Hochschule Furtwangen

Studiengang Medieninformatik

Sommersemester 2019

Datenverarbeitung in der Medienproduktion

Prozedurale Verteilung von Objekten zur Generierung von Landschaften

Projektgruppe: Nikola Hack, Robin Schwab, Tomislav Sever

Matrikelnummern: 256668, 256325, 256150

Eingereicht bei: Rainer Duda

Abgabetermin: 01.07.2019

Inhaltsverzeichnis

1. Grundlagen

Durch die immer weiter steigenden Anforderungen der VFX Industrie wird es immer wichtiger möglichst viele Prozesse und Arbeitsschritte zu automatisieren. Prozedurale Systeme können dabei viele Aufgaben übernehmen, die ansonsten durch lange, repetitive Arbeitsschritte fertiggestellt werden müssten.

Die Pipeline der VFX-Industrie bietet eine ganze Zoologie an Programmen, die für die verschiedensten Arten von Aufgaben benötigt werden. Damit es möglich ist Informationen zwischen den verschiedenen Programmen umherzuschieben und zu bearbeiten, kann man in den Meisten dieser Programme eigene Skripte schreiben.

So können beispielsweise eigene Werkzeuge zur Datenverarbeitung und -Bearbeitung geschrieben werden, Systeme zum Importieren/Exportieren von Daten erstellt werden oder sogar Systeme zur vollautomatischen, zufallsbasierten Erstellung von Objekten oder ähnlichem (prozedurale Generierung) geschrieben werden. Werden solche prozeduralen Systeme in Echtzeit eingesetzt (z.B. in Videospielen), bieten sie enorme Vorteile wie kleinere Dateigrößen, mehr Content und Variation durch Zufallsbasierte Operationen.

Systeme zur Prozeduralen Generierung folgen meist einem komplexen System von Regeln, die vom Nutzer vorher eingestellt werden können; Es gibt auch prozedurale Systeme, die hauptsächlich auf Zufallsgenerierung basieren. Um ein solches System zu erstellen werden zunächst die einzelnen Arbeitsschritte programmatisch beschrieben. Danach wird das menschliche Wissen über den zu generierenden Sachverhalt abstrahiert

und daraus Regeln eruiert, die den Sachverhalt möglichst allgemein beschreiben und gleichzeitig Schnittstellen bieten, um den Prozess benutzerdefiniert beeinflussen zu können. Die Ziele der Prozeduralen Generierung können je nach Situation anders sein. Man kann beispielsweise Texte generieren lassen, Layouts von Leveln in Videospielen können ebenfalls automatisch erstellt worden sein und in der 3D-Modellierung von Szenen können ganze Landschaften automatisch generiert sein. Die Methode der prozeduralen Generierung eignet sich hervorragend um sehr schnell Prototypen erstellen zu können, die anschließend manuell vom Nutzer detaillierter bearbeitet werden können.

Bei der prozeduralen Generierung werden heutzutage immer öfter KI-Systeme verwendet. Der Vorteil ist, dass für sehr komplexe Sachverhalte nicht zuerst viele Regeln aufgestellt werden müssen, sondern diese Muster/Regeln durch die KI gelernt werden können. Ein Beispiel für solche KI-Systeme sind General Adversarial Networks (GAN's), die beim training selbst überprüfen können, wie gut sie etwas generiert haben, um so schneller zu guten Ergebnissen zu kommen. Sie werden beispielsweise verwendet, um menschliche Gesichter zu erzeugen.

2. Projektidee

Ziel des Projekts ist es, einen Landschaftspinsel bereitzustellen, welcher prozedural verschiedene Biome erzeugt, darunter Wüsten, Wälder und Schneelandschaften. Der Nutzer soll die Möglichkeit haben, bestimmte Bereiche seines Mesh‘ anzumalen; je nach Farbe wird ein anderes Biom inklusive passender Objekte sowie einer passenden Textur erzeugt. Besonderer Fokus liegt auf der Verteilung der Objekte. Der Landschaftspinsel soll als Blender Add-On inklusive graphischer Benutzeroberfläche bereitgestellt werden.

