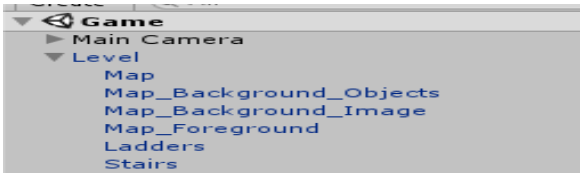




Wenn z.B. ein einzelnes Tile eine Auflösung von 16x16 hat, müssen die ***Pixels Per Unit*** im Unity Editor ebenfalls auf 16 eingestellt werden.

Um alle Objekte mit Hilfe des Tile-Systems in Unity richtig platzieren zu können, mussten mehrere Tilemaps erstellt werden.





Die “Map” stellt dabei den Grundriss der einzelnen Levelebenen dar, mit denen der Spieler kollidieren kann. Für die Hintergründe selbst brauchte es ebenfalls extra Tilemaps, mit denen der Spieler nicht kollidieren kann. “Background-Image” stellt dabei den Hintergrund dar, während sich die “Background-Objects” direkt davor befinden.. Im “Foreground” befinden sich Tiles, die sich vor dem Spieler befinden. Desweiteren wurden für Leitern und Treppen ebenfalls eine eigene Tilemap erstellt, da für diese eine eigene Mechanik erstellt werden musste. So wird z.B. bei Berührung mit dem Tilemap Collider für die Leitern die Steuerung so verändert, dass der Spieler diese hoch- und runterklettern kann.

Leveldesign

Beim Leveldesign standen wir von Anfang an vor der Frage, ob wir einen eher freien, oder einen eher realistischen Anspruch bei der Gestaltung der Level verfolgen sollten. Durch einen sehr freien Ansatz bei der Levelgestaltung könnte man die Spielmechanik der Schwerkraftaufhebung deutlich stärker einsetzen (wie wir schon in unserem ersten Testlevel sehen konnten). Gleichzeitig hatten wir jedoch auch einen narrativen Anspruch an unser Spiel, der sich auch im Leveldesign widerspiegeln sollte, Somit entschieden wir uns für einen eher realistischen Ansatz, so dass die Level auf dem Raumschiff so aussehen sollten wie auf einem "echten" Raumschiff. Als Vorbilder dienten dabei etliche Science-Fiction Filme wie etwa Alien, oder Spiele, wie etwa Metroid, Dead Space, oder auch System Shock. Die Aufteilung der Level wurde so Konzipiert, dass sie Unabhängig von denen Ebenen des Raumschiffs sein sollte. Das heißt, dass es zwar drei Level und drei Raumschiffsebenen geben sollte, diese aber nicht deckungsgleich sein sollten. Der erste Level war noch so entworfen, dass der Spieler dort (beinahe) die komplette unterste Ebene erforscht, mit dem zweiten Level sollte sich das aber ändern und der dritte Level sollte letztendlich auf allen drei Ebenen spielen.

Der erste Level sollte im unteren Bereich des Schiffes spielen, einen eher mechanischen Look haben, den Spieler einen Einblick in die Geschehnisse an Bord geben und zudem als Tutorial fungieren. Der Spieler solle hier zum ersten mal seine neuen Fähigkeiten ausprobieren können. Zudem sollte er aber auch schon ein paar einfache Rätsel lösen, die mit der Schwerkraftmechanik einhergehen. Diese ersten Rätsel sollen dabei noch recht einfach sein: schwebe an Stellen, zu denen man bei normaler Schwerkra nicht hinkommt, Sammle eine Schlüselkarte ein, oder bringe 3 Gegenstände (Reperaturkits) zu einem speziellen Punkt (Energiegenerator). Dabei sollte er durch die Spielumgebung lernen was alles mit der Schwerkraftmechanik möglich ist (etwa Gegenstände bewegen bzw. ziehen), ohne es hier schon Anwenden zu müssen. Dies sollte auf entsprechende Rätsel vorbereiten, die in einem späteren Spielabschnitt vorkommen sollten. Auch über die Geschichte sollte der Spieler informiert werden. So sollte erste Textnachrichten einen Einblick in die Geschichte und in einige der letzten Momente der Besatzungsmitgleider an Bord gewähren, bevor sie das Schiff verließen. Ansonsten ist der Aufbau recht linear. Der Spieler bewegt sich mehr oder weniger ausschließlich von links nach rechts, überwindet Hindernisse und repariert letztendlich den Hauptreaktor. Dies ermöglicht ihm zum Hauptaufzug zurückzukehren und auf die zweite Ebene zu wechseln.

