



The Thanos Principle

Dokumentation

David Bollmann - 5042634

Jonathan Verbeek - 5058288

Gruppe B

Gliederung

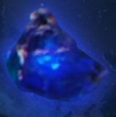
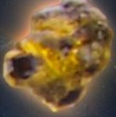


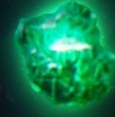

Motivation & Idee	2
Art & Design	3
Codestruktur & Architektur	10
Sound	14
Schlusswort	16
Anhang & Quellen	17

Gruppenaufteilung

Idee & Konzept	David, Jonathan
Art, Design, Level Design	David
Programmierung Engine und Spiellogik	Jonathan
Sounds	David, Jonathan
Musik	Jonathan

Motivation & Idee

Das Spiel "The Thanos Principle" basiert auf dem Marvel-Charakter "Thanos". Im Film **Avengers 3: Infinity War**, produziert von Marvel und veröffentlicht in 2018 wird er als Bösewicht dargestellt, der besondere Fähigkeiten durch den Einsatz verschiedener Steine, die sogenannten Infinity Stones, die er in einem Handschuh, dem sogenannten Gauntlet trägt, nutzen kann.

					
SPACE STONE	MIND STONE	REALITY STONE	POWER STONE	TIME STONE	SOUL STONE
Thor Captain America The Avengers	The Avengers The Winter Soldier Age of Ultron Civil War	The Dark World	Guardians of the Galaxy	Doctor Strange	???
Tesseract	Chitauri Scepter	Aether	Orb	Eye of Agamotto	???
Asgard Heimdall (2012)	Earth Vision (2015)	Knowhere Collector's Museum (2013)	Xandar Nova Corps Vault (2014)	Earth Kamar-Taj Library (2016)	???

1

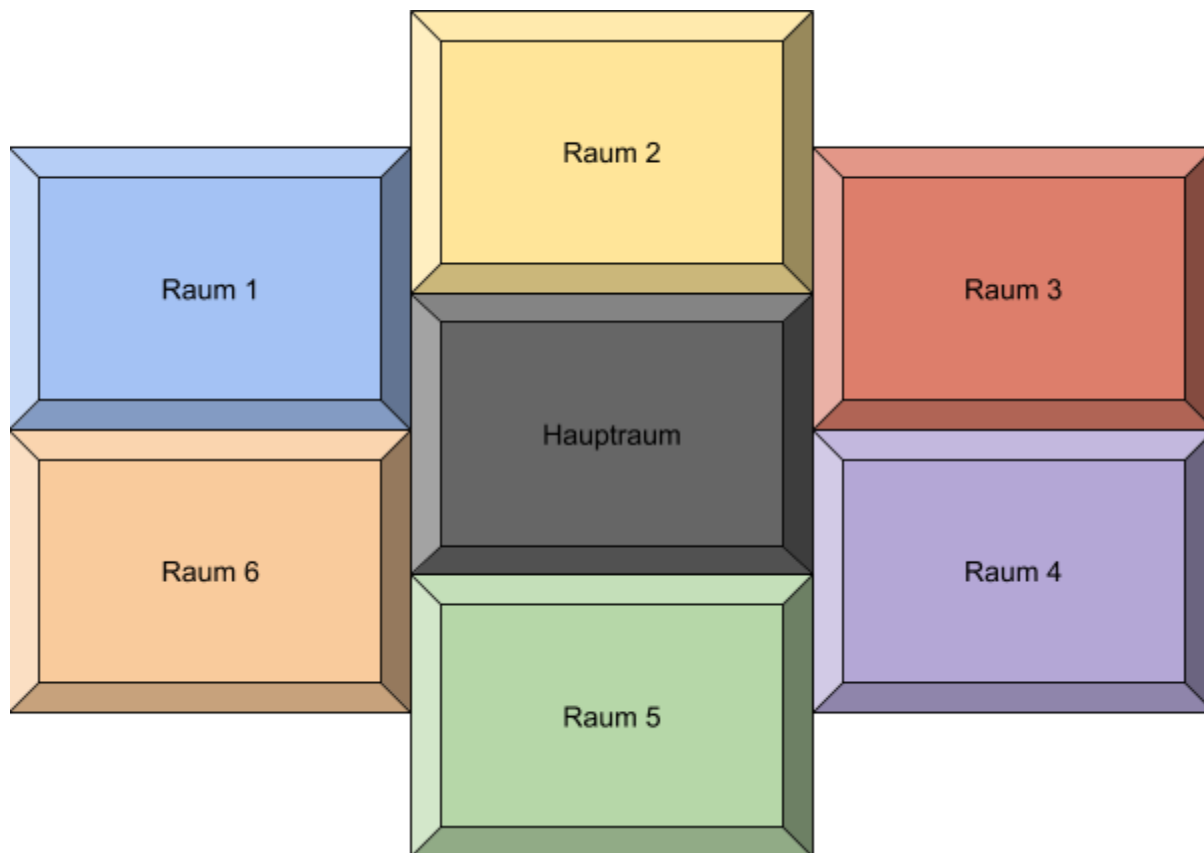
Aufgrund der Internet-Popularität des Charakters entschlossen wir uns schnell dazu, dass wir Thanos als Hauptfigur in unser Spiel einbauen möchten. Relativ schnell stand dann auch die Idee des Spiels: ein Puzzle-Spiel, in dem Thanos die Steine finden muss, um sie in seinen Handschuh einzusetzen, und sich damit schließlich seine Superkräfte zurück zu holen. Mit der richtigen Umsetzung erschien uns diese Idee als ein gutes Konzept, und auf der Suche nach einem guten Namen fiel uns **The Thanos Principle** als Erstes ein. Dies ist eine Anlehnung an das 2014 erschienene Puzzle-Spiel "The Talos Principle" vom polnischen Entwickler Croteam. Dass der Spieler sowohl in unserem Spiel, als auch in The Talos Principle bunte Steine sammeln muss und diese in Vorrichtungen einsetzen muss, wurde uns allerdings erst nach der Entwicklung klar.

