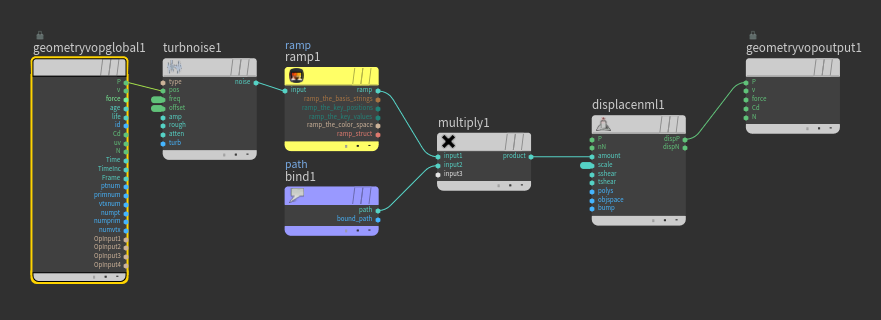
Hallo

1. Um unser Terrain zu erzeugen benötigen wir als aller erstes einmal ein **Feld.** Mit der **Grid-Node** erzeugen wir ein (in diesem Fall 25 x 30) Feld. Die Anzahl der **Zeilen** & **Spalten** werden nach Gemütslage erhöht. Je höher die Anzahl, desto mehr Details (Primitives) existieren im Feld.
2. An unser Grid setzen wir ein **Remesh**, damit wir später noch einmal besser mit den Details im Feld arbeiten können.
3. Nach dem Remesh wird unser komplettes Gitter in einer Farbe eingefärbt. Die Farbe selbst ist prinzipiell egal, wichtig ist in unserem Setup der **Farbwert für Rot**. Wir benutzen hier einen Grauton, der die Basis der Szene stellt.

Wir haben damit unsere Terrain-Basis gelegt. Im folgenden werden noch spezielle Teilgebiete angefertigt.

1. Neben dem Basisterrain-Setup liegen unsere Teilgebiet-Nodes. Wir erschaffen hier jeweils eine **Gruppe** für Grassland und die Berge, die später generiert werden sollen. Dazu werden beide Group-Nodes jeweils in unser erstes Grid eingespeist. Wir wählen die **Base Groups**, die wir gleich weglöschen wollen per Knopfdruck aus und drücken Enter.
2. An unsere Gruppen hängen wir jeweils ein **Boolean** an. Wir setzen die Basis-Gruppe B (also Grassland/Berge) und *subtrahieren A-B*. Am Ende sollte nur unsere gewünschte Fläche übrigbleiben.
3. Diese Flächen werden anschließend wieder in Farben eingefärbt (nochmal: der **Rot-Wert** ist hier wichtig!)
4. Neben Grass, Gebirge und Grundland wollen wir noch einen Pfad in unserer Szene haben. Dazu wählen wir die **Add**-Node aus und beginnen Punkte in der Scene-View zu setzen.
5. Da unsere Punkte und damit unser Pfad für später grob gesetzt sind können wir jetzt die Linie zu einer feineren Kurve formen. Eine häufig genannte Methode ist die **Convert**-Node, bei der unsere Punkte zu einer *NURBS-Kurve* geformt werden. Wir verwenden hier jedoch **Subdivide**. Subdivide gibt uns mehr Punkte entlang unseres Pfades, da unsere Tiefe auf 3 gesetzt ist.
6. Egal wofür man sich entscheidet, daran wieder eine **Color**-Node anhängen und diesmal auf *schwarz* setzen.   
     
   Damit steht jetzt unsere Karte!

Jetzt wo wir die Aufteilungen für später definiert haben wollen wir unsere Teile zusammenfügen bzw. überlappen.

