ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zemko** Jméno: **Jiří** Osobní číslo: **421475**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Počítačové hry a grafika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Procedurální generování modelů měst

Název bakalářské práce anglicky:

Procedural generation of city models

Pokyny pro vypracování:

Prostudujte existující metody pro procedurální generování modelu města. Soustřeďte se na metody umožňující řídit generování modelu pomocí existujících dat skutečných měst. Zmapujte automatické techniky, které umožní model města doplnit o statické detailní prvky jako je městský mobiliář, vegetace, zaparkované automobily, chodci na ulicích. Soustřeďte se zejména na globální strukturu modelu a smysluplné rozmístění detailních prvků. Při generování modelu uvažujte i výškový profil terénu získaný z dostupných zdrojů. Ve vhodném nástroji (např. Houdini) implementujte vhodnou metodu generování města a vygenerujte nejméně tři modely různých měst resp. jejich částí. Vyhodnoťte vygenerované modely porovnáním s fotografiemi skutečných měst.

Seznam doporučené literatury:

[1] Michael Schwarz and Pascal Müller. 2015. Advanced procedural modeling of architecture. ACM Trans. Graph. 34, 4, Article 10, 2015.
[2] Smelik, Ruben M., et al. 'A survey on procedural modelling for virtual worlds.' Computer Graphics Forum. Vol. 33. No. 6. 2014.
[3] Steinberger, Markus, et al. 'On□the□fly generation and rendering of infinite cities on the GPU.' Computer graphics forum. Vol. 33. No. 2. 2014.
[4] Schwarz, Michael, and Pascal Müller. 'Advanced procedural modeling of architecture.' ACM Transactions on Graphics (TOG) 34.4 (2015): 1-12.
[5] Nishida, Gen, et al. 'Interactive sketching of urban procedural models.' ACM Transactions on Graphics (TOG) 35.4 (2016): 1-11.
[6] Aliaga, D. G., Vanegas, C. A., & Benes, B. (2008). Interactive Example-Based Urban Layout Synthesis. ACM SIGGRAPH Asia 2008 Papers.
[7] Nishida, G., Garcia-Dorado, I., Aliaga, D. G., Benes, B., & Bousseau, A. (2016). Interactive Sketching of Urban Procedural Models. ACM Trans. Graph., 35(4), 130:1–130:11.
[8] Zhou, S., Yoo, I., Benes, B., & Chen, G. (2014). A hybrid level-of-detail representation for large-scale urban scenes rendering. Computer Animation and Virtual Worlds, 25(3-4), 243–253.
[9] Jana Kejvalová. Procedurální generování 3D modelu dle mapových podkladů. Diplomová práce, ČVUT FEL 2019.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Jiří Bittner, Ph.D. Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **14.09.2021** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **19.02.2023**

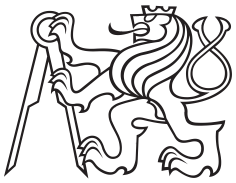
doc. Ing. Jiří Bittner, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.	
Datum převzetí zadání	Podpis studenta

Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Procedurální generování modelů měst

Jiří Zemko

Školitel: doc. Ing. Jiří Bittner, Ph.D.
Květen 2022

Poděkování

Chtěl bych tímto poděkovat svému vedoucímu práce, doc. Ing. Jiřímu Bittnerovi, Ph.D., za jeho ochotu, pomoc a cenné rady. Dále bych chtěl poděkovat Lucce za obětavou korekturu textu, ač mu plně nerozumněla. A v neposlední řadě svým kolegům, Koťátka a Pivotour, kteří mi poskytli nápady pro testovací materiály na rozbití mých algoritmů, což jim dozajista udělalo radost.

Prohlášení

Tímto prohlašuji, že jsem práci zpracoval samostatně a použil jsem pouze podklady a zdroje uvedené v této práci v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 20. května 2022

Abstrakt

Cílem této bakalářské práce je vytvořit co nejobecnější nástroj pro vytváření modelů měst se zaměřením na silnice a jejich okolí. Detaily jsou zaměřovány hlavně na globální hledisko, jako okolní překážky budov, parkoviště a terén.

Veškerá data pro generování jsou brána pouze z veřejně dostupných zdrojů a to převážně OpenStreetMap. Generování dat je co nejvíce na nich závislé, aby se výsledný model co nejvíce přiblížil realitě.

Ke generování byl využit procedurální modelovací software Houdini s možností psaní vlastních skriptů v jazyce Python a VEX.

Přesnost řešení je pak ověřována na několika setech stažených map s vizuálním srovnání za pomocí fotografií nebo Google Maps.

Klíčová slova: Houdini, procedurální generování, Python, silnice, město, terén

Školitel: doc. Ing. Jiří Bittner, Ph.D. Praha 2, Karlovo náměstí 13, E-421

Abstract

The aim of this bachelor thesis is to create the most general tool for creation of city models with focus at roads and their surroundings. Details are aimed mainly globally such as surrounding obstruction of buildings, parking and terrain.

All data used for model generating are public, mainly from OpenStreetMap. Data generation are as dependent on previous data as possible, so the outcome is as close to reality as possible.

For data generating was used procedural modeling software Houdini with possibility writing own scripts in Python and VEX language.

Solution accuracy is then verified on several downloaded map sets with visual comparation with either photos or Google Maps.

Keywords: Houdini, procedural generation, Python, road, city, terrain

Title translation: Procedural generation of city models

Obsah

1 Úvod **1**
1.1 Úvod do procedurálního modelování 1
1.1.1 Procedurální modelování obecně 1
1.1.2 Procedurální generování objektů 2
1.1.3 Procedurální generování měst . 3
1.1.4 Optimalizace procedurálních modelů 4
1.2 Stanovení cílů 4

2 Související práce **5**
2.1 OSM mapy 5
2.2 Houdini 5
2.2.1 Grafické rozhraní 6
2.2.2 Python skriptování 6
2.2.3 VEX skriptování 7

3 Realizace **9**
3.1 Základ mapy 9
3.2 OSM data 10
3.3 Návrh parkovišť 10
3.4 Návrh silnic 12
3.5 Návrh městského mobiliáře 13
3.5.1 Lampy 13
3.5.2 Parkující auta v ulicích 14
3.6 Návrh terénu 14
3.6.1 Zdroje potřebné ke generování 14
3.6.2 Tvorba terénu 15
3.6.3 Umístění prvků na terén 15
3.7 Optimalizace modelu 16

4 Texturování **19**
4.1 Zdroje texturování 19
4.2 Základní textura 19
4.3 Použité souřadné systémy 20
4.3.1 Světový geodetický systém 1984 20
4.3.2 Křovákovo zobrazení 20
4.4 Texturování z leteckých snímků . 21

5 Výsledky **25**
5.1 Přehled a měřítko výsledků 25
5.2 Porovnání výsledků 27
5.3 Limitace 30
5.4 Diskuze 33

6 Závěr **35**
6.1 Zhodnocení 35
6.2 Směřování další práce 35

Literatura **37**

A Uživatelský manuál **39**

B Tabulka přehledu miniatur modelů **43**

Obrázky

1.1 Vygenerované domy na základě gramatiky jazyka CGA++ a vychytávání vznikajících problémů na základě manipulace se vstupními daty a pozorování výsledků. [SM15] 2
1.2 Vytvořené bloky na základě různých vstupních dat. [SM15] 3

2.1 Příklad zapojení SOP. 6
2.2 Příklad VEX skriptu. 7

3.1 Importovaná mapa do Houdini přes OSM importer. 9
3.2 Příklad dat uložených v primitivech z map. 10
3.3 Vlastní doprogramované rozhraní pro ovládání generaci vlastních parkovišť pro testování. 11
3.4 Příklad výsledného testovaného pravidelného parkoviště bez umístěných vozidel. 12
3.5 Příklad výsledného testovaného náhodného parkoviště s umístěním vozidel. 12
3.6 Webové rozhraní Earthexploreru - červený polygon značí vyhledávání dat pro vybranou oblast, zatímnco modrý stopu jednoho datového souboru. 14

4.1 Základní texturování - Karlovo náměstí, pohled ze severní strany od Novoměstské radnice. 20
4.2 Česká republika ve Křovákově zobrazení (černě) v porovnání s WGS84 (oranžově).[WF22a] 21
4.3 Příklad souboru jgw - 1. řádek určuje inkrementaci ve směru x v souřadném systému o jeden pixel, 2. a 3. řádek otočení (ve většině map není), 4. řádek inkrementaci ve směru y v souřadném systému o jeden pixel (zde je vidět záporná hodnota, tedy obrací směr inkrementace) a 5. a 6. řádek určuje x a y souřadnici levého horního rohu mapy. [io17] 21
4.4 Schéma transformace načtené textury: 1. načtená textura 2. upravená velikost textury z poměru velikosti snímku a mapy na terénu 3. otočení textury o odchylku 4. posunutí o vzdálenost levých rohů textury a mapy na terénu. 22
4.5 Příklad texturování s pužitím popsané transformace, kde viditelně nesedí měřítko. 23

5.1 Příklad křižovatky v panelové solitérní zástavbě Kladně (vlevo google maps, vpravo model). 27
5.2 Příklad náměstí v Pardubicích v blokové zástavbě (vlevo google maps, vpravo model). 28
5.3 Příklad parkování v Českých Budějovicích v historické zástavbě (vlevo google maps, vpravo model). 28
5.4 Příklad historické zástavby ve Vídni (vlevo google maps, vpravo model). 29
5.5 Příklad použití lamp osvětlení v Dejvicích, ČVUT, Praha (vlevo google maps, vpravo model). 29
5.6 Příklad různých tříd silnic na Pražského povstání, Praha (vlevo google maps, vpravo model). 30
5.7 Detail rezidenční oblasti Pražského povstání, Praha. 30
5.8 Klenčí pod Čerchovem, vlevo letecký snímek ([kle17]), vpravo model. 31
5.9 Klenčí pod Čerchovem - obec ve svahu. 31
5.10 Kruhový objezd Magic Roundabout, Swindon, Anglie (vlevo google maps, vpravo model). 32
5.11 Příklad chybějících dat na obci Durmanec (Chorvatsko), kde vlevo vidíme vygenerovaný model z OSM dat a vpravo pohled Google Streetview, kde je jednoznačně více budov, než OSM data obsahovala). 32
5.12 Příklad špatně zvolených výšek budov u sousedících objektů. 33