1

<https://medium.com/netive-in/what-are-the-infinity-stones-some-things-you-need-to-know-before-avengers-infinity-war-2c789b762d78>

Art & Design

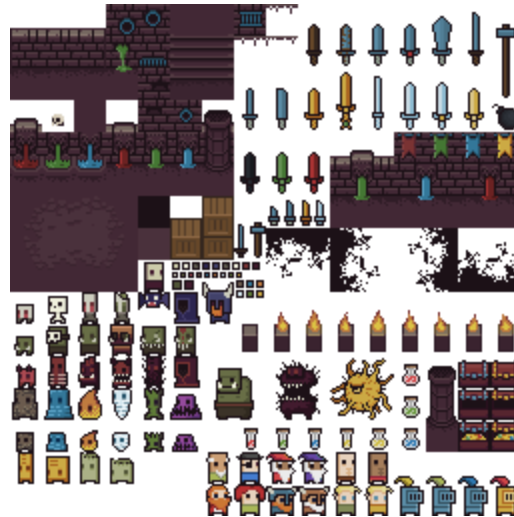
Mit den Überlegungen zum Look des Spiels beschäftigten wir uns direkt nach dem wir die Aufgabe im Labor erhalten hatten. Unser erster Gedanke war, das Spiel ähnlich wie das Spiel "The Binding of Isaac" in einem Dungeon-Look aus mehreren miteinander verbundenen Räumen zu gestalten, in dem der Spieler verschiedene Aufgaben/Rätsel lösen muss. Starten sollte der Spieler in einem mittleren Hauptraum, in dem sich Thanos' Infinity Gauntlet (der Handschuh) auf einem Podest befindet, mit dem nach erfolgreichem Einsammeln aller 6 Steine interagiert werden kann, um das Spiel abzuschließen. Ursprünglich war geplant, die Rätsel-Räume alle direkt an den Hauptraum mit Durchgängen anzuschließen.



Diese Idee wurde aber während des Design-Prozesses fallen gelassen und abgeändert (siehe [Level Design](#)). Jeder angrenzende Raum sollte ein kleines Rätsel enthalten welches beim Absolvieren einen der Steine zur Belohnung freigibt. Zuerst sollten die Räume unabhängig voneinander fungieren, jedoch war schnell klar, dass es deutlich spannender und herausfordernder wäre, wenn Elemente aus bestimmten Räumen zum Lösen von Rätseln aus anderen Räumen benötigt werden. Zudem waren wir uns einig, dass das Game im Pixelart-Style gehalten werden sollte, da wir beide Fans dieses Stils sind.

Farbwahl

Die generelle Atmosphäre der Map sollte dunkel und kühl sein, da der Schauplatz des Spiels ein Dungeon ist und es sich bei Thanos um einen Schurken handelt. Aus diesem Grund entschieden wir uns folgendes Tilesset für den groben Aufbau der Map zu verwenden.



Aus diesem Tilesset nutzten wir einige Elemente, restliche Objekte die wir für unser Level benötigten, um die Räume nach unserer Vorstellung zu gestalten, wurden selbst dazu entworfen. Dabei wurde darauf geachtet, dass diese Objekte zum ursprünglichen Tilesset passten. Das Farbschema und der "Look" rund ums dunkel-lila gefiel uns sehr gut und passte auch zum Protagonisten, ohne das Spiel zu bunt wirken zu lassen. Um die Dungeon-Atmosphäre noch weiter zu unterstreichen und zum Teil auch die Einsicht in benachbarte Räume zu mindern nutzen wir eine Vignette am Bildschirmrand. Bei den selbst erstellten Elementen für die Map wurde darauf geachtet, z.B. bei Holz die Farbe ähnlich zu halten und generell wurde eher auf kühle Grautöne gesetzt u.A. für Das Podest im Hauptraum oder Schalter Elemente an den Wänden. Einzige Ausnahme bilden die 6 Infinity Stones, welche absichtlich in sehr kräftigen bunten Farben erstellt sind, um diese zum Einem vom Rest der Map abzuheben und zum Anderen um ihr tatsächliches Aussehen nicht zu verändern.



Beim User Interface wurde auf unnötige Elemente verzichtet um den Bildschirm nicht zu überladen. Es gibt lediglich eine Anzeige für die 6 Steine am oberen linken Bildschirmrand, welche bei Beginn des Spiels dunkel und halbtransparent zu sehen ist und die Farben der zu sammelnden Steine nur minimal offenbart. Bei Erlangen eines Steins wird dieser oben durch eine kurze Animation in den Slots hinzugefügt. Die Reihenfolge der Slots ist vorbestimmt und entspricht der des Infinity Gauntlets. Zudem befindet sich am oberen rechten Bildschirmrand das Inventar mit einer begrenzten Anzahl an Slots, so dass der Spieler zu jedem Zeitpunkt maximal 2 Gegenstände mit sich führen kann. Die Slots für das Inventar sind in einem etwas kräftigeren Farbton gehalten als Boden und Wände der Räume, um sich von diesen abzuheben und das Inventar problemlos immer sichtbar haben.

