Textheft zur Bachelorarbeit im Studiengang GAME DESIGN

Die Arbeit umfasst ein künstlerisch-technisches Werkstück sowie ein Artbook

NORR

Procedurally generated 2D backgrounds for an endless sidescroller prototype

Eingereicht von Marie Günther s0553048

04. Februar 2019

1. Prüferin Professorin Susanne Brandhorst

2. Prüfer Professor Thomas Bremer

Hochschule für Technik und Wirtschaft Berlin

Fachbereich Gestaltung und Kultur

Exposé vom 18.07.2018 (überarbeitet)
zum Antrag auf die Bachelorarbeit
Marie Günther, s0553048, Studiengang GAME DESIGN, FB 5

NORR

Procedurally generated 2D backgrounds for an endless side-scroller prototype

Im Rahmen meiner künstlerisch-technischen Bachelorarbeit möchte ich einen prozedural generierten 2D Hintergrund konzipieren, visualisieren und technisch umsetzen. Das System soll beispielhaft in einem 2D endlos-Scrollers umgesetzt und erprobt werden.

Prozedurale Generierung von diversen Einzelteilen in Videospielen wird immer wichtiger zur flexiblen Content-Generierung. Gerade für kleinere, aber auch für größere Firmen ist das eine Möglichkeit eine große Breite an Inhalt für verhältnismäßigen Arbeitsaufwand zu bieten. Außerdem vereint es Design und Technik auf eine sehr interessante Art und Weise.
Deswegen möchte ich eine Demo/ einen Prototyp entwickeln, der veranschaulicht, wie prozedural generierte 2D Hintergründe aussehen könnten. Besonders wichtig ist mir dabei das ästhetische Verzahnen der einzelnen Assets.

Workflow

In der Konzeption und der Umsetzung möchte ich zunächst einem gängigen Workflow folgen. Nach der bildlichen Analyse von Recherchematerialien werde ich mögliche Module identifizieren und skizzieren. In Verbindung damit werde ich einen geeigneten Artstyle entwickeln und anhand einiger 2D Concepts austesten, wobei der Fokus auf der Wirkung und der Umsetzbarkeit liegt.
Auf der technischen Ebene werde ich erst einen Prototypen erstellen, um das prozedurale Generieren auszuprobieren und dann anhand der gewonnenen Erfahrung ein Tilesystem konzipieren. Dabei werde ich verschiedene Größen und Möglichkeiten der Verzahnung der Assets testen, um in Verbindung mit den Artconcepts schließlich den Sidescroller umzusetzen. Den ganzen Prozess werde ich in meiner Dokumentation zur Arbeit erörtern.

Werkstück

Das zentrale Werkstück meiner Arbeit wird ein 2D endlos-Sidescroller. Er soll die Hintergründe prozedural generieren und verschiedene Landschaftstypen mit unterschiedlicher Vegetation darstellen. Zum Werkstück gehören außerdem Zeichnungen und Skizzen, 2D Concepts, Pläne, Karten und Screenshots. Die Ergebnisse werde ich in einem Art Book mit seperatem Textteil vorstellen und dokumentieren.

Inhaltsverzeichnis

1 Einführung 6

1.1 Motivation 6

1.2 Umfang der Arbeit 7

2 Hauptteil 9

2.1 Technisches Konzept 9

2.1.1 Design Templates 12

2.1.2 Mehrfach-Generierung 12

2.1.3 Tile-Generator 13

2.2.1 Referenzen und Moodboard 14

2.2.2 Artstyle Tests 16

2.2.3 Kopier-Workflow 18

2.2.4 Die Key-Illustration als Umgebung, um einen Artstyle für andere Objekte zu kreieren 20

2.2.5 Farben 22

2.2.6 Setting 24

2.3 Kombination von TECH und ART 26

3 Digitaler Prototyp 28

3.1 Start 28

3.1.1 Bewegung 28

3.1.2 Spawnen/Erstellen der Hintergrundelemente 29

3.1.3 Regionen 31

3.1.4 Zonen 31

3.2 Mehrfach-Generierung 33

3.2.1 Code Qualität 34

3.3 Sprite Shapes 36

3.4 Content Creation 40

4 Auswertung 42

4.1 Feedback 42

4.1.1 Feedbacktreffen 42

4.1.2 Interim 44

4.3 Evaluation des Workflows 47

5 Anhang 50

5.1 Abbildungsverzeichnis 50

5.2 Quellenverzeichnis 52

6 Danksagung 53

7 Erklärung der Autorenschaft 55

1 EINFÜHRUNG

1.1 Motivation

Für meine Bachelorarbeit wollte ich etwas, das mich widerspiegelt. Ich wollte ein Thema bearbeiten, bei dem ich etwas Neues lerne. Zusätzlich wollte ich auf mein bereits vorhandenes Wissen aufbauen. Deswegen war mir relativ schnell klar, dass ich keine rein künstlerische oder rein technische Arbeit, sondern eine künstlerisch-technische Arbeit wollte. Die Idee war, beide Bereiche zu verknüpfen. Über meine Entscheidung für prozedural generierte Landschaften bin ich sehr froh, da sie beides kombiniert und beide Teile stark ineinander greifen und voneinander abhängig sind.
Ich denke meine Arbeit ist eine sehr gute Grundlage, um sich weiter mit prozeduraler Generierung zu beschäftigen.

Ich hatte in den letzten Monaten und Jahren immer besonders viel Spaß an Spielen, die es schafften Narration durch mehr als nur Texte zu vermitteln. Besonders interessant war dabei für mich, wenn Geschichte und Worldbuilding durch die Landschaft und deren Entdeckung erzählt wurde. Spiele wie „The Banner Saga“ oder „Kingdom Classic“ empfinde ich dabei als eine große Inspiration, da sie es schaffen eine unglaubliche Neugier zu wecken. In Kombination mit einem sehr entspannendem Gameplay hat mich das dazu motiviert, meine eigene Herangehensweisen an so ein Spiel zu entwickeln. Zusätzlich wollte ich nicht nur einen simplen Klon erschaffen, sondern meine eigenen Ideen umsetzen. Also habe ich den Fokus auf das Entdecken gesetzt und bin dabei schnell auf prozedurale Generierung aufmerksam geworden. Der Ansatz, etwas

prozedural zu generieren ist unglaublich interessant, da es auch einem kleineren Team die Möglichkeit gibt, sehr viel Inhalt zu bieten. Steht erstmal das Grundgerüst, ist es verhältnismäßig einfach, neuen Content zu erstellen und das bereits Vorhandene zu erweitern. Das war für mich sehr interessant, weil es mir so möglich war, mit wenigen Ressourcen eine große Vielfalt an Landschaften zu erstellen.
Außerdem empfand ich es als äußerst interessant einen Artstyle zu entwickeln, der genau auf die Bedürfnisse von prozeduraler Generierung zugeschnitten ist.

1.2 Umfang der Arbeit

Werkstück:

endloss scroller, 2D, ohne Gameplay (.exe)

Außerdem:

- Artbook
- Dokumentation/Textbook
- 2 min. Video des Prototypen
- 10 Screenshots
- 10 Abbildungen der Hintergrund Assets
- 10 Abbildungen zum Making-Of

2 HAUPTTEIL

2.1 Technisches Konzept

Für meinen technischen Prototypen habe ich mir zuerst die Frage gestellt, was ich genau will. Außderdem habe ich mir Probleme überlegt, die auftreten könnten. Dadurch wollte ich von Anfang an Limits festlegen, um später nicht meinen Fokus zu verlieren. Mögliche Fragestellungen waren:

- prozedurale Generierung, was eignet sich für mich?
- Verknüpfung der Teilstücke, wie könnte man das umsetzen?
- Tag-Nacht-Wechsel, was muss ich beachten?
- Artstyle in Bezug auf die technischen Limitierungen, was eignet sich und sieht gut aus?

Diese Fragen haben mir von Anfang an geholfen, einen Blick für das Endprodukt zu erhalten. Zum Beispiel war für den Tag-Nacht-Wechsel wichtig, dass die Sprites mit einer Unity-Lichtquelle interagieren können. Somit wurden also einige offene Fragen beim Aufsetzen des Projektes bereits leicht limitiert.

Parallel habe ich angefangen mich genauer mit prozeduraler Generierung zu beschäftigen. Welche Konzepte sind sinnvoll, was sollte man beachten und welche technischen Vorraussetzungen benötige ich in meinem Projekt. Da dies ebenfalls einen großen Impact auf die Grundstruktur meines Projekts hatte, habe ich die Recherche direkt zu Beginn des Projekts gestartet. Ein äußerst hilfreiches Video war „How (and Why) Spelunky Makes its Own Levels | Game Maker‘s Toolkit“ von „Game Maker‘s Toolkit“[1]. Darin wird Schritt für Schritt erklärt, wie die Logik hinter den Levels von Spelunky funktioniert. Durch die

1 https://www.youtube.com/watch?v=Uqk5Zf0tw3o

prozedurale Generierung der Levels, war der Entstehungsprozess durchaus interessant für mich. Das Video bezieht sich auf das Buch „Spelunky by Derek Yu“ von Derek Yu, dem Entwickler von Spelunky. Anhand dieses Videos habe ich mir die einzelnen Schritte aufgeschrieben und verschiedene Designprinzipien und Herangehensweisen an Zufall und Game Design herausgearbeitet. In meiner Arbeit gibt es kein Gameplay, die generierten Landschaften können also keine Auswirkung darauf haben. Trotzdem war es extrem interessant dieses Prinzip zu studieren, da ich so ohne Schwierigkeiten in der Lage wäre, Gameplay und Game Design mit in die prozedurale Generierung einfließen zu lassen.
Eine interessante Arbeitsweise, die ich aus dem Video entnommen habe, war das mehrfache „in sich generieren“. Dabei besteht die Levelerstellung aus mehreren Teilen. Als erstes wird die grobe Art des Raumes generiert. Für jede Art gibt es vordesignte Templates/Varianten. Auch das war ein wichtiger Input für mich. Prozedural Generieren heißt nicht, dass alles dem Zufall überlassen wird. Gerade in Hinblick auf Game Design ist es wichtig bestimmte Teilgebiete zu designen, um sicher zu gehen, dass sie auch wirklich funktionieren. Abwechslung und Diversität wird dann durch verschiedene Templates, sowie eine Generierung der Details erzeugt. So wird sicher gestellt, dass jedes Level spielbar ist und einen ähnlichen Schwierigkeitsgrad besitzt. Im Video werden die Templates dann nochmal leicht verändert, so dass das Grunddesign vorhanden bleibt, aber die Visualisierung unterschiedlich ist (z.B. durch verschiedene Texturen). Als letztes werden kontextsensitiv die Items im Level verteilt.

Ein weiteres Video, was mir sehr viel beim technischen Konzept geholfen hat, war das Video „RANDOM MAP GENERATOR - EASY UNITY TUTORIAL“ von „Blackthornprod“[2]. Hier wurde ebenfalls das Prinzip der mehrfachen Generierung genutzt.

