VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KOLEKCE NÁSTROJŮ PRO ZJEDNODUŠENÍ TVORBY HERNÍCH ASSETŮ V BLENDERU

COLLECTIONS OF TOOLS FOR EASY GAME ASSETS CREATION IN BLENDER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MILAN HRABOVSKÝ**
AUTHOR

VEDOUCÍ PRÁCE — **TOMÁŠ CHLUBNA, Ing.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

143506

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Student: **Hrabovský Milan**
Program: Informační technologie
Specializace: Informační technologie

Název: **Kolekce nástrojů pro zjednodušení tvorby herních assetů v Blenderu**
Kategorie: Počítačová grafika
Akademický rok: 2022/23

Zadání:

1. Naučte se základy práce s 3D modelovacím programem Blender (verze 3.2 a výše).
2. Seznamte se se skriptovacím Blender Python API.
3. Nastudujte a vyberte vhodné případy využití Blenderu pro herní vývoj.
4. Navrhněte add-ony, které zjednoduší modelování objektů pro vývoj her.
5. Nastudujte možná a existující řešení dané problematiky.
6. Navrhněte uživatelské rozhraní pro výsledné add-ony.
7. Add-ony implementujte a demonstrujte jejich použití na různých typech dat.
8. Zdokumentujte a zveřejněte výsledné add-ony.
9. Vytvořte video reprezentující výsledky vaší práce.

Literatura:

- Blender API documentation
- Adams, Ernest, and Joris Dormans. Game mechanics: advanced game design. New Riders, 2012. ISBN 0321820274, 9780321820273
- Eng, Lee Zhi. *Building a game with unity and blender*. Packt Publishing Ltd, 2015. ISBN-10: 178528214X
- Vaughan, William. *Digital modeling*. New Riders, 2011. ISBN-10: 0321700899

Při obhajobě semestrální části projektu je požadováno:
Body 1 až 6, experimenty vedoucí k bodu 7.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Chlubna Tomáš, Ing.**
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.
Datum zadání: 1.11.2022
Termín pro odevzdání: 10.5.2023
Datum schválení: 31.10.2022

Abstrakt

Práca sa zaoberá tvorbou doplnkov pre open source program Blender, ktorého úlohou je uľahčiť vytváranie herných assetov pre amatérskych vývojárov. Na rozdiel od ostatných add-onov, ktoré tieto problémy riešia podrobnejšie ale konkrétne, je toto riešenie istým spôsobom zjednotenie a zjednodušenie týchto nástrojov pre základné potreby užívateľa. Užívateľ si môže napríklad nechať vygenerovať sneh, charakter, obaľovacie boxy, spojený objekt s kostrou, levely detailov objektov alebo využiť nástroje na explodovanie objektov, nastavenie aktívnej textúry, vypočítanie hustoty pixelov pre objekt a načítanie objektov na prázdne body.

Abstract

The thesis deals with the creation of an add-on for the open source program Blender, whose task is to facilitate the creation of game assets for amateur developers. Unlike other addons that solve these problems in a more detailed but specific way, this solution is unification and simplification of these tools, for the basic needs of the user. For example, the user can generate snow, character, bounding boxes, merged object with skeleton, level of details, or use tools to explode objects, set active texture, calculate texel density for an object, and load objects on empty points.

Kľúčové slová

Blender addon, Herné assety, Tvorba hier, 3D modelovanie, Blender.

Keywords

Blender addon, Game assets, Game development, 3D modelelling, Blender.

Citácia

HRABOVSKÝ, Milan. *Kolekce nástrojů pro zjednodušení tvorby herních assetů v Blenderu.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Tomáš Chlubna, Ing.

Kolekce nástrojů pro zjednodušení tvorby herních assetů v Blenderu

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Tomáša Chlubny Ing. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Milan Hrabovský
9. mája 2023

Poďakovanie

Rád by som poďakoval môjmu vedúcemu, Tomáš Chlubna Ing., za spätnú väzbu, pomoc pri riešení problémov a ochotu so mnou spolupracovať na tejto práci.

Obsah

Kapitola 1

Úvod

Základom každej 3D videohry sú herné assety ktoré musel niekto vytvoriť a pripraviť. Veľké štúdia si na to platia skusených ľudí ale v prípade nezavislých vývojarov hier si často tieto assety musia robiť sami. V prípade že vývojar nemá dostatočné skúsenosti s nástrojmi pre týchto assetov, môže aj jednoduchá úloha zabrať hodiny práce.

Pre uľahčenie niektorých postupov, je cieľom tejto práce tvorba add-onu pre 3D editor Blender, ktorý bude obsahovať nástroje pre urýchlenie a uľahčenie procesov tvorby snehu, jednoduchých postavičiek, obaľovacích boxov, levelu detailov, prípravy objektov na textúrovanie, vizualizáciu objektov na prázdne body. Pre niektoré operácie existujú nástroje, avšak pre neskúsených uživateľov ponukajú príliš komplexné nastavenia ktorým nemusia rozumieť, alebo neponúkajú dynamický prístup k modifikovaniu parametrov. Výsledný addon je voľne dostupny na stiahnutie.

V kapitole 2 sú vysvetlené potrebné postupy a dôležíte informácie pre tvorbu herných assetov. Tieto informácie sú oporou pre implementáciu výsledného add-onu. Využitie konkrétnych nástrojov, návrh ako budu fungovať a vzhľad ako budu vyzerať uživateľské rozhranie pre modifikovanie parametrov sú popísané v kapitole 3. Kapitola 4 poskytuje informácie o tom ako sú jednotlivé nástroje implementované, ako k ním treba pristupovať a čo je výsledkom týchto operácii. Výsledné nástroje boli porovnané s manuálnymi postupmi potrebnými pre dosiahnutie rovnakého výsledku a otestvané potencionálnimi uživateľmi. Výsledky a porovnanie je možné nájsť v kapitole 5.

Kapitola 2

Teória

Pri tvorbe add-onu na uľahčenie tvorby assetov do hry, je potrebné mať radu znalostí o tom, čo je herný vývoj, čo sú herné assety a ako sú v hernom engine reprezentované. Dôležitý je proces tvorby herných assetov a čo všetko musia spĺňať, aby mohli byť použité vo finálnej publikácii hry. Ďalej je potrebné mať prehľad o tom, čo by najviac uľahčilo prácu neprofesionálnym vývojárom hier, keďže tento projekt je zameraný hlavne na proces tvorby herných assetov pre indie developerov[1]. V neposlednom rade treba byť oboznámený so systémom a nástrojmi, pomocou ktorých bude tento add-on vytvorený. Nasledujúce odseky podávajú bližšie informácie a vysvetlenie ku konkrétnym prvkom, z ktorých sa skladá výsledný produkt tejto práce.

2.1 Vývoj hier

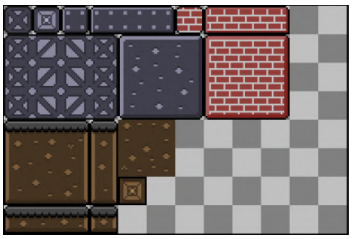
Vývoj hier je umenie, ktoré sa skladá z návrhu, vývoja a vydania hry. Pri tvorbe hry je dôležité myslieť na herné mechaniky, odmeny, zapojenie hráčov a dizajn úrovní. Vývojárom hry môže byť programátor, zvukový dizajnér, umelec, dizajnér a mnoho ďalších rolí, ktoré sú v tomto odvetví k dispozícii. Vývoj hier môže vykonávať veľké štúdio na vývoj hier alebo jednotlivec. Pokiaľ výsledný produkt umožňuje hráčovi interagovať s obsahom a manipulovať s hernými prvkami,je možné ho nazvať „hrou“ [4]. Na uľahčenie tvorby hier vznikli programy, ako sú napríklad Unreal Engine, Unity, Game Maker Studio a ďalšie programy s vlastným herným enginom. V dnešnej dobe sú tieto programy čoraz viacej používané ako malými vývojármi, tak aj veľkými hernými štúdiami.

2.1.1 Herné assety

Každá hra sa skladá z programu, ktorý ju ovláda, zvukov a vizuálnej stránky, ako sú napríklad objekty, mapy alebo postavy, ako je vidno na obrázkoch 2.1. Všeobecne môžeme assetom nazývať všetko, čo patrí do videohry [8]. Pri tvorbe herných assetov musí umelec dbať na to, aby spĺňal určité formálne požiadavky. Vizuálne assety musia vyzerať pekne, aby hráča neodradili a mal čo najlepší pôžitok z hry, avšak zároveň musia byť aj optimálne, aby nespotrebovali príliš veľa výkonu počítača.

[1] Nezavislí vývojári, ktorí nepatria pod žiadne herné štúdio

(a) 2D charakter

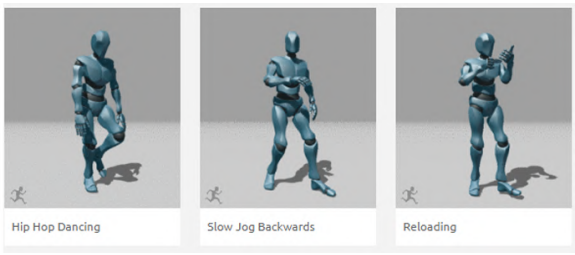


(b) 2D textúry

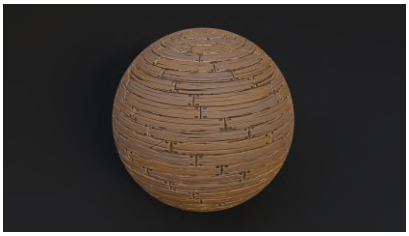
(c) 3D character

```
PlayerPrefs.SetInt("btc", 0);
PlayerPrefs.SetInt("cake", 0);
PlayerPrefs.SetInt("crowbar", 0);
PlayerPrefs.SetInt("halo", 0);
PlayerPrefs.SetInt("hammer", 0);
PlayerPrefs.SetInt("lightsaber", 0);
PlayerPrefs.SetInt("mushroom", 0);
PlayerPrefs.SetInt("pickaxe", 0);
PlayerPrefs.SetInt("pizza", 0);
PlayerPrefs.SetInt("ring", 0);
PlayerPrefs.SetInt("sweetroll", 0);
PlayerPrefs.SetInt("tetris", 0);
```

(d) Kódy, scripty



(e) Animácie postáv[2]



(f) 3D materíal

(g) Zvuky

Obr. 2.1: Obrázok ukazuje čo všetko herné assety môžu byť. Napríklad 2D postavičky (a), 2D textúry (b), 3D postavy (c), 3D anímácie (e), 3D materály (f), kódy / scripty (d), zvuky (g)

2.1.2 Level of Details (LOD)

Úroveň detailov je všeobecný termín, ktorý sa používa pri vykresľovaní objektov v herných enginoch. Objekty bližšie ku kamere sa vykresľujú s väčším počtom polygónov a sú detailnejšie, ako objekty, ktoré sa nachádzajú ďalej od kamery ako je zobrazené na obrázku 2.2. Všeobecne povedané, úroveň detailov je daná systémovými požiadavkami hry. Vzhľadom na výkon moderných procesorov je už badateľné len veľmi malé zhoršenie úrovne detailov [8]. V prvých videohrách boli zmeny úrovne detailov veľmi viditeľné. Objekty v popredí získavali viacej na definícii meshu, keď sa dostali do popredia. Išlo o jednoduchý systém, ktorý šetril výpočtový výkon pre iné úlohy a zároveň hráčovi poskytoval uchádzajúce po-

[2]Zdroj obrázku - https://www.mixamo.com/#/?page=1&type=Motion%2CMotionPack

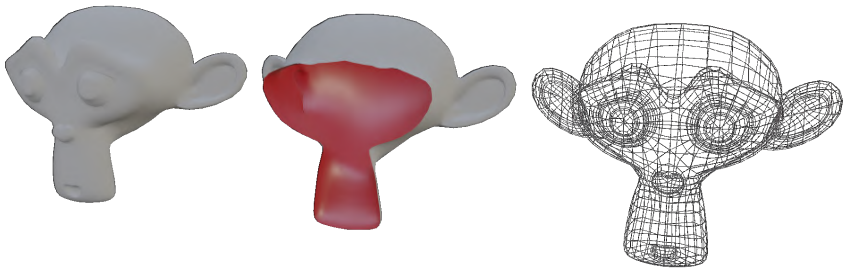
zadie. Moderné videohry stále používajú rovnaký postup, ale základná úroveň, z ktorej sa vykresľujú objekty, má oveľa vyšší počet polygónov, čo znamená, že len málo ľudí dokáže voľným okom rozpoznať rozdiel medzi popredím a pozadím.

Obr. 2.2: V smere zľava doprava je možné vidieť, ako objekty ďalej od kamery postupne strácajú na definícii pôvodnej geometrie, tvorí ich menší počet detailov a polygónov, čo zároveň znižuje výpočtovú náročnosť objektu.

2.2 Mesh

3D mesh je štrukturálna stavba 3D modelu, pozostávajúca z vrcholov, hrán a plôch. Vrchol tvorí jeden bod. Dva vrcholy tvoria hranu. Stena je tvorená tromi a viac vrcholmi alebo hranami. Najjednoduchšia forma steny sa nazýva polygón. Ide len o povrchovú reprezentáciu modelu, ktorá na rozdiel od volumetrickej, nemá žiadnu vnútornú výplň ako je možné vidieť na obrázku 2.3. 3D objekt je tvorený z jedného alebo viacerých meshov. Väčšinu 3D objektov vytvárajú umelci pomocou softvérov ako sú Maya, 3D Studio Max a Blender 3D [6].

3D meshe používajú referenčné body v osiach `X`, `Y` a `Z` na definovanie tvarov s výškou, šírkou a hĺbkou. V Blenderi sú tieto súradnice uložené vo forme vektorov. Súradnice môžu byť definované ako lokálne alebo globálne. Globálne súradnice vyjadrujú pozíciu v celej scéne, zatiaľ čo lokálne súradnice vyjadrujú relatívnu pozíciu voči objektu, s počiatkom v bode originu[3] objektu. Nad meshom je možné vytvárať maticové operácie, ako je posun, rotácia, zmena veľkosti, shear.



Obr. 2.3: Povrchová reprezentácia objektu. Je možné vidieť, že objekt zvnútra neobsahuje žiadnu výplň.

[3]Bod ktorý vyjadruje stred objektu, môže sa náchádzať mimo stredu geometrie

2.2.1 Bounding box

Bounding box, v preklade ohraničujúci box/kváder, je kváder o minimálnej veľkosti objektu tak, aby sa doň vmestil celý objekt [15]. V hrách sa používa na detekciu kolízie s inými objektmi, výpočtovo je menej náročný ako detekcia kolízie pre celý objekt, ktorý môže mať náročnú geometriu na výpočet, na rozdiel od bounding boxu, ktorý má v najjednoduchšej forme len 6 stien. Existuje viacero druhov bounding boxov. Najpoužívanejšie sú AABB (Axis Aligned Bounding Box), OBB (Oriented Bounding Box), Bounding sphere, Convex hull. Ich vizualizáciu v 3D priestore je možné vidieť na obrázku 2.4.

AABB

Osovo zarovnaný ohraničujúci box je box, ktorého hrany sú rovnobežné so súradnicovými osami. Je to najjednoduchšia forma ohraničujúceho boxu, ktorá sa používa na približné opísanie tvaru objektu. Jeho nevýhodou je, že neberie ohľad na lokálnu rotáciu objektu, a preto nemusí byť vždy presný.

OBB

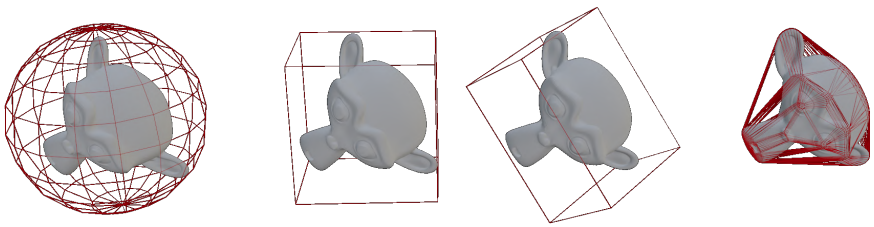
Orientovaný ohraničujúci box berie ohľad aj na lokálnu rotáciu objektu, takže je optimálnejší a presnejšie opisuje daný objekt. Jeho nevýhodou je zložitejší výpočet vzhľadom na lokálnu orientáciu objektu.

Bounding sphere

Ohraničujúca guľa je guľa opisujúca objekt, reprezentovaná stredom a priemerom, kde priemer je vzdialenosť medzi dvoma najvzdialenejšími bodmi objektu. Ohraničujúca guľa sa dá veľmi rýchlo testovať na vzájomnú kolíziu (dve gule sa pretínajú vtedy, ak vzdialenosť medzi ich stredmi je menšia ako súčet ich polomerov).

Convex hull

Konvexný obal je najmenšia konvexná množina, ktorá obsahuje objekt. Konvexný obal možno definovať ako priesečník všetkých konvexných množín obsahujúcich danú podmnožinu euklidovského priestoru. Zjednodušene povedané, obsahuje všetky dôležité body, ktoré definujú základný tvar objektu.



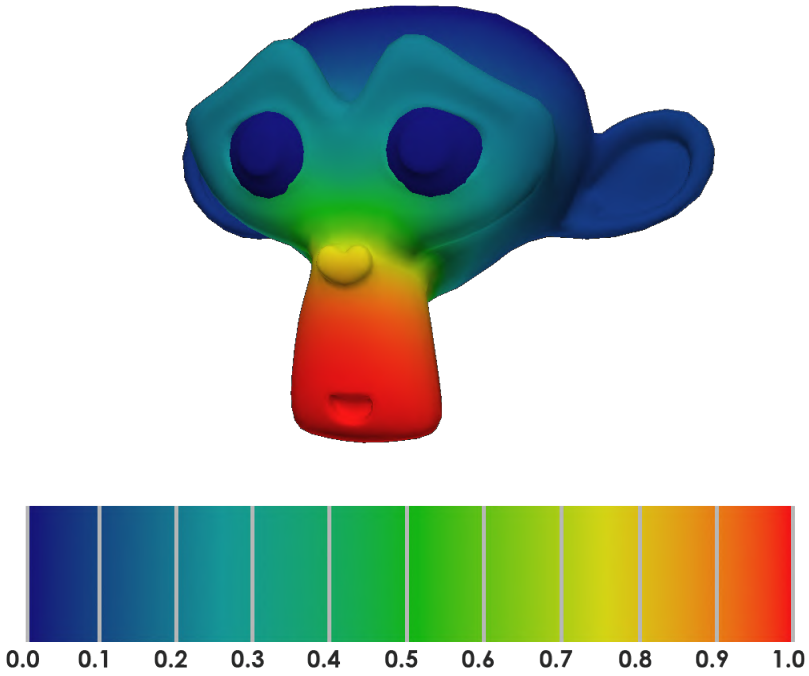
Obr. 2.4: Rozdielne druhy bounding boxov. V smere zľava doprava sa zvyšuje kvalita, lepšie opísaná geometria objektu, ale zároveň aj narastajúci čas a pamäťová náročnosť na výpočet kolízie.

2.2.2 Váhy vrcholov

Skupiny vrcholov môžu mať potenciálne veľmi veľký počet pridružených vrcholov, a teda aj veľký počet váh (viacej v sekcii 2.5). Weight Painting je metóda na udržiavanie veľkého množstva informácií o váhach, veľmi intuitívnym spôsobom.

Weight paint sa používa predovšetkým pri vytváraní kostry meshov (viacej v sekcii 2.5), kde sa skupiny vrcholov používajú na definovanie relatívnych vplyvov kostí na sieť. Používa sa však aj na ovládanie emisie častíc, hustoty vlasov, mnohých modifikátorov, tvarových kľúčov atď.

Váhy sa vizualizujú pomocou gradientu využívajúceho systém studených/horúcich farieb tak, že oblasti s nízkou hodnotou (s váhami blízkymi 0,0) sa zobrazujú ako modré (studené) a oblasti s vysokou hodnotou (s váhami blízkymi 1,0) sa zobrazujú ako červené (horúce). Sily rozdielných váh je možné vidieť na obrázku 2.5. Všetky hodnoty medzi nimi sa zobrazujú ako farby dúhy (modrá, zelená, žltá, oranžová, červená) [5].



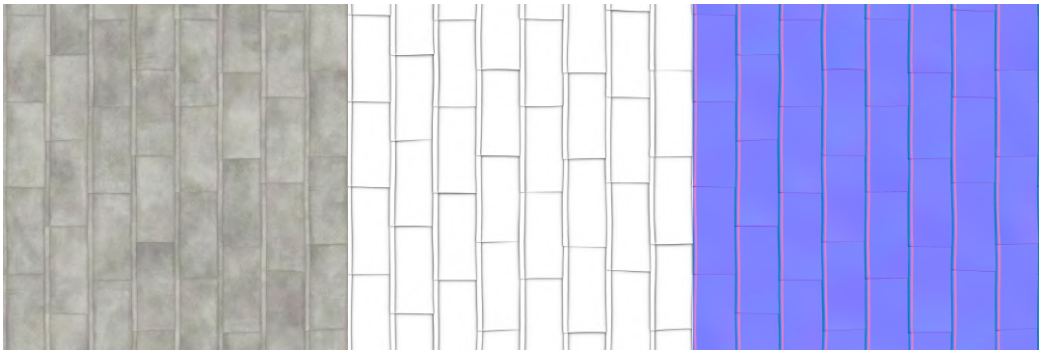
Obr. 2.5: Príklad použitia váhy pri tvorbe kostry a jej vplyv na geometriu objektu. Je možné vidieť, že spodné plochy objektu sú viacej ovplyvnené kostrou ako vrchné plochy.

2.3 Textúry

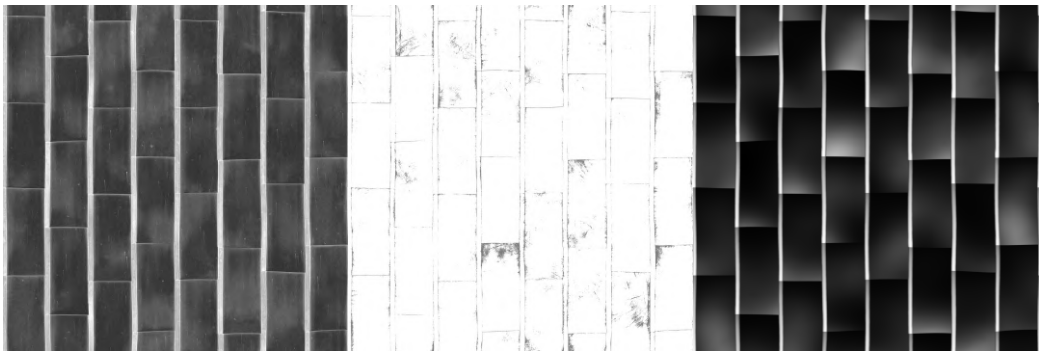
Každý 3D objekt je pokrytý rôznymi vrstvami textúr. Textúry sa môžu pohybovať od jednoduchých, opakujúcich sa vzorov, až po jedinečné obrázky vytvorené pre konkrétny 3D model a môžu potenciálne premeniť jednoduché tvary a scény na fotorealistické, sugestívne postavy a prostredia. Údaje v rámci materiálu zvyčajne obsahujú informácie týkajúce sa prvkov, ako je jeho farba alebo kombinácia farieb, stupeň odrazivosti alebo to, či je úplne nepriehľadný alebo do určitej miery priesvitný [1]. Vzhľad textúr a výsledný materál sú zobrazené na obrázku 2.6

Rôzne vlastnosti povrchu objektu môžu byť opísané nasledujúcimi typmi textúr:

- Diffuse/basecolor textúra popisuje farbu objektu.
- Roughness textúra popisuje odlesk.
- Metalic textúra popisuje metalický odraz.
- Ambient occlusion textúra definuje osvetlenie okolím a intenzitu odrazu svetla od objektu.
- Normal textúry definujú smer odrazu svetla objektu.
- Displacement textúry definujú povrch objektu.
- Alpha mapy predstavujú masku pre transparentné materiály.