Der Großteil aller eigenen Elemente wurde mit der gleichen Farbpalette angefertigt, sofern keine minimalen Änderungen nötig waren um sicherzugehen, dass alles stimmig miteinander aussieht.

Schriftwahl

Für das Intro des Spiels wird die Schriftart "Hauser Condensed" genutzt, da diese am ehesten der Schriftart entsprach, wie sie auf den Thanos Comic-Büchern verwendet wird und wir genau diesen Look auch für unser Spiel wollten. Da die Schriftart der Comic-Bücher erst hätte erworben werden müssen, nutzten wir stattdessen "Hauser" und erhöhten den Zeichenabstand in CSS, um den Look bestmöglich zu reproduzieren.



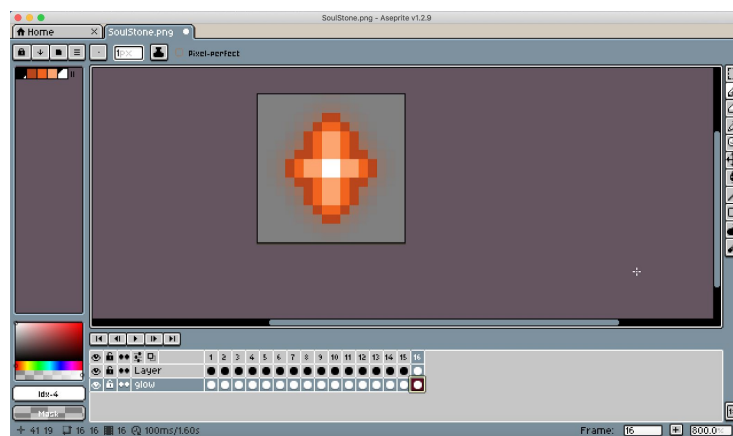
2

Diese Schriftart verwendeten wir ebenfalls für die Credits im Intro. Alle anderen Texte wie z.B. der Menütext oder die Texte, die im Spiel bei Interaktion erscheinen, nutzen die Font "Pixeled" da diese dem klassischen Pixelart-Look entspricht und uns aus einer Auswahl am besten gefallen hat.

² [http://marvel.wikia.com/wiki/File:Thanos_\(2017\)_logo.png](http://marvel.wikia.com/wiki/File:Thanos_(2017)_logo.png)

Tools (Aseprite/Tiled)

Da wir uns zu Anfang entschieden hatten das Spiel im Pixelart-Look zu gestalten, machten wir uns zuerst Gedanken, wie wir am besten Grafiken für unsere Zwecke anfertigen konnten. Wir entschieden uns dazu das Programm Aseprite³ zu nutzen. Aseprite bietet viele Tools um Pixel-Grafiken oder auch Animation zu erstellen und diese als Spritesheets abzuspeichern. Ein Spritesheet ist eine Textur, die alle Frames einer Animation beinhaltet. Die Engine kann dann jeden Frame zur Zeit nacheinander anzeigen und somit eine flüssige Animation abspielen.



In diesem Fall wurde zuerst der Stein in einer Größe von 16x16 Pixeln gezeichnet. Durch die helleren Orange-Töne sowie die weiße Mitte soll ein minimalistischer Lichteffect erzielt werden. Anschließend wurde die Ebene dupliziert und auf den unteren Layer ein Blur-Effekt angewandt, welcher ein "Schimmer"-Effekt simulieren soll. In den folgenden Frames (1-16) wurde die Transparenz des "Glow"-Layers schrittweise gesenkt und wieder erhöht um beim Abspielen den "Glow"-Effekt zu erhalten. Dieser ist z.B. im Endscreen zu sehen, wenn der Spieler ein Abschlussbild mit dem Infinity Gauntlet sowie allen sechs eingesetzten Steinen zu sehen kriegt. Der Effekt ist nur minimal, haucht dem Spiel jedoch etwas mehr Leben ein.

Der Thanos-Charakter wurde nach Vorlage nachgebaut und angepasst. Außerdem wurden drei Extra-Versionen angefertigt, die jeweils in eine andere Richtung schauen. Jede Version hat eine eigene Lauf-Animation um ein möglichst flüssiges Spielgefühl zu liefern.

³ <https://www.aseprite.org/>



Spritesheet: Front



Spritesheet: Back

Zudem sind die beiden Bilder, die im Outro nacheinander ein- bzw. ausgeblendet werden ebenfalls in Aseprite animiert und als .gif-Datei gespeichert, so dass diese durchgehend abgespielt werden.

Nachdem alle benötigten Sprites erstellt und animiert waren, wurden diese in Tiled geladen und dort genutzt um die Map zu zeichnen. Tiled ermöglicht es mit eigenen Tilesets eine Map Ebene für Ebene zu erstellen, welche anschließend in die Engine geladen werden kann (siehe [Codestruktur & Architektur](#))

Level Design

Bevor die Map in Tiled gezeichnet werden konnte, mussten sich erst sechs Rätsel für die Steine, sowie ein generelles Layout überlegt werden, da wir zuvor von der Idee abgerückt waren alle Räume direkt mit dem Hauptraum zu verbinden. Nachdem ein Konzept für die Räume überlegt war, wurden erste Versuche in Tiled gemacht dieses umzusetzen. Im Zuge der Optimierung wurden kleine Änderungen am Layout vorgenommen, sowie die Anzahl der Ebenen reduziert um eine bessere Performance beim Spielen zu erreichen. Letztendlich war die Map so angelegt, dass wir beide zufrieden waren.



