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

PROCEDURÁLNÍ GENEROVÁNÍ V UNITY

PROCEDURAL GENERATION IN UNITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PAVEL GOŠ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. TOMÁŠ MILET,**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student:	**Goš Pavel**
Program:	Informační technologie
Název:	**Procedurální generování v Unity** **Procedural Generation in Unity**
Kategorie:	Počítačová grafika

Zadání:

1. Nastudujte engine Unity a metody procedurálního generování.
2. Navrhněte plugin pro generování otexturovaných modelů.
3. Implementujte navržený plugin.
4. Zhodnoťte dosažené výsledky a vytvořte demonstrační video.

Literatura:

- dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a prototyp pluginu.

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

Vedoucí práce:	**Milet Tomáš, Ing.**
Vedoucí ústavu:	Černocký Jan, doc. Dr. Ing.
Datum zadání:	1. listopadu 2019
Datum odevzdání:	28. května 2020
Datum schválení:	1. listopadu 2019

Abstrakt

Cílem této práce je vytvoření procedurálního generátoru podzemních kobek v herním enginu Unity. V práci lze najít popis nejdůležitějších systémů tohoto enginu. Zároveň je zde popsán vývoj a implementace procedurálního generování půdorysu mapy za využití Perlinova šumu a její následné výplně objekty.

Abstract

The goal of this thesis is to develop a procedural generator of dungeons in the Unity game engine. The description of the most important parts of the Unity engine can be found in this thesis. The development and implementation of procedural generation of map layout using Perlin noise and interior decoration is described here as well.

Klíčová slova

Unity, herní engine, procedurální generování, L-systém, Perlinův šum, generování polygonální sítě, plugin

Keywords

Unity, game engine, procedural generation, L-system, Perlin noise, Polygon mesh generation, plugin

Citace

GOŠ, Pavel. *Procedurální generování v Unity.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Milet,

Procedurální generování v Unity

Prohlášení

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

.....
Pavel Goš
28. května 2020

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Tomáši Miletovi za odborné vedení mé práce, cenné rady, podnětné připomínky a čas, kterým přispěl k vypracování této práce.

Obsah

1 Úvod **3**

2 Teorie **4**

2.1 Procedurální generování 4

2.1.1 Generátor pseudonáhodných čísel 4

2.1.2 Perlinův šum 5

2.1.3 Formální gramatika 5

2.1.4 L - systém 6

2.2 Typy okolí pixelu 6

2.3 Voxel 7

2.4 Polygonální Síť 7

2.5 Pochodující čtverce 8

2.6 Herní engine 8

2.7 Unity 8

2.7.1 Nástroje editoru 10

2.7.2 Konzole Editoru 10

2.7.3 Inspektor 10

2.7.4 Komponenty 11

2.7.5 GameObject 11

2.7.6 Scéna 12

2.7.7 Asset Store 12

2.7.8 Herní Mód 12

2.7.9 ScriptableObject 12

3 Návrh řešení **13**

3.1 Návrh mapy 13

3.2 Půdorys mapy 13

3.2.1 Generování místností 13

3.2.2 Generátor chodeb 15

3.2.3 Generátor Perlinova šumu 16

3.2.4 Spojení generátoru chodeb a generátoru místností 16

3.2.5 Nástroje pro větší míru přizpůsobení vzhledu mapy 17

3.3 Vizualizace vygenerované mapy 18

3.4 Úprava Inspektoru 18

3.5 Generování vnitřní výplně mapy 19

3.5.1 Práce s objekty určené pro výplň 19

3.6 Prohlížení mapy pomocí kamery v herním režimu 19

4 Implementace řešení **21**
4.1 Nastavení generátoru v hierarchii scény 21
4.2 Generování mapy 22
4.2.1 Generátor místností 22
4.2.2 Generátor chodeb 23
4.2.3 Finalizace půdorysu 24
4.3 Perlinův šum 25
4.4 Generátor polygonální sítě 26
4.5 Generování objektů interiéru 28
4.6 Implementace kamery pro prohlížení interiéru 28
4.7 Vzhled výsledného vnitřku mapy 29

5 Závěr **31**

Literatura **32**

Kapitola 1

Úvod

Cílem práce bylo navrhnout a implementovat postupy procedurálního generování. Tyto postupy poté poskládat dohromady a vytvořit z nich plugin pro herní engine Unity. Tento plugin je konkrétně zaměřený na procedurální generování podzemních kobek, aby tak usnadnil vývojářům vývoj her, které obsahují podzemní kobky ve svých úrovních. Usnadnění spočívá v tom, že vývojář nemusí manuálně vytvářet mapu, ale stačí za pomocí několika tlačítek a nástrojů opakovaně generovat náhodnou mapu, dokud s ní vývojář není spokojený.

Moje volba enginu byla založena na základě základních zkušeností z vlastních pokusů o vývoj her a statistik, ze kterých vyplývá, že opravdu velké množství her na trhu je vyvíjeno v enginu Unity. Navíc má Unity velké množství různých tutoriálů, které usnadňují nováčkům počáteční seznamování se s herním enginem. Dále také poskytuje velice kvalitní dokumentaci, díky níž není problém cokoliv nastudovat a s pomocí enginu implementovat.

Generování je v tomto pluginu rozděleno na dvě části. Tyto části tvoří generování půdorysu a generování interiéru. Užití pluginu je poté takové, že se nejdříve vygeneruje půdorys mapy na základě zvolených parametrů generátoru. Poté se přidají objekty interiéru do seznamu objektů v druhém generátoru. Ten vygeneruje objekty náhodně v rozmezí již vygenerovaného půdorysu.

Kapitola 2

Teorie

V této kapitole a jejích podkapitolách jsou popsány a vysvětleny odborné pojmy užité v textu.

2.1 Procedurální generování

Procedurální generování [16] je soubor metod k vytvoření částečně náhodného obsahu pomocí počítače. Generování je řízeno danými pravidly, proto je jen částečně náhodné. Bez těchto pravidel by generování bylo chaotické a tudíž nevyužitelné.

Jeho účelem je vytvoření velkého množství obsahu s minimální námahou. Program tudíž s minimálním uživatelským zásahem vytváří obsah pro hráče. Toto urychluje vývoj her a může zvýšit i motivaci hru hrát stále dokola a udržet tak hráče ve hře. Tohoto využívá např. hra Minecraft, kde je procedurálně vygenerovaný terén, tudíž hráč hraje pokaždé v novém unikátním světě. Dalším příkladem je hra Diablo I, která má procedurálně vygenerované úrovně. Procedurálně lze ve hrách generovat např. terén, města, zbraně či nepřátele. Tato práce je zaměřena na generování podzemních kobek - jejich půdorysu a obsahu.

2.1.1 Generátor pseudonáhodných čísel

Na začátek je třeba zmínit, že neexistuje generátor opravdu náhodných čísel [12], proto je pouze pseudonáhodný. Počítač pracuje jen s předem danými instrukcemi, tudíž není schopen náhodnosti.

Pseudonáhodné generování generuje posloupnost čísel na základě jiných čísel. Kvůli determinističnosti je tato posloupnost vygenerovaných čísel periodická, tedy po určité době se začne opakovat. Toto lze však vyřešit dostatečně dlouhou periodou, takže není perioda detekovatelná.

Pro generování pseudonáhodných čísel existuje řada různých algoritmů. Nejčastěji generátory využívají princip lineárního kongruentního generátoru (vzorec viz. 2.1).

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

kde X je sekvence pseudonáhodných hodnot a platí:

- $m,\ 0 < m$ číslo, po jehož dělení se získá zbytek,
- $a,\ 0 < a < m$ násobitel,
- $c,\ 0 \leq c < m$ inkrement,

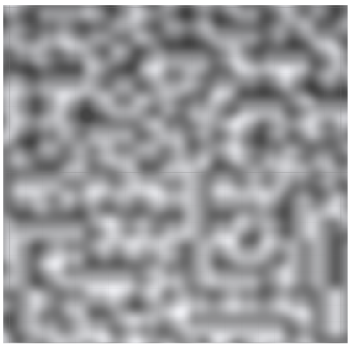
- $x_0,\ 0 \leq x_0 < m$ semínko/počáteční hodnota

2.1.2 Perlinův šum

Perlinův šum [1] (viz. obrázek 2.1) je vhodný pro generování grafického šumu. Je to pseudonáhodný vzor hodnot mezi hodnotami 0 a 1. Byl vytvořen v 80. letech Kenem Perlinem. Od jeho vytvoření je tento šum hojně využíván, např. v již zmíněné hře Minecraft. Důvodem úspěchu tohoto šumu je jeho přirozeně vypadající vzhled, způsoben přirozeně seřazenou posloupností čísel. Proto se velmi často využívá při procedurálním generování, např. při generování výškové mapy terénu, textur ohně nebo mraků.

Tato metoda využívá šumové funkce různé intenzity a měřítka. Perlinův šum je výsledkem součtu těchto šumových funkcí, přičemž každá tato funkce má jiné měřítko a intenzitu. Počet těchto funkcí se nazývá oktávy a pořadí funkce v součtu je oktáva.

Implementace se typicky skládá z třech kroků: definice mřížky s náhodnými gradient vektory, výpočet skalárního součinu vektorů a interpolace mezi těmito hodnotami.



Obrázek 2.1: 2D Perlinův šum

2.1.3 Formální gramatika

Formální gramatika [6] [7] se skládá z počátečního symbolu, ze kterého pomocí sady přepisovacích pravidel vzniká řetězec. V případě, že je jen jediný postup pro generování každého slova, pak je gramatika jednoznačná. Formální definice: Gramatika G je čtveřice (N, Σ, P, S), kde

- N je konečná množina neterminálů (znaky nepatřící do abecedy)
- Σ je konečná množina terminálů (znaky abecedy) tak, že žádný symbol nepatří do N a Σ zároveň
- P je konečná množina přepisovacích pravidel. Každé pravidlo je tvaru $(\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*$
- S je prvek z N nazývaný počáteční symbol

Pro příklad mějme abecedu obsahující symboly 'a' a 'b' a počáteční symbol 'S' a pravidla:

- $S \rightarrow aSb$
- $S \rightarrow ba$

Aplikujeme-li pravidla na počáteční symbol, dostaneme řetězce: $\{ba,\ abab,\ aababb,\ ...\}$

2.1.4 L - systém

L-systém [10] [11], též zvaný Lindenmayerův systém, je forma formální gramatiky. Skládá se z abecedních znaků, ze kterých vznikají řetězce. Pro vznik těchto řetězců existuje sada pravidel určující, jak se má příslušný symbol rozvinout. L-systémy jsou vhodné pro generování křivek, fraktálů či buněčných organismů. Původně byly vyvinuty pro popis růstu řas. Typické je však vyobrazení stromů či různých trav vzniklé pomocí L-systému. Základním typem je tzv. DOL-systém. Jedná se o deterministický bezkontextový L-systém. Formálně je to trojice $G = (\Sigma, S, P)$

- Σ je neprázdná množina symbolů, nebo-li abeceda, Σ^* je množina všech slov nad abecedou Σ, Σ^+ je množina všech neprázdných slov nad abecedou Σ,
- $S \in \Sigma^+$ konečné neprázdné slovo z abecedy Σ zvané axiom či semínko. Definuje počáteční stav L-systému,
- $P \subset \Sigma \times \Sigma^*$ je konečná množina přepisovacích pravidel
  - přepisovací pravidlo se zapisuje ve tvaru , $a \to S$, kde $a \in \Sigma$ a $S \in \Sigma^*$. Toto pravidlo definuje přepsání symbolu na slovo, které může být i prázdné
  - pro symbol $b \in \Sigma$, který se neobjevuje na levé straně žádného přepisovacího pravidla, je definována identita $b \to b$

L-systém na rozdíl od deterministické bezkontextové gramatiky nerozlišuje terminály a neterminály. Definuje pro neterminály identitu jako výchozí přepisovací pravidlo.

L-systémy se vytváří v iteracích. Iteracím se někdy také říká generace. Iterace jsou definovány rekurzivními pravidly, přičemž nultá iterace obsahuje samotný axiom, každá další iterace vzniká aplikací přepisovacích pravidel na výsledek předchozí iterace.

