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

BLENDER ADD-ON PRO GENEROVÁNÍ ASTRONOMICKÝCH OBJEKTŮ

BLENDER ADD-ON FOR ASTRONOMICAL OBJECT GENERATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MAREK HLÁVKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. TOMÁŠ CHLUBNA**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Hlávka Marek**
Program: Informační technologie
Název: **Blender add-on pro generování astronomických objektů**
Blender add-on for Astronomical Object Generation
Kategorie: Počítačová grafika

Zadání:

1. Naučte se základy práce s 3D modelovacím programem Blender (verze 2.9 a výše).
2. Seznamte se s Blender Python API.
3. Navrhněte add-on, který rozšíří Blender o možnost generování astronomických objektů a vesmírných scén.
4. Nastudujte možná a existující řešení dané problematiky.
5. Navrhněte uživatelské rozhraní pro výsledný add-on.
6. Add-on implementujte a demonstrujte jeho použití na různých typech dat.
7. Zdokumentujte a zveřejněte výsledný add-on pro použití dalšími uživateli.
8. Vytvořte video reprezentující výsledky vaší práce.

Literatura:

- Ebert, David S., et al. Texturing & modeling: a procedural approach. Academic Press, 2014. ISBN 1483297020, 9781483297026
- Lagae, Ares, et al. "A survey of procedural noise functions." *Computer Graphics Forum*. Vol. 29. No. 8. Oxford, UK: Blackwell Publishing Ltd, 2010.
- Parkes, Steve & Martin, I. & Dunstan, M. & Matthews, D.. (2004). Planet Surface Simulation with PANGU. *Space ops 2004 conference*. 10.2514/6.2004-592-389.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 5, experimenty vedoucí k bodu 6.

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

Vedoucí práce: **Chlubna Tomáš, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Metody procedurální modelování značně ulehčují proces modelování 3D objektů napodobujících prvky reálného světa, který je potřeba provádět čím dál tím efektivněji. Tato práce se zabývá procedurálním modelováním vesmírných objektů a jejich úpravou. Také se zabývá návrhem, implementací a publikování add-onu do open-source program Blender, který modelování těchto objektů umožnuje. Add-on je implementován v jazyce Python za pomocí Blender API. Blender je díky tomu rozšířen o rozhraní pro rychlé a pohodlné generování vesmírných objektů. Publikovaný add-on je dostupný open-source.

Abstract

Procedural modeling methods greatly facilitate the process of modeling 3D objects that mimic real-world elements, which needs to be performed more and more efficiently. This thesis deals with procedural modeling of astronomical objects and their modification. It also deals with design, implementation and publication of an add-on for the open-source program Blender, that enables modeling of these objects. The add-on is implemented in Python using the Blender API. As a result, Blender is extended with an interface for fast and convenient generation of astronomical objects. The published add-on is available open-source.

Klíčová slova

Procedurální generování, Vesmírné objekty, Planety, perlinův šum, Blender, Add-on, Počítačová grafika, Python

Keywords

Procedural generation, Space objects, Planets, perlin‘s noise, Blender, Add-on, Computer graphics, Python

Citace

HLÁVKA, Marek. *Blender add-on pro generování astronomických objektů*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Chlubna

Blender add-on pro generování astronomických objektů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Chlubny. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Marek Hlávka
11. května 2022

Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Tomáši Chlubnovi za ochotu, tipy na zlepšení doplňku a rychlou odezvu na mé dotazy při tvorbě práce. Také musím poděkovat všem uživatelům co mi pomohli doplněk otestovat a všem prátelům co mě během práce podporovali.

Obsah

Kapitola 1

Úvod

Hlavním cílem počítačové grafiky je co nejpřesněji napodobit podobu reálného světa, případně tuto imitaci dále libovolně upravovat. Ať už je to pomocí fotografií, 3D modelů nebo např. uměleckých filtrů. Generování virtuálního světa na základě reálného se však setkává s jedním zásadním problémem – náhodností. V reálném světě je každý objekt alespoň trochu odlišný než ostatní objekty stejného typu. tato unikátnost se díky přesnosti a determinismu počítačů obtížně přenáší do počítačové grafiky. U modelování jednoho modelu je tuto překážku možné obejít lidským faktorem, ale při modelování desítek či stovek modelů stejného typu to je velmi časové náročné. Proto je rychlé procedurální generování objektů stejného typu stále vyhledávanější, ať už kvůli prostému vykreslování snímků nebo kvůli filmové a herní tvorbě.

Cílem práce je vytvořit rozšíření do open-source programu Blender pro procedurální generování vesmírných objektů jako např. planety, hvězdy a meteory. Rozšíření podporuje úpravu různých parametrů, od procentuálního zaplavení planety, přes tvorbu realistických kráterů, až po přidávání textur na jednotlivé výškové profily objektů. Modely jsou přidávány jako jednotlivé objekty, které je možné dále upravovat standardními zabudovanými operacemi nad 3D scénou, což ve výsledku umožňuje detailní přizpůsobení scény.

Toto rozšíření přídává oproti existujícím řešením možnost modelování více druhů vesmírných objektů, jejichž povrch není nijak závisly na textuře, modelování vesmírného pozadí a širokou možnost úprav jednotlivách objektů.

V kapitole 2 jsou rozebrány obecné definice vesmírných objektů, problematika tvorby modelu koule a popis aplikace Blender a jeho aplikačního rozhraní. V kapitole 3 je popsán výběr vesmírných objektů pro toto rozšíření, popis návrhu uživatelského rozhraní a obecný popis tvorby objektů a jejich povrchů. V kapitole 4 je poté rozebráno využití aplikačního rozhraní aplikace Blender, konkrétní postup tvorby objektů, tvorby několika typů povrchů a popsány důležité funkce. V kapitole 5 je poté popsán způsob testování a publikace doplňku. Příloha A obsahuje snímky vykreslené hotovým doplňkem.

Kapitola 2

Teorie

V této kapitole jsou popsány minimální nutné znalosti pro porozumění problematice řešené v práci. Jedná se o teoretické informace týkající se modelu, vesmírných objektů, problematiky modelování sférických modelů, generování šumu, generování kráterů, Blenderu, jeho Python API a také jsou zde popsána existující řešení procedurálního generování vesmírných objektů.

2.1 Model

Model v počítačové grafice lze chápat jako obraz nebo imitaci vzoru, kterým může být konkrétní objekt v reálném světě a nebo fiktivní představa autora [5]. V počítačové grafice jsou modely často reprezentovány jako skupina bodů, tzv. vertexů, v prostoru. Jednotlivé modely jsou složené z polygonů, které jsou tvořené třemi nebo více vertexy.

Základem většiny vesmírných objektů je díky gravitačním silám koule. Existují tři nejběžnější přístupy jak rozprostřít body na kulový povrch a z nich pak vytvořit model koule: uv-sphere, fibonacciho koule a promítání na kulovou plochu.

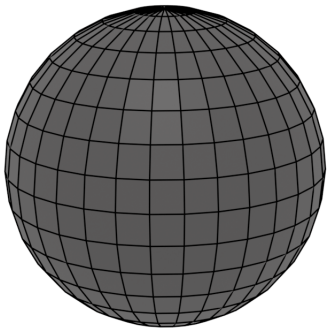
2.1.1 UV-Sphere

Tvorba tohoto modelu je založena na uv-mapování 2D souřadnic na 3D model. Celý princip spočívá ve vytvoření obdélníku, jehož délka je rovna délce obvodu kulové plochy, která se bude modelovat. Šířka tohoto obdélníku je rovna polovině jeho délky. Následně je tento obdélník obtočen kolem „rovníku“ kulové plochy, z čehož vznikne plášť válce. Tento plášť je následně zdeformován směrem k „pólům“ (k maximální a minimální Y souřadnici) kulové plochy, což je znázorněno na obrázku 2.1.

Pro modelování samotné koule není tento způsob příliš vhodný, neboť se velmi zvyšuje koncentrace vertexů při přibližování k pólům koule, a tudíž je model daleko více detailní než na „rovníku“. Tento postup je používán u jakéhokoliv texturování 3D modelu. Nejčastější využití najdeme v kartografii při každém plošném znázornění povrchu planety země na mapě.

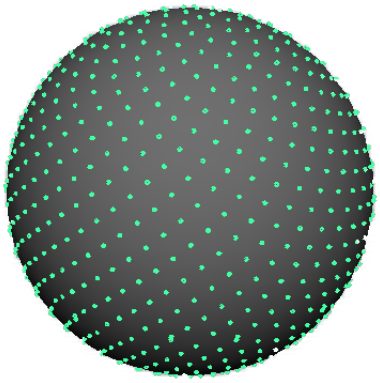
2.1.2 Fibonacciho koule

Problém nesourměné distribuce řeší jiný způsob rozprostření bodů na kulovou plochu: Fibonacciho koule (obrázek 2.2). Její generování je vytvořeno na základě Fibonacciho spirály,



Obrázek 2.1: Znázornění UV-sphere a její nerovnoměrnost rozložení polygonů na celou plochu.

díky čemuž jsou vzdálenosti mezi jednotlivými body relativně stejné velké, nezávisle na počtu vrcholů kulové plochy.[1] Problém ale vzniká v následujícím kroku.



Obrázek 2.2: Znázornění Fibbonaciho koule.

Pro modelování jakéhokoliv modelu je potřeba znát které body sousedí se kterými, kvůli tvorbě polygonů a celkového modelu. Při tomto přístupu ale v momentě přidání jednoho bodu změní svoji pozici téměř všechny předešle vytvořené body a tudíž se změní body jednotlivých polygonů. Tento přístup je k modelování samotné koule nevhodný a slouží pouze na rovnoměrné rozprostření bodů na kulovou plochu.

2.1.3 Promítání na kulovou plochu

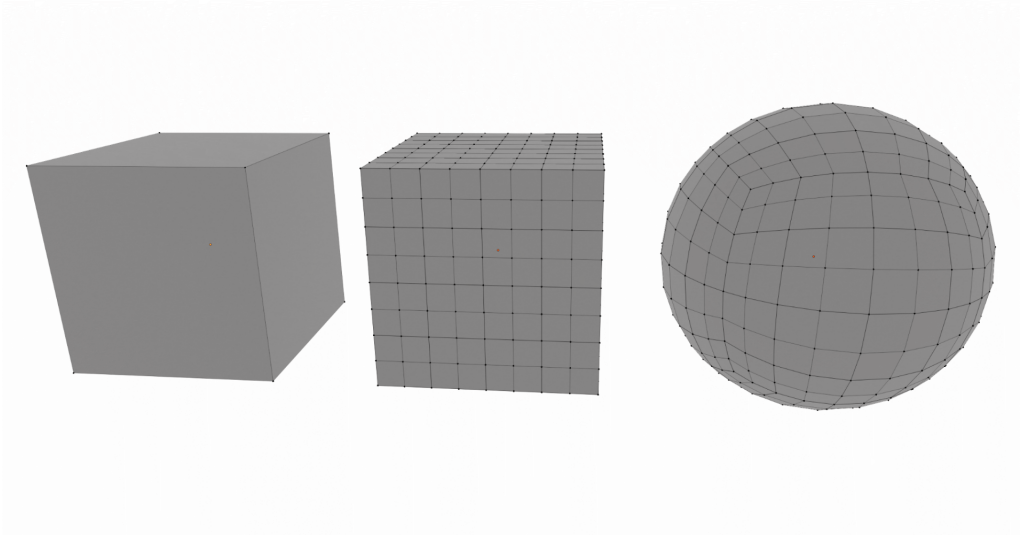
Oba problémy předchozích způsobů řeší promítáni jednodušších objektů na kulový povrch. Princip promítání spočívá ve vytvoření objektu, u něhož je známa pozice všech bodů a všechny polygony tvořené těmito body. Poté je určen bod S, který bude sloužit jako střed

[1] DEMO: https://www.redblobgames.com/x/1842-delaunay-voronoi-sphere/#demo

nově vznikající kulové plochy. U všech bodů známého objektu je pak změněna velikost vektoru určujícího jeho vzdálenost od středu S, díky čemuž je docíleno rozprostření na kulový povrch a splněna základní vlastnost kulové plochy – koule je tvořena množinou všech bodů v prostoru, jejichž vzdálenost od zadaného bodu (středu) je právě rovna zadanému poloměru.

Podobnosti kouli lze pak dosáhnout detailnějším výchozím objektem, resp. výchozím objektem s více body a polygony. Pokud bude jako výchozí objekt zvolena krychle, tak se po promítání na kulovou plochu nezmění vůbec, neboť všechny body krychle mají od jejího středu souhlasnou vzdálenost. Tento problém lze vyřešit dělením jednotlivých polygonů (stran) výchozího objektu. Např. pokud se každá stěna krychle rozdělí na čtyři dílčí stěny, sestávajících se středu stěny, dvou středů stran a jednoho původního vrcholu, lze na kulový povrch promítat místo šesti, dvacet čtyři bodů a tím pádem čtyřnásobek polygonů, díky čemuž lze napodobit tvar koule daleko lépe.

Při zvolení krychle jako výchozího objektu nastává při promítání na kulovou plochu opět problém shlukování bodů, tentokráte ale v menším množství a na osmi bodech. Čím více bodů bude ve výchozím objektu, tím lépe bude dosáhnuto rovnoměrné rozložení bodů na kulové ploše. Postup znázorněn na obrázku 2.3.

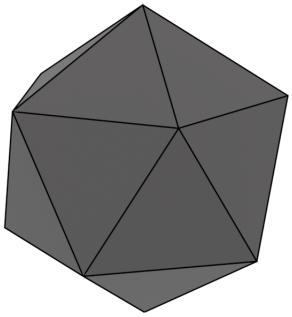


Obrázek 2.3: Znázornění krychle promítnuté na kulovou plochu.

2.1.3.1 Dvacetistěn

Poměrné složitý objekt, který tento problém do znatelné míry eliminuje je dvacetistěn, který je zobrazen na obrázku 2.4. Dvacetistěn (též ikosaedr) je trojrozměrné těleso, jehož stěny tvoří dvacet stejných rovnostranných trojúhelníků. Vzhledem k tomu, že tento objekt už do jisté míry připomíná kouli tak při promítání na kulovou plochu není deformace u jednotlivých vrcholů tak znatelná.

Dělení stran u dvacetistěnu je taktéž velmi jednoduché. Opět je každá strana rozdělena na čtyři dílčí strany s novými vrcholy ve středech hran dané strany. Tento proces lze opakovat periodicky stále za sebou. Při takovémto dělení se však počet nových bodů exponenciálně zvyšuje, tudíž už při velmi nízké úrovni dělení je potřeba velká výpočetní kapacita.



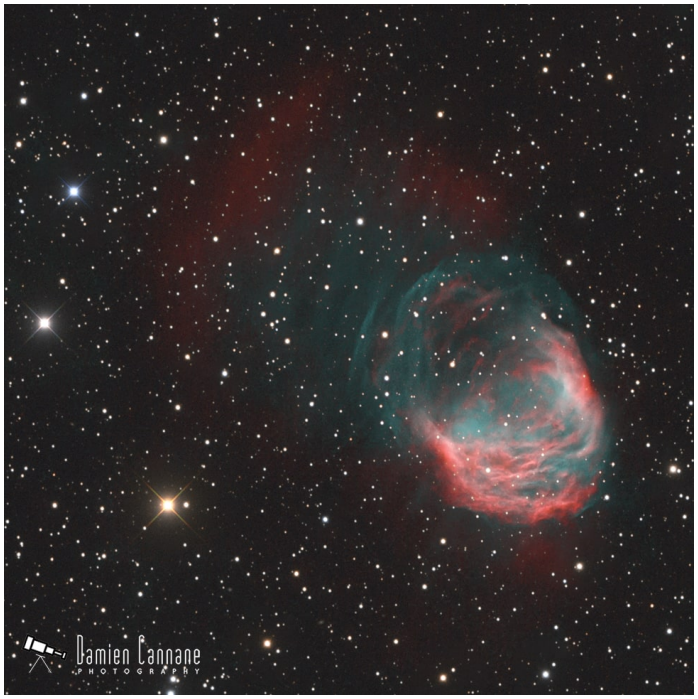
Obrázek 2.4: Znázornění Dvacetistěnu.

U druhého způsobu dělení stran je jakožto úroveň dělení potřeba brát počet částí, na které bude hrana rozdělena, a ne počet zanořených dělení stran. Nevýhoda tohoto přístupu je, že pro každé dělení je potřeba vycházet ze stejného objektu a nelze přecházet po úrovních „nahoru“ a „dolů“, jako u předchozí metody. Oproti tomu mezi její výhody patří, že je umožněna daleko lepší kontrola nad počtem bodů, které tvoří výsledný objekt, neboť tento počet nestoupá tak rapidně v závislosti na úrovni dělení.

2.2 Vesmírný objekt

Tento doplněk se zabývá modelováním vesmírných objektů. Vesmírné objekty, neboli objekty které lze nalézt ve vesmíru a mohou být složené buď z jednoho samostatného objektu, nebo z velkého množství různých objektů, se dělí do sedmi základních kategorií [4].

- **Hvězdy** - Nejčastěji viditelný objekt na noční obloze. Jedná se o plazmu drženou v kulovitém tvaru pomocí značné gravitační síly. Nejznámější hvězdou je střed Sluneční soustavy - Slunce. Dosud největší objevenou hvězdou je *UY Scuti*, která je 1700 krát větší než Slunce. Příklad mlhoviny složené z hvězd se nachází na obrázku 2.5.
- **Planety** - Planeta je vesmírný objekt, který má svou oběžnou dráhu kolem nějaké hvězdy. Její objem je natolik velký, aby planeta udržela kulovitý tvar, ale ne tolik, aby v jejím jádře vznikla termonukleární fúze. Ve sluneční soustavě je takových objektů 8.
- **Satelitní objekt** - Objekt, který díky své oběžné dráze obíhá kolem planety. Satelitní objekty se dělí na dva druhy - přírodní, tedy nevytvořené člověkem (např. Měsíc) a vytvořené člověkem, tedy přímo satelity, které do vesmíru vysílají různé vesmírné organizace.



Obrázek 2.5: Mlhovina Medusa[2]

- **Komety** - Kometa je objekt složený z ledu, typicky obíhající velmi rozsáhlou oběžnou dráhu. Při zvýšení její teploty se začíná část povrchu odpařovat, což vytváří viditelnou atmosféru a efekt „ocasu“.
- **Asteroidy** - Asteroidy jsou vesmírné objekty podobné planetám, avšak svou velikostí nedosahují jejich rozměrů. Velké asteroidy se mohou nazývat „planetky“. Mezi nejznámější představitele patří *Pluto*, které roku 2006 přestalo být klasifikováno jako planeta. Menší asteroidy mohou tvořit planetární pásy, které okolo sebe svou gravitací tvoří například Saturn.
- **Meteory a meteority** - Meteor je vesmírný objekt, který za sebou zanechává znatelnou stopu, neboť se jedná o asteroid, který vstoupil do atmosféry a je přitahovaný gravitační silou dané planety na kolizní dráhu. Pokud je meteor dostatečně objemný aby překonal celý pád atmosférou a dopadl na povrch, kde vytvoří kráter, tak se nazývá meteorit.
- **Galaxie** - Galaxie je souhrn obrovského množství hvězd a slunečních soustav, které jsou držené gravitační silou středu galaxie, kde se s největší pravděpodobností nachází černá díra, která tuto gravitační sílu vytváří. Sluneční soustava se nachází v galaxii Mléčná dráha, která podle odhadů obsahuje zhruba 200 až 300 miliard hvězd. Příklad galaxie je na obrázku 2.6.

Noční obloha viditelná ze Země vypadá jako složená čistě z hvězd. Ne všechny objekty vyzařující světlo však skutečně hvězdami jsou. Proto se zavádí pojem „Nehvězdný objekt“

[2]Dostupné z https://apod.nasa.gov/apod/ap220325.html.

Obrázek 2.6: Arp 78: Galaxie v souhvězdí Berana[3]

(Nonstellar object), který zahrnuje vesmírné objekty, které vyzařují světlo jako hvězdy, ale hvězdami nejsou [9]. Mezi tyto objekty patří typicky galaxie nebo např. mlhoviny.

2.3 Generování náhodných čísel

Veškerá počítačová odvětví, nejen počítačová grafika, se potýká s velmi zásadním problémem - generování náhodných čísel. Běžnému člověku se tento problém může zdát velice primitivní, ale pro počítač je to téměř neřešitelný problém. Každý jeho výpočet musí vycházet z nějakých vstupních údajů a vytvořit opravdu náhodnou hodnotu nelze, pokud není založený na skutečných fyzikálních zdrojích náhodnosti. Takové generátory náhodných čísel jsou sice opravdu náhodné, ale také velmi pomalé a pro běžné používání by velmi výrazně zpomalovaly jakékoliv výpočty. Z tohoto důvodu se zavedl nový pojem - pseudonáhodné generátory čísel. Jedná se o matematické funkce, které s omezenými vstupy dokáží generovat čísla, která se na první pohled jeví jako náhodná, ale ve skutečnosti je každé další číslo jen výsledkem rovnice, která vychází z předchozího čísla, předem daných hodnot, nebo třeba systémového času. Díky tomu je možné simulovat náhodné čísla jen s velmi malou odchylkou od reálného světa, která je pro běžné používání naprosto zanedbatelná.

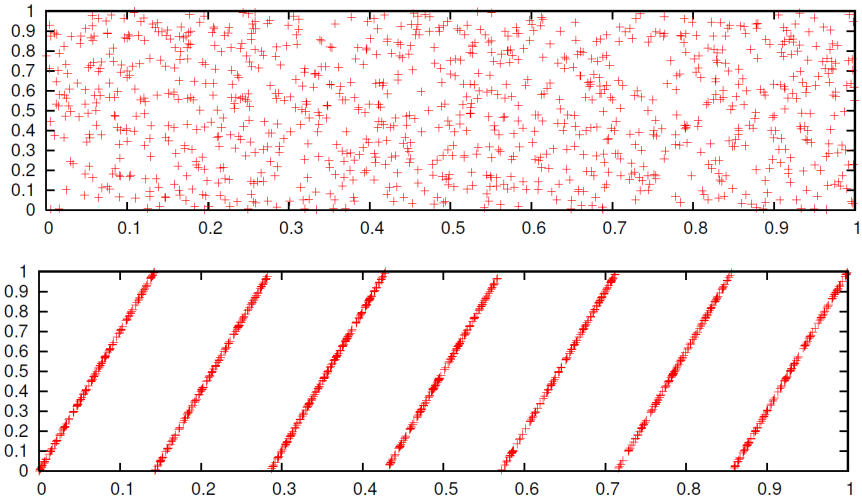
Jeden z nejstarších a nejjednodušších generátorů pseudonáhodných čísel je lineární kongruentní generátor. Je to jednoduchá rovnice:

$$x_{n+1} = (ax_n + b) \mod m \tag{2.1}$$

kde x_{n+1} je nově generované číslo, x_n je předchozí vygenerované číslo, a, b a m jsou konstanty na modifikaci generovaných čísel. Na základě této rovnice každé následující pseudonáhodné

[3]Dostupné z https://apod.nasa.gov/apod/ap220324.html.

číslo x_{n+1} závisí na třech vstupních konstantách, které ovlivňují celkové chování generátoru. Tyto konstanty musí být vhodně zvolené, nebo se ve vygenerovaných číslech relativně brzo objeví viditelný vzor, který se bude stále opakovat a tudíž nebude dosaženo kvalitního výsledku, což je znázorněno na obrázku 2.7. Pro kvalitní výsledky jsou nejznáměji zvolené konstanty $a = 69069$, $b = 1$ a $m = 2^{32}$ [8]. Jako výchozí hodnotu pro vypočítání prvního pseudonáhodného čísla lze použít např. systémový čas.