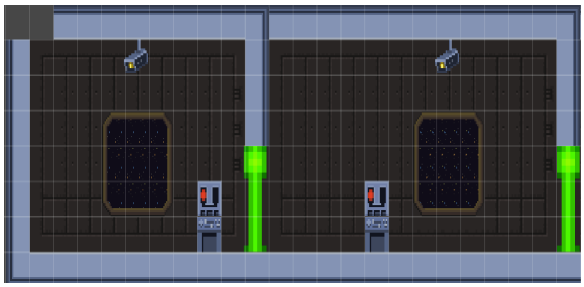
Der zweite Level beginnt auf der zweiten Ebene. Anders als beim ersten Level wird jedoch in etwa nur die Hälfte dieser Ebene erforscht. Der Spieler wird damit konfrontiert, dass er einen Sicherheitscode braucht um mit dem Aufzug auf die Kommandoebene gelangen zu können. Schon früh entschieden wir uns dafür, dass sich auf der zweiten Ebene ein Labortrakt befinden sollte. Wir sprachen wohl auch über einen Wohntrakt, jedoch sollten Labore das prägende Element sein. Zudem sollte diese Ebene zwar über Räume und Gänge mit einer geringeren Deckenhöhe verfügen, als auf der Ebene davor. Allerdings sollte diese Ebene über mehrere Stockwerke verfügen, die der Spieler über Treppen, Leitern und über Löcher im Boden erreichen sollte. Der Aufbau sollte somit in gewisser Weise unserem ersten Testlevel entsprechen, in dem wir zunächst nur unsere Spielmechanik getestet hatten. Ebenfalls früh hatten wir uns dafür entschieden, dass der Spieler später in den Labortrakt zurückkehren sollte, um dort einen Gegenstand o ä. zu holen. Somit war klar, dass nicht die komplette Ebene im zweiten Level betretbar sein sollte. Beim späteren Entwurf des Levels ergab sich, dass das Labor zu diesem Zeitpunkt für den Spieler komplett unerreichbar sein sollte. Stattdessen sollte nur die Wohnsektion begehbar sein. Aufgrund von Beschädigungen und verschlossenen Türen, sollte dieser Level eher einem kleinen Labyrith ähneln, durch dass sich der Spieler bewegen muss. Nachrichten in den Wohnungen der besatzung erzählen die Geschichte fort und geben Hinweise darauf, wie man weitere Teile des Levels für den Spieler öffnet (etwa durch die Preisgabe von Codes). Letztendlich findet der Spieler den Sicherheitscode um auf die Kommandoebene zu gelangen.