3. Architektur

Zum Verteilen der Objekte benutzen wir das Partikelsystem von Blender. Dieses generiert je nach Einstellungen eine unterschiedliche Menge an Haarpartikeln, welche

wir als Ursprung für die Objekte benutzen. Weil das Partikelsystem nur auf ganze Objekte angewendet werden kann, wird das Gebiet, auf welches die Objekte verteilt werden nicht durch einen Pinsel definieren, stattdessen werden nur ganze Flächen als Grundlage für ein Biom verwendet. Diese Fläche muss vom Benutzer selektiert werden und wird dazu vom restlichen Mesh getrennt, anschließend wird dem neu entstandenen Objekt eine Textur hinzugefügt. Dazu benutzen wir eine zum Biom passende Textur sowie die Standard-Werkzeuge von Blender für Materialien.

Daraufhin wird dem getrennten Mesh ein Partikelsystem hinzugefügt. Von diesem Partikelsystem werden die Raumkoordinaten der Partikel benötigt, an welche dann Platzhalterobjekte (in Blender nennt man sie "Empties") gesetzt werden. An der Position dieser Empties werden danach Objekte aus einer zuvor importierten Bibliothek gesetzt.

Schlussendlich werden dem zuvor separierten Objekt die generierten Objekte als Kind-Objekte angehängt und das Objekt wird wieder mit dem Ursprungsobjekt vereint. Außerdem wird die Szene aufgeräumt, zuvor angehängte Objekte sowie das Partikelsystem werden wieder entfernt.

4. Implementation

Zuerst werden die von uns bereitgestellten Objekte in Blender geladen. Wir benutzen dazu für jedes Biom eine andere vorgefertigte Bibliothek, die aus den Objekten sowie einer Bodentextur besteht. Die Bibliotheken folgen alle der gleiche Namenskonvention, sodass nach Selektion des Bioms automatisch die richtigen Objekte ohne weitere Abfragen hinzugefügt werden. Wenn man zum Beispiel ein Waldbiom erzeugt haben möchte, werden alle Objekte aus der Datei „forest_models.blend“ in die Szene geladen. Anschließend wird das selektierte Mesh vom restlichen Objekt getrennt. Dies ist leider nötig, weil das Partikelsystem von Blender nur auf ganze Objekte und nicht auf Teile davon (wie zum Beispiel einzelne Flächen) angewendet werden kann.

Leider gibt das Blender Werkzeuge zur Separation von Mesh nicht das separierte, neu entstandene Objekt zurück, weswegen wir Alternativlösungen in Anspruch nehmen mussten. Vor der Anwendung der Operation werden alle Objekte in der Szene in eine Liste kopiert, welche anschließend verglichen wird mit den Objekten, die sich nach der

Operation in der Szene befinden – das Objekt welches nur in der neuen Liste vorhanden ist muss entsprechend das neue, separierte Objekt sein. Leider ist derlei Vorgehen in Blender oftmals nötig, weil die Operationen alle auf die graphische Nutzeroberfläche auslegt sind, womit eine Selektion einfach durch Anklicken des entsprechenden Objekts erfolgen kann. Im Skript ist das allerdings natürlich nicht möglich.

Anschließend wird dem neu erzeugten Objekt ein Material mit Textur hinzugefügt. Wir haben uns hierbei dafür entschieden, das Node-basierte Materialsystem des Cycle-Renderers zu benutzen. Wir laden hierzu die zum Biom passende Textur und erzeugen ein neues Material mit einer Input-, einer Output- und einer Principled BSDF-Node. Zuletzt selektieren wir jeden einzelnen Vertex des Objekts und verwenden die unwrap-Funktion von Blender. Darauffolgend muss das erzeugte und konfigurierte Material nur noch dem Objekt hinzugefügt werden.

Als nächstes werden dem Objekt die Haarpartikel hinzugefügt. Dazu haben wir Standard-Einstellungen definiert, wobei sich die Werte hauptsächlich durch Tests ergeben haben.