2 https://www.youtube.com/watch?v=D4E0gZyNk-k

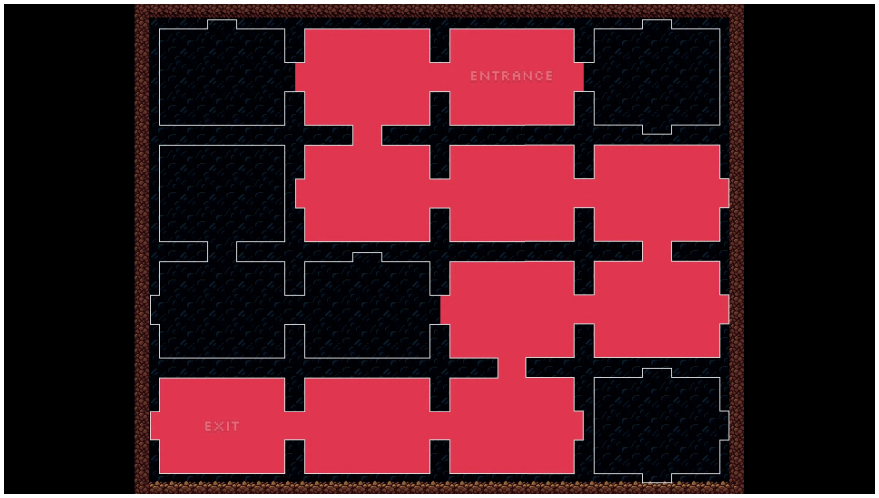


Abbildung 1: Videoauschnitt von Game Maker's Toolkit

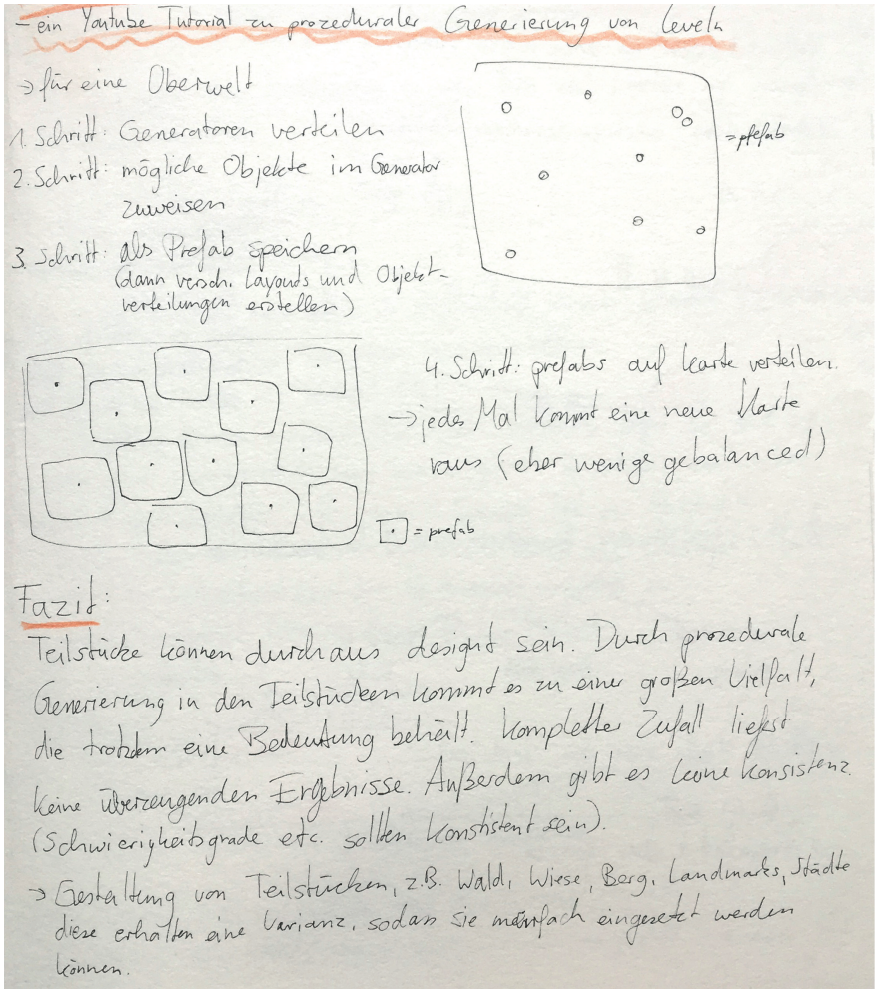


Abbildung 2: Notizen für das Generierungskonzept

Beide Beispiele hatten andere Perspektiven und Darstellungen, als ich sie nutzen wollte, aber trotzdem waren diese Videos sehr hilfreich. Nachfolgend sind die abgeleiteten, technischen Konzepte für mein Projekt gelistet:

2.1.1 Design Templates

Vor-designte Templates schaffen Ordnung, da sie ein konkretes Ziel verfolgen und dafür sorgen, dass dieses Grundziel auch auf alle Instanzen angewandt wird.

2.1.2 Mehrfach-Generierung

In Verbindung mit Design Templates hat die Mehrfach-Generierung das Potential eine riesige Vielfalt zu erschaffen.

Mein Projekt habe ich wie folgt aufgebaut (siehe Abb.3).

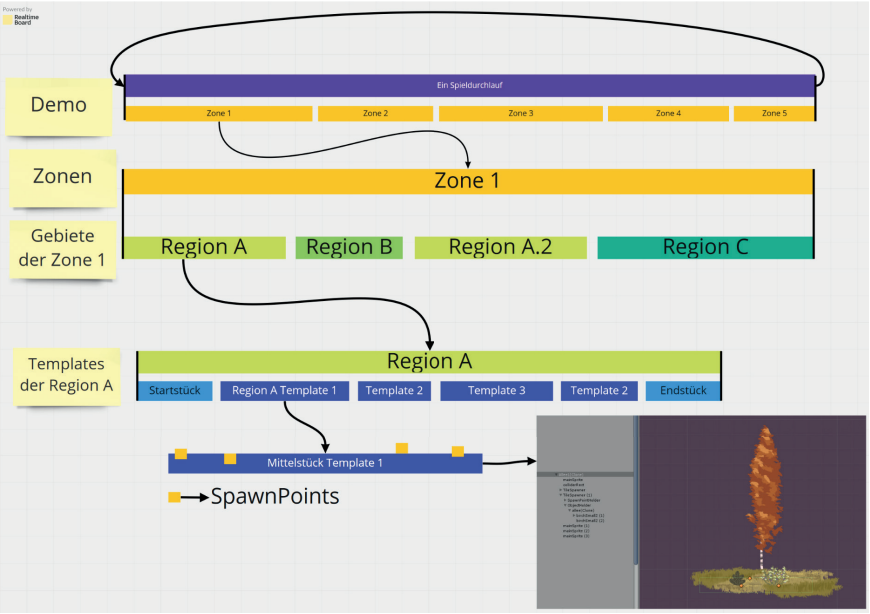


Abbildung 3: technischer Aufbau des Projektes, Schaubild

2.1.3 Tile-Generator

Tiles sind Kacheln, bzw. Details, die in die Landschaft eingebaut werden. Sie sind eigenständig und erweitern die Landschaft. Den Tile-Generator habe ich erstellt, um die Details in den Teilstücken der Gebiete schnell und unkompliziert variieren zu können. Dabei war es mir wichtig, dass das Script möglichst einfach zu bedienen und gleichzeitig sehr stark und vielseitig einsetzbar ist. Deswegen gibt es im Editor verschiedene Einstellungsmöglichkeiten, die eine große Wiederverwendbarkeit gewährleisten.

Der Tile-Generator besteht aus dem Script, mehreren Spawnpoints und einer Liste an spawnbaren Prefabs. Spawnpoints sind Positionen im Raum, an denen ein Detail erschaffen/eingesetzt wird. Die Spawnpoints werden per Hand gesetzt, ebenso, wie die Prefabs gezielt zugewiesen werden. Der Tile-Spawner ist damit Teil des Design Templates. Die Einstellungsmöglichkeiten sind dafür da, dass nicht nur z.B. Bäume generiert werden können.
Diese herausgearbeiteten Prinzipien kann man auch im fertigen Prototypen erkennen.

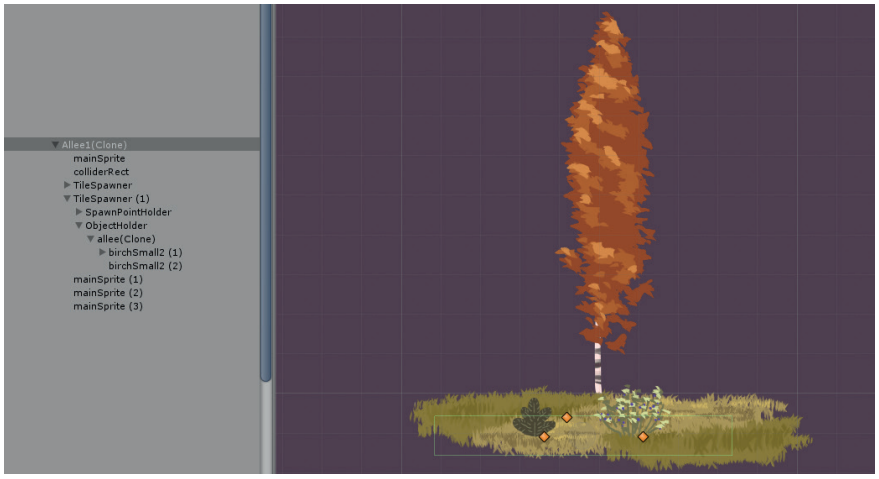


Abbildung 4: Aufbau im fertigen Prototypen

2.2 Künstlerisches Konzept

2.2.1 Referenzen und Moodboard

Da ich bisher recht wenig Erfahrung hatte was Environments angeht, habe ich für die künstlerische Konzeptionierung viel mit Referenzen gearbeitet. Gerade am Anfang habe ich sehr breit gefächert gesucht, also mich auf keinen speziellen Stil festgelegt. Zum Ende des Projekts hin habe ich gezielter nach Referenzen gesucht, die zu meinem Stil gepasst haben.

Neben Referenzen für den Artstyle und die Arbeitsweise, habe ich auch Fotoreferenzen für die Landschaft rausgesucht. Dafür habe ich Screenshots über Google Streetview erstellt. Ich habe mir die verschiedenen Landschaftstypen in Schweden angesehen und stellvertretend Screenshots gespeichert. Diese wurden Referenzen für die speziellen Bäume und Pflanzen Schwedens, aber auch für die Häuser und Straßen.
Gerade für die Natur waren diese Screenshots äußerst hilfreich, denn es ist recht schwer gute Referenzen für bestimmte Baumarten zu finden. Aber auch herauszufinden, welche Bäume und Pflanzen es in einer bestimmten Region gibt, gestaltete sich schwierig. Außerdem konnte ich dann meinen Stil auf die einzelnen Details anwenden und so authentische, sowie zum Stil passende Assets erstellen.

Ich denke für fast jedes Projekt sind Referenzen unglaublich wichtig. Mir haben sie sehr geholfen, um eine Richtung zu finden und die Landschaft authentisch zu gestalten. Auch bei zukünftigen Projekten werde ich so arbeiten.

Abbildung 5: Moodboard, Referenzen

Abbildung 6: Fotoreferenzen von Schweden

2.2.2 Artstyle Tests

Bei dem Artstyle ging es mir in erster Linie um die Umsetzbarkeit. Deshalb habe ich bei den ersten Tests komplett unabhängig vom Artstyle experimentiert und dann in Kombination mit Referenzen verschiedenste Sachen ausprobiert. Das war für mich eine relativ große Herausforderung. Außerdem kamen mehrere Schwierigkeiten hinzu. So verändert sich die Landschaft stets und ist dauerhaft in Bewegung. Eine starre Komposition war für mich deswegen keine Hilfe. Stattdessen habe ich mehrere Thumbnails gezeichnet, um ein Gefühl für mögliche Kompositionen zu erhalten. Anhand dessen habe ich mir überlegt, was interessant sein könnte, und wie die verschiedenen Landschaftstypen verbunden sein könnten. Dabei ist mir aufgefallen, dass man fast alle von mir gezeichneten Landschaften gut in verschiedene Ebenen einteilen könnte und gerade Vorder- sowie Hintergrund mehr oder weniger voneinander unabhängig sein können. Durch die Kombination von verschiedenen Vorder- und Hintergründen wäre es möglich, mit wenigen Sprites eine große Vielfalt zu erschaffen.

Ich habe eine Vielzahl von Artstyle Tests angefertigt und so meine Schwächen und Stärken analysiert.
Hilfreich war für mich also, die Anforderungen auf verschiedene Tests aufzuteilen. Die Thumbnails waren eher für das Verstehen von Landschaften und die Artstyle Tests haben sich gut für das Finden eines Artstyles geeignet, da ich damit sehr offen war und viel experimentieren konnte.

