Arbeitsprotokoll

Autoren: Mher Mnatsakanyan, Nils Rommel, Anton Scheibner

Unity

Die größte Herausforderung für uns alle war es zu lernen, wie man mit Unity arbeitet. Wir können behaupten, dass wir viel aus dem Projekt gelernt haben und wenn wir das Modul nochmal wiederholen würden, dann könnten wir vieles deutlich einfacher und schneller umsetzen.

Trotzdem hatten wir einige Probleme, die uns die Arbeit erschwert haben. Ein Problem war die Performance der Engine, die für lange Wartezeiten sorgte. Ein weiteres Problem war die komplexe Dokumentation. Zwar gibt es Online viele Foren und Tutorials, aber es war recht schwer sich zurecht zu finden, zumal die Engine sehr komplex ist und viele Funktionen bietet. So war es nicht immer einfach die passende Lösung aus vielen verschiedenen Möglichkeiten zu finden.

Preprocessing - Mher Mnatsakanyan/Anton Scheibner

Die Idee hinter dem Preprocesisng war es die Daten aus der CSV Datei in eine einfach zu handhabende Datenstruktur zu überführen.

Grundsätzlich war die Überlegung eine Art Heap für die Daten zu erstellen, in dem die Daten in einer Baumstruktur abgelegt werden. Die Baumstruktur sollte dabei so aufgebaut sein, dass die Daten in der hierarchischen Struktur der Repräsentation der Stadt entsprechen. D.h. zuerst haben wir uns überlegt, wie die Stadt und die Steuerung grundsätzlich aufgebaut sind und wie wir diese Struktur in einer Baumstruktur abbilden können. So haben wir die Daten in folgender Reihenfolge strukturiert:

Jahr -> Stadt -> Gebäudeklasse -> Facilities -> Gebäude

Im Prinzip hat jedes Jahr mehrere Städte, jede Stadt mehrere Gebäudeklassen, jede Gebäudeklasse mehrere Facilities und jede Facility mehrere Gebäude.
Mit dieser Überlegung haben wir das Preprocessing implementiert, wobei wir Zeile für Zeile durch die CSV Datei iteriert haben und die Daten in die entsprechende Struktur abgelegt haben. Die Überlegung da war falls aus der ersten Hierarchieebene ein Wert fehlt, wird die Zeile als komplett neues Element angelegt. Falls es bereits ein Element aus dieser Hierarchieebene gibt, so wird der Wert der nächsten Hierarchieebene sich angeschaut und die Schritte werden wiederholt. So garantiert man, dass am Ende die Daten in der richtigen Struktur abgelegt sind.

Einige Probleme bei der Entwicklung waren:

- die Schleife musste zum Teil mehrfach durchlaufen werden, da wir Durschnittswerte für gewisse Kategorien berechnen mussten
- es gab in der Entwicklungsphase neue Erweiterungswünsche, weshalb die Struktur des Baumes mehrmals angepasst werden musste
- wir haben recht simpel versucht die Liste zu sortieren, aber aus einem uns unerklärlichen Grund hat dann die Stadtgenerierung nicht funktioniert

Kamerasteuerung - Mher Mnatsakanyan/Anton Scheibner

In der ersten überlegung wollten wir eine einfache steuerung haben, wobei man als User nicht viel kontrolle hat, sondern die Kamera eine Stadt fokussiert und man nur durch das wechseln von Stadt zu Stadt die Kamera bewegen kann. Das hat sich aber als nicht so gut herausgestellt, da wir in der Prototypphase der Stadtgeneration festgestellt haben, dass die Städte nicht immer gleich groß sind und manche Städte sehr klein sind, andere wiederum sehr groß. Das hat dazu geführt, dass in großen Städten die Gebäude winzig waren, um was zu erkennen. Aus diesem Grund haben wir uns entschieden eine freie Kamerasteuerung zu implementieren. Zwar blieb das Prinzip der Stadtfokussierung erhalten, aber innerhalb einer Stadt kann man nun mit den Pfeiltasten die Kamera bewegen und mit Shift und Strg den Zoom steuern.

Probleme bei der Umsetzung waren:

- die Städte wurde in einem Schachbrettmuster generiert, was dazu führte, was dazu führte, dass die Zentrierung der Kamera durch die unterschiedlichen Größen der Städte nicht so einfach wie in der Theorie war
- die Kamera sollte sich flüssig bewegen, was aber durch die Performance der Engine erschwert wurde
- eine flüssige Bewegung, d.h. auch eine gute Wahl der Geschwindigkeit der Kamera, war schwer zu finden, vor allem abhängig vom Zoomlevel