Obrázek 2.7: Příklady výsledků kvalitně (nahoře) a nekvalitně (dole) zvolených konstant lineárního kongruentního generátoru.[4]

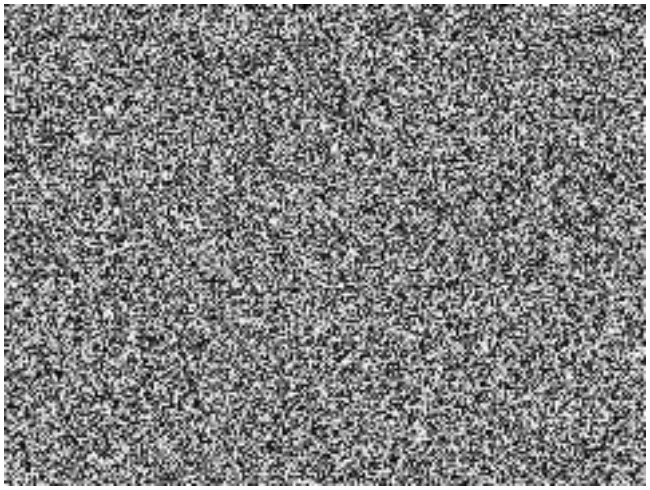
2.4 Šum

Základním problémem vytváření modelů na základě reálných objektů je nahodilost. V reálném světě se prakticky nevyskytují „dokonalé“ objekty, resp. objekty, které mají jednoduše popsatelný tvar, díky čemuž by tyto objekty bylo možné dokonale napodobit v počítačové grafice, neboť by k tomu stačilo několik matematických funkcí. Proto je potřeba v grafice tuto dokonalost, kterou je možné vytvořit, nějakým způsobem rozbít, aby vznikl výsledek, který je reálnému světu daleko více podobný. K tomuto účelu se používají funkce, které vytvářejí tzv. šum [2].

Základní stochastická funkce, kterou je k tomuto účelu možné využít se nazývá „bílý šum“ (white noise). Je to prostý generátor náhodných čísel, který není nijak ovlivněný jinými vstupy než těmi, které jsou potřeba k funkci takového generátoru. Slovy náhodný je myšleno pseudonáhodný, jelikož ve virtuálním světě je obtížné dosáhnout dokonalé náhodnosti. Takovou náhodnost promítnutou v podstatě na 2D texturu lze pozorovat na šumu ve starých analogových televizích (příklad na obrázku 2.8). Pokud se tato televize špatně naladila, byl viděn šum, který se dá připodobnit textuře tvořené pomocí právě takové funkce, kdy pro každý bod je vygenerován náhodná hodnota mezi danými hranicemi [3].

Ani toto chování ale v modelování a texturování není žádoucí. Kdyby byl tento bílý šum použitý při tvorbě modelů a textur, tak by díky své absolutní nezávislosti na vstupech mohl výsledný model nebo textura měnit body, které byly s pomocí šumu vytvořeny, při

[4]Převzato z prezentace k přednáškám z předmětu Modelování a simulace na VUT FIT (Peringer Petr a Hrubý Martin).



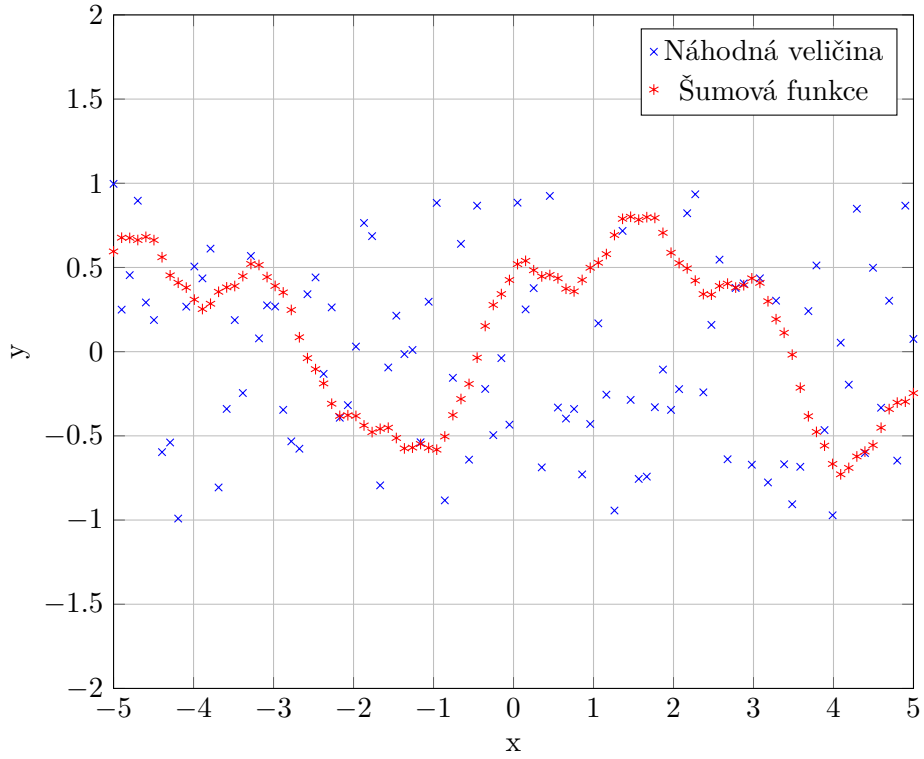
Obrázek 2.8: Příklad bílého šumu na 2D textuře.

každé změně snímku. Proto je potřeba vytvořit funkci podobnou bílému šumu, která ale bude závislá na hodnotě, jenž bude v čase neměnná. Pro tento účel byla zvolena pozice daného bodu [3]. Tato pozice ovšem nesmí být absolutní, protože pokud by se za takového stavu změnila pozice modelu nebo textury jako celku, nevyhnutelně by to změnilo i hodnoty šumu v daných bodech. Proto se používá pozice relativní, která je vztažena ke konstantnímu bodu v rámci modelu, obvykle středu. Takto se hodnota bude měnit jen v momentě, kdy se mění pozice bodů v rámci modelu, a nikoliv pokud se bude měnit pozice modelu jako celku v rámci nějaké scény.

S takto vyřešenými vstupy se může vyřešit obsah funkce, která bude tvořit použitelný šum. Jde o funkci, která po zadání vstupní hodnoty navrací číslo mezi <0;1> (některé specifické druhy vrací hodnoty mezi <-1;1>) a stejné číslo bude navracet v jakémkoliv stavu po zadání stejných vstupů. Ani takto vylepšený bílý šum však není úplně to, co je v modelování a texturování často požadováno. Problém nastává při „přiblížení“ šumu. V rámci přiblížení dochází ke zvýšení přesnosti bodů, pro které šum je šum počítaný. Pokud se pozice bodu změní, tak se zákonitě změní i výsledný šum v tomto bodě. Pokud je používaný generátor náhodných čísel kvalitní, tak se hodnota i pro sebemenší změnu vstupu velmi změní [3]. Proto je potřeba používat funkci, která pro relativně vzdálené body vrací většinou velmi odlišnou hodnotu, ale pro velmi blízké body bude výsledná hodnota podobná. Pro blízké body platí, že čím bližší jsou body, tím bližší bude jejich hodnota a tudíž bude výsledné vykreslení bodů na jedné ose připomínat více plynulou křivku, než čistě náhodnou veličinu, jak je naznačeno na obrázku 2.9.

Funkce, která umožní generování takového šumu se dá rozšířit do více rozměrů. Je možné vytvářet šum v jednodimenzionálním, v třídimenzionálním tak i v desetidimenzionálním prostoru. Pro tvorbu šumu se využívají dva základní přístupy (grafické příklady na obrázku 2.10):

- Hodnotový šum (value noise) – nejdříve se vyhodnotí pseudonáhodné hodnoty pro každý bod v dané mřížce pro tvorbu šumu a následná funkce, pro vyhodnocování vstupních bodů je vytvořena pomocí interpolace těchto bodů – sousední body v mřížce mají tudíž podobnou první derivaci
- Gradientní šum (gradient noise) – na rozdíl od hodnotového šumu se nejdříve vytvoří pseudonáhodné vektory pro každý bod v dané mřížce pro tvorbu šumu. Vektor je



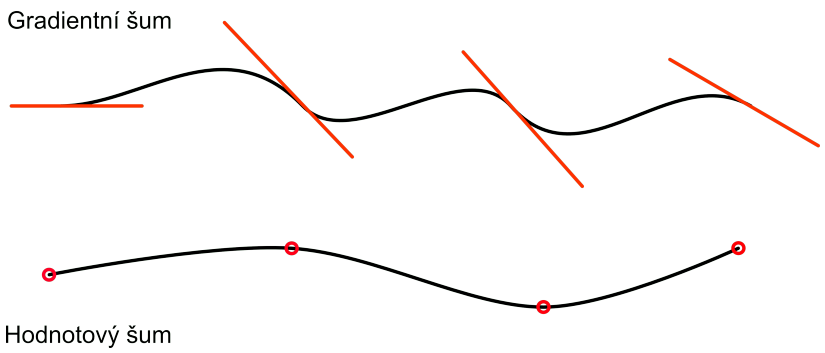
Obrázek 2.9: Znázornění rozdílu mezi náhodnou veličinou a šumovou funkcí při generování čísel.

na rozdíl od bodu definován dvěma rozšiřujícími údaji - směrem a velikostí, díky čemuž každý vektor v mřížce ovlivňuje výslednou křivku jiným způsobem než prostý bod. Následná funkce, pomocí které se poté vyhodnocují vstupní body je vytvořena pomocí interpolace těchto vektorů. Poddruh gradientního šumu je i Perlinův šum, který představil roku 1985 Ken Perlin [6].

2.4.1 Perlinův šum

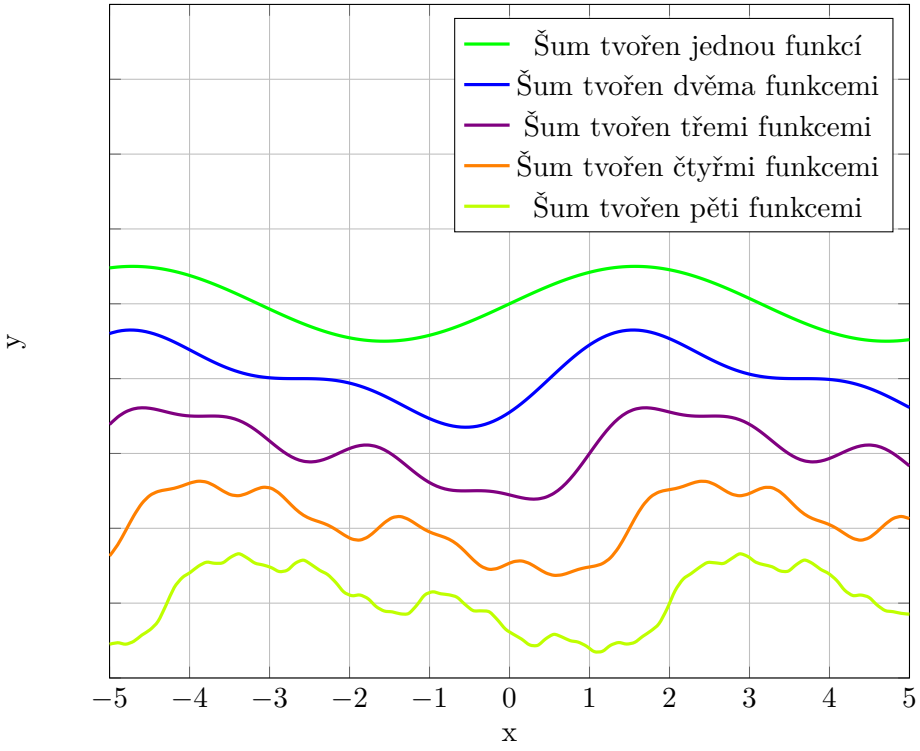
Jak již bylo zmíněno, Perlinův šum byl představen roku 1985 a následně použit ve filmu Tron, díky čemuž jeho autor Ken Perlin získal roku 1997 ocenění Academy Award for Technical Achievement [3]. V základu je to obecná funkce gradientního šumu, díky čemuž mají výsledné modely a textury po použití Perlinova šumu daleko více vzhled přírodní struktury a tedy se mnohem více podobají reálným předlohám. Pro využití v praxi je často využíván Perlinův šum ve dvou nebo třech dimenzích, který je používán pro tvorbu nejrůznějších přírodních tvarů, jako například mraků, krajiny a povrchů nejrůznějších objektů. Perlinův šum je sice v základě gradientní šum, ale samotná tato vlastnost by nestačila k dosáhnutí výsledného organického vzhledu vygenerované křivky/plochy. K tomuto efektu je potřeba použití několika šumových funkcí, které tvoří šum dohromady.

Nelze ovšem použít více takových funkcí s podobnými vstupními vlastnostmi, v takovém případě by totiž opět vznikl jen další „neorganický“ šum. Jednotlivé šumové funkce se musí lišit ve dvou základních vlastnostech - frekvenci a amplitudě. Amplituda je tzv. *intenzita funkce* a získá se rozdílem globální maximální a minimální hodnoty šumové funkce. Frekvence je obrácená hodnota vzdálenosti mezi jednotlivými body, která byly použity pro



Obrázek 2.10: Rozdíl mezi gradientím a hodnotovým šumem. Tvar gradientního šumu je řízen tečním vektorem na rozdíl od tvaru šumu hodnotového, který je tvořen interpolací bodů.

interpolaci dané šumové funkce. Na frekvenci poté přímo závisí hladkost průběhu šumové funkce. Tuto vlastnost sdílí se základními goniometrickými funkcemi. Čím více dílčích funkcí o vzrůstající frekvenci a klesající amplitudě se spojí do výsledného šumu, tím více organický bude jeho vzhled, což je znázorněno na obrázku 2.11.



Obrázek 2.11: Znázornění závislosti šumu na počtu funkcí, kterými je tvořený.

V dnešní době je Perlinův šum jedna z nejpoužívanějších variant jak simulovat organické tvary. Jsou na něm postavené i další varianty, které se snaží jej dále vylepšit. Mezi takové patří například simplexní šum (simplex noise), který se zaměřuje na zjednodušení Perlinova šumu a zejména pak na zmenšení potřebné výpočetní režie, která se začínala projevovat zejména při tvorbě šumu ve vyšších dimenzích.

Tvorba Perlinova šumu

Existují dva základní způsoby jak je možné vytvořit Perlinův šum - matematicky a graficky. Oba způsoby jsou si ve svém jádru velmi podobné, ale mohou se zdát naprosto odlišné.

- **Matematický způsob** - lze poměrně jednoduše vyjádřit spojení šumových funkcí jako prostý součet hodnot. Pokud se určí, že **frekvence** $= f$ a **amplituda** $= a$, pak lze napsat, že hodnota šumové funkce v daném bodě $x = a{\cdot}Noise(f{\cdot}x)$. Kdy konstanty a a f ovlivňují výslednou hodnotu úplně stejně jako u kterékoliv jiné matematické funkce. Za předpokladu, že amplituda se s každou úrovní snižuje a frekvence naopak zvyšuje se tedy dojde k výsledku, že:

$$\textbf{PerlinNoise}(x) = \sum_{i=0}^{n} a^{-i} \cdot Noise(f^i \cdot x) \tag{2.2}$$

  kdy n udává celkový počet dílčích šumových funkcí, které jsou použity pro stavbu daného Perlinova šumu, a^{-i} udává amplitudu a f^i frekvenci dané šumové funkce. Dílčí šumovou funkci lze taktéž nazvat oktávou.

- **Grafický způsob** - tento postup lze nejlépe popsat ve dvou rozměrech. Nejdříve je potřeba na ploše vyznačit mřížku z jednotlivými uzly, které budou mít mezi sebou jednotnou vzdálenost. Tyto body slouží jako jakési „záchytné body“, podle kterých se bude určovat hodnota šumu okolních bodů v rámci jednotlivých dílů mřížky. Pro každý takto určený bod je poté vygenerován pseudonáhodný dvourozměrný vektor, který bude sloužit jako základ pro šum. Následně je pro každý bod mezi těmito uzly vypočítán skalární součin (dot product), pro jeho nejbližší čtyři uzly. Poté, co je skalární součin vyhodnocený pro všechny body potřebné plochy, tak je celá plocha „uhlazena“ pomocí interpolace a výsledné hodnoty jsou přidělené jednotlivým bodům. Grafický postup na obrázku 2.12. Díky kombinaci skalárního součinu a interpolace je zajištěna dostatečná náhodnost, stejně jako jistota, že sousedící body budou mít s největší pravděpodobností podobnou výslednou hodnotu.

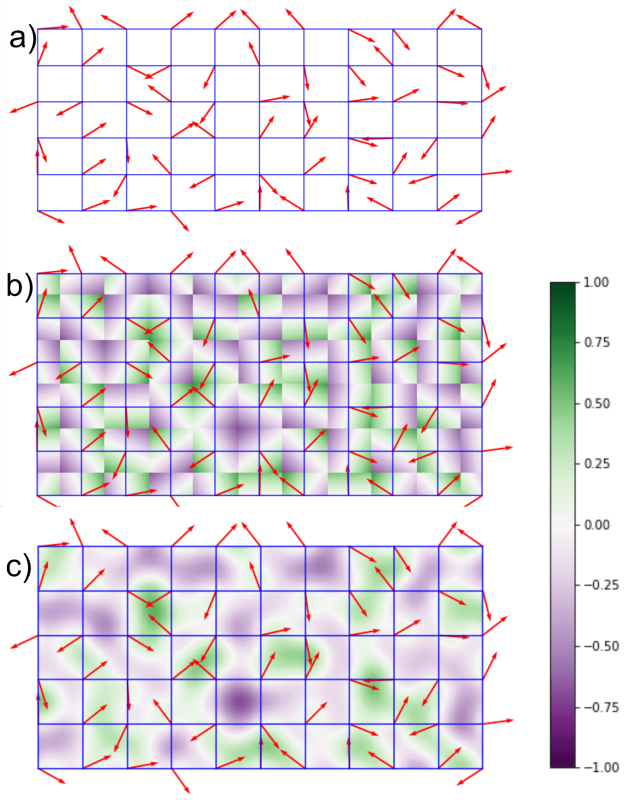
2.4.2 Tvorba terénu

Pro vylepšení věrohodnosti, zejména u tvorby povrchů/terénů je používána kombinace několika šumů. Tyto šumy jsou tvořeny rozdílnými frekvencemi (rozložení bodů na mřížce při počítání vektorů) a rozdílnými amplitudami (určuje množství vlivu aktuálního šumu na celkový výsledek). Pro tyto účely se používají dva další pojmy - persistence a lacunarita. Persistence určuje velikost amplitudy a lacunarita určuje velikost frekvence.

Typický výsledný šum je pak skládán z předem daných dílčích šumů – tzv. oktáv. Všechny oktávy jsou ve výsledku sečteny do výsledného šumu. S postupem tvorby šumu do následující oktávy je vydělena aktuální amplituda koeficientem persistence a frekvence je vynásobena koeficientem lacunarity. Tímto je dosaženo, že každá následující oktáva je „detailnější“ a má menší vliv na výsledný šum. Celý postup je velmi podobný tvorbě samotného dílčího šumu. Zde je ovšem jako vstupní dílčí funkce použitý celý šum, nikoliv jen jeho dílčí šumová funkce.

Pro představu si je možné předvést např. šum o třech oktávách (obrázek 2.13 demonstruje šum o čtyřech oktávách):

- 1. oktáva – Obecný tvar terénu – nížiny a pohoří.



Obrázek 2.12: Postup při grafickém vytváření Perlinova šumu. a) Určení vektorů b) Skalární součin c) Interpolace[5]

- 2. oktáva – Lokální kopce vrcholy a prohlubně v obecném tvaru terénu.
- 3. oktáva – Znatelné kameny, díry a různé menší detaily.

Takto lze pokračovat v podstatě do nekonečna, resp. do takového detailu, dokud nejsou rozdíly zanedbatelné.

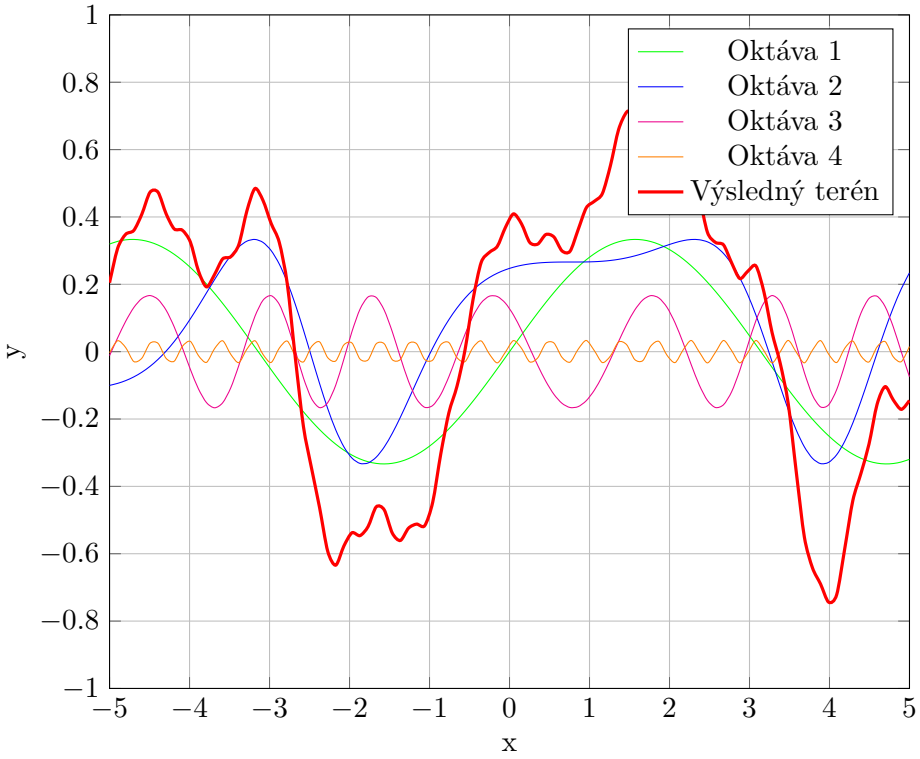
2.5 Generování kráterů

Pro modelování měsíců, asteroidů, případně některých částí samotných planet je potřeba modelovat krátery po dopadu jiných objektů.

2.5.1 Tvar kráterů

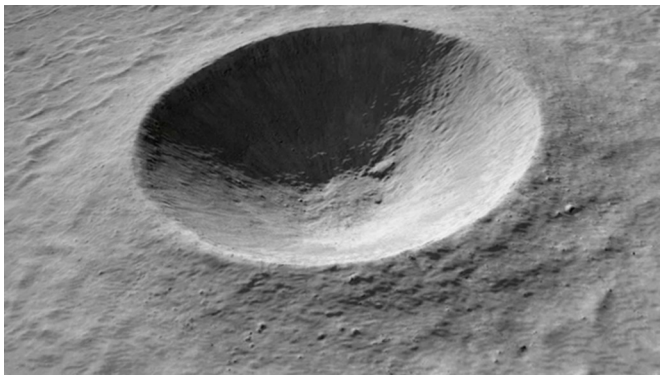
Každý kráter se skládá ze tří základních částí: samotné prohlubně v centru kráteru, která je vytvořena silou po samotném dopadu cizího tělesa, vyvýšené hrany kráteru, kterou vytvořila nárazová vlna po dopadu onoho objektu a množství malých kráterů, které vznikly oddělením a odmrštěním částí dopadajícího objektu při dopadu do blízkosti kráteru. Samotný kráter má pak ještě vyvýšené dno, které se postupem času zanáší okolním povrchem

[5]Převzato z https://en.wikipedia.org/wiki/Perlin_noise



Obrázek 2.13: Znázornění jednotlivých oktáv terénu a jejich vliv na výsledný terén.

planetky/měsíce a samotný kráter se tedy s postupem času stává méně výrazný, jak je vidět na obrázku 2.14. Tento proces ovšem zabírá v reálném čase miliony až miliardy let [7].



Obrázek 2.14: Příklad jednoduchého kráteru.[6]

2.5.2 Velikost kráterů

Krátery se dají rozdělit do několika kategorií podle velikosti, na niž pak záleží jejich obecný tvar [7]. Tyto kategorie jsou:

- Jednoduché krátery – typicky do 10 až 20 kilometrů v průměru, mají „miskovitý tvar“ a skoro zanedbatelné dno kráteru.

[6] Dostupné z https://svs.gsfc.nasa.gov/10792.

- Složité krátery – nad hranici průměru jednoduchých kráterů, mají znatelně strmé stěny a daleko výraznější dno, díky své velikosti se totiž daleko rychleji usazuje zbytek zvětralého povrchu na původním dně těchto kráterů.
- Vícekruhové krátery – tyto krátery mají typicky několik set kilometrů v průměru a skládají se z několika soustředných kruhů.
- Sekundární krátery – jak již bylo zmíněno dříve, tyto krátery jsou typicky malé a vznikají vymrštěním uvolněného materiálu z nárazu většího objektu do okolí centrálního kráteru. Mají typicky kruhovitý tvar díky menší dopadové rychlosti.