Typickým příkladem L-systému je tzv. Kochova vločka. Pro vygenerování části Kochovy vločky slouží následující gramatika:

- abeceda: F + −
- axiom: F
- přepisovací pravidla: F → F+F−F+F

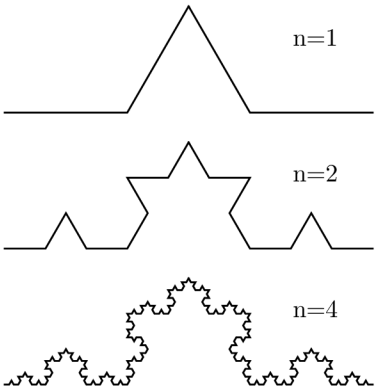
Význam symbolů abecedy: 'F' znamená vytvoření úsečky v daném směru, '+' otočí směr doleva o daný úhel, '−' naopak otočí směr doprava.

S úhlem otočení 60° pak pomocí želví grafiky vznikne obrázek 2.2.

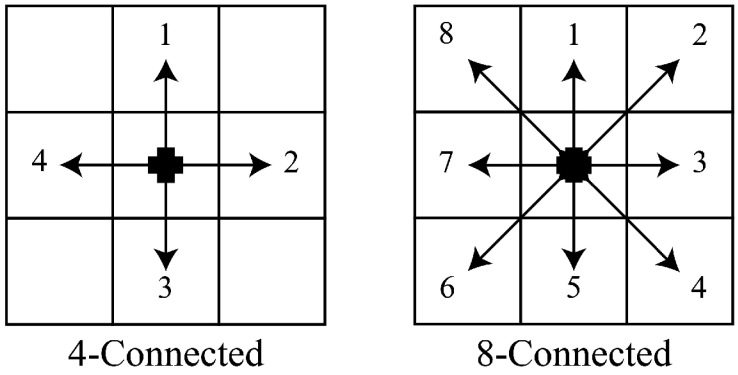
2.2 Typy okolí pixelu

Jelikož tato práce implementuje mapu jako dvoudimenzionální mřížku, tak je nutné zvolit si typ okolí jednotlivých bodů této mřížky.

Základními typy okolí pixelu ve dvou dimenzích jsou 4-okolí a 8-okolí [9] [8] (viz. obrázek 2.3). Existuje i 6-okolí, ale to není tak významné. Jestliže je aktuální pixel na pozici $[x, y]$, pak 4-okolí se skládá z následujících vedlejších pixelů: $[x+1, y]$, $[x-1, y]$, $[x, y+1]$, $[x, y-1]$. 8-okolí obsahuje stejné pixely jako 4-okolí, ale obsahuje ještě 4 další, těmi jsou: $[x+1, y+1]$, $[x-1, y-1]$, $[x-1, y+1]$, $[x+1, y-1]$.



Obrázek 2.2: Na obrázku lze vidět jednotlivé iterace tvorby Kochovy vločky. Ta vzniká pomocí želví grafiky, kdy se vykreslují úsečky o dané velikosti v daném směru. Obrázek je převzat z internetu[2].



Obrázek 2.3: Na obrázku je vidět čtyřokolí a osmiokolí pixelu. Šipky ukazují směry možných sousedních pixelů. Obrázek je převzatý z internetu[4].

2.3 Voxel

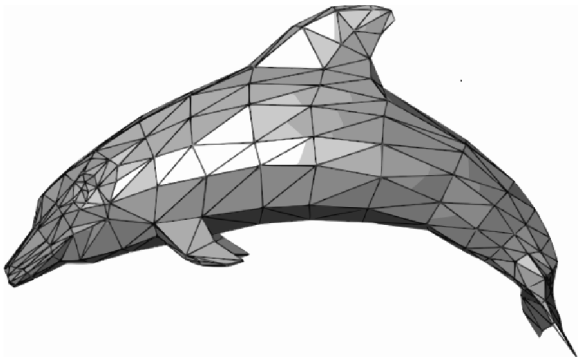
Reprezentace mapy v práci představuje voxelová mřížka. Voxel [2] reprezentuje hodnotu v trojrozměrné pravidelné mřížce. Voxely mají mezi sebou pravidelné rozestupy. Typicky však neobsahují jejich pozici, ta je vypočítána z relativní pozice vůči ostatním voxelům. Voxely ve hrách jsou často využívány pro reprezentaci terénu. Příkladem těchto her jsou: Minecraft, Roblox, Cube World a mnoho dalších.

2.4 Polygonální Síť

Polygonální síť [5] [4] (angl. Polygon mesh) nebo zkráceně mesh je geometrická reprezentace objektů v počítačové grafice. Základem je mnohostěnná plocha spojující několik vrcholů. Základními prvky jsou tedy plochy, vrcholy definující tyto plochy a hrany spojující tyto vrcholy. Plochy jsou typicky složeny z trojúhelníků či čtyřúhelníků z důvodu usnadnění jejich vykreslování, ale mohou se skládat i ze složitějších mnohoúhelníků. Směr, ze kterého

[2] https://en.wikipedia.org/wiki/File:Koch_curve_construction.svg

[4] https://deepai.org/publication/scan-flood-fill-scaff-an-efficient-automatic-precise-region-filling-algorithm-for-complicated-regions



Obrázek 2.4: Mesh složitějšího objektu. Objekt ve tvaru delfína se skládá z trojúhelníků, které dohromady tvoří celý jeho povrch. Čím více je povrch detailní, tím více je v oblasti trojúhelníků, aby tyto detaily bylo možné zobrazit (obrázek z internetu[6].)

bude viditelná, je definován pořadím spojení vrcholů. Je tedy nutné si stanovit, zda je potřeba vrcholy propojit po směru hodinových ručiček nebo naopak proti směru.

2.5 Pochodující čtverce

Pochodující čtverce [3], anglicky Marching squares, je algoritmus používaný v počítačové grafice. Tento algoritmus převádí dvoudimenzionální voxelovou mřížku na mesh. Jednotlivé voxely této mřížky jsou buď prázdné, nebo vyplněné. Algoritmus Pochodující čtverce zjistí, které voxely jsou vyplněné a na základě toho poté vytvoří trojúhelníky mezi nimi. Těchto variant, jak trojúhelníky vytvořit, je celkově 16, jak lze vidět na obrázku 2.5. Pro určení, která varianta se vygeneruje, je použita bitová reprezentace vrcholů. Každý vrchol je tedy jedním bitem bitové masky a může nabývat hodnot 1 nebo 0. Hodnota 1 značí vyplněný vrchol, naopak hodnota 0 značí prázdný vrchol (viz. obrázek 2.6).

2.6 Herní engine

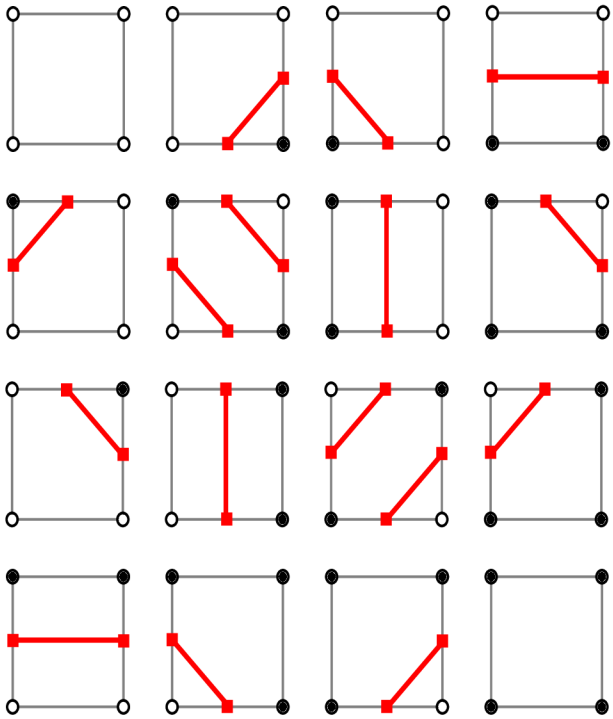
Herní engine [13] je vývojové prostředí poskytující obecné funkce pro tvorbu videoher. Jeho účelem je usnadnění a zefektivnění vývoje her a s tím související snížení nákladů na vývoj. Typicky engine poskytuje renderování 2D a 3D grafiky, simulaci fyzikálních zákonů, detekci kolizí objektů, zvuk, animace, apod. Účelem enginu je pomoci vývojářům se zaměřit přímo na vývoj jejich hry a nemuseli se tedy zabývat prvky jako jsou např. detekce kolizí, navigací či vytváření animací.

2.7 Unity

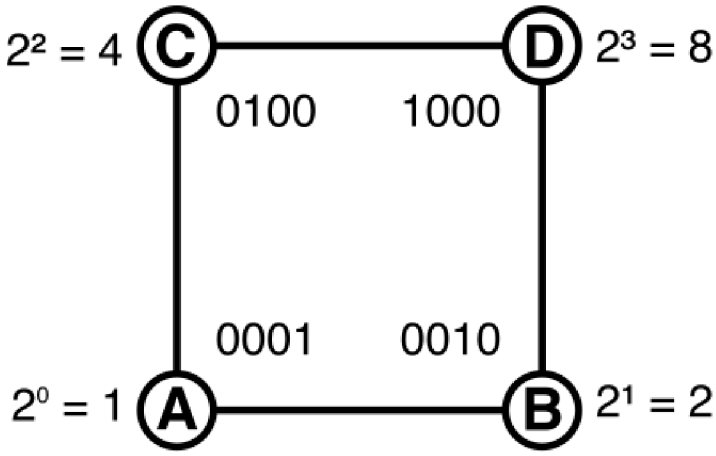
Herní engine použitý v této práce je Unity. Unity je multiplatformní herní engine, který podporuje platformy Windows, macOS i Linux. Dále pak je možné vyvíjet i hry pro konzole, mobilní zařízení i virtuální a rozšířenou realitu (VR).

Tento herní engine byl zvolen pro práci kvůli jeho popularitě. Momentálně patří mezi nejvíce rozšířené a oblíbené vývojové prostředí. To dokazují statistiky [14] sdělující, že více

[6] https://www.wikiwand.com/en/Polygon_mesh



Obrázek 2.5: Na obrázku je vidět způsob vytváření meshe při různých variantách využitých vrcholů. Například v pravém horním rohu je vidět výsledek generování, když jsou využity jen dolní dva vrcholy. Obrázek je převzat z internetu[8].



Obrázek 2.6: Na obrázku jsou vidět 4 vrcholy čtverce, který tvoří část voxelové mřížky. Těmto vrcholům jsou přiděleny pozice v bitové masce. Každý vrchol pak může nabýt hodnot 1 a 0. Hodnoty závisí na tom, zda je vrchol vyplněn či nikoliv. Výsledné číslo, které tvoří bitová maska, pak určuje způsob, jakým je mesh vytvořen. Obrázek je převzat z internetu[10].

[8] https://weekly-geekly.github.io/articles/358658/index.html
[10] https://catlikecoding.com/unity/tutorials/marching-squares/

než polovina nových mobilních her, 60 procent všeho obsahu pro virtuální realitu a alternativní realitu a polovina her napříč platformami je vytvořena v Unity.

2.7.1 Nástroje editoru

Při práci s pluginem [11] a prohlížení si vygenerované mapy je zapotřebí použití nástrojů dostupných v editoru Unity. Automaticky dostupné nástroje [15] jsou Hand Tool, Move Tool, Rotate Tool, Scale Tool, Rect Tool. Hand Tool slouží pro pohyb pohledu ve scéně. Move Tool pohybuje objekty ve směru os x, y, z. Rotate Tool umožňuje s objekty rotovat kolem daných os. Scale Tool mění měřítko objektu. Rect tool slouží pro manipulaci s obdélníkovými tvary, které může přemisťovat, měnit jejich měřítko a rotovat s nimi.

Kromě těchto předdefinovaných nástrojů si lze však vytvořit i vlastní nástroje, což je v pluginu využito pro editaci Perlinova šumu a generování dodatečných chodeb. K účelu vytvoření vlastního nástroje slouží předdefinovaná třída EditorTool, která má několik daných metod a vlastností pro usnadnění práce při vytváření vlastních nástrojů. Všechny tyto nástroje lze zvolit kliknutím na příslušné tlačítko na liště editoru Unity (viz. obrázek 2.7)

Obrázek 2.7: Ukázka panelu nástrojů v Unity editoru, na kterém jsou všechny nástroje poskytnuté editorem Unity. Tlačítko napravo na liště navíc umožňuje zvolit některý z vlastních nástrojů, které plugin implementuje.