Stadtgenerierung - Nils Rommel

Das erklärte Ziel der Stadtgenerierung war es, die Daten möglichst einfach und für den Nutzer auf einem Blick erkennbar zu visualisieren. Jedes Stadtobjekt hat dabei eine Methode create_city, welche eine solche Stadt erstellen soll. Jede Stadt besteht meist aus mehreren Distrikten. Jeder Distrikt besteht aus Häusern, welche wiederum in drei Größenklassen eingeteilt sind. Die Größenklasse beschreibt in dem Generierungsmodell die Grundfläche des Hauses. Es wird angenommen, dass die Grundfläche jedes Hauses ein Quadrat darstellt. Daher wird die Wurzel der Fläche eines Hauses als Ausgangspunkt für die Breite und Länge jenes verwendet. Dabei wurden zum Teil Vorfaktoren für eine bessere Skalierung verwendet. Diese haben keinen Einfluss auf die ursprüngliche Beziehung der Datenpunkte, da die Relation z.B. der Flächen zwischen diesem erhalten bleibt. Anschließend werden die Häuser auf einem Grid in der Unity-Umgebung platziert. Dabei ist es wichtig, eine möglichst enge Packungsdichte der generierten Häuser auf dem Grid zu erhalten, denn diese Fläche aller Häuser eines Distriktes platziert auf dem Grid, gibt die Größe des Distriktes wieder. Damit nun die Vergleichbarkeit zwischen Distrikten erhalten bleibt, müssen die Häuser möglichst eng gepackt sein, sodass es kaum „Leerraum“ gibt. Außerdem sollte die Größe der Häuser aus verschieden Distrikten auf dieselbe Weise berechnet werden. Dies konnte mit dem obigen Ansatz realisiert werden. Ursprünglich war die Idee 3D Modelle für die verschiedenen Haustypen der unterschiedlichen Distrikte zu verwenden. Dabei sollte es für die drei unterschiedlichen Häusergrößen unterschiedlich große Modell geben. Außerdem sollte es für jeden Distrikt-Typ ein anderes Modell geben. Dieser Ansatz kann durchaus als Ausblick dieses Projektes angesehen werden. Dabei sollte drauf geachtet werden, dass die Modelle in Ihrer Größe veränderbar sind, um die Vergleichbarkeit von Hausgrößen zu wahren.

Die erstellten Distrikte werden im Anschluss zusammengeführt. Dabei

wurde zunächst ein Ansatz verfolgt, welcher die Generierung einer möglich natürlichen Stadt als Ziel hatte. Dabei wurde ein Distrikt in der Mitte der Stadt platziert. Anschließend sollten die weiteren Distrikte im Kreis um diesen angeordnet werden. Auf diese Weise entsteht ein natürlich wirkendes Layout für die Stadt und die verschiedenen Größen der Distrikte, welche aufgrund der dynamischen Berechnung der Grundflächen dieser entsteht, spielen im Hinblick auf mögliche Kollisionen keine Rolle. Das Problem mit dieser Methode liegt wiederum in der Vergleichbarkeit der Daten. Zwei Städte lassen sich nur schwierig mit einem schnellen Blick in Bezug auf z.B. Distriktgrößen oder Gebäudeanzahlen in diesen vergleichen, wenn diese nicht in zwei unterschiedlichen Städten an denselben Stellen sind. Ein Beispiel: In Stadt A gibt es keinen Food Distrikt. Stadt B verfügt über alle 9 Bezirkstypen. Wenn der Food Distrikt als erstes erzeugt werden würde, läge dieser in der Mitte. Für die Stadt B ist dies der Fall. In Stadt A gibt es allerdings einen solchen Bezirk nicht, weshalb der nächste in der Reinfolge in der Mitte platziert wird. Eine schnelle Vergleichbarkeit der Städte ist in dem Fall nicht mehr möglich. Daher wurde eine simplere Methode verwendet, um die Distrikte zu einer Stadt zusammenzufügen gewählt. Dabei werden die Distrikte einfach von links nach rechts in einer Reihe platziert. Um auch hier für eine bessere visuelle Vergleichbarkeit zu sorgen, werden die Distrikte mit einer ihrem Typ zugeordneten Farbe markiert.

Auf die oben beschrieben Weisen werden also die Daten: Distriktgröße, Hausgröße und Anzahl Häuser über die verschieden Distrikttypen und Städte vergleichbar dargestellt. Der Datensatz lieferte zusätzlich zu jedem Haus seine EUI (Energy Use Intensity). Dieser Wert gibt an, wie viel ein Haus in einem Jahr im Hinblick auf Größe, Bauart, Alter usw. an Energie verbraucht. Das Ziel des Projektes war es, diese EUI in Abhängigkeit jener genannten Parameter darzustellen, um Muster visuell zu erkennen (z.B. war die Idee, dass alte Industriegebäude wahrscheinlich einen schlechteren EUI- Wert aufweisen). Daher sollte die EUI unabhängig von den anderen Parametern darstellbar werden. Als Visualisierungstechnik wurden die Balkendiagramme ausgewählt. Diese werden über den Häusern generiert und stellen entsprechend Ihrer Höhe und Farbe den relativen EUI-Wert, bezogen auf dem maximalen EUI-Wert im Datensatz dar. Anschließen kann über die Heatmap-Visualisierungstechnik das Baujahr oder das Energy Rating dargestellt werden. Dazu werden die entsprechenden Häuser eingefärbt. Gibt es keine Daten zu einem Gebäude, bleibt es grau.