2.6 Blender

Blender[7] je open-source aplikace vytvořena k tvorbě, úpravě a vykreslování 3D scén. Celá aplikace je multiplatformní, a tudíž funguje stejně dobře na operačních systémech Windows, Linux a Mac. V současnosti se tato aplikace řadí mezi aplikace na vrcholu 3D sad nástrojů.

Blender obsahuje širokou škálu nástrojů a možností, jak vytvořit požadovaný výsledek. Umožnuje vytváření modelů, textur, animací, a nebo například střih videa. Pro specifické účely práce s aplikací Blender jsou vytvořené editory, které splňují konkrétní požadavky na danou práci. Pro práci s obrázky slouží *Image* editor, pro práci s animačními křivkami slouží *Graph editor*, pro texturování zase *UV editor*. Nejčastěji se ovšem používá *3D Viewport*, který umožnuje pracovat s objekty v konkrétní 3D scéně. Každá konkrétní scéna je složená z minimálně jednoho objektu. Každý objekt má přesně danou pozici v prostoru, rotaci a velikost. Dalšími nezbytnými částmi každého objektu jsou další vlastnosti, jako seznamy vrcholů, hran a stran, textury a různé modifikátory, které upravují výsledný vzhled objektu.

Pracovat s objekty lze v různých režimech, kdy každý režim má vlastní účel, ke kterému slouží. Hlavní dva režimy, které se používají pro práci s objekty jsou:

- Object mode (Objektový režim) – slouží k práci s objektem jako takovým. To znamená, že umožňuje jeho přemístění v prostoru, rotaci a změnu velikosti, stejně jako aplikování dříve zmíněných modifikátorů. Dále umožňuje jednoduše přidávat, mazat nebo např. duplikovat objekty. Jako „nejmenší stavební jednotku“ je zde brán právě objekt.
- Edit mode (Editační režim) – slouží k úpravě konkrétního objektu, přesněji jeho geometrie. Lze pracovat základní stavební kameny každého modelu – vrcholy, hrany a strany. Jako „nejmenší stavební jednotka“ je zde vrchol (vertex).

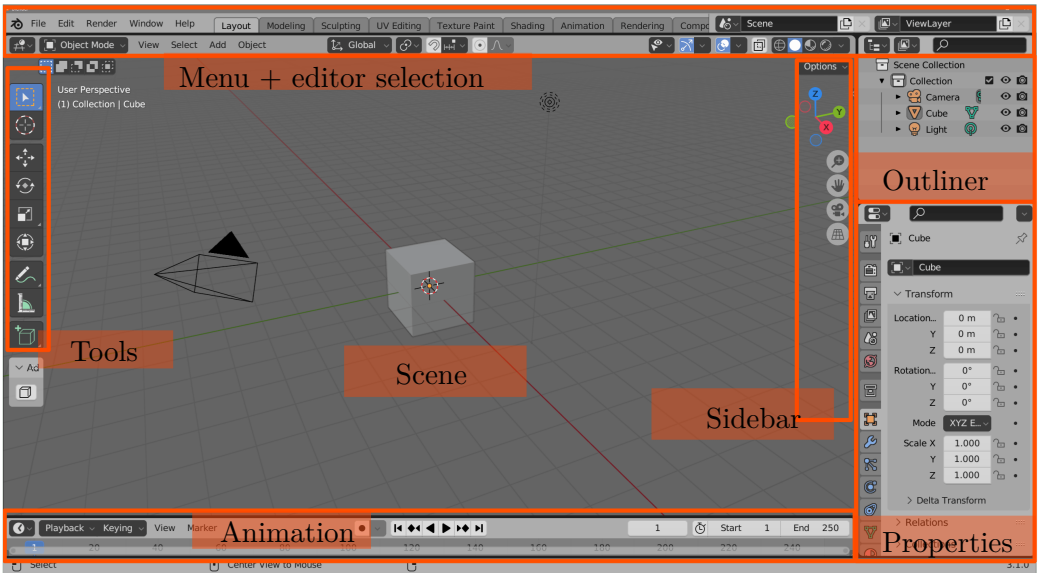
2.6.1 User Interface

Základem ovládaní programu Blender je velmi upravitelné grafické uživatelské rozhraní. Celé uživatelské rozhraní je rozdělené do čtyř hlavních částí, jak je znázorněno na obrázku 2.15:

- Scene view (scéna) – zde je v 3D prostoru zobrazena celá scéna a objekty v ní. Součástí scény jsou taktéž nástroje aktuálního režimu (Tools) a sidebar, kde jsou dodatečné informace, funkce a často panely a funkce add-onů.
- Outliner – zde je seznam všech objektů a celé hierarchie, které obsahuje scéna

[7] Program Blender je dostupný z https://www.blender.org/

- Properties – sekce na úpravu nejrůznějších vlastností scény a aktuálního objektu
- Animation – sekce se základní funkcionalitou pro animaci scény



Obrázek 2.15: Základní popis uživatelského rozhraní aplikace Blender.

2.6.2 Python API

Pro náročnější uživatele Blenderu je program vybaven rozhraním, skrze které je programátor schopný vytvořit skript, který přidá do Blenderu novou funkcionalitu. Blender Python API je založen na verzi Pythonu 3.x. Vytváření skriptů pro Blender je umožněno pomocí importovaného balíčku `bpy`. Mezi dříve zmíněnými editory je taktéž textový editor, který umožňuje právě tvorbu skriptů pro Blender. Skripty napsané v tomto editoru lze pak jednoduše spustit v integrované konzoli Blenderu, kterou lze samotný Blender z velké části ovládat. Druhá možnost je napsaný skript uložit a dále naimportovat jako add-on, který lze pak jednoduše spustit, kdykoliv je potřeba. Externě napsané skripty podporují více zdrojových souborů, které se importují jako komprimovaný soubor. Díky tomu je možné přidávat velmi komplexní skripty a skutečně si tak Blender přizpůsobit dle vlastních potřeb.

K balíčku bpy se přistupuje zejména skrze jeho moduly. Mezi nejpoužívanější patří:

- `bpy.data` – Přístup k datům, jejich tvorba, změna, přístup k atributům a kolekcím. Např. `bpy.data.objects["Icosphere"]`
- `bpy.context` – Přístup ke kontextu, který je právě vybrán uživatelem, aktivní scéně, nastavení různých nástrojů.
- `bpy.ops` – Přístup ke všem operátorům aplikace. Jsou typicky vyvolány stiskem tlačítka nebo spuštěním konkrétního operátoru z příkazové řádky. Každý operátor má dále parametry, které ovlivňují jeho výsledné chování.
  Např. `bpy.ops.mesh.primitive_cube_add()`.
- `bpy.props` – Přístup k existujícím typům a jejich úpravu. Pro tvorbu nových datových skupin a proměnných dané scény.

- `bpy.types` – Přístup k definicím typů, tříd. Definice pro rozšíření uživatelského rozhraní, tvorbu vlastních operátorů, atd.
- `mathutils` – Přístup k různým matematickým operacím (např. Perlinův šum) a potřebným datovým typům (např. Vektor).

Vytvoření daného rozšíření spočívá ve využití definovaných tříd v balíčku bpy, s přidáním vlastní funkcionality ve formě výpočtů a vstupů uživatele. Detailní popis informací pro práci s Blender Python API se nachází v dokumentaci [1].

2.6.3 Třída

Třída v Blenderu funguje v základu stejně jako třída v jakémkoliv jiném objektově orientovaném jazyce. Pro zajištění správné funkčnosti doplňků a API byly ve třídách pro Blender zavedeny doplňující informace/proměnné, které jsou vyžadované při registraci dané třídy. Naprosto nezbytnou částí pro registraci dané třídy je proměnná `bl_idname`, která slouží jako identifikátor v rámci celé aplikace. Díky této proměnné lze přistupovat k celé třídě, kterou reprezentuje a dále tuto třídu používat. Například pro umístění operátoru do menu pomocí funkce uvnitř panelu - `self.layout.menu(Operator_class_name.bl_idname)`. Mezi obecně používané proměné patří také `bl_label` (obsahuje název třídy, který je zobrazen v menu na panelech nebo kdekoliv v UI) a `bl_description` (obsahuje detailnější popis operátoru/ui prvku, který není zobrazen pokaždé, ale ulehčuje uživateli orientaci v UI). Takovýchto proměnných je daleko více (`bl_options, bl_parent_id, bl_space_type, atd.`), ale používají se pro specifické účely nebo jsou implementované jen pro určité dědičné třídy z této obecně definované třídy.

2.6.4 Operátor

Operátror je třída, která představuje souhrn operací s vnitřními daty, které jsou používány v Blenderu a které ve výsledku představují modelovanou scénu. Konkrétní operátor je v základu třída, která je zároveň podtřídou `bpy.types.Operator`, která zároveň teoreticky rozšiřuje obecnou třídu pro Blender API. Kromě standardních proměnných potřebných pro každou registrovanou třídu, je v případě operátoru ještě nutno uvádět proměnou `bl_options`, kde je ve většině případů (stejně jako v rámci implementace) využito nastavení *REGISTER a UNDO*, které jsou potřebné pro úspěšné zobrazení redo panelu, potřebného pro úspěšné vytváření vesmírných objektů, jelikož je využíván na předávání vstupních proměnných operátoru.

Invokace objektů je možná dvěma způsoby - operátor spuštěný kódem, typicky je potřeba aktuální operátor pro úspěšné dokočení operátoru druhého, a nebo spuštění uživatelem. Uživatel může také spustit operátor dvěma způsoby. Buď je odkaz (tlačítko) zabudován v uživatelském rozhraní a stačí na něj klinout, nebo je potřeba operátor spustit z Python konzole zadáním jeho identifikátoru v kompletní délce.

U vybraných operátorů v rámci implementace byla využita dědičnost z druhé třídy. Touto třídou je `AddObjectHelper` a byla využita u operátorů, jejichž účelem je přidávat objekty do 3D scény. Od této třídy operátory dědily schopnost automaticky zobrazovat na panelu pro vstupní parametry základní vstupy jako polohu, rotaci a měřítko, díky čemuž je přidávání nových objektů o něco přehlednější a jednodušší.

Názvy operátorů

Z důvodu častých duplicitních názvů u uživatelsky vytvořených tříd byly ve verzi 2.8 přidány konvence pro vytváření názvů těchto tříd. Tyto názvy se skládají za tří částí - názvu skupiny do které třída patří (nebo libovolný řetězec v `UPPERCASE`), separátoru názvu, kterým by měl být jedním z pěti typů (HT - header, MT - menu, OT - operátor, PT - panel, UL - UIList) a samotného názvu třídy. Výsledný název pak je ve tvaru - `CUSTOM_PREFIX_-{separator}_class_name()`.

2.6.5 Panel

Panel je třída, která je základním prvkem při vytváření jakékoliv nové části uživatelského rozhraní. Jakýkoliv specifický panel je třída, která je zároveň podtřídou `bpy.types.Panel`, která, stejně jako třída pro operátory, rozšiřuje výchozí obecnou třidu Blender API. Tato třída, na rozdíl od jiných tříd definovaných uživatelem, nepotřebuje definovanou proměnnou `bl_idname` pro správnou funkčnost. Tato proměnná je vyžadována pouze v případě, že identifikátor tohoto panelu bude potřeba pro dokončení panelu jiného. Tento případ může nastat, např. pokud bude tento panel brán jako rodičovský a bude v sobě mít jeden nebo více synovských panelů. V těchto panelech je poté potřeba nastavit proměnnou `bl_parent_id` na identifikátor rodiče.

Dalšími proměnnými, které jsou potřeba pro správnou funkčnost panelu, jsou proměnné `bl_space_type` a `bl_region_type`. První zmíněná proměnná určuje v jakém regionu bude panel zobrazen. Mezi regiony patří např. *ui, navigační panel* nebo *nástroje*. Druhá proměnná pak uchovává část uživatelského rozhraní, kde bude panel vykreslen. Panel díky tomu může být zobrazen ve kterémkoliv editoru, 3D zobrazení scény nebo např. v konzoli pro Python.

Metody

Jednotlivé třídy pak obsahují několik metod, které volá vnitřní systém Blenderu a specifikuje chování těchto tříd. Značná část funkčnosti celého doplňku je vytvořena právě v těchto metodách.

- `draw` - Díky přepsání výchozího tvaru této funkce lze specifikovat vlastní uživatelské rozhraní. Využívané zejména u panelů, případně u definice vlastního menu. Jednotlivé prvky (box, column, row, operator, menu, atd.) se můžou přidávat voláním konkrétních funkcí nad proměnnou daného panelu - `self.layout`.
- `poll` - Metoda kontrolující, jestli jsou splněny všechny prerekvizity pro spuštění hlavních metod třídy. Například metoda operátoru `execute` se nikdy nespustí, pokud selže tato metoda.
- `invoke` - Metoda, která je volána při interakci uživatele s uživatelským rozhraním. Zaznamenává vstupy jako klávesnici a pohyby myši.
- `execute` - Metoda, která specifikuje hlavní chování a definici operátoru. Je volaná po dokončení metody `invoke`, při undo/redo nebo při spuštění operátoru z Python konzole. Operátor je spuštěn se vstupy zadanými uživatelem, pokud nejsou zadané, použijí se výchozí hodnoty. Pro tuto metodu jsou určeny specifické návratové hodnoty, typicky `FINISHED` při úspěšném dokončení, jinak `CANCELLED`.

- `modal` - Metoda používaná u operátorů, které vyžadují neustálé spuštění. Jsou spuštěny do té doby, než navrátí hodnotu `FINISHED`.
- `cancel` - Tato metoda je volaná po ukončení metody `modal`.

Vlastnosti

Vstupní informace pro jednotlivé operátory nebo informace zobrazené na panelech mají formu vlasntností vstupů/vlastností (Properties). Standardnímu operátoru je možné specifikovat polohu, rotaci a měřítko, ale při tvorbě vlastních operátorů je možné tyto vstupy libovolně rozšířit. Tento panel lze rozšířit o několik typů těchto vlastností:

- **Integer Property** - Nejjednodušší typ vlastnosti, která obsahuje celé prosté číslo - u operátorů často používaná pro přesný počet jednotek - např. konkrétně v tomto doplňku pro určení počtu asteroidů v asteroidovém poli.
- **Float Property** - Vlastnost pro číslo s pohyblivou čárkou, je často používané pro určení nějakého faktoru. Doplňkem využíváno například pro určení procenta plochy planety, která bude tvořit vodu.
- **String Property** - Vlastnost pro přijímaní vstupního řetězce. Používá se především pro získání chtěného názvu objektu nebo například pro získání cesty k souboru, který je poté používán.
- **Boolean Property** - Prostý *true* a *false*. Používané často jako určení, zda se má část operátoru vykonat či nikoliv. Konkrétně v tomto doplňku určuje zda bude povrch planety upraven šumem nebo ne.
- **Vector Property** - Rozšiřuje vlastnosti type `Integer`, `Float` nebo `Boolean` do více rozměrů a mění je na vektory. Vektorové vlastnosti jsou používané Blenderem už ve výchozím stavu a to na určení konkrétní polohy, rotace a měřítka každého objektu v každé scéně v 3D prostoru. Počet dimenzí vektoru určuje parametr *size*. Tímto doplňkem používané k načtení barev operátory.
- **Enumeration Property** - Tento typ vlastnosti slouží jako výběr, neboli *menu*. V podstatě se jedná o vlastnost typu Boolean, ale rozšířenou o více možností. U operátorů se dá využít např. pro určení, který specifický podtyp je vyžadováno použít. Je zobrazována jako *dropdown menu*. V doplňku používána k přepínání panelů.
- **Collection Property** - Jedná se o vlastnost, která rozšiřuje výše zmíněně a umožnuje z nich vytvářet pole o jednotném typu.

Každá vlastnost má navíc své vlastní paramentry, které přesněji určují chování vlastnosti. Mezi nejtypičtější parametry patří `name` (určující název parametru - není to ale identifikáror), `description` (detailnější popis parametru), `min, max` (omezení hodnot, které parametr může nabývat) a `default` (výchozí hodnota parametru).

Většina těchto vlastností má ještě podtypy. Tyto podtypy upravují vstupní formát dané vlastnosti mnoha různými způsoby. Pokud se u vlastnosti typu *FloatVectorProperty* nastaví parametr *subtype* na hodnotu *COLOR* (barva), tak jako vstup bude namísto tří (záleží na počtu dimenzí daného vektoru) float hodnot zobrazeno pole pro výběr barvy.

Mezi nejčastěji přidávané vlastnosti tímto doplňkem patří úroveň detailu objektu (Integer Property) nebo faktor šumu (Float Property), který bude ovlivňovat jak moc šum změní samotný povrch daného objektu.

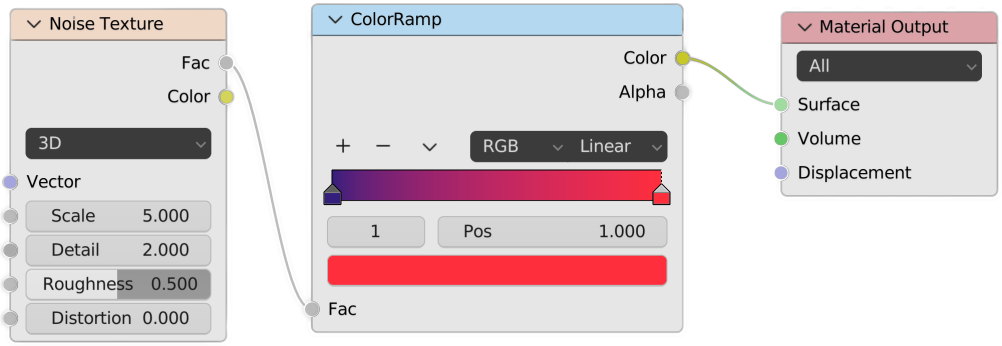
2.6.6 Materiály

Materiály jsou vestavěná funkcionalita Blenderu, která proces úpravy vzhledu povrchu velmi vylepšila. Místo přiřazování barev vrcholům je možné přiřadit daný materiál polygonu. Každý polygon může mít vlastní materiál. Je možné také přiřadit materiály celému objektu. V tom případě je zobrazen pouze materiál, který je označen jako aktivní.

Materiál je v podstatě souhrn více funkcí nebo procedur, jenž popisuje jak se pro daný bod určují jeho výsledné vlastnosti jako je barva, odrazivost světla nebo např. průhlednost. Tvorba materiálu je velmi grafická a intuitivní. Je pro ni používaná další funkcionalita Blenderu a sice Shader Nodes.

2.6.7 Shader Nodes

Shader nodes slouží k relativně jednoduché tvorbě povrchu objektů. Celý proces je v podstatě funkce, která z volitelných vstupů pro daný bod počítá jeho vlastnosti. Celá tato struktura je složena z uzlů (Node), kdy každý uzel představuje jistou operaci. Mezi takové operace patří např. získání polohy bodu, rozdělení vektorů na složky x, y a z, nebo např. převod číselné hodnoty na barvu pomocí grafické osy. Jednoduchý příklad je zobrazen na obrázku 2.16.



Obrázek 2.16: Příklady uzlů. Šumovový uzel (zleva), uzel pro převod hodnoty na barvu, uzel pro výstup materiálu.

Tyto uzly se pak propojí korespondujícími vstupy a výstupy, a výsledný materiál je potom připojen do uzlu z názvem *Material Output* (Výstup materiálu). Tento uzel reprezentuje vlastnosti, ovlivňující vzhled všech bodů polygonu, u kterých bude daný materiál zvolený jako aktivní.

2.7 Existující řešení

V této sekci jsou popsány existující produkty, které se zabývají řešením podobné problematiky.

2.7.1 True-Space

Jeden z nejlepších doplňků, co se týče tvorby vesmírných prostředí, je TrueSpace.[8] TrueSpace není freeware ani open-source. TrueSpace umožnuje tvorbu různých záběrů na

[8] True-Space je dostupný z https://blendermarket.com/products/true-space.

vesmírné objekty (příklad na obrázku 2.17). Umožňuje přidat galaxie, planety nebo např. prostý pohled na vesmír. Tyto scény jsou vytvořeny v kvalitě od 4k do 18k v závislosti na přání uživatele nebo schopnostech stroje, na kterém je Blender aktuálně spuštěný. Takto přidané scény umožnují rozmanité nasvětlení objektů, nejrůznější rotace, posuny, kontrasty a expozice.

Všechny objekty, ať už planety nebo galaxie jsou ovšem pouze 2D iluzí v rámci scény. Veškerá tvorba je vytvořena na materiálu, kterým je tvořeno pozadí aktuální scény v Blenderu. To má za následek několik limitací v rámci tvorby požadované scény. Například pokud by uživatel chtěl vytvořit animaci „obletu“ galaxie, tak toho nemůže docílit. Tento fakt se snaží nahradit již výše zmíněným osvětlením, které velmi dobře tvoří iluzi, že na scéně jsou skutečně objekty, které nějaké světlo vyzařují. Další faktor je přidávání různých planet na scénu. Díky tomu, že vše je na materiálu pozadí, tak by tato modifikace byla velmi komplikovaná (i když v rámci doplňku jako takového možná). Na podobné účely je tento doplněk tudíž nevhodný.

Obrázek 2.17: Ukázka z dopňku True-Space.

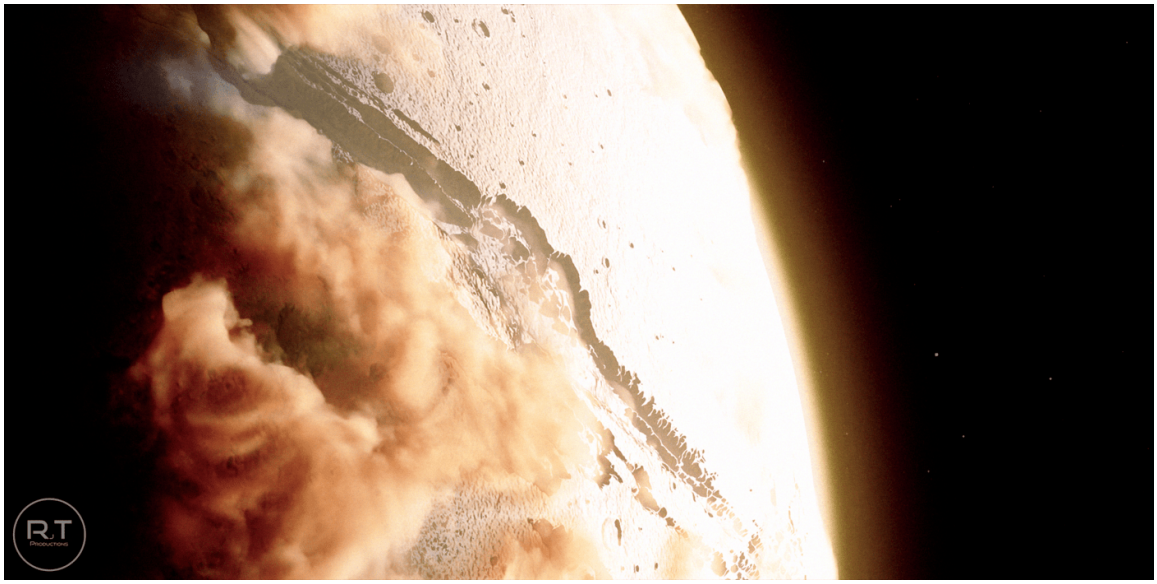
2.7.2 True-Earth/Mars

Tyto doplňky jsou od stejného autora jako výše zmíněný TrueSpace, avšak na rozdíl od něj používají trochu odlišný přístup a modelují jen specifický typ objektů (True-Earth[9] modeluje jen planety podobné Zemi a True-Mars[10] jen planety podobné Marsu). TrueSpace ukládá celou scénu do jednoho materiálu, který je následně uložený do pozadí scény. V případě těchto dvou doplňků je vytvořen objekt, v jehož povrchu je uložen materiál, ve kterém je veškerá tvorba textury dané planety. Doplněk je velice detailní, zejména co se týče povrchu planety. Umožnuje nejrůznějšími způsoby upravovat oceány, mraky, celou atmosféru, jednotlivé odrazy světla v závislosti na místě odrazu atd. Na rozdíl od předchozího doplňku lze s modely jakkoliv manipulovat v prostoru, a tvořit tak rozmanitější rendery nebo animace (příklady na obrázku 2.18).