(a) Textúra farby (b) Amient Occlussion mapa (c) Normal mapa



(d) Roughness mapa (e) Metalic mapa (f) Height mapa

(g) Výsledný materál

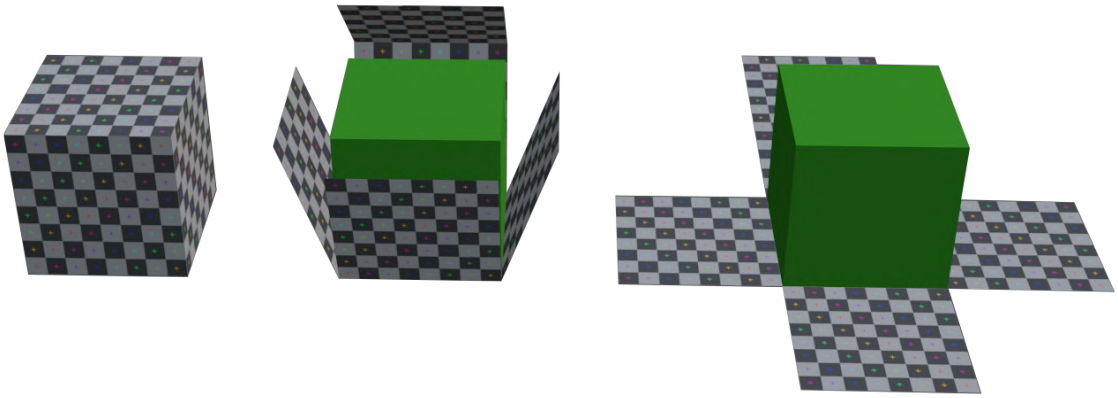
Obr. 2.6: Rozdielne typy textúr, popisujúce jeden materiál (a - f). Výsledný materiál (g) skladajúci sa zo základnej farby (a), ambient occlusion (b), normal mapa (c), odlesk (d), železitosť (e), height mapa (f).

V každej hre je potreba aj ďalších vlastností objektov, ako napríklad špinenie alebo opotrebenie objektu. Takýmto textúram stačí len jedna informácia a tou je rozsah, ktorý je definovaný pomocou intenzity bielej farby na jednom kanále textúry, vďaka čomu je možné mať až 4 rozdielne informácie na jednej textúre. Tieto mapy sa väčšinou označujú ako mask mapy.

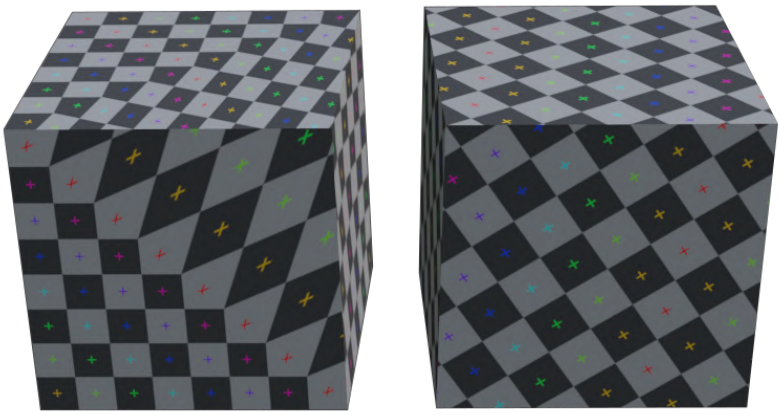
2.3.1 UV Mapping

UV mapa je 2D reprezentácia 3D objektu, ktorá sa používa na statické mapovanie textúr. U a V reprezentujú osi v 2D priestore [9]. UV mapa sa môže skladať z jedného alebo viacero UV štvorcov, kde každý štvorec predstavuje inú textúru. V prípade, že sa UV mapa skladá z viacero UV štvorcov, ide o U Dimension (ďalej iba UDIM) mapu. UDIM je systém posunu UV Mapy v `U` a `V` osi, kde je každý štvorec očíslovaný. UDIM mapovanie umožňuje mať viacero textúr s rozdielnym rozlíšením na jednej UV Mape.

Proces tvorby UV mapy sa volá UV mapping (rozbaľovanie na UV súradnice podobne ako na obrázku 2.7). Základom tvorby UV mapy sú „švy“ (z anglického slova „seam“). V mieste švu sa 3D model „rozstrihne“ a jeho plochy sa oddelia na UV mape. Keďže nie je možné reprezentovať 3D objekt v 2D priestore, prichádza tak k istej deformácii. Deformácia udáva, ako veľmi sa musel zmeniť pôvodný tvar polygónov na to, aby sa prispôsobili 2D priestoru. Príliš veľké skreslenie spôsobí viditeľné zmeny na finálnom zobrazení textúry na 3D objekte tak ako na obrázku 2.8. V hrách je potrebné, aby všetky objekty mali rovnakú mierku na textúre (texeldensity a jej využitie je vysvetlené v sekcii 2.3.3). Jeden objekt môže mať viacero UV kanálov, kde každý kanál môže obsahovať inak rozloženú UV mapu. Toto je vhodné napríklad vtedy, ak jeden kanál obsahuje informácie o materiáloch a ďalšie o svetle.



Obr. 2.7: Vizuálne zobrazenie rozbaľovania 3D objektu.



(a) Natiahnutie UV mapy (stretch) (b) Viditeľné švy

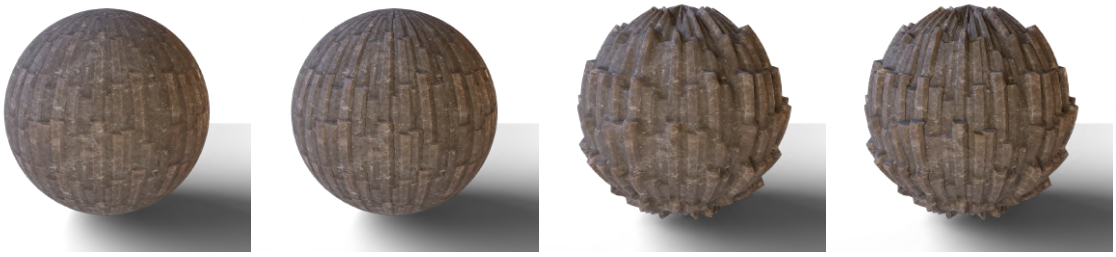
Obr. 2.8: Obrázok (a) ukazuje deformáciu, ak uhol hrán na UV mape je rozdielny od uhlov jednotlivých hrán tvoriacich danú plochu. Obrázok (b) zobrazuje problematiku viditeľných švov, keď sa hrany susediace so sebou nachádzajú každá na inom mieste textúry.

2.3.2 Bump, normal, displacement textúry

Bump mapy vytvárajú ilúziu hĺbky na povrchu 3D modelu. Textúry sa vytvárajú umelo pomocou odtieňov sivej a jednoduchých osvetľovacích trikov namiesto toho, aby sa jednotlivé nerovnosti a praskliny museli vytvárať ručne. Bump mapy nepridávajú modelu žiadne dodatočné rozlíšenie. Detaily ktoré pridávajú sú falošné. Pre zobrazenie detailov využívajú hodnoty v odtieňoch sivej na poskytnutie informácií do výšky alebo do hĺbky.

Normal mapy možno považovať za novší, lepší typ bump mapy. Rovnako ako pri bump mapách, aj pri normálových mapách sa do geometrie v scéne nepridáva žiadne dodatočné rozlíšenie. Normálová mapa používa informácie RGB, ktoré priamo zodpovedajú osiam X, Y a Z v 3D priestore. Táto informácia RGB hovorí 3D aplikácii o presnom smere orientácie normálov pre každý polygón. Normálové mapy sa pravidelne používajú v hernom priemysle pri prenášaní detailov z objektu s vysokým rozlíšením na objekt s nízkym rozlíšením. Nevýhodou normal máp je, že nevedia zobraziť vertikálne plochy.

Displacement mapy fyzicky upravujú povrch meshu, na ktorý sú aplikované. Aby bolo možné použiť na objekt displacement mapy, zvyčajne sa najskôr musí na objekt aplikovať subdivision (rozdelenie objektu na viacej polygónov), aby bolo možné lepšie popísať detail povrchu. Výhodou displacement máp je, že sa dajú v skutočnosti vytvoriť z modelu s vysokým rozlíšením alebo namaľovať ručne. Podobne ako bump mapa, aj displacement mapa sa skladá z hodnôt v odtieňoch sivej. Displacement a bump mapy sa môžu kombinovať. Rôzne vyśledky ich kobinácii je možné vidieť na obrázku 2.9.



(a) Objekt bez Bump a Displacement mapy (b) Bump mapa (c) Displacement mapa (d) Bump + Displacement mapa

Obr. 2.9: Rozdielne kombinácie textúr popisujúcich povrch a ich výsledný efekt. Na obrázku (a) je možné vidieť, že objekt má plochú geometriu a neobsahuje žiadne tiene. Po pridaní normálovej mapy (obrázok (b)), ktorá pridáva informácie kam smerujú plochy, je možné vidieť, že pribudli aj tiene, ale povrch geometrie nebol zmenený. Na obrázku (c) je možné vidieť, že po pridaní displacement mapy sa zmení povrch geometrie, avšak chýba informácia o tom, kam smerujú jednotlivé plochy, a preto nie je možné vidieť zatienenie objektu. Na poslednom obrázku pri skombinovaní oboch textúr je vidno, že sa zmenila geometria objektu a plochy majú svoj smer, takže je možné vidieť aj zatienenie objektu.

2.3.3 Texel density

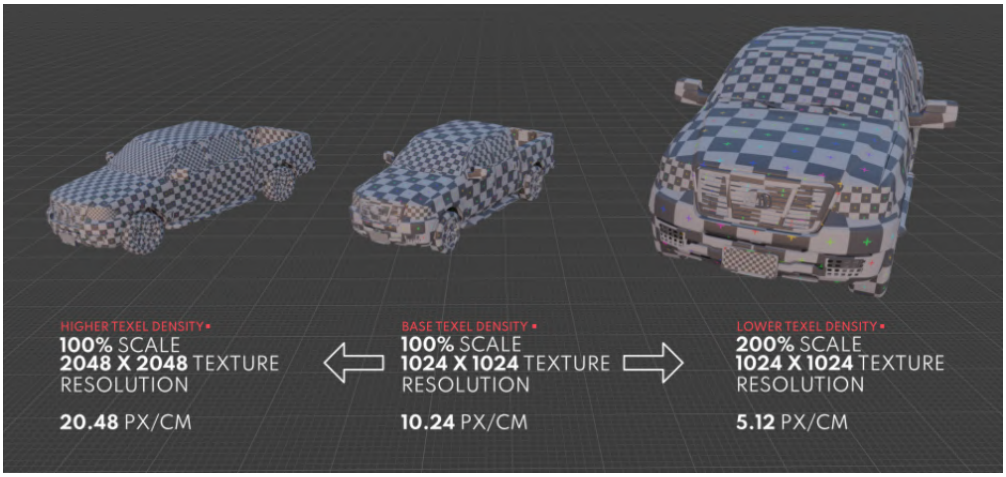
Texel density (hustota textúr) je merná jednotka, ktorá sa používa na to, aby boli textúry assetov v porovnaní s ostatnými v celom svete súdržné. Rozdielý veľkostí je vidno na obrázku 2.10. Meria sa v pixeloch na centimeter (napr. 2,56px/cm) alebo v pixeloch na meter (napr. 256px/m).

V hrách si možno všimnúť nesúlad medzi prvkami vo svete, keď textúry jedného prvku vyzerajú rozmazane a textúry druhého ostro. Často je to spôsobené nesúladom v hustote textúr. Zatiaľ čo samotné rozlíšenie týchto textúr sa môže v jednotlivých hrách a typoch hier meniť, hustota textúr je o konzistentnej mernej hodnote, ktorá je platná pre celú hru. Texel density závisí od dvoch vecí: veľkosti objektu a rozlíšenia textúry [7]. Vyššie rozlíšenie textúry a rovnaká mierka objektu spôsobí vyššiu hustotu textúry (menej rozmazaná textúra). Rovnaké rozlíšenie textúry a väčšia mierka objektu spôsobí nižšiu hustotu textúry (rozmazanejšia textúra).
Texel density sa vypočíta nasledujúcim vzorcom:

$$Res \cdot A_{UV}/A_{obj} = TD \tag{2.1}$$

$$napr.\ 1024px/100cm = 10.24px/cm \tag{2.2}$$

Kde jednotlivé parametré znamenajú:
Res - Rozlíšenie textúry
A_{UV} - Perenctuálne porkytie plochy UV mapou
A_{UV} - Veľkosť plochy označených stien na objekte
TD - Výsledná hustota



Obr. 2.10: Porovnanie hustoty textúr na rozdielnych rozlíšeniach textúry a veľkosti objektu. Pri zmenšení rozlíšenia textúry klesne hustota textúry. Takisto aj pri zväčšení rozmeru objektu klesne hustota textúry.

Odporúčaná hustota textúry na základe vzidalenosti od kamery je [12]:

- **Pohľad z tretej osoby** - $5.12px/cm$
- **Statický pohľad** - $1.28px/cm$ *alebo* $2.56px/cm$
- **Pohľad z prvej osoby** - $10.24px/cm$, môže byť aj viacej, záleží od blízkosti objektu ku kamere
- **Objekty v pozadí** - majú adaptívnu hustotu textúr (mip mapy), kde sa upravuje ich rozlíšenie na základe vzdialenosti od kamery.

2.3.4 Tvorba textúr

Textúrovanie je proces pridávania textúr do 3D objektu. Zahŕňa vytváranie textúr (buď z fotografií alebo od začiatku), aplikáciu textúr na 3D objekty a aplikáciu konečných detailov. Na vytvorenie textúr existujú k dispozícii tri hlavné techniky. Textúry môžu byť vytvorené ručne (namaľované). Môžu to byť naskenované materiály z reálneho sveta a premenené na textúry (Photoscan) , alebo môžu byť vygenerované počítačom (Procedurálne generovanie). Často sa tieto metódy kombinujú.

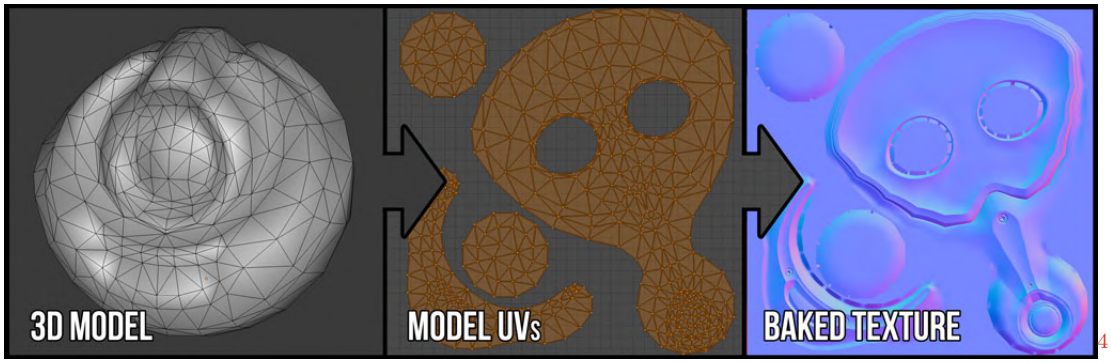
Ručne vytvorené textúry dávajú kontrolu a slobodu nad tvorbou. Je možné pridať prvky ako škrabance alebo opotrebovanie. Táto metóda je často využívaná pri tvorbe Štylizovaných textúr, ktoré pôsobia na užívateľa ako

ručne maľované štetcom. Naskenované a procedurálne generované textúry majú svoje obmedzenia, a preto sa často tieto dve metódy kombinujú, aby textúra pôsobila čo najreálnejšie. Z hľadiska optimalizácie sú členité povrchy objektov prevádzané na normal a displacement textúry. Grafik vytvorí 2 modely. Jeden, ktorý obsahuje plné detaily, ale má veľký počet polygónov a druhý, zjednodušený, ktorý má základný tvar a menej polygónov. Následne sú dáta z modelu s vysokými detailmi prenesené na objekt s nízkymi. Proces vytvárania špecifickej textúry pre objekt sa nazýva Bake (z anglického slova piecť, „zapekjú“ sa infomácie do pixelov). Postup pri tvorbe textúr je zobrazený na obrázku 2.11. Pre tvorbu

[3]Zdroj obrázku - https://www.beyondextent.com/deep-dives/deepdive-texeldensity

textúr existujú špeciálne nástroje ako napríklad Adobe Substance Painter, Marmoset Toolbag 3, Autodesk Maya, Blender, atď.
Pri zapekaní textúr je potrebné aby objekt spĺňal, určité požiadavky:

- Objekt musí mať rovnomernú veľkosť (x = 1, y = 1, z = 1)
- Musí obsahovať UV mapy (rozloženie 3D objektu na 2D súradnice popisujúce jeho povrch)
- Pohyblivé časti by mali byť od seba oddelené a mať medzi sebou určitý rozostup, pretože sa objekty môžu navzájom ovplyvňovať, napr. pri tvorbe ambient occlusion textúr alebo normal máp.
- Výsledný objekt by mal obsahovať jeden materiál a výsledné textúry popisujúce jeho vzhľad



[4]

Obr. 2.11: Proces tvorby textúr - najskôr sa vymodeluje objekt, potom sa rozloží na UV mapy a následne sú vytvorené (upečené) textúry.

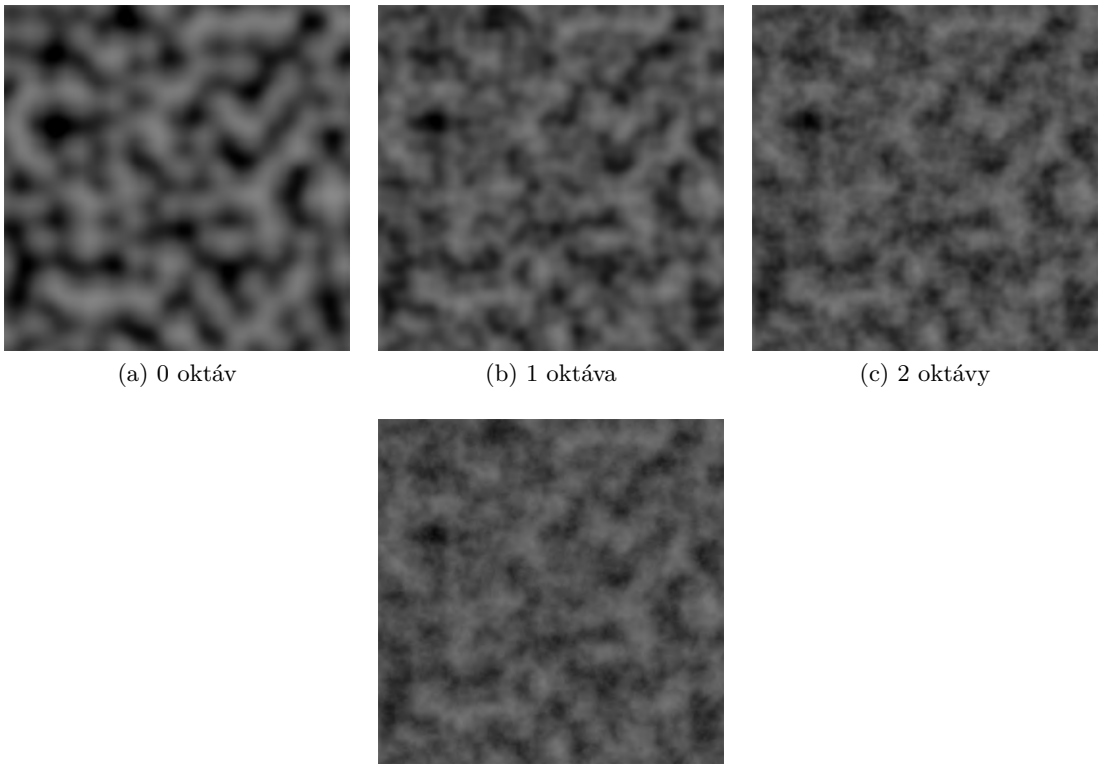
2.4 Textúra šumu

Šum alebo textúra šumu je pri procedurálnom generovaní jeden z najúspešnejších základných nástrojov používaných na generovanie vizuálnych detailov. Šum je náhodne generovaný neštruktúrovaný vzor, ktorý je možne použiť pre rôzne účely pri procedurálnom textúrovaní. Napríklad na zvrásnenie povrchu objektu, oblaky, vlny, náhodný pohyb animovaných postáv alebo generovanie náhodného rozsahu bodov (pixelov). Vo všeobecnosti sa často používa vo filmovej produkcii a aj vo videohrách. V súčasnosti je implementovaný v každom významnom balíku 3D grafického softwéru. Procedurálny šum sa veľmi rýchlo vyhodnocuje a má nízku pamäťovú náročnosť. Okrem toho s vhodnou kombináciou matematických operácií a sadou parametrov, je možné šum použiť na generovanie rôznych vzorov. Náhodné vzory sa často opisujú vo frekvenčnej oblasti. Zatiaľ čo v priestorovej oblasti signál určuje hodnotu pre každé miesto v priestore, vo frekvenčnej oblasti sa signál určuje amplitúdou a fázou pre každú frekvenciu. V prípade neštruktúrovaných vzorov je však fáza náhodná a neprispieva k užitočným informáciám. Preto sa šum často opisuje pomocou jeho výkonového spektra, ktoré špecifikuje veľkosť každej frekvencie a ignoruje fázu. Perlin a Hoffert uviedli túto definíciu: *Šum je aproximácia bieleho šumu s pásmovým obmedzením na jednu oktávu* [13].