2.7.2 Konzole Editoru

Plugin využívá konzole editoru Unity pokud je zapotřebí sdělit vývojáři zprávu o chybě či nějaké upozornění. Okno této konzole [15] slouží pro zobrazení těchto chyb, upozornění nebo jiných zpráv vytvořených v Unity. I samotný programátor může pro účely ladění vypisovat zprávy v tomto okně. Umístění okna bývá typicky v dolní části editoru, pod náhledem na scénu, případně jej lze nalézt v menu Window.

2.7.3 Inspektor

Inspektor [15] je okno v editoru, které zobrazuje momentálně zvolený GameObject (vysvětlení pojmu GameObject viz. kapitola 2.7.5). Konkrétně zobrazuje komponenty připnuté na daný GameObject a jeho vlastnosti (viz. obrázek 2.8). Tudíž je tu možné vidět skripty, pozici, rotaci nebo různé komponenty, např. Mesh Renderer pro vykreslení meshe. V tomto okně je možné tyto komponenty, skripty a vlastnosti přímo modifikovat. Tato možnost je využita pro nastavení parametrů generování v pluginu.

V případě skriptů je možné upravovat hodnoty proměnných, bez editace samotného skriptu. Tohoto lze využít i za běhu programu a zjistit tak ideální hodnoty proměnných při vývoji. Dále toto okno poskytuje možnost připnout na GameObject další komponenty a skripty.

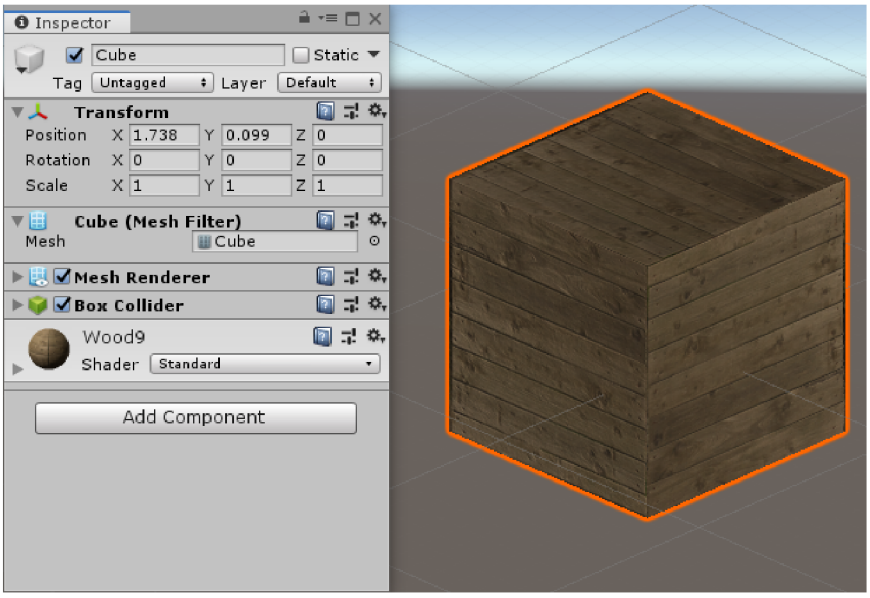
[11] Plugin, jinak také zásuvný modul, je typ programu, který nedokáže fungovat sám o sobě.

2.7.4 Komponenty

Komponenty [15] jsou důležitou částí herních objektů. Pomocí komponentů lze upravovat chování a vlastnosti daného objektu. Každý objekt může mít více komponentů a lze je i vhodně kombinovat. Příkladem této kombinace v Unity je například Rigidbody a Collider. Rigidbody přidá objektu chování simulující fyzikální zákony, takže například podléhá gravitaci. Collider zase umožňuje kolidovat a interagovat s jinými komponenty typu Collider, takže objekt při nárazu do jiného objektu jím neprojde, ale koliduje s ním. Každý komponent lze i upravovat. Po připnutí k nějakému objektu lze v inspektoru upravit jeho vlastnosti a hodnoty parametrů. Navíc komponenty mohou i uchovávat reference na jakékoliv jiné komponenty. Úplně nejdůležitější komponentem, který musí mít úplně každý herní objekt, je komponent Transform. Ten uchovává pozici, měřítko a rotaci objektu. Bez tohoto komponentu by tudíž objekt neměl pozici.

2.7.5 GameObject

Tato struktura zastupující herní objekty je objektem v Unity. Tyto objekty jsou kontejnery uchovávající vlastnosti a chování [15]. K herním objektům lze připnout skripty definující jeho vlastnosti a chování (viz. obrázek 2.8). Tyto skripty mohou být součástí Unity nebo si je může uživatel sám vytvořit a chování definovat. Prázdný herní objekt si uchovává jen svou pozici, rotaci a měřítko. Dále lze ještě využít komponenty zabudované v Unity, které se starají například o detekci kolizí. Kombinací těchto prvků poté vznikají jedinečné herní objekty s unikátními vlastnosti a chováním.



Obrázek 2.8: Na obrázku je vidět objekt typu kostka. Tato kostka má komponentu Transfrom stejně jako všechny ostatní herní objekty v Unity. Dále je k ní připnutý Mesh Filter a Mesh Renderer zajišťující vše kolem zobrazení meshe kostky. Dále kostka obsahuje Box Collider, který způsobuje detekci kolizí s ostatními objekty obsahující Collider. Jako poslední je v inspektoru kostky vidět materiál kostky.

2.7.6 Scéna

Scéna [15] v Unity obsahuje kompletní prostředí hry, např. objekty, osvětlení, kamery, apod. Scénu si lze představit jako jednu úroveň hry. Což znamená, že hra může mít více scén a každá tato scéna může obsahovat a typicky obsahuje jiné objekty, dekorace a prostředí. Tyto scény lze tedy v Unity vytvářet, přepínat mezi nimi a případně i mazat.

2.7.7 Asset Store

Unity Asset Store [15] je knihovna výtvorů vytvořených členy komunity a poskytnutých ostatním vývojářům. Tyto výtvory mohou být zdarma, ale i za poplatek. Mezi tyto výtvory patří široká škála věcí, např. textury, materiály, animace, objekty, části projektů nebo i příklady celých projektů. Přístup k této knihovně je na webu nebo přímo v Unity editoru. Objekty použité pro generování dekorace mapy jsou získány právě z této knihovny.

2.7.8 Herní Mód

Herní režim [15] je jedním z hlavních funkcionalit editoru Unity. Tento mód umožňuje vývojáři spustit projekt přímo z editoru Unity. Pro vstup do herního módu stačí zmáčknout na tlačítko na horní liště nad oknem zobrazující scénu. Herní mód simuluje to, jak by se projekt choval, kdyby byl již hotový a samostatně spustitelný. Změny komponentů a skriptů během herního režimu se resetují po jeho ukončení. Toto je užitečné například pro ladící účely a v pluginu je to využito pro prohlížení mapy po její finalizaci.

2.7.9 ScriptableObject

ScriptableObject [15] je v Unity datový kontejner, který umožňuje ukládání velkého objemu dat. Je nezávislý vzhledem k instancím tříd. Hlavní využití tohoto kontejneru je redukce paměti spotřebované projektem. Toho je docíleno tak, že se nemusí ukládat duplicitní kopie hodnot. Jednotlivé instance typu ScriptableObject se po vytvoření uloží ve zvolené složce projektu jako asset (asset je položka, kterou lze v projektu použít). Objekty sloužící pro dekoraci mapy v pluginu jsou zastoupeny právě tímto kontejnerem

Kapitola 3

Návrh řešení

Tato kapitola popisuje návrh řešení pluginu pro herní engine Unity. Tento plugin poskytuje generátor, který procedurálně generuje mapu. Mapa je specifikována na podzemní kobky, které se často vyskytují ve hrách. Typickou hrou využívající úrovně v podzemních kobkách je Diablo 1. Diablo úrovně také generuje procedurálně, tudíž je pro návrh práce částečně vzorem a inspirací.

3.1 Návrh mapy

Pro generování mapy je nutné nějak navrhnout způsob přístupu k ní. Tento plugin pracuje s prostorem, který je vymezen pro generování mapy, jako s voxelovou mřížkou (co je to voxel je popsáno v kapitole 2.3). Každý voxel má svoji relativní pozici v rámci mřížky a uchovává o sobě některé další informace, např. zda je na dané pozici vygenerovaná dekorace, anebo je pozice prázdná.

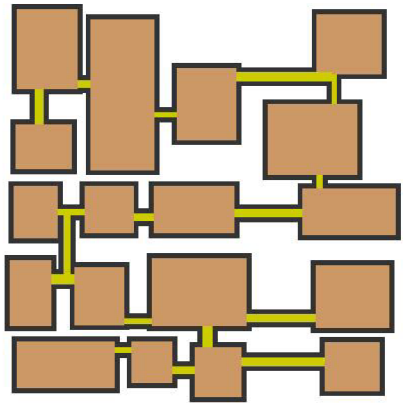
3.2 Půdorys mapy

Při procedurálním generování je jako první velice viditelný samotný půdorys mapy, tudíž je nejdůležitějším aspektem, na který je v práci kladen největší důraz. Pro generování půdorysu mapy byly prostudovány následující algoritmy: Tunneling Algorithm, BSP Tree, Random Walk, Cellular Automata, Broguelike, City Buildings, Maze With Rooms a Messy BSP Tree. Z uvedených algoritmů byl na základě vzhledu výsledku generování (viz. obrázek 3.1) zvolen algoritmus Binary Space Partition Tree Generation.

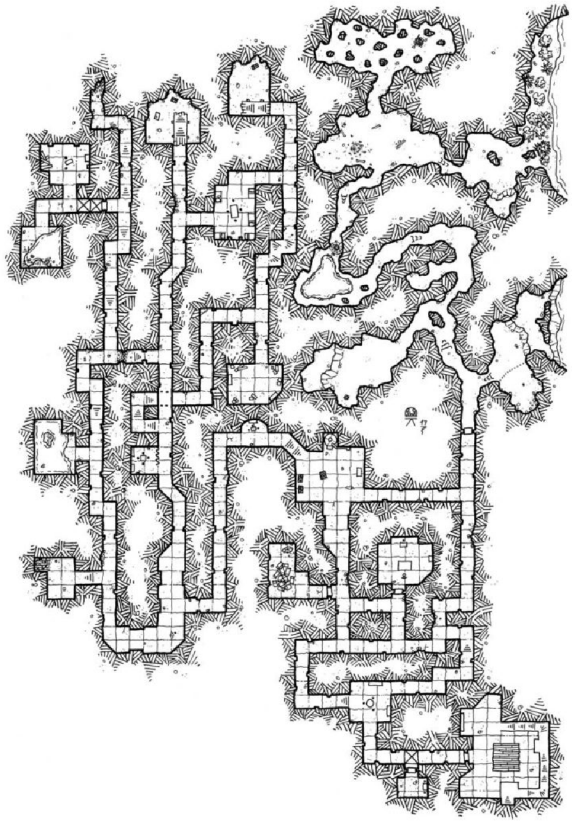
3.2.1 Generování místností

Pro generování je nutné si zvolit objekt nesoucí jednotlivé generátory v hierarchii scény. Generátory lze využít za pomoci elementů uživatelského rozhraní v inspektoru. V něm je možné nastavit parametry generování, jako jsou např. výška a šířka mapy, výška zdí, minimální a maximální velikost vygenerovaných místností, apod. Na závěr se mapa vygeneruje po stisknutí tlačítka určeného pro generování.

Pro generátor místností je využit algoritmus Binary Space Partition Tree Generation. Ten je vhodný pro účel generování jednoduchých podzemních kobek. Pro nějaké 2D úrovňové hry je dostatečný, cílem této práce je však poněkud více realistický půdorys (viz. obrázek 3.2). Generátor je tedy upraven. Generování chodeb je z tohoto generátoru odstraněno a je vytvořen speciální generátor zaměřující se pouze na chodby. Chodby z generátoru



Obrázek 3.1: Ukázka výsledku generování podzemních kobek pomocí algoritmu Binary Space Partition Tree.[2].



Obrázek 3.2: Ukázka složitějšího půdorysu. Na obrázku je vidět, že místnosti nejsou v poměru k chodbám tak časté. Naopak chodby tvoří větší část půdorysu a navíc jsou různě široké, mění směr a občas jsou slepé. Tento obrázek je převzat z internetu[4].

jsou odstraněny, jelikož spojují místnosti příliš jednoduchým způsobem – spojují nejbližší místnosti, neexistují slepé chodby a vůbec se nevětví.