[9]True-Earth je dostupný z https://blendermarket.com/products/true-earth
[10]True-Mars je dostupný z https://blendermarket.com/products/true-mars

I tyto doplňky mají ovšem své limitace. Jak už vypovídá název, tak oba modelují jen dvě planety a to Mars a Zemi. Mohou se pomocí nich tvořit planety které nejsou Mars ani Země, jsou těmto planetám velmi podobné, ale nelze modelovat planety odlišné. Stejně jako předchozí doplněk i tento nabízí velmi kvalitní textury a obecně kvalitní výkon.

(a) True-Earth



(b) True-Space

Obrázek 2.18: Ukázky z dopňků a)True-Earth a b)True-Mars

Kapitola 3

Návrh

V této kapitole je popsaný výběr modelovaných objektů, návrh na tvorbu astronomických objektů a jejich detailů. Také je zde popsán návrh uživatelského rozhraní a v neposlední řádě návrh použití výsledného rozšíření pro aplikaci Blender.

3.1 Výběr vesmírných objektů

Výběr vesmírných objektů, které tímto doplňkem bude možné modelovat je omezený. I přes tento fakt se doplněk snaží umožnit uživateli modelovat co nejvíce druhů vesmírných objektů, které je ve vesmíru možno najít. Doplňkem nebude možné modelovat jakékoliv člověkem vytvořené vesmírné objekty, neboť v takovém případě by musel doplněk umožnovat vytvořit prakticky cokoliv. Mezi objekty, které bude možno modelovat patří:

- planeta
- hvězda
- měsíc (není myšleno konkrétní Měsíc na orbitě Země)
- galaxie
- černá díra
- asteroid a pole asteroidů
- plynový obr (podle definice je planeta, ale z důvodu implementačních rozdílností je zde bráno jako separátní typ objektu)

Bude možné modelovat i vesmírné pozadí, aby nemusela každá vzdálená hvězda být modelována jako separátní objekt.

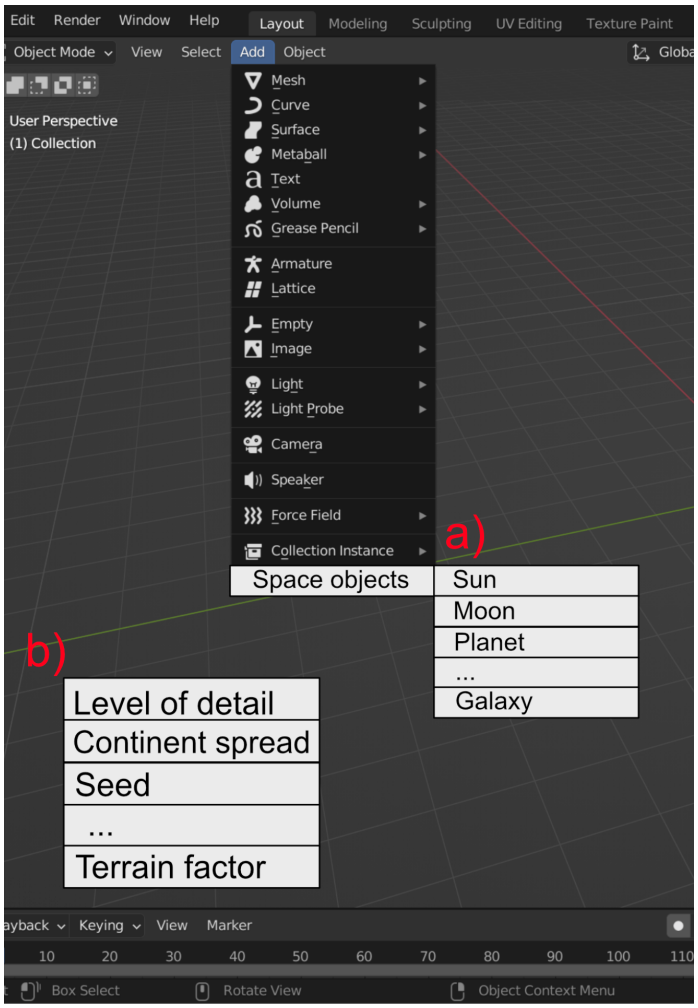
3.2 Uživatelské rozhraní

Uživatelské rozhraní je velmi podstatná část jakéhokoliv doplňku. Představuje totiž spojku mezi uživatelem a samotným doplňkem. Z tohoto důvodu by mělo být uživatelské rozhraní co nejvíce přehledné, logicky rozložené a promyšlené tak, aby uživatel nemusel složitě hledat prvky doplňku na místech, kde nejsou předpokládané a ničil si tak svůj workflow. Z tohoto

důvodu je uživatelské rozhraní doplňku rozděleno na tři části, kde každá část bude reprezentovat jeden krok z tvorby vesmírného objektu - přidání objektu, úprava tvaru a finální vzhled celého modelu.

3.2.1 Add Menu

Častá operace tohoto doplňku je přidávání různých objektů. Ať už prostých kulových těles, nebo celých přednastavených vesmírných objektů. Tyto možnosti jsou zakomponované jako prosté menu, které bude přidáno do *Add menu* (výchozí menu Blenderu pro přidávání objektů do scény). Toto přidané menu bude obsahovat obecné modely, které bude možné poté upravovat v dalších krocích. Kromě těchto možností bude obsahovat také dílčí menu pro přidávání přednastavených astronomických objektů, které už budou mít přednastavený vzhled (např. planeta, která bude připomínat zemi, plynový obr podobného vzhledu jako Jupiter, atd.), aby byla co nejvíce ulehčena práce uživatele. Celé menu bude zároveň sloužit jako seznam všech vesmírných modelů, které bude doplněk umožňovat vytvořit, jak je naznačeno na obrázku 3.1. Menu bude jednoduše rozšiřitelné, pro případ přidávání dalších funkcionalit doplňku.



Obrázek 3.1: Návrh uživatelského rozhraní doplňku. Menu pro přidávání objektů (a) a panel pro tvorbu objektů (b).

3.2.2 Panel pro tvorbu objektů

Při spuštění výchozích operátorů v Blenderu (např. `primitive_cube_add()`) je v levém dolním rohu vytvořen dočasný panel, který je dostupný dokud operátor neukončí svoji činnost. Tento panel je využitý také pro přidávání vesmírných objektů do scény. Slouží pro úpravu vstupních hodnot při tvorbě objektů. Uživatel pomocí tohoto panelu zadává vstupní informace, které jsou použity pro tvorbu samotného objektu. Kromě polohy, rotace a měřítka jsou tímto doplňkem také vyžadovány informace jako úroveň detailu objektu, velikost terénu, velikost kráterů nebo celistvost kontinentů na planetě.

3.2.3 Práce s vytvořeným objektem

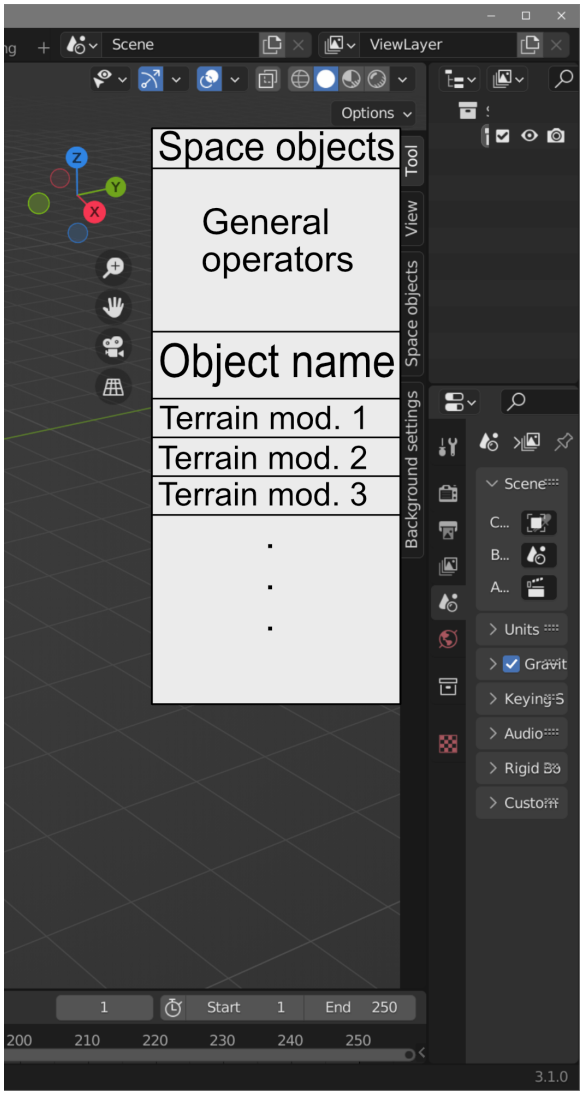
Obecnou zvyklostí, co se týče tvorby add-onů v Blenderu, je umisťovat funkcionalitu add-onu, která přímo nesouvisí s konkrétním pracovní plochou, do postranního panelu, který je naznačen na obrázku 3.2. Vytvoří se tak plocha, která se týká daného add-onu a uživatel tak nemusí složitě hledat v celém uživatelském rozhraní části daného add-onu. Stejný postup byl použit i zde a veškerá funkcionalita, kromě přidávání objektů a přímé práce s materiály, byla umístěna do postranního panelu, kde k ní bude mít uživatel snadný přístup.

Každý typ objektu bude mít rozdílný panel, z důvodu odlišných upravitelných vlastností, např. není možné, aby model slunce měl nad sebou mraky. Na postranním panelu bude úprava vzhledu, resp. barev a tvarů vytvořených objektů. Pokud se ovšem bude jednat o detailnější objekt (planeta, měsíc, asteroid) nebo o objekty, kde je tvar předvídatelný (slunce), tak bude tvorba tvaru objektu již nedostupná, jak je popsáno v sekci 3.2.2. Tento postup je zaveden z důvodu, že objekty jsou tvořeny pomocí operátorů. Díky tomu se tvar objektu upravuje poměrně jednoduše dokud je operátor aktivní, jakmile se však ukončí jeho proces, tak je vytvořený objekt přidán na scénu. Poté je již složité upravovat jeho tvar a zároveň si uchovávat tvar výchozí, aby bylo možné pro stejný *seed* tvořit pokaždé stejný objekt.

3.3 Tvorba objektů

Základem každého modelu, který je vytvořen doplňkem, je samotná geometrie složená z bodů a polygonů. Vzhledem ke gravitaci ve vesmíru se dá předpokládat, že většina vesmírných objektů bude tvořena kulovými objekty, což platí pro většinu modelů v tomto doplňku. Avšak pár typů objektů má daleko složitější tvar. Ty by v případě vyžadování detailního 3D modelu zabralo takový výpočetní výkon, že to v poměru ke zkvalitnění výsledného modelu není efektivní. Tvorba samotných objektů je proto rozdělena na dva základní postupy:

- Objekty, které již bez materiálu evokují svůj výsledný typ. Pro jejich tvorbu je často potřeba upravovat polohu jednotlivých vertexů. Mezi takové patří například planety, asteroidy nebo planetární prstence. Jejich povrch se nejčastěji upravuje opět pomocí šumu, aby šlo co nejlépe simulovat věrohodný povrch např. planety. Pro tento účel je používán výše zmíněný Perlinův šum, který má velmi organický vzhled v kombinaci s dostatečnou komplexností výsledného výstupu. Používá se např. u planety, kdy už u samotného objektu bez vykresleného povrchu je jasně vidět tvar planety.
- Objekty, které mají jako výchozí tvar nějaký typický geometrický tvar. Jejich výsledný tvar a vzhled bude pak upravován pomocí povrchu a materiálů z důvodu jejich výrazné



Obrázek 3.2: Návrh uživatelského rozhraní doplňku - panel pro práci s vytvořenými objekty.

složitosti. Např. galaxie, kterou jako objekt tvoří krychle a tvar galaxie je tvořen až v materiálu.

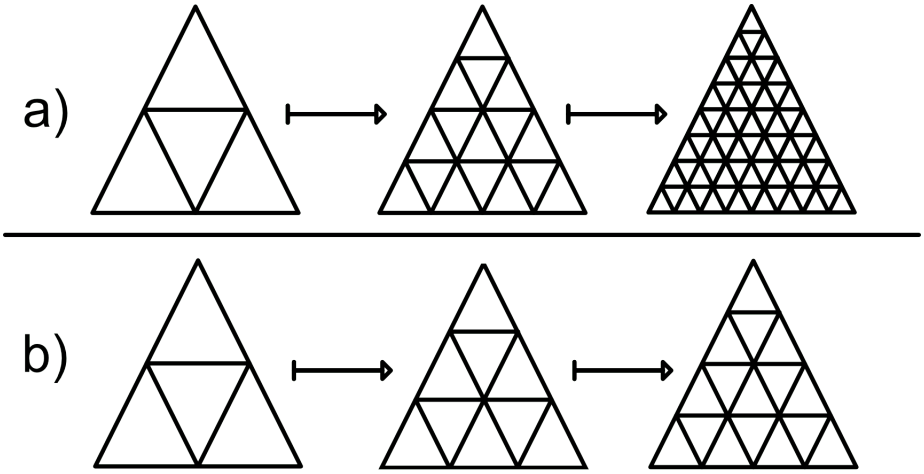
3.3.1 Modifikovaná tvorba koule

Pro některé objekty tvořené kterýmkoliv z výše zmíněných postupů, lze použít výchozí operátory definované přímo Blenderem. Pro některé z těchto objektů, které by mohly využívat kouli definovanou standardním způsobem, bylo nutné tvorbu koule předělat. Bylo tak nutné učinit z důvodu nevhodného dělení polygonů, jak bylo uvedeno v sekci 2.1.3.1. Vstupní vlastnosti má vylepšený operátor pro tvorbu koule prakticky totožné, tudíž mnoho uživatelů nepozná na první pohled žádný rozdíl. Ten se projeví až při zvyšování úrovně dělení, kdy pro výchozí dělení trojúhelníků poskytnuté Blenderem, se počet výsledných bodů (tím pádem vertexů, které budou tvořené na základě těchto trojúhelníků) daleko drastičtěji zvyšuje. Testování této metody je v sekci 5.1.

Důvodem tohoto drastického zvyšování bodů je rozdílný postup pro hledání nových bodů a následně polygonů v trojúhelníku:

- **Výchozí způsob Blenderu** - Výchozí způsob Blenderu má jednu nezanedbatelnou vlastnost - rekurzivitu. Díky tomu může být proces dělení trojúhelníků vypočítán z předchozí úrovně a ne úplně z výchozího stavu. Celý princip spočívá v tom, že u každé strany trojúhelníku je určen její střed a pomocí středních příček (základní vlastnost každého trojúhelníku) dojde k rozdělení na čtyři dílčí trojúhelníky a ty jsou pak používány místo trojúhelníku původního. Na každý z těchto nově vytvořených trojúhelníků lze aplikovat úplně stejný postup a lze takto dělit trojúhelníky kolikrát je zapotřebí, ale počet vrcholů se pokaždé velmi zvětší.

- **Modifikovaný způsob** - tento způsob má oproti výchozímu způsobu Blenderu jednu nevýhodu, a to že rozdělení jednotlivých trojúhelníků je pokaždé počítáno z výchozího trojúhelníku. Teoreticky lze odvodit nově rozdělený trojúhelník z trojúhelníku právě děleného, protože při každém dělení je v podstatě přidána pod základnu základna nová, která má o jeden bod víc než ta předchozí. Ale v praxi, kdy má každý bod vypočítanou polohu v prostoru je toto velmi obtížné. Proto je pokaždé nově dělený trojúhelník odvozen z výchozího trojúhelníku o třech bodech. Stejně jako u předchozího způsobu jsou děleny strany a vytvořené středy se poté propojí středními příčkami, ale rozdíl je v tom, že strany nejsou děleny pouze na poloviny. Při použití prvního stupně dělení tak jsou sice děleny strany na poloviny, ale při druhém stupni jsou děleny již na třetiny, při třetím na čtvrtiny atd. Tento způsob zobrazuje obrázek 3.3. Pravidlem tudíž je, že stupeň dělení určuje počet nově vzniklých bodů na každé straně.



Obrázek 3.3: Ukázka způsobů dělení trojúhelníku. a) Výchozí způsob Blenderu. b) Modifikovaný způsob doplňku.

Tento modifikovaný způsob je pak použit operátorem pro tvorbu tří druhů vesmírných objektů - planety, měsíce a asteroidů. Tyto objekty mají povrch upravený Perlinovým šumem (v případě měsíce taktéž krátery), tudíž se u nic nejvíce projeví rozdílná frekvence bodů rozprostřených na kulovou plochu.

3.3.2 Princip tvorby šumu/terénu

Tvorba terénu na každé planetě je založena na šumu. Díky jeho vlastnostem se jedná o ideální prostředek pro tvorbu povrchu planet. Základní popis tvorby povrchu pomocí šumu je popsán v sekci 2.4.2. Tento postup však stačí jen k tvorbě základního povrchu. Pro tvorbu sofistikovanějšího a realističtějšího terénu je potřeba šum využít více způsoby než jen pro prostou změnu výšky terénu.

3.3.2.1 Biomy

Stejně jako v reálném světě je povrch planet rozdělen na jednotlivé „biomy“. Celý povrch, který je přetvářen šumem je rozdělen na dílčí části. V reálném světě má každý biom jiný povrch v závislosti na počasí, průměrné roční teplotě nebo např. na živočiších, kteří v daném biomu žijí. Všechny tyto vlastnosti nejsou do doplňku přeneseny, vzhledem k tomu, že takové detailní vlastnosti nepodporuje. Samotnou diverzitu povrchu ovšem lze efektivně využít. Celý povrch je na výše zmíněné části rozdělen právě šumem, který je použit jako vstupní hodnota pro určení příslušnosti aktuálního bodu na povrchu k danému biomu.

Princip pro rozdělení bodů je jednoduchý. Pro každý bod je vygenerování hodnota specifickou funkcí pro tvorbu šumu. Vstupem je poloha onoho bodu v prostoru a výstupem hodnota mezi <0;1). Tato hodnota je vynásobena počtem chtěných biomů na povrchu planety. U výsledného čísla je poté smazána jeho desetinná část, díky čemuž vzniklo číslo, které udává index biomu, ke kterému bude daný bod náležet. Tento způsob ovšem způsobuje, že biomy, které spolu přímo nesousedí tak mají daleko menší pravděpodobnost, že budou mít hranice na samotném povrchu. Z tohoto důvodu je potřeba zvolit sousedící biomy logicky, např. mírné podnebí a vedle lehce horského terénu. Pokud by se vedle sebe zařadily například pouště a horské ledovce, tak by výsledek vypadal velmi nerealisticky. V závislosti na výše uvedeném by měly být veškeré biomy seřazeny za sebou podle logické souvislosti - dobrý způsob je například podle podnebných pásů, jak je zobrazeno na obrázku 3.4.

Obrázek 3.4: Jednoduché seřazení biomů používané doplňkem a velikosti jejich vlivů.

Díky skutečnosti, že příslušnost jednotlivých bodů je určena šumem, lze jednoduše měnit velikosti jednotlivých biomů, rozprostření po ploše nebo například přizpůsobit jejich polohu na povrchu planety tak, aby odpovídala daleko lépe požadavkům uživatele.

Každý biom má pak svou vlastní specifickou funkci pro tvorbu šumu, která co nejlépe simuluje povrch v biomu, který se zrovna tento modelovaný biom snaží napodobit. Například v hornatém biomu je povrch daleko více různorodý co se týče výšky, tudíž frekvence jeho šumu je daleko vyšší než u biomu, která modeluje planiny.

Avšak jednoznačné rozdělení bodů do biomů způsobuje jeden velký problém. Na hranicích jednotlivých biomů může naprosto jednoduše dojít ke stavu, že sousedící vertexy budou mít naprosto nepřirozeně odlišnou výšku a to z důvodu, že pro každý z nich byla použita jiná varianta šumu, a u výpočtu změny výšky vertexu není brán ohled na jeho sousedící vertexy, které nepatří ke stejnému biomu jako právě počítaný vertex. Tento problém lze

vyřešit předěláním způsobu pro rozdělení bodů k biomům, což popisuje alg. 1 (naznačeno na obrázku 3.5):

1. Na interval <0;1> jsou rozmístěny body, které představují jednotlivé biomy, Jejich počet odpovídá počtu chtěných biomů. Jejich vzájemná vzdálenost určuje množství efektu daného biomu na modelovaném povrchu.
2. Je vygenerována hodnota pro daný počítaný bod x.
3. Pro vygenerovanou hodnotu je vypočítána vzdálenost od každého bodu. Tato vzdálenost představuje faktor příslušnosti bodu k danému biomu. Volitelně je každý faktor odmocněn pro více rozpoznatelné rozdíly mezi biomy.
4. Pro daný bod jsou vypočítané všechny šumy pro jednotlivé biomy.
5. Výsledky těchto šumů jsou pronásobeny s jejich korespondujícími faktory příslušnosti.
6. Jednotlivé výsledky jsou sečteny a součet představuje změnu povrchu na daném bodu.

input : biomeNoises, currentPoint
output: terrainChange
$terrainChange = 0$
for $i \leftarrow 0$ **to** $biomeNoises_{size}$ **by** 1 **do**
  $biomes[i] = i/biomeNoises_{size}$
  $biomeWeight = distance(currentPoint, biomePosition)$
  $terraintChange+ = biomeWeight \cdot biomeNoises[i]$
end

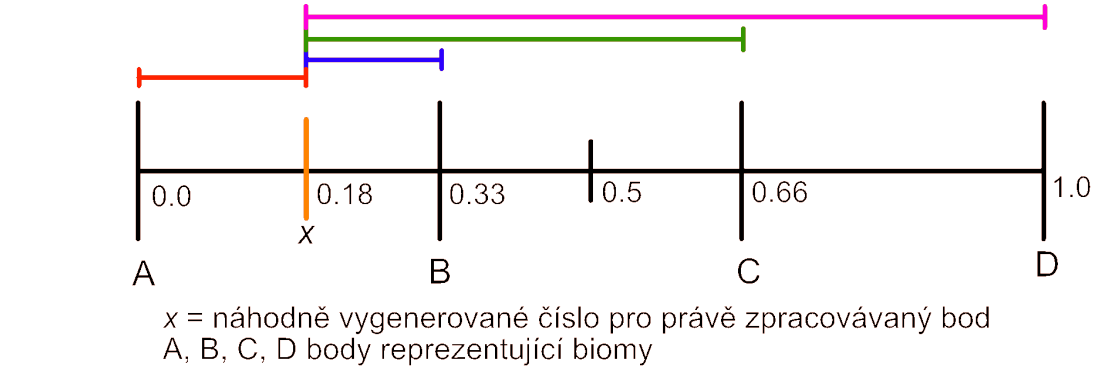
Algorithm 1: Algoritmus výpočtu modifikátorů šumových funkcí

Tento postup zaručuje, že čím více bude daný bod v „centru“ biomu, tím více bude ovlivněn čistě šumem pro daný biom. Naopak, pokud bude bod na pomezí dvou biomů, tak jeho výsledný šum se bude skládat ze dvou dílčích šumů. V důsledku toho bude přechod mezi jednotlivými biomy plynulý, i když bude zachována unikátnost biomů a biomy, které jsou vlastnostmi velmi vzdálené, tak na sebe vzájemné budou mít zanedbatelný vliv.

3.3.3 Princip tvorby kráterů

Jak již bylo zmíněno v sekci 2.5.1, tvar každého kráteru je složen ze tří hlavních částí: stěny kráteru, dna a vyvýšeného okraje nad jeho stěnami. Samotný kráter je potřeba modelovat mnohokrát, protože většina vesmírných těles se na své cestě vesmírem nezvládne vyhnout veškerým objektům se kterými se může srazit. Proto je potřeba krátery vytvářet způsobem, který zajistí, že kvalita výsledných kráterů bude minimálně srovnatelná s kvalitou objektu, na kterém je kráter vytvořen.

Jako asi nejvíce intuitivní metoda se jeví rozdíl objektů. Tato vestavěná funkce Blenderu by byla úspěšně použitelná pokud by se jako odečítaný objekt brala koule, případně koule, která bude mít upravený tvar pro simulaci dna kráteru. Tento postup se může zdát vhodný pro simulování malého počtu kráterů, ale ve větším množství už je velmi časově náročný a může zabrat drtivou většinu zdrojů stroje, na kterém je daná scéna modelována, neboť na daném objektu můžou být desítky nebo stovky kráterů, a pro každý z nich je potřeba vytvořit separátní objekt k vytvoření rozdílu.