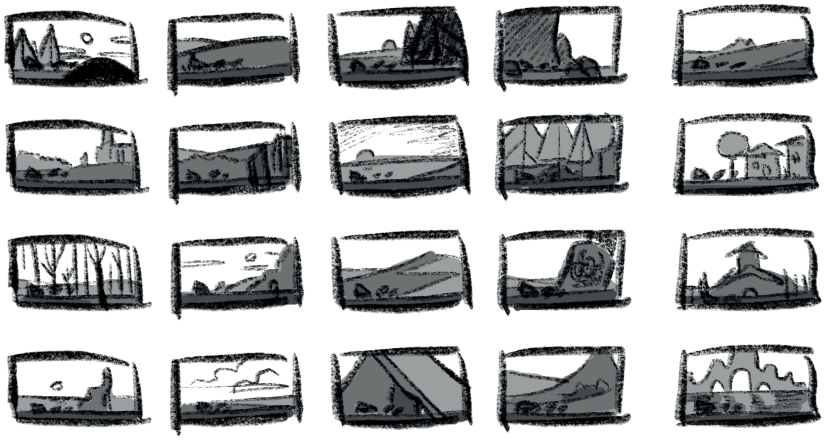


Abbildung 7: Thumbnails

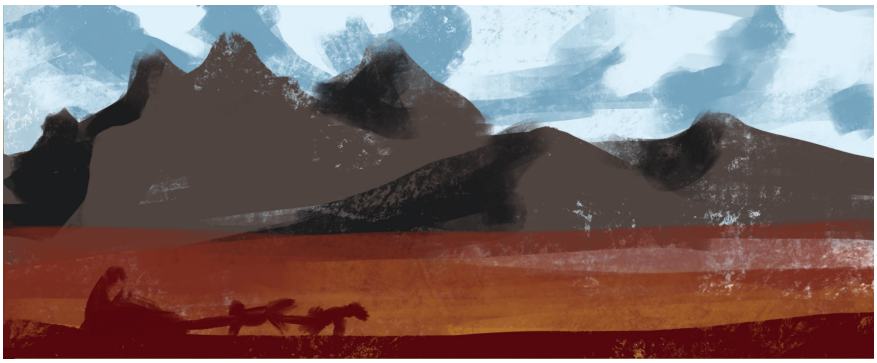


Abbildung 8: Artstyle Test 1

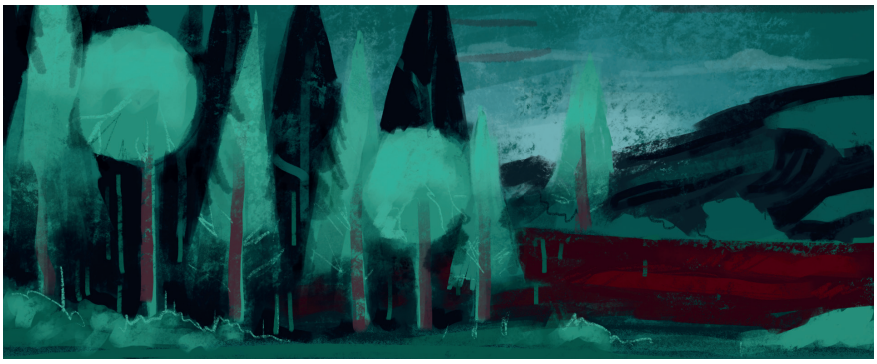


Abbildung 9: Artstyle Test 2

2.2.3 Kopier-Workflow

Um einen passenden Artstyle zu finden, habe ich verschiedene Workflows ausprobiert. Von sehr freien Tests, über relativ direkte Kopien. Der Workflow, der mir wohl am meisten geholfen hat, war der Kopier-Workflow.

Dafür habe ich mir eine Illustration ausgesucht, deren Stil mir sehr gefallen hat. Dann habe ich dieses Bild genau analysiert. Warum gefällt mir der Stil, wie werden die Formen genutzt, wie die Farben, Texturen, Details, etc. Dann habe ich das Ganze kopiert. Ich habe die Farben aus dem Bild genommen und versucht das Bild so originalgetreu wie möglich zu kopieren. Der Fokus lag dabei auch auf Dingen wie: wie viele Ebenen verwende ich, welche Designelemente kann ich wiederverwenden, etc.
Durch das Kopieren habe ich gelernt was den Stil ausmacht. Außerdem habe ich festgestellt, was mir vielleicht nicht so gefällt, was ich verbessern könnte und wie ich meine eigenen Ideen einbringen kann. Meine Illustration habe ich dann als Grundlage genommen, um den Stil auf andere Objekte anzuwenden. z.B. Natur. Durch den Wechsel des Settings, ist der Stil etwas ganz eigenes geworden. Es nutzt einen ähnlichen Aufbau, aber ich konnte eine eigene Richtung erschaffen.

Nachdem ich mit dem Stil für die Häuser zufrieden war, konnte ich Skizzen machen und den Stil auf die schwedischen Häuser anwenden. Dann habe ich ART und Technik/TECH verschmolzen und ein Konzept entwickelt, wie ich aus einer geringen Anzahl an Häusern eine große Vielfalt erzeugen kann. Die Häuser bestehen aus zwei Teilen, die beliebig kombiniert werden können. Dafür habe ich in Unity einen House-Creator gebaut, der die Teile zusammensetzt.

Abbildung 10: Die Originalillustration von Benjamin Plouffe

Abbildung 11: Meine Kopie

2.2.4 Die Key-Illustration als Umgebung, um einen Artstyle für andere Objekte zu kreieren

Nach dem Kopier-Workflow hatte ich einen Artstyle für Häuser. Da mein Projekt aber hauptsächlich Landschaft sein sollte, habe ich das als Ausgangslage/ Key-Illustration genutzt, um anhand des bereits bestehenden Stils einen Artstyle für alles Organische zu entwickeln.

Dafür habe ich die Illustration als Umgebung genutzt, um verschiedene Stile für Objekte wie Bäume und Büsche auszuprobieren. Durch den ständigen direkten Vergleich zum festegelegten Artstyle, konnte ich extrem gut feststellen, ob der aktuelle Artstyle für Bäume etc. mit dem anorganischen Artstyle funktioniert. Außerdem hatte ich so bereits eine grobe Richtung und verschiedene Regeln, die ich auf die Bäume anwenden konnte. Das hat mir sehr viel geholfen. Zusätzlich habe ich mir viele verschiedene Referenzen angeguckt, um auf Ideen zu kommen. Besonders die Laubbäume empfand ich als große Herausforderung. Es war schwer den richtigen Grad zwischen Realismus und Stilisierung zu finden. Aus diesem Grund bin ich sehr viele Iterationen durchgegangen, bis ich bei dem Stil angekommen bin, den ich am Ende benutzt habe.

Im Gegensatz dazu fand ich es verhältnismäßig einfach einen Stil für die Nadelbäume zu finden. Ihre Form konnte ich besser abstrahieren und im passenden Stil umsetzen.

Abbildung 12: Die Baum-Tests neben dem fertigen Haus

2.2.5 Farben

Bei den ersten Artstyle Tests ist mir aufgefallen, dass es mir recht schwer fällt Farben für eine komplette Szenerie zu wählen. Man muss verschiedenste Dinge beachten, wie zum Beispiel Atmosphäre, Tageszeit, Harmonie und Disharmonie von verschiedenen Gegenständen. Alle Farben müssen miteinander funktionieren, aber es soll auch spannend aussehen. Am Anfang war das sehr schwierig für mich.

Meine erste Lösung war eine Farbpalette, die die Farbauswahl deutlich eingeschränkt hat. Ich habe mir dafür verschiedene Farbpaletten angeschaut und dann mit diesem Wissen eine Eigene erstellt. Die Farben waren sehr warm und es gab viele Orange-/Rottöne. Beim praktischen Einsatz ist mir immer mehr aufgefallen, dass die Naturfarben (also Grün etc) nicht gut mit dem Rest funktionieren.

Deswegen habe ich dann mit Fotoreferenzen gearbeitet, um die Palette zu aktualisieren und die Farben anzupassen. Dafür habe ich mir passende Fotos aus meinem Recherche-Repertoir (Screenshots von Google Streetview) gesucht und Farben direkt aus dem Bild genommen. Diese habe ich dann in den Kontext meines Projekts gesetzt, also teilweise etwas verdunkelt oder aufgehellt, und auf meine bereits bestehenden Assets angewandt. Das hat sehr gut funktioniert und sorgt jetzt dafür, dass die Landschaften sehr organisch und spannend aussehen.

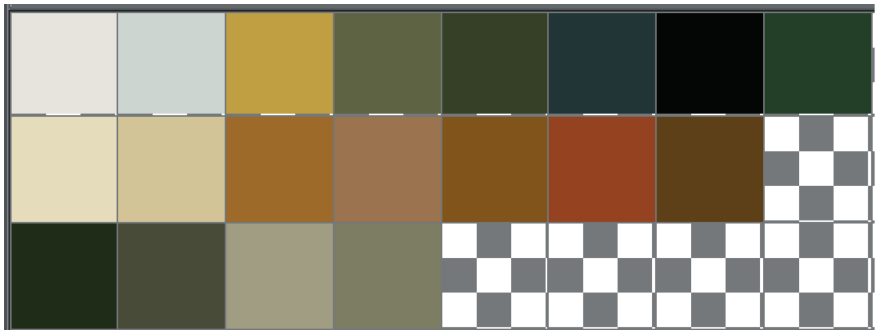


Abbildung 13: Color Palette

Abbildung 14: Eines der Referenzfotos für die Farben

2.2.6 Setting

Meine erste Idee sollte in Alaska spielen. Der Spieler sollte einen Hundeschitten fahren und hätte die Möglichkeit, in regelmäßigen Abständen Input zu liefern, um die Schlittenhunde anzutreiben. Das Ganze sollte auf meinem Charakter „Edgar“ aus dem ersten Semester aufbauen.

Mit diesem Setting im Hinterkopf habe ich die ersten Wochen an dem Artstyle gearbeitet und dabei immer wieder gemerkt, dass es mir schwer fällt diese Landschaft umzusetzen. Außerdem gab es das Problem, dass ich immer Schnee (mindestens auf dem Weg) benötige. Man hätte kleine Animationen machen können, um das Problem der Glaubhaftigkeit zu lösen, aber ich denke alles in allem wäre das recht aufwendig geworden und das Endergebnis wäre nicht so vielfältig geworden, wie es jetzt ist.
Der größte Pluspunkt für das Alaska Setting war, dass dort ein interessantes Gameplay vorhanden wäre. Gameplay sollte aber in meinem Prototypen keine Rolle spielen, also habe ich mich entschieden, nochmal komplett offen an das Setting heranzugehen.

Erst hatte ich überlegt eine Art Kutsche in einer Fantasywelt zu entwickeln. Aber nach weiterem Recherchieren und Feedbacktreffen hat sich immer mehr die Idee eines modernen Settings durchgesetzt. Ausschlaggebend war dabei auch die Aussage, dass das zu dem Zeitpunkt vorhandene Projekt das Gefühl eines Roadtrips weckt. So bin ich dann schließlich zum Motorrad gekommen und aus der finalen Sicht bin ich sehr zufrieden mit der Entscheidung.