[2] https://sites.google.com/a/umn.edu/grammatical-item-generation-language/demos/demo-4---bsp-dungeon

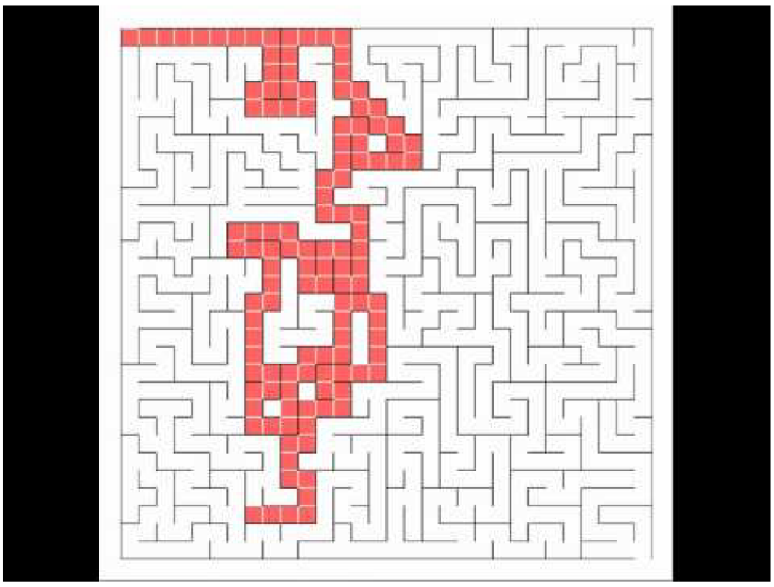
[4] https://cz.pinterest.com/pin/344736546469282245

3.2.2 Generátor chodeb

Pro generování chodeb byl navrhnut nový generátor, který generuje pouze chodby. Základem tohoto generátoru je princip jakéhosi semínka letícího ve větru. To se navíc v náhodných bodech dělí na další semínka. Trasa letu tohoto semínka je výsledná chodba a její šířka závisí na délce letu.

Stejně jako u chodeb je nutno nastavit parametry generátoru. Parametry jsou nastaveny na výchozí hodnoty, ale ty nemusí být ideální ve všech případech. Při nastavování těchto parametrů pak lze určit například velikost kroku při generování.

Generátor je realizován algoritmem Growin Tree, který se chováním velice blíží principu semínka. Tento algoritmus slouží pro procedurální generování bludiště (viz. obrázek 3.3). Je velmi podobný algoritmům Recursive Backtracker a Prim's Algorithm. Growing Tree zjednodušeně funguje tak, že se na začátku typicky náhodně zvolí buňka (pozice na mapě) a přidá se na seznam zvolených buněk. Poté se zvolí buňka z tohoto seznamu a ta se podívá do jejího okolí a zvolí si z něj další buňku, která ještě nebyla navštívena. Okolí je zjednodušené na čtyřokolí(co je to čtyřokolí viz. kapitola 2.2). Vybraná buňka se také přidá do seznamu. Pokud buňka nemá v okolí žádného nenavštíveného souseda, pak je ze seznamu odebrána. Tento proces se opakuje, dokud není seznam prázdný.



Obrázek 3.3: Na obrázku je příklad vygenerovaného bludiště pomocí algoritmu Growing Tree. Zvýrazněná trasa je nedokončená správná cesta z levého horního rohu do pravého dolního. Obrázek je převzat z internetu[6].

Tento algoritmus tvoří bludiště, proto je modifikovaný, aby tvořil chodby méně cyklické a více slepé. Tohoto efektu je docíleno volbou vždy poslední buňky ze seznamu zvolených buněk. Tím je proces tvorby chodby méně náhodný, jelikož se generuje stále jen aktuální chodba a uchovávají se místa jejího budoucího větvení. V těchto místech vznikají nové větve chodeb až po ukončení generování té aktuální.

Další modifikací je omezení větvení chodeb. Původní algoritmus končí až při zaplnění celé mapy. Tento efekt není pro účel práce žádoucí. Proto je aplikovaná pravděpodobnost na jejímž základě generátor rozhoduje, zda se aktuální buňka rozvětví. Tato pravděpodobnost

[6] https://www.youtube.com/watch?v=anZlAmtaV1s

se aplikuje pro všechny sousední buňky, čímž je určen směr nových chodeb a je omezeno jejich množství.

Poslední změnou je změna délky kroku. Krokem v tomto kontextu je vzdálenost sousední buňky od aktuálně zvolené. Takže například při kroku o velikosti 3 se buňka podívá na své sousedy ve vzdálenosti 3, nikoliv přímo vedle ní. Efekt této modifikace je takový, že se směr chodeb nemění tak často a půdorys chodeb tak vypadá přirozeněji.

3.2.3 Generátor Perlinova šumu

Pro generování chodeb je využit Perlinův šum. Tento šum je vygenerován dalším generátorem, kterému jsou předány parametry mapy, konkrétně výška a šířka. Generátor Perlinova šumu pracuje navíc s parametry, které se nastavují pomocí posuvníků v inspektoru. Změna těchto posuvníků způsobuje změnu vypočítaných hodnot Perlinova šumu a to se ihned projeví na objektu v pozadí mapy, sloužící pro zobrazení tohoto šumu ve scéně editoru. Uživatel tak vidí okamžitě vliv změny parametru a může je tak snáze nastavit na ideální hodnoty.

3.2.4 Spojení generátoru chodeb a generátoru místností

Pro spojení generátoru se nabízí několik možností. Základní myšlenkou spojení je vygenerovat chodby i místnosti na stejné pozici. Obě mapy mají stejnou velikost. Z každé vygenerované mapy jsou převzaty nějaké části, které se spojí a zkompletují. První možností je využít struktury generátoru chodeb. Tudíž začít na počátečním bodu generátoru chodeb a postupně procházet chodby tak, jak byly vygenerovány. To znamená postupně procházet chodby a uchovávat si počátky nových větví chodeb, které ještě nebyly navštíveny. Při tomto průchodu se zároveň kontroluje kolize s mapou místností. V případě kolize se rozhodne, ze které mapy by se vygenerovaný výsledek uchová, např. pomocí hodnot Perlinova šumu.

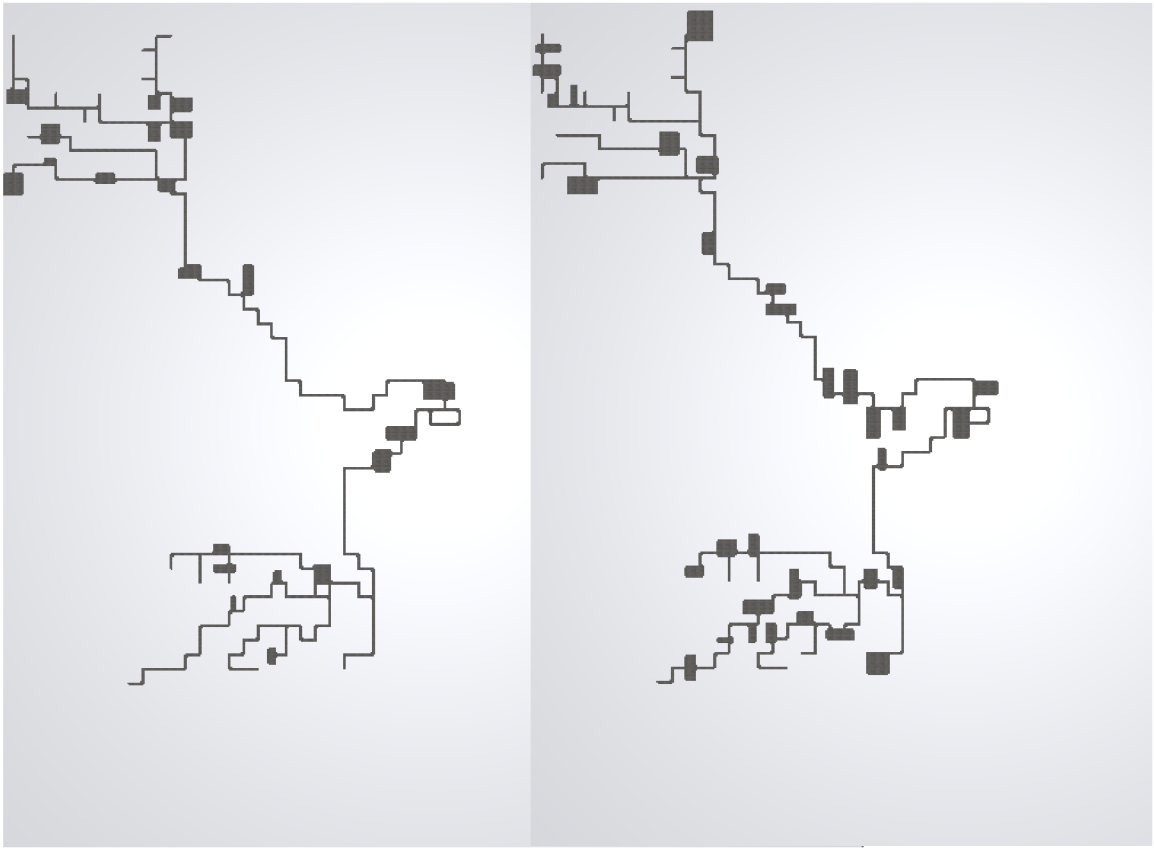
Další možností je procházet vymezenou oblast postupně z levého horního rohu až po pravý dolní roh. Toto je možné jelikož oba generátory pracují s mapou jako s mřížkou. Při tomto průchodu se kopírují hodnoty obou vygenerovaných map, tzn. jak mapy chodeb, tak mapy místností, do dvojrozměrného pole reprezentující mřížku mapy. Při tomto kopírování je kontrolováno jakým způsobem daná hodnota dlaždice vznikla. U prázdné dlaždice se zkopíruje jen hodnota. Pokud je však na dané pozici místnost, tak se určí, zda je tato dlaždice napojena na chodbu. Napojení místnosti znamená, že v okolí dlaždice místnosti se nachází dlaždice chodby. V případě napojení se celá místnost zkopíruje do mapy. Chodby se kopírují do mapy vždy.

V této práci je zvolena druhá varianta přístupu, jelikož je snazší a přímočařejší. Chodby zůstávají stejné, kromě částí, kde jsou překryty místnostmi. Zároveň jsou izolované místnosti elegantně vyloučeny a jejich počet je více náhodný.

Finální generátor volá generátory chodeb a místností a spojí jejich vygenerované výsledky dohromady. Toto spojení nemusí být vždy pro uživatele vyhovující, tudíž je umožněno přegenerovat zvlášť chodby a místnosti. Mapu a její části lze generovat opakovaně k dosažení požadovaného výsledku (viz. obrázek 3.4).

Po vygenerování chodeb s místnostmi jsou zpřístupněna tlačítka pro změnu šířek chodeb a změnu měřítka mapy. Změna šířek chodeb tu je za účele dosažení o něco více realističtějšího vzhledu mapy. Změna měřítka slouží pro optimalizaci rychlosti generování. Jelikož generování mapy o rozloze 500 x 500 a větší může zabrat půl sekundy a více v závislosti na stroji, na kterém Unity běží. Uživatel tak může vygenerovat půdorys mapy na zmenšené

ploše, např. 200 x 200 a po dosažení vhodného výsledku zvětšit mapu bez změny vzhledu místností či chodeb. Kolikrát se mapa zvětší je nutno nastavit v parametru generátoru.



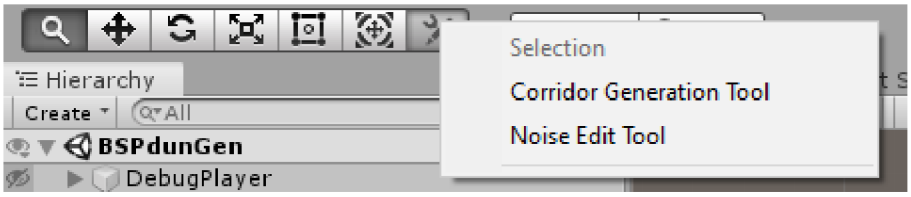
Obrázek 3.4: Na obrázku je vidět dvě různé verze výsledku generování. V obou verzích jsou stejné chodby, ale místnosti jsou měněny. Toto přegenerování místností je možné provádět opakovaně.