1. Hierzu wählen wir die Node **Attribute Transfer** dreimal hintereinander. Unsere Basis ist erst unser…Basisgrid! Der zweite Input ist dann unser Grassland. Attribtransfer transferiert diesen zweiten Input auf das Basisgrid und erschafft ein neues, verschmolzenes Grid.   
   Selbes wird dann mit den Bergen und dem Pfad jeweils wiederholt.  
   Am Ende sollte ein Feld mit 4 verschiedenen farblich unterlegten Teilgebieten zu sehen sein.
2. Wir fügen an dieser Stelle ein **Transform** ein, um unsere gesamte Szene leicht zu kippen. Das mag an anderer Stelle auch möglich sein, jedoch haben wir hier den Vorteil, dass alles was später an Generation erfolgt von dieser Schieflage ausgeht und auch wirklich nach *oben* zeigt!
3. Nach dem Transform schreiben wir mit **Attribute Wrangle** eine Expression:   
    *f@path = @Cd.r;*Wir erschaffen hier also eine Float-Variable namens „path“, deren Wert mit dem Rot-Wert der einzelnen Teile belegt wird. Ergo hier bei uns alles was den Pfad darstellt bekommt den Wert 0, das Grassland den Wert 0.15, das Gebirge 1 und der Rest 0,3.
4. Hier wird dann anschließend das erste große vollbracht. In der **Attribute VOP**-Node sehen wir (nachdem wir in die Node reingehen) einen Haufen an Einstellungen:  
     
     
   Hier wird in zwei Schritten entschieden wie viel Noise-Effekt auf unser Terrain projiziert wird. Es gibt einmal ein generelles Grundrauschen, dass der obere Pfad festlegt. Das untere **Bind** importiert unsere Path-Variable und multipliziert den jeweiligen Wert mit unserem Grundrauschen. Sprich unser Grassland bekommt ein Grundrauschen \* 0.15, unser Gebirge das volle Grundrauschen und unser Pfad bleibt genauso wie er ist[[1]](#footnote-1).  
     
   Unser Terrain steht jetzt! ***[Screenshot, Screenshots vorher]***
5. Da wir Höhlen in unseren Bergen haben wollen bauen wir uns noch zwei Tunnel. Wir nehmen uns zwei **Tubes**, *schließen die End Caps*  und finden Werte, die uns gefallen. Eine dreieckige Form passt zu unserer Szene später ziemlich gut, also ziehen wir die *Columns auf 3 runter*. Die *Height*  legt die Länge unserer Röhre fest, da wir sie auf der X und Z Achse um 90° gedreht haben liegt diese auch flach auf.  
   *Rows*  können bei der **Bend**-Operation wichtig sein, dazu aber dann später mehr.
6. Unsere Tubes bekommen dann eine Farbe, die wir später haben wollen. Der Farbwert Rot spielt hier übrigens keine Rolle mehr, da wir von dem Displacement auf dem Terrain abgekoppelt sind und unsere beiden Tunnel Geometrien *später* mit dem Rest zusammenfügen.
7. Mit **Polybevel** runden wir unsere Tube leicht ab.
8. **Bend** erlaubt es uns unsere Tubes in gewünschter Richtung um eine festgelegte Achse zu biegen, in unserem Fall um 90°.
9. Mit **Transform** passen wir die Position auf unsere jew. gewünschte Lage an.
10. Anschließend **Remeshen** wir unsere Tubes um für die nächste Node
11. **Mountain** mehr Dreiecke zum Verformen zur Verfügung zu stellen.
12. Da wir beide Röhren mit der Szene in einem Schritt erschaffen wollen, **Mergen** wir diese erst um dann anschließend in einer
13. **Boolean**-Node die beiden Inputs Terrain und Höhlen zu vereinigen, bzw. die Schnittmengen zu entfernen. Wichtig ist hierbei, dass die Eingänge der Höhle auch direkt an der Bergen anliegen!
14. Dieses neue Terrain bekommt noch ein kleines Becken für unseren See, den wir im Schnelldurchlauf folgendermaßen aufbauen:   
    Tube > Transform > Mountain > Remesh
15. **Booleaniere** wieder beide Teile für ein neues Gesamtmesh und die Szene sollte so aussehen:  
      
    Ein Bild, das gegessen enthält.