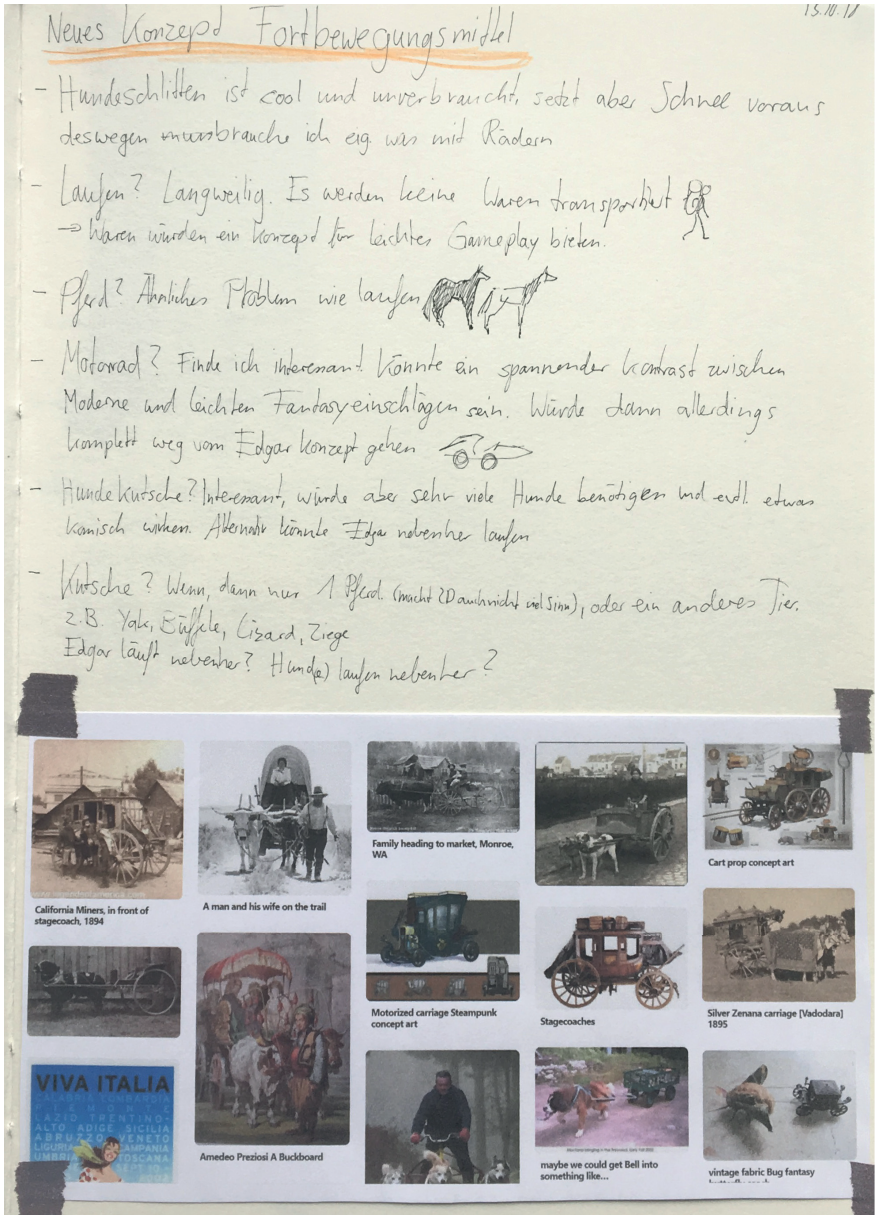


Abbildung 15: ein weiteres Konzept für das Setting

Abbildung 16: Recherche Motorrad Setting

2.3 Kombination von TECH und ART

Da TECH und ART in meiner künstlerisch-technischen Arbeit eng verzahnt sind, habe ich regelmäßig zwischen den beiden Teilgebieten gewechselt. So ist der Fortschritt in allen Gebieten gleichmäßig und ich konnte von den gegenseitigen Erfahrungen beider Gebiete profitieren.

Zum Beispiel habe ich bei meinen Artstyle Tests schnell festgestellt, dass ich bei Landschaften gut mit Ebenen arbeiten kann. Dieses Wissen konnte ich dann auf die technische Seite in Unity übertragen und den Aufbau der Sorting Layer nach dem gleichen Prinzip anlegen.

Dieser Wechsel zwischen den Bereichen war sehr hilfreich, da ich stets neue Inspiration und Einflüsse bekommen habe und ich keinen Bereich vernachlässigt habe.
Aber auch Limitationen und Vorraussetzungen waren gebietsübergreifend.
Die Implementierung der Assets/Medieninhalte wurde maßgeblich durch den Artstyle bestimmt und gleichzeitig brachte die prozedurale Generierung gewisse Herausforderungen und Anforderungen an den Artstyle mit sich.

Ein weiteres Beispiel für die Vorteile vom parallelen Arbeiten an beiden Bereichen waren die Farben. Ausgehend von meinem digitalen Box-Prototypen hatte ich die Idee, die Assets im besten Falle weiß zu gestalten. Beim Spawnen in der Engine, könnte ich dadurch die Farben leicht variieren, oder ein Sprite in verschiedenen Farben für verschiedene Gebiete wiederverwenden. Als ich dann am Artstyle weitergearbeitet habe, ist mir aber recht schnell aufgefallen, dass es bei den meisten Sprites gar nicht möglich war, diese einfarbig weiß zu gestalten. Mit diesem Wissen habe ich dann meinen Prototypen angepasst und die ursprüngliche Idee verworfen. Die parallele Arbeitsweise hat dabei dafür gesorgt, dass mir dieses Problem sehr früh aufgefallen ist. Dadurch konnte ich es schnell ändern und mir ein besseres Konzept überlegen.

Im Endeffekt habe ich mich dann dazu entschlossen, die Sprites farbig zu importieren und in den Regionen die Farben leicht zu variieren, um eine höhere Farbvarianz zu erschaffen. Das Ganze lässt sich sehr gut im orangenen Birkenwald beobachten.

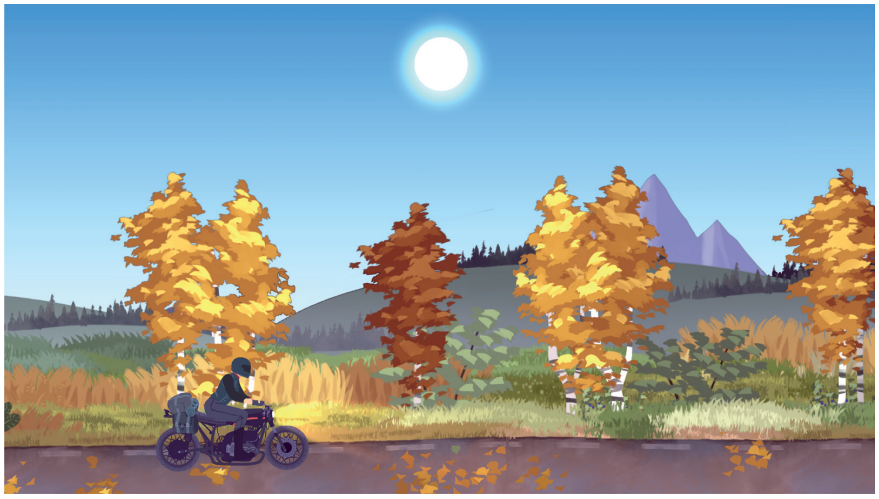


Abbildung 17: Farbänderungen im orangenen Birkenwald

3 DIGITALER PROTOTYP

3.1 Start

Ich habe versucht, bereits von Anfang an einen digitalen Prototypen zu erstellen. Damit wollte ich Konzepte direkt ausprobieren und integrieren. Im Verlaufe des Studiums und des Praktikums, hatte ich bereits mehrere Möglichkeiten, Spiele von Anfang an aufzubauen. Da es verschiedene Möglichkeiten gibt ein Spiel zu entwickeln, habe ich geschaut, was für mich in diesem Fall am Besten funktioniert. Im Ergebnis habe ich mich dafür entschieden, nicht mehrere Prototypen für verschiedene Mechaniken zu entwickeln, sondern alles in einem Projekt weiterzuentwickeln. Meine Gründe dabei waren vor allem, dass meine Vorhaben gut überschaubar waren und teilweise recht stark ineinander griffen. Das Projekt zwangsweise in verschiedene Gebiete aufzuteilen, hätte dem Ganzen nur geschadet.

3.1.1 Bewegung

So habe ich mit der grundlegendsten Sache angefangen, der Bewegung. Auch hier gab es mehrere Möglichkeiten, zwischen denen ich mich entscheiden musste. Den Charakter durch die Szene zu bewegen, klingt im ersten Moment natürlicher. Aber das sich ständig verändernde Koordinatensystem und die nicht vorhandene „Fixpunkte“ bringen große Probleme mit sich. Deshalb habe ich mich dafür entschieden, das Koordinatensystem fest an einer Stelle zu lassen und den Hintergrund zu bewegen. Das hat sich auch als eine sinnvolle Entscheidung herausgestellt.

3.1.2 Spawnen/Erstellen der Hintergrundelemente

Der nächste große Punkt war das Spawnen von Hintergrundelementen. Diese habe ich am Anfang des Projekts immer gleich gespawnt und in der weiteren Entwicklung des Prototypen dann zufällig. So habe ich langsam das prozedurale Generieren integriert. Hierbei habe ich dann auch direkt die Designprinzipien eingebaut, indem ich das Regionen-Prinzip implementiert habe.

Schwierig war es dafür zu sorgen, dass die gespawnten Teile nahtlos aneinander abschließen, da sie sich bewegen. Um so genau wie möglich zu sein, wird die exakte Position des letzten Stücks als Basis genommen. Dann wird die Zeit gemessen, die zum Instantiieren benötigt wird und mit der Geschwindigkeit verrechnet. So erhalte ich die genaue Position, an der das nächste Teilstück anfangen muss.

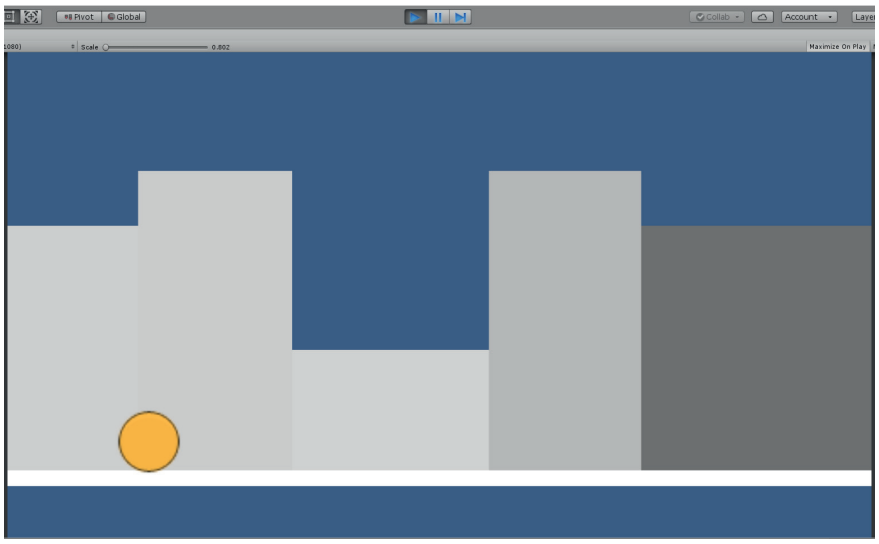


Abbildung 18: Prototyp mit Bewegung und Tile Generierung

```
private RegionPart SpecifyRegionPart(ObjectMove _lastObject)
{
    //don't need to specify the regionPart when background
    if (layer != 0 && layer != 1)
    {
        return RegionPart.Middle;
    }

    if(_lastObject.regionPart == RegionPart.End) //after an end part always follows an start part.
    {
        return RegionPart.Start;
    }
    else
    {
        if (Random.Range(0f, 1f) <= regionList[nextRegion].lengthPossibility) //chance is higher that the region continues than that it ends.
        {
            if (isGround)
            {
                if (nextRegionForeground == 0 || nextRegionForeground == 2 || nextRegionForeground == 3 || nextRegionForeground == 10)
                {
                    print("zoom back");
                    GameManager.i.SpawnZoomTrigger(GameManager.ZoomStates.Normal);
                }
            }
            return RegionPart.End;
        }
        else
        {
            return RegionPart.Middle;
        }
    }
}
```

Abbildung 19: Script, Regionpart-Picker, hier erfolgt die Auswahl, ob Start- Mittel- oder Endstück gewählt wird

```
private int SpecifyRegion(RegionPart _lastPart, int _lastRegion)
{
    if (_lastPart == RegionPart.End) //if the old region ends, pick a new region (maybe more limits)
    {
        List<Region> possibleRegionList = new List<Region>();

        foreach(Region _region in regionList)
        {
            if (_region.regionNumber == _lastRegion) //not 2 of the same region after one another
                continue;
            if(_region.IsPossible() && _region.suitableZones.Contains((int)GameManager.i.currentZone))
            {
                possibleRegionList.Add(_region);
            }
        }

        Region regionToUse = possibleRegionList[Random.Range(0, possibleRegionList.Count)];
    }

    Region regionToUse = possibleRegionList[Random.Range(0, possibleRegionList.Count)];
    regionToUse.RegionGetsUsed();
    if (regionToUse.regionNumber == cityRegionNumber)
        cityCounter++;

    //zoom in when grassland etc
    if (regionToUse.regionNumber == 6 || regionToUse.regionNumber == 7 || regionToUse.regionNumber == 8)
    {
        if (Random.Range(0f, 1f) > 0.6f)
        {
            //print("zoom out");
            GameManager.i.SpawnZoomTrigger(GameManager.ZoomStates.ZoomOut, 5);
        }
    }
    //zoom out when forest, city, etc
    if (regionToUse.regionNumber == 0 || regionToUse.regionNumber == 2 || regionToUse.regionNumber == 3)
    {
        if (Random.Range(0f, 1f) > 0.6f)
        {
            //print("zoom in");
            GameManager.i.SpawnZoomTrigger(GameManager.ZoomStates.ZoomIn);
        }
    }

    return regionToUse.regionNumber;

}
else //the next part has to be from the same region as the last.
{
    return _lastRegion;
}
```

Abbildung 20: Script, Region-Picker, hier wir die nächste Region gewählt, abhängig von verschiedenen Bedingungen

3.1.3 Regionen

Jede Region besitzt ein Start-, ein End- und beliebig viele Mittelstücke. Dieses Script (Abb. 19) schaut, welchen Typ das letzte Teilstück hatte. Immer am Ende einer Region, also mit der Auswahl des Startstücks, wird eine zufällige nächste Region bestimmt. Diese wechselt erst wieder beim nächsten Startstück. Das sorgt dafür, dass eine gewisse Ordnung in die Zufälligkeit gebracht wird. Durch das Verändern einer Variable ist es möglich Regionen potentiell länger oder kurzer zu gestalten (je niedriger der Wert von 0-1, desto länger ist die Region) Die Variable gibt an, wie wahrscheinlich es ist, dass ein Endstück gewählt wird. Ist der Wert 0.5, beträgt die Wahrscheinlichkeit, dass die Region direkt beendet wird 50%.