3.2.5 Nástroje pro větší míru přizpůsobení vzhledu mapy

Pro zvýšení míry možností zásahu do vygenerované mapy je přidán způsob editace Perlinova šumu a možnost vygenerovat nové chodby k již existujícím chodbám na zvoleném místě a ve zvoleném směru. K tomuto účelu se zcela hodí možnost poskytovaná enginem Unity – vytvoření vlastního nástroje, který lze zvolit na horní liště Unity editoru a poté ho následně pomocí myši použít pro předdefinovaný účel.

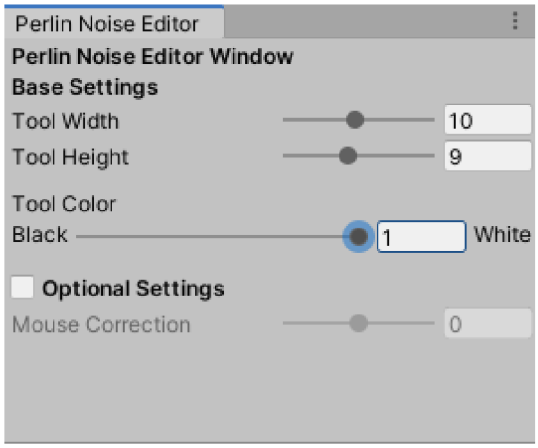
Pro vygenerování nových chodeb na dané pozici a v daném směru je tudíž možné pomocí volby příslušného nástroje (viz. obrázek 3.5). Chodba pak povede směrem tahu myši a její počátek bude na počáteční pozici myši při kliknutí.

Pro editování Perlinova šumu je zapotřebí jej nejdříve zobrazit, aby bylo viditelné, kde je oblast pro chodby nedosažitelná nebo naopak dosažitelná. Proto je v pozadí vyhraněné plochy pro mapu objekt s texturou zobrazující hodnoty Perlinova šumu. Následně pro editaci tohoto šumu je třeba zvolit vytvořený editační nástroj. V moment aktivace nástroje se uživateli zobrazí okno s parametry nástroje (viz. obrázek 3.6) a pohled v editoru se zaměří na plochu zobrazující Perlinův šum. Parametry nástroje lze ve vzniklém okně měnit dle



Obrázek 3.5: Na obrázku je vidět horní lišta editoru Unity, na které lze zvolit předdefinované nástroje. Jak lze vidět je tu již několik nástrojů implementovaných v samotném Unity a poté úplně vpravo tu je tlačítko pro vlastní nástroje, kde jsou vytvořené nástroje pro editování Perlinova šumu a generovaní dodatečných chodeb.

potřeby. Nástroj slouží jako štětec, při stisknutém levém tlačítku myši se mění hodnoty Perlinova šumu na pozici odpovídající pozici myši. Toto poskytuje možnost manuálního omezení prostoru, kam se chodby šíří. Zároveň to částečně umožňuje si určit tvar půdorysu chodeb.



Obrázek 3.6: Na obrázku je vidět okno potřebné pro nástroj editující Perlinův šum. Toto okno umožňuje nastavovat parametry nástroje. Těmito parametry se tedy nastavuje rozměr nástroje a jeho barva pro editování.

3.3 Vizualizace vygenerované mapy

Po vygenerování místností nebo chodeb a poté i finální sloučené mapy je třeba tento výsledek i zobrazit. K tomuto účelu je generována vlastní polygonová síť. Tato síť je generovaná pomocí algoritmu Pochodující čtverce(popis algoritmu viz. kapitola 2.5), který je vhodný pro generování 2D ploch.

3.4 Úprava Inspektoru

Pro generování mapy i následné výplně je využito možností inspektoru v editoru Unity. Skripty generátorů jsou připnuty k jednomu hernímu objektu. Tyto skripty lze vidět v inspektoru a také jejich hodnoty upravovat.

Pro změnu hodnot parametrů je uživatelské rozhraní upraveno, aby bylo přehlednější a zároveň k větší kontrole hodnot uživatelských vstupů. Parametr určující směr chodeb je proto možné nastavit pouze na předem definované hodnoty v rozbalovacím seznamu. Parametry, které mají pevně dané rozmezí, jsou zastoupeny posuvníky se zadanými maximy a minimy. Tímto je docíleno větší přívětivosti použití pluginu.

Pro samotné generování pak slouží tlačítka, která použijí nastavené parametry a pokusí se vygenerovat příslušný výsledek. V případě úspěchu se na scéně v editoru ukáže vygenerovaná polygonální síť mapy. V případě neúspěchu se vypíše chybové hlášení či upozornění do konzole editoru. Tlačítka jsou navíc podmíněna, aby nebylo možné měnit šířky neexistujících chodeb nebo měnit měřítko nevygenerované mapy.

Pro finální zpřehlednění parametrů jednotlivých generátorů půdorysu jsou všechny parametry přesunuty do jednoho skriptu, který volá dané generátory. Tím se parametry zobrazí v jedné sekci inspektoru. Dále jsou parametry rozděleny do podsekcí s nadpisy, aby bylo jasné, které aspekty generování ovlivňují (viz. obrázek 3.7).

Výjimkou jsou parametry Perlinova šumu, které byly ponechány v sekci skriptu generující tento šum. U toho parametru přesunuty nejsou, jelikož v jsou nastaveny na začátku a pak se již s nimi většinou nepracuje a sekce parametrů generátoru by byla již velice přeplněná až nepřehledná. Generování vnitřní výplně podobně jako Perlinův šum také zůstává v sekci skriptu implementující toto generování, jelikož po dokončení půdorysu již není třeba parametry generátoru půdorysu měnit, takže všechny tyto parametry by byly rušivé při další práci.

3.5 Generování vnitřní výplně mapy

Pro vygenerování vnitřní výplně je zvolen princip založený na L-systému. Tudíž každý bod mapy je označen jednoznakovým či víceznakovým symbolem určující jeho pozici v rámci mapy. Tyto znaky jsou poté na základě přepisovacích pravidel měněny. Na základě těchto přepsaných symbolů se na závěr vytvoří objekty na daných pozicích mapy.

3.5.1 Práce s objekty určené pro výplň

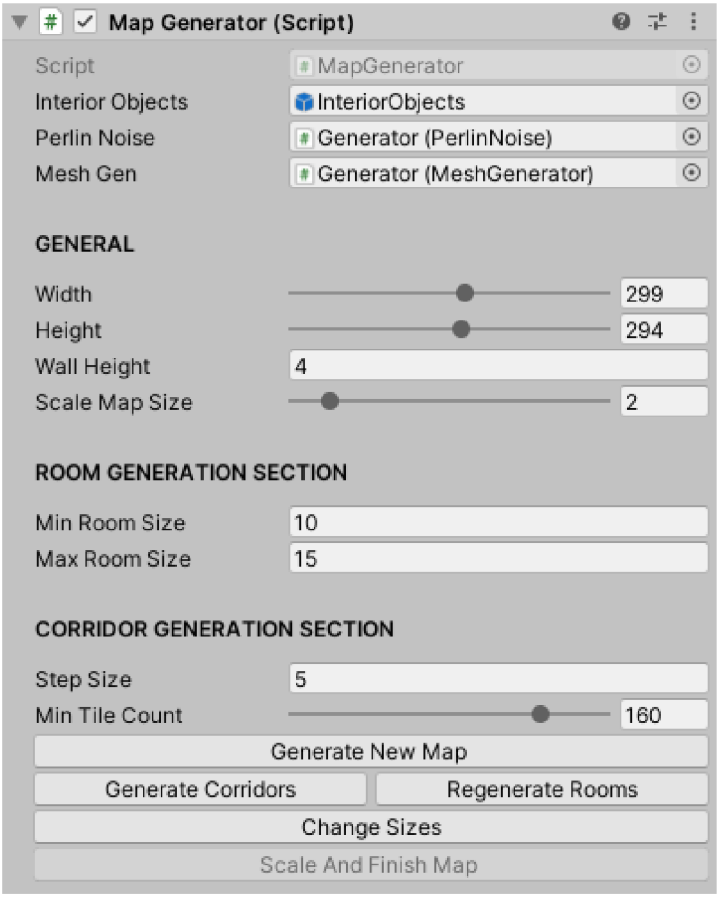
Pro práci s objekty určené pro výplň mapy je potřeba si uchovat několik dat. Mezi ně patří např. symbol, který daný objekt zastupuje nebo jeho rozměry. Pro tento účel je zapotřebí datového kontejneru. V práci je využit kontejner typu ScriptableObject, který engine Unity poskytuje. Plugin proto obsahuje datové kontejnery, např. datový kontejner s názvem furniture. V souborovém systému projektu lze poté vytvořit položku typu furniture, která zastupuje objekt vnitřní výplně, konkrétně nábytku. Například pro využití sudu jako dekorace je třeba vytvořit položku furniture a v ní nastavit referenci na daný objekt a další parametry této dekorace. Pro generování tohoto objektu je nutné přidat položku do seznamu objektů.

3.6 Prohlížení mapy pomocí kamery v herním režimu

Při generování půdorysu mapy stačí okno scény Unity editoru k tomu, aby si uživatel prohlídl vygenerovaný výsledek a usoudil tak, zda je vygenerovaný výsledek ten, který požaduje nebo nikoliv.

Během generování vnitřní výplně mapy je sice náhled na scénu v editoru dostačující, avšak možnost prohlédnout si mapu z pohledu hráče po dokončení mapy je prakticky nutná.

Pro tyto účely plugin obsahuje kameru s připnutými skripty, které umožňují v herním módu Unity pohybovat kamerou pomocí tlačítek klávesnice. Pohyb je umožněn klasicky na ose x, navíc je ale i možné měnit výšku, tudíž pohyb na ose y.



Obrázek 3.7: Na obrázku je vidět okno zobrazující parametry generátoru půdorysu. Tyto parametry jsou přesunuty do jedné sekce, navíc jsou odděleny mezerami a nadpisy, aby bylo jasnější, který generátor daný parametr ovlivňuje. Na spodní části obrázku lze vidět tlačítka pro samotné generování.

Kapitola 4

Implementace řešení

V této kapitole je podrobněji popsána implementace této práce, jejíž návrh je popsaný v kapitole 3. Pro implementaci je použit herní engine Unity, jelikož právě pro tento engine má být práce pluginem. Jako programovací jazyk je použit jazyk C#, který daný engine podporuje a je v něm nejvíce používán.

Verze herního enginu použitá při vývoji byla převážně 2019.1.14f1. Momentálně nejnovější oficiální verzí je 2019.3.13f1, na kterou jsem Unity aktualizoval v závěru implementace a otestoval, zda je vše kompatibilní. Výsledkem se ukázalo, že vše funguje v pořádku. Jako vývojové prostředí pro psaní a editování skriptů sloužil nejdříve Visual Studio Code, později však Microsoft Visual Studio 2019. K přechodu došlo kvůli lepší podpoře enginu Unity. Toto IDE umožňuje řadu funkcionalit zpříjemňující programování v Unity. Příkladem těchto funkcionalit je například automatické přejmenování skriptu pokud je přejmenována třída, která je v tomto skriptu implementovaná. Další velice důležitou funkcionalitou, kterou přináší až nové Microsoft Visual Studio 2019, je automatické překládání skriptů již při programování. Dříve se skripty překládali až po přechodu do okna editoru Unity, což velice zdržovalo.

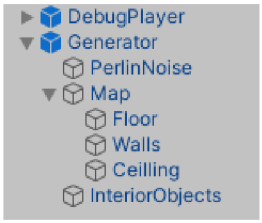
Pro refaktorování kódu bylo navíc využito rozšíření ReSharper od firmy JetBrains, které upozorňuje na špatné návyky, jakými jsou např. špatně pojmenované funkce, proměnné, špatná práce s pamětí, apod.

4.1 Nastavení generátoru v hierarchii scény

Pro generování mapy je nejdříve potřeba vytvořit potřebné herní objekty a těm přiřadit skripty definující jejich chování. Pro tento účel je vytvořena následující struktura objektů (viz. obrázek 4.1):

- Hlavním objektem je objekt s názvem *Generator*. Ten má k sobě připnuty veškeré skripty potřebné pro generování mapy. Takže generátory půdorysu, Perlinův šum, generování interiéru, apod.
- Další objekt je *PerlinNoise*. Tento objekt je vytvořen z předdefinovaného objektu Cube. Využit je pro zobrazení Perlinova šumu, tudíž je využit spíše jako 2D čtyřúhelníkové pozadí mapy při generování půdorysu.
- *Map* je prázdný objekt, který zastřešuje další dva objekty *Floor*, *Walls* a *Ceilling*.