[4] Zdroj obrázok - https://cgcookie.com/posts/big-idea-baking

Perlin noise

V roku 1985 Perlin predstavil Perlinov šum, populárnu procedurálnu funkciu šumu [10]. Perlinov šum sa najčastejšie implementuje ako dvoj, troj alebo štvorrozmerná funkcia, avšak môže byť definovaný pre ľubovoľný počet rozmerov. Implementácia sa zvyčajne skladá z troch krokov: definovanie mriežky náhodných gradientových vektorov, výpočet bodového súčinu medzi gradientovými vektormi a ich posunmi a interpolácia medzi týmito hodnotami. Výsledok súčtu každej ďalšej funkcie pridanej k Perlinovmu šumu sa nazýva oktáva. S vyšším počtom oktáv sa zvyšuje aj úroveň detailov Perlinovho šumu ako je vidieť na obrázku 2.12, avšak od určitého počtu oktáv (od určitej frekvencie), je úroveň detailov tak vysoká, že ju nie je možné zobraziť pomocou pixelov (amplitúda je príliš nízka). Dokonalou ukážkou využitia Perlinovho šumu je hra Minecraft, kde je za pomoci 3D Perlinovho šumu vygenerovaný svet a jaskyne v ňom.



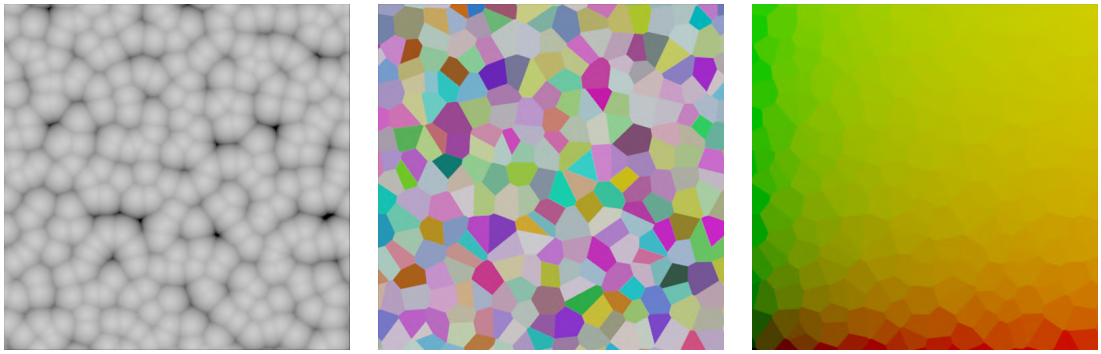
(a) 0 oktáv (b) 1 oktáva (c) 2 oktávy

(d) 15 oktáv

Obr. 2.12: Na obrázku je možné vidieť že od určite´ho počtu oktáv je menej vidno výsledné zmeny

Voronoi šum

Vornoi šum alebo Worleyho šum [11] je funkcia šumu, ktorú zaviedol Steven Worley v roku 1996. Vďaka svojmu organickému vzhľadu sa v počítačovej grafike používa napríklad na tvorbu materiálov kože, nepravidelných dlaždíc a zvrásnenia povrchu. Algoritmus funguje na princípe tvorby bunkovej štruktúry z bodov, ktoré sú náhodne rozmiestnené v 2D alebo 3D priestore. Bunková štruktúra je založená na vzdialenosti medzi bunkami ako je možné vidieť na obrázku 2.13. Worleyho šum pripomína bunkovú štruktúru v biológii. V matematike je možné nájsť využitie ako vornoi diagram, ktorý rozdeluje roviny na oblasti blízke každej z danej množiny bodov.



(a) Vzdialenosť medzi najbližšími bodmi

(b) Farba jednotlivých bodov

(c) Pozícia jednotlivých bodov

Obr. 2.13: Ukážka rozdielnych vlasnosti ktoré Worleyho textúra ponúka, jednotlivé vlastnosti sa dajú využiť na rôzne učely. Napríklad vzdialenosť medzi bodmi (a) na tvorbu bublinkove´ho povrchu, farba (b) na tvorbu mozaiky a pozícia (c) na tvorbu povrchu kože.

2.5 Skinned mesh/Rigging

Skinned mesh je mesh, ktorý má váhu kostí (viacej o váhe v sekcii 2.2.2). Pohybom kosti sa mesh deformuje podľa toho, ako veľmi je ovplyvnený váhou kosti. Deformácia meshu pripomína deformáciu kože na kĺbe ľudského tela, z čoho vzniklo pomenovanie skinned mesh (koža - z anglického slova „skin“). Rigging je proces vytvárania kostry pre 3D model, aby sa mohlo objektom pohybovať [16]. Najčastejšie sa používa na objekty, ktoré majú spojitý mesh (viacej v sekcii 2.2), kde bez rigu modelu nie je možné vytvoriť animácie, pretože 3D model nie je možné deformovať. 3D model sa riguje pomocou kostry, ktorá sa skladá z kostí (bones), ktoré môžu byť medzi sebou prepojené, alebo samostatne oddelené od ostatných kostí. Každá kosť sa skladá zo spojov (kĺbov), v mieste ktorých sa môže spájať s ostatnými kosťami. Prepojenie medzi kosťami tvorí strom, kde sa nadradená kosť volá parent a podradená child bone. V mieste kĺbu vzniká pohyb kosti a tak vzniká aj miesto, kde sa deformuje 3D model. Existujú dva prístupy ako možno pohybovať kosťami a ako sa navzájom ovplyvňujú: Inverse Kinematics a Forward Kinematics.

2.5.1 Inverse/Forward kinematics

Inverzná kinematika [3] znamená, že pohyb podriadenou kosťou v kostre môže ovplyvniť pohyb nadradených kostí. Podriadenou kosťou je možné ľubovoľne pohybovať/rotovať. Pre

zvyšné kosti je následne vypočítaná ich pozícia. Táto metóda je využívaná hlavne pri charakteroch, kde pri tvorbe animácie je jednoduchšie pohnúť jedným bodom a následne sa tak ohne napríklad celá ruka, ako pri kinematike smerom dopredu (Forward kinematics), kde je potreba každú kosť rotovať zvlášť.

Kinematika smerom dopredu znamená, že vplyv kostí medzi sebou bude nasledovať stromovú štruktúru. Kosti sa ovplyvňujú smerom dopredu, to znamená, že nadradená kosť môže ovplyvniť podradenú, ale podradená nemôže ovplyvniť nadradenú. Keďže pozícia nadradených kostí nie je vypočítaná, tak v prípade, keď sú kosti medzi sebou prepojené, je možné kosťami len rotovať v kĺbe a nie pohybovať. Táto metóda sa skôr využíva pri mechanických modeloch, pretože animátor tak má väčšiu kontrolu nad modelom.

2.5.2 Animácie

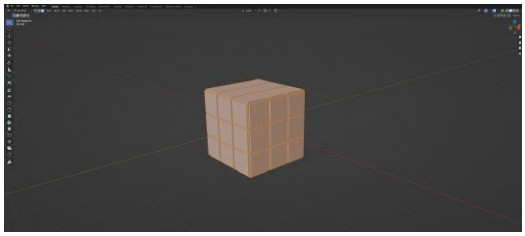
Animácia v 3D priestore, tak ako v 2D priestore, znamená zachytenie pohybu v čase. Každá animácia sa skladá zo snímkov, ktoré predstavujú mernú jednotku animácie (frekvencia snímkov za sekundu, v skratke FPS). FPS udáva plynulosť animácie. Čím je frekvencia vyššia, tým je obraz plynulejší. Výsledná dĺžka animácie sa tak dá vyjadriť ako počet snímkov za seknudu × dĺžka animácie v sekundách. Je potreba však myslieť na to, že vyšší počet snímkov za sekundu potrebuje viacej výpočtového výkonu. Každá zmena v čase je zachytená pomocou kľúča, ktorý zachytáva v danom bode (čase) informácie o objekte. V 3D svete je pri tvorbe animácií potrebné, aby objekt obsahoval kostru, v prípade že je potreba hýbať s jeho súčasťami (viacej v sekcii 2.5). Proces tvorby animácie pri 3D objekte pozostáva zo zachytenia pôvodných (počiatočných) dát v čase a výsledných (konečných dát v čase). Zachytávajú sa informácie ako rotácie, pohyb, veľkosť a tvar objektu. Každý kľúč v čase môže obsahovať rozdielne hodnoty týchto informácií. Každý objekt môže mať viacero klipov, kde každý klip predstavuje rozdielnu animáciu. V hernom priemysle je veľký počet klipov na jeden objekt bežnou vecou. Postavy obsahujú niekoľko druhov animácií, ktoré predstavujú rozdielne činnosti charakteru, ako napríklad chôdza, výskok, idle stav a mnoho ďalších.

2.6 Blender

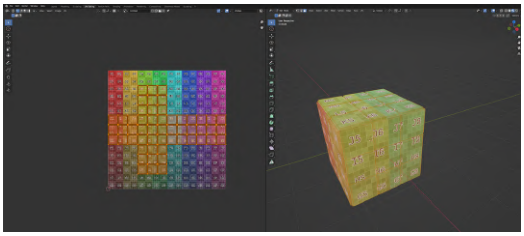
Blender[5] je bezplatný open source balík nástrojov pre tvorbu 3D. Podporuje celú 3D technológiu, ako je modelovanie, rigging, tvorba animácií, simulácie, renderovanie, kompozícia, sledovanie pohybu, strih videa a tvorba hier, niektoré možné využitia sú zobrazené na obrázku 2.14. Blender je distribuovaný pod licenciou GNU General Public License (GPL), a tak má verejnosť možnosť vykonávať malé aj veľké zmeny v základnom kóde, čo vedie k novým funkciám, opravám chýb a lepšej použiteľnosti.

Software má verejné rozhranie API pre skriptovanie v jazyku Python, pomocou ktorého môžu užívatelia upravovať alebo vytvárať vlastné nástroje. Blender je multiplatformový a funguje na všetkých zariadeniach Windows, Linux a Macintosh. Je veľmi obľúbený medzi indie vývojármi, keďže sa jedná o open source program a poskytuje všetko potrebné na tvorbu assetov do hry. Zároveň je veľmi jednoduchý na ovládanie a má širokú podporu verejnosti, čo sa týka návodov, nástrojov alebo dokumentácie. V posledných rokoch začínajú prechádzať aj veľké spoločnosti na Blender, a to práve kvôli prístupnému kódu a jeho cene (nemá žiadnu cenovku, ale je možné vývoj podporiť) [5].

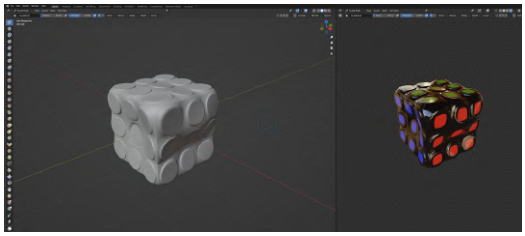
[5]Blender - https://www.blender.org/download/



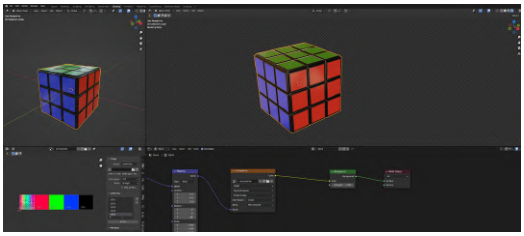
(a) Modelovanie



(b) Textúrovanie a tvorba textúr



(c) Sculpting

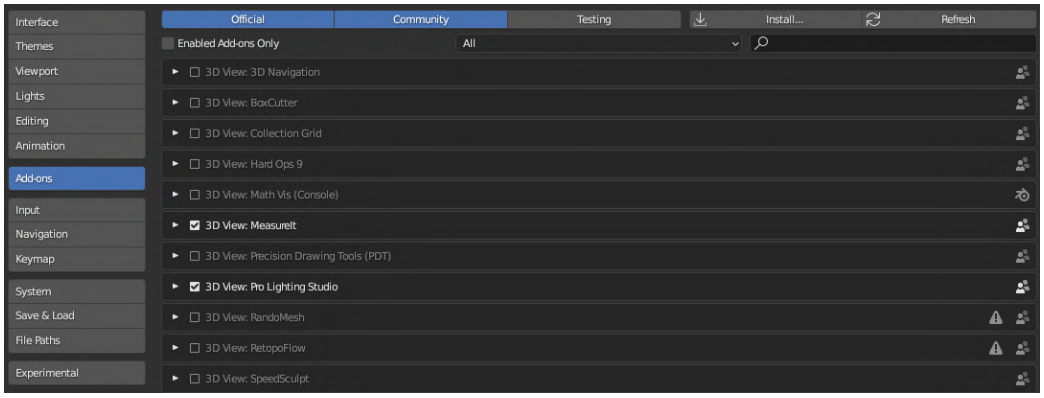


(d) Renderovanie

Obr. 2.14: Ukážka možných využití programu Blender

2.6.1 Add-ony

Vďaka tomu, že je Blender open source a má prístupné API a zdrojový kód, má širokú podporu verejnosti vo vylepšovaní a pridávaní nových funkcionalít. Funkcionality vytvorené užívateľmi sa nazývajú Add-ony (preložené ako dodatky). Základná verzia Blenderu obsahuje už nejaké komunitné, Blenderom schválené add-ony ako napríklad na obrázku 2.15, ktoré je možné aktivovať v Edit → Preferences → Add-ons. Užívateľ si takisto môže nainštalovať vlastné Add-ony, ktoré si môže sám vytvoriť, alebo ich môže zakúpiť na stránkach, ako sú napríklad Gumroad[6], Blendermarket[7] a ďalšie.



Obr. 2.15: Niektoré zo základnej ponuky add-onov v Blenderi 3.4

2.6.2 Modifikátory

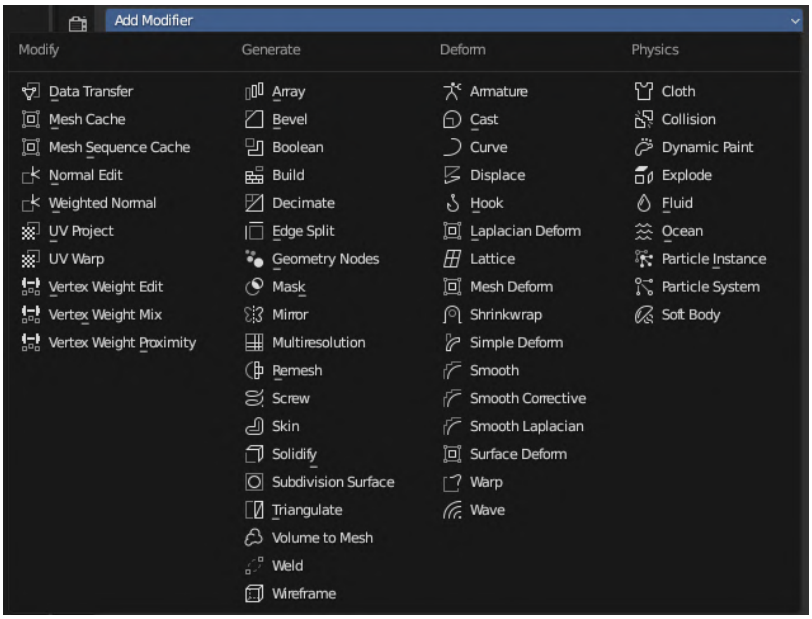
Modifikátory sú nedeštruktívne automatické operácie, ktoré ovplyvňujú a upravujú geometriu objektu. Modifikátory uľahčujú mnohé operácie, ktoré by v prípade ručného prevedenia

[6]Gumroad - https://gumroad.com/
[7]Blendermarket - https://blendermarket.com/

zabrali veľa času a deformovali by základnú geometriu objektu [5]. Napríklad booleanovské operácie, subdivision, extrude a mnoho ďalších. Modifikátory menia spôsob zobrazenia a vykresľovania objektu, zatiaľ čo geometria nie je reálne ovplyvnená a je ponechaná v základnom stave, čo zabezpečuje jednoduchšiu manipuláciu s objektom a jeho prípadnú úpravu pred finálnym aplikovaním modifikátorov. Každý objekt môže mať viacero modifikátorov, ktoré sú vykonávané postupne podľa poradia tak, ako sú zoradené v zozname. Každý modifikátor je možné následne aplikovať, čo vykoná trvalé zmeny na geometrii objektu. V prípade, že modifikátory nie sú aplikované v takom poradí v akom boli zoradené v zozname, je možné, že výsledná geometria bude rozdielna od tej, ktorá bola vizualizovaná. K vybranému objektu je možné pridať modifikátory pomocou ponuky Add Modifier. Každý nový modifikátor je pridaný na koniec zoznamu modifikátorov objektu, takže vie ovplyvňovať geometriu objektu ako posledný v poradí.

Blender obsahuje štyri kategórie modifikátorov, ktoré je možne vidieť na obrázku 2.16:

- **Modify** - Nástroje podobné nástrojom Deform, zvyčajne však neovplyvňujú priamo geometriu objektu, ale niektoré iné údaje, ako napríklad skupiny vrcholov.
- **Generate** - Ide o konštruktívne/deštruktívne nástroje, ktoré ovplyvnia celú topológiu modelu. Môžu zmeniť celkový vzhľad objektu, alebo doň pridať novú geometriu.
- **Deform** - Operácie, ktoré menia tvar objektu bez toho, aby menili jeho topológiu.
- **Simulate** - Sada modifikátorov, ktorá predstavuje fyzikálne simulácie.

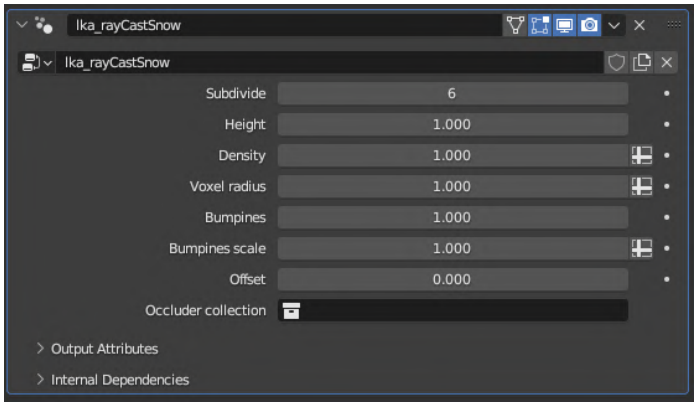


Obr. 2.16: Zoznam modifikátorov pre Blender 3.4

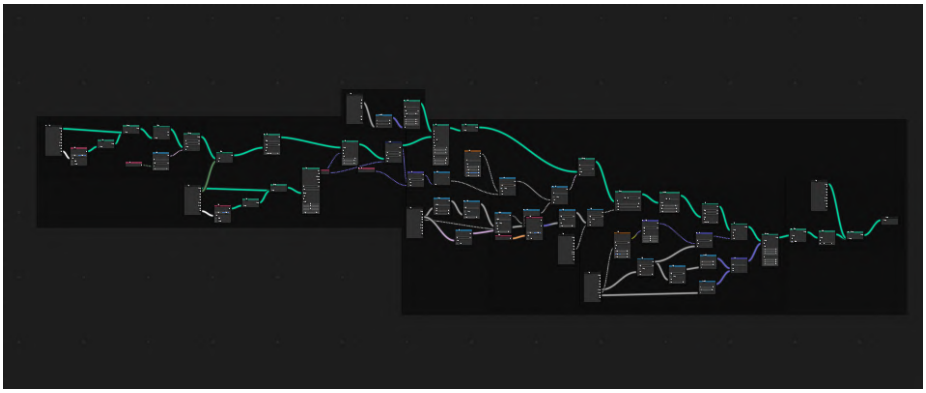
2.6.3 Geometry nodes

Verzia Blenderu 3.0 a vyššie priniesla funkcionalitu geometry nodes. Ide o systém spájania uzlov, v ktorom môže každý uzol predstavovať nejakú matematickú operáciu, operáciu s vlastnosťami objektu alebo iné operácie, ktoré modifikujú geometriu objektu. Veľkou výhodou je, že nemodifikujú priamo geometriu objektu, ale iba vizuálne, tzv. „nedeštruktívne“

modelovanie, vďaka čomu je možné zachovať geometriu v čo najjednoduchšom tvare a modifikovať ju až do momentu, pokiaľ nie je zmena geometrie trvalo aplikovaná. Geometry nodes ponúkajú možnosť definovania vlastných vstupných a výstupných premenných, takže užívateľ môže meniť hodnoty pohodlne aj v rozhraní modifikátorov ako je vidno na obrázku 2.17, ktorý sa môže pri veľkom počte operácií a uzlov ľahko stať neprehľadným ako zobrazuje obrázok 2.18.



Obr. 2.17: Výsledný modifikátor, ktorý obsahuje len potrebné hodnoty modifikujúce geometriu objektu, uľahčujúce prístup pre užívateľa, ktorý tak nemusí manuálne hľadať správny uzol v strome, ak potrebuje pozmeniť finálne vlastnosti modifikátora.



Obr. 2.18: Neprehľadný strom uzlov

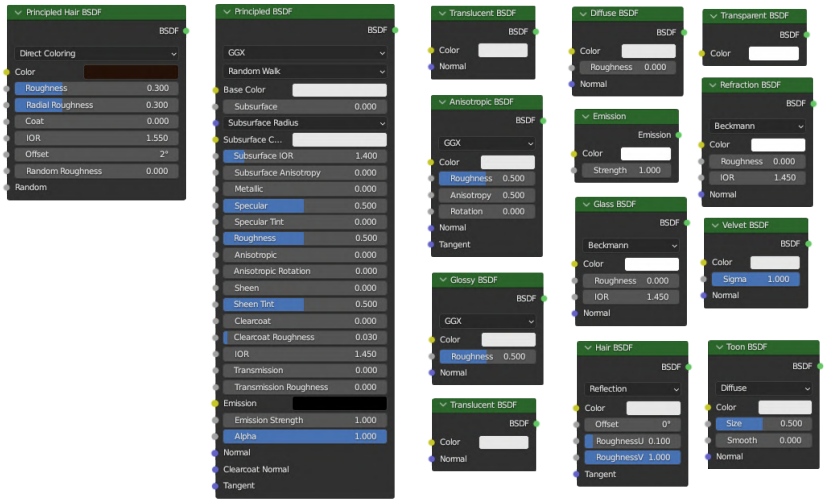
2.6.4 Shader editor

Shader editor je editor na tvorbu materiálov, ktoré sa používajú na vykresľovanie povrchu objektu. Materiály, ktoré sa používajú na render v cycles a EEVEE[8], sú definované ako strom uzlov, kde každý uzol predstavuje nejaký parameter [5]. Materiály môžu byť procedurálne generované, to znamená popísané nejakou matematickou funkciou alebo výpočtom. Druhá forma popisu materiálu je pomocou rasterovej textúry. Obidva spôsoby je možné v shader editore kombinovať a získať tak výsledný materiál pre objekt. Je dôležité si uvedomiť, že materiál nie je textúra. Jeden materiál popisujúci grafické vlastnosti objektu sa môže skladať z viacero textúr. Shader editor je základom pre úpravu materiálu. Jeho veľkou

[8]Blender render bežiaci v reálnom čase, postavený na OpenGL, zameraný hlavne na rýchlosť vykresľovania

výhodou je, že materiály môžu byť prevedené na textúru a následne exportované a použité v ľubovoľnom hernom engine.

Blender 3.4 predvolene obsahuje 14 shaderov ktoré je vidno na obrázku 2.19, pomocou ktorých je možné dosiahnuť rozdielne výsledky. Najpoužívanejší je Principled BSDF, ktorý obsahuje všetky vlastnosti materiálu.