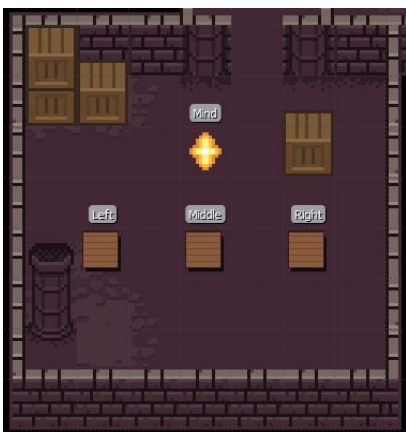
Wir entschieden uns die Map mit Fluren anzulegen, welche die Räume mit dem Hauptraum verbinden um so einen natürlicheren Aufbau zu erzielen. Mit dem neuen Aufbau war es nötig, die Karte zu erkunden und nicht nur Raum für Raum abzugehen, um die Steine einzusammeln. Elemente wie Treppen und Säulen wurden ebenfalls eingebaut, um der Map etwas mehr Leben einzuhauchen. Weitere Deko-Elemente wie z.B. Fackeln und Schädel wurden zum Schluss hinzugefügt und sind nicht rein für die Optik eingebaut, sondern agieren auch als Bestandteil eine Rätsels für welches man die ganze Map genau untersuchen muss.



Dieses Rätsel verlangt den meisten Aufwand vom Spieler um es zu lösen. Die Schalter an der Wand sind erst nach Lesen der Notiz, die sich in dem Raum befindet der ursprünglich durch einen Felsen blockiert ist, benutzbar. Die Notiz fordert den Spieler auf, ein Rätsel zu lösen, für welches er die gesamte Map ablaufen und sich bestimmte Merkmale merken und anschließend in einer Rechenaufgabe nutzen muss. Das Ergebnis dieser Rechenaufgabe ergibt umgerechnet ins Binärsystem eine Lösung, die mit vier Schaltern dargestellt werden kann.



In diesem Raum muss der Spieler sich die Reihenfolge der Lichter merken. (Die Lichter wechseln nach dem Interagieren durch "E" abwechselnd die Farbe zu Blau.) Gibt der Spieler anschließend die Reihenfolge durch erneutes Interagieren richtig wieder erhält er den Stein. Ansonsten färben sich alle Lichter Rot und kurz darauf wieder weiß und der Spieler muss die Aufgabe erneut lösen.



Bei diesem Rätsel müssen die drei Druckplatten mit Objekten belegt werden. Eine Platte kann der Spieler selbst betätigen, indem er sich auf dieser platziert, zum Betätigen der anderen Platten muss zum einen ein Objekt aus einer Kiste gesucht werden und zurück zum Raum gebracht werden. Die 3. Platte lässt sich erst betätigen, nachdem der Spieler mit dem Hammer den großen Fels im rechten Flur zerstört und dessen Trümmer zur Platte gebracht hat.

Intro / Outro

Damit das Spiel nicht einfach nur startet oder ggf. nur ein Menü zum Starten öffnet, haben wir uns entschlossen, ein Intro und Outro für das Spiel anzufertigen, welches gleichzeitig für die Credits genutzt wurde. Die Idee beim Intro war es, zuerst den Titel unseres Spiels mit einer Animation "einschieben" zu lassen und diesen dann auszublenden, anschließend unsere Namen ein und wieder auszublenden und als letzten Schritt das Menübild und den Text "PRESS SPACE TO START" auf diesem pulsieren zu lassen. Die Schrift für den Titel sollte zum einen lila gehalten werden, so wie der Thanos selber, zum anderen aber auch einen angedeuteten Licht-Effekt haben, um einen Retro-Videospiel-Look zu erzielen. Dies war mit einem einfachen Farbverlauf ziemlich gut umzusetzen.

Als das Intro bereits fertig war und wir uns im Laufe der Entwicklung Gedanken zu Sound Effekten und evtl. Musik machten, kam ich auf die Idee das Intro mit Musik zu hinterlegen und die Animation an diese anzupassen. Für die Intro Musik wird das Stück "The Sanctuary" von Tyler Bates aus dem "Guardians of the Galaxy" Soundtrack verwendet und so geschnitten, dass es für das Intro passend wirkt. Der Titel der im Intro zuerst eingeschoben wird, wird nun abgestimmt mit der Musik, hoch skaliert und langsam ausgeblendet, die Credits ebenfalls abgestimmt zum dramatischen Teil der Musik ein- und ausgeblendet. Anschließend lädt mit einem weiteren dramatischen Dröhnen das Menübild und der Spieler wird gebeten das Spiel zu starten.