- Floor je objekt obsahující komponenty nezbytné pro vytvoření polygonové sítě. Těmi jsou Mesh Filter a Mesh Renderer. V tomto objektu se tedy vytváří polygonová síť podlahy celé mapy.
- Walls je objekt obsahující komponenty Mesh Renderer, Mesh Filter potřebné pro generování polygonové sítě a navíc ještě komponent Mesh Collider, který umožňuje detekci kolizí s tímto objektem. V tomto objektu se tedy vytváří polygonová síť zdí celé mapy.
- Ceilling je objekt shodný s objektem Floor. Jediným rozdílem je, že tento objekt je určený pro polygonální síť stropu.
- *InteriorObjects* je prázdný objekt. Slouží pouze jako kontejner pro vygenerované objekty, které slouží jako dekorace.



Obrázek 4.1: Na obrázku je zobrazena hierarchie objektů tvořící generátor mapy v pluginu. Objekt Generator je kontejnerem pro všechny ostatní objekty, jelikož je nejdůležitějším a má k sobě připnuty skripty veškerých generátorů. Objekt PerlinNoise zobrazuje Perlinův šum. Map je prázdným objektem, který jen obsahuje další objekty Walls, Floor a Ceilling, které obsahují vygenerovanou polygonovou síť. Posledním objektem je InteriorObjects, který zastřešuje vygenerované instance dekoračních objektů.

4.2 Generování mapy

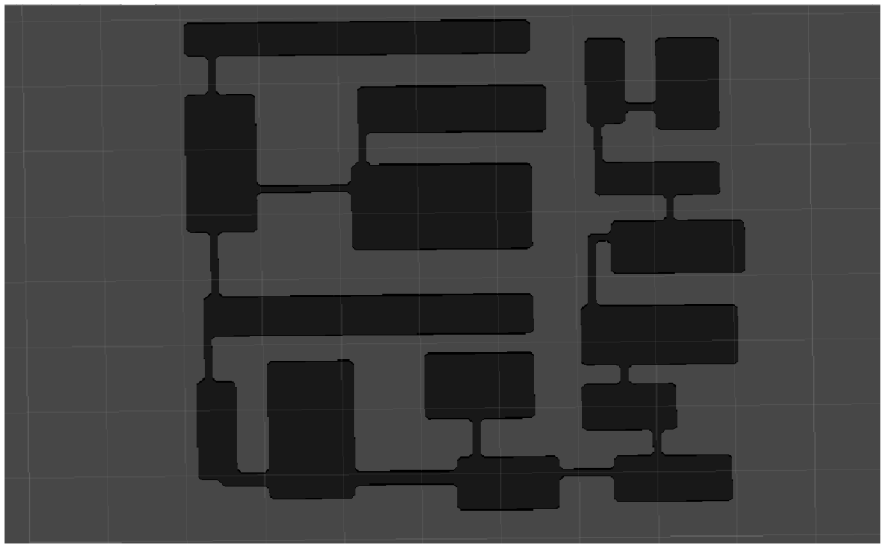
Pro generování mapy slouží základní 4 skripty. Mezi ně patří MapGenerator, RoomGenerator, CorridorGenerator a InteriorGenerator. Tyto skripty implementují stejnojmenné třídy, jejichž metody jsou použity. První tři třídy slouží pro generování půdorysu mapy a zbývající čtvrtá generuje dekoraci vnitřní části. Při generování půdorysu jsou využity další dvě důležité třídy: MeshGenerator a PerlinNoise.

Mapa je implementovaná jako voxelová mřížka (vysvětlení pojmu voxel viz. kapitola 2.3), jak již bylo popsáno v kapitole 3.1. Tohoto je docíleno pomocí 2D pole, kde indexy tohoto pole jsou využity jako koordináty x a y pozice v rámci mapy.

4.2.1 Generátor místností

Implementace místností vychází z algoritmu Binary Space Partition Tree Generation, jak již bylo zmíněno v kapitole 3.2.1. Jak je vidět na obrázku 4.2, algoritmus generuje místnosti v dané oblasti. Velikosti těchto místností jsou náhodné, ale v omezeném rozmezí.

Pro účel generování půdorysu podzemních kobek byl tento algoritmus implementován následujícím způsobem. Na začátku je určena velikost mapy a tudíž čtyřúhelníku, se kterým se pracuje. U tohoto útvaru je zkontrolováno, zda je listem aktuálního stromu. Pokud ano,



Obrázek 4.2: Ukázka vygenerovaných místností pomocí algoritmu Binary Space Partition Tree Generation. Tento algoritmus na daných rozměrech vygeneroval místnosti o velikosti v zadaném rozmezí. Ty se poté na základě vzdáleností mezi nimi pospojovali jednoduchými chodbami.

f

tak je zkontrolováno, zda je šířka útvaru větší než maximální šířka místnosti nebo zda je výška větší než maximální výška místnosti. Pokud je některá z podmínek pravdivá, tak se přejde k procesu dělení. V případě, že je výška i šířka útvaru menší než daná maxima, tak je navíc vygenerováno náhodné číslo pomocí pseudonáhodného generátoru čísel. Pokud je toto číslo větší než zadaná mez (implementace pravděpodobnosti), tak je také zavolaná dělící funkce.

V dělící funkci se nejdříve zkontroluje, zda je šířka i výška čtyřúhelníku větší než minimální rozměry místnosti. Pokud již některý rozměr dosáhl minima, není možné místnost dále dělit. V případě, že jsou rozměry větší než daná minima, zkontroluje se poměr výšky a šířky. Pokud je znatelně větší výška mapy, dojde k dělení v horizontálním směru. Pokud je však větší šířka mapy, dochází k dělení vertikálnímu. Jestliže je poměr přibližně roven jedné, tudíž se tvar místnosti blíží tvaru čtverce, pak se určí směr dělení náhodně.

Při samotném dělení je nutno ještě určit pozici, kde se útvar rozdělí na dva. Tato pozice se znovu určí náhodně v rozmezí velikosti útvaru. Při této náhodné volbě pozice dělení je však ještě zajištěno, aby oba vzniklé útvary byly větší než minima rozměrů místnosti.

Tento proces se opakuje pro každý list aktuálního stromu. Proces dělení daného útvaru končí v okamžiku, kdy je dosaženo nějakého požadovaného rozmezí velikosti místností. Jakmile jsou všechny místnosti vygenerovány, tak se zapíší do 2D pole mapy.

4.2.2 Generátor chodeb

Generátor chodeb je implementován na základě algoritmu Growing Tree, jak je již zmíněno v sekci 3.2.2.

Implementace generátoru je tedy taková, že se nejdříve nastaví pravděpodobnosti, do kterých směrů se chodby šíří. Tyto směry jsou pouze 4, jelikož generátor pracuje s čtyřokolím. Přičemž dva směry jsou vždy velice pravděpodobné, aby se chodby měli kam šířit a větvit

a ty zbylé dva směry jsou možné, ale velice nepravděpodobné. Poté se přejde ke generování samotného stromu chodeb.

Při generování chodeb se vytvoří dvoudimenzionální mapa. Tato mapa reprezentuje voxelovou mřížku. Nejdříve se všechny voxely nastaví na hodnotu 0, která značí, že je voxel prázdný. Následně se zvolí počátek generovaných chodeb, který je typicky na počátku souřadnicových os (pokud není určeno jinak). Počáteční bod se přidá do seznamu zvolených buněk a začíná generování chodeb.

Ze seznamu se vybere poslední přidaná buňka. Ta zkontroluje své okolí ve vzdálenosti zadaného kroku. Pokud již bylo okolí buňky navštíveno, tak je ze seznamu odstraněna. U dosud nenavštívených sousedních buněk je vyhodnocena hodnota vygenerovaného Perlinova šumu. Hodnota tohoto šumu společně s hodnotou pravděpodobnosti šíření v daném směru určí, zda se tímto směrem vytvoří nová část chodeb či nikoliv. Buňky, na které se chodby vygenerují, jsou přidány do seznamu, naopak ty, na které se chodby šířit nebudou jsou označeny jako navštíveny. Voxely mezi původní buňkou a zvolenou sousední jsou nastaveny na hodnotu 1, aby se tu vygenerovala polygonová síť. Po vyhodnocení všech čtyř okolních buněk se proces opakuje. Generování skončí v momentu, kdy je seznam prázdný.

Po sloučení chodeb a místností jsou ještě změněny šířky chodeb, aby se přiblížili více k reálnému vzhledu chodeb. Pro tento účel je zvolena metoda taková, že je nalezena nejdelší chodba, případně některá z těch nejdelších a ta je rozšířena. Všem ostatním chodbám se pak také změní šířka v závislosti na sousedních chodbách, aby tato změna nebyla příliš drastická (viz. obrázek 4.3). Těchto variací, jak budou místnosti široké, je typicky mnoho, tudíž plugin poskytuje možnost opakované změny těchto šířek bez nutnosti znovu vytvářet celou mapu. Vývojář je proto schopný měnit tyto šířky chodeb, dokud s nimi není spokojený.

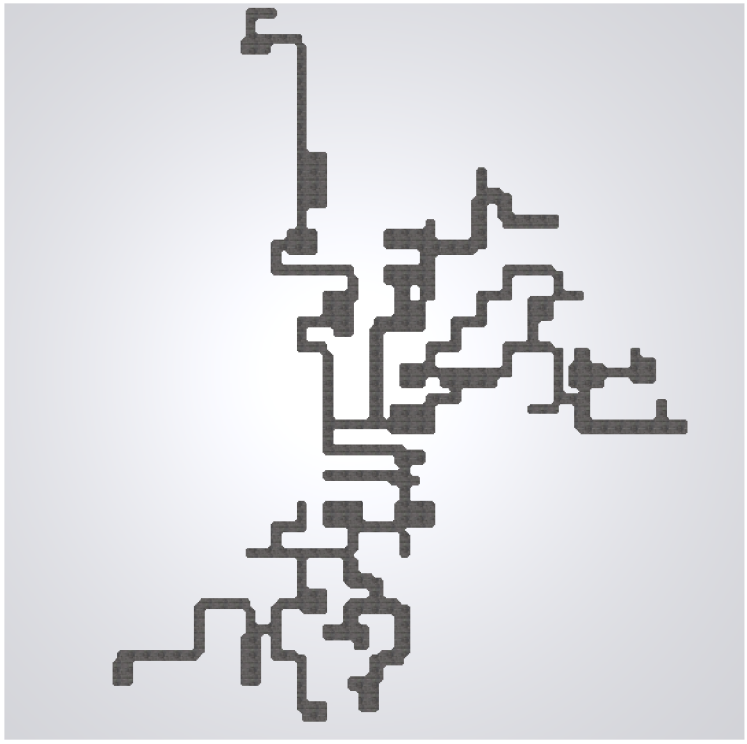
Pokud však nebylo vygenerováno minimální množství dlaždic chodeb, tak je výsledek smazán a generátor se pokouší vygenerovat chodby znovu. Tato snaha je provedena maximálně pětkrát, aby bylo zamezeno nekonečné smyčce v případě, kdy je okolí chodeb příliš nevhodné pro generování (např. v případu nepříznivých hodnot Perlinova šumu). Pokud je snaha o vygenerování půdorysu s aktuálními parametry generátoru neúspěšná, vypíše se do konzole editoru varovná zpráva o tom, proč se nevygeneroval žádný výsledek.

4.2.3 Finalizace půdorysu

Pro spojení generátoru místností a generátoru chodeb je vytvořeno ještě několik úprav. Místnosti se generují po celé ploše vymezené mapy. Chodby však nikoliv. Proto je plugin umožňuje kontrolu, kam vygenerované chodby směřují. Jelikož generátor chodeb pracuje s čtyřokolím, postačí možnost manipulace s pravděpodobnosti větvení ve čtyřech směrech.

Pro ještě více přirozeně vypadající vzhled chodeb je využito Perlinova šumu, jak je již zmíněno v kapitole 4.2.2. Ten slouží pro vymezení oblastí mapy, kam se budou chodby větvit (viz. obrázek 4.4). Toto způsobí, že některé oblasti mapy jsou pro chodby nedosažitelné, zatímco jiné jsou zase natolik vhodné, že se v nich chodby větví více.