Der Übergang zwischen dem zweiten und dritten Level ist fließend. Nach dem Betreten der Kommandoebene gelangt man zur Bürcke des Schiffes, doch diese ist verschlossen. Wenn man versucht die Brücke zu betreten meldet sich der Captain, der sich auf der Brücke befindet und den Spieler dazu zwingt in den Labortrakt zu gehen um dort etwas für ihn zu holen und es dann zur Fähre im Hangar des Schiffes zu bringen (die unterste Ebene). Im Gegenzug will der Captain dem Spieler verraten, wo sich die Person befindet, die der Spieler sucht - seine Zwillingsschwester. Der Spieler muss sich also wieder zur zweiten Ebene bewegen, kann dor aber nun den zuvor geschlossenen Labortrakt betreten. Ähnlich wie zurvor bewegt er sich über mehrere Stockwerke durch einem zum Teil stark zerstörtes Labor. Über Enviormental Storytelling erhält der Spieler einblick über die Experimente die hier wohl stattgefunden haben müssen. Größere Teile der Laborausrüstung wurde Zerstört, in mindestes einem Labor gab es eine Explosion und an einigen Wänden breitet sich eine schwarze, flüßig-moosige Lebensform aus. Hier sollen die schwierigsten Rätsel auf den Spieler warten. So muss er etwa lose Kabel mit den passenden Steckdosen mithilfe seiner Gravitationsfähigkeit wieder verbinden, oder Gegenstände, die sich in Lüftungsschächten o.ä. befinden, an die er normalerweise nicht herankommt "herausziehen". Weiterhin sollen der labyrithartige Aufbau wie auch das Managment von verschlossenen Türen, die mithilfe von Zahlencodes und Schlüsselkarten geöffnet werden müssen, den Spieler vor Hindernisse stellen. Letztendlich findet der Spieler dass, was er dem Captain bringen soll und begibt sich wieder auf die erste Ebene. Dort übergibt er es ihm und erwährt nun, dass der Captain die Zwillingsschwester des spielers in seiner persönlichen Suite auf der Kommandoebene befindet. Der Spieler erhält einen entsprechenden Zahlencode und begibt sich nun dorthin zurück. In der suite angekommen findet er letztendlich nur eine Textnachricht, die aussagt, dass die Zwillingsschwester des Spielers vom Captain bei vor Evakuierung des schiffes eingesperrt wurde. Ihr gelang es jedoch letztendlich die persönliche Fluchtkapsel des Captains zu hacken und flh zum nächstgelegenen Planeten. Nach dem Lesen der Nachricht aktiviert sich die Slbstzerstörung des Raumschiffes und der Spieler muss unter Zeitdruck zum Anfang des Spieles zurückkehren.

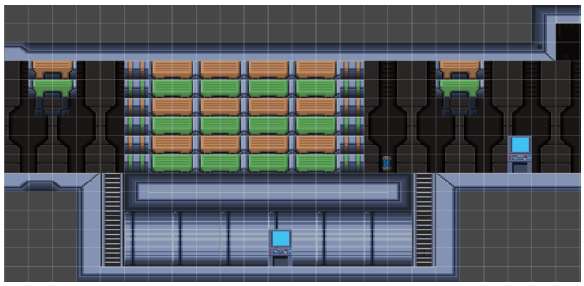
Der grobe Spielablauf des ersten Levels

Beim Starten des Spiels befindet sich der Spieler im ersten Levelabschnitt. Dieser dient dazu, die grundlegenden Mechaniken des Spiels zu erlernen. Es handelt sich also um ein kleines Tutorial.

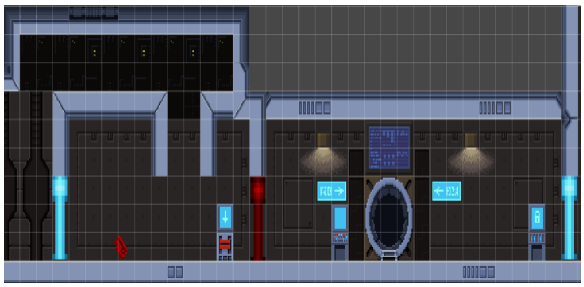
Zu Beginn ist der Spieler durch eine beschädigte Stelle ins Raumschiff gelangt, was den Alarm ausgelöst hat. Da die Luft hinausströmt, findet der Spieler sich in einem schwerelosen Vakuum wieder. Erst das Schließen der Tür stellt die künstlicher Schwerkraft des Raumschiffs her und der Spieler kann sich normal bewegen.



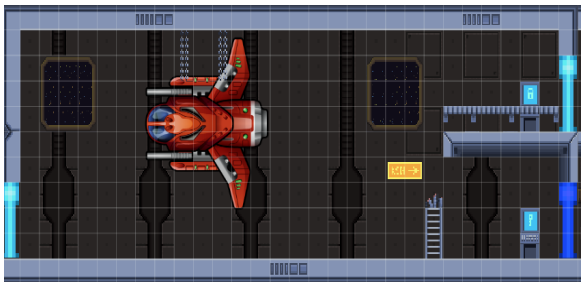
Als nächstes kommt der Spieler in einem Lagerraum. Da der ganze Raum mit Kisten blockiert ist, muss der Spieler durch einen Tunnel klettern. Auf der anderen Seite des Lagerraums findet er dann das Schwerkraftgerät, mit dem er “fliegen” kann.