Das Outro sollte beim erfolgreichen Abschließen des Spiels abgespielt werden und angezeigt werden bis der Spieler das Spiel beendet. Dazu wurde in Aseprite eine Animation des Infinity Gauntlet mit allen eingesetzten Steinen angefertigt, um die sich ein minimalistisch gehaltener Schein dreht. Da Stan Lee während der Arbeiten an unserem Spiel verstorben ist und wir unser ganzes Spiel auf einem Marvel Charakter aufgebaut haben, entschlossen wir uns, eine weitere Animation als Tribut hinter dem ursprünglichen Endscreen anzuhängen. Diese sollte abspielen nachdem das alte Ende langsam ausgeblendet und das neue Endbild eingeblendet wurde. Für diese Szene entschloss ich mich, zum einen ein Bild von Stan Lee in Aseprite zu laden und runter zu skalieren, um den Pixelart-Look des Spiels weiterhin einzuhalten. Außerdem wählte ich ein Bild welches Thanos als Farmer zeigt. Die Szene ist fiktiv, spielt in Thanos' Zuhause und soll seine Verbundenheit zu Stan Lee symbolisieren. Die Bilder hängen beide in einem komplett dunklen Raum an der Wand, welcher nur durch den Schein einer Kerze erleuchtet wird. Hierzu wurde zuerst die Kerze gezeichnet, die Flamme mit Frames in Aseprite animiert und anschließend mit Blur-Effekt weichgezeichnet um einen besseren "Feuer-Look" zu erzielen. Abschließend legte ich einen halbtransparenten schwarzen Layer über das gesamte Bild und entfernte Frame für Frame, angepasst an die Bewegung der Flamme, mit verschiedenen Radiergummi-Werkzeugen Flächen aus dem Layer. So wurde eine stimmige Beleuchtung des Raums abhängig von der Flammengröße erreicht.

Codestruktur & Architektur

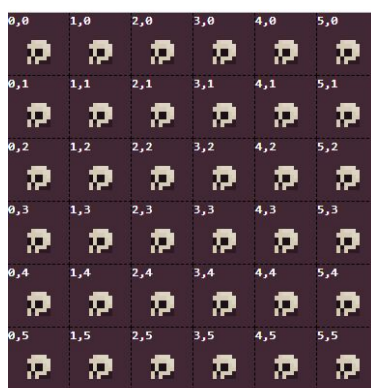
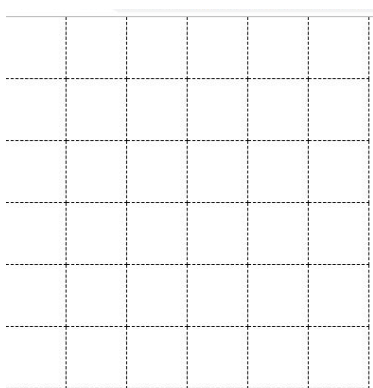
Das gesamte Spiel ist, den Vorgaben entsprechend, in JavaScript programmiert, und verwendet HTML und CSS als Auszeichnungssprache. Der JavaScript Code ist in Module strukturiert und verwendet ES6-Klassen, um eine gewisse Ordnung reinzubringen. Das ganze Spiel wurde ohne Verwendung von Third-Party-Libraries und/oder jQuery programmiert, stattdessen wurde dafür ein robustes eigenes Framework geschrieben.

Unterteilung in Engine und Game

Die Grundidee hier war den Sourcecode der "Engine", also des Frameworks für die Spielkarte, Sprites, Objekte, etc. getrennt vom Gameplay zu halten. Daher als Erstes der Gedanke: "was soll meine Engine denn können?". Da wir ein Grid-basiertes Top-Down Spiel entwickeln, braucht es ein Grid, dass verschiedene Tiles anzeigen kann. Außerdem braucht es interagierbare Objekte, um diese anzuzeigen braucht es (animierte) Sprites. Da der Spieler sich bewegen können muss, braucht es ein Input-System, und muss auch Kollision unterstützen, um zu verhindern, dass der Spieler durch Wände laufen kann. Nach dem wir uns dieses "feature set" überlegt haben, ging es an die Entwicklung der Engine.

Spielbereich / Map & Grid

Das Grid des Spiels besteht im Kern aus div-Elementen, die in Zeilen eingeordnet werden. CSS unterstützt sogenannte Grids⁴, in denen eine feste Größe, und Anzahl an "Tiles" pro Zeile festgelegt werden kann. Die Anzahl der Zeilen erstellt man dann manuell. Dank dieses Features ist es möglich, für jedes Tile in der Map ein div-Element zu erstellen, und diesem per CSS (unter Nutzung von `background-position` und `background-size`) einen Ausschnitt aus dem großen Tileset zuzuweisen, den es anzeigen soll.



Erste Versionen des Grid-Systems im Browser

⁴ https://developer.mozilla.org/de/docs/Web/CSS/CSS_Grid_Layout

Das Grid funktionierte, und wir stellten uns die Frage, wie wir die Map überhaupt gestalten und entwickeln wollen. Da wir uns für den Open-Source Map-Editor Tiled⁵ entschieden haben, musste eine Möglichkeit gefunden werden, diese Maps in unser Grid zu importieren. Um dem sauberen Code-Stil zu Gute zu kommen und diese Aufgabe zu erledigen, entstand die Klasse `TiledMapImporter`, die genau diesen Prozess übernimmt. Die Klasse lädt die Map im JSON-Format, und liest die verschiedenen Ebenen, Tilesets und Objekte entsprechend ab und setzt sie in unserem Grid um.