Pro samotné spojení těchto dvou generátorů je zvolen přímočarý postup. Tím je generování dvou map stejné velikosti a na stejné pozici. Jedna mapa vzniká pomocí generátoru chodeb, druhá je naopak výsledkem generátoru místností. Tyto mapy se překryjí a kolidují. Tato kolize je detekovaná a mapy jsou sloučeny. U vygenerovaných místností je určeno, které jsou dosažitelné a ty jsou uchovány. Zbytek izolovaných místností je smazáno, jelikož nejsou použity. V implementaci je toto realizováno průchodem všech dlaždic daného rozmezí mapy a kontrola, zda je na dané dlaždici chodba či místnost. Chodby jsou zachovány všechny, jelikož slepé chodby, případně chodby, které nevedou do místnosti, se v reálných kobkách



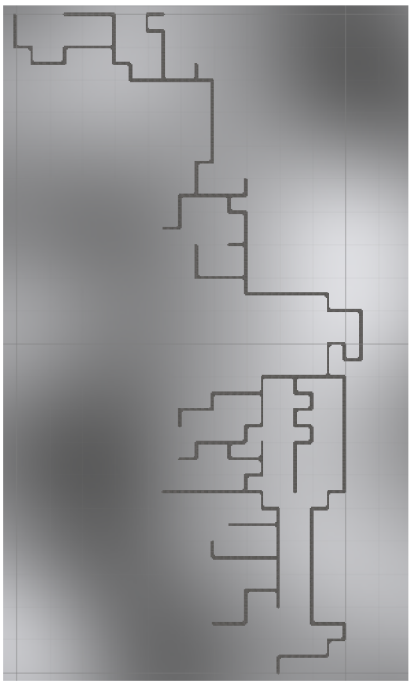
Obrázek 4.3: Na obrázku je vidět půdorys kompletní mapy a změny šířek vygenerovaných chodeb. Šířky jednotlivých chodeb jsou závislé na vedlejších chodbách, aby jejich změny nebyly velké.

vyskytují. Tímto vzniká nová mapa, která již obsahuje jen dosažitelné místnosti a chodby, které vypadají více jako z reálného světa, již nejsou jednoduché a nespojují jen nejbližší místnosti.

Na závěr, pokud je již dosaženo požadovaného výsledku, je vytvořeno tlačítko umožňující zvětšit mapu pomocí změny měřítka. Tato funkce je přidaná z důvodu urychlení práce s generátory. Jelikož generování mapy o velikosti 250x250 je rychlé a výsledek je vygenerován okamžitě v době stisku tlačítka, ale generování mapy o velikosti 500x500 a větší je již znatelně pomalejší. Může se jednat o půl sekundy až sekundu více na jedno vygenerování mapy. Což značně ruší při snaze vytvořit si požadovaný výsledek mapy, kdy je zapotřebí i několikrát mapu přegenerovat. Toto zvětšení funguje tak, že se každá dlaždice rozšíří na několik dalších kolem sebe a je uložena na novou vypočítanou pozici v mapě. Zároveň jsou však zachovány veškeré poměry velikostí stran (viz. obrázek 4.5).

4.3 Perlinův šum

Perlinův šum je implementován pomocí funkce PerlinNoise z knihovny Mathf. Šum je vygenerován v rozmezí určeném pro půdorys mapy. Vygenerované hodnoty jsou poté zobrazeny na textuře ve scéně a lze je upravovat pomocí nástroje v editoru. Touto editací lze manuálně přímo ovlivnit půdorys mapy (viz. obrázek 4.6).

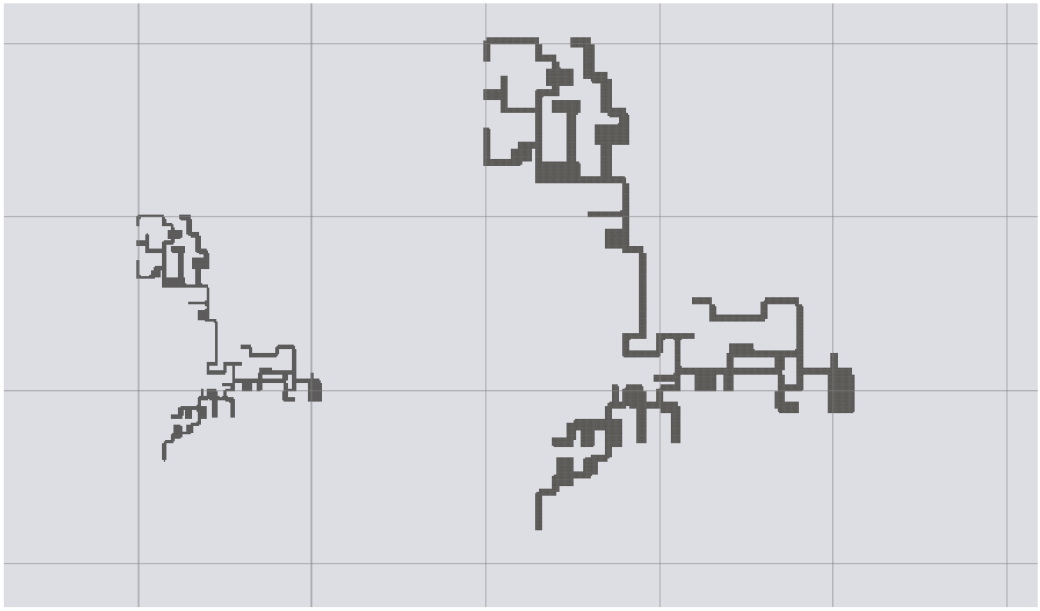


Obrázek 4.4: Na obrázku je vidět vygenerovaná síť chodeb řízená pravděpodobnostmi, kam se mají chodby větvit. Šířka chodeb je zatím všude stejná, ta se změní v další fázi generování. Dále je v pozadí těchto chodeb textura zobrazující Perlinův šum, který také ovlivňuje směr chodeb. Tmavé plochy jsou tudíž pro chodby těžko dosažitelné až nemožné. Světlé plochy jsou naopak velmi příznivé pro větvení chodeb.

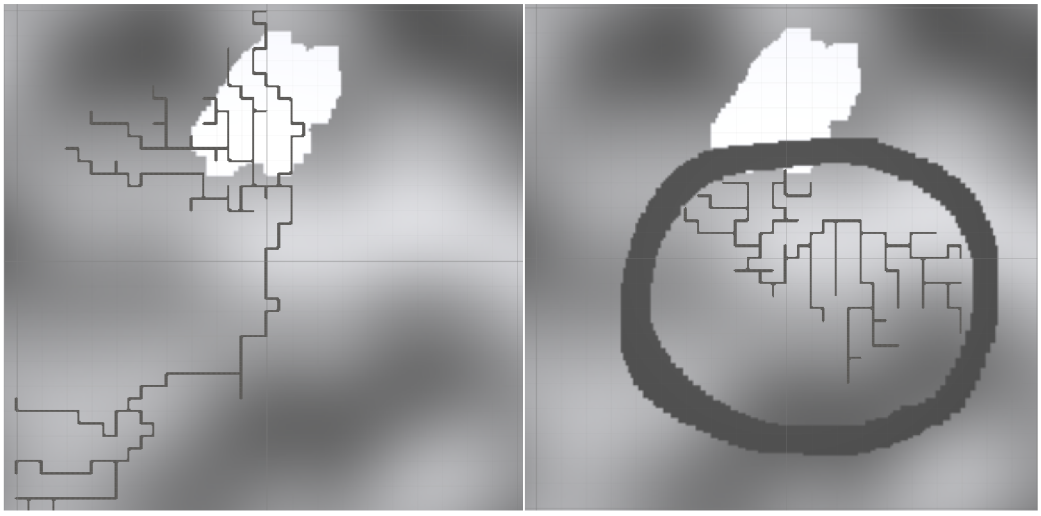
4.4 Generátor polygonální sítě

Jelikož je při generování půdorysu mapy vytvořeno dvoudimenzionální pole, které představuje mapu ve formě voxelové mřížky, tak je i generování polygonální sítě takto implementováno. Mřížka předaná generátoru se skládá ze dvou hodnot. Těmito hodnotami jsou 1 a 0, kde 1 značí mesh a 0 prázdno. Postup generátoru je tedy takový, že si vytvoří seznamy vrcholů a trojúhelníků. Následně se z dvourozměrného pole mapy vytvoří mřížka čtverců. Tyto čtverce, se skládají ze čtyř vrcholů a čtyř dalších bodů, které leží v polovině délek jednotlivých stran. Vrcholy jsou označeny jako kontrolní body a body mezi nimi, slouží jen jako jakési pomocné body při generování. Tento princip generování vychází z algoritmu Marching squares (viz. kapitola 2.5).

Při tvorbě těchto čtverců se vždy zkontroluje, které kontrolní body mají mít vygenerovanou polygonální síť, a které jsou naopak prázdné. Tento údaj je nutné uchovat v číselné podobě. Tohoto je docíleno tak, že se vytvoří bitová maska, jejíž jednotlivé bity reprezentují kontrolní vrcholy čtverce. Aktivní vrchol značí bit 1 a neaktivní zase 0. Tato bitová maska poté tvoří číslo, které zastupuje danou variantu tvorby sítě. Na základě tohoto čísla se tedy rozhodne o způsobu generování tohoto čtverce. Body nutné pro danou variantu polygonální sítě jsou poté použity jako vrcholy tvořící trojúhelníky. Při tvorbě trojúhelníků se jejich vrcholy nejdříve přidají do seznamu vrcholů. Během tohoto vkládání se zároveň uloží jejich indexy, které reprezentují pozici v poli či seznamu (v závislosti na implementaci) vrcholů. Následně se vždy trojice těchto indexů přidá do seznamu trojúhelníků, aby mohl být trojúhelník vygenerován. Pořadí těchto indexů musí být zároveň přidáno ve směru hodinových



Obrázek 4.5: Na obrázku je příklad vygenerované mapy pomocí generátoru. Na levé polovině je mapa v rozmezí velikosti 200 x 200. Generování takové velikosti mapy je rychlé, ale nemusí vyhovovat potřebám využití. Tudíž ji lze zvětšit, jak lze vidět v pravé polovině obrázku. Zvětšení zachovává vzhled a poměry mapy, jediné, co se změní, je měřítko. Díky tomuto vzniká v tomto případě mapa o rozměrech 400 x 400. Tyto větší rozměry by trvalo generovat více krát příliš dlouho, tudíž pří generování půdorysu je vhodné generovat v menších rozměrech a až výslednou mapu poté zvětšit [2].



Obrázek 4.6: Obrázek ukazuje možnosti editování Perlinova šumu za pomoci vlastního nástroje. Na levé polovině obrázku je chodbám ulehčeno šíření za pomoci zcela bílé barvy. Naopak na pravé polovině je vidět omezení chodeb v zcela černém kruhu, přes který se chodby nejsou schopny rozšířit.

ručiček, aby z daného pohledu byla polygonální síť trojúhelníku viditelná. V případě po-

řadí proti směru hodinových ručiček, by trojúhelník byl viditelný pouze z protějšího směru pohledu.

Po vytvoření všech trojúhelníků se seznamy vrcholů a trojúhelníků zkopírují do příslušných polí komponentu Mesh Filter, který poskytuje engine Unity. Dále je nastaven index buffer tohoto komponentu na 32 bitový, jelikož je polygonální síť generována v jednom kuse. Tudíž u větších rozměrů mapy občas nestačil index buffer 16 bitový, jelikož mapa již měla množství vrcholů větší než 65535. Na závěr jsou vypočítány UV koordináty potřebné pro správné natažení materiálu na polygonální síť.

4.5 Generování objektů interiéru

Generování mapy je implementováno na základě L-systému, jak je již zmíněno v kapitole 3.5. Konkrétně je generování implementováno tak, že jednotlivé dlaždice jsou označeny jednopísmenným či vícepísmenným symbolem. Tento symbol značí sémantiku oblasti, ve které se dlaždice nachází. Například všechny dlaždice místností jsou označeny symbolem 'R' (zkratka odvozená od anglického slova "room"), výjimkou pak jsou dlaždice na vstupu do místnosti a u zdi místnosti. Vstupní místnosti mají označení 'RE' (symbol značící zkratku anglických slov "room entry") a dlaždice u zdí pak 'RO' (symbol značící zkratku anglických slov "room outline"). Podobně se označí všechny dlaždice chodeb.