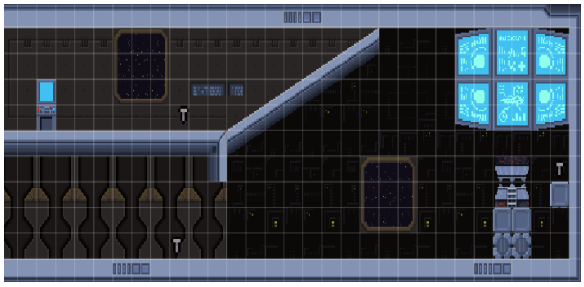
Um in den nächsten Raum zu gelangen, muss der Spieler durch eine Art Lüftungsschacht fliegen. Während er in den Lüftungsschacht fliegt, kann der Spieler die rote Karte mit dem Schwerkraftgerät mitziehen. Diese braucht er, um in den nächsten Raum zu gelangen. Anschließend kommt der Spieler an einem Fahrstuhl vorbei, welcher aber nicht funktioniert, da keine Energie mehr vorhanden ist.



Der Spieler muss also erstmal weiter gehen und gelangt in einem Hangar. Dort kann er entweder oben oder unten weitergehen. Jedoch ist die Leiter nach oben zerstört, sodass er das Schwerkraftgerät einsetzen muss. Für den unteren Weg benötigt der Spieler einen Code, den er auf dem oberen Weg findet.



Außerdem erfährt der Spieler, dass er drei Nanoreperaturkits sammeln muss, um den Reaktor wieder in Gang zu bringen. Hat er dies geschafft, kann der Spieler den Fahrstuhl benutzen und gelangt in den 2. Levelabschnitt. Das Tutorial ist somit abgeschlossen.



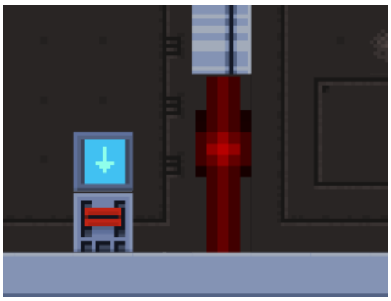
Scripte /Programmierung

Ein Großteil der Arbeit war die Programmierung. Darunter fallen die Mechaniken wie z.B. das Schwerkraft-Feature sowie kleinere Funktionen (Steuerung, öffnen von Türen, betätigen der Hebel, reparieren des Reaktors, einsammeln von Gegenständen usw.) Auch das Ein- und Ausblenden von UI Elementen (Pause Menü, Batterieanzeige, Benutzen-Taste usw.) musste programmiert werden. Es gibt um die **25** Scripte in dem Projekt.

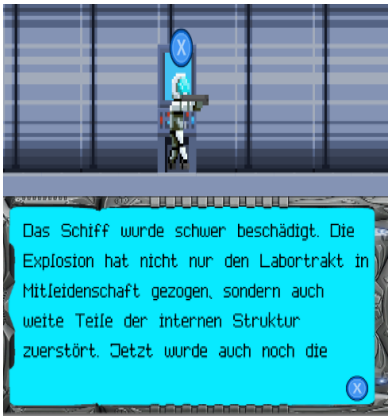
Da dieses Projekt im Rahmen eines zukünftigen Projekts weiterentwickelt werden soll, war es auch wichtig, die Scripte /den Code möglichst so zu erstellen, dass diese in Zukunft einfacher wiederverwendet bzw. erweitert werden können. Dies hat dann auch den Zeitaufwand der Programmierung erhöht.

Das Hauptfeature war die **Schwerkraftmechanik**. Dafür wurden zwei Scripts erstellt. Einmal das Script für das Schwerkraftgerät selbst. Dieses ist dafür verantwortlich, dass sich das Gerät ein- und ausschaltet und je nachdem den Stand der Batterie überprüft. Bei dem zweiten Script handelt es sich um den Bereich, der um den Spieler erschaffen wird. Dieser hat die Aufgabe, Gegenstände anzuziehen, die sich in dem Bereich befinden.