Auch stellte sich die Frage, ob es nicht sinnvoller wäre, den HTML5 Canvas zu nutzen. Ein kurzer Performance-Test bestätigte jedoch meine Annahme, dass es keinen Sinn macht, diese zu verwenden, da diese pro Animation-Frame die ganze Map neu hätten rendern müssen und die Map größtenteils nur statisch ist und sich nicht verändert. Es war somit besser, die komplette Map im DOM abzubilden. Hier war die Performance zwar auch nicht ganz optimal, aber sehr viel besser (auch, wenn man bedenkt, dass der Browser beim Bewegen des Spielers 10 (Ebenen) * 50 * 45 (Map-Größe) DOM-Elemente neu zeichnet).

Sprites

Ein wichtiger Bestandteil von 2D Engines sind Sprites. Meist werden sie für bewegbare Objekte und Animationen verwendet. Es war klar, dass unsere kleine Engine das auch können muss. Nach längerem Ausprobieren verschiedener Möglichkeiten habe ich mich dann dazu entschieden, dies mit dem HTML5 Canvas zu realisieren, da dieser mir die Möglichkeit bietet, bestimmte Ausschnitte von größeren Bildern auf eine Fläche zu "projizieren". Im Gegensatz zu den Tiles der Map sind in unserer Welt maximal 20 Sprites gleichzeitig aktiv, ein zu großes Performance-Risiko gingen wir also damit nicht ein.

Unsere Sprites sind, wie die meisten in solchen Spielen, als Spritesheets abgelegt. Das erfordert es, wie bei einem Daumenkino jeden Frame nacheinander anzuzeigen, um somit eine Animation zu zeigen. Zum Glück gibt es eine Überladung der `CanvasRenderingContext2D.drawImage()`⁶ Funktion, die nur eine bestimmte Region eines Source-Bildes anzeigen kann. Perfekt also für unser Vorhaben. Zusammen mit einer `tick`-Funktion, die periodisch aufgerufen wird, wird nun jeden Canvas-Tick (wird vom Browser aufgerufen, um das Canvas zu aktualisieren), der nächste Frame des Spritesheets angezeigt. Um eine variable Geschwindigkeit der Animation zu implementieren, habe ich eingebaut, dass nur alle n Frames das Sprite neu gezeichnet wird. Die Sprites waren die Grundlage für sämtliche interagierbare Objekte im Spiel.

⁵ <https://www.mapeditor.org/>

⁶ <https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/drawImage>

Objekte

Objekte sind relativ simpel gehalten, dennoch gehören sie zur wichtigsten Infrastruktur der Engine. Sie sind gegliedert in eine `GameObject` Klasse, die ein Sprite zum Anzeigen von Kisten, den Steinen und auch schließlich vom Spieler selbst beinhaltet. Außerdem verfügt ein `GameObject` über die Fähigkeit, seine Position im world-space (siehe [World & Screen Space](#)) zu verändern. Desweiteren ist ein `onInteract`-Callback zuweisbar, eine simple JavaScript-Funktion, die ausgeführt wird, wenn der Spieler mit dem Objekt interagiert. Dies erlaubt eine simple Schnittstelle zwischen Gameplay-Logik und den Objekten.

Spieler & Input-System

Das wohl wichtigste am Spiel ist der spielbare Charakter, Thanos. Auch er ist im "Herzen" ein `GameObject`, dessen Position allerdings über ein Input-System manipuliert werden kann, um ihn zu bewegen. Dieses System wird, genau wie die Sprites auch, über `window.requestAnimationFrame` periodisch aktualisiert. Auch wenn der Name der Funktion von Animationen spricht, funktioniert dies für andere Anwendungen, im Prinzip ist dies nur ein Callback, der in einem hohen Intervall (bei Möglichkeit 60 Mal die Sekunde) ausgeführt wird - und uns auch somit eine sofortige Input-Auswertung ermöglicht. Basierend darauf, welche Bewegungstasten (W,A,S,D oder die Pfeiltasten) gedrückt werden, wird ein zweidimensionaler Bewegungsvektor berechnet, der die Richtung, in die sich der Spieler im nächsten Frame bewegen soll, angibt. Dieser Vektor wird dann skaliert um die Geschwindigkeit festzulegen, und schließlich zur aktuellen Position des Spielers addiert. Das Ergebnis ist eine flüssige Bewegung. Basierend auf diesem Vektor wird zusätzlich das Spritesheet gewechselt, damit Thanos in die richtige Richtung schaut beim Laufen.

Kollision und Hit-Testing

Die Kollision war eins der schwierigeren Dinge der Engine. Die Überlegung existierte, Box2D⁷ zu implementieren, was allerdings eine zu übertriebene Lösung für ein einfacheres Problem gewesen wäre. Die erste Version des Kollisions-Systems enthielt folgende Überlegung: der Map-Editor Tiled unterstützt benutzerdefinierte JSON-Eigenschaften für Tile-Ebenen, sodass wir dort einfach Ebenen als `collidable` makieren konnten. Die Engine überprüft in der `tick`-Funktion des Spielers mittels `document.elementsAtPoint`⁸ welche DOM-Elemente vor dem Spieler liegen, und filtert diese nach `div`-Elementen, also Tiles, die das `collidable`-Attribut gesetzt haben. Sollte das Ergebnis positiv sein, stoppt die Bewegung des Spielers, und der Charakter bleibt stehen. Das Hit-Testing ist ebenfalls in der Lage, Objekte vor dem Spieler zu erkennen, und mit diesen auf Knopfdruck zu interagieren, um eine bestimmte Aktion des Objektes auszuführen.