5.13 Špatné řešení parkoviště ve tvaru L, Poruba, Ostrava. 33
5.14 Příklad, kde se kříží osvětlená a neosvětlená ulice, kde umisťování lamp probíhá nekorektně (dvě ulice směrem k nám jsou neosvětlené a směrem dopředu jsou 2 osvětlené). 34

Tabulky

3.1 Porovnání 2 testovacích modelů a jejich doby vytváření bez optimalizace, s částečnou a plnou. 17

5.1 Tabulka statistik vybraných modelů. 25

B.1 Tabulka statistik vybraných modelů. 43

Kapitola 1
Úvod

1.1 Úvod do procedurálního modelování

1.1.1 Procedurální modelování oběcně

Procedurální modelování je obor výzkumů, ve kterém se aktivně lidé projevují již přes 40 let. Toto modelování je převážně aplikováno na generaci modelů terénu, měst, rostlin, budov, silnic, sítí řek a dalších chaoticky pravidelných objektů. Samotná definice není jednotná, ale obecně uznávaná definice je, že procedurální modelování je technika, která nabízí různé nebo podobné výsledky na základě jejích procedur, nebo programů .

Hlavní výhody spočívají v obecnosti a že můžete získávat různé použitelné výstupy na základě vložených dat, kde dochází k obohacování vstupních dat. Zde odpadá práce se samotným modelováním a odhadováním detailů člověkem. V případě zpracovávání většího množství dat procedurální modelování tak přislibuje značné urychlení automatizací procesu. Další výhodou je zároveň právě datová komprese, kdy výsledkem by měla být co nejmenší halda přebytečných neužitečných dat, které se vyfiltrují, nebo vygenenerují, pouze pokud jsou potřeba.

Výhoda se ovšem stává zároveň i nevýhodou, kdy je zde značná komplexicita a hledání správných algoritmů pro produkci použitelných dat může trvat někdy dlouho. Obzvláště pokud se člověk snaží napodobovat struktury, které vznikly chaotickými procesy v přírodě. Zde si nemůžete být jistí stupněm úprav, a zda vám nechybí některá data pro přesnou produkci. Zde se pak setkává několik oborů z důvodů složitosti. Ověřování správnosti se pak provádí také náročně, kdy pokud netušíte na 100% jak generování funguje, můžete očekávat vám neznámé chyby.

Podle typů přístupů k řešení si pak můžeme rozdělit jednotlivé typy modelování.

- Prvními pokusy byly stochastické přístupy, kdy byly na pevno nastaveny hodnoty - ve většině šumové generátory. Ty se nejčastěji používaly na tvorbu náhodných terénů.
- Pokusy s umělou inteligencí - používání algoritmů pro hledání nejkratších cest pro generaci silnic, nebo směru růstu rostlin, při pokusu přiblížit se

co nejvíce slunci.

- Čistě procedurálním přístupem bychom mohli nazvat simulace v převážně v oblastech fyziky pro modelování terénu při nasimulování eroze, generování porostu v simulaci souboje o světlo a živiny, nebo vytvoření města na základě pohybu lidí.

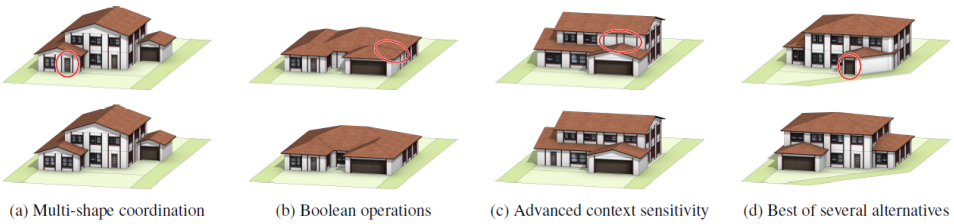
- Existuje přístup na základě vytvoření "gramatiky", kdy je řečeno při jakých datech by se mohlo jakým způsobem pokračovat. Do tohoto způsobu spadají L-systémy pro generování fraktálů, listů rostlin. Nebo tvarové gramatiky pro právě generování budov.

- V některých případech může být procedura poháněna čistě daty, kdy se bude učit z okolního světa jak má postupovat se svým podobným systémem.

Většina procedurálních modelování avšak nezůstává pouze u jednoho přístupu, ale dost často tyto způsoby kombinuje pro dosažení lepších, obecnějších výsledků, které se pak dají uživatelem snadněji korigovat [Sme14].

1.1.2 Procedurální generování objektů

Zajímavým příkladem procedurálního generování je práce v Esri R&D Center Zurich [SM15]. Použili gramatický přistup pro generování rodinného domu, kdy předvedli ovšem navíc, že samotný postup je jednoduchý, ovšem musí v něm existovat spoustu výjimek (obr. 1.1), které můžeme očekávat i v nadcházejícím projektu. Sice můžete vygenerovat dům, který splňuje základní požadavky stavby, ale je potřeba zanést problém dále, a zeptat se, zda-li se vám tak líbí. Zda-li se dá do všech dveří vstoupit. Jestli kvůli transformačním operacím není potřeba zavést nový druh oken, jelikož některé booleanovské operace nezpůsobili příliš dlouhou zeď bez obyčejných oken. Samozřejmě tyto chyby nastávají i v reálu, ale třeba je pro naše potřeby nechceme a snažíme se vytvořit princip "dokonalé stavby".



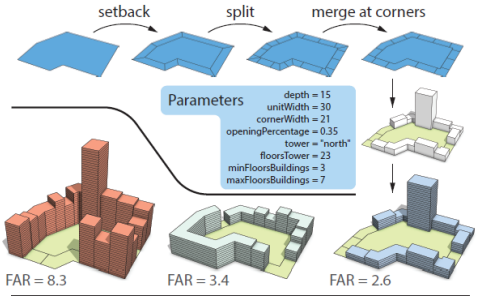
(a) Multi-shape coordination (b) Boolean operations (c) Advanced context sensitivity (d) Best of several alternatives

Obrázek 1.1: Vygenerované domy na základě gramatiky jazyka CGA++ a vychytávání vznikajících problémů na základě manipulace se vstupními daty a pozorování výsledků. [SM15]

1.1.3 Procedurální generování měst

Při generování měst je možné postupovat několika způsoby. Ten "nejjednodušší"je nakrmit algoritmus daty již existujících měst a vytvořit podle něj další. Takovýto přístup zvolili při výzkumu na Purdue University v roce 2008 [AVB08]. Řešili problém doplňování chybějících bloků, nebo rozšiřování stávajících měst. Jejich postup spočíval v analýze okolního města a pak replikací tvarů ulic pro vytvoření nových struktur. Tvary ulic se mohou velice lišit, podle polohy, historického kontextu, nebo míry provozu. Rozhodně je rozdíl v tvaru organické sítě, které vznikaly bez architektů, ale měly řád lidí, kteří na něco navazovali (Např. Staré město pražské), nebo pokud byl jasně předem dán urbanistický plán města nového s převahou rovných ulic a myšlenkou snadnější dopravní komunikace (Např. ulice Washingtonu DC navrhované dle stylu zahrad francouzských romantických zahrad). Dále jelikož vytvářeli pouze ideu mapy, došlo k rozvržení bloků podle zvyklostí a nakopírování stávajících budov z načtených dat do nově vytvořených bloků.

Problémy třetího rozměru, aby se zachovala co nejkvalitnější tzv. střešní krajina, řešili opět pánové z již zmíněného Esri R&D Center Zurich [SM15]. Rozšířili dále svou práci z pozemku na celé bloky a snažili se rozvrhovat výšky budov podle velikostí bloků, dovolené stavební výšky, koeficientem zastavěnosti, a dalšími... Opět šlo o použití jednoduché gramatiky, která rozhodovala o spojování, rozdělování a další členěnění objektů (obrázek 1.2).



Obrázek 1.2: Vytvořené bloky na základě různých vstupních dat. [SM15]

Složitějším urbanistickým počinem pak bylo účelové rozmisťování budov na základě občanské vybavenosti. v projektu Purdue university a UC Berkeley [VGDA+12]. Nejen že se snažili najít inspiraci v přírodě, kde řešili jednotlivou prosluněnost bytů na základě výšky a stínění ostatních objektů, snažili se i právě dohledat co nejkomfortnější bydlení na základě dochozí vzdálenosti k potřebným objektům. Objektům jako jsou školy, práce, parky, potřeba nakupovat, atd., podle které se mají normově plánovat nová města. Modely pak mohly poskytovat obrovskou variabilitu, podle kladení důrazů na různé indikátory míry komfortu pro žití ve městě.

Existuje několik programů, které pro představu nabízí převedení dat map do modelu města. Tyto programy celkem dobře zmapovala diplomová práce Procedurální generování 3D modelu dle mapových podkladů z roku 2019 [Kej19]. Existuje je jich hned několik, každá nabízí různou míru realističnosti, ale obecně nenabízí příliš velkou míru variability a uživatelské ovladatelnosti. Mimojiné se musí občas doplnit konkrétními modely, tudíž jde o procedurální generaci částečnou, kdy nemusí fungovat na obecných případech (protože všechny přístupy byly testovány kvůli objektivitě na konréktním výřezu

mapy).

1.1.4 Optimalizace procedurálních modelů

S komplexnější generováním přichází limitace přístrojů, na kterých výpočet probíhá. Jak rychlostní, tak hlavně paměťová, kvůli práci s větším množstvím dat. Proto je složitější systémy nutno optimalizovat.