Obr. 2.19: Preddefinované shadery v Blenderi 3.4, každý z nich je iného typu a ma rôzne vlastnosti. napriíklad Transparent BSDF pracuje s priehľadným materálom, Glass BSDF imutuje materál skla, Diffuse BSDF obsahuje základnú farbu. Zjednotením niektorých shaderov dokopy je Principled BSDF, ktorý obsahuje všetky vlastnosti potrebné pre materál ako napríklad odlesk, priehľadnosť, svietivosť, farba, železitosť atď.

2.6.5 Programovianie v Blenderi

Blender ponúka prístup ku všetkým operátorom, dátovým typom a nástrojom na modelovanie pomocou Pythonu. Tento prístup dovoľuje modelovať rôzne objekty, upravovať ich a vykonávať operácie nad nimi, volať všetky Blenderom definované operátory a operácie a vykonávať matematické výpočty a kontroly. Pomocou Python scriptu je možné vytvoriť Blenderom definované dátové typy, ktoré môžu odovzdávať a zachytávať rozdielne hodnoty, s ktorými môže užívateľ ľubovoľne manipulovať. Aby bolo možné pristupovať k týmto dátovým typom, je potrebné ich registrovať v scéne ako vlastný dátový typ. Dátové typy v Blenderi prepájajú užívateľské rozhranie s Pythonovským scriptom. Blender taktiež ponúka triedy, z ktorých je možné následne vygenerovať vlastné užívateľské rozhranie alebo operátory. Každá trieda vytvorená v add-one musí byť pred použitím registrovaná v triedach Blenderu. V opačnom prípade nie je možné k danej triede pristúpiť.

2.7 Vzorce na výpočet obsahu mnohouholníka

Pri výpočtoch hustoty textúry (viacej v sekcii 2.3.3), je potrebné vypočítať obsah plochy UV mapy v UV editore. Existuje viacero vzorcov, pomocou ktorých je možné vypočítať plochu mnohouholníka daného bodmi, ktoré sú opísané kartézskymi súradnicami v 2D priestore.

V prípade, že mnohouholník je záporne orientovaný, veľkosť plochy vyjde tiež záporná. V tomto prípade je možné získať veľkosť plochy ako absolútnu hodnotu z veľkosti vypočítanej oblasti. Najznámejšie vzorce na výpočet plochy sú lichobežníkový vzorec, trojuholníkový vzorec a šnúrkový vzorec [2] .

Šnúrkový vzorec

Šnúrkový (z anglického výrazu „shoelace“ - šnúrky do topánok) vzorec, [14] známy tiež ako Gausova metóda, je matematický algoritmus, pomocou ktorého je možné vypočítať aj plochu zložitého mnohouholníka, kde jednotlivé hrany tvoriace mnohouholník môžu zasahovať cez seba a navzájom sa krížiť. Šnúrkový sa nazýva preto, lebo krížové násobenie súradníc vrcholov tvoriacich mnohouholník pripomína navliekanie šnúrok do topánky. Základom šnúrkového vzorca je súčet 2×2 determinantov (viac v sekcii 2.3) s pozíciami jednotlivých vrcholov v protismere hodinových ručičiek tvoriacich mnohouholník. Šnúrkový algoritmus tento proces optimalizuje do jedného determinantu:

$$|A| = \frac{1}{2} \cdot \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \dots & \dots \\ x_n & y_n \\ x_1 & y_1 \end{vmatrix} \tag{2.3}$$

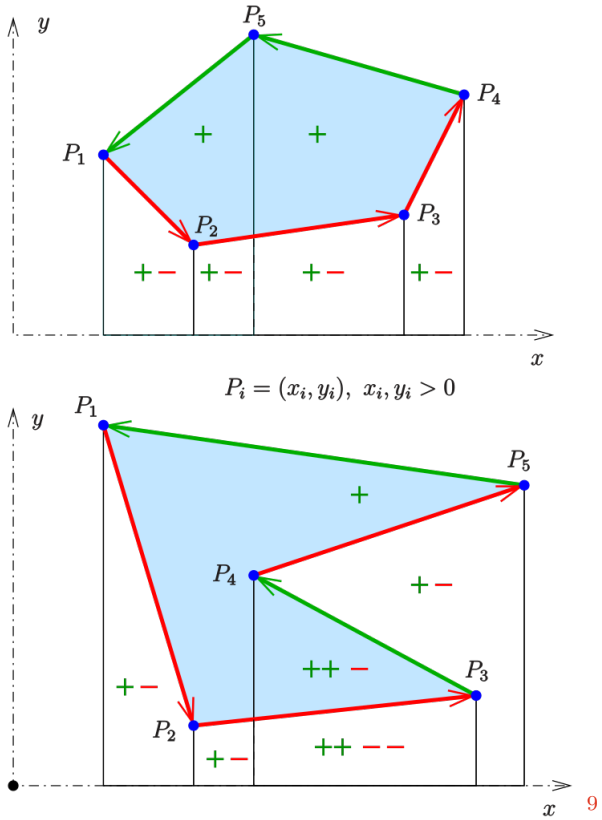
Kde jednotlivé parametré znamenajú:
A - Veľkosť plochy mnohouhoľníka.
X - X-ová súradnica bodu tvoriacého mnohouhoľník.
Y - Y-ová súradnica bodu tvoriacého mnohouhoľník.
n - Posledný bod tvoriací mnohouhoľník, v protismere hodinových ručičiek.

Lichobežníkový vzorec

Lichobežníkový vzorec 2.4 sčítava postupne všetky orientované plochy lichobežníkov tvoriacich mnohouholník ako je vidno na obrázku 2.20:

$$|A| = \frac{1}{2} \cdot \sum_{i=1}^{n} (y_i + y_{i+1}) \cdot (x_i - x_{i+1}) \tag{2.4}$$

Kde jednotlivé parametré znamenajú:
A - Veľkosť plochy mnohouhoľníka.
X - X-ová súradnica bodu tvoriaceho mnohouhoľník.
Y - Y-ová súradnica bodu tvoriaceho mnohouhoľník.
i - Poradie bodu tvoriacého mnohouhoľník, v protismere hodinových ručičiek.
n - Počet bodov tvoriacích mnohouhoľník.



[9]

Obr. 2.20: Na obrázku možno vidieť, ako je mnohouholník rozdelený na jednotlivé lichobežníky s vrcholmi v bodoch tvoriacich tento mnohouholník. Pre každý lychobežník je postupne vypočítaná plocha. Súčet plôch lychobežníkov sa rovná veľkosti plochy mnohouhoľníku.

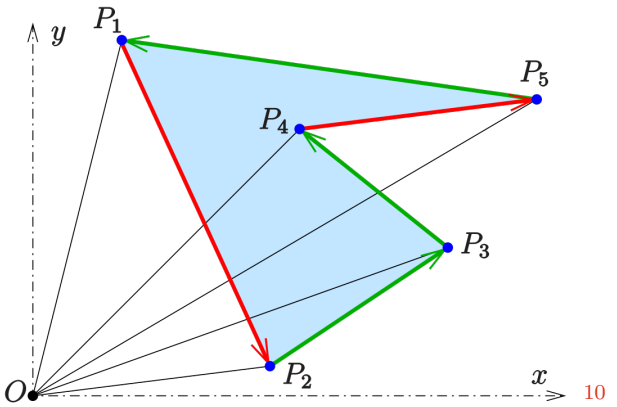
Trojuholníkový vzorec

Trojuholníková formula sčítava postupne všetky orientované plochy trojuholníkov tvoriacich mnohouholník ako je vidno na obrázku 2.21:

$$|A| = \frac{1}{2} \cdot \sum_{i=1}^{n} \begin{vmatrix} x_i & x_{i+1} \\ y_i & yi+1 \end{vmatrix} \tag{2.5}$$

Kde jednotlivé parametré znamenajú:
A - Veľkosť plochy mnohouhoľníka.
X - X-ová súradnica bodu tvoriaceho mnohouhoľník.
Y - Y-ová súradnica bodu tvoriaceho mnohouhoľník.
i - Poradie bodu tvoriacého mnohouhoľník, v protismere hodinových ručičiek.
n - Počet bodov tvoriacích mnohouhoľník.

[9] Zdroj obrázku 1.22 - https://en.wikipedia.org/wiki/File:Trapez-formel-prinz.svg



Obr. 2.21: Na obrázku možno vidieť, ako je mnohouholník rozdelený na jednotlivé trojuholníky s vrcholmi v bodoch tvoriacich tento mnohouholník. Pre každý trojuholník je postupne vypočítaná plocha. Súčet plôch trojuholníkov sa rovná veľkosti plochy mnohouholníku.

[10]Zdroj obrázku - https://en.wikipedia.org/wiki/File:Trapezformel-3eckform.svg

Kapitola 3

Návrh

Cieľom práce je tvytoriť add-on ktorý ponúka sadu nástrojov. Nástroje je možné rozdeliť do 4 kategorii: `Mesh Tools`, `Simple Character Generátor`, `Snow Generátor`, `Texture Tools`. Každá kategória obsahuj rozdielne typy nástrojov podľa názvu. Mesh tools pracuje s objektami a geoemtriou objektov. Generátory generujú pre uživateľa sneh a postavy. Texture tools ponúkajú operácie pre proces tvorby textúr.

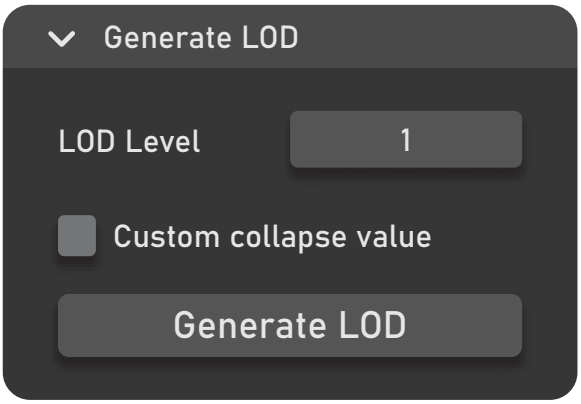
3.1 Mesh Tools

Sada nástrojov Mesh Tools ponúka užívateľovi operácie často potrebné pri tvorbe herných assetov. Tieto operácie uľahčujú aktivity nad objektmi, ktoré by v prípade manuálneho vykonávania zabrali viacej času.

3.1.1 Generate LOD

Pomocou tejto funkcie, ktorú ponúka Blender add-on, môže užívateľ jedným kliknutím vygenerovať z ľubovoľne vybraného objektu rôzne úrovne detailov (viacej v sekcii 2.1.2). Úrovne detailov, ktoré môže užívateľ vybrať sú v rozmedzí 1 až 4, keďže za nultú úroveň je považovaný objekt v plných detailoch. Užívateľ si vyberie objekt, následne zvolí úroveň ako je vidno na obrázku s návrhom 3.1 a nechá vygenerovať objekt s nižším počtom detailov. Tento objekt je následne pomocou modifikátora Decimate v Blenderi upravený na požadovanú úroveň. Modifikátor je použitý na kópii objektu, aby sa tak zachovala aj originálna verzia objektu.

Na zníženie počtu detailov a zjednodušenie geometrie je použitá možnosť modifikátora collapse, ktorá spája trojuholníky meshu podľa nastaveného pomeru. Pre každý objekt sa vypočíta počet polygónov, z ktorých sa skladá objekt a na základe vypočítaného počtu polygónov je následne zvolená hodnota, podľa ktorej sa určí, ako veľmi a koľko trojuholníkov sa spojí. Tento výpočet zabezpečuje ochranu pre objekty, ktoré majú nízky počet polygónov tvoriacich mesh objektu. Táto ochrana objektov je potrebná v prípade, že užívateľ zvolí na objekt s nízkym počtom polygónov hodnotu úrovne detailov, ktorá by odstránila až príliš veľké množstvo detailov tvoriacich objekt, ktorý by následne stratil dôležité plochy definujúce geometriu. Výsledná geometria by tak nebola použiteľná v hre. Parameter `Custom collapse value` ignoruje vypočítane hodnoty a používa hodnotu uživateľom zvolenú. Po zjednodušení geometrie je modifikátor aplikovaný, aby sa zmeny vykonali natrvalo. Následne je kópia objektu premenovaná tak, aby užívateľ vedel, z akého objektu je LOD objekt vytvorený a o akú úroveň detailov ide.



Obr. 3.1: Návrh uživateľského rozhrania pre generovanie levelu detailov.

3.1.2 Bounding Box

Bounding box generátor dáva možnosť užívateľovi vygenerovať bounding volume (viacej v sekcii 2.2.1) okolo vybraného objektu. Bounding volume objekty sú v hrách využívané na detekciu kolízií. Užívateľ si vyberie z ponuky druhov bounding volume objektov ako je možné vidieť na obrázku 3.2, ktoré je možné vygenerovať a následne na zvolený objekt vygeneruje bounding box. V ponuke sú 4 druhy bounding boxov: AABB, OBB, Bounding sphere a Convex hull. Každá možnosť ponúka rozdielnu presnosť obalenia objektu a náročnosť na výpočet detekcie kolízie.

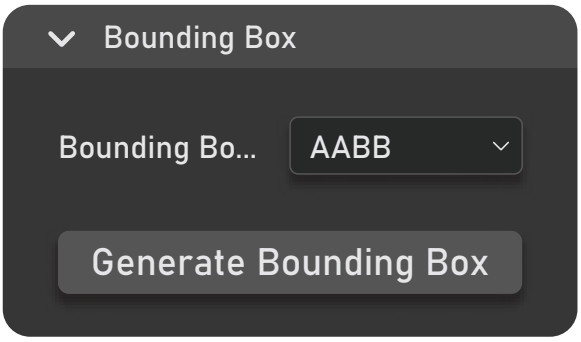
Všetky druhy bounding boxov sú vygenerované iba pre jednotlivé objekty, keďže je zbytočné generovať box pre kolekciu objektov, ktoré následne môžu zmeniť svoju pozíciu a nie sú pevne pripojené k objektu.

V prípade, že si užívateľ vyberie možnosť AABB, vygeneruje sa box zarovnaný s osami sveta. Ak má objekt globálnu rotáciu, táto rotácia je prenesená na vygenerovaný bounding box, aby sa rotácie objektu aj bounding boxu zhodovali. Rotácie sú prenesené aj pre možnosti OBB a Convexhull. V prípade bounding sphere je táto informácia zbytočná, keďže rotácia nemá vplyv na tvar gule. V prípade OBB je zohľadnená aj lokálna orientácia objektu a jeho pozícia. Rotácie sú následne vypočítané podľa tvaru meshu.

Pre možnosť bounding sphere je z meshu objektu zistený stred geometrie podľa priemernej pozície jednotlivých bodov tvoriacich objekt. Priemer gule je vypočítaný ako 2 krát vzdialenosť najvzdialenejšieho bodu od zisteného stredu.

Na Convex hull je použitá operácia z Blenderu, ktorá vygeneruje z meshu Convexhull. Pred použitím tejto možnosti musí byť najskôr objekt skopírovaný, aby nedošlo k modifikovaniu pôvodnej geometrie objektu. Následne je potrebné ešte odstrániť z duplikátu pôvodnú geometriu, ktorá zostala po generovaní.

Každý vygenerovaný bounding box dostane názov pôvodného objektu s príponou `_bbox`, aby užívateľ vedel, že ide o ním vygenerovaný bounding box.



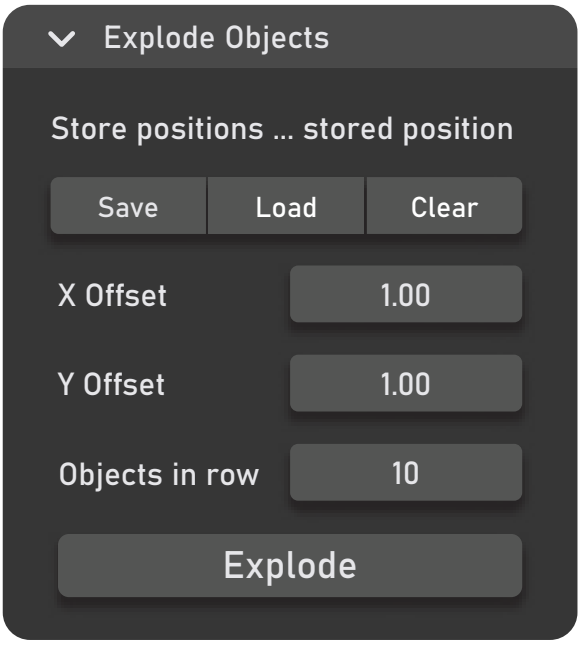
Obr. 3.2: Návrh uživateľského rozhrania pre generovanie bounding boxov.

3.1.3 Explode Objects

Pri tvorbe textúr na objekty je potrebné, aby pohyblivé časti modelu boli mimo dosahu lúčov, ktoré prenášajú informácie o osvetlení objektu okolím na textúru objektu. V opačnom prípade sa môžu objekty navzájom ovplyvňovať. Pre toto je potrebné, aby si užívateľ vždy rozhádzal objekty, čo pri zložitejšom modeli môže zabrať zbytočne veľa času.

Nástroj Explode objects vykoná rozhádzanie za užívateľa. Užívateľ si na osi X a Y zvolí vzdialenosť medzi origin bodmi objektov a počet objektov v jednom riadku ako je možné vidieť na obrázku 3.3. Na základe týchto informácií sú objekty posunuté od seba tak, aby boli medzery medzi objektmi podľa hodnôt, ktoré si nastavil užívateľ. Posunuté sú všetky objekty, ktoré si užívateľ označil. Posun nemá vplyv na pozíciu objektov, ktoré neobsahujú mesh dáta (viacej v sekcii 2.2. Ostatné typy objektov sú ignorované.

Užívateľ má možnosť uložiť pozície objektov, aby si následne vedel objekty vrátiť do pôvodného stavu. V prípade, že by užívateľ chcel pridať ďalšie objekty a rozhádzať ich pozíciu a má označené aj objekty, ktoré už pozíciu majú uloženú, môže ich pozíciu uložiť bez toho, aby bola prepísaná pozícia objektov, ktoré už uloženú pozíciu majú. Ak by potreboval užívateľ prepísať pozície už uložených objektov, má možnosť vymazať označené objekty zo zoznamu a môže tak nanovo uložiť ich pozíciu.

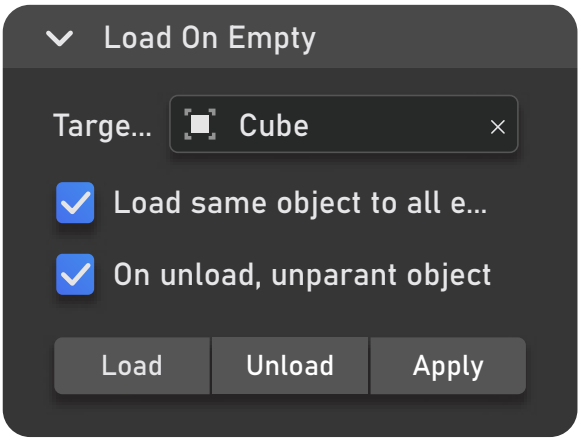


Obr. 3.3: Návrh uživateľského rozhrania pre posuv objektov od seba.

3.1.4 Load On Empty

Táto funkcia je veľmi užitočná v prípade, keď užívateľ potrebuje vizualizovať objekty, ktoré sa budú dodatočne načítať na prázdne body modelu v hre. V prípade, že užívateľ má ako aktívny objekt zvolený empty object, má možnosť si na tento objekt načítať ľubovoľný objekt zo scény ako je možné vidieť na obrázku 3.4. V opačnom prípade je táto možnosť zablokovaná. Užívateľ si vyberie target object (ľubovoľný objekt v scéne) a následne ho načíta. Pri načítaní sa vytvorí kópia objektu, ktorá zdieľa všetky dáta s pôvodným objektom. Takéto prevedenie ponecháva možnosť užívateľovi editovať originálny objekt s tým, že všetky zmeny na originálnom objekte sú prenesené na každú kópiu v scéne. Každá kópia objektu je následne pripojená k objektu ako child object a dedí tak jeho pozíciu, rotáciu a veľkosť.

Pokiaľ užívateľ zvolí možnosť `Load same object to all empties`, tak na každý aktívny empty objekt je načítaný rovnaký objekt, aký je target objekt v prípade aktívne zvoleného empty objektu. V opačnom prípade je na každý empty objekt načítaný ľubovoľne zvolený objekt. Užívateľ má taktiež možnosť tieto kópie tzv. „odnačítať“ alebo aplikovať tieto objekty a vytvoriť z nich tak objekty, ktoré majú vlastné dáta a už nezdieľajú informácie s pôvodným objektom. Užívateľ ich tak môže ľubovoľne modifikovať bez ovplyvnenia geometrie ostatných kópií a originálneho objektu.



Obr. 3.4: Návrh uživateľského rozhrania pre načítanie borázkov na prázdne body.

3.1.5 Convert To Skeleton

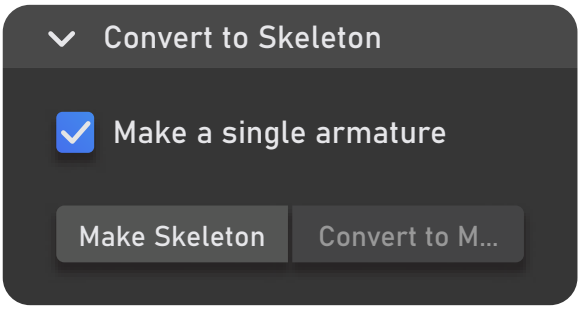
Niektoré herné enginy vyžadujú, aby objekty, ktoré obsahujú animácie mali kostru, pretože nepodporujú priamy pohyb objektu bez kostry. V takýchto prípadoch je potrebné, aby užívateľ pripravil objekt v 3D softwéri. Tento proces môže byť pre neskúsených ľudí komplikovaný a zdĺhavý. Práve v tomto prípade je pre užívateľa nápomocný nástroj Convert To Skeleton a convert To Mesh, ktorý tento proces vracia. Parametre a operácie je možné vidieť na obrázku 3.5.

Táto funkcia prevedie všetky objekty typu mesh na jeden spojený objekt obsahujúci kostru, pomocou ktorej je možné jednotlivými súčasťami pohybovať. Tento nástroj je vhodný pre mechanické modely skladajúce sa z viacero pohyblivých objektov, ako napríklad vozidlá, budovy, rôzne stroje a nie pre organické modely, ako sú charaktery, zvieratá alebo rastliny pretože neposkytuje možnosť generovania spojitej kostry, všetky kosti sú oddelené.

Užívateľ si vyberie jeden a viacej objektov a následne nechá program vygenerovať kostru a spojený objekt. Na pozícii počiatočného bodu každého objektu sa vytvorí jednoduchá kosť, ktorá bude pohybovať daným objektom. Každá kosť obsahuje rovnakú pozíciu, rotáciu a veľkosť ako objekt, ktorým bude pohybovať. Každému objektu je priradená skupina vertexov, ktorá obsahuje všetky vertexy daného objektu, aby bolo možné pri spojení všetkých objektov do jedného rozlíšiť, ktorý mesh patrí ku ktorej kosti. Po vygenerovaní každej kosti sú všetky vybrané objekty následne spojené do jedného. Všetky dáta o objektoch, z ktorých sa výsledný objekt skladá sú uložené pre budúcu potrebu.