⁷ <https://box2d.org/>

⁸ <https://developer.mozilla.org/en-US/docs/Web/API/DocumentOrShadowRoot/elementsFromPoint>

[illegible]

Screen & World Space

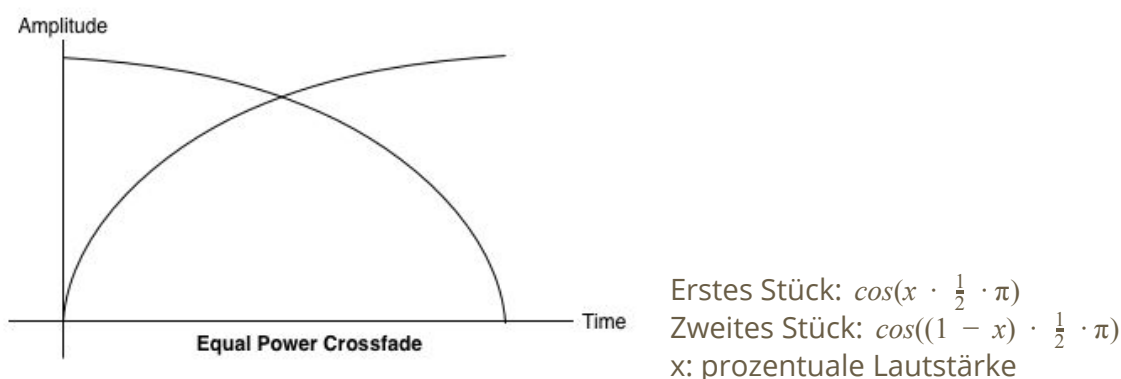
Beispiel: Koordinate 0,0 liegt in der oberen linken Ecke des Browserfensters, 1920,1080 würde bei einem Full-HD Bildschirm im Vollbildmodus unten rechts liegen - in der Realität ist dieser Bereich aufgrund der Taskleiste und Browser-Leiste oft kleiner.

World-space-Koordinaten geben die Position von Elementen in der Map an. So kann ein Objekt beispielsweise bei 2500,2500 liegen, welches erst sichtbar wird, wenn der Spieler sich dort hinbewegt, da der Viewport-Bereich kleiner ist. Da die Elemente alle am Grid ausgerichtet sind, sind die meisten world-space-Koordinaten ein Vielfaches von 64 (der

Größe der einzelnen Tiles im Grid). Als Einheit wird bei den Koordinaten in Pixeln gerechnet.

Audio

Das Audio-System der Engine wurde komplett in HTML5's Audio-API⁹ realisiert. Alle Sound-Dateien des Spiels werden zu Beginn in Buffer geladen, sodass sie on-demand schnell abgespielt werden können. Die Musik ist prozedural, das heißt, je nach dem, wie viel Steine der Spieler aufgesammelt hat, verändert sich die Musik. Bei einem Wechsel wird zuerst der Buffer des nächsten Stückes geladen, und dann mit Hilfe von Gain-Nodes zwischen den beiden Stücken gleichmäßig gewechselt. Damit der Wechsel glatter und flüssiger verläuft, ist er nicht komplett linear, sondern an eine mathematische Funktion angepasst.



Sound

Die Soundelemente des Spiels wurden größtenteils von uns selber hergestellt, um so genau die Sounds im Spiel zu benutzen, wie wir uns sie vorgestellt haben. Das beinhaltet Kisten, Fußschritte, verschiedene Laser-Sounds, Papier, Scheinwerfer, Einsammeln der Steine und Fackeln. Um diese Sounds aufzunehmen wurde unsere Kreativität gefragt, und für jeden Sound haben wir eine Lösung gefunden, welche wir dank professionellem Equipment schnell verwirklichen konnten.

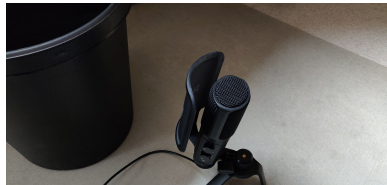
⁹ <https://www.html5rocks.com/en/tutorials/webaudio/intro/>



Kisten-Öffnen und
-Schließen



Papier-Knittern und
Schieben einer Kiste



Scheinwerfer-Sound: Schlag
mit nachträglich viel Hall



Laser-Sound: synthetisch
produziert

Aufgenommen wurden die Sounds mit dem RØDE NT-USB Mikrofon, die synthetisch produzierten mit dem Yamaha MOXF8-Synthesizer. Geschnitten wurde mit Audacity.

Musik

Die Hintergrundmusik des Spiels ist ebenfalls selbst produziert und entstand, nachdem das Spiel vom Design her fertiggestellt war. So hatte ich die Möglichkeit, die Musik atmosphärisch an den kühlen, dunklen Look&Feel der Map anzupassen. Es wurde ein dunkler, pulsierender Synthesizer-Sound verwendet, der dieses Theme musikalisch unterstützt, um die Immersion von Dunkelheit und Kälte beizubehalten. Die Musik ist prozedural: je mehr Steine der Spieler aufnimmt, desto "heller" wird die Musik.