Zjednodušení, co se týče renderování výsledků, je nesnažit se generovat to, co nebude viděno ve výsledku. Touto problematikou se zabývali v článku On-the-fly Generation and Rendering of Infinite Cities on the GPU [Ste14]. Představili problematiku a ukázali provedení filtrace budov na základě generačního generování a vybírání konkrétních objektů, které byly na místě zobrazit.

Dalším zjednodušením je zavedením LOD (Level of detail - úroveň detailu), který se běžně používá pro optimalizaci ve hrách. Čím je objekt vzdálenější, tím jednodušší může být jeho model, takže bude uchovávat méně dat při jeho zobrazení (ne však jako instance před zobrazením uložená samotná). Tuto problematiku pokryli např. v článku A hybrid level-of-detail representation for large-scale urban scenes rendering [ZYBC14]. Předvedli návrhy zjednodušování ploch a sjednocování bodů vzhledem ke vzdálenosti od kamery, které samozřejmě podpořili daty o snížení potřeby paměti.

1.2 Stanovení cílů

Předchozí rešerše ukázaly, že dost prací se věnovalo procedurálnímu modelování budov, měst a uličních sítí, ale ne tolik už detailům ulic, a doplňkům jako parky, pakroviště a další. Proto se tato práce chtěla odlišit od ostatních.

Cílem je pokusit se navrhnout podle map skutečných míst infrastukturu města jinou než budov - zaměřeno hlavně na silnice a bezprostřední okolí. Čerpat pokud možno co nejvíce z informací těchto map, aby se dosáhlo co nejbližšího přiblížení reality do města. Dimenzování silnic, rozplánování parkovišť, osvětlení ulic a umístění aut zaparkovaných do ulic, čerpat co nejvíce ze základu státních norem pro korektní rozmístění a dimenzování jednotlivých prvků. Pokusit se o co nejobecnější přístup, aby se dosáhlo dobrých výsledků na většině náhodně i specificky vybraných případů.

Kapitola 2
Související práce

2.1 OSM mapy

Open street maps (dostupné z odkazu) je projekt, který se snaží vytvářet a distribuovat geografická data zdarma. Vzhledem k tomu, že jsou zdarma, může spousta uživatelů používat tyto data a dokonce je rozšiřovat a opravovat o další data. Exportovaná data pak jsou ve formě grafu souboru s koncovkou ".osm"s uzly obsahující pak spoustu informací využitelných dále, které nejsou k vyčtení z obvyklých map (jako je třeba výška zástavby, druh povrchu, typ pozemku...), což je složení různých typů map dohromady. Jeho využití, které je zdarma, se spoustou informací, je pro tuto práci ideální.

2.2 Houdini

Houdini je 3D animační software od firmy SideFX, který byl vyvinut z nástrojů Prism Suite pro procedurální generování, první verze v roce 1996. Tento software používají známé animační společnosti, včetně Pixaru a Disneyho [WF21]. Byly v něm animovány i známé filmy jako Zootopia, nebo Frozen 2 [fro20].

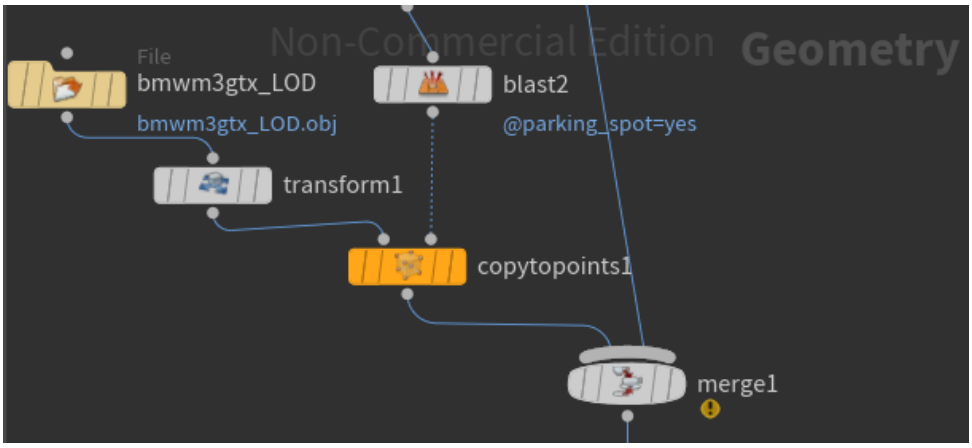
Hlavní výhodou pro tento projekt je, že podporuje procedurální generování modelů. Dále podporuje modelování pomocí vlastních skriptů jazyka Python, nebo vlastnímu jazyku VEX, který vychází z C. Houdini je sice založen na C++, ale většina jeho nově přidaných funkcí je založena právě na Pythonu. Projekt pracuje s oběma jazyky, protože každý poskytuje jiné výhody. Mimo jiné je schopen používat i expresivní funkce a odkazy v kolonkách pro hodnoty, čímž je možnost vyhnout se zbytečnému množství dalších krátkých skriptů pouze za účelem použití proměnných případně jejich matematickou úpravu.

Houdini má 5 typů licencí, rozdělené na Komerční, Indie a Vzdělávací. Pro tuto práci byla zvolena verze vzdělávací APPRENTICE a dále se bude tak popisovat pouze tato verze software. Tato verze bohužel nedovoluje exportovat modely mimo Houdini. Renderování se zde provádí s vodotiskem a nelze udělat větší rozměry, než-li 1280 na 720 pixelů.

2.2.1 Grafické rozhraní

Houdini dovoluje jak klasické modelování pouze v hlavním okně, tak ale hlavně podporuje takzvané blokové modelování, kdy jsou objekty a geometrické prvky rozděleny do hierarchie jednotlivých bloků, na které lze zvlášť aplikovat transformace dalších bloků, což je základ procedurálního modelování.

Vše co vytvoříte (včetně vašeho modelování v okně), totiž vytvoří v projektu "node"(uzel) a ty se dále propojují sítí s dalšími uzly (viz obr. 2.1), které Houdini představuje jako "operátory", které mají různý vstup na kterém operují a tudíž mohou dávat různé výsledky. Základní typy operátorů jsou SURFACE (týkající se geometrie jako takové), CHANNEL (pro pohyb), DYNAMIC (pro simulace), VEX (pro nastavení shaderů), COMPOSITING (pro práci s obrázky) a RENDER (pro výstupy). Pro práci modelování nás bude zajímat hlavně SURFACE nody a jejich nové vytváření (zkráceně SOPS - surface operators). Výhodou v používání SOPS je, že si můžete vždy vybrat, který krok vaší generování prostředí chcete zobrazit a výsledná geometrie se pak dá manipoluvat zapojením konkrétních SOPS do outputu a vytvářet tak nedestruktivní postupy generování.



Obrázek 2.1: Příklad zapojení SOP.

Dále, pokud se budeme pohybovat mimo operátory a dostaneme se přímo ke geometrii, jednotlivým vertexům (hranám modelu), primitivům, bodům nebo skupinám lze přiřazovat tzv. atributy jako vlastnosti. Ve většině se jedná o data typu string (ale mohou nabývat i jiných typů), která se dají dále využít jako informace pro další operátory, co mají s danou geometrií provést, případně jako jejich doplňující hodnotu míry transformace. [Houb].

2.2.2 Python skriptování

Houdini sice dovoluje skriptovat přímo v jeho vlastním software, ale je to poněkud nekomfortní, pokud píšete delší skripty. Proto je i výhodou, že lze spojit s vlastním skriptovacím nástrojem, jako je např. Visual studio. Při správném nastavení lze i dosáhnout ve Visual Studiu (nebo jiném podobném

software) možnosti našeptávání základu knihovny Houdini "hou" [Houa]. Tento zmíněný Python modul obsahuje všechny podmoduly a třídy, které Houdini běžně využívá. Nejdůležitější třídou pro nás bude geometry. Dále je možnost využívat i dodatečných Python knihoven, ovšem uživatel je nucen doplnit cesty do systémových proměnných, aby je Houdini rozeznával - používá totiž vlastní nainstalovaný Python ve složce s Houdini.

2.2.3 VEX skriptování

Jak již bylo zmíněno výše, VEX je založen na jazyce C, jakož i jeho syntax (viz obrázek 2.2) a tudíž rychlejší než Python (podle tvůrců dosahuje až rychlosti jazyka C). Je rozšířen o další funkce a uživatel si je schopen vytvářet i vlastní. Inspiraci si dále bere hodně i z jazyka C++ (má funkce pro řazení a složitější datové struktury) a RenderMan. Jeho použití je právě v jednotlivých typech operátorů pro Rendering, modeling, particles a další. [Houc]

V projektu je použit převážně v uzlech, které podporují použití skriptu na veškeré vybrané atributy (attribewrangle), který zastupuje foreach smyčku pro vybrané detaily - konkrétní data vázaná k prvkům jako jsou body, vertexy nebo primitiva - nebo jmenované prvky samotné a nová data pro ně tak vytvořit.

VEXpression

```
@minlat = detail(1, "min_lat");
@minlon = detail(1, "min_lon");
//calculating shift
f@shiftx = -(detail(1, "min_lon") - detail(0, "myminlon") - 0.5);
f@shiftz = -(detail(1, "min_lat") - detail(0, "myminlat") - 0.5);

//adjusting to metres
@shiftx = @shiftx * 15 * 3601;
@shiftz = @shiftz * 30 * 3601;
```

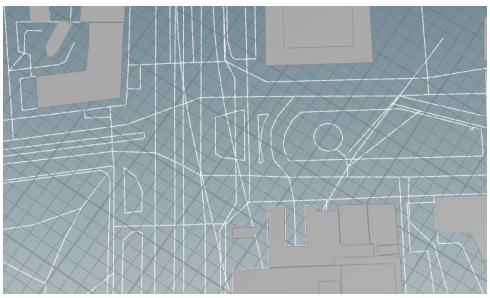
Obrázek 2.2: Příklad VEX skriptu.