Obrázek 3.5: Znázornění přislušnosti vygenerovaného bodu k daným biomům. Získání informace jak moc bude výsledná výška daného bodu ovlivněna jednotlivými šumy všech biomů. Bod x na obrázku leží v blízkosti biomů A a B, kterými bude ovlivněn silně, lehce bude ovlivněn biomem C a jen slabě biomem D.

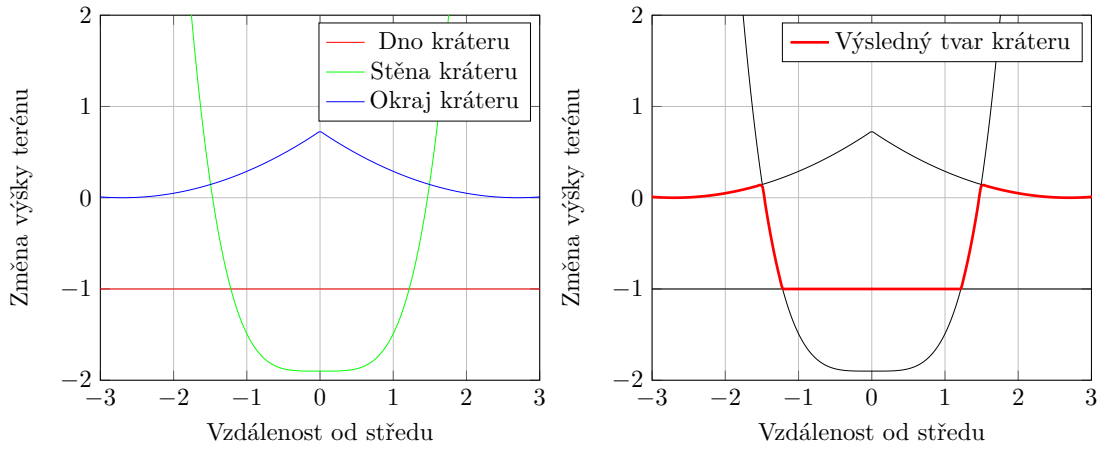
Způsob, který využívá tento doplněk, je odlišný od rozdílu objektů. Je založený na principu úpravy polohy vertexů, konkrétně jeho vzdálenosti od středu modelovaného objektu. Postup je podobný jako u aplikování šumu na povrch objektu, rozdíl je však v tom, že k výpočtu změny vzdálenosti od středu je místo šumu použitá skupina funkcí, které dohromady simulují tvar kráteru:

- **Stěna kráteru** - Na první pohled se může zdát že stěny kráteru mají přímý lineární tvar, ale vzhledem k tomu, že kráter byl vytvořen dopadem jiného objektu, je nárazová vlna kulovitého tvaru. S ohledem na tuto skutečnost lze stěny kráteru věrohodně simulovat parabolou ve tvaru: $f(x) = b + x^{2a}$, kde lze konstantami a a b ovlivňovat sklon a rozevření paraboly. Na obrázku 3.6 je znázorněna vlevo zeleně.

- **Dno kráteru** - Dno kráteru vzniká zanášením kráteru prachem a jiným materiálem z povrchu vesmírného objektu na kterém kráter je. Z toho důvodu není celý tvar kráteru parabola, ale v jisté výšce je utnutá většinou za pomocí přímky, případně šumu nebo je možnost na tento nově vzniklý povrch aplikovat opět menší krátery nebo jakékoliv úpravy terénu. Za předpokladu, že se dno paraboly kráteru utne přímkou, pak platí, že: $g(x) = c$, kde c je výška dna terénu. Na obrázku 3.6 je znázorněna vlevo žlutě.

- **Hrana kráteru** - Hrana kráteru vzniká hromaděním odlétajících částí planety a asteroidu při srážce. Díky faktu, že daleko více těchto částí doletí jen lehce za okraj kráteru a výrazně méně jich poletí dál, tak výsledný tvar hrany kráteru má opět parabolický tvar. V tomto případě ovšem není vrchol paraboly ve středu kráteru, ale v místě za jeho hranicí, kde už tento kráter prakticky splývá s okolním povrchem. Tvar hrany kráteru by mohl být přibližně vyjádřen jako: $h(x) = d \cdot (|x| - k)^2$, kde konstanta d určuje strmost hrany a konstanta k určuje horizontální vzdálenost vrcholu paraboly od středu kráteru. Na obrázku 3.6 je znázorněna vlevo modře.

Výsledný tvar kráterů je pak vytvořený minimem z funkcí pro hranu a stěnu kráteru. Toto minimum je porovnáno s dnem kráteru, a z těchto dvou hodnot je uchována pouze ta maximální, která je brána jako výsledný tvar. Těmito kritérii je vybrána v každém bodě ta funkce, která zrovna zobrazuje tvar tvořeného kráteru, což je vidět na obrázku 3.6. Osa x pak znázorňuje vzdálenost bodu od středu a osa y pak změnu povrchu objektu.



Obrázek 3.6: Znázornění funkcí pro tvorbu kráteru. Vlevo - hrana kráteru (modře), stěna kráteru (zeleně) a dno kráteru (červeně). Vpravo - tvar kráteru po vybrání minima maxima z předchozích funkcí.

3.3.4 Rozložení a velikost kráterů

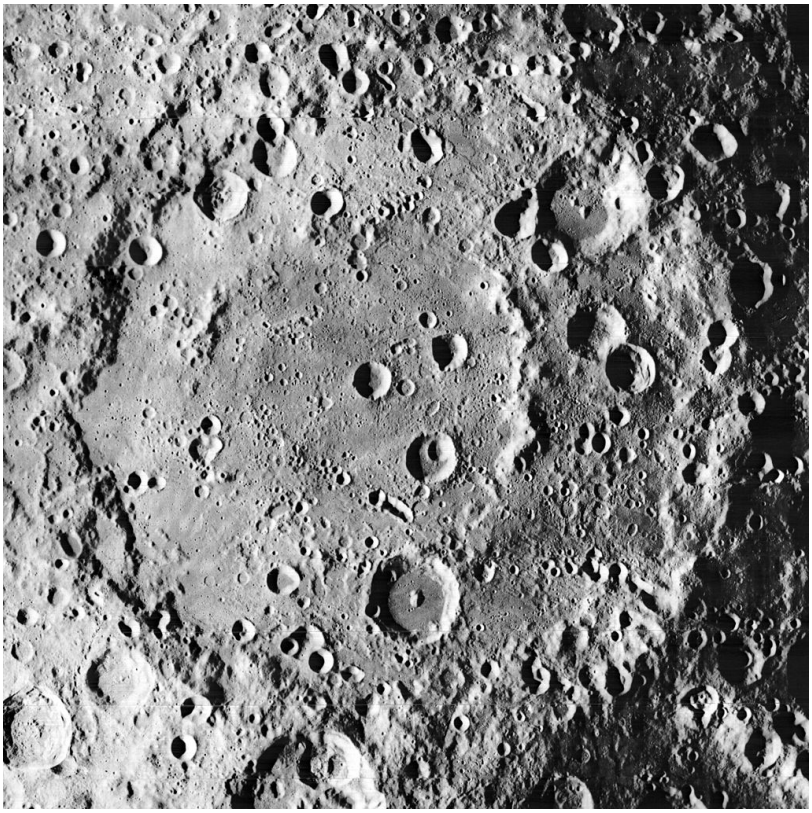
Jak bylo řečeno v předchozí části, tak poloha každého kráteru je určena jeho středem, neboť z vertikálního pohledu má každý kráter kruhový tvar. Princip tvoření kráterů se skládá z vybrání bodu, kde kráter bude vytvořený, výpočtu změn poloh vertexů planety a samotné transformace povrchu objektu. Avšak kvůli potenciálně vysokému počtu kráterů (které můžou čítat stovky nebo tisíce) a samotných vertexů planety (kterých může být desetitisíce, statisíce nebo i miliony) je naprosto zbytečné, aby se přepočítávala poloha každého vertexu, ze kterého je planeta složena.

Poloměr kráteru může být jen o něco menší než poloměr planety, ale daleko častěji bude jen zlomkem poloměru planety. Z této skutečnosti může být usouzeno, že povrch jednotlivého kráteru bude ve většině případů zabírat jen zlomek povrchu planety a tudíž kráter nebude mít vliv na většinu plochy povrchu planety. Vzhledem k tomuto faktu, je naprosto zbytečné počítat výše zmíněnou funkci v sekci 3.3.3 pro veškeré vertexy planety. Dalším důvodem je také skutečnosti, že pří výpočtu změnu povrchu planety dojde ke stavu, kdy aktuálně počítaný bod x je od středu kráteru vzdálen více, než vrchol paraboly, která tvoří hranu kráteru. Funkce pro tuto hranu, a tudíž pro celý kráter, pak navrací nežádoucí hodnoty. Proto je velmi vhodné přetvářet povrch planety jen v určitém okruhu okolo středu kráteru. Aby bylo zároveň zabráněno nežádoucím návratovým hodnotám funkce pro hranu kráteru, tak poloměr tohoto okruhu je totožný s bodem kde se protíná parabola tvořící hranu kráteru a původní povrch objektu.

Dalším problémem při tvorbě kráterů je jejich rozmísťování na samotnou planetu a určování jejich velikosti. Co se týče určování polohy kráterů, tak nejjednodušším a zároveň nejreálnějším způsobem je prosté náhodné určení bodu na povrchu planety. Ve vesmíru létají vesmírné objekty relativně nahodile, pokud se nebere v potaz gravitace nebo oběžné

dráhy vesmírných těles. Tyto faktory by mohly mít za důsledek, že krátery se budou objevovat daleko více na jedné straně planety než na druhé. V této verzi doplňku však gravitace ani oběžné dráhy nejsou implementované, tudíž je nejlepší varianta uniformní rozprostření na celý povrch.

Odlišný přístup je ovšem u určování velikosti kráterů (velikostí se rozumí poloměr a hloubka). Jedna z možností je, že budou všechny krátery stejně velké v závislosti na vstupu uživatele, který tuto velikost zadá při tvorbě planety s krátery. Ovšem není realistické, aby všechny krátery měly stejný poloměr. Jako další možnost lze použít náhodné rozdělení, jako u určování polohy. Nelze předpokládat, že ve vesmíru jsou veškeré objekty rozdělené podle velikosti rovnoměrně. Z jakýchkoliv zdrojů a fotografií (příklad na obrázku 3.7) lze zjistit, že menších objektů je ve vesmíru daleko více než těch velkých. Např. v jakémkoli pásu asteroidů se nachází daleko víc malých asteroidů a „prachu“ než těch velkých. Z těchto faktů lze poměrně jednoduše usoudit, že pro určování velikosti kráterů bude nejlepší možnou volbou geometrické rozdělení hodnot. Toto rozdělení zajišťuje, že čím větší bude velikost kráteru, tím méně bude kráterů o takové velikosti vygenerováno. Výsledkem je, že bude vygenerována planeta (měsíc) např. se dvěma velkými krátery, které zůstaly po velkých srážkách a množstvím malých kráterů, které se tvoří stále, ať už po malých asteroidech nebo třeba po vesmírném odpadu.



Obrázek 3.7: Zobrazení rozdělení velikosti kráterů na povrchu Měsice.[1]

[1]Převzato z https://www.scienceabc.com/nature/universe/why-are-impact-craters-on-the-moon-round-and-not-some-other-shape.html.

3.3.5 Překrývání kráterů

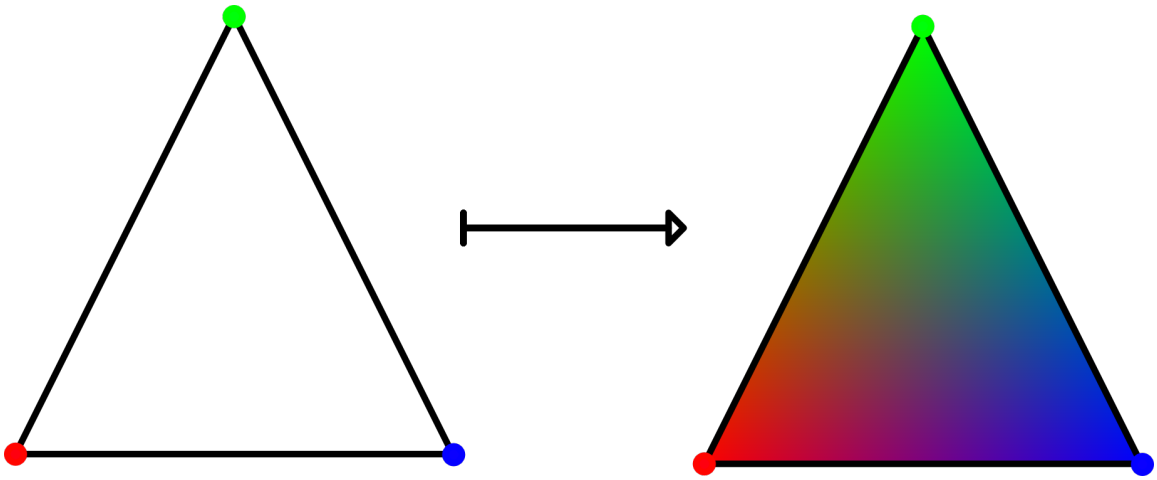
Při tvorbě a rozmísťování kráterů vzniká nevyhnutelně problém s překrýváním kráterů. Spočívá v tom, že může nastat situace, kdy je na objekt vygenerováno více než jeden samostatný kráter. S počtem kráterů však roste šance, že pro některé body, které budou mít nejméně dva krátery společné, bude vypočítáno několik různých změn vzdálenosti od středu. V takovém případě není jasné, kterou změnou se řídit a jak má tím pádem změněn povrch v daném bodě. Určit jak moc má který kráter vliv na výslednou změnu povrchu lze v zásadě třemi způsoby:

- **Vliv posledního** - Tento princip je velmi jednoduchý. Za předpokladu, že je výpočet vlivu každého kráteru počítán postupně, lze změnu vlivu vždy přepsat novým kráterem. Tento způsob je rychlý, avšak díky tomu, že každý kráter je počítán z původního povrchu, vyvstává problém. Vliv kráteru, který je na daném bodě počítán poslední, absolutně nezohledňuje své okolí, tudíž přepíše všechny předchozí změny povrchu daného bodu. Tento fakt má za důsledek velice nerealistické změny povrchu, zejména na hranicích posledního kráteru. Pokud například budou dva krátery do sebe zasahovat tak moc, že nový kráter bude končit zhruba ve středu staršího, tak vzniká znatelný problém. Nový kráter vyzdvihne povrch nad původní hodnotu, ale sousedící vertex bude daleko pod původním povrchem. V důsledku velkého rozdílu výšky sousedících vertexů tak vzniká velmi nerealistický povrch.
- **Spojení vlivů** - V tomto přístupu jsou na rozdíl od ignorování veškerých kráterů brány v potaz vlivy všech kráterů. Stejně jako v předchozím případě je postupně vypočítán vliv každého kráteru, který na daný bod zasahuje. Místo postupného přepisování výsledného vlivu jsou však naopak tyto vlivy sčítány. Tímto procesem je sice zajištěno, že vliv každého kráteru na změnu povrchu planety je zajištěn, ale sečtené vlivy můžou dosahovat takových hodnot, že povrch bude ve výsledku opět velmi nerealistický. Například pokud bude vytvořeno několik kráterů ve velmi blízkém okolí, tak jejich body, které budou patřit do jejich dna a zároveň budou průnikem více těchto kráterů, budou mít až nesmyslně velkou hloubku, tudíž celé výsledné dno bude mít velmi prudce stupňovitý tvar.
- **Kombinace** - Pro realistický dojem je potřeba spojit výhody obou předchozích metod - udržení hodnot v relativně velkých hranicích a zároveň jednoznačně ukázat vliv, tvar a velikost všech kráterů na povrchu objektu. Z druhé potřebné vlastnosti lze usoudit, že do výsledného povrchu budou zasahovat všechny krátery jako v druhé možnosti. Tyto vlivy ale nemůžou být ve své původní velikosti, neboť také v reálném vesmíru krátery postupem času mizí. Taktéž je velice nepravděpodobné, že by asteroidy dopadly v jeden moment. I v případě, že by dopadly, tak vytvoří jen jeden společný kráter. V důsledku těchto faktů je nový princip následovný. První kráter, který ovlivní daný bod bude brán jako jeho hlavní kráter. Taktéž bude jeho vliv na povrch vypočítán naprosto normálně a použitý ve své původní velikosti. Pro kterýkoliv další kráter počítaný pro tento bod bude jeho změna připočítána jen ve zlomkové velikosti, aby na výsledném povrchu byly znatelné jeho stopy na povrchu, ale zároveň naprosto neurčil její celkový vzhled.

3.4 Vzhled povrchu objektů

Díky tomu, že čistě pomocí modelování povrchu objektů lze zdánlivě simulovat i tvar, je tvorba povrchu vesmírných objektů tak možná ještě důležitější, než samotná tvorba jeho tvaru. Pro tvorbu povrchů v tomto doplňku byly zvolené tzv. *materiály*. Pokud by nebyl použit tento postup, tak by se povrch modelů mohl řešit pomocí dvou postupů:

- **Barva vertexů** - asi nejjednodušší způsob, jak zajistit barvu modelu je přiřadit každému vrcholu určitou barvu. Barva celého polygonu je pak určována pro každý bod v něm jako interpolace mezi jeho vrcholy. Z toho vyplývá, že čím blíž je daný bod v rámci polygonu blíže vrcholu který má např. červenou barvu, tím silnější bude červená složka v jeho vlastní barvě, jak je zobrazeno na 3.8
- **Přímé texturování** - daleko známější postup např. ze starších videoher. Je vytvořena textura (prostý obrázek), který je pak pomocí relativních souřadnic převeden na požadovaný tvar. Tento postup je prakticky totožný s tvorbou UV-sphere, který byl zmiňován již v sekci 2.1.1. Postup je zobrazen na obrázku 3.9.

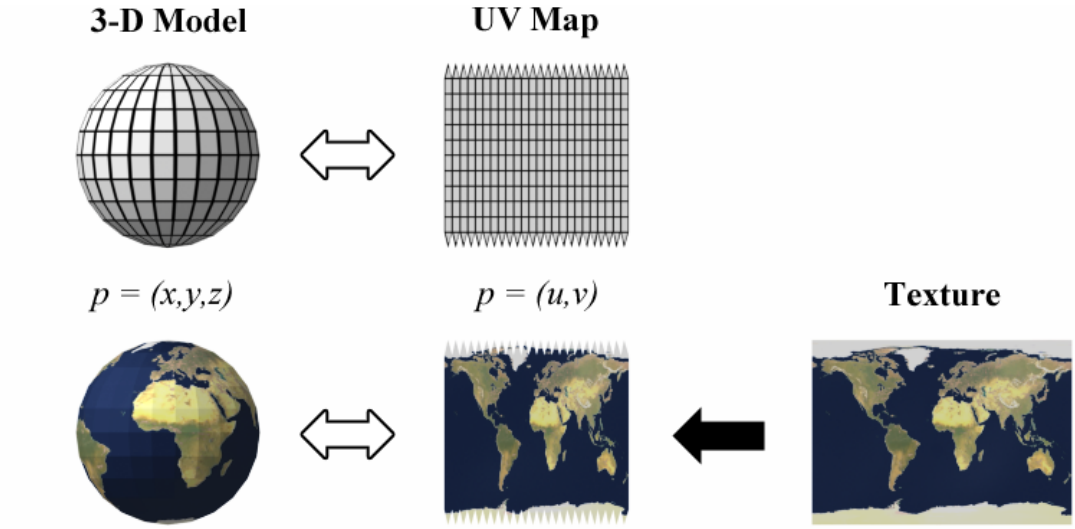


Obrázek 3.8: Zobrazení postupu interpolace barev vertexů.

3.4.1 Planetární materiály

Pro účely tohoto doplňku byly navrhnuty čtyři typy materiálů, které budou korespondovat s vlastnostmi/úrovní propracovanosti daného typu objektu:

- **Povrch se závislostí na výšce terénu** - Tento typ materiálu je určen zejména pro hlavní typ objektu celého doplňku - planety. Základním vstupem každého bodu je jeho vzdálenost od středu objektu, díky čemuž lze kvalitně simulovat např. vodní hladinu, kopce, hory nebo horské ledovce. Vzdálenost každého bodu symbolizuje jeho příslušnost k danému typu povrchu. Pro vodní hladinu bude určena uživatelem hodnota a veškeré body, které budou pod ní, tak budou automaticky spadat pod tento typ povrchu. Je tak nemožné pro stávající verzi doplňku simulovat pevninu, která by byla pod „nadmořskou výškou“.



Obrázek 3.9: Zobrazení postupu texturování polygonů pomocí koordinátů u, v.[2]

- **Povrch závislý čistě na náhodném šumu** - Tento typ materiálu je využíván pro objekty, které mají tvar podobný základním geometrickým tvarům (hvězda) nebo narušení tohoto tvaru nema vliv na jeho barvu a vlastnosti povrchu(měsíc, asteroidy). Zpravidla jsou založeny na šumu, který je v několika iteracích smíchán dohromady a výsledek je podán jako faktor pro barevnou škálu, kterou již definuje uživatel.
- **Objekty které jsou tvořené samotným materiálem** - Tyto objekty jsou často geometrické objekty jako kvádr a koule. Veškerá složitost vesmírného objektu je tvořena materiály, které simulují tvar, průhlednost a barvu. Tento postup je použitý u vesmírných objektů, které by byly nesmírně složité tvořit pomocí detailních objektů jako takových. Jedná se např. o černou díru nebo galaxii.
- **Specifické materiály** - materiály požité k unikátním účelům jako např. mraky, které závisí čistě na náhodném šumu, nebo planetární prstenec, kde je barva a průhlednost bodu závislá na vzdálenosti od středu.

[2]Převzato z https://wblog.wiki/cs/UV_sphere#wiki-1.

Kapitola 4

Implementace

V této části práce je popsána samotná implementace projektu. Je zde popsána struktura doplňku, jeho využití API Blenderu, způsoby pro vykreslování samotných objektů a funkce, které nespadají přímo pod tvorbu objektů, ale jsou pro funkčnost doplňku nezbytné. Dále jsou zde uvedené zajímavě implementované funkce, které byly potřeba pro překonání složitějších problémů v rámci doplňku. Jako poslední jsou rozebrané způsoby testování a následné publikování doplňku.

4.1 Struktura

Celková struktura doplňku musí odpovídat předem definované struktuře vícesouborového doplňku. Pokud by se jednalo o doplněk jednosouborový, stačil by prostý python script s definovanými funkcemi `register()` a `unregister()` pro registraci jednotlivých tříd a dalších potřebných věcí. Vícesouborový doplněk ovšem potřebuje inicializační soubor `__init__.py`, který prochází všechny moduly, které má v sobě uvedené a pro tyto moduly volá opět funkce `register()` a `unregister()`, aby bylo zajištěno, že všechny potřebné třídy, funkce, atd. byly inicializovány a dostupné pro použití, ať už samotnému doplňku, nebo přímo Blenderem z jeho zabudované Python konzole.

Samotné funkce `register()` a `unregister()` obsahují funkce pro registraci jednotlivých tříd/vlastností, které všechny musí být podle konvencí Blenderu. Pokud se jedná o registraci třídy, je ji potřeba registrovat pomocí `bpy.utils.register_class()`. Pokud se jedná o nějakou vlastnost (*Property*), je potřeba využít objekt `bpy.types.Scene.custom_name`, kde `Scene` značí scénu, v rámci které bude daná vlastnost registrována.

Soubor `__init__.py` musí obsahovat také obecné informace, které jsou potřeba k registraci doplňku. Tyto informace se uvádějí ve struktuře `bl_info`, ve které jsou obsažené informace jako název doplňku, jeho kategorie, autor, verze a minimální potřebná verze Blenderu pro jeho úspěšnou registraci a používání.

Celý doplněk je strukturován jako hlavní adresář, ve které je uložen inicializační skript `__init__.py` společně se všemi ostatními podadresáři, ve kterých jsou uložené soubory s třídami a funkcemi, které skládají celý doplněk. Tyto podadresáře jsou:

- **craters** - Obsahuje soubor pro tvorbu kráterů, funkce na výpočet jejich tvaru a velikosti.
- **general** - Obsahuje soubory pro tvorbu objektu, který pro účely tohoto doplňku nahrazuje defaultní Icosphere. Obsahuje třídu jako takovou a operátor, který pro při-

dávání této modifikované koule slouží. Stejně tak je zde funkce pro výpočet šumu pro úpravu povrchu a rozdělení bodů mezi biomy. Dále také obsahuje soubory, které umožnují práci nad vytvořenou planetou. Obsahuje také operátory pro přidávání externích objektů - planetárního prstence a mraků, operátor pro přidání barevného povrchu na planetu a operátor pro vyhlazení hranic mezi vodou a pevninou na planetě.