Ein weiteres Feature ist das **Tür-System**. Diese lassen sich über die verschiedenen Terminals bedienen. Für die einzelnen Türen gibt es ein Script, welches diese einfach öffnen und schließen lässt. Die Logik, unter welchen Bedingungen eine Tür geöffnet werden kann, steckt in den sogenannten Terminals. Während bei einem normalen Terminal ein einfacher Druck auf die Benutzen-Taste ausreicht, überprüft das **KeyCardTerminal**, ob der Spieler die benötigte Schlüsselkarte bei sich hat. Das **LeverTerminal** stellt die Hebel dar, welche immer eine Tür öffnen und gleichzeitig eine andere schließen. Das **KeyPadTerminal** erwartet einen Code vom Spieler, den dieser eingeben muss. Erst bei Eingabe des richtigen Codes öffnet sich die Tür. Ist der Code einmal korrekt eingegeben, kann die Tür jederzeit ohne erneute Eingabe geschlossen und wieder geöffnet werden.



Mithilfe des **Dialog-Systems** ist es möglich, dem Spieler Texte über einen Computer anzuzeigen. Somit erfährt er nach und nach, was auf dem Schiff vorgefallen ist. Mit der Benutzen-Taste kann der Spieler den “Dialog” schließen bzw. den nächsten Satz einblenden lassen.



Fazit und Lessons Learned

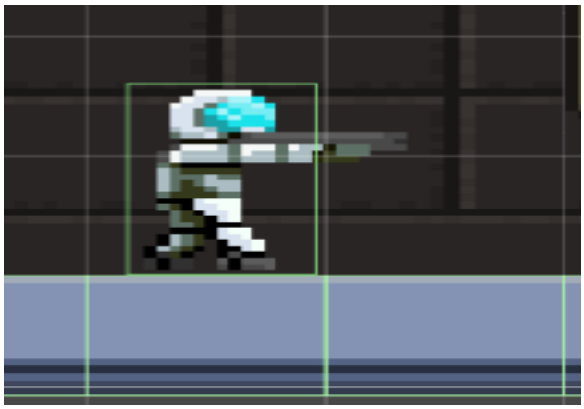
Im Rahmen des Projekts konnten wir die Erfahrung machen, wie viel Arbeit in einem “einfachen” 2D Platformer stecken kann. Allein die Anzahl der Scripts für die Mechaniken wuchs immer weiter an. Auch merkten wir, dass uns trotz eines umfangreichen Tilesets aus dem Asset Store noch viele Grafiken und Animationen fehlten, sodass nach weiteren Grafiken recherchiert werden musste. Dadurch lernten wir, dass ein Grafiker im Team einen großen Vorteil bietet, da so alle Grafiken für das Spiel individuell erstellt werden können.

Deshalb haben wir den Fokus erstmal auf die Mechaniken, die Soundeffekte und dem Storytelling gelegt.

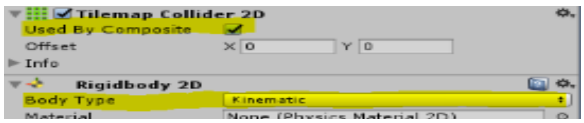
Im Folgenden noch ein paar Beispiele, auf die wir während des Projektes gestoßen sind:

Tilemap-Collider Problem

Die Map mit dem normalen Tilemap-Collider (links) und mit dem Composite-Collider (rechts)