Kapitola 3

Realizace

3.1 Základ mapy

Základní verze houdini se dá zdarma rozšířit o další položky SIDEFX LABS TOOLS, které obsahují například nástroje pro práci s daty OSM map, které poskytuje například již zmíněná stránka openstreetmap.org. Pro import map byl použit blok osm import. Jak imporotvaná mapa vypadá můžete vidět na obrázku 3.1 Další použitelným nástrojem, byl osm filter, který dovoluje filtrovat zvláště data z komunikací, budov a ostatních dat. Lze toho dosáhnout i za pomoci bloku Blast, který je již v základní verzi Houdini, ale rozdíl je, že osm filter, již obsahuje všechny výjimky, které by se jinak musely jednotlivě vypsat.



Obrázek 3.1: Importovaná mapa do Houdini přes OSM importer.

Dále, jelikož se projekt bude zajímat hlavně o zaplnění městského itineráře, nikoliv budovami samotnými, byl použit i modul osm building, který z dat budov dokáže vytvořit jejich bloky, pokud jsou v datech uložena informace o výšce. V případě že nejsou, může je modul buď vynechat, nebo se snažit odhadnout data na základě okolních budov. Tato verze se ukázala celkem uvěřitelná v případě solitérních budov s chybějícími daty, ovšem dost nepřesné vyjádření vzniká u úzkých a nízkých řadových budov, jako jsou například stavby jednotlivách garáží.

Posledním využitým blokem ze sady SIDEFX LABS TOOLS týkající se OSM map, bylo využití road generator, který posloužil jako základ pro vytvoření, zatím nepřesných silnic, jehož spřesnění se bude řešit dále v tomto projektu.

Dalším přidáním na realističnosti modelu je základní terén, který je vytvořen obalením silnic za pomoci funkce bound, která je umístí do vepsaného obdélníku, který byl následně extrudován do výšky 0,5 m. Dále byla extrudována kopie silnic a posunuta, aby za pomocí funkce boolean, mohla být

odečtena od terénu a vznikla tak díra v místě silnice simulující zapuštěnost do terénu ve městě (tento systém se pak změní zavedením vlastního terénu).

3.2 OSM data

Samotná tabulka s daty je dost zajímavá, hlavně v tom, kolik informací se dá z toho vytěžit. Proto se v tomto projektu pokoušíme co nejvíce pracovat přímo z daty této tabulky. Jednotlivá data, jsou vždy navázána k patřičným primitivům (čáry, nebo polygony). Samotné body obsahují pouze informace, kde se nacházejí, dokonce i vzhledem k mapě světa (tzn. obsahují i zeměpisnou délku, šířku a výšku nad mořem), ale pro zjednodušení, byl model převeden pouze do roviny, aby se práce projektu mohla soustředit více na jiné aspekty procedurálního generování. Data z budov, zatím vynecháme, protože se tato práce budov přímo netýká. Ale například ze silnic, se dá zjistit spousta informací - pokud jsou tam uložené. Například, jaké třídy, případně typu je, zda-li je osvětlení, kolik pruhů má ve které směru, zda-li má chodník a na jaké straně, jména ulic, maximální povolenou rychlost, primární využití, jednosměrnost, a mnoho dalších (obr. 3.2). Podobně jsou na tom ostatní data i budovy.

	_spe	height	hgv	highway	historic	historic_shop	history	hou_id	hou_name	hou_node	hou_way
8				primary				472230365	5. května	0	1
9				primary				472230366	5. května	0	1
10				secondary			Retrieved f	4415194	Na Pankráci	0	1
11				secondary				8880188	Lounských	0	1
12				secondary			Retrieved f	8882023		0	1
13				secondary				139573550	náměstí Hrd	0	1

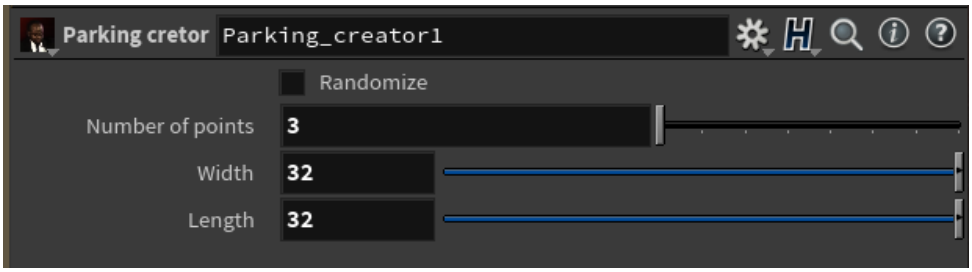
Obrázek 3.2: Příklad dat uložených v primitivech z map.

3.3 Návrh parkovišť

Nejdelším a nejkomplexnějším řešením tohoto projektu bylo umisťování aut na parkoviště. Za pomoci ostatních dat z osm filteru, bylo možné najít data týkající se pouze parkovišť, které byly vyfiltrovány za pomoci funkce Blast s nastavením "amenity=parking"(to znamená, smazali vše, co v kolonce amenity neobsahovalo vlastnost "parking").

Pro zjednodušení a otestování co největší obecnosti byl založen zvlášť projekt týkající se pouze parkoviště. Byl v něm vytvořen vlastní skript pro generování obdélníkových parkovišť, jejichž délka a šířka byla možná regulovat již uvnitř Houdini, dále vytvořena možnost naivního generování náhodného tvaru parkoviště, u kterého se dalo nastavit opět přímo v Houdini počet vrcholů (viz obr. 3.3). Generování náhodnosti probíhala vytvořením X náhodných bodů, poté jejich seřazením v listu s prioritou úhlové vzdálenosti proti směru hodinových ručiček a následném vytvoření polygonu z něj, podobného právě z OSM dat. Uspořádáním jsme se tak vyhli možnosti, že by se některé

vertexy (hrany modelové sítě) křížili a vznikal by tak non-manifold objekt (tzn. geometrie, která nemůže existovat v trojrozměrném prostoru, plochy které sebou navzájem prochází - čímž by dále v algorimtu vznikaly neočekávané chyby).



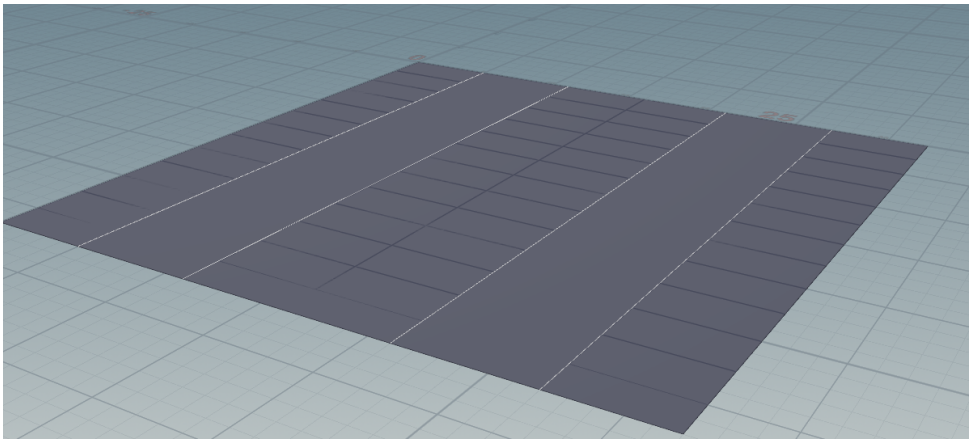
Obrázek 3.3: Vlastní doprogramované rozhraní pro ovládání generaci vlastních parkovišť pro testování.

Uvažujme, že parkoviště bude složeno nyní z parkovacích míst a přístupové cesty k němu. Pro zjednodušení jsme předpokládali, že nebude záležet na tom, kde bude přístupová cesta začínat, jen když bude vést přímo k okraji. Takové parkoviště, aby obsáhlo co nejvíce parkovacích míst, pak vypadá, že je složeno z rovnoběžných přístupových cest lemovaný parkovacími místy. Vycházeli jsme z norem ČSN [csn], které stanovují minimální velikost parkovacích míst pro kolmé parkování (v našem případě) a to konkrétně 2,5m * 5m s potřebnou šířkou přístupové cesty alespoň 6m. Tyto data byly základem pro náš algoritmus řešení.

Pro získání dalších dat bylo provedeno získání vzdáleností jednotlivých bodů od jednotlivých hran parkoviště, z nichž byla vždy vybrána maxima (nejvzdálenější bod pro každou hranu parkoviště). Pro tyto vzdálenosti, jsme se pak snažili nalézt optimální vzdálenost, kde by byl co nejmenší zbytek po našem modulu cesty a 2 parkovacích míst okolo, tzn. 2*5 + 6, tedy 16 metrů šířky modulu.

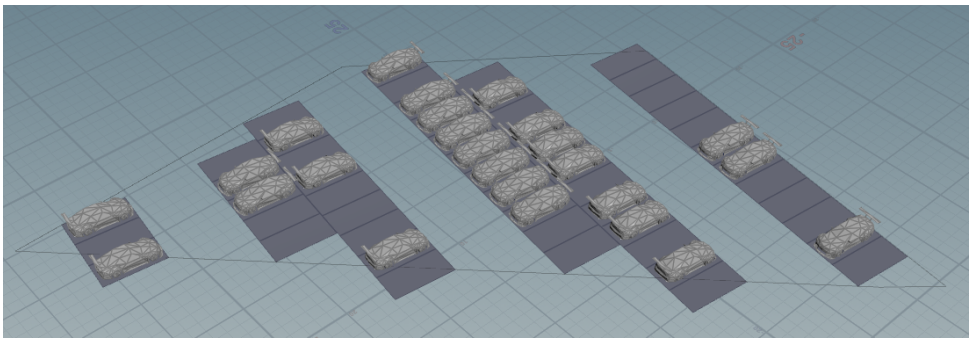
Po nalezení nejoptimálněší hrany pro vytvoření modulů kolmo k ní, jsme začali umisťovat body pro jednotlivé cesty, kde budou křížit hranici parkoviště, čímž jsme vytvořili právě geometrii příjezdových cest. Tyto hranice pak byly rozděleny podle vzdálenosti jednotlivých míst stání od sebe, na které pak geometri stání (obdélník 2,5 * 5 m) byly rozkopírovány. Natočení byly korigovány pomocí normály první hranice, ze které byly prováděny geometrické posuny kopií přímek, aby byly vygenerovány hranice cesty (viz obr. 3.4). Pro zjednodušení bylo povoleno skriptu kopírovat stání tak, aby částečně zasahovaly mimo objekt parkoviště.