Po přidělení těchto symbolů jsou pak vytvořena pravidla určující, jak se mají dané symboly přepsat. Vzniklý přepis značí, zda se má na dlaždici vytvořit objekt nebo dlaždice zůstane prázdná. V případě, že nezůstane prázdná, určuje tento symbol také jaký objekt se tu umístí.

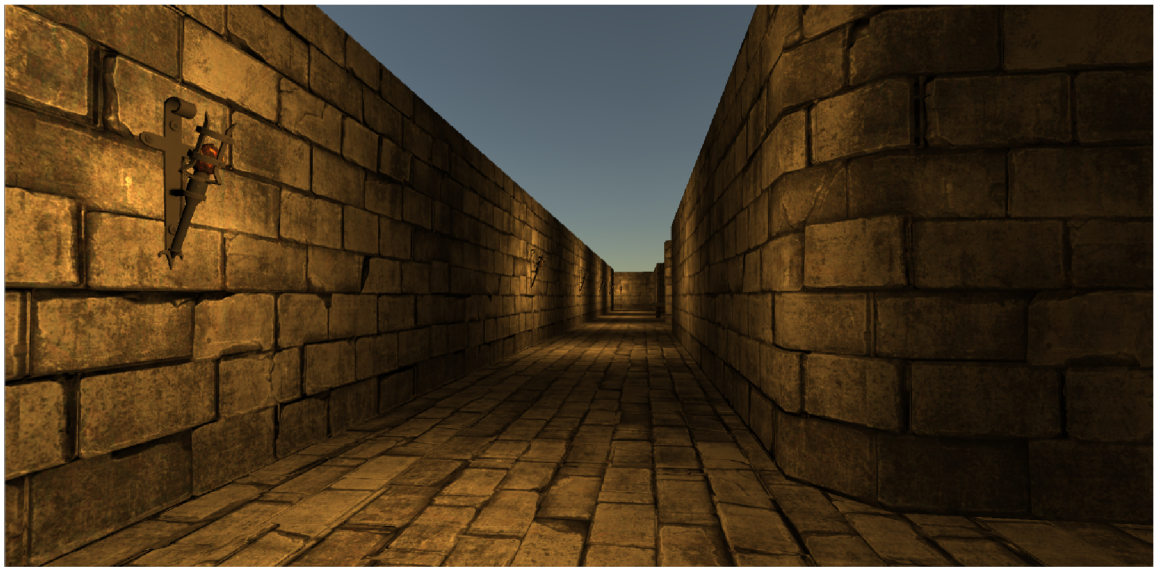
Tento přepis pravidel je podmíněný. Takže v případě, že by měl na dané pozici vzniknout objekt, který se do dané oblasti nevejde, pak k přepisu nedojde. K přepisu také nemusí dojít kvůli pravděpodobnosti, která určí, že pozice zůstane prázdná. Tento přepis je navíc ovlivňován předchozím přepisem symbolu. To způsobí, že vedle objektů budou jen objekty, které by se mohly vedle původního objektu vyskytovat. Takže například vedle sudu stojí s větší pravděpodobností další sud, jelikož sudy bývají často ve shlucích.

Objekty pro výplň jsou získány převážně ze zabudovaného Unity Store, kde komunita poskytuje její obsah, který vytvořila. Těchto objektů, které by se pro tento plugin hodilo a nebylo za poplatek, nebylo mnoho, ale pro ukázku možností generátoru je jejich množství postačující.

4.6 Implementace kamery pro prohlížení interiéru

Pro prohlížení vygenerované mapy s dekoracemi uvnitř je vytvořena ladící postava hráče, která se může volně pohybovat ve směru osy x a y, jak je již zmíněno v kapitole 3.6. K implementaci pohybu kamery po osách x a y je využito metod, které poskytuje komponent Character Controller, který je implementován v enginu Unity. Dále je implementována možnost se rozhlížet do stran na místě tak, že objekt, který je oddělený od zbytku těla postavy, simuluje hlavu a tudíž se hýbe zatímco zbytek stojí na místě. Ukázka z prohlížení mapy pomocí kamery viz. kapitola 4.7.

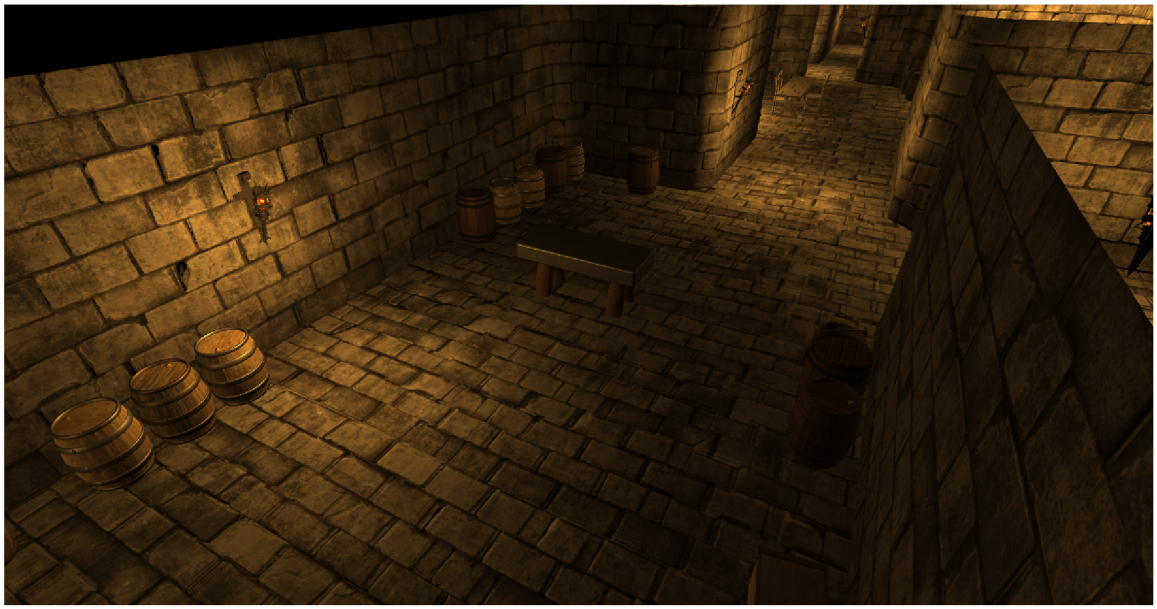
4.7 Vzhled výsledného vnitřku mapy



Obrázek 4.7: Na obrázku je vidět ukázka vnitřních dekorací vygenerované mapy. Na tomto konkrétním obrázku je vidět, že chodby nejsou tak hustě zaplněny objekty, jelikož by na chodbách tolik objektů být nemělo. Hlavní dekorací chodeb je pouhé osvětlení.

Obrázek 4.8: Na obrázku je vidět ukázka vnitřních dekorací vygenerované mapy. Na tomto konkrétním obrázku lze vidět výplň části místnosti. Jelikož se tu jedná o místnost, tak v ní je větší množství objektů než na chodbách.

Obrázek 4.9: Ukázka vygenerované místnosti, ve které jsou podél zdí sudy a další vhodné objekty. Dále je ve středu místnosti vygenerován stůl s židlemi náhodně umístěnými kolem něj.



Obrázek 4.10: Ukázka vygenerované místnosti, ve které jsou podél zdí sudy a ve středu místnosti je vygenerován stůl pro řemeslníka.

Kapitola 5

Závěr

Cílem této práce bylo vytvořit plugin, který procedurálně generuje podzemní kobky. Toto zahrnovalo seznámení se se základy procedurálního generování jako je pseudonáhodný generátor čísel, generování Perlina šumu a jeho použití. Dále bylo zapotřebí obeznámit se s často používanými postupy pro generování půdorysu podzemních kobek a také se základy formálních gramatik, jejichž principy byly využity při generování výplně mapy. Seznámil jsem se také s herním enginem Unity, který značně usnadňuje vývoj videoher. V neposlední řadě jsem si blíže nastudoval zásady objektově orientovaného programování, se kterými jsem neměl dosud žádné zkušenosti, a které jsou při programování v Unity zcela nutné a zásadní.

Výsledkem práce je tedy generátor, poskytující snadný způsob vytvoření mapy pro hru, která by měla obsahovat podzemní kobky. Generátor byl vytvořen tak, aby byl přívětivý pro práci s ním a zároveň do něj nemusel vývojář příliš zasahovat.

Všechny body zadání práce byly tedy splněny. Práce implementuje několik postupů procedurálního generování a jako celek generuje ukázku mapy podzemních kobek. Jako možná vylepšení v budoucnosti by bylo zlepšení půdorysu mapy. K tomu by mělo dopomoci použití osmiokolí při generování chodeb, což značně komplikuje implementaci, ale mohlo by to zvýšit míru realističnosti vzhledu. Dále bych zapracoval na komplexnosti generování vnitřních dekorací a množství objektů k tomu využitých, případně bych si sám zkusil nějaké dekorace vytvořit pomocí nějakého grafického nástroje jako je např. Blender. Dalším důležitým krokem by poté bylo rozdělení polygonální sítě mapy na více částí, což by umožnilo vytvářet mnohem větší celky.

Literatura

[1] Academy, K. *Perlin noise* [online]. Khan Academy, 2020 [cit. 2020-05-10]. Dostupné z: https://www.khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-noise/a/perlin-noise.

[2] Chmielewski, S. a Tompalski, P. Estimating outdoor advertising media visibility with voxel-based approach. *Applied Geography*. Říjen 2017, sv. 87, s. 1–13. DOI: 10.1016/j.apgeog.2017.07.007.

[3] Coding, C. *Marching Squares, partitioning space* [online]. Catlike Coding, 2018 [cit. 2020-05-10]. Dostupné z: https://catlikecoding.com/unity/tutorials/marching-squares/.

[4] Edelsbrunner, H. *Geometry and Topology for Mesh Generation*. Cambridge University Press, 2001. ISBN 978-0-521-79309-4.

[5] Frey, P. a George, P.-L. *Mesh Generation*. 2. vyd. ISTE Ltd and John Wiley & Sons, Inc, 2008. ISBN 978-1-84821-029-5.

[6] Hordějčuk, V. *Formální gramatika* [online]. voho.eu, 2020 [cit. 2020-05-10]. Dostupné z: http://voho.eu/wiki/formalni-gramatika/.

[7] Jiang, T., Li, M., Ravikumar, B. a Regan, K. Formal grammars and languages. Leden 2010, s. 4–9.

[8] Kampffmeyer, M., Dong, N., Liang, X., Zhang, Y. a Xing, E. ConnNet: A Long-Range Relation-Aware Pixel-Connectivity Network for Salient Segmentation. *IEEE Transactions on Image Processing*. Duben 2018, PP, s. 2–6. DOI: 10.1109/TIP.2018.2886997.

[9] MathWorks. *Pixel Connectivity* [online]. The MathWorks, Inc., 2020 [cit. 2020-05-10]. Dostupné z: https://www.mathworks.com/help/images/pixel-connectivity.html.

[10] Morphocode. *Intro to L-systems* [online]. Morphocode, 2020 [cit. 2020-05-10]. Dostupné z: https://morphocode.com/intro-to-l-systems/.

[11] Przemyslaw, P. a James, H. *Lindenmayer Systems, Fractals, and Plants*. Springer-Verlag New York, 2013. ISBN 978-1-4757-1428-9. Dostupné z: http://algorithmicbotany.org/papers/#abop.

[12] Singla, Y. *Pseudo Random Number Generator (PRNG)* [online]. GeeksforGeeks, 2020 [cit. 2020-05-10]. Dostupné z: https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/.

[13] UNITY. *Game engines—how do they work?* [online]. Unity, 2020 [cit. 2020-05-10]. Dostupné z: https://unity3d.com/what-is-a-game-engine.

[14] UNITY. *This is why creators choose Unity* [online]. Unity, 2020 [cit. 2020-05-10]. Dostupné z: https://unity.com/our-company.

[15] UNITY. *Unity User Manual (2019.3)* [online]. Unity, 2020 [cit. 2020-05-10]. Dostupné z: http://docs.unity3d.com/.

[16] VANDRAKE. *Procedural Generation in Game Development* [online]. Vandrake, 2020 [cit. 2020-05-10]. Dostupné z: https://www.davidepesce.com/2020/02/24/procedural-generation-in-game-development/.