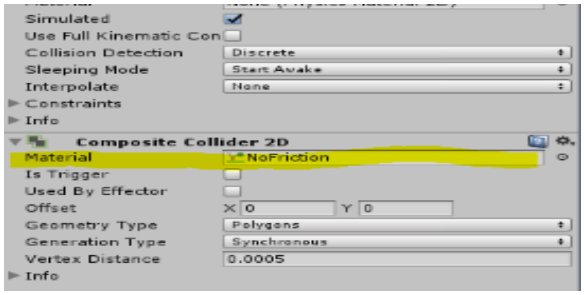


Der Composite-Collider mit hinzugefügtem *Physics Material 2D*



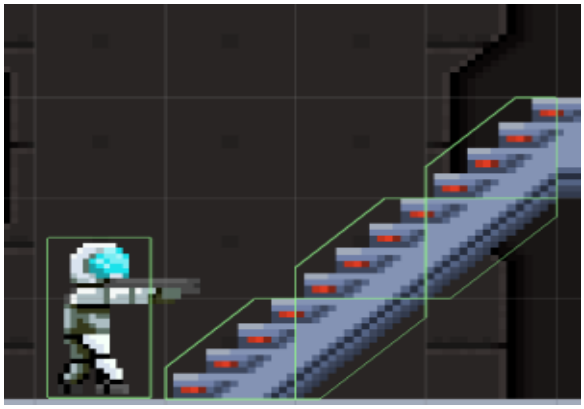
Der normale *Tilemap-Collider* führte in der Anfangsphase dazu, dass der Spieler zwischen den einzelnen Tiles hängen bleiben konnte. Dies trat sehr zufällig auf. Nach einer Recherche wurde klar, dass die Behebung des Problems sehr einfach ist. Bei dem *Tilemap-Collider* muss einfach der Haken bei *Used By Composite* gesetzt werden. Anschließend kann de *Composite-Collider* als Komponente hinzugefügt werden. Anschließend wird automatisch ein *Rigidbody 2D* hinzugefügt bei dem der *Body Type* auf *Kinematic* gestellt werden muss, da die *Tilemap* sonst eine Schwerkraft hat und nach unten fällt. Dieser sorgt dafür, dass nicht mehr jedes einzelne Tile einen Collider hat, sondern die gesamte Tilemap als Collider zusammengefasst wird. So kann der Spieler nicht mehr zwischen den einzelnen Tiles hängen bleiben.

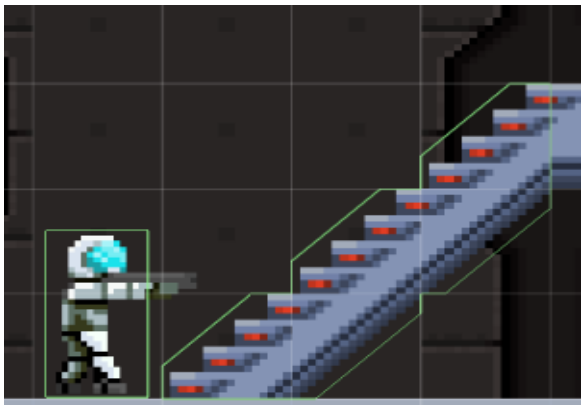
Ein weiteres Problem bestand aber weiterhin: Sobald der Spieler gegen eine Wand gesprungen ist, blieb er dort hängen, solange man sich in Richtung der Wand bewegte. Aber auch dafür war die Lösung letztendlich einfach. Man benötigte nur ein *Physics Material 2D*, welches in Unity sehr einfach erstellt werden kann. Bei diesen setzt man einfach die *Friction* und *Bo nciness* auf 0 und fügt dieses dem Composite-Collider hinzu.



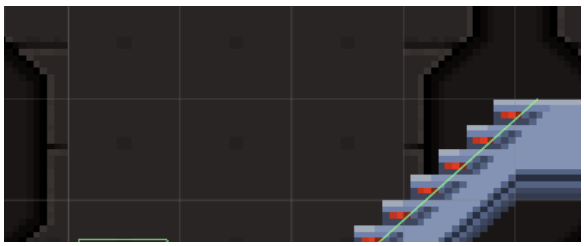
Problem mit den Stair-Collider

Treppen mit normalen Tilemap-Collider (links) und mit Composite-Collider (rechts)





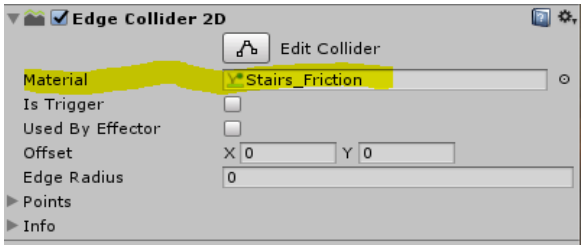
Collider für die einzelnen Treppen einfügen