Um dies zu realisieren habe ich 7 loopbare Versionen erstellt, jede Version enthält einen Ton mehr gegenüber der vorherigen Version, sodass nach ein paar Steinen ein heller Akkord ertönt. Bei den letzten beiden Steinen wechselt der Akkord, um eine Spannung aufzubauen und dem Spieler zu signalisieren, dass es zum Ende hin geht. Beim Interagieren mit dem Gauntlet - und damit beim Abschließen des Spiels, wird gewechselt auf eine Version die einen fröhlichen Dur-Akkord spielt, der sich nach und nach aufbaut. Um der Tribut-Szene an Stan Lee etwas mehr Emotionalität zu verleihen, blendet der Synthesizer aus und eine selbst eingespielte Piano-Musik begleitet das Outro.

Produziert wurde die Musik mit Cubase Elements 10, Keyscape und dem Yamaha MOXF8.

Schlusswort

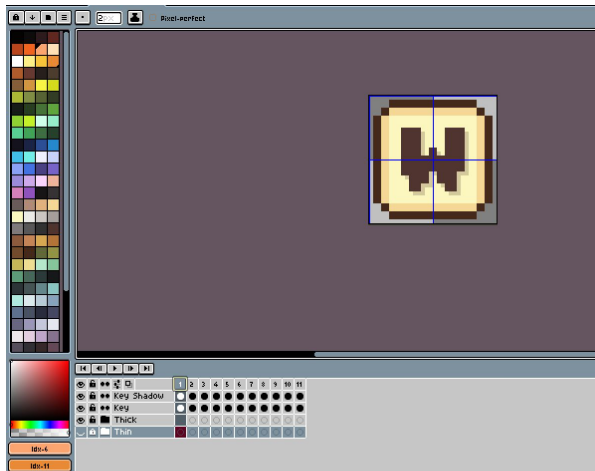
Das Projekt war eine tolle Möglichkeit, uns als Team näher kennenzulernen, und rauszufinden, wer von uns welche Fähigkeiten hat. Die Gruppenaufteilung war sinnvoll, um möglichst effizient zu arbeiten, und Kommunikationsprobleme im Team gab es kaum. Die Arbeit verlief konzentriert und zielstrebig. Auch gelernt haben wir, dass wir die Zeit die wir für die Entwicklung brauchen im Design-Dokument viel zu niedrig eingeschätzt haben, allein die Entwicklung des Konzeptes und der Engine nahm ca. 70% der Zeit in Anspruch.

Verbesserungswürdig ist...

...zum Großteil die Zeiteinteilung. Der Aufwand, den wir betrieben haben, war zwei Monate vor Abgabe noch recht gering, erst gegen Ende fingen wir an, auf Hochtouren zu arbeiten. Das nächste Mal sollte dies etwas gleichmäßiger verlaufen, um Stress zu vermeiden. Auch am Code hätte man einiges verbessern können, gegen Ende fühlten sich einige Lösungen wie Workarounds an, für die es mit Sicherheit mit mehr Zeit bessere Lösungen gegeben hätte. Ansonsten sind wir mit dem Projekt sehr zufrieden und freuen uns auf einen möglichen nächsten Teil in den nächsten Semestern.

Anhang & Quellen

Weitere Bilder



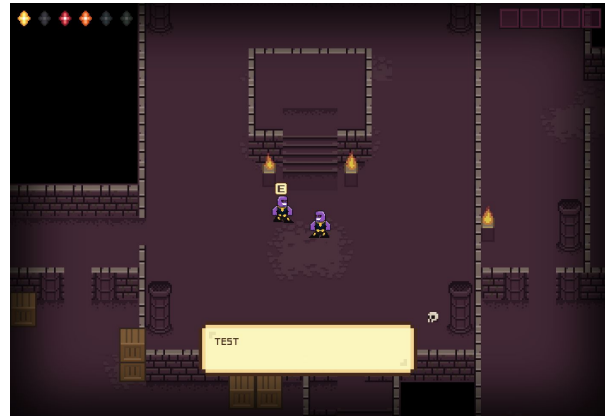
Produktion der Key-Overlays in Aseprite



Erster Erfolg, eine Tiled-Map ins Spiel zu importieren



"Der Text muss mittig sein" - David, 4:52 Uhr morgens



Erfolg, die UI zu implementieren

Quellen

Schriftart "Hauser" - Pixel Sagas - letzter Zugriff 23.01.2019

<https://www.dafont.com/hauser.font>

Schriftart "Pixeled" - OmegaPC777 - letzter Zugriff 23.01.2019

<https://www.dafont.com/pixeled.font>

Thanos Farmer Fanart - elleyart - letzter Zugriff 23.01.2019

<https://www.instagram.com/p/BmqYlypHFE0>

Dungeon Tileset - 0x72 - letzter Zugriff 23.01.2019

<https://0x72.itch.io/16x16-dungeon-tileset>

Thanos Comic Art - Marvel - letzter Zugriff 23.01.2019

<https://bookriot.com/2017/12/11/comics-to-read-about-thanos/>

Stan Lee - bgr.com - letzter Zugriff 23.01.2019

<https://bgr.com/2018/11/12/stan-lee-dies-marvel-comics-age-95/>

Intro Theme "The Sanctuary" - Tyler Bates - letzter Zugriff 23.01.2019

<https://www.youtube.com/watch?v=dc8pLanmBUs>

Unsere Arbeitsmotivation "Stratus" - Uyama Hiroto - letzter Zugriff 23.01.2019

<https://www.youtube.com/watch?v=MEgS9493djY>