- **operators** - Tento adresář obsahuje operátory, které netvoří přímo objekty jako takové, ale stále jsou nezbytné pro tvorbu vesmírné scény. Jsou to operátory na tvorbu vesmírného pozadí a operátor, který zajišťuje objektům, které vyzařují světlo, záři u vykreslených snímků, jelikož render engine *Cycles* používá k tomuto účelu složitější způsob než rychlejší *Eevee*. Adresář obsahuje také podpůrné operátory.

- **other_obj** - Momentálně nejobsáhlejší adresář. Obsahuje soubory, které tvoří jiné vesmírné objekty než je obecná planeta. Těmito objekty jsou - *slunce, měsíc, galaxie, plynový obr, asteroid a asteroidové pole, černá díra a mlhovina*. Soubory obsahují operátory pro jejich tvorbu, výpočty a samotnou tvorbu materiálů pro vytvoření celého objektu na jednom místě.

- **settings** - Adresář obsahuje pomocné třídy a funkce, které jsou využívány na různých místech ve zbytku doplňku a ulehčují tak práci. Např. v souboru `my_node_tree.py` je třída `MyNodeTree`, která simuluje datovou strukturu stromu pro ukládání uzlů v materiálu a ulehčuje tak práci s těmito uzly.

- **sphere** - V tomto adresáři jsou uložené funkce, které umožňují tvorbu dříve zmiňovaného modifikovaného kulového objektu. Soubor `triangle_division.py` obsahuje funkce pro nový způsob dělení trojúhelníků, tvorby polygonů z těchto dílčích trojůhelníků, následnou eliminaci duplicitních vertexů a tvorby pole vertexů, které slouží jako vstupní data pro tvorbu objektu. Soubor `plain_object.py` poté zajišťuje tvorbu základního icosaedru a jeho modifikaci funkcemi pro dělbu trojúhelníků, promítání na kulovou plochu a aplikování šumu pro tvorbu povrchu.

- **ui** - Zde jsou uložené soubory pro tvorbu uživatelského rozhraní, zejména pravého bočního panelu, který zajišťuje individuální úpravy každého typu již vytvořeného objektu. Je zde separátní soubor pro každý unikátní objekt, který lze vytvořit tímto doplňkem.

Takto strukturovaný adresář je nutné komprimovat do formátu ZIP. Tento formát zaručuje, že při instalaci doplňku budou všechny souboru na jednom místě a bude je možno předat Blenderu jako jeden celek.

4.2 Využití API Blenderu

Pro správnou a srozumitelnou funkčnost celého doplňku bylo potřeba využít několik Blenderem definovaných konceptů. Tyto jednotlivé koncepty - zejména operátor, panel a menu jsou popsány z hlediska funkčnosti a použití, neboť velká část doplňku spočívá právě v těchto dvou konceptech.

- **Operátor** - Jedná se o třídu, která je doplňkem využívaná na tvorbu a přidávání objektů do scény, případně na jednorázové nastavení v Blenderu (zapnutí a vypnutí Bloom efektu)

- **Panel** - Tato třída je využívaná čistě jako základ uživatelského rozhraní. Slouží jako plocha do které se přidávají upravitelné vlastnosti, které jsou dostupné uživateli přímo ze scény.
- **Menu** - Třída, která se využívá pro přidávání operátorů do hlavního menu Blenderu. Zároveň slouží jako seznam všech operátorů, které lze přidat a mají podobný účel.

4.3 Uživatelské rozhraní

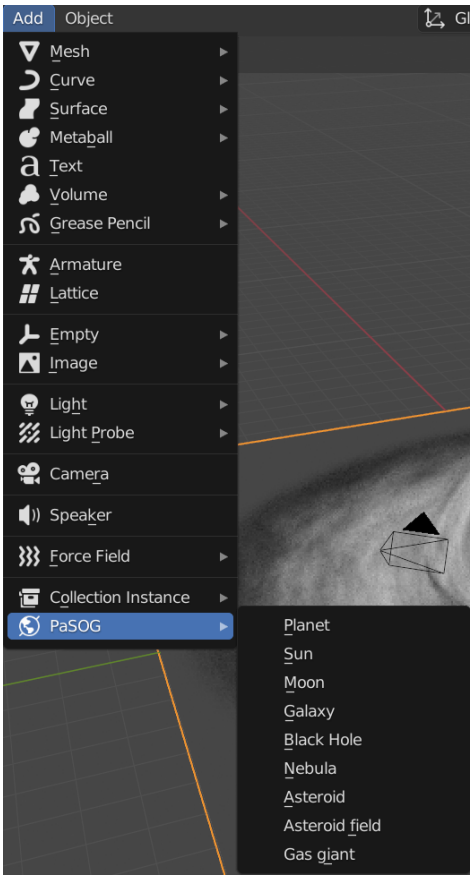
Celé rozhraní tohoto doplňku bylo vytvořeno tak, aby nijak nenarušovalo celistvost Blenderu a nebránilo v práci, pokud zrovna není tento doplněk využíván. Jak již bylo zmíněno v sekci 3.2, tak celé uživatelské rozhraní bylo rozdělené na tři samostatné části, ke kterým však byla přidána část čtvrtá z důvodu oddělení nastavení pozadí celé scény.

- **Add menu** - Část uživatelského rozhraní, která je velmi jednoduchá a slouží k prostému výběru objektu, který bude chtít uživatel modelovat. Slouží zároveň jako seznam dostupných objektů, jak je zobrazeno na obrázku 4.1. Tvořeno třídou `bpy.types.Menu`, což je v jádru kombinace `bpy.types.Panel` a `EnumProperty`.
- **Panel operátoru** - Tento panel slouží pro zadání vstupních proměnných operátoru, který daný objekt vytváří. U objektů, které mají povrch tvořený materiálem zobrazeným na povrchu je tento panel poměrně prázdný, jsou zde jen tři základní vektory (pozice, rotace a měřítko), případně ojedinělá doplňující proměnná, např. poloměr. Pokud ovšem operátor tvoří složitější objekt, např. planetu, tak vstupních proměnných je daleko více. Například pro planetu se jedná o úroveň detailu objektu, poloměr, „seed“ pro náhodnost, celistvost kontinentů, procentuální plocha zaplavená vodou (nebo obecně rovným povrchem) a výška terénu planety (panel zobrazen na obrázku 4.2). Pro měsíc se místo vody na povrchu objektu upravuje velikost a rozložení kráterů.
- **Panel pro úpravu povrchu** - Pomocí tohoto panelu uživatel pracuje již s vytvořeným objektem. Každý panel je instancí třídy `bpy.types.Panel`, ve které jsou vykreslené jednotlivé vlastnosti pro úpravu vzhledu. Vzhled panelu se liší na základě právě zvoleného objektu. Každý objekt vytvořený pomocí tohoto doplňku má v sobě zapsaný svůj typ, na jehož základě panel zobrazuje informace a vlastnosti relevantní pro daný objekt. Povrch každého objektu je rovněž tvořen jiným materiálem s jinými uzly (příklady těchto materiálů jsou v sekci 4.5.2) a tudíž nelze upravovat každý objekt stejně. Panel je zobrazen na obrázku 4.3. Příklady upravitelných vlastností zobrazených na panelu dle typu objektu:
  - **Planeta** - V případě planety je její povrch rozdělený na čtyři základní „podnebí“, z důvodu rozmanitosti povrchu. Každou část lze separátně upravovat tak, aby bylo možno docílit požadovaného modelu.
  - **Slunce, Měsíc, Asteroidy, Plynový obr** - V případě těchto objektů dominuje panelu úprava barev povrchu a také nastavení šumu, který se snaží v rámci povrchu „rozbít“ příliš rovné a plynulé tvary.
  - **Galaxie, Mlhovina** - Díky skutečnosti, že tyto dva objekty patří mezi dříve zmiňované „složité objekty“, je v rámci tohoto panelu upravován i jejich celkový

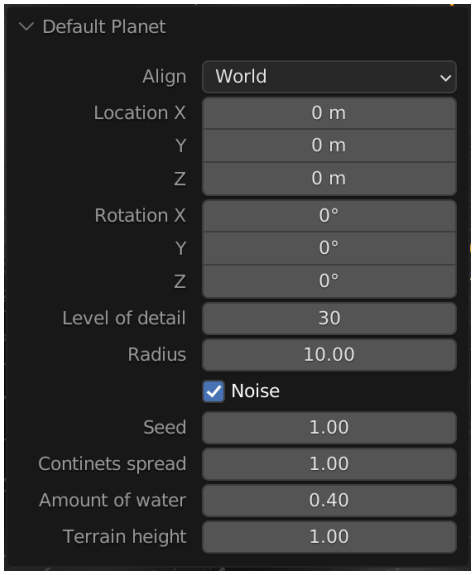
tvar, jejich rotaci, velikost nebo rozmístění. Stejně jako u předchozích objektů lze upravovat i barva.

– **Černá díra** - Panel pro černou díru je poněkud specifický. Samotný objekt se skládá ze dvou částí a tudíž se v rámci panelu musí určit, kterou část chce uživatel v daný moment upravovat. Lze upravovat velikost, rotaci, rozložení nebo samotnou barvu a její texturování. Díky specifičnosti těchto dvou objektů jsou zde i dva operátory na označení obou částí černé díry a taktéž na navrácení lokace disku na lokaci samotné černé díry.

Na každé instanci tohoto panelu se navíc nachází operátor pro přidání záře na výsledný render, pro objekty, které vyzařují světlo. Tento operátor je rozebrán v sekci 4.6.

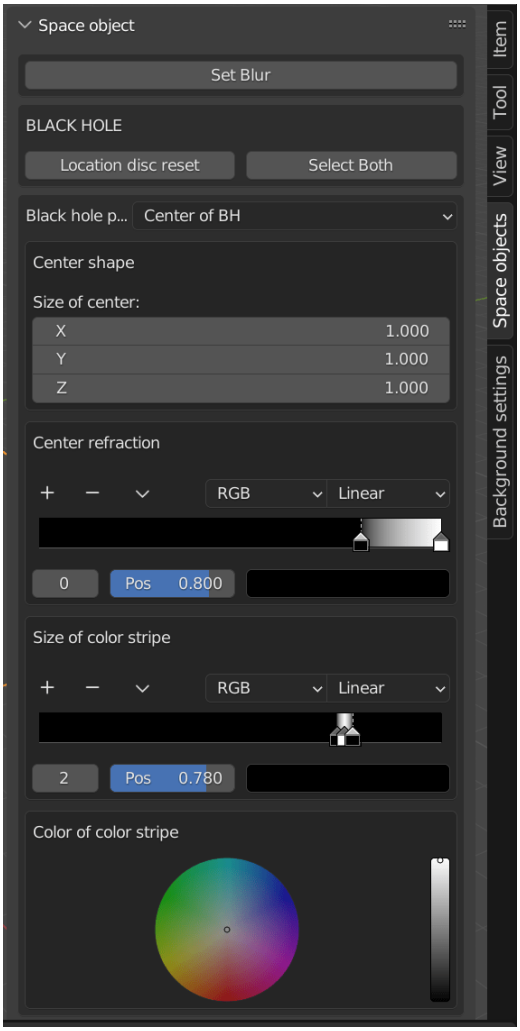


Obrázek 4.1: Část uživatelského rozhraní pro výběr přidávaného objektu v *Add menu*.



Obrázek 4.2: Panel zobrazující vstupní proměnné pro vytvoření daného objektu (v tomto případě planeta).

- **Panel pro úpravu pozadí** - Z důvodu rozdílného účelu od předchozích panelů, byl vytvořen druhý postranní panel pro úpravu pozadí. Panel je opět instancí třídy `bpy.types.Panel` a je složen ze dvou částí. V první části je operátor, který vytvoří materiál pomocí `ShaderNodes` simulující vesmírné pozadí. V druhé části jsou pak zobrazené jednotlivé vlastnosti pozadí, které se dají upravovat, jak je zobrazeno na obrázku 4.4. Mezi tyto vlastnosti patří velikost a barva různých kategorií hvězd a také struktura samotné barvy vesmíru, která není ani v reálném vesmíru čistě černá z důvodu rozsáhlých mlhovin a „oblaků“.

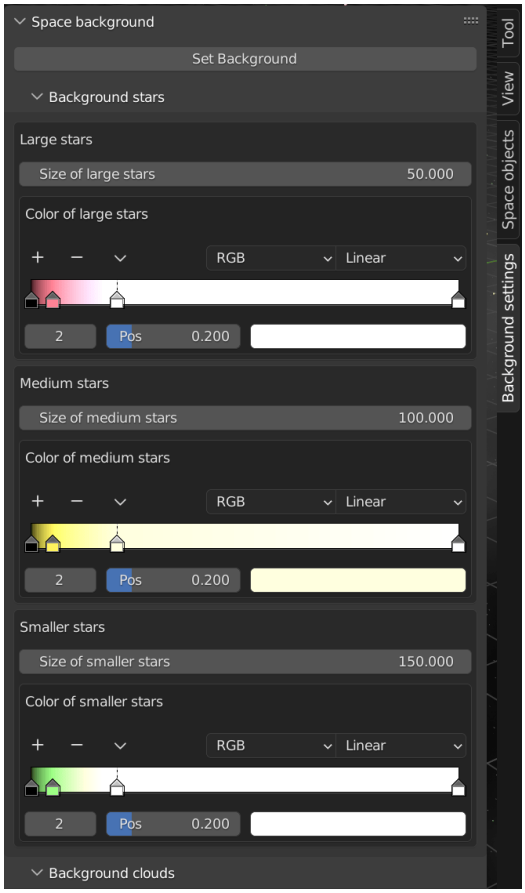


Obrázek 4.3: Panel zobrazující možnosti úpravy povrchu vytvořeného objektu (zde konrétně černá díra).

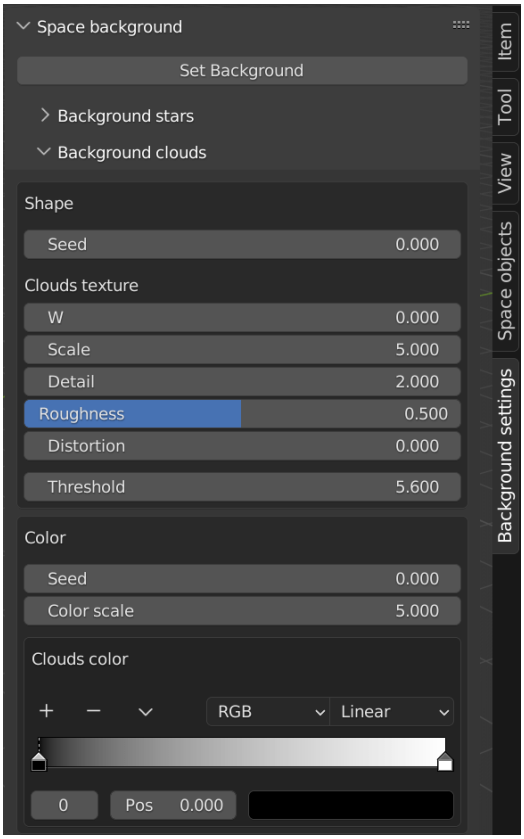
4.4 Objekt

Jak již bylo zmíněno, tak k tvorbě objektů byly použity dva postupy, zejména z důvodu snahy vylepšit implicitní přístup Blenderu:

- **Prostý operátor** - Jedná se o implicitní a jednoduchý způsob přidání jednoduchých objektů do scény, který je zabudovaná v Blenderu. Jde o vestavěné operátory, typicky ze třídy `bpy.ops.mesh`, konkrétně tímto doplňkem nejčastěji používaný operátor `bpy.ops.mesh.primitive_cube_add()`. Tento operátor slouží k přidání primitivní krychle, která je pak dále jednoduše upravitelná pomocí měřítka v jednotlivých osách k dosažení kýženého výsledku. Využíváno např. k tvorbě základního tvaru disku černé díry.
- **Tvorba jednotlivých vertexů a polygonů** - Jedná se o klasický způsob tvorby objektů ze „surových dat“. K vytvoření takového objektu slouží funkce `from_pydata()`, která má jako své parametry tři základní údaje potřebné k tvorbě objektu ve formě



(a) Část pro upravování hvězd - velikost, barva a množství.



(b) Část pro upravování „mraků“ v celkové barvě vesmírného pozadí.

Obrázek 4.4: Panel zobrazující možnosti úpravy pozadí.

polí - pole vertexů, pole hran a pole polygonů. Tyto údaje je potřeba vypočítat před zavoláním této funkce. V tomto doplňku je tento postup používán k tvorbě modifikované koule popsané v sekci 3.3.1. Detailnější postup výpočtu těchto potřebných údajů je rozepsán v následující sekci 4.4.1.

4.4.1 Vytvoření modifikované koule

Tvorba výchozích vertexů pro modifikovanou kouli je uložena v souboru `plain_object.py`. Celý postup vychází z dříve zmíněného ikosaedru, který je vytvořený pomocí tří obdélníků. Poměr stran těchto obdélníků je tzv. *Zlatý řez*. Tyto obdélníky jsou umístěny na každou osu a po určitém propojení jejich dvanácti vrcholů vznikne výsledný ikosaedr. Jeho jednotlivé trojúhelníkové strany jsou poté rozděleny na dílčí trojúhelníky (rozebrané v sekci 3.3.1). Proces probíhá ve funkci `divide_triangles` uložené v souboru `trinagle_division.py`. Takto rozdělený objekt má ovšem stále tvar ikosaedru. Aby bylo možné tímto objektem napodobit tvar koule, je potřeba pro každý vertex takto vytvořeného objektu změnit jeho vzdálenost od středu na konstantní hodnotu, která bude pro všechny vytvořené vertexy stejná. Tento proces probíhá ve funkci `project_to_sphere`, která takto vytvoří objekt

připomínající kouli, který po aplikování `smooth shade` modifikátoru (neboli modifikátoru plynulého stínování) je od koule prakticky nerozeznatelný.

Tento vytvořený kulový objekt je poté využíván doplňkem k tvorbě kulových objektů místo výchozích operátorů Blenderu pro tvorbu koule. Používáno pro tvorbu planety, slunce, měsíce, asteroidů, plynového obra a centra černé díry.

4.4.2 Úprava povrchu objektu

Vytvořená koule z předchozí sekce ovšem není dostatečná reprezentace některých objektů. Povrch koule je ještě nutné upravit dvěma způsoby podle typu výsledného objektu:

- **Krátery** - Krátery jsou tvořené v souboru `craters.py`. Nejdříve je na povrch výchozího objektu rozprostřen počet bodů zadaný uživatelem, které slouží jako středy tvořených kráterů. Každý kráter je instancí třídy `Crater` a má tak své separátní údaje jako hloubka, šířka nebo velikost hrany. Pomocí rovnic ze sekce 3.3.3 je poté vypočítán nový povrch upravovaného objektu postupně vůči každému kráteru. Díky tomu může být výpočet povrchu upravovaného objektu velmi časově náročný, neboť povrch může mít velmi mnoho vertexů. Pokud se dva krátery překrývají, tak je nová výška povrchu určena kombinovanou metodou, tudíž veškeré krátery které budou daný objekt upravovat budou mít daleko menší vliv než kráter první, jak bylo zmíněno v sekci 3.3.5. Používáno k tvorbě měsíce.

- **Šum** - Šum pro tvorbu terénu je počítán v souboru `planet_noise.py`. Pro každý bod jsou vypočítány tři varianty šumů ve funkcích `plains`, `mountains` a `hills` a nulová hodnota pro příslušnost bodu do rovného povrchu (voda). Poté je pomocí funkce `area_type` vypočítána příslušnost daného bodu k výše zmiňovaným šumům. Podle této příslušnosti jsou upraveny modifikátory těchto šumů a je sečten výsledný povrch. Rozdíl ve výpočtu je v momentě, kdy je rozhodnuto jestli bod přísluší do rovného povrchu nebo do povrchu upravovaného šumem.

  $$\texttt{surface\_change} \begin{cases} \texttt{area\_type\_retval} \leq \texttt{water\_amount} & 0 \\ \texttt{area\_type\_retval} > \texttt{water\_amount} & \sum(\text{noise functions}) \end{cases}$$

  Kdy `water_amount` značí množství povrchu planety, které bude mít nulová šum. Tento rozdíl má ovšem za následek velmi ostrou hranici mezi těmito dvěma typy povrchů, neboť ji tvoří hrany jednotlivých polygonů a lze tak rozlišit jednotlivé trojúhelníky. Tento problém je řešen v sekci 4.6.3.

4.5 Výsledný povrch

Pro úpravu povrchů již vytvořených objektů s neměnnými vertexy byly použity materiály. Díky použití právě materiálů může zdatnější uživatel jednoduše upravovat materiály vytvořené tímto doplňkem a docílit tak specifičtějších výsledku.

4.5.1 Tvorba materiálů

Jak již bylo řečeno, materiály jsou tvořené z tzv. *Nodes*, neboli uzlů. Každý uzel má specifický účel a používá se k docílení různých výsledků. Souhrn uzlů se nazývá *NodeTree* nebo také „strom“. Materiál je v podstatě zapouzdřený strom, který může mít libovolné vstupy

ze scény. Výstupem je samotný povrch definovaný pro každý bod polygonu, kterému je přidělen.

Tvorba každého materiálu je uložena vždy ve stejném souboru, jako je uložen operátor pro tvorbu daného objektu, neboť kromě planety je tvorba objektu a jeho povrchu prováděna pomocí jednoho operátoru. Pro jednodušší práci s materiálem a jednotlivými uzly byla vytvořena třída `MyNodeTree`, která má v sobě uložené jak jednotlivé uzly, tak i všechny propojení mezi nimi. Hlavní metodou této třídy je `my_link`, která zjednodušuje propojování jednotlivých uzlů, neboť jejím vstupem může být buď daný uzel a index jeho vstupu/výstupu, nebo přímo daný socket pro propojení.

Mezi nejčastěji používané uzly patří `Noise texture node` pro tvorbu šumu ze vstupního vektoru, `Color ramp node` pro převádění číselné hodnoty na barevný gradient nebo `Math node` pro nejrůznější výpočty.

4.5.2 Příklady materiálů

Samotné materiály se od sebe velmi liší v závislosti na modelovaném objektu. Objekty jako planeta, měsíc nebo slunce mají materiál, který čistě určuje barvu daného bodu na povrchu (zjednodušené příklady jsou na obrázku 4.5). Na rozdíl od toho objekty jako galaxie nebo mlhovina mají část materiálu určenou k vytvoření objektu pomocí proměnné `density`, což zaručuje, že části objektu, kde nebude zasahovat jeho výsledný tvar tvořený materiálem, budou průhledné a je tak dosaženo iluze složitého tvaru. Příklad materiálu, jeho uživatelského rozhraní a vykresleného výsledku na obrázku 4.6.

4.6 Speciální operátory

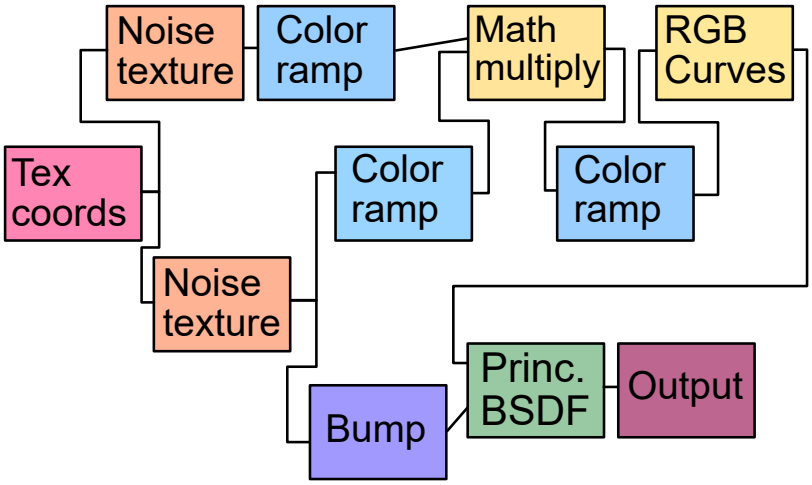
V této sekci jsou uvedené funkce, které mají velmi specifický účel a nelze je přímo zařadit do jiné kategorie. Také jsou zde funkce, které je díky jejich implementaci vhodné lépe popsat.