Aby sa prepojil výsledný objekt s kostrou, je k výslednému objektu pridaný modifikátor Armature, ktorý obsahuje vygenerovanú kostru, aby sa prepojila kostra s objektom a dalo sa tak pohybovať jeho časťami. Pohyb objektov využíva forward kinematics (viacej v sekcii 2.5.1). V prípade, že užívateľ vypne možnosť `Make a single armature`, výsledná kostra nie je spojená do jednej, ale každá kosť pohybujúca objektom je braná ako zvlášť armatúra. V tomto prípade je k objektu pridaný modifikátor Armature pre každú jednu kosť.

Užívateľ má taktiež možnosť vrátiť objekt do pôvodného stavu a rozdeliť ho na objekty podľa toho, ako boli oddelené pred spojením do jedného objektu. Pri tomto procese sú využité informácie, ktoré sú uložené v prípade tvorby kostry. Objekt je rozdelený podľa jednotlivých skupín bodov, pomocou ktorých je určené, ktorá geometria patrí ktorému objektu. Po oddelení dostanú všetky objekty naspäť svoje meno, pozíciu, rotáciu, veľkosť a pozíciu počiatočného bodu, ktorú daný bod obsahoval.



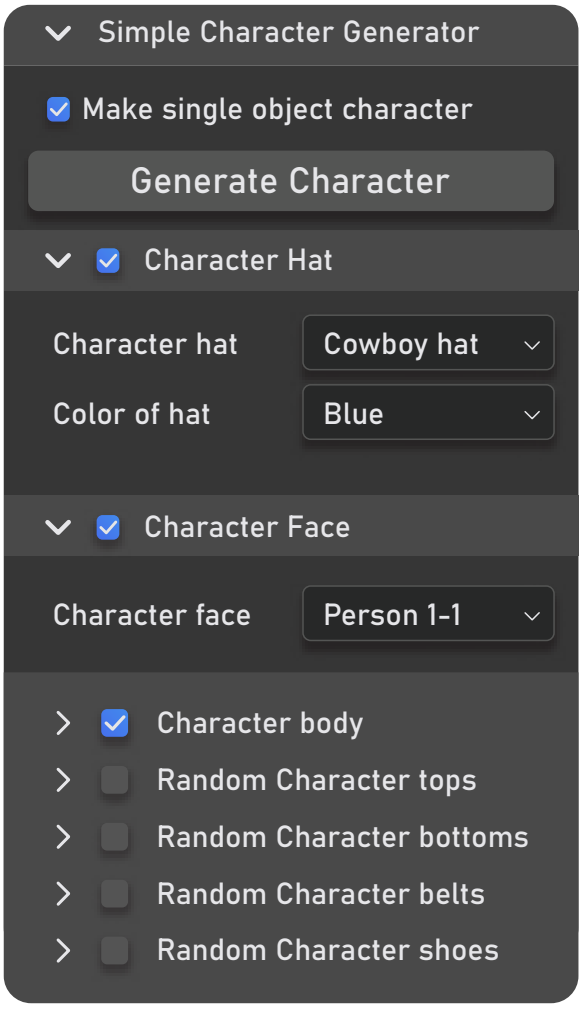
Obr. 3.5: Návrh uživateľského rozhrania pre vygenerovanie spojeného objektu ktorým je možne pohybovať pomocou kostry.

3.2 Simple Character Generator

Je to nástroj, ktorého úlohou je vygenerovať jednoduchý charakter, ktorý je použiteľný v hernom engine. Užívateľ má na výber zo 6 postáv a piatich kategórií oblečenia, kde každá kategória má rôzne typy a farby oblečenia. Užívateľ si tak môže nakonfigurovať postavu aká mu vyhovuje ako je vidieť na obrázku 3.6. V prípade, že si nechce manuálne vyberať, má možnosť pri každej kategórii náhodne vybranej kombinácie z danej kategórie. Následne si môže nechať vygenerovať charakter.

Pri generovaní je možné zapnúť parameter `Make a single mesh`, ktorý po naimportovaní všetkých súčastí spojí charakter do jedného objektu. V opačnom prípade sú všetky časti oddelené, ak by si užívateľ chcel doupraviť jednotlivé časti, z ktorých sa charakter skladá, podľa seba. Pri obidvoch možnostiach je charakter pripravený a rignutý (viacej v sekcii 2.5), aby bolo možné jednotlivými časťami pohybovať a vytvoriť tak potrebné animácie do hry.

Každá jedna časť charakteru je ručne vymodelovaná, pripojená ku kostre a uložená v knižnici s obsahom, ktorý si užívateľ importuje do svojej scény. Názov každej časti sa skladá z predpony, ktorá označuje o akú časť charakteru ide a prípony, ktorá označuje farbu danej časti. Všetky názvy sú v rámci knižnice unikátne, aby sa zabránilo nechcenému importovaniu iných súčastí ako je treba. Každá časť je textúrovaná pomocou textúry, ktorá obsahuje 16 jednoduchých farieb. Tento materiál sa importuje spolu s objektmi v prípade, že scéna užívateľa už tento materiál neobsahuje.



Obr. 3.6: Návrh uživateľského rozhrania pre vygenerovanie jednoduchého charkateru.

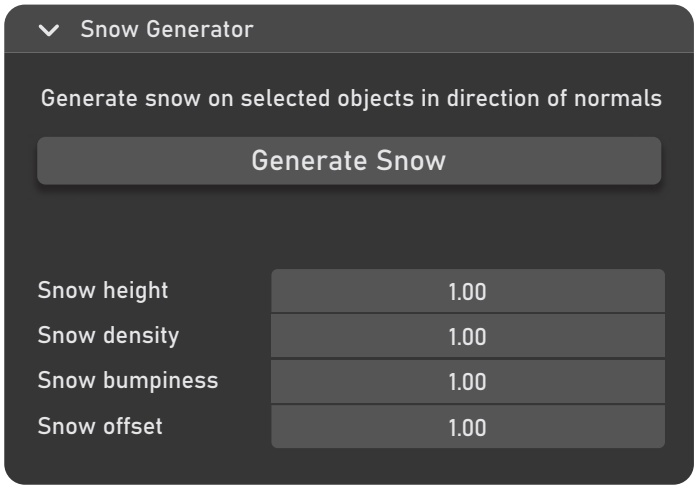
3.3 Snow Generator

Ide o generátor, ktorý vygeneruje sneh na povrchu zvoleného objektu, ktorý si môže následne užívateľ ľubovoľne dynamicky upravovať a vidieť tak aktuálne zmeny. V prípade, že je užívateľ v edit móde objektu, vygeneruje sa mu sneh len zo zvolených plôch. V opačnom prípade sa vygeneruje sneh na povrchu objektu v smere normálov každej plochy, z ktorých sa objekt skladá.

Sneh je vygenerovaný pomocou postupnosti modifikátorov, ktorých hodnoty budú vždy vypočítané podľa parametrov, ktoré si užívateľ zvolil ako je vidieť na obrázku 3.7. Užívateľ môže meniť hodnoty výšky snehu, hrboľatosť snehu, hustotu snehu a posun v osi Z. Pre vytvorenie a modifikovanie výšky snehu bude použitý modifikátor Extrude, ktorý vytiahne plochy v smere ich normálov. Následne bude na rozdelenie plôch použitý modifikátor Subdivision, aby mohol sneh získať hrboľatosť. Na tvorbu hrboľatosti snehu bude použitý modifikátor Displace, kde sa ako textúra použije procedurálne generovaná textúra Perlinovho šumu (viacej v sekcii 2.4). Na zmenu hustoty snehu bude využitá funkcia Blenderu Geometry nodes, ktoré pomocou Voronoi šumu (viacej v sekcii 2.4) a jednoduchej kombinácie uzlov odstránia časť geometrie snehu.

Každá užívateľom zmenená hodnota musí byť vypočítaná a následne musia byť podľa výsledkov nastavené parametre modifikátorov. V opačnom prípade by napevno nastavené hodnoty mohli ovplyvňovať ostatné modifikátory a výsledok by nebol požadovaný. Na zníženie počtu polygónov výslednej geometrie snehu bude použitý modifikátor Decimate, ktorý spojí trojuholníky v určitej blízkosti.

Každý vygenerovaný sneh bude obsahovať procedurálne generovaný materiál, ktorý bude imitovať textúru snehu. Pomocou textúry Perlinovho šumu (viacej v sekcii 2.4) bude dosiahnutá hrboľatosť a detail materiálu. Materiál bude uložený v knižnici a v prípade, že sa nenachádza v scéne užívateľa bude importovaný do scény. Materiál bude vždy automaticky aplikovaný na sneh, ktorý sa vygeneruje.



Obr. 3.7: Návrh uživateľského rozhrania pre vygenerovanie snehu.

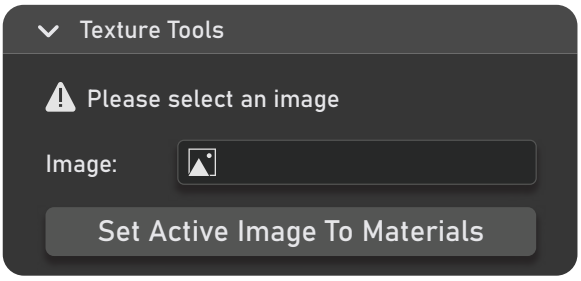
3.4 Texture tools

Ide o sadu nástrojov pre prácu s textúrami. Tieto nástroje sa nenachádzajú v 3D View ako všetky predošlé, ale v UV Editing okne, kde sa pracuje s UV mapami.

3.4.1 Set active image

Funkcia Set Active Image slúži na to, aby si užívateľ pri pečení textúr nastavil aktívny obrázok, na ktorý bude textúra vypečená. Pri tvorbe textúr je potrebné, aby každý materiál mal aktívny obrázok, na ktorý budú finálne dáta zapísané.

V prípade, keď scéna obsahuje veľký počet objektov a každý objekt obsahuje viacero materiálov, nastavenie aktívneho obrázku zaberie naozaj veľa času a veľmi ľahko sa stane, že sa na nejaký materiál aj tak zabudne. Táto funkcia rieši všetky tieto problémy za užívateľa. Užívateľ si len zvolí, aký obrázok chce aby bol aktívny ako je vidieť na návrhu 3.8. Funkcia prejde všetky objekty, ktoré má užívateľ vybrané a v každom objekte prejde všetky materiály. Pre každý materiál nastaví aktívny obrázok, ktorý si užívateľ zvolil. V prípade, že daný materiál už obrázok obsahuje a nemá žiadne pripojenia v shader editore (viacej v sekcii 2.6.4), je tento uzol nastavený ako aktívny. V opačnom prípade je vygenerovaný nový uzol typu image texture, kde je nastavený obrázok, ktorý si užívateľ zvolil.



Obr. 3.8: Návrh uživateľského rozhrania pre nastavenie aktívneho obrázku.

3.4.2 Texel density tools

V prípade, že užívateľ tvorí textúry na objekty do hry, chce, aby objekty s rovnakou viditeľnosťou na kamere mali rovnakú hustotu texelov (viacej v sekcii 2.3.3). Vypočítať hustotu texelov manuálne je takmer nemožné. Funkcia Texel density tools tieto hodnoty vypočíta za užívateľa.

Užívateľ si vyberie objekty a následne si v editmóde vyberie plochy, pre ktoré chce vypočítať hustotu texelov. Pre každú plochu je následne vypočítaná hustota pomocou vzorca:

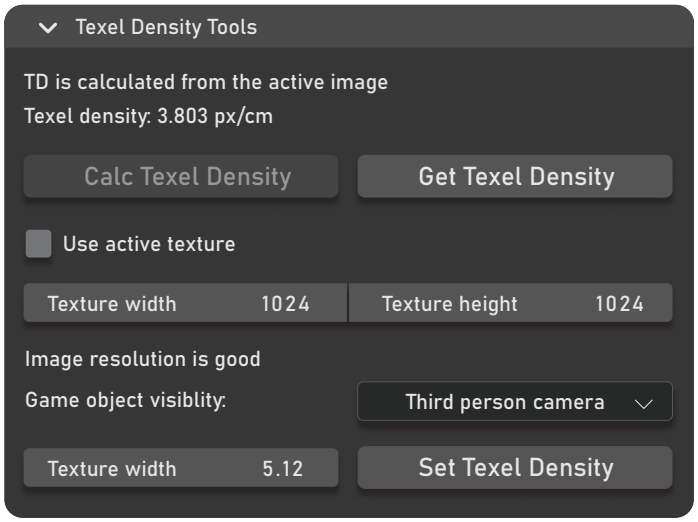
$$A_{UV\ f} \cdot Res / A_{s\ f} \tag{3.1}$$

Kde jednotlivé parametré znamenajú:
A_{fuv} - plocha UV mapy označenej steny
Res - rozlíšenie textúry
A_{fs} - plocha označenej steny

Pre výpočet veľkosti plochy UV mapy je použitý shoelace algoritmus (viacej v sekcii 2.7). Následne je táto hodnota odmocnená, aby sa získal rozmer počet pixelov / centimeter. V prípade, že si užívateľ vyberie viacero stien, je vypočítaný aritmetický priemer zo súčtu hodnôt texelov všetkých plôch.

V základnom nastavení je hustota pixelov vypočítaná pre aktuálny obrázok otvorený v UV Editore. Užívateľ má možnosť vypnúť parameter `Use active texture` a následne tak zadať vlastnú výšku a šírku obrázku, pre ktorý je následne hustota vypočítaná. Užívateľ si vyberie o aký druh viditeľnosti objektu na kamere ide a následne je informovaný, či je výsledná hustota dostatočná, alebo by mal upraviť rozlíšenie zvolenej textúry ako je vidieť na obrázku 3.9. Nemusí mať preto znalosť o tom, čo je vhodné pre aký objekt.

V prípade, že má užívateľ informáciu o požadovanej hustote, môže si ju nastaviť do okna s hodnotou Texel density a následne tak upraviť zvolené plochy, aby dosahovali požadovanú hodnotu. V prípade, že užívateľ nevie aký druh hodnoty potrebuje pre daný objekt, výber typu viditeľnosti objektu na kamere mu po zvolení hodnoty automaticky nastaví odporúčanú hodnotu pre daný typ objektu. V prípade, že chce užívateľ zmeniť hustotu texelov aj na iných objektoch a má už vypočítanú hodnotu, môže si túto hodnotu preniesť do poľa s cieľovou hodnotou pomocou tlačidla `Get Texel Density` (vypočítaná hodnota sa zobrazuje len na 3 desatinné hodnoty, avšak v pamäti počítača bude uložený celý desatinný rozvoj čísla, a preto sa neodporúča hodnotu kopírovať).



Obr. 3.9: Návrh uživateľského rozhrania pre prácu s hustotou pixelov.

Kapitola 4

Implementácia

Funkcionalita add-onu je implementovaná v jazyku Python pomocou modulov, ktoré sú dostupné v Blenderi a pomocných modulov na výpočet. Základom Blenderu je modul bpy. Každá funkcia add-onu sa skladá z užívateľského rozhrania, ktoré je vytvorené pomocou triedy typu Panel, operátoru, ktorý je vytvorený pomocou typu Operátor a vlastností, ktoré sú vytvorené na základe typu PropertyGroup. Všetky typy je možné nájsť v module bpy.types.

Každý panel je vykreslený pomocou metódy draw(). Panel je rozdelený na riadky a stĺpce, ktoré obsahujú označenia, parametre pre danú funkciu a tlačidlá odkazujúce sa na operátor, ktorý je zavolaný v prípade kliknutia na tlačidlo. Operátory obsahujú ID, pomocou ktorého tlačidlo vie, ktorý operátor má zavolať. Operácie sú vykonané pomocou metódy execute a musia končiť návratovou hodnotou 'FINISHED' alebo 'CANCELLED'. Pred vykonaním každej operácie sa pomocou metody poll() kontroluje podmienka, či daná operácia môže byť vykonaná. Ak metóda vráti pravdivostnú hodnotu True, môže sa operácia vykonať. V opačnom prípade je tlačidlo zablokované a užívateľ naň nemôže kliknúť.

Pre vstup užívateľských parametrov sú použité hodnoty, ktoré využívajú Blenderom preddefinované dátové typy. Ku každému parametru je možné pristúpiť pomocou pointeru, ktorý odkazuje na parametre vytvorené pre potrebu add-onu.

Aby bol zaistený prístup k operátorom, parametrom a panelom tejto práce v Blenderi, každá trieda je pri spustení Blenderu automaticky zaregistrovaná a pri vypnutí automaticky odregistrovaná. Pre prístup k parametrom add-onu je vytvorený pointer v scéne, ktorý odkazuje na tieto parametre. K tomuto pointru sa dá pristúpiť kedykoľvek počas chodu programu Blender pomocou bpy.types.Scene.lka, kde lka značí odkaz na parametre.

4.1 Mesh Tools

Rozhranie pre nástroje Mesh tools obsahuje hlavný panel obaľujúci ostatné panely, ktoré obsahujú parametre ku konkrétnym funkcionalitám. Panel nástrojov Mesh tools je lokalizovaný v 3D View.

4.1.1 Generate LOD

Pre vykonanie tejto operácie musia zvolené objekty obsahovať geometriu a užívateľ sa musí nachádzať v objektovom móde. V opačnom prípade je táto operácia zablokovaná. Následne sa získa pomer, podľa ktorého sa upraví geometria objektu. Pomer je uložený v kolekcii parametrov ako celé číslo s minimálnou hodnotou 1 a maximálnou hodnotou 4, pretože úroveň 0 je vyhradená pre objekt v blízkosti kamery v maximálnej kvalite. Viacej ako 4 úrovne je zbytočné generovať, pretože vo vyšších úrovniach už objekty obsahujú minimálnu potrebnú kvalitu na zobrazenie. Pre vyššie úrovne detailov je zvolený nižší pomer, takže sa vo výslednej geometrii spojí viacero trojuholníkov dokopy. Následne je každý objekt pomocou cyklu cez všetky označené objekty v scéne postupne zduplikovaný. Pri každej duplikácii sú všetky objekty odznačené a až potom zavolaný operátor na duplikovanie. Inak by boli duplikované aj objekty, ktoré sú označené, ale práve na nich neprebiehajú operácie.

Pre každý zduplikovaný objekt je nastavené meno pôvodného objektu a prípona informujúca o akú úroveň detailov sa jedná. Následne je vypočítaný počet trojuholníkov, z ktorých sa objekt skladá, aby v prípade nízkeho počtu nedošlo k deštrukcii modelu, ktorý nemôže byť použitý. Počet trojuholníkov je vypočítaný sumou počtov bodov, z ktorých sa skladá každý mnohouholník tvoriaci mesh objektu mínus 2 (Počet trojuholníkov v každom mnohouholníku je o dva menší ako počet bodov). Pokiaľ je počet trojuholníkov menší ako 1000 (pri nižšiom počte trojuhoľníkov, aktuálne nastavenia kompletne deformujú objekt), výsledný pomer v tretej a štvrtej úrovni detailov je nastavený na vyššiu hodnotu. Následne je objekt nastavený ako aktívny a aplikovaný modifikátor, aby sa zmeny previedli permanentne. Tento postup je vykonaný pre každý jeden objekt, ktorý je označený.

4.1.2 Bounding Box

Funkcia vygeneruje pre označené objekty jeden zo 4 zvolených typov bounding boxov (viacej v sekcii 2.2.1). Pre vykonanie tejto funkcie je potrebné, aby mal užívateľ označený aspoň jeden objekt obsahujúci geometriu a nachádzal sa v objektovom móde. Ako prvý je získaný typ bounding boxu, ktorý si užívateľ vybral. Následne je pre každý objekt skonštruovaný box podľa výberu.

Pre box typu AABB je použitá vlastnosť objektu v Blenderi, ktorá vie pozície jednotlivých bodov boxu, z ktorých je následne skonštruovaný výsledný mesh pre finálny box. V prípade boxu typu OBB už nie je možné použiť túto vlastnosť objektu a je potrebné jednotlivé body vypočítať manuálne [15]. Pre výpočet sa zaistia všetky body, z ktorých sa objekt skladá. Následne je pomocou kovariácie a vlastných vektorov zistený smer, ktorým je objekt lokálne otočený. Ďalej sú všetky body objektu premietnuté do vypočítaného vektorového priestoru. Z týchto bodov sú zistené maximálne a minimálne pozície bodov pre každú os. Rozdiel medzi týmito bodmi udáva dĺžku jednej hrany boxu. Následne je zistená vzdialenosť bodu od stredu boxu a vypočítané jednotlivé body tvoriace daný box. Ako posledné je vrátená orientácia boxu do pôvodného priestoru, z ktorého je vytvorený finálny objektovo orientovaný box.

V prípade bounding sphere je použitý modul bmesh na skonštruovanie gule. Pre skonštruovanie gule je potrebné poznať priemer a stredový bod. Tieto body sú získané z objektu, pre ktorý je box vytváraný. Stredový bod je vypočítaný ako priemer lokácií všetkých bodov, z ktorých sa objekt skladá a priemer je vypočítaný ako 2 krát vzdialenosť najvzdialenejšieho bodu od stredu.

Pre convex hull je taktiež využitý modul bmesh, ktorý vytvorí z kópie bodov objektu convex hull. Následne je potrebné odstrániť nepoužitú geometriu a vnútornú geometriu, ktorá ostala po vytvorení convex hull objektu.

Pre všetky druhy vygenerovaných boxov je následne nastavená pozícia, rotácia a veľkosť, aby sa zhodovali s objektom, pre ktoré boli vygenerované. Pre nastavenie rotácie je mesh boxu zrotovaný pomocou maticových operácií v jeho aktuálnej pozícii invertovanými hodnotami uhlov, ktoré sú nastavené ako výsledné, aby sa zmenila globálna rotácia objektu bez zmeny lokálnej rotácie. Pre nastavenie veľkosti je veľkosť boxu zmenená na 1/cieľová veľkosť, následne je aplikovaná veľkosť a potom je veľkosť zmenená na požadovanú.

4.1.3 Explode Objects

Užívateľ si nastaví hodnoty v akej vzdialenosti chce, aby sa objekty od seba nachádzali. Na to, aby táto operácia mohla byť vykonaná sa kontroluje podmienka, či sa užívateľ nachádza v objektovom móde. Následne sú získané hodnoty od užívateľa a nastavené medzery pre X-ovú a Y-ovú os. Pozícia prvého objektu v X-ovej osi je vypočítaná ako:

$$N_{obj} \cdot Space/2 \cdot -1 \tag{4.1}$$

Kde jednotlivé parametré znamenajú:
N_{obj} - počet objektov v riadku
$Space$ - medzera medzi objektami

Aby sa objekty náchadzali rovnomerne na oboch stranách Y-ovej osi. Počiatočná pozícia pre objekty na Y-ovej osi je 0, pretože ide o prvý rad objektov. Následne je pre každý označený objekt postupne nastavená pozícia, na ktorej sa bude nachádzať. Pre každý objekt v rade je k jeho aktuálnej pozícii pripočítaná hodnota medzery na X-ovej osi, ktorá je nastavená ako výsledná pozícia pre nasledujúci objekt.