3.1.4 Zonen

Später habe ich dann die Regionen in Zonen organisiert. Dies richtet sich nach dem Designprinzip der Strukturierung. Das Spiel ist in 5 große Zonen aufgeteilt und jede Zone besitzt eine Liste an potentiell möglichen Regionen. Das heißt aber nicht, dass jede Region auch wirklich im Verlauf der Demo gespawnt wird. Das sorgt für einen Spannungsverlauf (die Regionen am Anfang sind eher bunt, die zum Ende hin eher dunkel), ohne dass jeder Spieler den exakt gleichen Verlauf hat. Es sorgt also ebenfalls dafür, dass der Zufall moderiert wird.

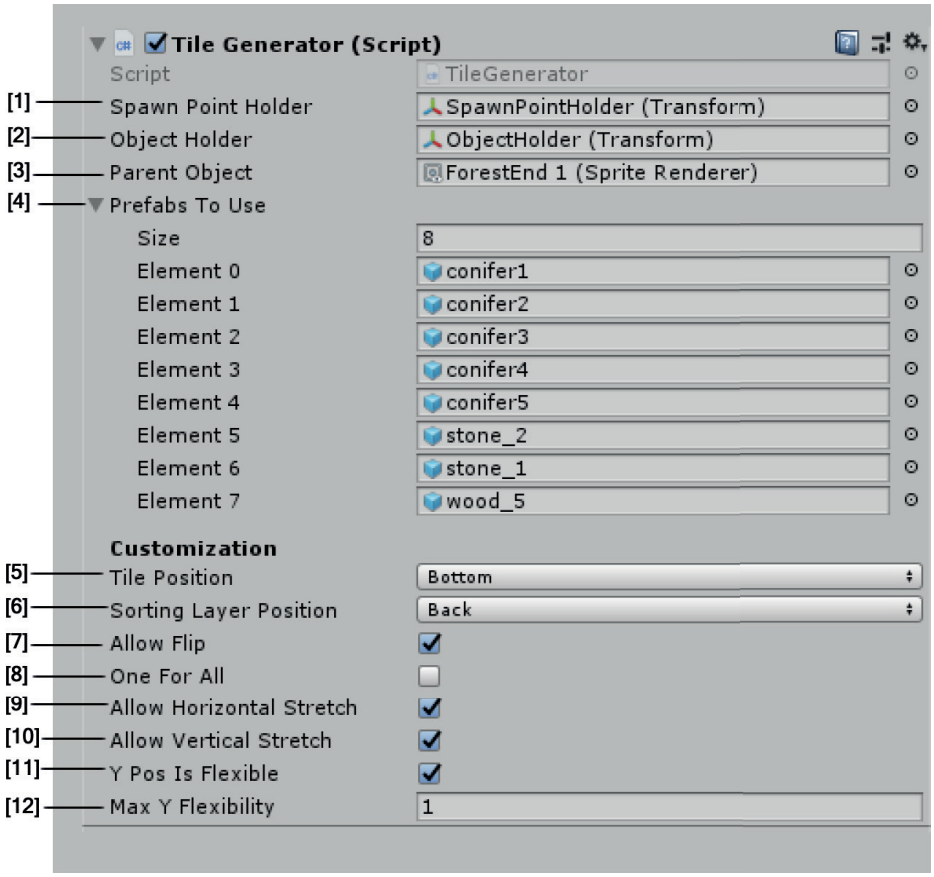


Abbildung 21: Editoransicht des Tile-Generators

3.2 Mehrfach-Generierung

Das Prinzip der Mehrfach-Generierung habe ich unter anderem in den Tile-Generatoren benutzt. Verankert im Regionenprinzip, gibt es verschiedenen Teilstücke (Start, End und Mittel), wobei es von jedem Typ mehrere handdesignte Templates/Variationen gibt. Unterstützt wird die Vielfalt, durch die Tile-Generatoren.
Im Verlauf werde ich detailliert auf die einzelnen Individualisierungsmöglichkeiten des Scripts eingehen.

[1], [2] und [3] sind Referenzen, die bei allen gleich sind. Sie verweisen auf den grundsätzlichen Aufbau. Also zum Beispiel, in welchem GameObject die Spawnpoints gespeichert sind. So kann man Anzahl und Position der Spawnpoints beliebig verändern, ohne an den Zuordnungen etwas ändern zu müssen.
[4] ist die Liste der möglichen Details. Je mehr es sind, desto geringer ist die Wahrscheinlichkeit, dass ein bestimmtes Detail gewählt wird. Möchte man eine höhere Priorität für bestimmte Details, kann man diese auch mehrfach hinzufügen. Außerdem ist es möglich, beliebig viele „blanks“ hinzuzufügen, also leere Prefabs/Details.
[5] bestimmt die Ausrichtung des Details. Also ob der Mittelpunkt, oder die untere Kante als Bezugspunkt gewählt wird. Bäume würde man durch die unterschiedlich großen y-Werte eher an der unteren Kante ausrichten. Fenster an einem Haus eher an dem Mittelpunkt.
[6] bestimmt die Positionierung innerhalb der SortingLayer. Der SortingLayer wird automatisch vom ParentObject übernommen. Die „Order in Layer“ ist abhängig von dem ausgewählten Codewort. „Back“ = -100 – 0, „Middle“ = 0 – 100, „Front“ = 100 – 200;

[7] gibt die Möglichkeit das Sprite zu spiegeln. Gerade bei organischen Objekten ist das eine einfache und effektive Methode, um Vielfalt zu erschaffen. Bestimmte Objekte, z.B. mit Schrift, sollten hingegen nicht gespiegelt werden.
[8], ist dieser Bool true, wird nicht für jeden Spawnpoint einzeln ein Detail ausgewählt, sondern nur einmal eins am Anfang für alle.
[9] und [10] geben die Möglichkeit, Details im Rahmen von 10% zu strecken.
[11] ermöglicht [12], der Wert gibt dabei an, wie weit nach unten der Spawnpoint versetzt werden kann. Eine leichte horizontale Versetzung gibt es immer.

3.2.1 Code Qualität

Da meine Arbeit nicht nur eine künstlerische, sondern auch eine technische Arbeit ist, war es mir wichtig, dass der Code strukturiert und sinnvoll ist. Ich habe sehr viel Wert darauf gelegt Funktionen sinnvoll generisch zu gestalten. Deshalb ist dieses Script sehr universell einsetzbar.

Ich habe im kompletten Projekt strikte Namenskonventionen benutzt und versucht den Code so gut es geht zu kommentieren. Das sorgt dafür, dass ich mich auch nach einer Pause, weil ich zum Beispiel an Assets gearbeitet habe, gut im Code zurecht finde.

Außerdem habe ich versucht, regelmäßig Code zu verbessern (Refactoring). Das sorgt dafür, dass der Code immer an der Gegenwart angepasst und genau auf das zugeschnitten ist, wofür er benötigt wird. So konnte ich so Umwege und unnötige oder veraltete Stellen im Code entfernen, die ansonsten Auswirkungen z.B. auf die Performance hätten.

```
private void SpawnObjects()
{
    int index = Random.Range(0, prefabsToUse.Count);
    foreach (Transform point in spawnPointList)
    {
        if (!oneForAll) //choose a new index for every point
        {
            index = Random.Range(0, prefabsToUse.Count);
        }
        GameObject nextPrefab = prefabsToUse[index];

        //changing scale
        Vector3 originalLocalScale = nextPrefab.transform.localScale;
        if (allowHorizontalStretch)
        {
            nextPrefab.transform.localScale = new Vector3(nextPrefab.transform.localScale.x *
        }                                Random.Range(0.9f, 1.1f), nextPrefab.transform.localScale.y, 1);
        if (allowVerticalStretch)
        {
            nextPrefab.transform.localScale = new Vector3(nextPrefab.transform.localScale.x,
        }                                 nextPrefab.transform.localScale.y * Random.Range(0.9f, 1.1f), 1);

        //POSITION
        Vector2 spawnPosition = Vector2.zero;
        if (tilePosition == TilePosition.center) //spawn the center of the sprite at the given point
        {
            spawnPosition = new Vector2(point.position.x, point.position.y);
        }
        else if (tilePosition == TilePosition.bottom) //spawn the bottom of the sprite at the given point
        {
            spawnPosition = new Vector2(point.position.x, point.position.y + [...])
            if (yPosIsFlexible)
            {
                spawnPosition -= new Vector2(0, Random.Range(0, maxYFlexibility));
            }
        }

        //INSTANTIATE
        GameObject nextObject = Instantiate(nextPrefab, spawnPosition, Quaternion.identity, objectHolder);

        SpriteRenderer renderer = parentObject;
        if (parentObject == null)
        {
            renderer = this.GetComponent<SpriteRenderer>();
        }

        //vary the order in sorting layer
        SpriteRenderer nextObjectRenderer = nextObject.GetComponent<SpriteRenderer>();
        nextObjectRenderer.sortingLayerID = renderer.sortingLayerID;
        if (sortingLayerPosition == SortingLayerPosition.back)
        {
            nextObjectRenderer.sortingOrder = Random.Range(-100, 0);
        }
        else if (sortingLayerPosition == SortingLayerPosition.middle)
        {
            nextObjectRenderer.sortingOrder = Random.Range(0, 100);
        }
        else if (sortingLayerPosition == SortingLayerPosition.front)
        {
            nextObjectRenderer.sortingOrder = Random.Range(100, 200);
        }

        nextObject.transform.position = spawnPosition; //needs to be assigned as the localPosition!
        nextObject.transform.position = new Vector3(nextObject.transform.position.x +
                    Random.Range(-0.4f, 0.4f), nextObject.transform.position.y, nextObject.transform.position.z);
        nextPrefab.transform.localScale = originalLocalScale;

        if (allowFlip)
        {
            if ((Random.Range(0, 2) == 1))
            {
                nextObjectRenderer.flipX = true;
            }
        }
    }
}
```

Abbildung 22: Script, Tile Generator

3.3 Sprite Shapes

Mit Unity 2018.1 kam ein neues Feature für die 2D Entwicklung[1]. Die Sprite Shapes. Sie ermöglichen es, ein Sprite einer Fläche oder einer Linie zuzuweisen und diese dann dynamisch zu verändern. Das sorgt bei bestimmten Artstyles für einen freieren Umgang mit Sprites, da nun sehr flexibel auf Veränderungen z.B. im Leveldesign eingegangen werden kann. Und auch ich konnte dieses neue Feature nutzen. Für die Landschaften, wie ich sie umgesetzt habe, ist die Technik zwar ungeeignet, aber für simple Formen, wie z.B. die Straße, bot sie viele Vorteile.