    Automatisch generierte Beschreibung

Für die nächsten paar Schritte benötigen wir wieder neue **Gruppen**. Zwar haben wir ja bereits ganz am Anfang schon welche definiert, Houdini kann mit diesen allerdings nicht mehr arbeiten, da diese Attribute auf unser Grid wieder und wieder transferiert wurden. Also legen wir einfach neue, etwas spezifischere Gruppen an.

1. Die Gruppen, die für uns relevant sind, sind
   1. Gras
   2. Baum
   3. (Blumen)
   4. Schilf
   5. Steine
   6. Schnee
   7. Pfad

Diese werden alle per Group Select in der Scene View ausgewählt, um später Scatter-Points und Farben drauf setzen zu können.

Ein Bild, das Text, Screenshot, Elektronik enthält.

Automatisch generierte Beschreibung

1. Nach unseren Groups wandern wir hier in zwei Richtungen: einmal links ist unser gesamt Scatter-Point-Setup, welches uns erlaubt Objekte in der Szene einzuspeisen. Rechts färben wir unsere Szene mit den gewünschten *eigentlichen Farben* ein. Da unsere Anfangsfarben soweit aber zur Szene passen sind diese jedoch alle bis auf die Weg Farbe ausgegraut.



Die Farbe des Pfades ändern wir, indem wir bei der Color-Node die Gruppe anwählen.

1. **Attribute Blur** verwischt die Grenzen unseres gewählten Attributes (*Cd* für Color) in der gesamten Szene.
2. Nach dem Aufbau unserer Grund-Terrains und dem Setzen der Farben reduzieren wir noch die Polygon-Anzahl mithilfe von **PolyReduce** auf 50%.
3. **Edgecusp** (bzw. **Facet**) daran gekettet sorgen für schöne Lichteffekte in unserem Low-Poly-Setup, da die Flächen härtere Übergänge bekommen.

Hieran angekettet wird nur noch ein letztes **Merge**, deshalb springen wir noch einmal zurück zur eben genannten Gabelung.

Ein Bild, das Text, Computer, Screenshot enthält.

Automatisch generierte Beschreibung

Dieses Setup ist ziemlich groß, jedoch passiert hier in jedem Block prinzipiell dasselbe.

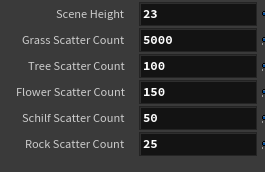
1. Wir fangen mit einer weiteren **Color**-Node an. Diese färbt unsere gesamte Szene in schwarz ein, um in den nächsten Schritten einen möglich hohen Kontrast zu ermöglichen.

Nehmen wir als Beispiel nun das Gras-Setup ganz rechts:

1. Unser erster Knoten ist wieder eine **Color**-Node. Hier wird eine unserer vordefinierten Gruppen ausgewählt (in diesem Fall „grass“) und die Farbe weiß auf eben nur diese Fläche, die von *grass* beansprucht wird, verteilt.
2. Daran angeknüpft schreiben wir in **Attribute Wrangle** eine neue Float-Variable *density. Density* merkt sich wiederum die Rot-Werte aus dem ihr gegebenem Input – sprich alles was weiß markiert ist.
3. In **Scatter** importieren wir unser *density*-Attribut und tun damit unsere Punkte deutlich dichter als gewöhnlich setzen. In unserem Gras-Block haben wir hier 5000 Punkte gesetzt, also wird später 5000-mal je ein Gras-Objekt auf einen Punkt gesetzt.
4. [In der **Attribute VOP** setzen wir zufällige Farben für unser Gras. Hierbei wird jedem einzelnen Punkt ein Wert zwischen 0 und 1 zugewiesen, der dann die Helligkeit bestimmt]
5. **Color**ize färbt dieses Farbattribut Cd dann mit einer von uns definierten Farbrampe ein.
6. **Attribute Randomize** nutzen wir, um diverse Attribute pro Punkt zufällig aufzusetzen. Hier bekommt jeder Punkt einmal ein *pscale*- und ein *orient*-Attribut, festgelegt durchein Minimum und Maximum ausgehend vom Originalobjekt. Bei *Pscale* interessiert uns nur die Höhe, wir gehen also nur in eine Dimension. *Orient* ist für die Rotation verantwortlich, wir benötigen hier also 4 Dimensionen, wobei die nach oben [und unten] auf 0 gesetzt sind.
7. In einem **For-Each-Block** wird dann unser Gras eingespeist und auf jeden einzelnen Punkt (samt vordefinierter Attribute) gesetzt.  
     