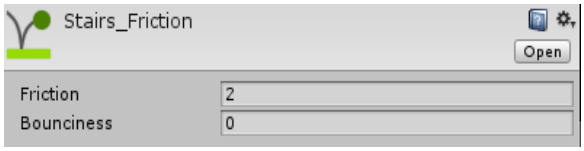


Im späteren Spielverlauf sollen auch noch Treppen eingefügt werden, die ebenfalls im grundlegenden Tileset enthalten sind Bei den Treppen gab es allerdings das Problem, dass sowohl de *Tilemap-Collider* als auch der *Composite-Collider* verhindern dass der Spieler die Treppe betreten kann, ohne zu springen. würde vor jeder Stufe hängen bleiben. Die einfachste Lösung war es, für die beiden Treppenarten (steile und flache Treppen für jede Richtung einen *Edge-Collider* zu erstellen und manue einzufügen. So haben die Treppen eine eigene Tilemap, jedoch ohne eigenen *Tilemap-Collider*.

Ein weiteres Problem war dann, dass der Spieler die Treppen z schnell hochgekommen ist, sodass er am Ende der Treppe hoc "gesprungen" ist. Um dieses Problem zu lösen, wurde für den *dge-Collider* ein *Physics Material 2D* mit dem Namen *Stairs_F ction* erstellt. Der Spieler verliert dadurch je nach eingestelltem *Friction* Wert die Geschwindigkeit auf der Treppe. Mit einem Wert von 2 konnte zunächst das beste Ergebnis erzielt werden, um das Problem zu lösen.

Einfügen eines Physic2D Materials für den Stair-Collider, damit Spieler nicht zu schnell die Treppe hoch sprintet.





Authentizität des Sounddesigns

Während der Recherche nach ersten Soundplatzhaltern wurde schnell deutlich, dass die meisten vorgefertigten Sounds nicht unseren Erwartungen entsprachen. Viele der Klänge waren zwar für sich genommen eine gute Bereicherung, trugen jedoch dazu bei, dass keine einheitliche Atmosphäre geschaffen werden konnte. Außerdem waren einige der Anforderungen so speziell, dass es schlichtweg kein vergleichbares bereitgestelltes Soundset gab. Aufgrund dessen wurden viele der Sounds nicht eins zu eins übernommen, sondern nach eigenen Vorstellungen nachgebaut und verändert. Die neugewonnene Erfahrung mit diesem Prozess lässt spätere Probleme deutlich einfacher beheben, sorgte in diesem Projekt jedoch für einen hohen Zeitaufwand.

Zeitaufwand

Im Folgendem eine Übersicht der aufgewendeten Stunden.

Person	Bereich	Stunden gesamt

T. Lamping	Organisation • Teambesprechung • Doku • Sonstiges	**35** • 20 • 10 • 5
	Grafik und Design • Mapping 1. Abschnitt • Schwerkrafteffekt • Recherche nach Grafiken	**15** • 5 • 5 • 5
	Programmierung • Erster Prototyp • Schwerkraftmechanik • Sonstige Mechaniken • UI • Bugfixes • Code Cleaning	**130** • 10 • 35 • 20 • 15 • 30 • 20

Person	**Bereich**	**Stunden gesamt**
M. Laudon	Organisation • Teambesprechung • Doku • Sonstiges • Konzeptentwürfe	**38** • 20 • 8 • 6 • 4
	Grafik und Design • Grafikrecherche • Überarbeitung des Menüs & der Schriften	**10** • 5 • 5
	Sound • Soundrecherche	**125** • 15

	• Aufbau einer Soundlibrary • Sounddesign • Aufnahmen • Editing • Soundtrackrecherche • Kompositionsentwürfe • Arrangement • Soundtrackumsetzung und Editing	• 12 • 20 • 15 • 20 • 9 • 12 • 8 • 14
	Q & A • Exploratives Bug Testing	**8** • 8

Person	**Bereich**	**Stunden gesamt**
S. Morlok	Organisation • Teambesprechung • Projektmanagement • Doku • Sonstiges	**50** • 20 • 15 • 10 • 5
	Leveldesign • Recherche • Entwurf Level 1 • Entwurf Level 2 • Entwurf Level 3	**75** • 20 • 15 • 20 • 20
	Unity • Einarbeitung • Polishing Level 1	**30** • 20 • 10
	Texte • Texte Level 1 • Texte Level 2	**15** • 5 • 10
	Q & A • Testing	**10** • 10