4.6.1 Tvorba pozadí

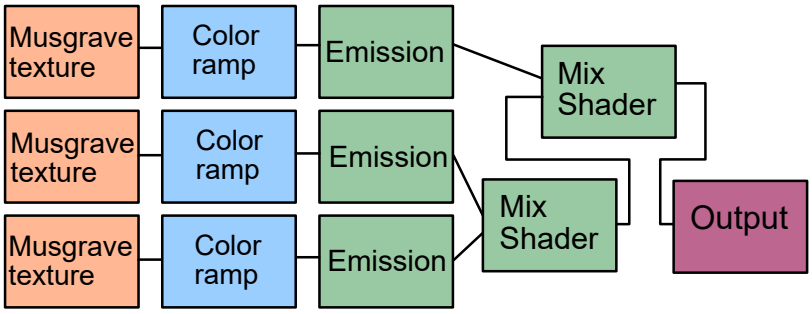
Tvorba pozadí je velmi podobná tvorbě povrchu libovolného objektu tvořeného tímto doplňkem. Samotnému pozadí totiž lze přiřadit materiál. Tento materiál má sice drobné odchylky v dostupnosti některých uzlů, ale v jádru jsou si velmi podobné. Tvorba celého pozadí je uložena v souboru `background.py`. Samotné pozadí je pak tvořeno tzv. Voroného texturou v kombinaci převodu hodnoty této textury na barevný gradient. Tato textura má vzhled sítě buněk (příklad na obrázku 4.7), kterou je možno spatřit v přírodě. Při správném typu ořezu (pomocí uzlu `Color_ramp`, jak je naznačeno na obrázku 4.8) těchto buněk pak vznikne rovnoměrné rozprostření bodů na plochu, které je možno využít jako zobrazení hvězd ve vesmíru. Zobrazení základu hvězd je naznačeno na obrázku 4.9.

4.6.2 Tvorba záře

Pro renderování videí a snímků vytvořených tímto doplňkem je předpokládáno, že se bude především používat render engine *Cycles*. Umožňuje daleko kvalitnější snímky, ale má jednu nevýhodu a tou je záře. Pokud by byl využitý engine *Eevee*, tak lze tuto záři, kterou vydávají objekty vyzařující světlo (slunce, hvězdy, galaxie) simulovat pomocí funkce, kterou má tento engine zabudovanou - Bloom. Tato funkce ovšem v *Cycles* není a je ji tedy nutné nahradit jiným způsobem. Vzhledem k tomu, že oba dva render enginy fungují na jiném principu, tak není možné v *Cycles* tuto funkci nikterak simulovat při prostém zobrazení



(a) Materiál tvořící povrch měsíce. Základním vstupem je šumová textura, která je závislá na konkrétních souřadnících bodu na povrchu.



(b) Materiál tvořící povrch slunce. Základním vstupem je variace šumové textury (Musgrave texture), která není závislá na vstupních souřadnících, neboť každá vrstva může být upravována separátně.

Obrázek 4.5: Příklady materiálů používané na úpravu povrchu vytvořených objektů.

scény jako v *Eevee*. Tohoto efektu lze dosáhnou jen úpravou výsledného vyrenderovaného snímku/videa a to v editoru `Compositing`. Tento editor opět na první pohled připomíná materiál s jednotlivými uzly. Z důvodu uživatelské přívětivosti se opět s uzly pracuje, navzdory tomu, že vstupem je vyrenderovaný snímek a jednotlivé uzly představují operace nad tímto snímkem. Sestavené uzly jsou zobrazené na obrázku 4.10.

Pro dosažení kýženého efektu je nutné použít uzel s názvem `Glare` s parametrem `glare_type` nastaveným na hodnotu `Fog Glow`. Tento uzel zajistí, že veškeré vykreslené body, které budou světlejší než specifická úroveň, budou mít kolem sebe přidanou záři, která bude demonstrovat jejich svit.

Použití render enginu *Cycles* má však i svou nevýhodu. Vyrenderované snímky mají v některých případech tendenci být poznamenané „šumem“ a vypadají poté jako neostré nebo nekvalitní. Toto lze vyřešit dvěma způsoby:

- **Nastavení render enginu** - Přímo render engine *Cycles* má zabudovaný parametr `Denoise`, který tento nechtěný šum odstraňuje. Tento proces ovšem prodlužuje samotné renderování a v tomto doplňku není standardně využíván.
- **Postrenderová úprava** - Stejně jako přidání záře, tak i odstranění šumu lze vyřešit přidáním dalšího uzlu do Compositing editoru. V tomto případě se jedná o uzel

s názvem `Denoise` a funguje velmi podobně jako odstraňování šumu přímo enginem, jen s tím rozdílem, že pracuje až přímo s vyrenderovaným snímkem. Tento postup je používán tímto doplňkem ve výchozím stavu a operátor pro zapnutí/vypnutí záře a odstranění šumu je na každém panelu pro práci s vesmírným objektem.

4.6.3 Úprava hranice rovného a nerovného povrchu tvořeného šumem

Pokud se u objektu typu planeta modifikuje povrch pomocí terénu a zároveň vstupní proměnná, která určuje procento vodní hladiny je větší než nula, tak vzniká problém plynulosti hranice dvou typů povrchů. Vzhledem k faktu, že celý objekt je tvořen polygony ve tvaru trojúhelníku a jednotlivé vertexy mají od svých sousedních vertexů stejnou úhlovou vzdálenost, tak v momentě, kdy se modifikuje povrch změnou vzdáleností od středu tak je velmi zřetelná hranice mezi typy povrchů, která má neplynulý a velmi „zubatý“ tvar, neboť je tvořena čistě stranami rovnostranných trojúhelníků.

Tento problém lze vyřešit poměrně jednoduše, a to změnou polohy hraničních vertexů. Tento proces je implementován funkcí `water_edges` obsaženou v souboru `water_border.py`. Princip změny pozice hraničních vertexů je následovný:

1. Jako první se pomocí cyklu projdou všechny vertexy daného objektu a je zkontrolováno, které vertexy splňují podmínku hraničního vertexu. Hraniční vertex musí mít alespoň jeden sousedící vertex se vzdáleností od středu rovnou nebo menší původnímu poloměru koule (Dopněk neimplementuje vodní hladiny jiné „nadmořské výšky“) a zároveň alespoň jeden s touto vzdáleností větší. Tím jsou vyloučeny veškeré vertexy, které jsou uprostřed rovného nebo upraveného povrchu.

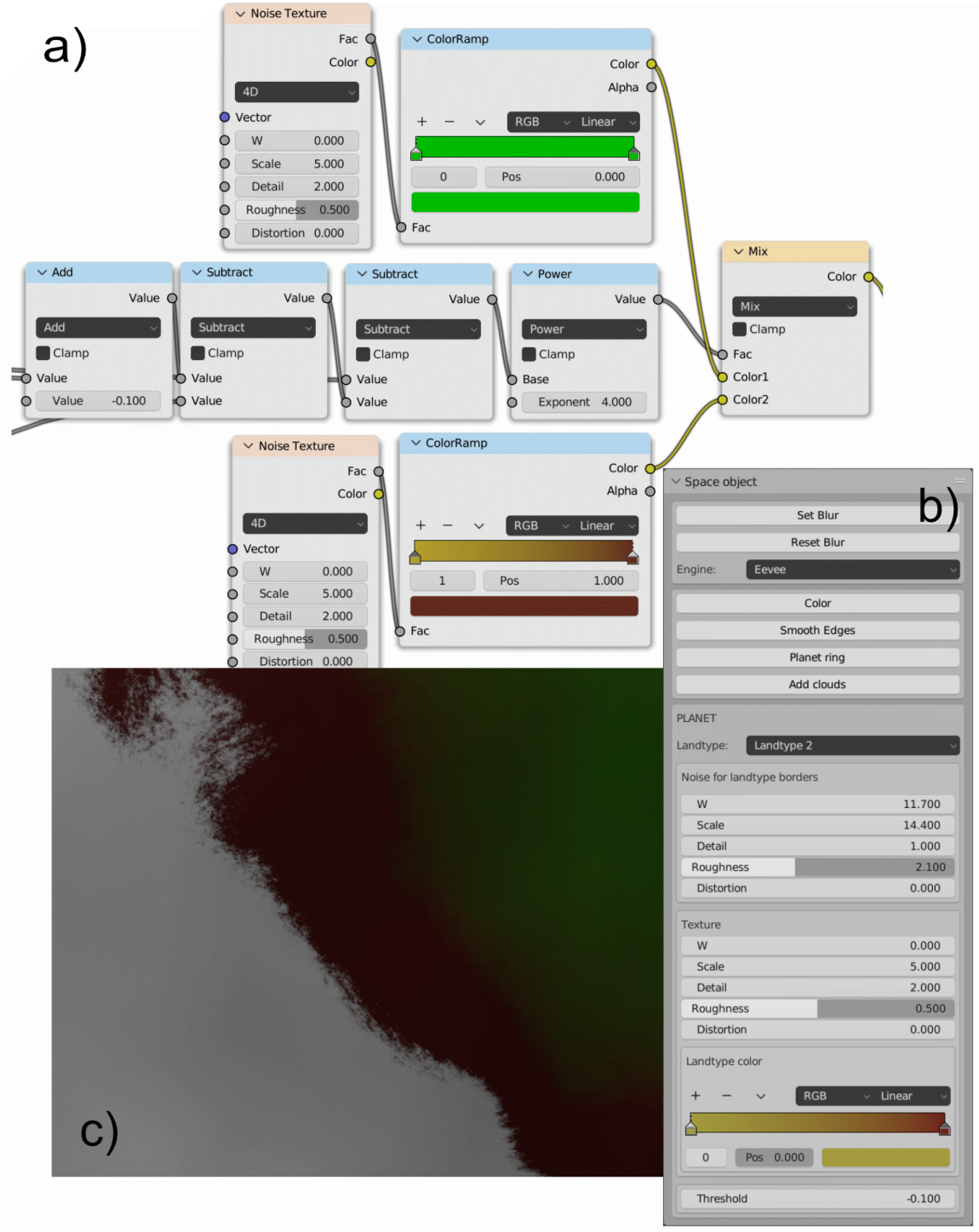
2. Takto vznikne pole s vertexy, které jsou hraniční. Mezi nimi se ovšem mohou nacházet i vertexy redundantní (splňují podmínky, avšak není pomocí nich tvořená nejširší hranice), neboť jeden hraniční vertex může mít mezi sousedícími vertexy několik vertexů hraničních. Díky tomu by pak výsledná hranice nebyla křivka složená z vertexů jdoucích jeden za druhým. Pro každý hraniční vertex je tedy zkontrolováno, zda v případě jeho odeberání z hraničních vertexů, tak zbylé hraniční vertexy budou stále tvořit smyčky, neboli každý hraniční vertex bude sousedit právě se dvěma hraničními vertexy (stejně jako hranice např. jezera je tvořena smyčkou). Pokud ano, tak je tento vertex odstraněn.

3. Po odstranění redundantních vertexů z vybraného pole, tvoří toto pole hraniční vertexy, které tvoří křivky ve tvaru minimálně jedné smyčky.

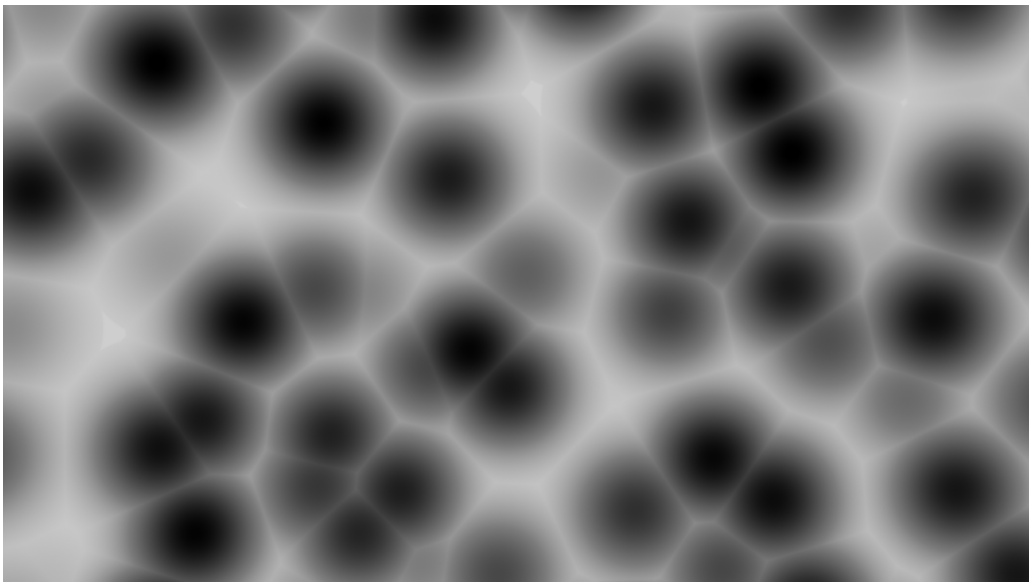
4. Poloha těchto vertexů je poté upravena ve `for` cyklu, kde je poloha aktuálního vertexu přiblížena středu vzdálenosti mezi jeho sousedícími hraničními vertexy.

5. Pro realističtější efekt jsou ještě upraveny i polohy sousedících vertexů (jsou zprůměrovány vzálenosti od jeho sousedních vertexů), které nejsou hraniční, ale to jen v případě, že jejich vzdálenost od aktuálně upravovaného vertexu je výrazně delší, než průměr vzdálenosti jeho sousedících vertexů. Tímto se eliminují výrazně velké polygony, které by jinak mohly tímto postupem vzniknout.

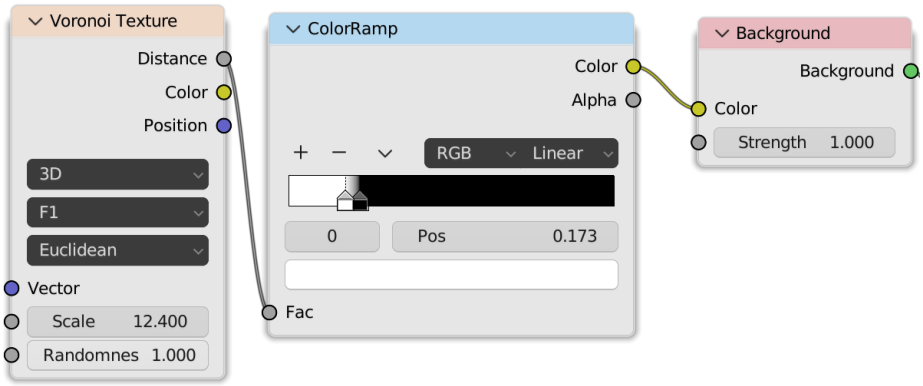
6. Tímto je dosaženo daleko plynulejší hranice rovného povrchu a povrchu tvořeného šumem, což je demonstrováno na obrázku 4.11.



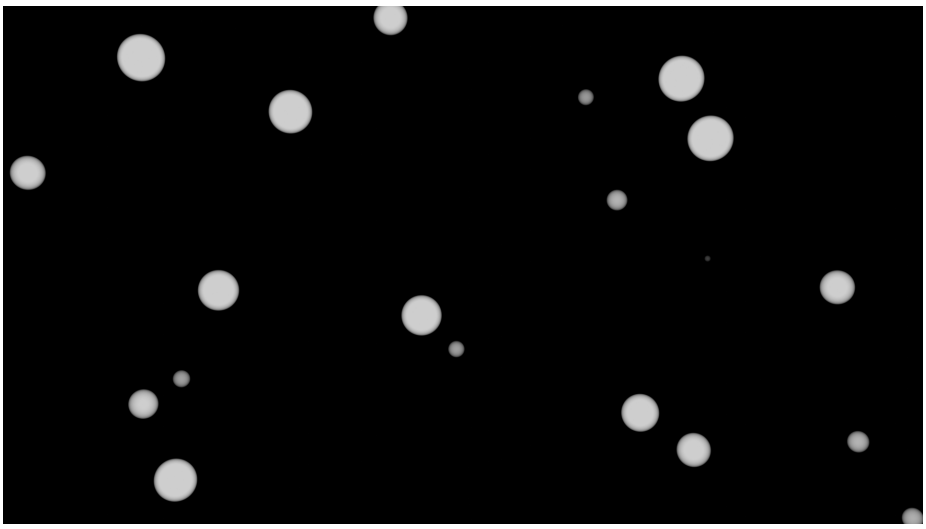
Obrázek 4.6: Zobrazení a) části materiálu, b) jeho uživatelského rozhraní a c) vykresleného povrchu, která je modifikovaný touto částí materiálu.



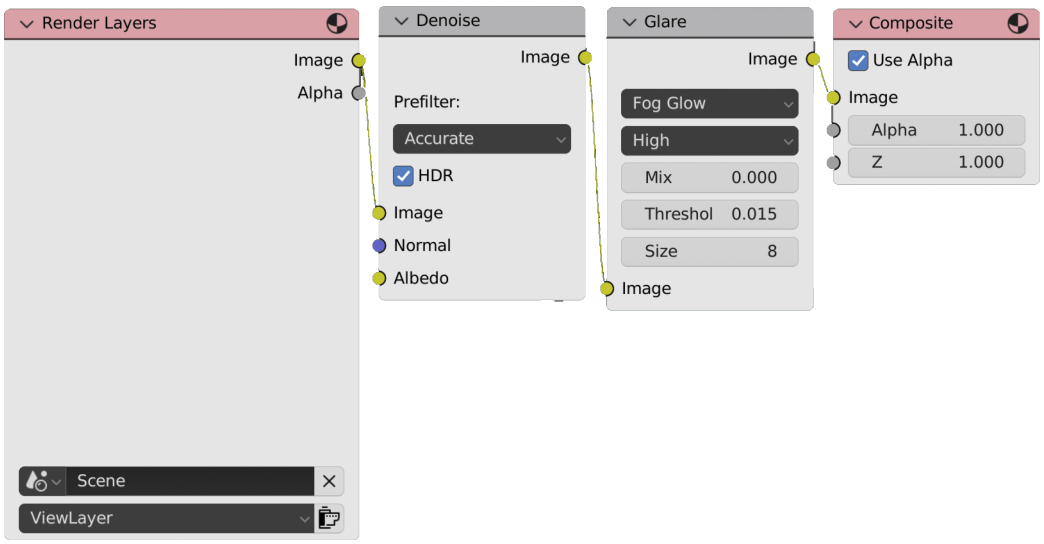
Obrázek 4.7: Zobrazení Voroného textury.



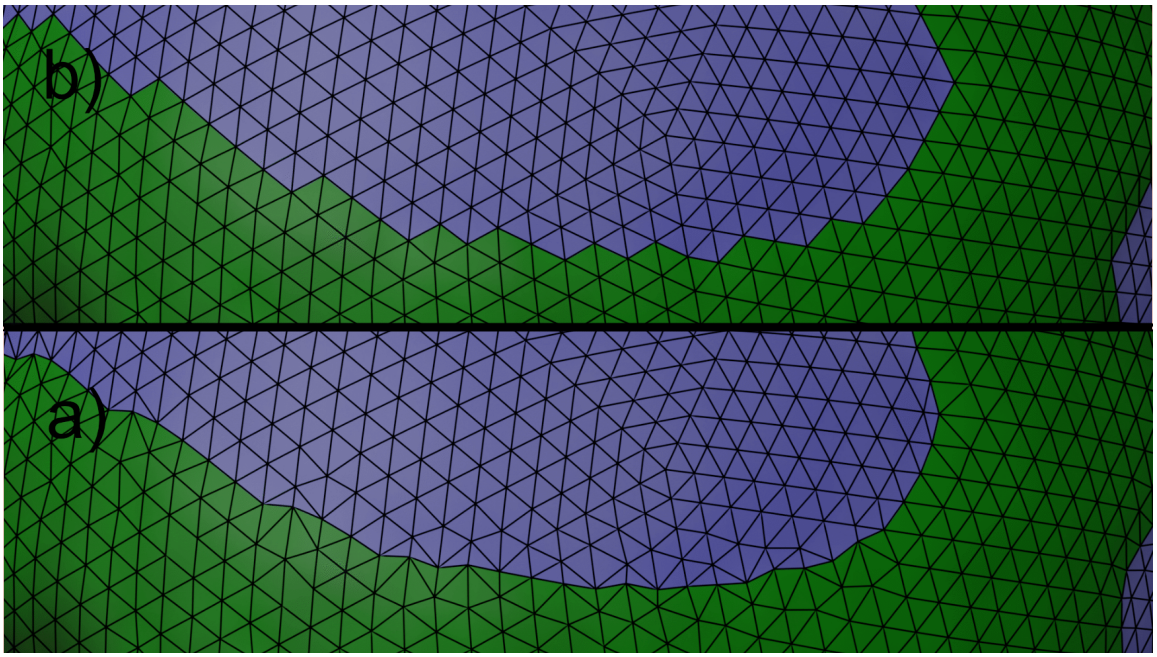
Obrázek 4.8: Zobrazení ulzů tvožící základ hvězd.



Obrázek 4.9: Zobrazení ořezane voroného terxtury, která slouží jako základ pro tvorbu hvězd.



Obrázek 4.10: Zobrazení uzlů v Compositing editoru pro přidání záře a odstranění šumu.



Obrázek 4.11: Ukázka upravené hranice mezi dvěma typy povrchů. a) Upravené hranice povrchů. b) Hranice bez úpravy.

Kapitola 5

Testování a publikování

V této kapitole jsou rozebrány způsoby testování a publikování hotového doplňku.

5.1 Testování

Testování doplňku muselo zohlednit zejména časovou náročnost vytváření objektů a složitost vytvořeného objektu pomocí nové metody.

5.1.1 Složitost modifikované koule

Jak již bylo zmíněno, pro tento doplněk byl zvolen nový přístup dělení polygonů pro tvorbu koule. Od výchhozí metody Blenderu se liší hlavně v možnosti lépe kontrolovat počet vertexů, které bude koule obsahovat, což bylo dokázáno tvořením a následným dělením jednoho trojúhelníku. Ze dvaceti takových trojúhelníků je pak v samotném doplňku vytvořen celý ikosaedr.

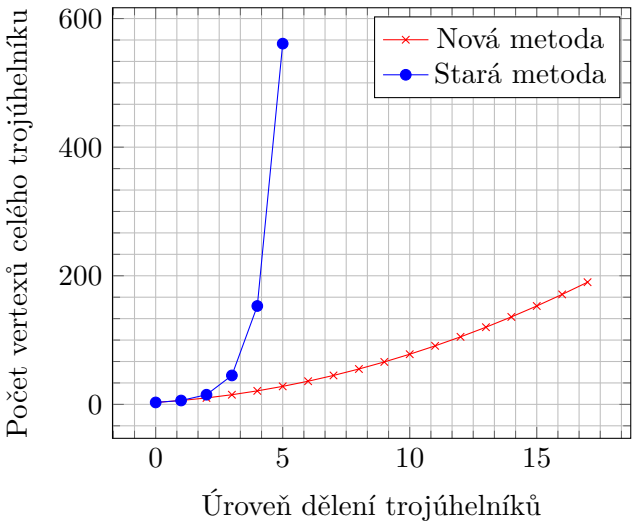
Test byl proveden vytvořením trojúhelníku, který byl následně dělen po jednotlivých úrovních. Výsledek tohoto testu je uveden v grafu 5.1, ze kterého je jasně vidět, že výchozí mtoda Blenderu už při malých úrovních dělení trojúhelníku dosahuje velmi velkého počtu vertexů. Následně bylo zjištěno, že rozdílnou změnu počtu vertexů mezi oběma metodami lze pozorovat na počtu vertexů na hranách trojúhelníku. Pro tyto vertexy byly zjištěny vztahy, které udávají počet vertexů na hraně v závislosti na hraně předchozí:

- Pro novou metodu doplňku je nový počet vertexů na straně trojúhelníku dán vztahem $x_{n+1} = x_n + 1$, kde x_{n+1} je počet verexů právě rozdělené strany, x_n je počet vertexů dělené strany a n je úroveň aktuálního dělení.
- Na rozdíl od toho, výchozí metoda má vztah daný $x_{n+1} = (2x_n) - 1$, kde opět x_{n+1} je počet verexů právě rozdělené strany, x_n je počet vertexů dělené strany a n je úroveň aktuálního dělení.