Dalším objeveným problémem byla malá parkoviště, která byla obsažená v datech již bez příjezdové cesty a byla značená na mapě. Taková parkoviště měli nájezd přímo ze silnice. Proto zde již byla přidána výjimka pro parkoviště, kde maximální vzdálenost bodů od hranice, někde měla 4 - 6 metrů pro generování pouze parkovacích míst navazující na konkrétní hranici (rozsah vzdálenosti volen kvůli přesnosti map).



Obrázek 3.4: Příklad výsledného testovaného pravidelného parkoviště bez umístěných vozidel.

Následně jsou umístěné rozkopírované modely aut (autorova tvorba z předmětu VGO ze zimního semestru 2020/2021). Opět v Houdini je možnost nastavit procentuální zaplnění parkoviště. Rotace aut je vypočítána z normály první hrany geometrie parkovacího místa a následně předána jako informace o normále pro blok Houdini Copy to points, které byly vygenerovány z umístění parkovacích míst (viz obr. 3.5).



Obrázek 3.5: Příklad výsledného testovaného náhodného parkoviště s umístěním vozidel.

3.4 Návrh silnic

Již bylo zmíněno, že jsme využili základu silnic za pomocí Road generator a že OSM data obsahují spoustu užitečných informací o silnicích. Bohužel neobsahují konkrétně šířku silnice, ale obsahují počet pruhů. Opět sáhneme pro alespoň základní informace, ze kterých vycházíme, do norem ČSN. Tentokrát se týkají velikosti jízdních pruhů a silnic. ČSN normy definují opět pouze minimální velikost [Kob20]. A to 2 * 0,25 m vodící proužek s minimální

šířkou jednoho pruhu 2,5 m. Proto v následujícím skriptu bylo vždy přidáváno 0,5 m za krajnice a minimální šířka vynásobena počtem pruhů. Pokud nebyl uveden počet pruhů, pak bylo základně rozhodnuto o počtu 2 pruhů. Jelikož se ČSN nezmiňují o jiném, než minimálním rozměru, další rozměry byly odhadnuty a následně upravovány podle porovnávání Google maps a výsledného modelu, aby silnice fungovaly co nejvíce v obecných případech. Šířka silnice běžně bývá dost variabilní a má spoustu proměnných a výjimek, nicméně bylo postupováno, aby velikosti odpovídaly přesnosti v co nejvíce případech. Konkrétní šířky pruhů byly zvoleny:

- Silnice první třídy - 3,5 m
- Silnice druhé třídy - 3,0 m
- Silnice třetí třídy - 2,75 m
- Ostatní silniční komunikace - 2,5 m

Problém ovšem bylo předání dat šířky silnice pro následující blok v Houdini. Houdini totiž v dalších skriptech je schopno pouze používat proměnných, které jsou vytvořené čistě v Houdini a architektura vlasních bloků pro skript nedovoluje využít výstup jinak než jako geometrii. Jediné řešení je tak využití některých z atributů, které se budou jmenovat stejně, jako vstupní atribut následujícího bloku. V našem případě "width". Jenže width je jedna ze základních proměnných Houdini, kterou využívá u jiných objektů a i když není v konkrétním objektu používaná, je skrytá a zamknutá pro skriptování v Pythonu. Proto bylo nutné nejprve přes Houdini všem silnicím tento atribut vytvořit, ke kterému pak už lze přes Python scripting přistupovat a měnit (i když byl stejnojmenný originál skryt a uzamčen pro změny).

3.5 Návrh městského mobiliáře

3.5.1 Lampy

V programu Bledner si autor vytvořil vlastní model lampy, odpovídající zhruba rozměrům běžných lamp u chodníků. Norma ČSN se opět zmiňuje pouze o výšce umístění přístupu k opravě, nikoliv o výšce celé lampy. Dále norma zmiňuje minimální vzdálenost od vozovky a to je 0,6 m [aJK18], se kterou bylo počítáno pro umístění. Vzdálenosti opět nějsou dány, jediné co se tomu přibližuje je poznámka o "dostatečné prosvícenosti", která není dále nijak podložena. Proto byla zvolená vzdálenost jednotlivých lamp 40 m, které odpovídají zhruba běžné vzdálenosti, která se samozřejmě liší.

Pro výběr silnic bylo opět použito funkce Blast, za pomoci které jsme vyfiltrovali ty, které měly uvedeno, že jsou osvícené (tudíž, pokud jsou některé reálně osvícené a nemají tak uvedeno v OSM datech, jelikož jsou neúplná, takové silnice jsou vynechány).

3.5.2 Parkující auta v ulicích

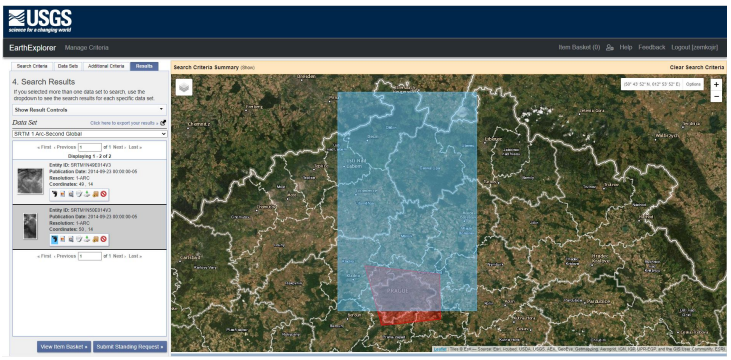
Umístění parkujících aut do ulic byl udělán podobným způsobem jako umístění lamp. Rozdílem je jiná filtrace, jiné vzdálenostní rozmístění, pozice vůči silnici a přidání parametru zaplněnosti. Opět silnice byly vyfiltrovány, aby auta byla umístěna pouze v rezidenčních oblastech (je nesmyslné, aby někdo parkoval mimo vyhrazená místa v silnicích vyšších tříd a spojů pro zásobování, kde bývají zákazy zastavení). Auta jsou pak umístěna přímo ke kraji ulice k obrubníku. Navrhováno vždy pro každou geometrii hranice silnice.

3.6 Návrh terénu

3.6.1 Zdroje potřebné ke generování

Veškeré zdroje ohledně terénu byly vybrány ze stránky USGS - earthexplorer (dostupné z [ear]), což jsou stránky státního geologického účadu USA, obdoba geoportálu ČÚZK. Po registraci je uživateli dovoleno stahovat jejich volně dostupná data, včetně satelitních snímků soukromých i NASA. Cílem bylo najít ideální satelitní snímky s daty výškových profilů pro využití ke generování terénu.

Jediná dostupná globální data byla z mise SRTM (Shuttle Radar Topography Mission [NAS]), která pokrývá vždy čtyřúhelníky jednoho stupně zemské délky a šířky s přesností na 30 metrů (viz obrázek 3.6). Existuje možnost stáhnout v různých formátech (BIL, DTEF a GEOTIFF), kde nejideálnější pro extrakci dat je GeoTiff, který pro naše účely funguje jako obyčejný tiff rastrový soubor, jen má barvu pixelu podle nadmořské výšky (v odstínech šedi). Soubor má tak 3601×3601 pixelů pokrývající tak plochu obdélníku o délce zhruba 108 kilometrů v našich zemských šířkách.



Obrázek 3.6: Webové rozhraní Earthexploreru - červený polygon značí vyhledávání dat pro vybranou oblast, zatímnco modrý stopu jednoho datového souboru.

3.6.2 Tvorba terénu

Samotná tvorba terénu začala vytvořením mřížky se čtverci 30×30 metrů, a načtení geotiffu dle UV mapy. Bohužel stažená data mají pravotočivý systém souřadnic a Houdini používá levotočivý. Došlo tedy matematickou úpravou k nápravě (nastavit měřítko - scale na -1 vůči ose z a otočení o 270 stupňů okolo osy y). Poté bylo potřeba zarovnat aktuální souřadnice gridu se staženou mapou. Víme, že vždy geotiff začíná od celého stupně, které má i ve jméně a od toho se vypočte rozdíl od minimální zemské šířky a délky, které jsou uloženy jako vlastnost primitiva OSM mapy.

Následuje ořezávání bodů pouze na oblast zobrazovaného modelu. Ořez je zvolen podle minimální a maximální velikosti zemské šířky a délky zvětšenou o 30 metrů (velikost jednoho čtverce) na každou stranu. Zbylé body gridu jsou vyzdviženy na svou výškovou úroveň podle hodnoty uložené pro jednotlivé body. Velikost se pohybuje od 0 do 1, proto je potřeba přenásobit velikostí datového prostoru dedikované jednomu pixelu - 32767 (16 bitová hloubka geotiffu).

Pro dokončení terénu se dodá objem terénu za pomoci prvku "extrude volume", kde se dá vytvořit terén se základnou v rovině, která je zvolená o 2 metry níže než je minimální nadmořská výška výřezu. Z boku je pak díky tomu patrnější průřez terénu a uživatel tak může snadněji zhodnotit jeho rozdíly.

3.6.3 Umístění prvků na terén

Pro umístění budov na terén, byly potřeba vyfiltrovat budovy z OSM mapy a vynést jejich jednotlivé hraniční body k terénu. Dále byl nalezen jejich nejnižší bodu střetu s terénem, kam byly posunuty jejich půdorysy a dále vytvořeny konkrétní budovy za pomoci již zmíněných "osm buildings". Nevýhodou je ovšem, že občasnou svažitostí terénu a předpokladu výšky budovy z dat OSM map, se může občas stát, že část budovy skončí pod terénem. Tato varianta ale byla zvolena jako vhodnější oproti tomu, že by základy budov mohly být ve vzduchu.