   Bei den Blumen und Bäumen kommt hier noch ein **Switch** hinzu, das durch eine Funktion zufällig aus einer Inputmenge pro Punkt das nächste Objekt auswählt.
8. Unsere letzte Node im Terrainsetup ist ein finales **Merge**, dass das Terrain mit den dazu gecasteten Teilobjekten vereint!

**BONUS**

Houdini erlaubt es uns auch Parameter in Abhängigkeiten zu anderen zu setzen. Was wir also beispielsweise machen können, ist eine **Null**-Node aufzusetzen, deren Aufgabe es ist, einige Kernparameter leicht erreichbar anzupassen.



Hier können wir also die Höhe der Szene und die Anzahl der Scatter-Punkte für jedes einzelne Teilobjekt festlegen!

[Das Aufsetzen dieser Funktion erklären?]

**KAMERA-Drohne**

Die Kamerafahrt ist vom Aufbau relativ simpel.

Was wir brauchen sind

a) einen **Pfad**, den die Kamera entlangläuft

b) eine **Null**-Node, die die Position der Kamera relativ zur Zeit trackt

c) eine **Kamera** (duh)

Der Aufbau des Pfads läuft genauso ab wie der innerhalb des eigentlichen Terrains. Erzeuge in einem neuen Objekt *cam\_path* Punkte und schaffe eine Kurve aus diesen. Ursprünglich war gedacht, dass der Kamerafahrt exakt derselbe Pfad ist, bloß ein wenig nach oben versetzt. Wir haben jedoch einen interessanteren Weg für diese Fahrt gefunden und damit dies Punkte nochmal manuell von Hand angepasst. Neu hinzu kommt hier eine *CAM\_SUB\_OUT* benannte **Null**-Node.

Zurück in der Objektansicht sehen wir noch eine weitere **Null**‑Node mit dem Namen *PATH\_SUBDIVIDED*. Diese wird in die **Kamera** eingespeist und sagt dieser, wie die wo langzufahren hat.

Beim Erzeugen dieser Null setzen wir den Haken bei *Enable Constraints* auf ja und sehen ein kleines Feld aufploppen. Ganz rechts haben wir die Möglichkeit ein Constraint direkt zu erschaffen, alternativ machen wir das manuell in der Node selbst. Außerdem wollen wir hier einen Slider für die Kameraposition hinzufügen, diese Null steuert also die Kamera komplett.

 -> 

Klicken wir uns nun in das Constraints-**CHOP Network** rein sehen wir die *getworldspace*-Node. An diese hängen wir ein **Follow Path** an und setzen den *SOP Path* auf unseren Kamerapfad *CAM\_SUB\_OUT* und schreiben unter *Position* die Funktion *ch(“../../pos2“)*, wobei pos2 der vorher hinzugefügte float-Wert „Position“ der Null-Node ist.

In PATH setzen wir dann die Position 0 auf unser Startframe und 1 auf unser Zielframe. Den Rest dazwischen berechnet Houdini selbst.

1. [↑](#footnote-ref-1)