User Interface - Mher Mnatsakanyan/Anton Scheibner

Das User Interface sollte möglichst einfach und intuitiv sein, damit sich der Nutzer schnell zu recht finden kann. Dazu teilt sich das User Interface im Wesentlichen in vier Bereiche auf, die Schaltfläche um von einer Stadt zur Nächsten zu wechseln, die Schaltfläche um zwischen den verschiedenen Jahren zu wechseln, die Buttons um verschiedene aufbereitete Daten über die generierte Stadt zu legen und der Bereich mit der Legende, welche den einzelnen farblich gekennzeichneten Distrikten der Stadt die jeweilige Funktion zu weist.

Der Bereich zum Wechseln zwischen den einzelnen Städten, der entsprechenden Jahre, befindet sich am unteren Bildschirmrand. Durch das Drücken der entsprechenden Pfeiltaste nach rechts wechselt die Ansicht zur nächsten Stadt des jeweiligen Jahres. Erreicht man durch durchklicken die letzte Stadt des Jahres kommt man durch ein weiteres Mal klicken NICHT zur ersten Stadt zurück. Um wieder zur vorherigen Stadt zu kommen, kann man die Pfeiltaste nach links nutzen, auch hier gilt, dass man bei der ersten Stadt nicht zur letzten Stadt kommt, wenn man die Pfeiltaste nach links anklickt. Zur besseren Orientierung wird der Name der aktuell dargestellten Stadt zwischen den beiden Pfeiltasten zur Navigation zwischen den Städten angezeigt.
Am oberen Bildschirmrand in der Mitte befindet sich der Bereich um zwischen den einzelnen Jahren hin und her zu wechseln. Hier gelten die gleichen Mechaniken wie beim Wechseln der Städte, man kann nicht zwischen der letzten und ersten Stadt hin und her wechseln. Das Jahr in welchem man sich gerade befindet wird, ähnlich wie der Name der Stadt, zwischen den beiden Pfeilen zum Wechseln des Jahres angezeigt. Beim wechseln des Jahres bleibt man bei der zuvor ausgewählten Stadt, welche sich höchstens in ihrer Form ändert in Folge von strukturellen Veränderungen der Stadtzusammensetzung.

Am linken Bildrand oben befindet sich das Auswahlmenu für die Daten, welche über die Stadt gelegt werden können und ein Button um die Belegung wieder zu löschen. Zur Auswahl der Daten gehören Energiebewertung (Energy Rating), Baujahr (Year Build), EUI (Energy Use Intensity). Der Nutzer kann sich damit Baujahr/Energiebewertung vergleichend mit dem EUI-Wert anschauen. Mit dem Reset-Button kann der User das Mapping wieder löschen und hat die ursprüngliche Stadt wieder.

Am rechten Rand oben findet der Nutzer eine Legende zur Erklärung

der farblich markierten Distrikte. Jeder Distrikt enthält lediglich solche Gebäude, die in die Kategorie der jeweiligen Farbe fallen. Insgesamt gibt es neun verschieden Arten von Distrikten, wie zum Beispiel: Essen (Food), Lagerhaus (Warehouse), Bildung (Education), Gesundheit (Health Care). Die Legende zeigt zu jeder Zeit alle Distrikte und die zugehörige Farbe, auch wenn einzelne Distrikte in der angezeigten Stadt gar nicht vorhanden sind. Das dient der Übersichtlichkeit und hilft dem Nutzer sofort zu erkennen, welche Gebäudetypen nicht in der Stadt in dem Jahr zu finden waren.

Organisation - Mher Mnatsakanyan/Anton Scheibner

Wir haben uns als Gruppe über Discord und Github organisiert. Wobei Discord zur Kommunikation und Teilen von Links diente und Github uns das gleichzeitige Arbeiten am Projekt ermöglichte. Anfänglich waren Wir zu viert, das vierte Mitglied sprang aber recht früh ab, weil er sich inhaltlich etwas anderes von dem Modul erhofft hatte. Da dies recht zeitig passierte, entstand keine große Lücke in der Organisation.

Probleme gab es vereinzelt bei Discord, weil Discord vereinzelt keine Push-Benarichtigungen verschickte, wenn neue Nachrichten im Chat waren und somit sich die Kommunikation manchmal etwas in die länge zog. Außerdem kam es anfänglich ab und zu zu langen Wartezeiten beim Pushen und Pullen des Github-Repositorys, wenn benötigte Libraries in das Projekt eingefügt worden sind.