Stejným způsobem byly vyneseny i silnice a extrudovány 20 cm do hloubky, kde byly rozdílem od terénu získán detailnější terén i se silnicí (s rezervní extruzí nad terén o 10 cm, abychomse vyhnuli zbytečným artefaktům).

Parkoviště byla získána pak průřezem jejich projekce na terén, kde jsme byli nuceni odstranit jejich non-manifold verze z důvodu špatného zobrazování a artefaktů. Non-manifold polygony byly způsobené již špatnými daty z OSM grafu, kdy vertexy parkovišť nejsou správně seřazeny a nejsme schopni v rámci této práce je snadno zrekonstruovat vzhledem k občasné nepravidelnosti jejich tvarů. V modelu jsou tudíž zobrazeny jen téměř bezartefaktové parkoviště (s umístěním aut i do zbylých je však počítáno), které je vyzdviženo nad terén jako jeho doplnění.

Jednotlivé doplňující prvky jako lampy a auta jsou vyneseny na povrch terénu za pomocí posunu jejich kopírovacích bodů, které se zobrazují na terén raycastingem - VEX funkce intersect s nastaveným vektorem "Ray_dir"na

8449 m (velikost Mount Everestu - žádná výška terénu by neměla být větší než tato hodnota) ve směru osy y. Velikost vektoru je nutná pro tuto funkci, aby protínala směrem od bodu i terén, pouhý směr bohužel nestačí. Funkce intersect pak vrací hodnotu -1 pokud se nepovedla, jinak vrací číslo primitiva. [int]

```
vector raycast = chv("Ray_dir");
vector @hitpoint;
vector uvw;

int correct = intersect(1, @P, raycast, @hitpoint, uvw);

if(correct >= 0)
{
    @P = @hitpoint;
}
```

3.7 Optimalizace modelu

Pro ušetření paměti a urychlení cookingu modelu (vytváření a propočítávání modelu v Houdini) byla zvolena následující opatření ve smyslu redukovat co nejvíce použití funkce boolean, která je značně pomalá a některé její přesnosti se dají zanedbat.

Co se týče délky vytváření modelu, samotné načítání dat a texturování jsou ve výsledku zanedbatelné vůči délce vytváření samotného meshe (sítě modelu). Nejdéle trvají obecně jakékoliv operace s booleanem. Takže ořezávání terénu a hlavně prořezávání budov terénem, které bylo potřeba provádět pro každou zvlášť. Podobný problém i se silnicemi, kterých je ale obecně méně modelů než budov. Samotné výpočty a umisťování budov na terén tvořily přes 50% času tvorby modelu.

Ořezávání mřížky terénu bylo provedeno mazáním bodů, které jsou menší nebo větší než hranice zeměpisné délky a šířky zvětšené o 30 na každou stranu. Dále triangulace gridu před tvorbou terénu pro usnadnění práce booleanům, kdy se snáze propočítává průřez rovinným trojúhelníkem, nežli nerovinným polygonem.

Dále nahrazení co nejvíce booleanu raycastingem hraničních bodů, které byly potřeba přesunout na terén. Nejdůležitější a největší zrychlení je konkrétně u budov, kdy se přenesly raycastem pouze hraniční body (pouze body kde se střetávají jednotlivé hrany) a z nich se hledal nejnižší. Toto zjednodušení může přinést chybu, kdy hraniční bod nebude nejnižším bodem na terénu, ale budova by seděla v "dolíku"a její hrany by v určitých bodech byly níže, než hraniční body polygonu a budova tak může viset ve vzduchu. Nicméně už z praktického hlediska takových případů je naprosté minimum z důvodu nevýhody stavby takovéto budovy, kdy by dešťová voda stékala a hromadila se u paty budovy, čímž by znatelně snižovala její fyzickou životnost. Průchod algoritmu budovami pak vypadá ve VEX pseudokódu následovně:

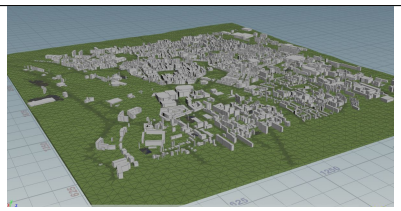
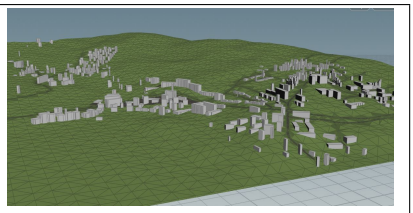
```
foreach(building)
{
    foreach(Point P)
    {
        raycast = (0, 8849, 0);

        int correct = intersect(P, raycast, &hitpoint);

        if(correct >= 0)
        {
            P = hitpoint;
        }
        adddetailattrib(minimum(P.y));
    }
    foreach(Point P)
    {
        map_height = detail("ymin");

        P.y = map_height;
    }
}
```

Zrychlení pak vypadalo naměřené na 2 následujících výřezech mapy (měřeno v programu Houdini) viz tabulka 3.1. Testování proběhlo na osobním notebooku s procesorem Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz, přepínatelnou grafikou NVIDIA GeForce 1060 with Max-Q Design a 16,0 GB paměti RAM (15,8 GB použitelných)

		
Obec	České Budějovice	Lipno nad Vltavou
Rozměry km^2	5,3	3,3
Počet budov	4020	531
neoptimalizováno	cca 40 minut	1:22 min
optimalizace pouze triangulací	4:32 min	0:27 min
optimalizováno	0:51 min	0:09 min

Tabulka 3.1: Porovnání 2 testovacích modelů a jejich doby vytváření bez optimalizace, s částečnou a plnou.

Kapitola 4

Texturování

4.1 Zdroje texturování

Pro fotorealističnost bylo potřeba opět zvolit vhodné textury. Co se týče globálního hlediska, nejpřesnější volně sehnatelné snímky země jsou z mise satelitu Sentinel 2 (opět dostupné z earthexplorer). Snímků dané oblasti je vždy několik, a zastínění oblačností dosahuje minim i pod 10% (což celkem využitelné je v kobminaci s ostatními snímky). Hlavní nevýhodou je však jeho přesnost, který dosahuje maximálních hodnot 10 m (jeden pixel je 10×10 metrů), což není úplně vhodné pro texturování v porovnání šířkou jízdního pruhu silnice, který se pohybuje od 2,5 metru.

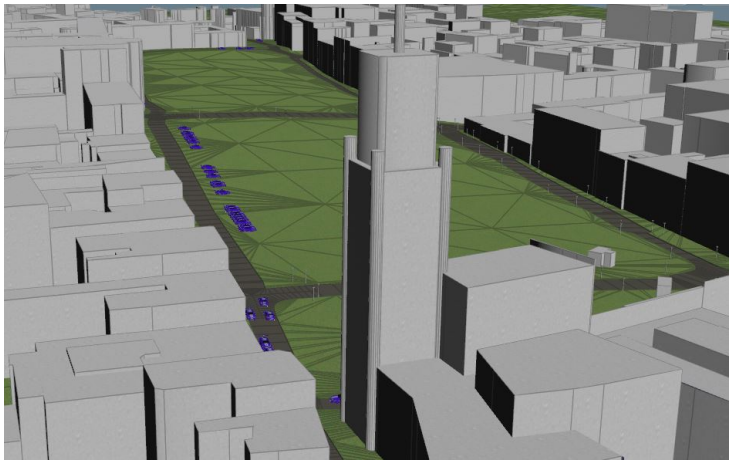
Z tohoto důvodu bylo potřeba zvolit snímky podrobnější, a to letecké. Ideální volně dostupné snímky jsou například na geoportálu ČUZK. Jejich přesnost je až 0,15 m (což je pro naše případy výhodné). Tímto se však budeme muset lokalizovat dále pouze pro Českou republiku. Geoprotál dovoluje stahovat opět výřezy podobně jako OSM mapy v rastrových souborech, s možností zvolení velikosti obrázku a stáhnutím geolokačního souboru (který usnadní umisťování textury a její automatizaci). Problém ovšem je, že Česká republika používá souřadný systém S-JTSK, zatímco dosavadní zdroje byly vázány na světový souřadný systém WGS84, na který bude třeba provést transformaci.

4.2 Základní textura

Byla vytvořena základní textura pro globální případy, kde by se pak nedala použít pro jiné případy než České republiky. Zdroje textur jsou uvedeny v textovém dokumentu ve složce s texturami. Jediná vlastní textura je pro auta, která je zhotovena i s modelem jako semestrální práce z předmětu VGO autorem bakalářské práce.

Vše bylo potřeba i procedurálně namapovat, což Houdini také dovoluje. Pro terén a silnice šlo použít zjednodušené UV mapování, kdy byla textura promítnuta na rovinu XZ. Pro uv mapping budov pro pocit zvolené pokračující textury betonu byla použita metoda "uv unwrap", která byla provedena automaticky přes 6 promítacích rovin (kolmé na sebe ze všech stran). Lampy

byly mapovány cylindrickým uvunwrap a model auta již své mapování měl ze semestrální práce připravené (viz obrázek 4.1).



Obrázek 4.1: Základní texturování - Karlovo náměstí, pohled ze severní strany od Novoměstské radnice.