Ich habe ein Template gestaltet und dann ein Script geschrieben, mit dem beim Spawnen der Straße die Form beeinflusst wird. Spriteshapes in der Linienform besitzen auf ihrem Verlauf beliebig viele Points. Diese besitzen Werte wie die Position und die Rotation der Verformung. Damit jeder Streckenabschnitt etwas anders aussieht, gehe ich die Liste der Points durch und verändere den y-Wert. Dieser Wert liegt zufällig zwischen zwei Werten, die im Editor eingegeben werden können. So kann man mit diesem Script flache Straßen oder hügelige Feldwege erstellen. Nach einigen Versuchen habe ich auch festgestellt, dass die Spannen relativ klein sein müssen, damit die Straße nicht zu unruhig wird. Damit alle Straßenteile ordentlich verbunden sind, habe ich seamless/fortlaufende Texturen erstellt und den sich überlappenden Abschnitten den gleichen y-Wert gegeben. Der y-Wert des letzten Points wird gespeichert und für den ersten Point des nächsten Abschnitts benutzt. Die Points danach werden wieder zufällig bestimmt.

1 https://blogs.unity3d.com/2018/09/20/intro-to-2d-world-building-with-sprite-shape/

Abbildung 23: Straßen Assets, seamless Textures

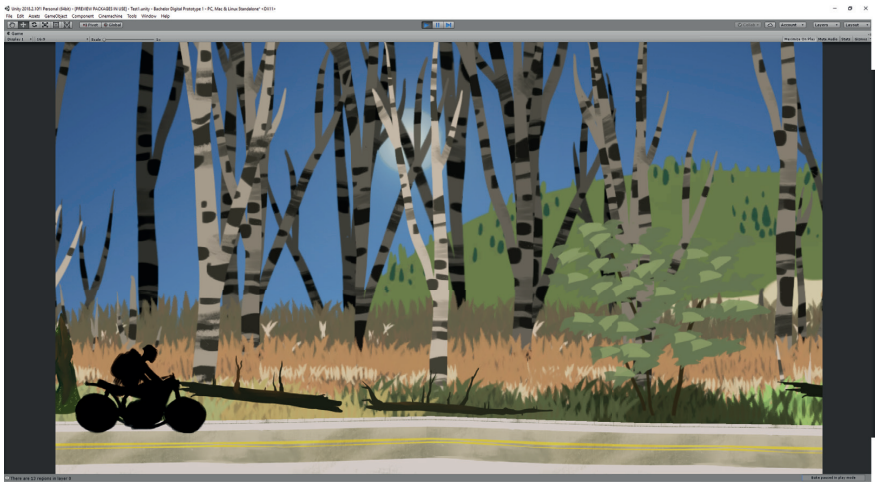


Abbildung 24: Zwischenschritt mit altem Straßenasset im Prototypen

3.4 Animation

Ich habe bisher noch nicht so viel Erfahrung mit Animationen in Unity gesammelt. Gerade für 2D Movement oder ähnliches habe ich bisher hauptsächlich das Asset DOTween[1] benutzt, da man damit im Code Werte animieren kann. Das ging bei den Post Processing Effekten leider nicht, deswegen musste ich auf das Unity eigene Animationstool zurückgreifen.
Ich habe Animationen erstellt und diese State-Veränderungen an Variablen geknüpft. Zum Beispiel wird der Post Processing Effektwechsel für die Nacht, Dämmerung und Tag durch einen Trigger ausgelöst. Das eignet sich an dieser Stelle, da es ein Kreislauf ist, der immer gleich abläuft.

Dadurch, dass ich damit bisher eher weniger gearbeitet hatte, musste ich viel Recherche betreiben. Außerdem hat mir Yannick sehr dabei geholfen. Es war sehr interessant, sich mit den einzelnen Einstellungsmöglichkeiten auseinander zu setzen. Gerade Animationslayer sind äußerst interessant. In meinem Projekt habe ich die Möglichkeiten bisher nur oberflächlich genutzt. Ich denke es würde sich lohnen, sich auch in Zukunft genauer damit auseinander zu setzen.

Probleme hatte ich vor allem in der Polishing-Phase, in der es darum ging Übergänge sauber zu gestalten. Es sollten also zum Beispiel nicht gleichzeitig zwei Post Processing Effekte angewandt werden, da das gerade auf Android für Performance Probleme sorgte. Da nachts manchmal ein anderer Post Processing Effekt verwendet werden sollte, habe ich mit Layern gearbeitet. Aber ich bin nicht ganz zufrieden mit den Übergängen. Da ist durchaus noch Verbesserungspotential vorhanden.

---

1 http://dotween.demigiant.com/documentation.php

Alles in allem bevorzuge ich das Animieren im Code mit Hilfsmitteln wie DOTween, da ich mehr Kontrolle über alles habe. Gerade der Fakt, dass Unity Animationen löscht, wenn GameObjects umbenannt oder verschoben werden, ist extrem unintuitiv und anstrengend. Außerdem habe ich das Gefühl, gerade ineinander greifende Prozesse besser im Code unter Kontrolle zu haben. Aber das könnte auch daran liegen, dass ich mich nicht genug mit Unity Animationen beschäftigt habe.

Ich denke für zukünftige Projekte würde ich wenn möglich eher DOTween benutzen, aber ich möchte mich trotzdem auch weiterhin mit Unity Animationen auseinander setzen.

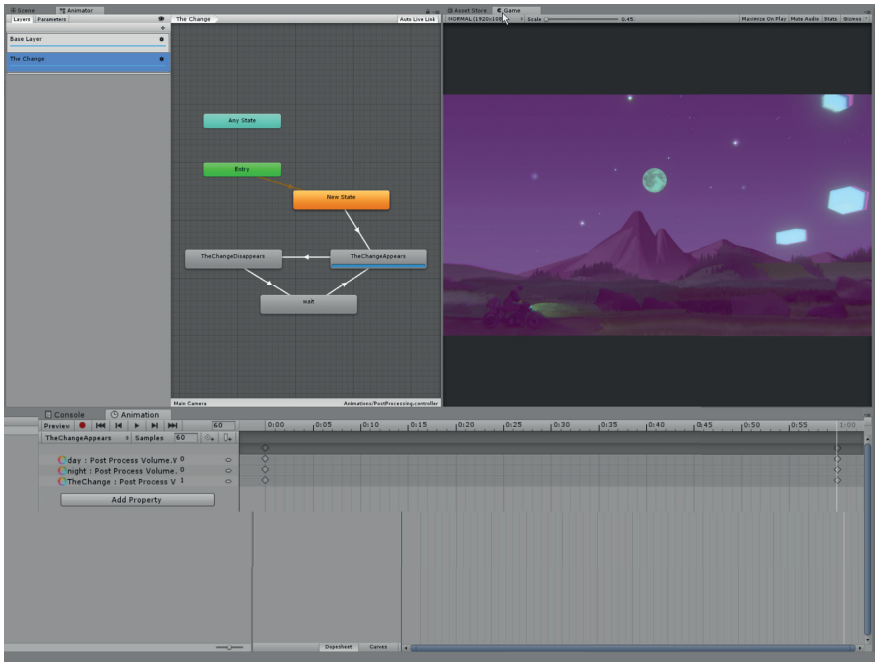


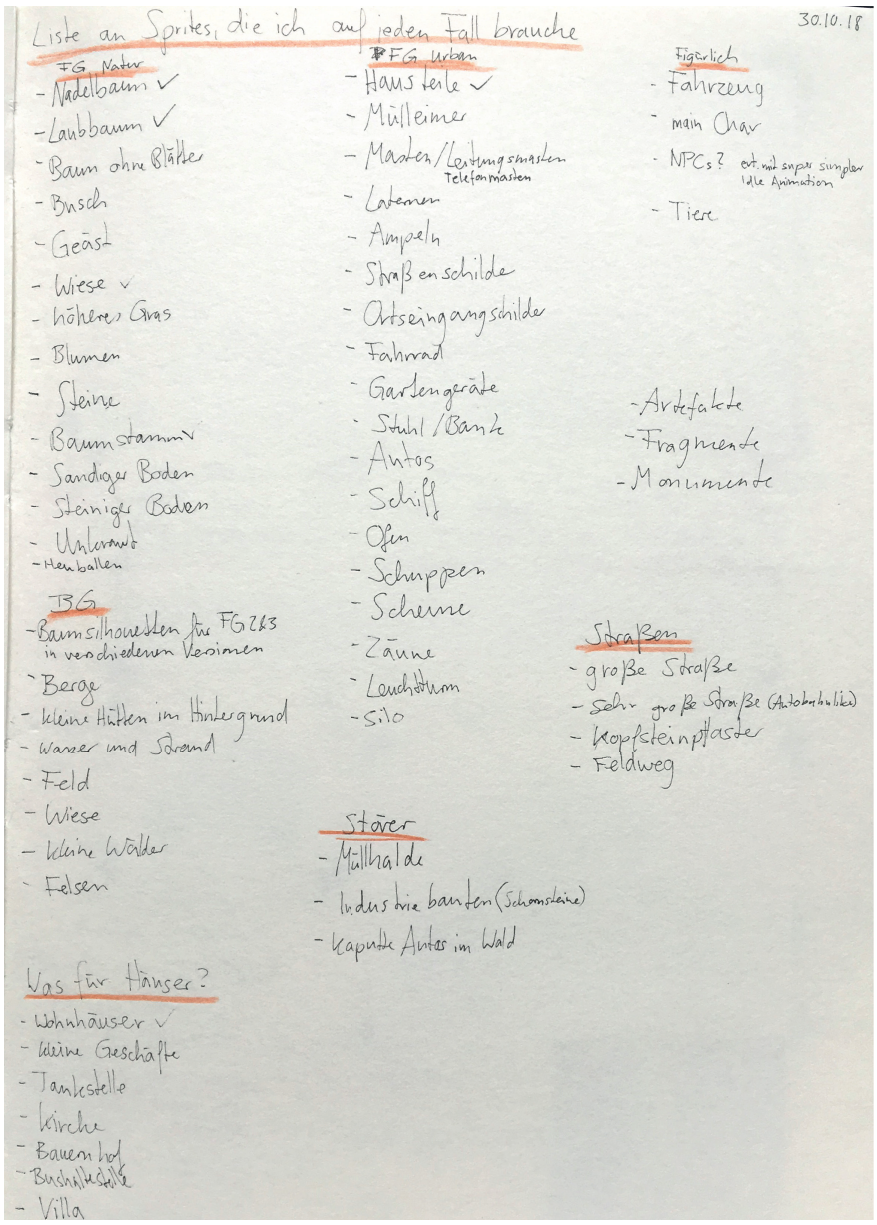
Abbildung 25: Unity Editor, Animator- und Animation-Fenster

3.4 Content Creation

Ab Anfang November hatte ich mich auf ein Setting festgelegt und konnte langsam anfangen, einen Plan für die Content-Creation zu erstellen.
Es war sehr schwer sich vorzustellen, wie das Spiel später aussehen sollte und es war schwer einzuschätzen, ob der gewählte Artstyle passt. Deswegen war es extrem wichtig, dass ich die ganzen prototypischen Block-Grafiken durch richtige Grafiken ausgetauscht habe. Das hat anfangs sehr lange gedauert, weil ich schon viel mit den Proto-Grafiken aufgebaut hatte.

Nachdem ich überall die richtigen Grafiken eingesetzt hatte, konnte ich Ansatzpunkte für Überarbeitung erkennen, sowie bereits funktionierende Stellen identifizieren. Dieses Einbauen der Grafiken war also ein extrem wichtiger Punkt in der Entwicklung. Danach konnte ich den Prototypen viel einfacher erweitern und die Veränderung in der Gesamtheit des Prototypen einschätzen.

Nachdem grundsätzlich überall richtige Asset Grafiken eingebaut waren, habe ich mir eine Assetliste angelegt. So konnte ich auch recht schnell erfassen, wie groß das Projekt wird und wie viele verschiedenen Regionen realistisch sind.
Die Assets auf der Liste habe ich dann nach und nach abgearbeitet, wobei ich nach einer leichten Priorisierung vorgegangen bin.
Das hat grundsätzlich ganz gut funktioniert. Nur bei den Bäumen musste ich ein paar Assets ersetzen, da der Artstyle am Anfang der Production-Phase noch nicht 100%ig feststand. Ich konnte aber auch die Production-Phase nicht beliebig nach hinten verschieben. Das Problem lag aber nicht bei der Assetliste, sondern an meiner geringen Erfahrung mit Environment. Ich denke beim nächsten Mal würde das besser laufen.