Für die Verteilung der Objekte in der Szene werden nur die Raumkoordinaten der einzelnen Partikel benötigt. Auf die Plätze der Partikel werden nun zunächst Platzhalterobjekte (sogenannte "Empties") gesetzt. Bei der Platzierung eines neuen Empties wird zunächst überprüft, ob es sich weit genug weg von anderen Empties befindet. Die minimale Entfernung kann vom Nutzer eingestellt werden und sorgt letztendlich für die Dichte der verteilten Objekte in der Szene, umso kleiner die minimale Entfernung ist, desto dichter zusammen werden alle Empties gesetzt. Ist die Bedingung der minimalen Distanz nicht erfüllt, wird das gesetzte Empty wieder gelöscht.

Die zu verteilenden Objekte werden noch vor dem Platzieren aus der von uns erstellten Bibliothek geladen. Die Objekte werden einer Liste hinzugefügt, aus welcher dann zufällig für jedes von uns gesetzte Empty ausgewählt wird, welches Objekt platziert werden soll. Sobald dies geschehen ist, beginnen die Cleanup-Operationen, um das Mesh des Bodens wieder in den Ursprungszustand zurückzuführen.

Zuerst werden alle automatisch gesetzten Empties wieder aus der Szene entfernt. Anschließend werden die generierten Objekte einer Gruppe hinzugefügt, damit diese vom Nutzer leichter verwendet oder auch wieder gelöscht werden können. Zuletzt werden die importierten Objekte sowie das Partikelsystem wieder aus der Szene

entfernt, außerdem wird das vorher separierte Objekte wieder mit dem Ursprungsobjekt zusammengefügt.

Um dem Nutzer eine grafische Schnittstelle bieten zu können, wurde das Projekt als Add-On implementiert. Hierzu wird im Header der Python-Datei ein Dictionary definiert, welches einige Metadaten zur Beschreibung des Add-Ons enthält. Darunter befindet sich neben des Projektnamens auch eine Beschreibung, sowie die Kategorie der zu erstellenden Erweiterung. Möglich ist auch eine Versionsangabe, sowohl für die zu nutzende Blender Version, also auch dem Stand der Erweiterung. Ebenfalls wichtig sind Funktionen, die bei der Registrierung und Deregistrierung des Add-Ons ausgeführt werden.

Diese minimale Struktur kann nun mit verschiedenen Klassen und Methoden erweitert werden. Zunächst haben wir die bereits erstellten Skripte in die Datei miteingefügt. Dabei viel auf, dass Klassenmethoden immer mit dem Parameter „self“ aufgerufen werden müssen. Außerdem können verschiedene Methoden innerhalb einer Klasse lediglich mit dem Klassennamensaufruf zusammen ausgeführt werden, sodass das anfängliche Skript umgeschrieben werden musste.