V prípade, že sa v rade nachádza počet objektov koľko si užívateľ zvolil do jedného radu, k Y-ovej pozícii je pripočítaná hodnota medzery medzi objektmi na osi Y, hodnota X-ovej pozície je vynulovaná a pokračuje sa odznovu s novým posunom v osi Y. Nové pozície sú nastavené len pre objekty, ktoré obsahujú geometriu. Objekty ostatných typov sú ignorované, pretože nebudú ovplyvňovať textúry geometrie, pre ktorú chce užívateľ vytvoriť textúry.

Na ukladanie pozície je vytvorená pomocná trieda obsahujúca zoznam, kde sú uložené pozície objektov. Operácie Save, Load a Clear následne dedia od tejto triedy, aby zabezpečili globálny prístup k tomuto zoznamu. Ak chce užívateľ uložiť pozície, pre každý označený objekt sa skontroluje, či sa už objekt v zozname nenachádza, aby si užívateľ neprepísal pozície. Ako kľúč pre každú pozíciu je použitý názov geometrie objektu (názov objektu sa príliš často mení, takže by sa stratil objekt, pre ktorý je informácia uložená). Pri načítaní objektov sa podľa mena geometrie objektu získa pozícia zo zoznamu a následne sa nastaví pre daný objekt. V prípade, že chce užívateľ vymazať hodnotu, sa pre všetky označené objekty nájde hodnota v zozname, ktorá sa vymaže. Následne tak užívateľ môže znovu uložiť pozíciu daného objektu.

4.1.4 Load On Empty

Je to nástroj, ktorý uľahčí načítavanie pozícií rôznych objektov na prázdne body. Užívateľ musí mať označený aspoň jeden objekt typu EMPTY. Následne si užívateľ zvolí `Target object`, ktorý sa načíta na pozíciu prázdneho bodu. Po zvolení objektu v ponuke je prázdnemu bodu pridaná vlastnosť, ktorá drží ukazovateľ na zvolený objekt.

Pre zaistenie aktualizácie cieľového objektu každého prázdneho bodu je použitý callback, ktorý pri každej zmene v scéne kontroluje, či ide o prázdny objekt, ktorý má už priradený cieľový objekt a následne túto informáciu aktualizuje pre užívateľa v paneli s nástrojom. Následne je pre každý označený prázdny bod načítaný príslušný objekt. V prípade, že nejde o prázdny objekt alebo prázdny objekt nemá priradený žiaden cieľový objekt, je tento objekt preskočený a užívateľ je informovaný o chýbajúcom cieli.

Každý objekt, ktorý je načítaný je kópiou inštancie originálneho objektu, ktorá neobsahuje vlastné dáta, čo zabezpečuje, že užívateľ môže objekt modifikovať a všetky zmeny sú prevedené na všetky načítané objekty. Zároveň je týmto spôsobom ušetrené miesto v pamäti, keďže nejde o samostatné objekty.

Objekty sú nastavené ako potomkovia prázdnych bodov, takže zdieľajú veľkosť, rotáciu a pozíciu prázdneho bodu. Každý prázdny bod má následne priradenú vlastnosť, ktorá nesie odkaz na kópiu objektu, aby bolo možné rozlíšiť originál od duplikátu.

Užívateľ má možnosť použiť parameter `Load same object to all empties`, ktorá odignoruje cieľové objekty individuálnych prázdnych bodov a na každý označený bod načíta cieľový objekt aktuálne zvoleného prázdneho bodu. Na každý prázdny bod môže byť načítaných viacero objektov. Funkcia unload object vymaže všetky kópie objektov, ktoré sú načítané na každom prázdnom bode a odstráni všetky vlastnosti objektu obsahujúce cieľový objekt a zoznam načítaných objektov.

Funkcia Apply konvertuje všetky objekty na objekty obsahujúce samostatné dáta, takže následne môžu byť ľubovoľne modifikované. Ak si užívateľ zvolil parameter
`On unload unparent objects`, tak sú všetky objekty oddelené od empty objektu a získajú vlastnú pozíciu, rotáciu a veľkosť v scéne.

4.1.5 Convert To Skeleton

Make skeleton je funkcia, ktorá prevedie užívateľovi všetky označené objekty na jeden objekt obsahujúci kostru, ktorá dokáže pohybovať predtým samostatnými objektmi. Pre vykonanie tejto operácie je potrebné, aby mal užívateľ označený aspoň jeden objekt obsahujúci geometriu, nachádzal sa v objektovom móde a neobsahoval objektovú vlastnosť `is_skeleton`, ktorá označuje už spojený objekt.

Pri vykonávaní funkcie sa ako prvé uložia dáta o pozícii, rotácii a veľkosti objektu do pomocného zoznamu, ktorý je neskôr uložený ako vlastnosť objektu. Následne je pre každý objekt vytvorená skupina bodov, ktorá rozlišuje v spojenom objekte pohyby jednotlivých častí. Každá skupina má predponu `arm_` pre označenie, že bola vytvorená programom a odlíšila sa od ostatných skupín bodov, ktoré môže objekt obsahovať. Následne sú všetky objekty spojené do jedného, pridané vlastnosti objektu `is_skeleton`, ktorý označuje, že ide o programom vygenerovanú štruktúru `skeleton_dat` a obsahujúce informácie o pôvodných objektoch a `single_armature_prop` informujúca, či ide o objekt obsahujúci jednu kostru. Každý parameter je chránený proti prepisu užívateľa prepísaním pôvodnej get funkcie pre danú vlastnosť.

Po spojení objektov do jedného je pre jednotlivé, na začiatku oddelené objekty, vygenerovaná kosť v počiatočnom bode každého objektu s jeho rotáciou a veľkosťou.

Každá kosť je pomenovaná ako skupina bodov pohybujúca daným objektom. V Blenderi sú jednotlivé kosti namapované na skupiny bodov zhodujúce sa s názvom danej kosti.

Po vygenerovaní každej kosti je objektu priradený modifikátor Armature. V prípade, že si užívateľ vypol možnosť `Make single armature` je objektu pridaný modifikátor pre každú kosť zvlášť. V opačnom prípade sú všetky kosti spojené do jednej výslednej kostry.

Convert to mesh je funkcia, ktorá vracia v ľubovoľnom bode tento proces do pôvodného stavu, keď boli objekty oddelené, okrem niektorých informácií, ako napríklad modifikátory, ktoré objekty obsahovali. Na vykonanie tohto procesu je potrebné, aby objekt obsahoval vlastnosť `is_skeleton` a `skeleton_data`. Podľa skupín bodov sú následne oddelené všetky objekty a premenované na `separated_object`. V opačnom prípade by im Blender priradil názov, ktorý by si sám zvolil.

Z každého objektu sú odstránené skupiny bodov, ktoré boli potrebné pre kostru, všetky modifikátory pre pohyb kostrou a vlastnosti objektu, ktoré boli nosičom dát. Každému objektu je nastavený pôvodný názov, pozícia, rotácia a veľkosť. Pre nastavenie rotácie je mesh objektu pomocou maticových operácií zrotovaný v jeho aktuálnej pozícii invertovanými hodnotami uhlov, ktoré sú nastavené ako výsledné, aby sa zmenila globálna rotácia objektu bez zmeny lokálnej rotácie. Všetky kostry, ktoré objekt obsahoval sú odstránené zo scény.

4.2 Simple Character Generator

Pre užívateľa je pripravený charakter, ktorý si môže ľubovoľne nakonfigurovať z ponuky častí, z ktorých sa charakter skladá. Každá časť má pripravený skinned mesh (viac v sekcii 2.5), pomocou ktorého je charakter rignutý (viac v sekcii 2.5) pre potrebu pohybu jednotlivých častí postavy. Postava, oblečenie a doplnky obsahujú materiál, ktorý sa skladá z jednoduchej textúry obsahujúcej 16 farieb. Pre každý doplnok a oblečenie je pripravených 7 rôznych farieb. Názov každej časti charakteru a oblečenia sa skladá z predpony, ktorá označuje typ a prípony, ktorá označuje farbu. Na základe tohto názvu sa následne pristupuje k objektom pri importovaní do scény. Všetky časti charakteru, materiál a pripravená kostra sú uložené v knižnici add-onu. Celkový počet rozdielnych kombinácií, ktoré je možné vytvoriť z charakteru, doplnkov a farieb je 441 183 750, bez farieb je to 52500.

Aby užívateľ mohol importovať charakter, musí sa nachádzať v objektovom móde. Následne si zvolí konfiguráciu požadovaného charakteru. Pre každú časť má možnosť zvoliť si manuálne alebo nechať náhodne vybrať kombináciu doplnku a farby. Pre náhodné generovanie je využitý modul random, ktorý vyberá náhodne zo zoznamu doplnku a zoznamu farby. Výhoda je, že modul nepoužíva true random náhodný výber, vďaka čomu sa negenerujú maximálne rozdielne farebné kombinácie.

Pre importovanie objektov je následne získaná konfigurácia, ktorú si užívateľ zvolil. Následne je každá časť charakteru a kostra postupne importovaná do scény a uložený odkaz na daný objekt, aby bol zaistený prístup pre ďalšie použitie. Každý objekt je importovaný podľa jedinečného názvu v knižnici. V prípade, že v scéne neexistuje materiál pre charakter, je importovaný aj ten. Keďže Blender nepozná cestu k add-onu, bolo potrebné využiť modul pathlib na zaistenie absolútnej cesty ku knižnici pre každý operačný systém.

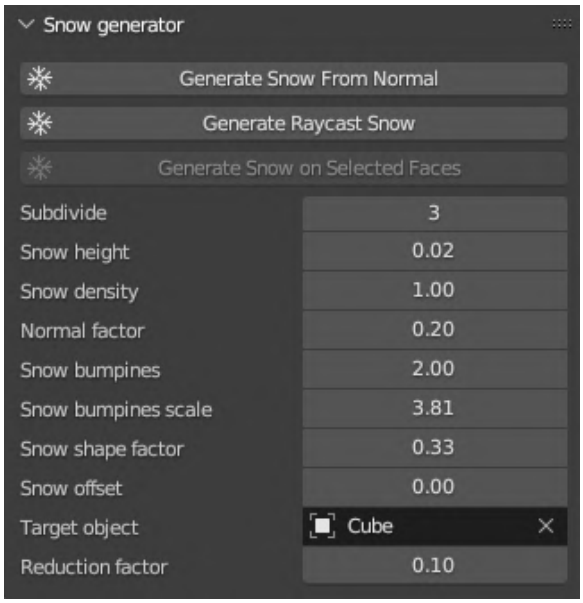
Po dokončení importovania každej časti do scény je pre každý diel charakteru pridaný modifikátor Armature, ktorý prepojí charakter s kostrou. Základne sú všetky časti charakteru oddelené, aby si ich mohol v prípade potreby užívateľ upraviť. Ak by ale nepotreboval zasahovať do modelu, má možnosť zapnúť parameter `Make single object character`, ktorý mu pripraví výsledný charakter ako jeden objekt.

4.3 Snow Generator

Implementácia funkcionality na generovanie snehu je vo finálnej verzii add-onu odlišná od návrhu. Namiesto prístupu, kde sa generoval sneh pomocou modifikátorov v smere normálov, bol plne využitý modifikátor GeometryNodes, čo vytvorilo nové možnosti a prístupy. Vo finálnej verzii má užívateľ 2 možné prístupy: Generate Snow From Normal a Generate Raycast Snow. Obidve metódy sú plne modifikovateľné s výhodou, že sa v reálnom čase prispôsobujú objektu a jeho rotácii. Užívateľ má z panelu add-onu prístup k hodnotám, ktoré upravujú rôzne vlastnosti vygenerovaného snehu. Pre prepojenie panelu s modifikátorom je použitý callback, aby sa spätne aktualizovali parametre v paneli.

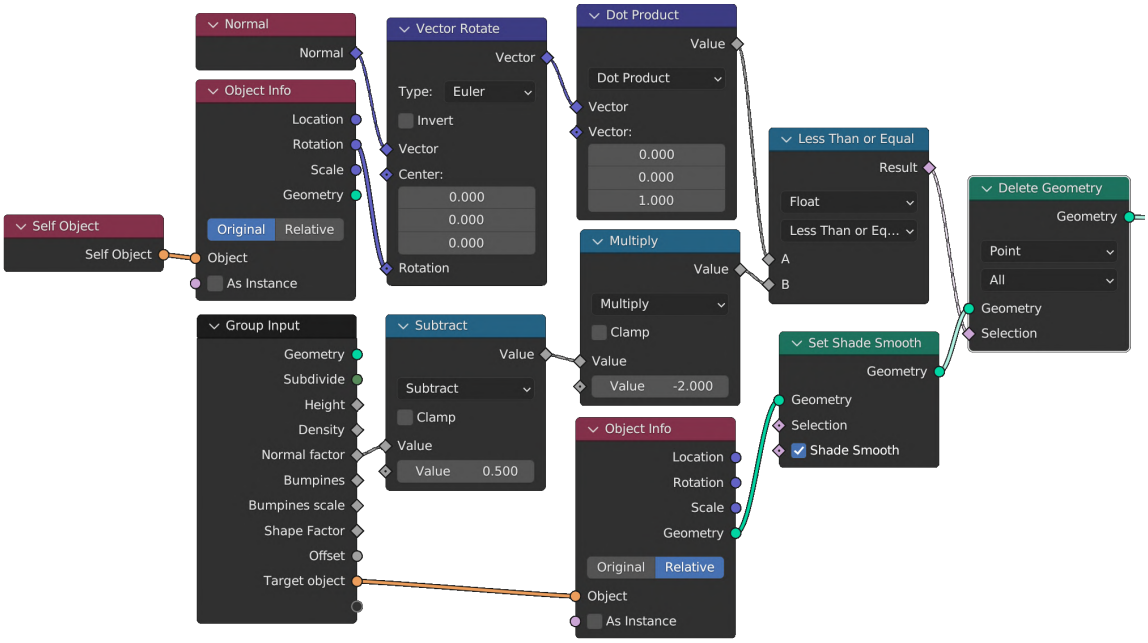
4.3.1 Generate Snow From Normal

Tento prístup dáva užívateľovi možnosť vygenerovať sneh na povrchu objektu v smere plôch, ako je vidno na obrázku 4.6. Pri rotácii objektu ostane sneh vždy na vrchných stranách objektu a prispôsobí sa zmenám. Po vygenerovaní sa užívateľovi zobrazí ponuka možností hodnôt, pomocou ktorých môže modifikovať vlastnosti snehu, ako je vidieť na obrázku 4.1. Pre vytvorenie snehu je modifikátor aplikovaný na primitívnu plochu, z ktorej je sneh vygenerovaný na cieľovom objekte.



Obr. 4.1: Parametre pre úpravu snehu vygenerovaného pomocou metódy z normálov. Subdivide - počet úrovní rozdelenia geometrie, height - výška snehu, density - hustota snehu, normal factor - veľkosť tolerancie normál stien, bumpiness - sila hrboľatosti, bumpiness scale - veľkosť hrboľov, shape factor - ako veľmi bude geometria snehu prispôsobená tvaru plochy, offset - posuv v osi Z, target - objekt, na ktorom sa vygeneruje sneh, reduction factor - tolerancia pri redukcii geometrie snehu.

Ako prvé sú z cieľového objektu získané plochy, z ktorých smerov bude následne vygenerovaný sneh. Pozri obrázok 4.2. Z celkovej geometrie sú odstránené plochy, ktorých normály majú menšiu hodnotu ako faktor určujúci rozsah smeru normál jednotlivých plôch. Do úvahy je braná rotácia objektu, z ktorého je vygenerovaný sneh, aby v prípade zrotovania tohto objektu nebola ovplyvnená rotácia výsledného snehu.

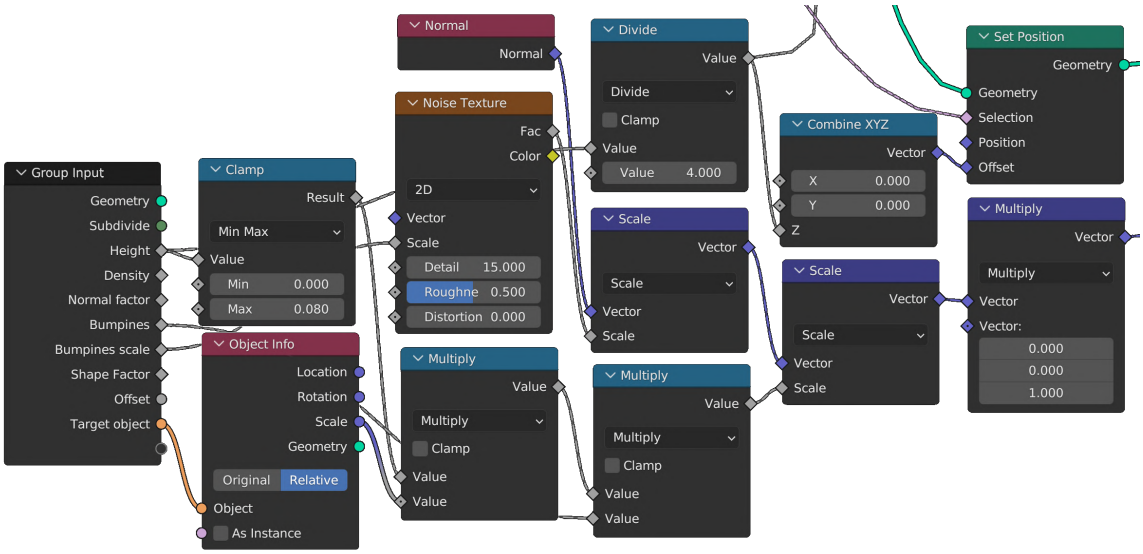


Obr. 4.2: Strom uzlov, pomocou ktorých sú získané plochy, pre ktoré je vygenerovaný sneh. Výsledkom je kópia geometrie plôch, ktoré sú ďalej spracované. Vrchná časť uzlov zabezpečuje rovnakú rotáciu aj v prípade, keď je objekt, z ktorého je generovaný sneh zrotovaný.

Následne sú plochy, ktoré ostali po odstránení nepotrebnej geometrie rozdelené pomocou subdivision na jemnejšiu geometriu, ktorá je prevedená na trojuholníky. Pomocou Voronoi textúry sú odstránené plochy podľa rozsahu, ktorý si zvolil užívateľ. Pozri obrázok 4.3. Výsledná geometria je vytiahnutá smerom hore, prevedená znovu na trojuholníky a ešte raz rozdelená pomocou subdivision.

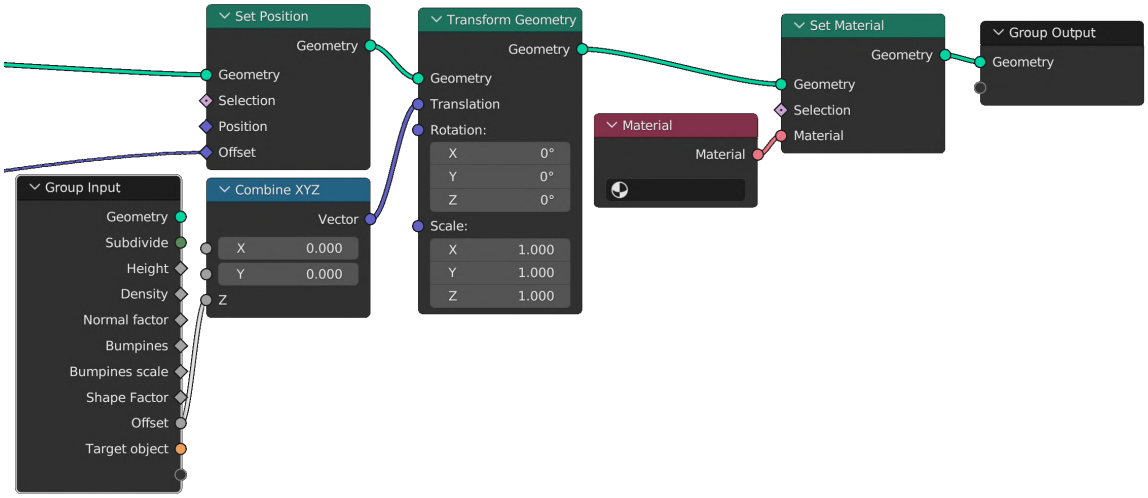
Obr. 4.3: Strom uzlov, pomocou ktorých je modifikovaná hustota snehu. Geometriu je potrebné rozdeliť, aby z nej mohli byť odstránené plochy, ktoré zodpovedajú čiernym miestam na textúre šumu. Uzol Extrude mesh je kontrolovaný pomocou parametru výšky, ktorého hodnota v tomto prípade nesmie presiahnuť 0.04, inak by sa zakrivené plochy vysúvali aj do bokov, čo by nepôsobilo realisticky.

Z geometrie, ktorá bola vytiahnutá, je získaná pozícia vrchných plôch, ktorých pozícia v osi Z je modifikovaná ako výška snehu, ktorú si užívateľ zvolil. Následne je pomocou textúry šumu pridaná hrboľatosť snehu v smere normálov jednotlivých plôch, obmedzených na menenie pozície iba v Z-ovej. Pozri obrázok 4.4. Hrboľatosť je násobená veľkosťou objektu pre prípad, že by objekt obsahoval nejednotnú veľkosť. Pomocou parametrov môže užívateľ kontrolovať silu hrboľatosti a veľkosť hrboľov.



Obr. 4.4: Strom uzlov pridávajúci vygenerovanému snehu hrboľatosť pomocou noise textúry a modifikujúci výšku objektu. Z postupnosti uzlov je vidno že nieje možné použiť vytiahnutie geoemtrie na kontrolu výšky, ale je potrbné modfikovať pozíciu v Z-ovej osi pre vrchné plochy vytiahnutej geoemtrie. Taktiež je vidieť že veľkosť textúry šumu je násobená veľkosťou objektu, aby v prípade nejdenotnej veľkosti objektu, vygenerovaný sneh, zachoval veľkosť hrboľov

Výslednej geometrii je v prípade potreby nastavený užívateľom zvolený posun v osi Z. Pozri obrázok 4.5. Ako posledný je objektu priradený procedurálny materiál, ktorý imituje povrch snehu, ako je vidno na obrázku 4.15.

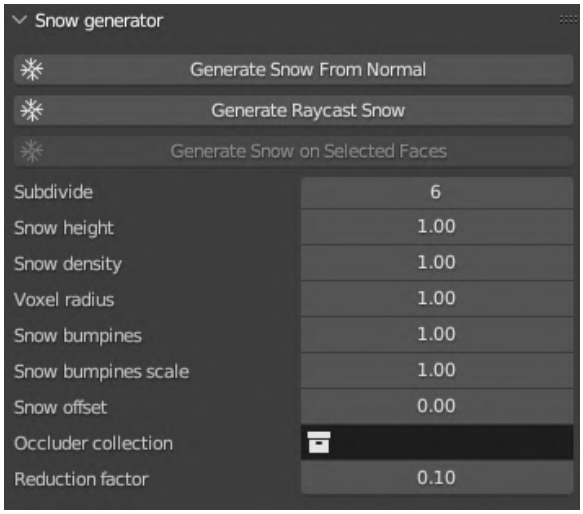


Obr. 4.5: Strom uzlov modifikujúci posun v Z-ovej osi a výsledné priradenie materiálu. Uzol Materiál nemá priradenú žiadnu hodnotu, pretože táto hodnota je nastavená až po importovaní modifikátora. V opačnom prípade by sa materiál v scéne duplikoval.