Liste an Sprites, die ich auf jeden Fall brauche

30.10.18

FG Natur
- Nadelbaum ✓
- Laubbaum ✓
- Baum ohne Blätter
- Busch
- Geäst
- Wiese ✓
- höheres Gras
- Blumen
- Steine
- Baumstamm ✓
- Sandiger Boden
- Steiniger Boden
- Unterwuchs
- Heuballen

BG
- Baumsilhouetten für FG 2&3 in verschiedenen Versionen
- Berge
- kleine Hütten im Hintergrund
- Wasser und Strand
- Feld
- Wiese
- kleine Wälder
- Felsen

FG Urban
- Hausteile ✓
- Mülleimer
- Masten / Leitungsmasten
  Telefonmasten
- Laternen
- Ampeln
- Straßenschilder
- Ortseingangsschilder
- Fahrrad
- Gartengeräte
- Stuhl / Bank
- Autos
- Schiff
- Ofen
- Schuppen
- Scheune
- Zäune
- Leuchtturm
- Silo

Figürlich
- Fahrzeug
- main Char
- NPCs? evt. mit super simpler Idle Animation
- Tiere

- Artefakte
- Fragmente
- Monumente

Straßen
- große Straße
- sehr große Straße (Autobahnähnlich)
- Kopfsteinpflaster
- Feldweg

Störer
- Müllhalde
- Industriebauten (Schornsteine)
- kaputte Autos im Wald

Was für Häuser?
- Wohnhäuser ✓
- kleine Geschäfte
- Tankstelle
- Kirche
- Bauernhof
- Bushaltestelle
- Villa

Abbildung 26: Assetliste

4 AUSWERTUNG

4.1 Feedback

4.1.1 Feedbacktreffen

Regelmäßiges Feedback hat mir im Verlauf des Projekts sehr geholfen. So war ich oft bei den Sprechstunden meiner Professorin Susanne Brandhorst und es gab regelmäßige Feedbacktreffen mit Kommilitonen. Für diese Termine habe ich meist ein bestimmtes Thema oder mehrere Themengebiete vorbereitet, zu denen ich Kritik wollte. Am Anfang fiel es mir etwas schwer diese Kritik einzuordnen. Da ich oft sehr viel Feedback mit sehr unterschiedlichen Ansichten erhalten habe, musste ich mir überlegen, wie ich damit umgehe.
Später bin ich dazu übergegangen erstmal alles unkommentiert aufzuschreiben und die Leute reden zu lassen. Das hat dazu geführt, dass zwar nicht jeder Kritikpunkt besonders hilfreich war, aber da ich die Leute nicht unterbrochen oder mich erklärt habe, sind teilweise äußerst interessante Ideen entstanden, auf die ich im Rahmen eines engen Frage-Antwort-Korsetts nicht gekommen wäre. Die angesprochenen Punkte habe ich alle ungefiltert aufgeschrieben und nach dem Feedbacktreffen Schritt für Schritt analysiert.

Mein Fokus lag beim Analysieren vor allem auf der Frage: Gehört das zum Umfang meines Projekts und Ist das umsetzbar? Sollten diese Punkte nicht zutreffen, habe ich das Feedback erstmal beiseite gelegt. Das restliche Feedback habe ich spezifisch für jeden Fall einzeln mit dem abgeglichen, was ich verbessern wollte und ob es in meine Vision passt.

Dieser Umgang mit Kritik hat für mich ziemlich gut funktioniert und mir viel geholfen.

Ein Thema, bei dem ich viel Feedback bekommen habe, waren die Anomalien. Ich bin so vorgegangen, dass ich Referenzen und Illustrationen rausgesucht habe, die für mich gut die Stimmung die ich wollte widergaben. Außerdem habe ich Mockups erstellt und prototypisch Assets in Unity integriert.
Das alles konnte ich dann den anderen zeigen und sehen, was für Ideen bei ihnen entstanden. Durch das sehr offene Thema und eher wenig Beschränkung meinerseits, sind äußerst interessante Ideen entstanden.
Im Endeffekt habe ich mich für die Cubes im Himmel entschieden und anhand von Susannes Feedback Blitze ausprobiert. Auch diese Iteration konnte ich dann Kommillitonen zeigen und mich so Schritt für Schritt an das Endergebnis rantasten.

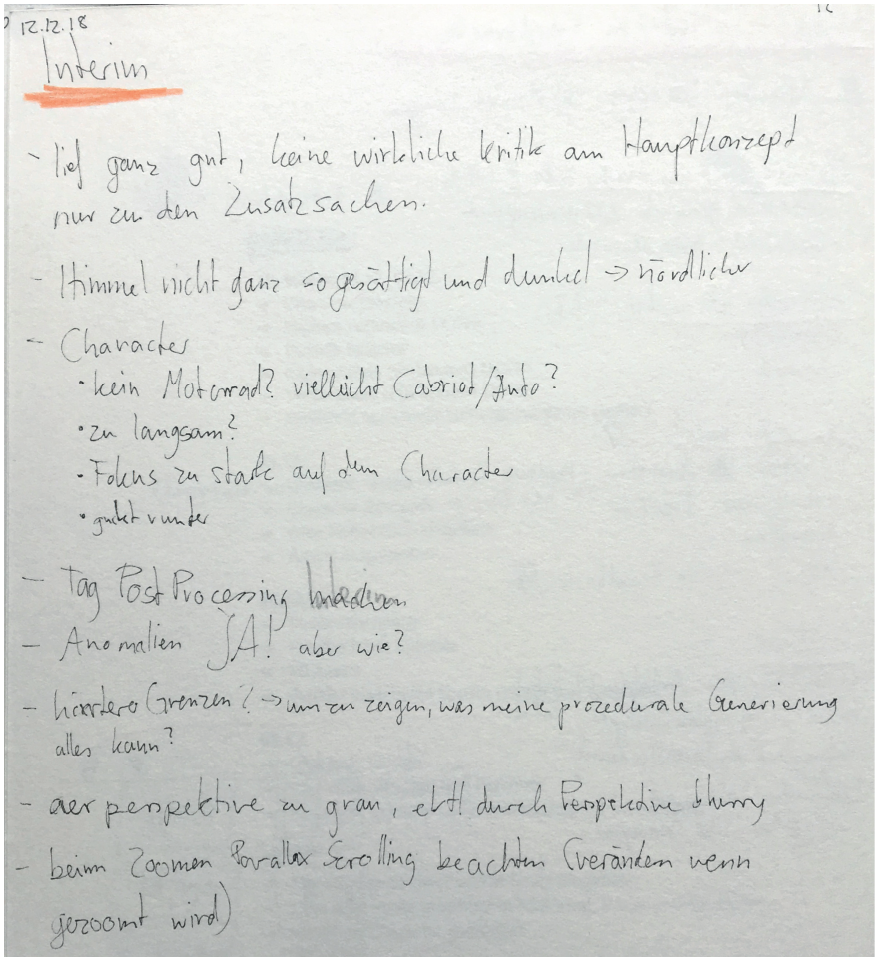
Abbildung 27: Feedback Treffen

4.1.2 Interim

Die Interimpräsentationen fanden am 12.12.2018 von 10-17 Uhr statt. Jeder Bachelorkandidat hatte 5 Minuten Zeit sein Projekt kurz zu präsentieren. Anschließend gab es 10-15 Min Feedback. Es ging darum, die grundsätzliche Idee, den aktuellen Stand und das Vorhaben für den Rest der Projektzeit darzustellen.

Auf Grund der begrenzten Präsentationszeit, habe ich versucht meine Präsention auf das Essentielle zu beschränken. Gestartet habe ich mit einer kurzen Vorstellung meiner Idee. Ich habe meinen Workflow und Arbeitsweise erklärt und den aktuellen Stand dargestellt. Für das weitere Vorgehen, habe ich um Meinungen und Kritik gebeten. Während die Anderen Feedback gegeben haben, lief die Demo im Hintergrund. Das war eine sehr gute Idee, da die Zuschauer direkt auf das eingehen konnten, was sie sahen, statt sich alles merken zu müssen. Außerdem war so für alle ersichtlich, worum es geht.

Abbildung 28: Interim Feedback



12.12.18

Interim

- lief ganz gut, keine wirkliche Kritik am Hauptkonzept nur zu den Zusatzsachen.
- Himmel nicht ganz so gesättigt und dunkel → nördlicher
- Character
  - kein Motorrad? vielleicht Cabriot/Auto?
  - zu langsam?
  - Fokus zu stark auf dem Character
  - guckt runter
- Tag Post Processing machen
- Anomalien JA! aber wie?
- hintere Grenzen? → um zu zeigen, was meine prozedurale Generierung alles kann?
- aer perspektive zu grau, ekt! durch Perspektive blurry
- beim Zoomen Parallax Scrolling beachten (verändern wenn gezoomt wird)

Abbildung 29: Notizen zum Interim Feedback

4.2 Vision Keeping

Ein sehr wichtiger Part bei größeren Projekte ist es, eine genaue Vision davon zu haben, was man am Ende erreichen möchte. Diese Vision muss nicht bereits von Anfang an komplett ausgearbeitet sein, aber es ist sinnvoll es nicht zu lange aufzuschieben. Wenn man weiß, was man möchte, kann man deutlich einfacher planen. Außderdem ist es so einfacher den Zeitplan im Auge zu haben und sich nicht in Details zu verlieren. So schützt man sich davor, im Endeffekt nicht fertig zu werden.

Beispielsweise bestand bei mir das Problem, dass ich immer das Gefühl hatte, noch Gameplay hinzufügen zu müssen. Nachdem ich mir dann ein Visionboard mit klaren Regeln aufgestellt hatte, wusste ich genau, was zu meinem Projekt gehört und was nicht. Das war sehr hilfreich. Aber auch beim allgemeinen Visualisieren des Vorhabens hat mir das Vision Board sehr geholfen.

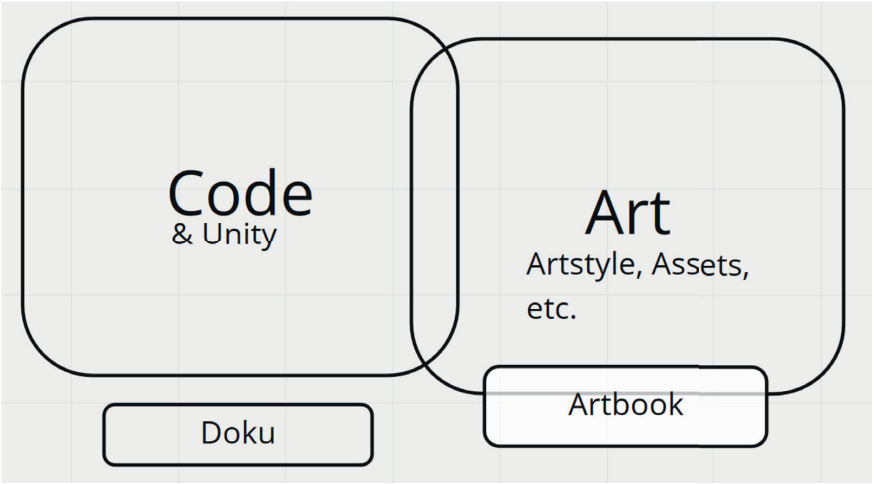


Abbildung 30: Vision Board

4.3 Evaluation des Workflows

Ich denke, im Großen und Ganzen hat mein Workflow ganz gut funktioniert. Ich habe fast alles umsetzen können, was ich mir vorgenommen habe. Das lag u.a. daran, dass ich recht diszipliniert und regelmäßig an dem Projekt gearbeitet habe.
Gerade ein wirkliches Arbeitszimmer in der Uni zu haben, war für mich ein wichtiger Punkt. Dort konnte ich jeden Tag hingegangen und ohne größere Ablenkung arbeiten. Nach der produktiven Zeit konnte ich nachhause gehen und mich ausruhen. So bin ich die meiste Zeit sehr motiviert in die Uni gegangen und das hat sich natürlich auch positiv auf das Projekt ausgewirkt. Ein weiterer Vorteil eines „Büros“ war, dass neben mir noch andere dort gearbeitet haben, ich also nicht allein war. Wenn ich Probleme hatte, nicht weiter kam, oder einfach Meinung und Feedback zu etwas wollte, konnte ich ganz unkompliziert meine Kommillitonen fragen.