  Skript pro výpočet bodů trojúhelníku - `vert_test.py`.

5.1.2 Časová náročnost modifikované koule

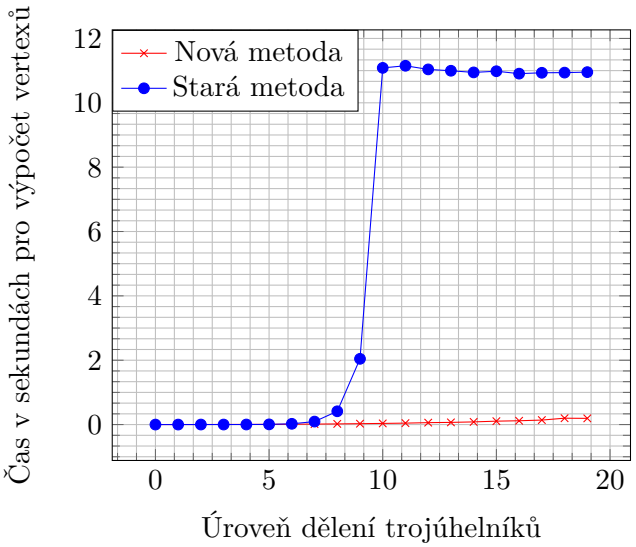
Časová náročnost úzce souvisí s předchozím testem pro počet vertexů. Vzhledem k tomu, že počet vertexů výchozí metody stoupá daleko rychleji, než u metody nové, tak se dá předpokládat, že i časová náročnost bude následovat podobnou křivku.



Obrázek 5.1: Zobrazení počtu vrcholů trojúhleníku s rostoucí úrovní dělení.

Z výsledku testu 5.2 je zřejmé, že pro prvních několik úrovní dělení to také platí, avšak výchozí metoda Blenderu u úrovně deset se zastaví na desetisekundové hranici a tu dále prakticky nepřesáhne. Nové metodě doplňku sice potřebný čas stoupá pomaleji, avšak nezastaví se na určitém stabilním čase, ale stoupá do nekonečna. Tudíž při velkých úrovních dělení bude výchozí metoda Blenderu rychlejší pro tvorbu kulového objektu, než metoda nová.

Skript do Blenderu pro testování časové náročnosti - `time_test.py`.



Obrázek 5.2: Zobrazení času potřebného pro výpočet nového kulového objektu v aplikaci Blender (Na procesoru AMD Ryzen 5 3400G).

5.2 Publikování

Výsledný doplněk byl publikován a je dostupný platformě na *GitHub*.[1] Dopněk byl po celou dobu vývoje udržován na této platformě, avšak v jiném repozitáři. Tato skutečnost umožňuje, že celý doplněk je open-source a jednoduše stáhnutelný. V souboru `Readme.md` je k dispozici základní uživatelské příručka s ukázkami objektů modelovaných tímto doplňkem. Součástí publikace je taktéž demonstrační video,[2] které ukazuje základní funkčnost doplňku.

[1] Doplněk dostupný na https://github.com/MarekHlavka/Planet-and-space-objects-generator
[2] Video dostupné na platformě YouTube: https://youtu.be/c0Qs0dUp8_I

Kapitola 6

Závěr

Cílem práce bylo vytvořit doplněk do aplikace Blender, pomocí kterého je možné procedurálně generovat jednotlivé objekty nacházející se ve vesmíru.

K řešení práce bylo nejprve potřebné nastudovat vesmírné objekty, zejména tvorbu terénu a kráterů, neboť jednotlivé typy kráterů a terénů se mezi sebou značně liší. Dále bylo potřeba nastudovat teorii a praktické použití šumu, konkrétně Perlinova šumu, pro realistickou simulaci modelovaných objektů. Následně byla navrhnuta první verze uživatelského rozhraní, která byla později upravována ke zlepšení práce s doplňkem. Poté byly pomocí série návrhů a implementací realizovány jednotlivé typy objektů v rámci celého doplňku. K výše uvedému bylo potřeba nastudovat a získat zkušenosti nejen se samotnou aplikací Blender ale také s jeho API, na kterém je celý doplněk založen.

Doplněk je schopný modelovat několik vesmírných objektů a nabízí přímé úpravy povrchu u některých z nich. Veškerý výsledný povrch je implementován procedurálně - lze vytvořit nespočet různých variant objektů, i nerealisticky vypadajících. Žádný povrch není modelován pomocí textury, tudíž povrch žádného objektu není závislý na žádném vstupním snímku skutečně existujícího vesmírného objektu. V příloze A se nachází příklady modelů, které je doplněk schopen modelovat.

Výsledkem práce je open-source publikovaný doplněk na platformě *GitHub*, kde si jej uživatel může stáhnout a začít s ním modelovat vesmírné objekty.

Doplněk byl implementován tak, aby jej bylo možné následně různě rozšiřovat. Například tvořit celé sluneční soustavy, které budou realistické skrze oběžné dráhy, gravitační síly, vlastnosti planet v závislosti na jejich poloze vůči hvězdě nebo samotné přidávání dalších druhů vesmírných objektů pro modelování.

Literatura

[1] *Blender Python API overview*. Jan 2022. Dostupné z: https://docs.blender.org/api/current/info_overview.html.

[2] Bailey, M. a Cunningham, S. *Graphics Shaders: Theory and Practice*. USA: A. K. Peters, Ltd., 2009. ISBN 1568813341.

[3] Ebert, D., Musgrave, F., Peachey, D., Perlin, K., Worley, S. et al. *Texturing and Modeling: A Procedural Approach: Third Edition*. Elsevier Inc., prosinec 2002. ISBN 9781558608481.

[4] Finlay, W. *Concise Catalog of Deep-sky Objects: Astrophysical Information for 500 Galaxies, Clusters and Nebulae*. Springer London, 2003. ISBN 9781852336912. Dostupné z: https://books.google.cz/books?id=t35J1QRSFAEC.

[5] Kučerová, H. Pojem modelu a pojmový model v informační vědě. *Knihovna: knihovnická revue*. 2018, sv. 29, č. 2, s. 5–32. ISSN 1801-3252.

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

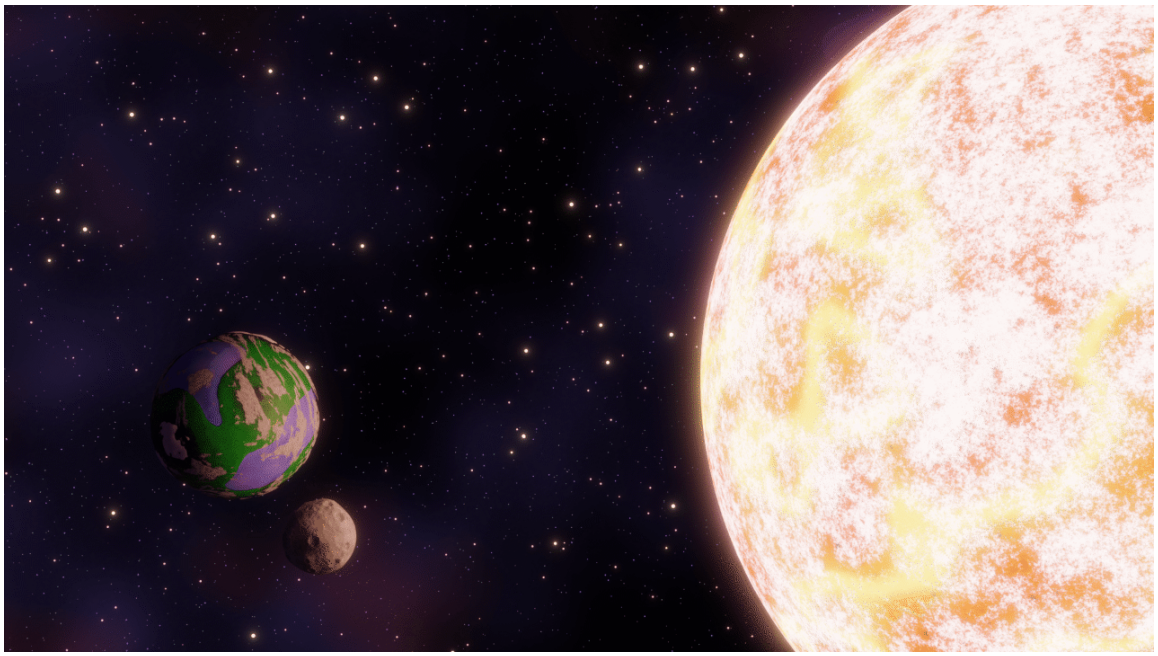
[7] Parkes, S., Martin, I., Dunstan, M. a Matthews, D. Planet Surface Simulation with PANGU. In: *Space OPS 2004 Conference*. DOI: 10.2514/6.2004-592-389. Dostupné z: https://arc.aiaa.org/doi/abs/10.2514/6.2004-592-389.

[8] Peringer, P. a Hrubý, M. *Modelování a simulace* [Materiál k přednášce]. FIT VUT Brno.

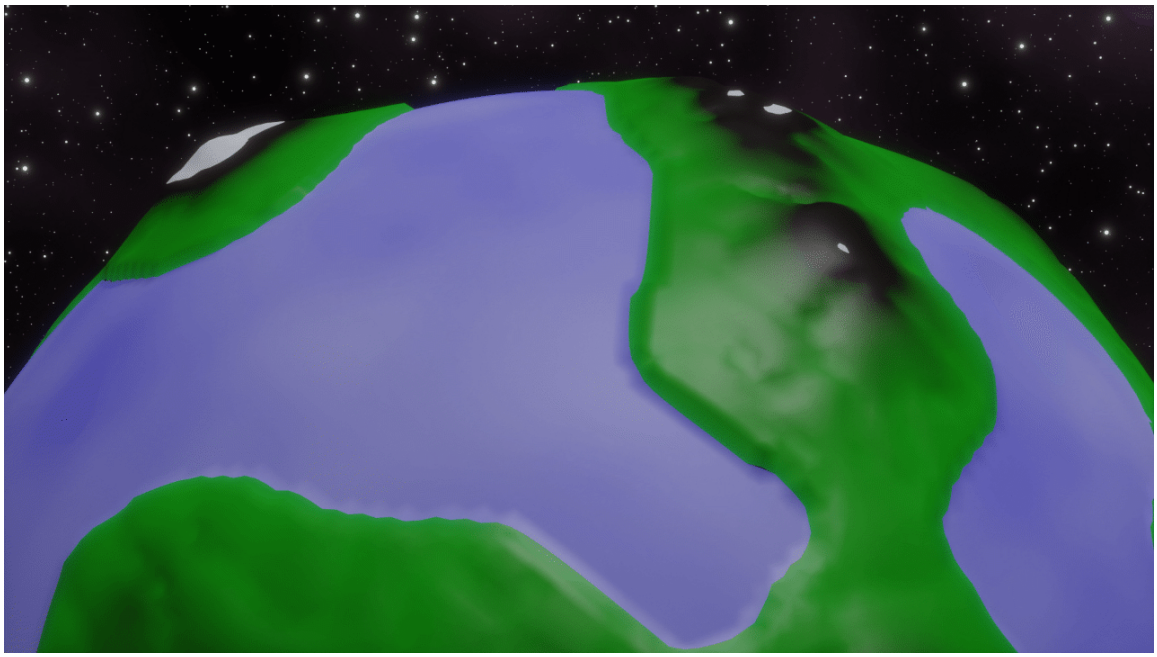
[9] Sulentic, J., Dreyer, J. a Tifft, W. *The Revised New General Catalogue of Nonstellar Astronomical Objects*. University of Arizona Press, 1973. ISBN 9780816504213. Dostupné z: https://books.google.cz/books?id=IZjvAAAAMAAJ.

Příloha A

Výstupy doplňku

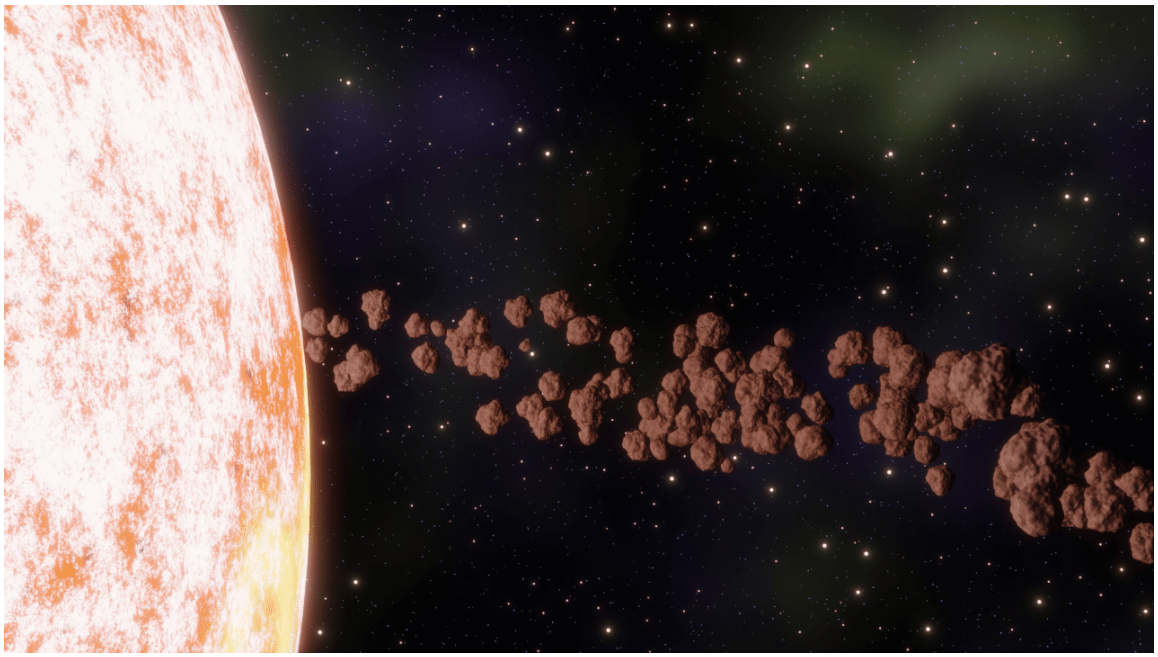


Obrázek A.1: Slunce s planetou a měsícem

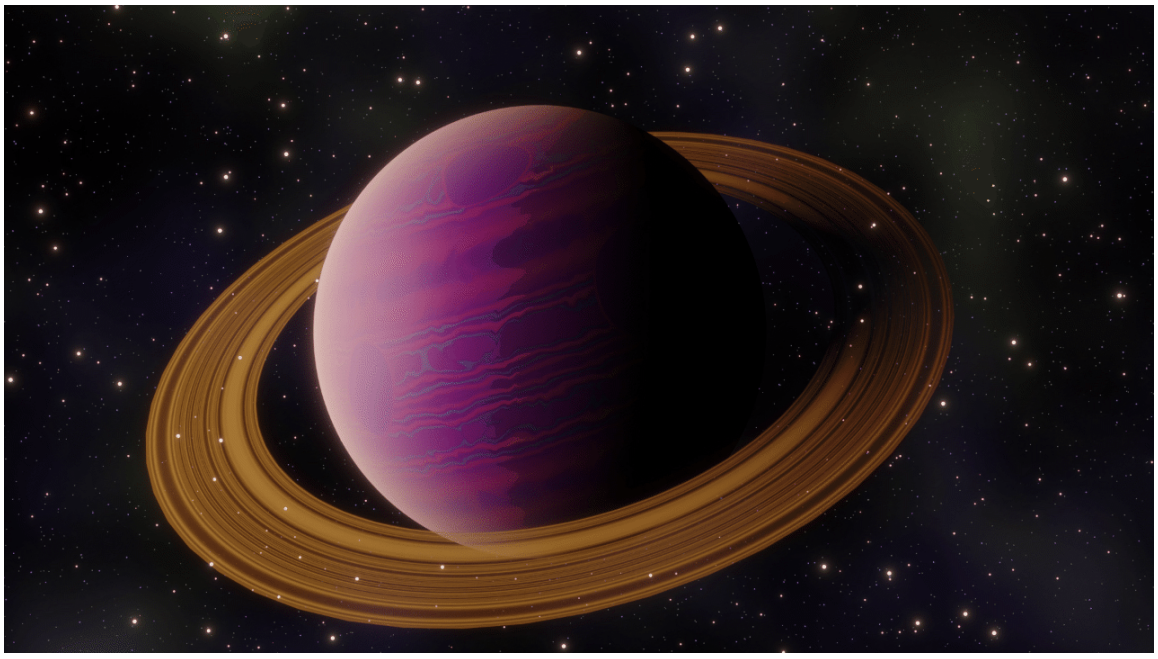


Obrázek A.2: Detailnější zobrazení výsledného povrchu.

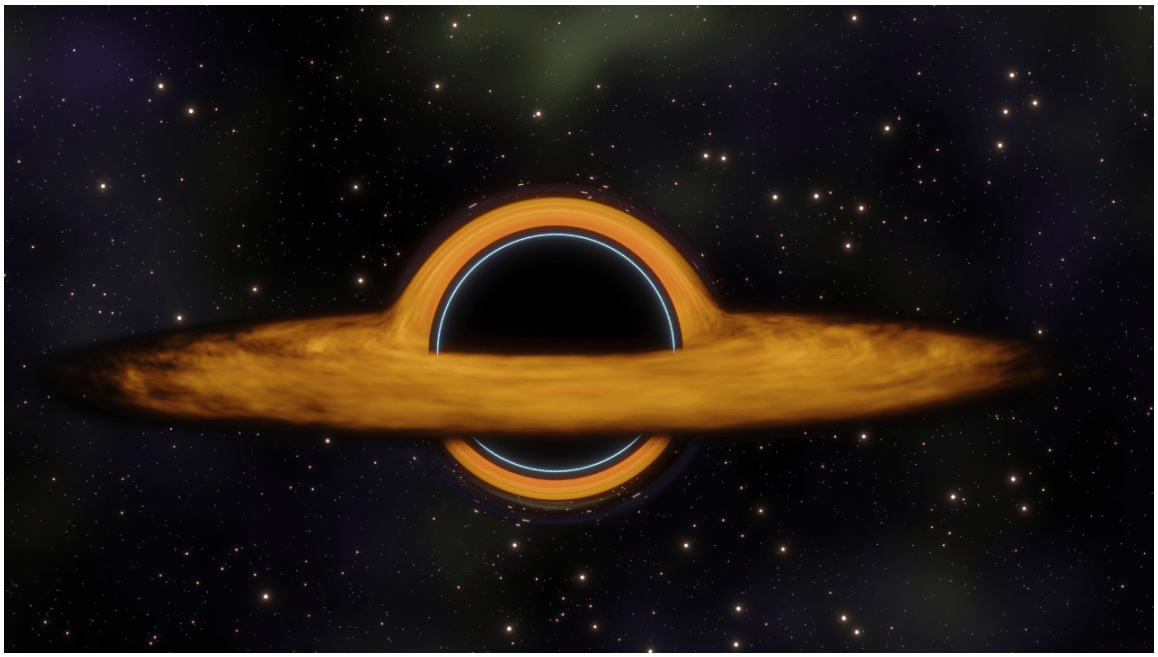
Obrázek A.3: Detailnější zobrazení kráterů.



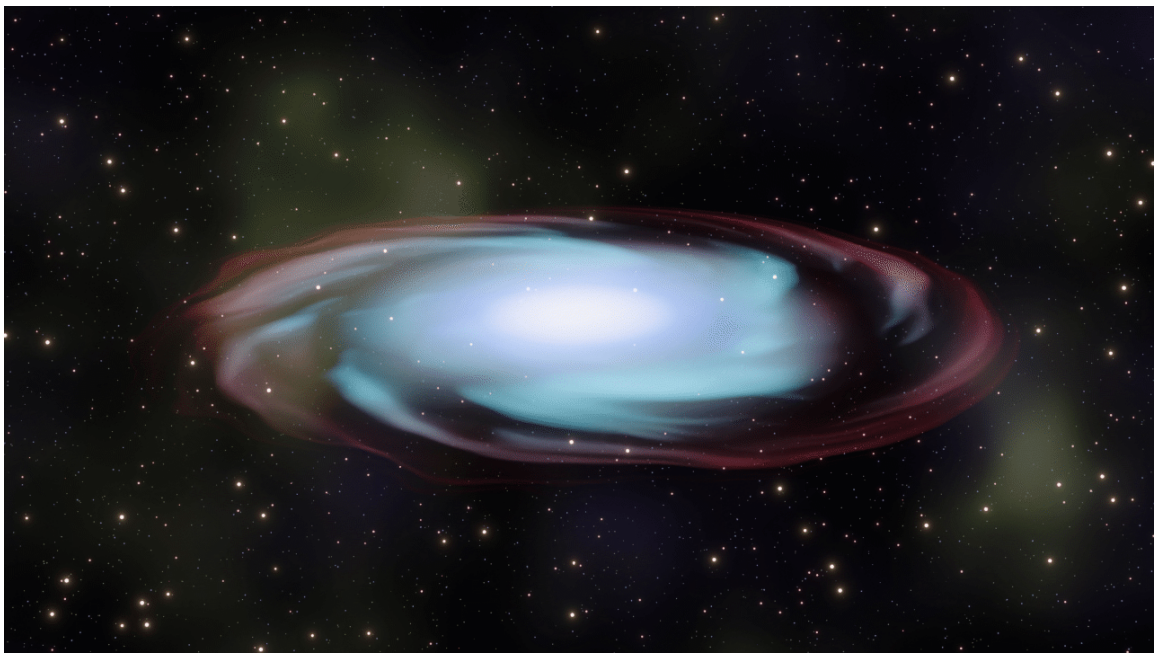
Obrázek A.4: Slunce s polem asteroidů



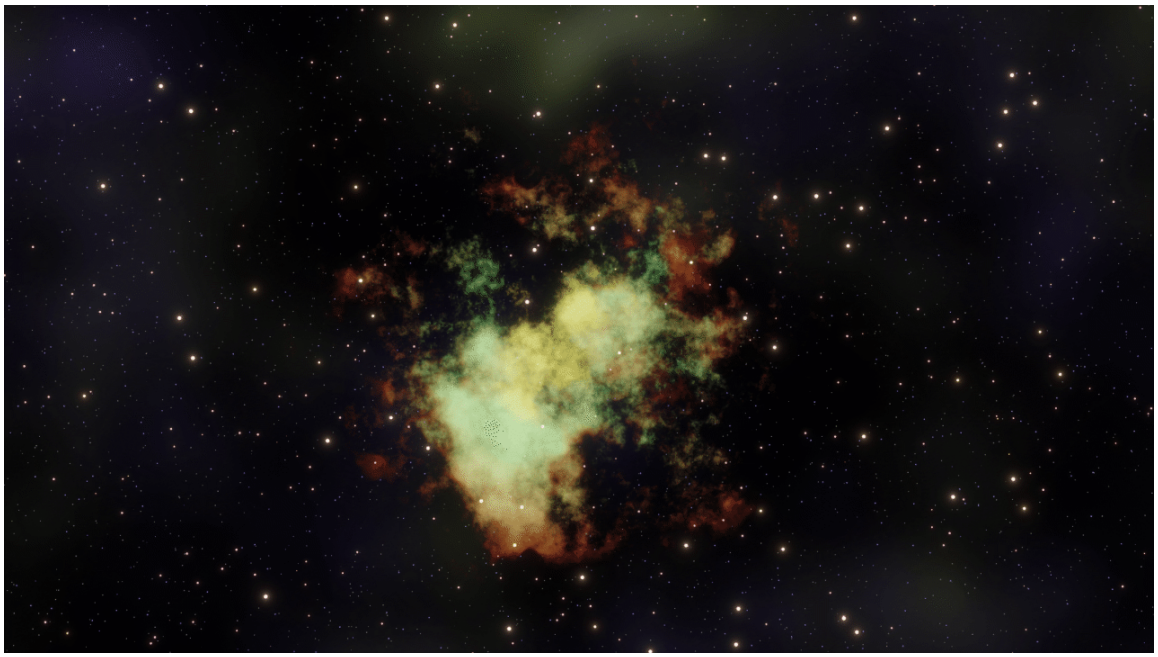
Obrázek A.5: Plynový obr s planetárním prstencem



Obrázek A.6: Černá díra



Obrázek A.7: Galaxie



Obrázek A.8: Mlhovina