Obr. 4.6: Príklad možného výsledku vygenerovaného pomocou metódy v smere normálov plôch. Z obrázku je vidno, že sneh sa vygeneroval len na plochách smerujúcich hore.

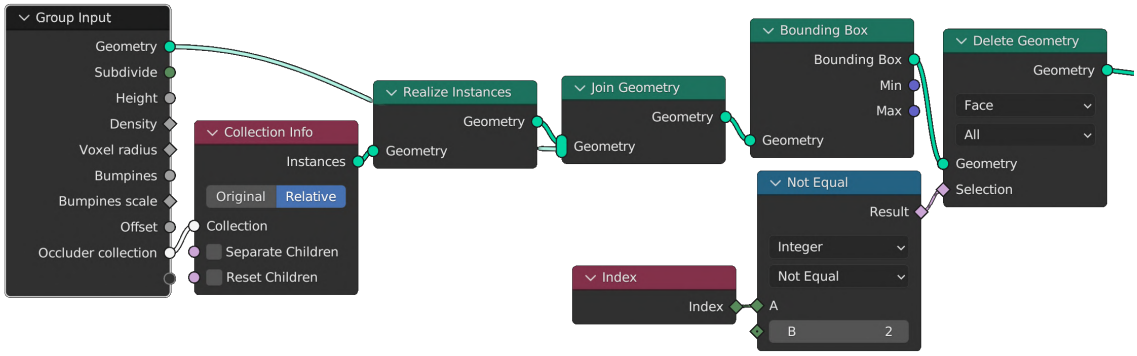
4.3.2 Generate Raycast Snow

Tento prístup, na rozdiel od predošlého, využíva Raycast. Raycast vysiela lúče smerom na objekty a v mieste zásahu objektu lúč končí. Vďaka tomuto je možné odsledovať objekty, ktoré zakrývajú iné časti a vytvoriť tak na ich povrchu sneh, ktorý nebude ďalej pokračovať. Užívateľ si môže ľubovoľne modifikovať parametre, ktoré je vidno na obrázku 4.7.



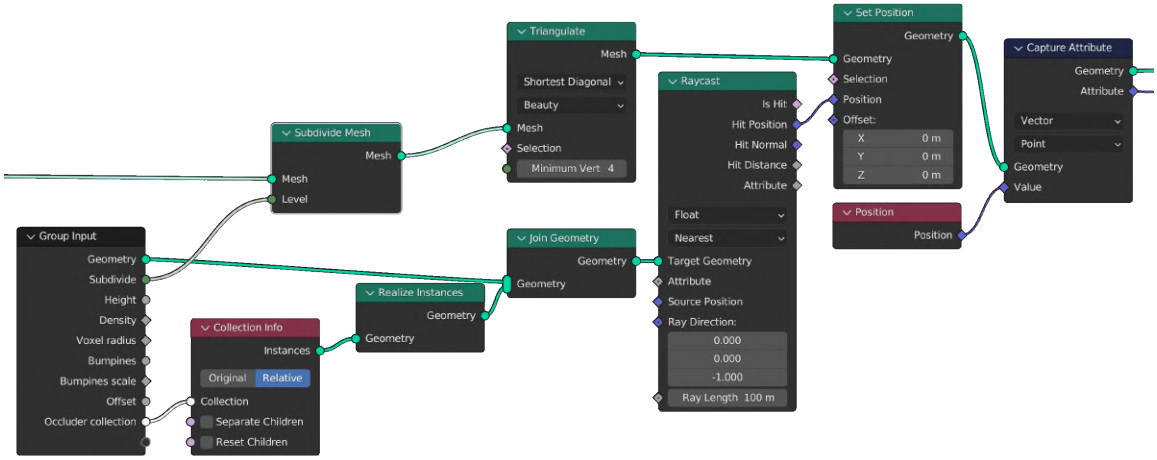
Obr. 4.7: Parametre pre úpravu snehu vygenerovaného pomocou metódy vrhania lúčov. Subdivide - počet úrovní rozdelenia geometrie, height - výška snehu, density - hustota snehu, voxel radius - veľkosť polomeru oblasti, v ktorej sa budú generovať voxely, bumpiness - sila hrboľatosti, bumpiness scale - veľkosť hrboľov, offset - posuv v osi Z, occluder collection - kolekcia objektov, ktoré budú prekrývať sneh, reduction factor - tolerancia pri redukcii geometrie snehu.

Ako prvý je vygenerovaný bounding box z objektu, na ktorý je vygenerovaný sneh a kolekcie, v ktorej sa nachádzajú objekty zakrývajúce objekt. Pozri obrázok 4.8. Z bounding boxu je následne odstránená všetka geometria okrem vrchnej plochy.



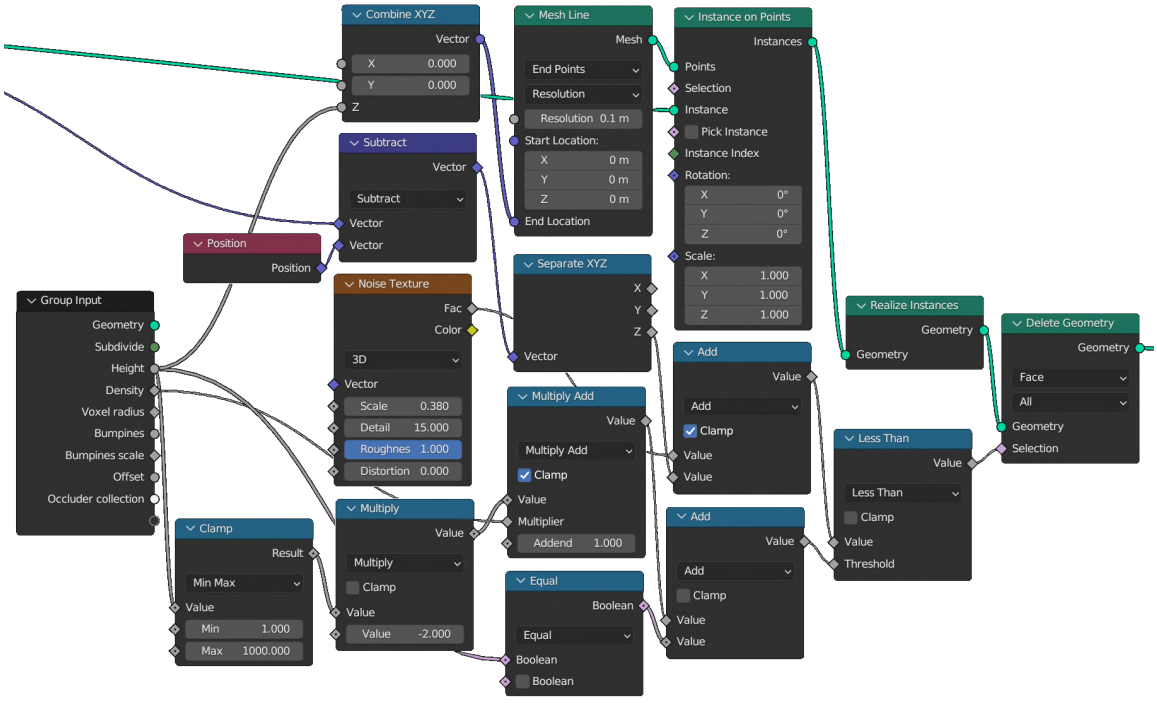
Obr. 4.8: Strom uzlov generujúci plochu nad objektom a kolekciu objektov, ktoré zakrývajú objekt, pre ktorý je vygenerovaný sneh. Plocha je vygenerovaná z bounding boxu, ktorého všetky plochy okrem vrchnej sú odstránené.

Následne je vygenerovaná plocha rozdelená na jemnejšiu geometriu, z ktorej sú vyslané lúče na cieľový objekt a kolekciu. Lúče, ktoré zasiahli objekty sú zachytené a každá plocha geometrie je posunutá na miesto dopadu lúča. Pozri obrázok 4.9.



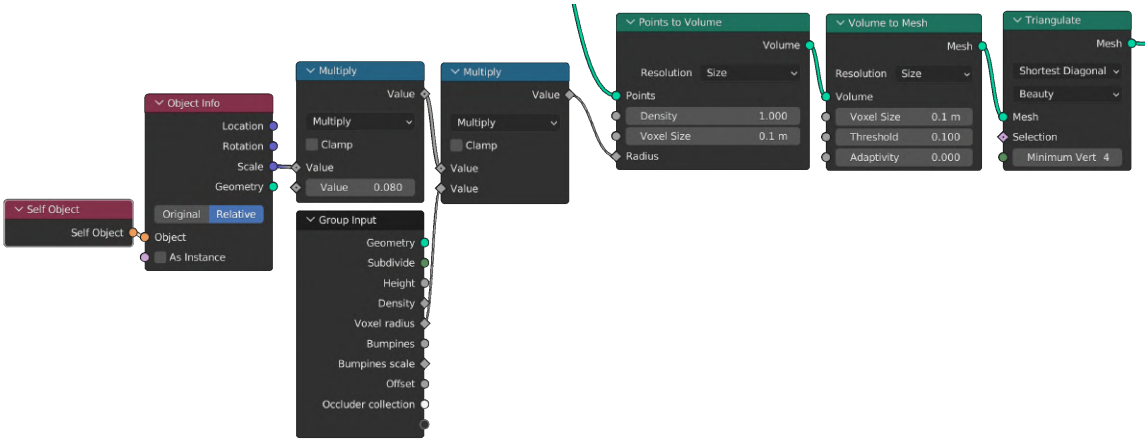
Obr. 4.9: Strom uzlov, ktorý vykonáva raycast a zachytáva pozíciu dopadnutých lúčov. Geometrie plochy je najskôr potrebné rozdeliť, aby sme mali viac detailov pre generovanie snehu.

Pre kontrolu výšky sú plochy duplikované a posunuté v osi Z. Pre kontrolu hustoty je použitá 3D textúra šumu, ktorá odstraňuje plochy podľa čiernych miest na textúre. Pozri obrázok 4.10. V prípade, že je sneh príliš vysoký, tak je pri znižovaní hustoty znižovaná aj celková výška snehu.



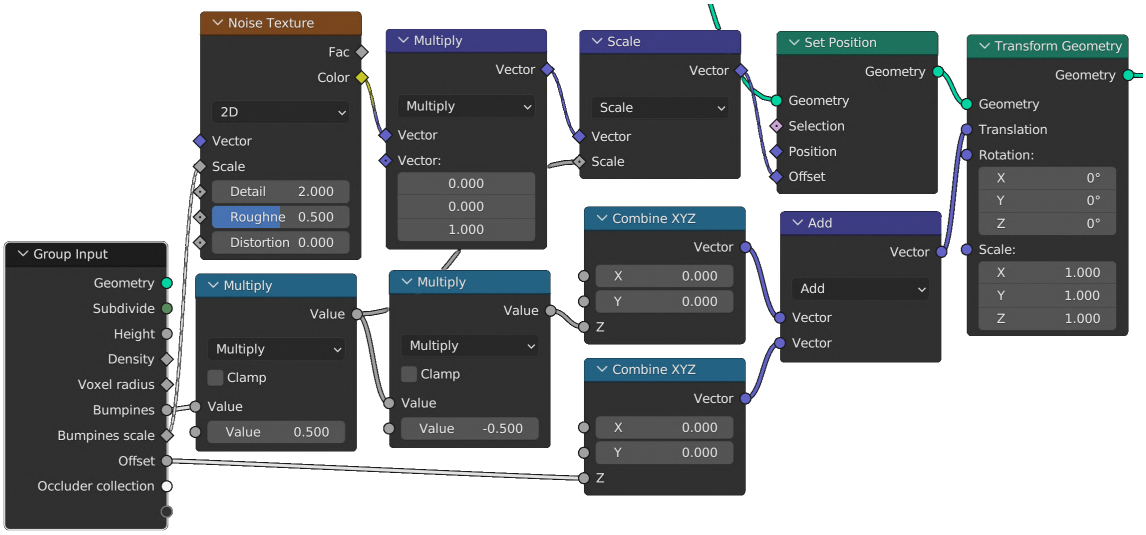
Obr. 4.10: Strom uzlov kontrolujúci výšku a hustotu vygenerovaného snehu. Z dopadnutých lúčov je zachytená plocha, ktorá kopíruje miesta dopadu lúčov. Táto plocha je prevedená na body sú duplikované a posunuté v Z-ovej osi, aby modifikovali výšku snehu. Hustota snehu je upravovaná podľa čiernych miest na textúre šumu.

Následne sú všetky body prevedené na objem (volume) pomocou voxelov. Užívateľ musí modifikovať veľkosť voxelov a počet iterácií subdivision v prípade, že ide o väčšie plochy, aby sa správne vygeneroval výsledný sneh. Pre zachovanie adaptivity voxelov je ich veľkosť násobená veľkosťou objektu. Pozri obrázok 4.11. Následne je z vygenerovaného objemu vygenerovaná geometria, ktorá tvorí sneh. Vygenerovaná geometria je prevedená na trojuholníky, aby bolo možné lepšie modifikovať tvar objektu.



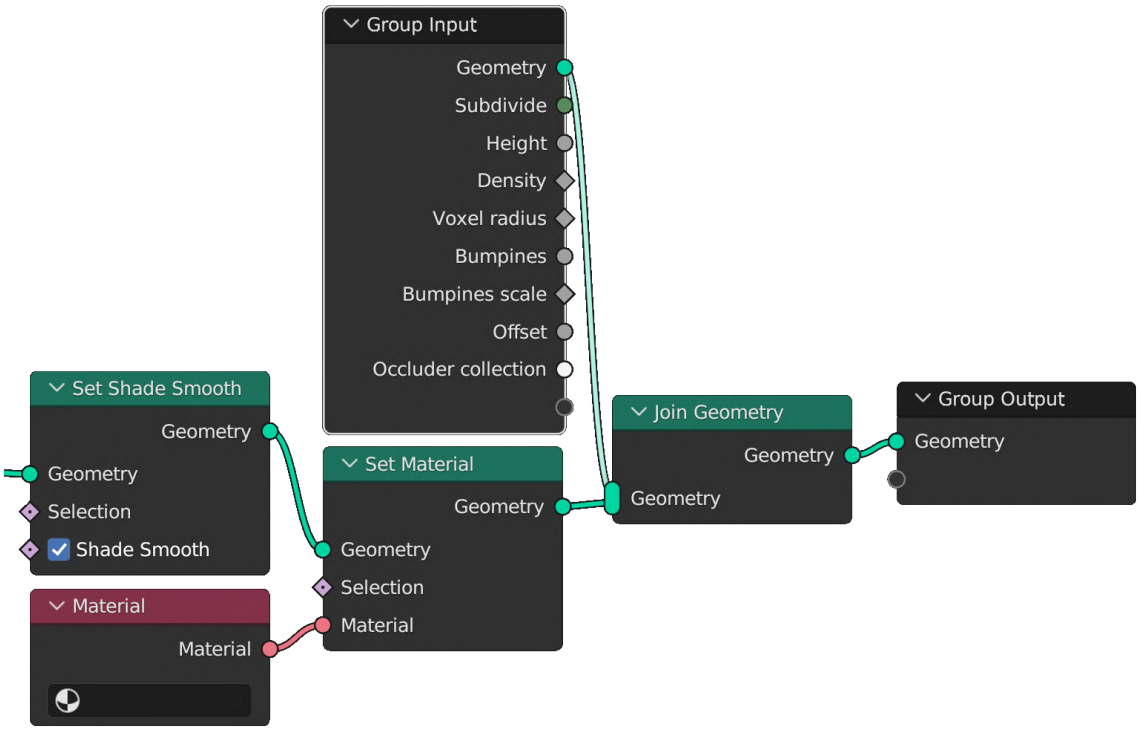
Obr. 4.11: Strom uzlov generujúci z plôch výslednú geometriu. Na obrázku je možné vidieť, že body dopadu geometrie sú prevedené na objem, ktorý je následne prevedený na mesh. V tomto prístupe nebolo možné využiť uzol vytiahnutia, pretože sa tu nachádzajú aj kolmé plochy okolo bodov dopadu lúčov.

Vygenerovanej geometrii je pomocou textúry šumu pridaná hrboľatosť. Pozri obrázok 4.12. Užívateľ môže ľubovoľne modifikovať silu a veľkosť hrboľov. V prípade potreby môže užívateľ pridať snehu posuv v osi Z.



Obr. 4.12: Strom uzlov, ktorý pridáva snehu hrboľatosť a posúva ho v osi Z. Pre pridanie hrboľatosti je použitá textúra šumu, ktorej veľkosť je násobená veľkosťou objektu, aby sa zachovali veľkosti hrboľov aj pri nerovnomernej veľkosti objektu.

Ako posledné je snehu priradený materiál imitujúci textúru snehu (obrázok 4.15) a výsledný sneh je spojený s pôvodnou geometriou objektu. Pozri obrázok 4.13. Na obrázku 4.14 je možné vidieť ako raycast funguje pri prekrývaní snehu objektmi.



Obr. 4.13: Strom uzlov pridávajúci materiál a generujúci výslednú geometriu. Z postupnosti uzlov je možné vidieť, že k objektu je na konci pridaná pôvodná geometria, aby sme nestratili geometriu, na ktorej je sneh vygenerovaný. Uzol Materiál nemá priradenú žiadnu hodnotu, pretože táto hodnota je nastavená až po importovaní modifikátora. V opačnom prípade by sa materiál v scéne duplikoval.

Obr. 4.14: Príklad možného výsledku vygenerovaného pomocou metódy vrhania lúčov. Z obrázku je možné vidieť, že miesta, ktoré su zakryté plochami nie sú zasnežené.

Obr. 4.15: Procedurálne vytvorený materiál imitujúci sneh.

Pre každý vygenerovaný sneh je pridaný modifikátor decimate, ktorý zjednodušuje finálnu geometriu a počet trojuholníkov.

4.4 Texture tools

Sada nástrojov Texture tools je lokalizovaná v UV Editore. Nástroje pre nastavenie aktívneho obrázku sa nachádzajú v priamom paneli a nástroje pre prácu s hustotou pixelov sa nachádzajú v podpaneli nachádzajúcom sa v hlavnom paneli.

4.4.1 Set active image

Pre nastavenie aktívnej textúry musí mať užívateľ zvolený aspoň jeden objekt obsahujúci geometriu. Následne si užívateľ zvolí textúru, ktorú chce nastaviť ako aktívnu. Pole na zvolenie obrázku je typu PointerProperty, ktoré obsahuje ukazovateľ priamo na obrázok. Parameter obrázku je obmedzený typovo iba na obrázky, ktoré obsahuje daná scéna.

Po zavolaní operátora prejde program všetky objekty, ktoré sú označené. V každom objekte prejde všetky materiály a spracuje ich. Pri spracovaní materiálu sa ako prvé skontroluje, či materiál obsahuje nejaký aktívny uzol, ktorý je typu obrázková textúra. Ak áno, skontroluje sa, či obsahuje požadovanú textúru. Ak ju neobsahuje, textúra je následne nastavená.

V prípade, že aktívny uzol nie je požadovaného typu, kontroluje sa každý uzol v Shader editore (viac v sekcii 2.6.4) a hľadá sa uzol typu obrázková textúra. V prípade, že sa takýto uzol nájde, skontroluje sa, či neobsahuje prepojenia s ostatnými uzlami. V prípade, že ich obsahuje, je to textúra materiálu a nie textúra, na ktorú sa majú vypiecť informácie. Program hľadá ďalší uzol požadovaného typu, ktorý neobsahuje žiadne pripojenia, aby mohol nastaviť požadovanú textúru. Ak sa nenájde v strome žiaden uzol typu obrázková textúra, tento uzol je vytvorený, nastavený ako aktívny, zvolený je obrázok, ktorý užívateľ chce nastaviť. Pre urýchlenie chodu programu je každý skontrolovaný materiál uložený do pomocného zoznamu, aby v prípade, že ten istý materiál obsahuje aj iné objekty, nebol zbytočne kontrolovaný a nespotrebovával výpočtový čas.

4.4.2 Texel density tools

Pre použitie operátora na výpočet hustoty textúry musí byť užívateľ prepnutý do edit módu objektu. V prípade, že si nechal zvolenú možnosť použitia aktívneho obrázku z UV Editora sa ako prvá vykonáva kontrola, či má nejaký obrázok zvolený, aby bolo možné vypočítať hustotu pixelov. V prípade, že v UV Editore nie je žiadna textúra aktívna, je operácia zrušená.

Pre výpočet sa ako prvé prepne scéna do edit módu a naspäť, aby sa zaistila aktualizácia plôch, ktoré má užívateľ označené. Následne je pre každý označený objekt vypočítaná plocha v 3D View a plocha UV mapy. Pre výpočet je použitý modul bmesh, ktorý obsahuje nástroje pre prácu s geometriou objektu. Z geometrie objektu sa vytvorí nový bmesh, čo zaistí kópiu objektu, ale iba v operačnej pamäti a vytvorí sa zoznam z označených plôch, pre ktoré chce užívateľ vypočítať hustotu pixelov. Následne sa zaistí zoznam bodov tvoriacich UV mapu každej plochy.

Pre výpočet plochy UV mapy je použitý Shoelace algoritmus (viac v sekcii 2.7). Na výpočet plochy označených stien 3D objektu je použitá funkcia blenderu `calc_area()`.

Následne je vypočítaný počet pixelov/m^2 pomocou vzorca:

$$H \cdot W \cdot A_{UV}/A_{obj} = A\ [px/m^2] \tag{4.2}$$

Kde jednotlivé parametre znamenajú:
H - Výška textúry v pixeloch.
W - Šírka textúry v pixeloch.
A_{UV} - Percentuálne pokrytie UV mapou.
A_{obj} - Veľkosť plochy označených stien.
A - Výsledná hustota v px/m^2

$$TD\ [px/cm] = \sqrt{A}/100 \tag{4.3}$$

Kde:
TD - Výsledná hustota v px/cm

Pre získanie px/cm je táto hodnota odmocnená druhou mocninou a vydelená 100. Pre každú plochu je uložená vypočítaná hustota textúry, aby sa v prípade viacero plôch mohol následne vypočítať priemer. Pre urýchlenie výpočtov je použitá knižnica NumPy. Výsledná hodnota je uložená do pomocného parametru, ku ktorému má prístup panel obsahujúci nástroje, ktorý následne zobrazí vypočítanú hodnotu zaokrúhlenú na 3 desatinné miesta.

Pre nastavenie vlastnej hustoty textúry je zavolaný operátor na získanie hustoty textúry, ktorý zaistí aktuálnu hodnotu. Následne je vypočítaný faktor veľkosti pomocou vzorca požadovaná veľkosť / vypočítaná veľkosť, ktorý udáva ako veľmi sa musí aktuálna veľkosť zmeniť. Tento faktor je použitý pri zmene veľkosti UV mapy ako veľkosť, na ktorú sa má UV mapa prispôsobiť.