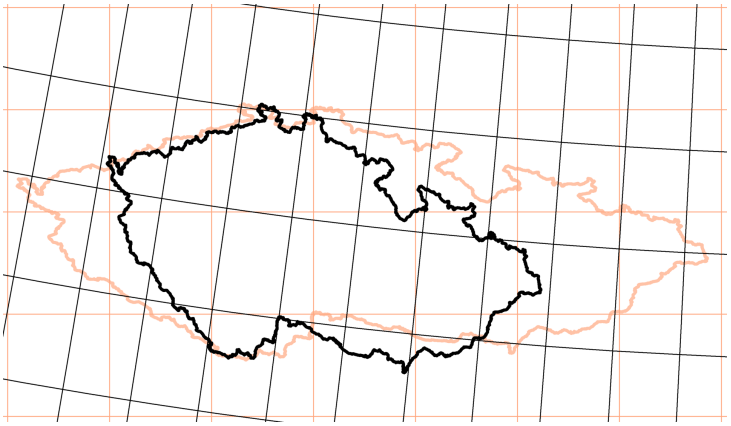
4.3 Použité souřadné systémy

4.3.1 Světový geodetický systém 1984

WGS (world geodetic system) je vydán ministerstvem obrany USA a je celosvětově uznávaný standard pro navigování v satelitů, kartografii a geodésii. Aktuálně se používá verze z roku 1984 s kódem EPSG:4326 (předchozí verze 72, 66 a 60). Je zobrazován jako čtvercová síť promítnutá na geoid (EGM84, odchylka od WGS eplipsoidu o -105 až 85 m) s referencí počátku jako nultého poledníku se středem v centru Země (s možnou odchylkou 2 cm). [WF22b]

4.3.2 Křovákovo zobrazení

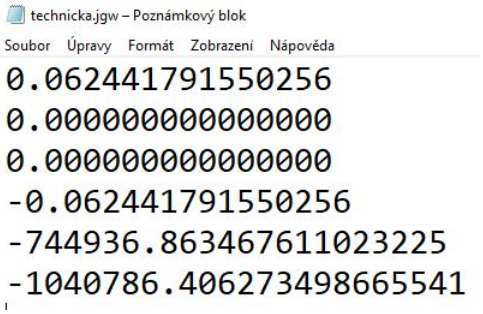
Zobrazení, které používá souřadný systém S-JTSK, bylo vytvořeno kartografem Josefem Křovákem v roce 1922. Jedná se o konformní ekvidistatní zobrazení (zachovává délky) pro Českou republiku a Slovensko. Exaktní Křovákovo zobrazení je složené a skládá se ze zobrazení Besselova elipsoidu na Gaussovu kouli a zobrazení z této koule na plochu kužele. Navíc s posunem severu tímto způsobem se stává naprosto ojedinělým typem zobrazení ve světě a proto ho většina GIS software ani nepoužívá [GT08]. Proto jeho převody se provádějí ve většině případů pouze aproximačně, protože naprosto přesný převod plochy mapy je téměř nemožný. Pro Českou republiku je ovšem tento systém závazný pro všechna státní mapová díla z důvodu vyšší přesnosti (obrázek 4.2), ač se nepoužívá např. ve školních atlasech (kvůli zmíněné odchylce severu). [WF22a]



Obrázek 4.2: Česká republika ve Křovákově zobrazení (černě) v porovnání s WGS84 (oranžově).[WF22a]

4.4 Texturování z leteckých snímků

Nejideálnější co se týče možných rozměrů, bylo zvoleno stahování souborů leteckých snímků s rozlišením 9000 na 9000 pixelů (velikost 27,8 MB, o moc větší soubory se nedaly volně z geoportálu stahovat) s nabízeným rozlišením 1:273 (což pokrývá oblast cca 500 na 500 metrů - nejlepší vyzkoušený poměr velikost plochy a rozlišení). S daným souborem byl stažen i lokalizační soubor (k formátu jpg se druží jgw (viz obrázek 4.3)).



Obrázek 4.3: Příklad souboru jgw - 1. řádek určuje inkrementaci ve směru x v souřadném systému o jeden pixel, 2. a 3. řádek otočení (ve většině map není), 4. řádek inkrementaci ve směru y v souřadném systému o jeden pixel (zde je vidět záporná hodnota, tedy obrací směr inkrementace) a 5. a 6. řádek určuje x a y souřadnici levého horního rohu mapy. [io17]

Problémem bylo převádění souřadnic z S-JTSK do systému WGS, nicméně existuje pythonová knihovna "pyproj", která slouží geografům pro rozšíření pro různé programy, která mimo jiné má transformaci souřadnic mezi různými systémy [spo22]. Problémem je, že Houdini přichází se svým vlastním nainstalovaným Pythonem a nepoužívá jazyk systému. Navíc existuje několik aktuální verzí Houdini, které používají různé verze Pythonu (2.7 nebo 3.7), na které si musí uživatel dát pozor, pokud chce stahovat aktuálnější knihovny. Pro houdini se však dá stáhnou vlastní pip (Conda nelze úspěšně použít) přes jeho terminál a musí se instalovat mírně jiným způsobem, než je doporučeno v dokumentaci knihovny pyproj.

Pro zjednodušení, vzhledem že už s jistými nepřesnosti pracujeme kvůli použitým souřadným systémům, budeme předpokládat, že po transformo-

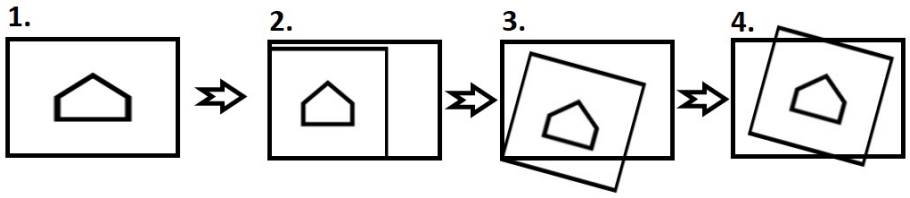
vání mapy z S-JTSK na systém WGS bude mapa stále obdélníková a pouze změněná velikost a otočení (jinak bychom pracovali s nepravidelným čtyřúhelníkem). Díky tomu budeme moci využít některých společných vlastností, které mají mapy a texturové mapy. Můžeme tak promítnout letecký snímek přímo do terénu a mírně pak upravit UV mapování, které lze jednoduše měnit pouze, pokud pracujeme s obdélníkovou texturovací mapou.

První hlavní výhoda je, že textury se mapují od levého spodního rohu (zde se nachází souřadnice [0,0] v texturovacích souřadnicích). Vzhledem k tomu, že inkrementace ve směru y leteckého snímku je záporná (viz obrázek 4.3), bude základní souřadnice uvedená v souboru také levým dolním rohem, ač základně bývá rohem horním. [io17].

Ovšem pokud chceme texturovat terén, známe souřadnice WGS pouze u originální mapy. Jelikož je terén o trochu rozšířený oproti původní mapě, musíme to brát v potaz ve výpočtu a pokud budeme převádět souřadnice leteckých snímků na systém WGS, musíme ho porovnávat s upravenou velikostí mapy a posunutým levým rohem.

Z danými zjednodušenými předpoklady po počátečním otočení UV mapy stejně jako u terénu (270 stupňů vůči ose y a zrcadlení v ose z pro prohození součadných os) z důvodu souřadného systému originálních OSM map, můžeme opět upravit UV mapu dalším nodem Houdini "uvtransform". Pro další zjednodušení, předpokládáme, že vybraný letecký snímek z většiny pokrývá oblast terénu.

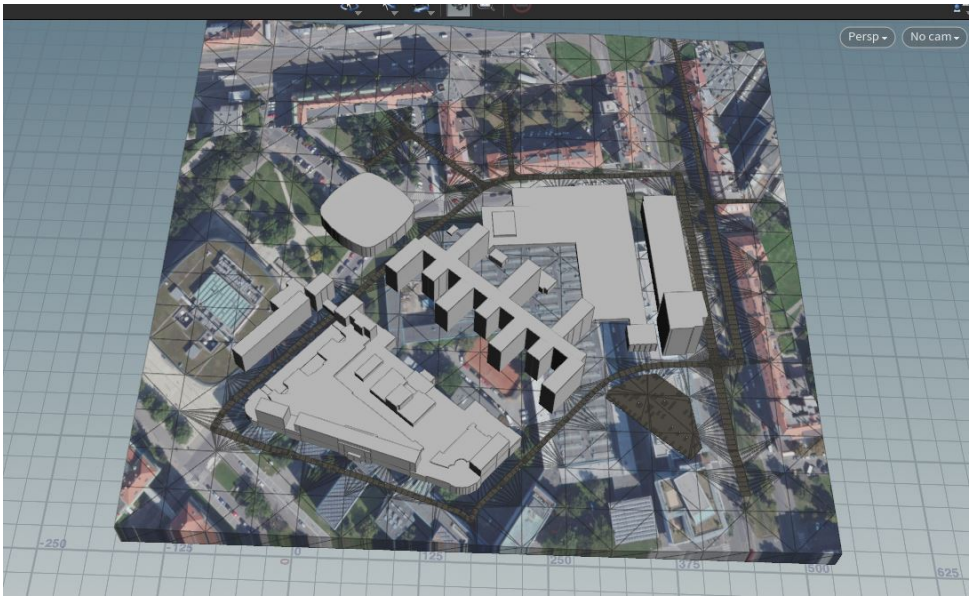
Pro výpočet použijeme 3 převedené body z S-JTSK do WGS souřadnic a to originální levý dolní roh, levý horní a pravý dolní (po inkrementaci ze souboru). Z těchto bodů zjistíme snadno velikost uvažovaného obdélníku mapy a tudíž i poměr pro zvětšení/zmenšení textury. Dále mezi levým dolním a levým horním rohem můžeme za pomoci goniometrických funkcí získat odchylku úhlu vůči poledníku (zvolená funkce arcustangens pro menší chybovost zaokrouhlováním velikostí). A nakonec posun textury mezi levým dolním rohem leteckého snímku a levým dolním rohem mapy, po úpravě velikosti vzheledem k rozšíření na terén (reftextura). Pokud je terén větší než záběr, je automaticky necháno opakování textury. Přibližný postup transformace textury je vidět z obrázku (schéma na obrázku4.4).



Obrázek 4.4: Schéma transformace načtené textury: 1. načtená textura 2. upravená velikost textury z poměru velikosti snímku a mapy na terénu 3. otočení textury o odchylku 4. posunutí o vzdálenost levých rohů textury a mapy na terénu.