Ich musste für viele Bereiche komplett neue Arbeitsabläufe entwickeln. Das betraf vor allem die Gebiete, die im bisherigen Studienverlauf nicht in meinem Fokus standen. Das beste Beispiel sind dabei die Hintergrundassets. Ich habe im zweiten Semester bereits einmal Hintergrundassets erstellt, allerdings gab es eine starke Aufgabenteilung. Im Bachelorprojekt habe ich alles selbstständig erstellt und musste mir dafür einen eigenen Workflow überlegen.
Grundsätzlich bin ich so vorgegangen, dass ich erst verschiedene Skizzen gemacht habe und mir dann die besten rausgesucht und coloriert habe. Die fertigen Assets habe ich auf einem Spritesheet im „Power of 2“ Format angeordnet und in Unity importiert. Musste ich etwas ändern oder ersetzen, habe ich das direkt im Spritesheet gemacht.

Für andere Bereiche hatte ich bereits Workflows. Zum Beispiel beim Erstellen des Artstyles. Allerdings muss man dabei sehr nah am Projekt arbeiten und neue Teil-Arbeitschritte hinzufügen oder entfernen, je nachdem was benötigt wird.
Im Bachelor Projekt habe ich, wie auch in den Projekten zuvor, viel mit Referenzen gearbeitet. Auch Thumbnails haben sich als sehr hilfreich herausgestellt. Sie waren, gerade als Graustufenillustrationen, ein guter Ersatz für Silhoutten. Diese haben hier nicht besonders gut funktioniert.
Die Artstyle-Findungsphase ist recht lang gewesen, aber ich denke, das ist meiner fehlenden Erfahrung im Environmentdesign geschuldet und würde beim nächsten Mal deutlich schneller funktionieren.

Für das Grundgerüst habe ich viel recherchiert und nebenbei ein analoges Lernjournal geführt. Wie in den vergangenen Projekten war das sehr hilfreich, da ich so direkt Gedanken und Ideen festhalten konnte. Ich würde sogar sagen, dass es für mich besser funktioniert hat, als die Projekte davor. Durch die fehlenden Teammitglieder hat das Buch diese Position besetzt und alle Ideen protokolliert.

Auch mein Unity Workflow hat gut funktioniert. Ich war sehr schnell und flexibel, und konnte gut Dinge ausprobieren. Auch die strikten Code- und Namenskonventionen haben sich bewährt.

Bei den nächsten Projekten würde ich eine ähnliche Vorgehensweise wählen. Gerade die Sachen, die ich gelernt habe, bzgl. Artstyle, Environment, Farben, etc. werden beim nächsten Mal noch besser funktioneren.

4.4 Abschließendes Fazit

Im Rahmen des Bachelors und des Designprojekts habe ich einen 2D Endlosscroller mit prozedural generierten Landschaften entwickelt. Ich habe fast alles, was ich mir vorgenommen hatte, umsetzen können. Ich konnte den Arbeitsumfang der einzelnen Bereiche realistisch abschätzen, worauf ich sehr stolz bin. Zwischenzeitlich bin ich ein klein wenig von meiner Vision abgekommen, aber dank des guten Feedbacks u.a. von Susanne, habe ich wieder zurück gefunden und konnte mich dann stärker als zuvor auf mein Projekt konzentrieren.

Das regelmäßige Arbeiten war ein wichtiger Punkt, den ich auf jeden Fall auch in der Zukunft beibehalten werde. Für die nächsten Projekte würde ich früher anfangen eine Quantitätsplanung zu erstellen. Das habe ich in diesem Projekt recht lange hinausgezögert. Vielleicht hätte ich sogar noch mehr schaffen können.

Zusammenfassend bin ich mit dem Ergebnis meiner Arbeit zufrieden. Das Konzept wurde gut umgesetzt und zeigt die Möglichkeiten prozeduraler Generierung. Beide Bereiche, ART und TECH, greifen eng ineinander und erzeugen ein interessantes Ergebnis. Ich freue mich auch sehr, dass bei dem Zuschauer tatsächlich das Gefühl der Neugier aufkommt. Man ist sehr gespannt, was als nächstes kommt und die Demo fühlt sich entspannend an.
Ich freue mich, dass ich mein bisher gesammeltes Wissen erweitern und in diesem Projekt einsetzen konnte.

5 ANHANG

5.1 Abbildungsverzeichnis

Abb. 1 S. 11 | Screenshot von Youtube, https://www.youtube.com/watch?v=Uqk5Zf0tw3o 22.01.19 14:38

Abb. 2 S. 11 | Eigene Darstellung

Abb. 3 S. 12 | Eigene Darstellung

Abb. 4 S. 13 | Eigene Darstellung

Abb. 5.1 S. 15 | aus dem Spiel Oxenfree, http://nightschoolstudio.com/wp-content/uploads/2015/03/oxenfree_screen_6.jpg 12.01. 14:39

Abb. 5.2 aus dem Spiel The Banner Saga, https://www.ingame.de/files/2014/02/test-the-banner-saga-screenshot-1.jpg, 12.01. 14:49

Abb. 5.3 aus dem Spiel Kingdom: Classic, https://mir-s3-cdn-cf.behance.net/project_modules/fs/a9663d42257281.57cd7878eff4c.png 12.01. 14:54

Abb. 5.4 Künstler: Simon Stålenhag, https://dangerousminds.net/content/uploads/images/simon7lkand-kfsa.jpg 12.01. 14:51

Abb. 5.5 Künstler: James Gilleard, https://1.bp.blogspot.com/-vFq6HqUDGxk/WKJ2bnJBvtI/AAAAAAAA-J4M/F07OYOaiugw6iMFiuBciPi4VL0ki8X9oACLcB/s640/Road.jpg 12.01. 14:35

Abb.5.6 Künstlerin: Kim Smith, https://66.media.tumblr.com/512f4766233870761d62e2be9c0e1f9a/tumblr_npoybkhDWp1qhttpto3_640.jpg 12.01. 14:55

Abb. 5.7 Künstlerin: Kim Smith, https://i.pinimg.com/originals/bb/89/e0/bb89e0d0383512e810bf-56d35eda9c05.png 12.01. 14:57

Abb. 5.8 Künstler: Bastien Grivet, https://66.media.tumblr.com/b7b7866fa1df0c2623db73625674a6f3/tumblr_o9cick8waK1roapxto1_500.jpg 12.01. 15:04

Abb. 5.9 Künstler: Brian Edward Miller, https://static1.squarespace.com/static/520542b0e4b0cc-1763c00169/t/53dc00e2e4b02fd0a5edfd7a/1406927079037/?format=1500w 12.01. 15:02

Abb. 5.10 Künstler: unbekannt, https://i.pinimg.com/originals/4b/81/cd/4b81cd7b5ddd8bf6d71b4f-d971aa8959.png 12.01. 15:07

Abb. 5.11 Künstler: Eyvind Earle, https://i.pinimg.com/originals/95/b1/c0/95b1c06741dfcdbe6a38afd-6f70b3183.jpg 12.01. 14:37

Abb. 5.12 Künstler: Benjamin Plouffe, https://66.media.tumblr.com/06921212f969565b1ec6b-847fe63ef70/tumblr_mnogatMMea1qhttpto1_1280.jpg 12.01. 15:03

Abb. 5.13 Künstlerin: Marie-Laure Cruschi, http://mobeltrends.com/wp-content/uploads/2018/05/778bf-7b89403b33d3504ee49d87294a1.jpeg 12.01.19 14:34

Abb. 5.14 Künstler: Marcin Jakubowski, https://scontent-ber1-1.xx.fbcdn.net/v/t31.0-8/12314316_504749173024921_1257306719593011406_o.jpg?_nc_cat=108&_nc_ht=scont-ent-ber1-1.xx&oh=68c2315ae55b5eaaa5340d6020a06599&oe=5CC3A07A 12.01. 15:05

Abb. 6 S. 15 | Screenshots von Google Street View, https://www.instantstreetview.com/

Abb. 7 S. 17 | Eigene Darstellung

Abb. 8 S. 17 | Eigene Darstellung

Abb. 9 S. 17 | Eigene Darstellung

Abb. 10 S. 19 | Künstler: Benjamin Plouffe, https://66.media.tumblr.com/06921212f969565b1ec6b-847fe63ef70/tumblr_mnogatMMea1qhttpto1_1280.jpg 12.01. 15:03
Abb. 11 S. 19 | Eigene Darstellung
Abb. 12 S. 21 | Eigene Darstellung
Abb. 13 S. 23 | Eigene Darstellung
Abb. 14 S. 23 | Screenshot von Google Street View, https://www.instantstreetview.com/
Abb. 15 S. 25 | Foto mit Screenshot eines Pinterestboards, https://bit.ly/2S7mGlc
Abb. 16 S. 25 | Foto mit Screenshot eines Pinterestboards, https://bit.ly/2S7mGlc
Abb. 17 S. 27 | Eigene Darstellung
Abb. 18 S. 29 | Eigene Darstellung
Abb. 19 S. 30 | Eigene Darstellung
Abb. 20 S. 30 | Eigene Darstellung
Abb. 21 S. 32 | Eigene Darstellung
Abb. 22 S. 35 | Eigene Darstellung
Abb. 23 S. 37 | Eigene Darstellung
Abb. 24 S. 37 | Eigene Darstellung
Abb. 25 S. 39 | EIgene Darstellung
Abb. 26 S. 41 | Eigene Darstellung
Abb. 27 S. 43 | Eigene Darstellung
Abb. 28 S. 44 | Eigene Darstellung
Abb. 29 S. 45 | Eigene Darstellung
Abb. 30 S. 46 | Eigene Darstellung

5.2 Quellenverzeichnis

Game Maker‘s Toolkit, How (and Why) Spelunky Makes its Own Levels | Game Maker‘s Toolkit: https://www.youtube.com/watch?v=Uqk5Zf0tw3o

Blackthornprod, RANDOM MAP GENERATOR - EASY UNITY TUTORIAL, https://www.youtube.com/watch?v=D4E0gZyNk-k

Unity Blog über Sprite Shapes, https://blogs.unity3d.com/2018/09/20/intro-to-2d-world-building-with-sprite-shape/

GDC, Practical Procedural Generation for Everyone , https://www.youtube.com/watch?v=WumyfLEa6bU

Verwendete Software/Assets:

- Unity 2018.2.10
- Clip Studio Paint
- Procreate
- Adobe InDesign CS6
- Pinterest.de
- Realtimeboard.com
- DOTween, http://dotween.demigiant.com/documentation.php
- Instant Screenshot, https://assetstore.unity.com/packages/tools/instant-screenshot-24122
- Unity Sprite Shapes
- Unity Post Processing Stack
- Font: Amatic SC von Vernon Adams

6 DANKSAGUNG

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei meiner Bachelorarbeit und dem Studium unterstützt haben.
Allen voran bei meinen Eltern und meiner Schwester. Sie waren stets für mich da und haben mich mit Ratschlägen, Ermutigungen und Vertrauen begleitet.

Ein ganz besonderer Dank geht auch an meine Professoren Susanne Brandhorst und Thomas Bremer. Ich bin sehr dankbar für die Unterstützung und das Feedback, aber auch für die Möglichkeit hier studiert haben zu dürfen.

Danke auch an meine Freunde und Kommillitonen, die mir stets mit guten Tipps und Hinweisen beseite standen.
Danke an Anja, dass du immer Zeit für mich hast und mich in allem unterstützt, was ich mir vornehme.
Danke an Eva, für die Gespräche und die tolle Zeit.
Danke an Lara, für die Freundschaft seit dem ersten Semester.
Danke an Noah, für das Inlineskaten und Filmeschauen.
Danke an Yannick, Huy, Till und Friedemann, für die Tipps, das Feedback und die motivierende Arbeitsatmosphäre.

7 ERKLÄRUNG DER AUTORENSCHAFT

Hiermit erkläre ich, Marie Günther, dass ich zur Erstellung dieser Arbeit nur die von mir gekennzeichneten Quellen, Texte, Bildverweise und Hilfsmittel, sowie von mir selbst erstellten Inhalte verwendet habe.
Verwendete Inhalte von fremden Quellen habe ich im Text gekennzeichnet.

Marie Günther

Berlin den