Požadovaná hustota pixelov je automaticky aktualizovaná na odporúčanú hodnotu vždy pri zmene typu viditeľnosti objektu. Informácia o rozlíšení textúry objektu je vypočítaná ako požadovaná hustota textúry + odchýlka 75%. V prípade presahu zhora je užívateľovi odporúčané znížiť rozlíšenie textúry, v prípade presahu zospodu je odporúčané rozlíšenie textúry zvýšiť.

Kapitola 5

Testovanie

Každý nástroj bol priebežne testovaný počas implementácie. Výsledný produkt bol poskytnutý skupine potenciálnych užívateľov.

5.1 Testovanie užívateľmi

Výsledný produkt testovalo viacero užívateľov s rozdielnymi skúsenosťami a zameraniami pri tvorbe modelov. Jednou skupinou (počet ľudí 4) boli modderi, ktorí tvoria modely do rozdielnych hier a druhú skupinu tvorili vývojári hier (počet ľudí 2), ktorí sa viac zameriavajú na programovanie a nemajú veľké skúsenosti s tvorbou modelov. Užívateľom bolo vysvetlené, kde sa aké nástroje nachádzajú, ale nebolo im vysvetlené ako sa ovládajú, aby reakcie na intuitívnosť neboli skreslené. Testovanie prebiehalo cez online hovor na Discorde, takže spätná väzba bola v reálnom čase.

Na základe spätnej väzby boli niektoré nedostatky následne odstránené alebo opravené. Užívateľské rozhranie bolo dostatočne intuitívne aj v prípadoch, keď užívatelia neboli oboznámení ako dané nástroje používať. Užívatelia ocenili popisy každého operátora a každého parametru, ktoré add-on obsahuje, vďaka čomu vedeli, čo daná hodnota alebo operácia robí.

Pri testovaní bolo vidieť, že každá skupina využíva iné nástroje. Skupina modderov viac využívala nástroje Explode objects, Generovanie bounding boxov, Load On Empty, Convert To Skeleton a nástroje pre prácu s textúrami, zatiaľ čo skupina tvorcov hier využívala skôr nástroje na generovanie levelu detailov, generovanie charakteru a generovanie bounding boxov.

Jeden z užívateľov, ktorý robí hry do bodu, kedy dostal add-on na testovanie netušil o tom, že existuje niečo ako hustota pixelov a prečo je dôležitá, čo sa odzrkadlilo aj na jeho spätnej väzbe, kedy netušil ako nástroj používať. Po rýchlom vysvetlení a naštudovaní daného problému ocenil, že nástroj ponúka možnosť výberu typu viditeľnosti objektu a odporúčané hodnoty, čo mu uľahčilo rozhodovanie o správnej hodnote hustoty. Následne ocenil aj informáciu o tom, či má jeho použitá textúra pre daný objekt dostatočné rozlíšenie.

Väčšina užívateľov označila generovanie snehu ako „COOL“ a pochvaľovali si dynamické prispôsobenie snehu na daný objekt. Avšak táto funkcionalita si našla len jedného užívateľa pre finálne použitie, ktorým bol modder zameraný na tvorbu budov do hier. Nízka odozva pre generovanie snehu mohla byť spôsobená nízkym počtom užívateľov, ktorý výslednú prácu testovali.

5.2 Meranie času

Jedným z hlavných zámerov práce je urýchliť proces tvorby herných assetov. Nasledujúce porovnania ukazujú, koľko času ušetrí užívateľ pri použití nástrojov oproti tomu, keby vykonáva daný proces manuálne. Čas je meraný na osobe, ktorá má dlhoročné skúsenosti s programom Blender, pozná klávesové skratky a spôsoby ako urýchliť procesy. V prípade začiatočníkov alebo ľudí čo Blender nepoužívajú pravidelne sú časy výrazne vyššie.

Generate LOD
Manuálny postup: Najskôr je potrebné zduplikovať objekty, následne pridať modifikátor Decimate a nastaviť hodnotu, aplikovať modifikátor pre všetky objekty.
Add-on: Užívateľ si vyberie požadovaný level a nechá vygenerovať LOD objekty.
Ušetrený čas: Maximálne do pol minúty. V prípade, že objekt obsahuje modifikátory, je potrebné manuálne aplikovať každý modifikátor, čo predlžuje čas približne o 2 sekundy za každý objekt.
Výhody: Rýchlejší proces, netreba nastavovať hodnoty a premenovávať objekty.
Nevýhody: Absencia rôznych nastavení pri zjednodušovaní objektu.

Bounding Box
Manuálny postup: V prípade OBB, čo je najzložitejší druh boxu, je potrebné, aby užívateľ vytvoril kváder, následne našiel správnu orientáciu a správne prispôsobil box na objekt.
Add-on: Užívateľ si vyberie typ boxu a nechá vygenerovať.
Ušetrený čas: Niekoľko minút až desiatok minút, záleží od orientácie objektu a typu boxu.
Výhody: Jednoduchý na ovládanie, šetrí čas. Presnosť, ktorú užívateľ odhadom nezvládne.
Nevýhody: Možnosť len 4 druhov boxov.

Explode Objects
Manuálny postup: Každý objekt posunúť na náhodnú pozíciu. V prípade, že užívateľ chce vrátiť objekty do pôvodného stavu, musí mať kópiu všetkých modelov, čo zaberá viac pamäte a je to znateľne náročnejšie na údržbu.
Add-on: Stačí nastaviť hodnoty.
Ušetrený čas: Čas narastá s vyšším počtom objektov, ušetrených je niekoľko minút.
Výhody: Rýchlejší, možnosť uloženia pozície bez prepisu.
Nevýhody: Žiadne.

Load On Empty
Manuálny postup: Pre každý empty objekt zduplikovať cieľový objekt a nastaviť ho ako potomka a posunúť ho na stred objektu. V prípade odstránenia objektov je proces potrebné opakovať znovu pre každý objekt.
Add-on: Pre každý objekt je potrebné nastaviť cieľový objekt alebo vybrať možnosť rovnakého objektu pre všetky prázdne body.
Ušetrený čas: V prvej iterácii pár minút, záleží od počtu objektov. V každej ďalšej iterácii ušetrený čas narastá lineárne, absencia nastavovania cieľového objektu.
Výhody: Znateľne rýchlejší, jednoduchší, možnosť načítania a odstránenia objektu.
Nevýhody: Žiadne.

Convert To Skeleton
Manuálny postup: Vytvoriť kosť pre každý objekt v jeho pozícii s jeho rotáciou. Pridať každému objektu skupinu bodov, ktorá ho ovláda. Spojiť model a prepojiť ho s kostrou. Vrátenie do pôvodného stavu iba cez ctrl + Z, čo nie je dobré riešenie, ak bol vykonaný veľký počet operácií od vytvorenia spojeného objektu.
Add-on: Označiť objekty a nechať vygenerovať kostru.
Ušetrený čas: Záleží od počtu objektov, niekoľko desiatok minút, až pár hodín.
Výhody: Vytvorenie kostry jedným kliknutím, možnosť vrátiť objekty do pôvodného stavu.
Nevýhody: Limitované iba na mechanické objekty, nie pre organické modely. Nie je možné vrátiť pôvodné modifikátory objektom.

Simple Character Generator
Manuálny postup: Vymodelovať postavu, oblečenie, pripraviť pre každé oblečenie textúry. Pre každý objekt je potrebné pripraviť skinned mesh (viac v sekcii 2.5) a pripraviť kostru.
Add-on: Zvoliť konfiguráciu postavičky a nechať vygenerovať.
Ušetrený čas: Niekoľko hodín, v prípade, že by užívateľ chcel viacero oblečení je rozdielny čas viac ako 10 hodín. V prípade užívateľa, ktorý sa nezaoberá modelovaním je to v rádoch niekoľkých dní.
Výhody: Veľmi rýchly, veľký počet možností, možnosť náhodného generovania, všetko pripravené pre pohyb postavy.
Nevýhody: Obmedzené na jednoduchú postavu, ktorá je pripravená. Užívateľ si ale môže všetko prispôsobiť ako potrebuje.

Snow Generator
Manuálny postup: Buď pomocou kombinácie modifikátorov alebo sculptingom. Je potrebné manuálne upravovať a prispôsobovať.
Add-on: Pár kliknutí.
Ušetrený čas: Desiatky minút až hodiny.
Výhody: Modifikácia v reálnom čase, adaptívny podľa povrchu a rotácie objektu, možnosť zakrývania objektmi, primeraný počet trojuholníkov výslednej geometrie.
Nevýhody: Pre veľké objekty môže byť náročnosť na výpočet vysoká pri modifikovaní hodnôt v reálnom čase.

Set Active Image
Manuálny postup: Pre každý objekt je potrebné nastaviť pre každý materiál aspoň jedenkrát aktívnu textúru.
Add-on: Vybrať obrázok a nastaviť.
Ušetrený čas: Záleží od počtu objektov a materiálov. Cca 2 sekundy pre každý materiál na objekte + 3 sekundy za každý objekt.
Výhody: Rýchly, jednoduchý.
Nevýhody: Žiadne.

Texel Density Tools
Manuálny postup: Hustotu pixelov vypočítať manuálne pre zložitejšie objekty nie je možné, jediná možnosť je použiť voľne dostupný add-on pre prácu s texel density.
Add-on: Stačí zvoliť plochy a nechať vypočítať hustotu. V prípade nastavenia hustoty stačí zvoliť hodnotu.

Ušetrený čas: Jediné, čo je možné merať, je prispôsobenie veľkosti UV mapy podľa inej UV mapy. V tomto prípade je možné ušetriť nejaké sekundy oproti manuálnemu procesu. Netreba študovať jednotlivé hodnoty pre objekty.
Výhody: Ponúka možnosti odporúčanej hustoty pixelov podľa typu viditeľnosti objektu, informuje užívateľa o tom, či je vypočítaná hustota dostačujúca.
Nevýhody: Neponúka toľko možností ako konkurenčný add-on, veľkú časť ale bežný užívateľ nevyužije.

Výsledky nástrojov je možné vidieť na obrázkoch v sekcii B.

Kapitola 6

Záver

Výsledkom tejto práce je add-on, ktorý ponúka sadu užitočných nástrojov pre tvorbu herných assetov. Nástroje sú určené pre začiatočníkov a užívateľov, ktorých priame zameranie nie je tvorba herných assetov. Každý nástroj ponúka všetky potrebné parametre, aby užívateľ využil jeho funkcionalitu naplno a prispôsobil si ju podľa danej potreby a situácie.

Výhodou tejto práce je, že obsahuje mnoho nástrojov, ktoré ponúkajú dostatočnú možnosť prispôsobenia podľa danej potreby. Neobsahuje však komplexné nastavenia, v ktorých by sa užívateľ mohol jednoducho stratiť, alebo by ich nepotreboval. Práve pre absenciu komplexných nastavení nie je výsledok tejto práce určený pre profesionálov, aj keď niektoré funkcie určite ocenia aj dlhoroční, skúsení tvorcovia herných assetov. Ďalšou výhodou tejto práce je dynamický prístup, ktorý dovoľuje užívateľovi kedykoľvek vykonávať zmeny a ponúka krajšie a lepšie modifikovateľné výsledky ako niektoré voľne dostupné konkurenčné add-ony.

Spätná väzba od užívateľov ukázala, že výsledná práca je dostatočne intuitívna na používanie aj pre začiatočníkov. Niektoré funkcie by potrebovali detailnejší popis na použitie, ale kvôli prehľadnosti obsahujú len skrátenú verziu popisu. Užívatelia prácu ocenili a potvrdili, že add-on budú aktívne využívať pri tvorbe modelov do hier.

Do budúcna by mohol byť add-on rozšírený o generátory budov a ciest alebo procedurálnej tvorby ostrovov, ktoré by boli plne modifikovateľné. Taktiež by mohla byť pridaná funkcionalita inteligentného rozbaľovania modelov na UV mapy.

Literatúra

[1] Adobe. *3D texturing and Adobe Substance 3D*. 2023. Online, accessed February-2023. Dostupné z: https://www.adobe.com/mt/products/substance3d/discover/3d-texturing.html.

[2] Allgower, E. L. a Schmidt, P. H. Computing Volumes of Polyhedra. *Mathematics of Computation*. American Mathematical Society. 1986, zv. 46, č. 173, s. 171–174. ISSN 00255718, 10886842. Online, accessed April-2023. Dostupné z: http://www.jstor.org/stable/2008221.

[3] Aristidou, A., Lasenby, J., Chrysanthou, Y. a Shamir, A. Inverse Kinematics Techniques in Computer Graphics: A Survey. *Computer Graphics Forum*. 2018, zv. 37, č. 6, s. 35–58. DOI: https://doi.org/10.1111/cgf.13310. Online, accessed April-2023. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13310.

[4] Bethke, E. *Game Development and Production*. Wordware Pub., 2003. ITPro collection. ISBN 9781556229510. Online, accessed February-2023. Dostupné z: https://books.google.cz/books?id=m5exIODbtqkC.

[5] Community, B. O. *Blender - a 3D modelling and rendering package*. Stichting Blender Foundation, Amsterdam: Blender Foundation, 2018. Online, accessed February-2023. Dostupné z: https://docs.blender.org/manual/en/latest/index.html.

[6] Contributor, T. *3D mesh*. 2016. Online, accessed February-2023. Dostupné z: https://www.techtarget.com/whatis/definition/3D-mesh.

[7] Dries, T. *Texel Density*. 2022. Online, accessed February-2023. Dostupné z: https://www.artstation.com/a/24186644.

[8] Eberly, D. *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. CRC Press, 2006. ISBN 9781482267303. Online, accessed February-2023. Dostupné z: https://books.google.cz/books?id=TnwZBwAAQBAJ.

[9] Flavell, L. UV Mapping. In: *Beginning Blender: Open Source 3D Modeling, Animation, and Game Design*. Berkeley, CA: Apress, 2010. ISBN 978-1-4302-3127-1. Online, accessed May-2023. Dostupné z: https://doi.org/10.1007/978-1-4302-3127-1_5.

[10] Hart, J. C. Perlin Noise Pixel Shaders. In: *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*. New York, NY, USA: Association for Computing Machinery, 2001, s. 87–94. HWWS '01. DOI: 10.1145/383507.383531. ISBN 158113407X. Online, accessed April-2023. Dostupné z: https://doi.org/10.1145/383507.383531.

[11] Hettinga, G. J., van Beckhoven, R. a Kosinka, J. Noisy gradient meshes: Augmenting gradient meshes with procedural noise. *Graphical Models*. 2019, zv. 103, s. 101024. DOI: https://doi.org/10.1016/j.gmod.2019.101024. ISSN 1524-0703. Online, accessed April-2023. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1524070319300153.

[12] Iezzi, L. *Texel Density: All you need to know HQ*. 2021. Online, accessed February-2023. Dostupné z: https://www.artstation.com/a/24186644.

[13] Lagae, A., Lefebvre, S., Cook, R., DeRose, T., Drettakis, G. et al. A Survey of Procedural Noise Functions. *Computer Graphics Forum*. 2010, zv. 29, č. 8, s. 2579–2600. DOI: https://doi.org/10.1111/j.1467-8659.2010.01827.x. Online, accessed April-2023. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2010.01827.x.

[14] Lee, Y. a Lim, W. Shoelace Formula: Connecting the Area of a Polygon and the Vector Cross Product. *The Mathematics Teacher*. Reston VA, USA: National Council of Teachers of Mathematics. 2017, zv. 110, č. 8, s. 631 – 636. DOI: 10.5951/mathteacher.110.8.0631. Online, accessed April-2023. Dostupné z: https://pubs.nctm.org/view/journals/mt/110/8/article-p631.xml.

[15] Manocha, D., Lin, M., Brooks, F. a Gottschalk, S. Collision Queries using Oriented Bounding Boxes. September 2000, s. 40. Online, accessed May-2023.

[16] Orvalho, V., Bastos, P., Parke, F. I., Oliveira, B. a Alvarez, X. A Facial Rigging Survey. *Eurographics (State of the Art Reports)*. 2012, s. 183–204. Online, accessed April-2023.

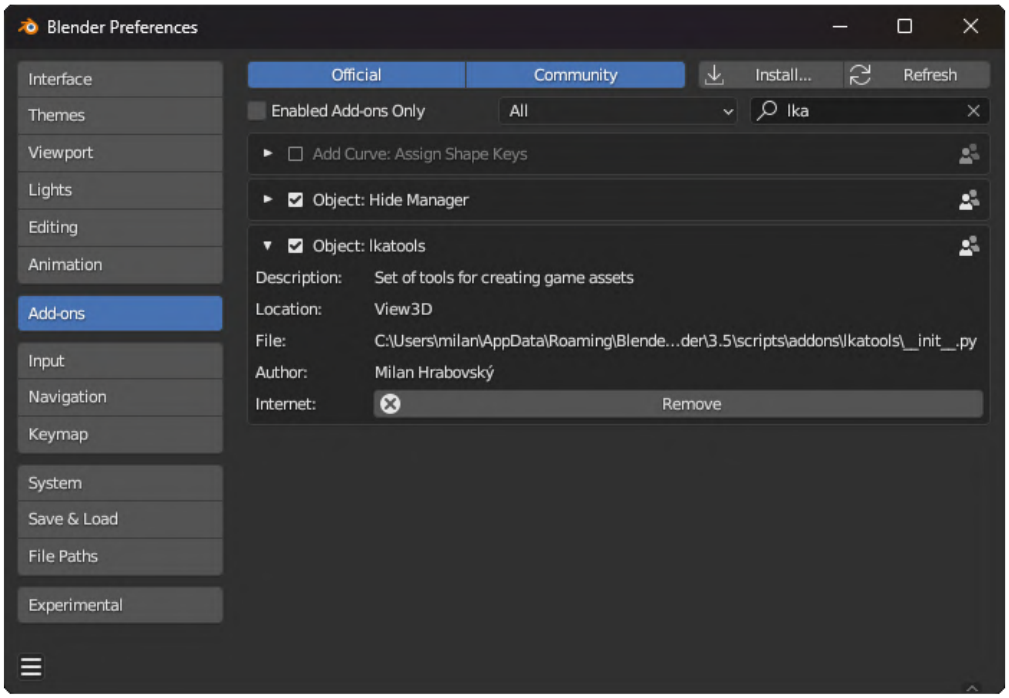
Príloha A

Inštalácia

Add-on je voľne dostupný na adrese: https://lkaminco.gumroad.com/l/lkatools. V prípade záujmu môže uživateľ zaplatiť ľubovoľnú sumu.

Pre inštaláciu je potrebné stiahnť add-on s vyššie uvedeného odkazu. Následne je potreba nainštalovať add-on do Blenderu. V Blenderi Edit → Preferences → Add-ons. V záložke add-onov je následne potrebné kliknúť na Install a v kontextovóm okne zvoliť stiahnutý .zip súbor s add-onom. Po nainštalovaní je potrebné potvrdiť kliknutím na zaškrtávacie políčko ako je vidno na obrázku A.1, aby sa nástroj spustil v Blenderi.

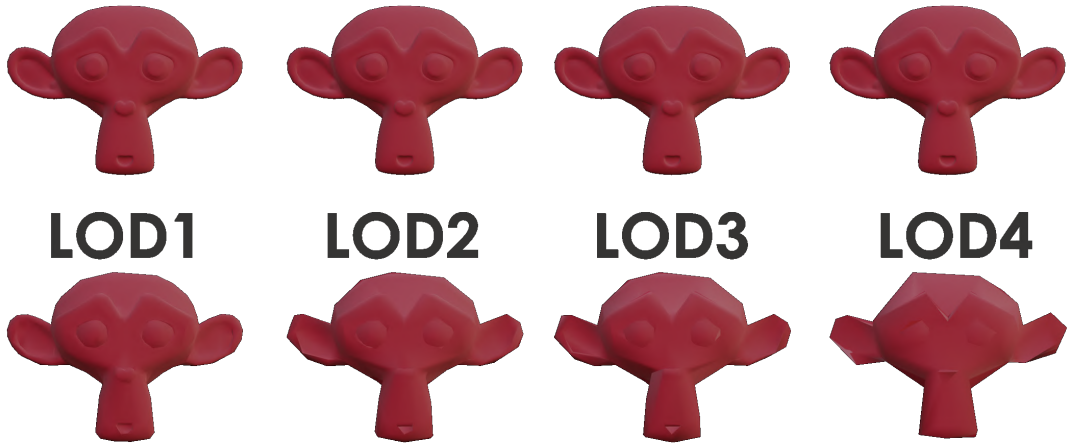
Nástroje na generovanie snehu, charakterov a prácu s geometriou objektu je možné nájsť v 3D View v záložke lkatools. Nástroje na prácu s textúrami sa nachádzajú v záložke lkatools v UV Editore.



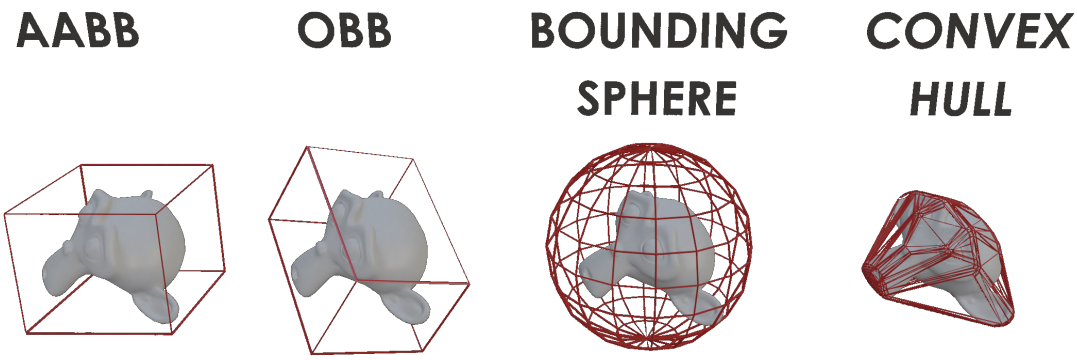
Obr. A.1: Postup inštalácie add-onu v Blenderi 3.4

Príloha B

Obrázky výsledkov nástrojov



Obr. B.1: Nástroj an generovanie levelu detailov pre objekty. Je možné vidieť že každá úroveň postupne znižuje na definicii geometrie.



Obr. B.2: Vygenerované obaľovacie boxy pre objekty. Je možné vidieť ako každý druh boxu obaľuje objekt s inou presnosťou

Obr. B.3: Nástroj Load On Empty. Na spodnom obrázku je možné vidieť že na každý prázdny objekt je načítaný cieľový objekt.

Obr. B.4: Nástroj explode objects. Je možné vidieť že na spodnej fotke sú objekty v nastavenej vzdialenosti od seba..

Obr. B.5: Príklady možných vygenerovaných charakterov.

Obr. B.6: Na obrázku je možné vidieť že charakter je pripravený a v hernóm engiene je možné pohybovať jeho jednotlivými časťami. Ukážka je v Unreal Engine 5.