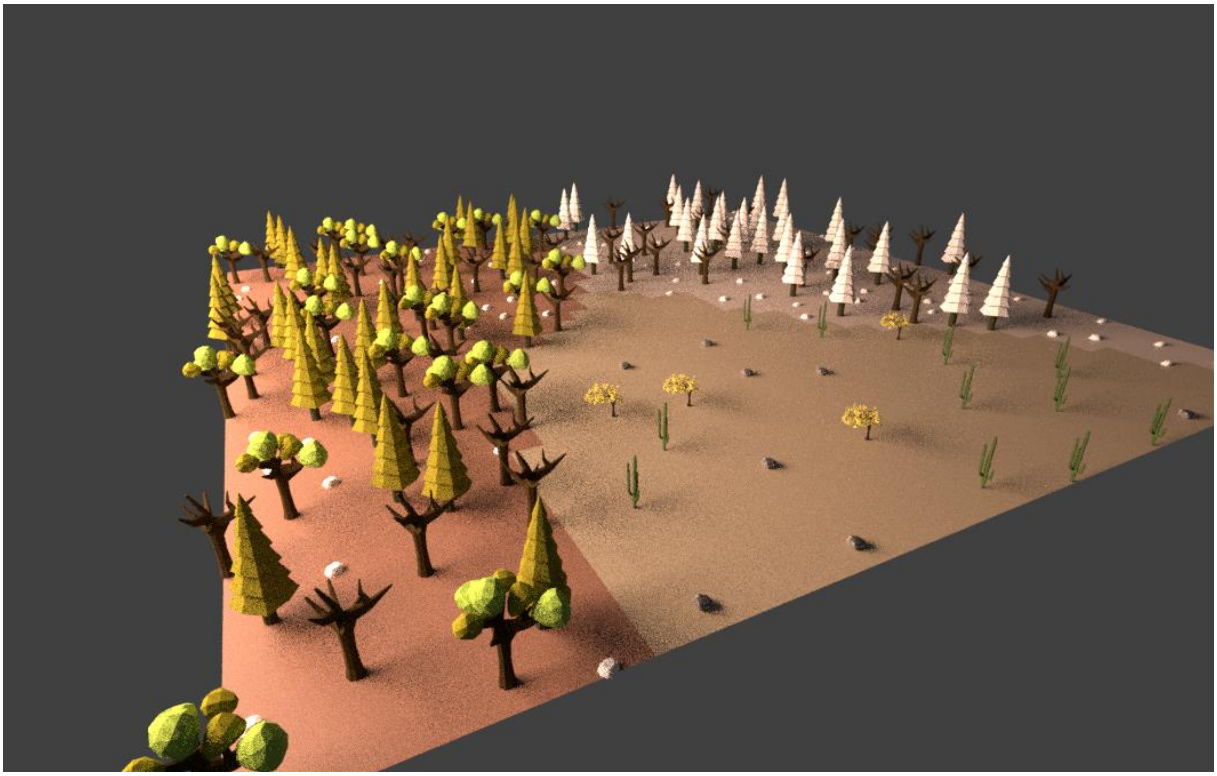
Um die grafische Benutzeroberfläche zu realisieren, wurde von uns eine Klasse „Settings“ angelegt. Hier wurden über sogenannte Properties verschiedene Einstellungsmöglichkeiten geschaffen - so beispielsweise eine Ordnerpfadauswahl um auf das Verzeichnis mit den Models für die einzelnen Biome zu zeigen. Dies geschah über eine String-Property, die als subtype „DIR_PATH“ enthielt, um auf der Oberfläche ein Ordnerauswahldialog öffnen zu können. Des Weiteren haben wir eine Int-Property hinzugefügt, um dem Nutzer die Möglichkeit zu bieten, die minimale Distanz zwischen den später platzierten Objekten zu verändern. Außerdem konnten wir mithilfe einer Enum-Property eine Auswahlmöglichkeit für die Art des Bioms relativ einfach realisieren.

Um die einzelnen Properties auch auf der Oberfläche anzuzeigen, wurde eine weitere Klasse definiert, die ein Panel implementiert. Da in Blender viele Interaktionsmöglichkeiten über solche Panels ablaufen, konnten wir unsere grafische Benutzeroberfläche hier mit integrieren. Mittels einfachen Meta-Daten mit dem Präfix „bl_“ wird der Ort des Panels definiert. Hierbei entschieden wir uns für das

„Tool“ Panel, sowohl im Object, als auch im Edit-Mode. Über eine „draw“-Funktion wird das Panel letztlich realisiert. Hierbei kann man auch Einfluss auf das Layout nehmen und die vorher definierten Properties einbinden. Die eigentliche Skriptfunktion wird über einen sogenannten Operator realisiert, den man in Form eines Buttons mit in das Panel integrieren kann.

Schließlich müssen in den Registrierungsfunktionen am Ende der Python-Datei die verschiedenen Klassen eingebunden werden.

Das Endergebnis sieht folgendermaßen aus:



5. Reflektion

Durch die intensive Projektarbeit konnten wir viele Erfahrungen sammeln und gleichzeitig Inspirationen für zukünftige Objekte gewinnen.

Bisher beschränkte sich unsere Erfahrung mit Blender nur auf den graphischen Modellierungsbereich, weswegen sich zuerst ein Adaptionsprozess vollziehen musste. Wir haben Blender damals schon als Programm mit sehr guter und umfänglicher Funktionalität kennenlernt, welches allerdings wie einige Open Source-Projekte etwas

unter einer nicht einheitlichen Nutzeroberfläche leidet. Dieser Eindruck wurde beim Verwenden der Python-API bestätigt. Alle wichtigen Funktionen sind vorhanden, diese funktionieren auch meistens wie erwartet, allerdings ist mitunter schwierig zu erahnen, was erwartet ist. Es ist zum Beispiel von enormer Wichtigkeit, vor unterschiedlichen Operationen den Kontext anzupassen; allerdings ist es oftmals schwierig herauszufinden, was der richtige Kontext ist. Die Fehlermeldungen sowie Debugmöglichkeiten sind leider häufig etwas spärlich und nur wenig hilfreich. Ähnlich spärlich ist auch die API-Dokumentation; ähnliche Einträge sind oft ohne Verlinkungen an verschiedenen Orten verteilt und manche Informationen sind gar nicht aufzufinden. Außerdem gibt es für die gleiche Funktionalität oft mehrere Funktionen, weil Blender neben den Funktionen für die graphische Nutzeroberfläche auch einige Python-spezifische Funktionen bereitstellt, die etwas performanter ausgeführt werden. Das ist manchmal hilfreich, macht die Gesamtaufgabe aber eher komplexer, vor allem weil manche Funktionen nur mit besonderen Einstellungen oder dem richtigen Kontext funktionieren und andere wiederum andere Voraussetzungen benötigen.

Aus den oben genannten Gründen hat sich das Einarbeiten in die Thematik als sehr zeitintensiv herausgestellt. Es hat sich allerdings auch gezeigt, dass derlei Probleme mit der nötigen Beharrlichkeit immer lösbar sind.

Einige unserer Ideen waren leider nicht so umsetzbar wie anfangs geplant, zum Beispiel die Umsetzung als Pinsel ist leider nur schwerlich möglich, weil sich das Partikelsystem von Blender wie bereits erwähnt nur auf ganze Objekte anwenden lässt.

Die Übergänge zwischen Biomen flüssig zu gestalten stellte sich ebenfalls als zu große Herausforderung heraus. Da wir auf Blenders Partikelsystem zurückgreifen, um die Objekte zu platzieren und jedes Biom einzeln nacheinander generiert wird, haben wir zum Zeitpunkt der Platzierung der Objekte keine weiteren Informationen darüber, welche Biome an das momentan generierte angrenzen, beziehungsweise wie die Biome überlappen.

Nichtsdestotrotz konnten wir einen tieferen Einblick in die 3D-Modellierung gewinnen und mussten Lösungen für nicht triviale Probleme finden. Außerdem haben wir einen Eindruck davon bekommen, welche Funktionen zur Modellierung benötigt werden und wie die Daten intern strukturiert sind.

Mit etwas mehr Zeit wären einige Optimierungen möglich gewesen.

Zum einen könnte man erneut versuchen, einen Pinsel zum Selektieren des Bereichs für ein Biom zu verwenden. Dafür müsste man wohl eine andere Lösung als das Partikelsystem von Blender finden, zum Beispiel indem man einen eigenen Algorithmus implementiert, der Partikel auf einen ausgewählten Bereich verteilt. Alternativ könnte man auch Vertexe an die Ränder des angemalten Bereichs setzen und das Mesh an diesen Stellen trennen. Dabei müsste man das Mesh allerdings stark verändern und wir sind uns unsicher, ob dieser Ansatz zu guten Ergebnissen führen würde.

Außerdem könnte man versuchen die Textur an den Übergängen zu verändern, um wirklich realistische Übergänge zu schaffen.

Die Platzierung der Objekte kann ebenfalls noch besser gelöst werden. Eine Möglichkeit für die Erweiterung des Platzierungsalgorithmus ist die Einführung eines zweiten Durchlaufs. Im ersten Durchlauf werden große Objekte, wie beispielsweise Bäume im Wald-Biom platziert. Im zweiten durchlauf sollen dann zwischen die großen Objekte möglichst viele kleine Objekte, wie Gras oder Steine gesetzt werden, um die Szene detaillierter zu gestalten. Dazu könnte man, wenn die zu Platzierenden Objekte bekannt sind, Platzierungswahrscheinlichkeiten für jedes einzelne Objekt setzen, um noch mehr Kontrolle über die Generierung der Landschaften zu erhalten.

Trotz der Schwierigkeiten sind wir angesichts der Komplexität der Aufgabe stolz auf unsere Ergebnisse. Wir konnten im Laufe des Projekts viele Inspirationen für zukünftige Projekte sammelln und haben uns ein tieferes Verständnis für die Methodiken im Bereich der Visual Effects angeeignet.