Bohužel celý postup takto ztroskotá na špatném předpokladu správnosti

stažených dat. U stažených OSM dat nekoresponduje naprosto přesně jejich maxima a minima zemských šířek a délek s velikostí vytvořených dat funkcí OSM import. Krom toho, že jsme některé věci vyfiltrovali (jako jsou vodní toky, značené pozemky, atp.), tak ani tak přesně nesedí poměr původní mapy co se týče velikosti šířky a délky vůči velikosti modelu mapy v meterech. Takže nedokážeme získat pokaždé správný poměr velikostí textury a mapy a jediné co by bylo správně, je pouze vypočtený úhel otočení, protože ten nezávisel na původní mapě, pokud byla správně natočená (viz obrázek 4.5).



Obrázek 4.5: Příklad texturování s pužitím popsané transformace, kde viditelně nesedí měřítko.

Kapitola 5
Výsledky

5.1 Přehled a měřítko výsledků

Pro co nejpřesnější a nejobecnější ověření správnosti vytváření modelu bylo použito větší množství map (35 map bez vyloučených pro nedostatek dat). Mapy byly vybírány jednak pro známost autora, aby se co nejsnadněji potvrdila správnost braných referencí. Další výběry map probíhaly podle důležitosti konkrétních bodů, které by mohly tvořit zajímavé detaily a výjimky, kde by se ověřila funkčnost generování na neobvyklých případech.

Co se týče velikosti modelů, jedná se vždy o části čtvrtí městských celků, případně i celých vesnic, pokud je obec menší velikosti. Pro ověřování funkčnosti textur z leteckych snímků byly voleny velikosti map v rámci pár bloků. Podrobnější data pro porovnání u význačnějších map najdete v tabulce 5.1. Pro lepší vizuální srovnání je zde příloha s obrázky viz tabulka B.1 (v tabulce jsou dodány i jména souborů a minimální zemské šířky a délky pro jednoduchou přehlednost při otevírání projektu).

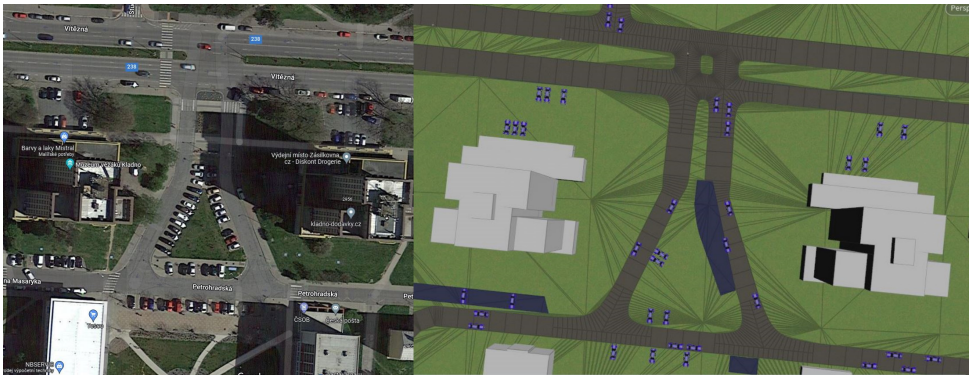
Tabulka 5.1: Tabulka statistik vybraných modelů.

Název celku	název souboru (*.osm)	zeměpisná šířka a délka	rozměry (m)	převýšení (m)	počet budov	charakter
Ústí nad Labem	Aussig	50°14°	2340x3134	272	5014	kombinace různých zástaveb
České Budějovice	Budweiss	48°14°	2449x2161	45	4020	centrum a okolí
Černošice	cernosice	49°14°	1980x1873	169	753	volná zástavba
Kampus ČVUT Dejvice	dejvice	50°14°	756x864	48	224	rovnoměrný mobiliář

Název celku	název souboru (*.osm)	zeměpisná šířka a délka	rozměry (m)	převýšení (m)	počet budov	charakter
Holešovice (Praha)	holesovice	50°14°	1296x1152	31	710	moderní urbanistické plánování
Chrast	chrast	49°15°	3206x4252	78	1315	vesnice
Nové Město (Praha)	karlak	50°14°	1332x1404	74	1187	terén okolí Karlova náměstí
Kladno	Kladen	50°14°	1908x2269	87	3381	kombinace různých zástaveb
Klenčí pod Čerchovem	klenci	49°12°	3997x4395	445	587	vesnice ve svahu
Klíše (Ústí nad Labem)	klise	50°14°	1115x880	168	193	bloková zástavba čtvrti
Linec	Linz	48°12°	1404x1728	29	1366	různé typy silnic
Lipno nad Vltavou	Lipno	48°14°	2089x1585	190	531	obec s kopcem u vodní plochy
Pardubice	Pardubitz	50°15°	2593x3314	27	3778	kombinace různých zástaveb
Pražského povstání (Praha)	podoli	50°14°	1152x1296	78	831	různé typy silnice
Poruba (Ostrava)	Poruba	49°18°	1944x3387	78	2155	moderní urbanistické plánování
Vídeň	Wien	48°16°	2161x1801	67	2053	historické centrum s Ringstrasse
Velká Dobrá	Dobra	50°14°	936x1621	54	312	Obec s širokými silnicemi

5.2 Porovnání výsledků

Nyní si můžeme předvést výsledky na konkrétních vybraných různých městských situacích. Detaily byly vybrány z větších modelů, ze kterých by nebyla patrná přesnost bez zaměření na následující detaily. Na obrázku 5.1 je vidět celkem vysoká podobnost co se týče umístění parkovišť a směrů silnic. Rozdíl můžeme nalézt v křižovatce, kde by si náš nástroj vytvořil ostrůvek, jelikož jsou jednotlivé směry vedeny v mapách jako odlišné vozovky. Takovýto ostrůvek se vynechává z důvodu šetření materiálu a zbytečných přidávání překážek do vozovky. Nicméně dále jsou souběžné pruhy odděleny travními ostrůvky, což odpovídá i v modelu.

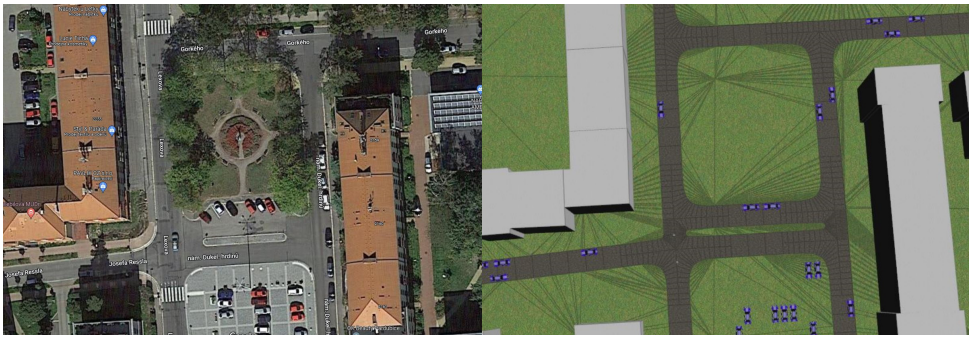


Obrázek 5.1: Příklad křižovatky v panelové solitérní zástavbě Kladně (vlevo google maps, vpravo model).

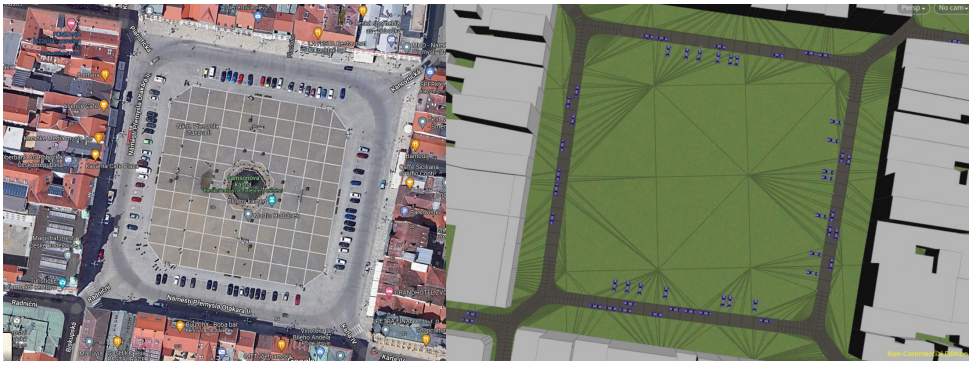
Jako druhý příklad si uvedeme blokovou zástavbu města Pardubice. Vybraný obrázek 5.2 zobrazuje náměstí s další částí přeměněnou na parkoviště. Parkoviště sice vypadá situačně odlišně, nicméně algoritmus dokázal vytvořit ještě úspornější parkoviště, než se zde aktuálně nachází. V tomto případě přesnější algoritmus než skutečnost bohužel nedokáže nahradit tzv. lidský faktor, kdy dojde k mírnému chaotickému generování na základě toho, že neměli dána naprosto pevná pravidla.

Příkladem hustého zabydlení, kde se hůře dimenzují silnice jsou např. historická centra, která byla dimenzována co nejužší pro obranu a kolikrát pro jednosměrný provoz vozů s dominujícími pěšími zónami. Takový situační příklad zde podporují České budějovice (viz obrázek 5.3), které mají i uvnitř centra plánované parkování. Povšimněte si, že jak v realitě, tak i v modelu jsou parkovací místa nedostačující a auta začínají stavět i v ulicích. Algoritmus projektu bohužel nemá rozlišování přechodu parkoviště a silnice, tudíž jsou auta v modelu občas i u parkoviště (což jako zastavení zakázané není). Na příkladu je zrovna vidět absence parkovišť, jelikož jejich základní model byl považován za poškozený a tak zahozen pro vykreslování. I tak jsou zde zaparkovaná auta.

Dalším pěkným příkladem hustého zabydlení je příklad Vídně, kde se jim povedlo vytvořit i více normové silnice už díky plánování se širšími ulicemi



Obrázek 5.2: Příklad náměstí v Pardubicích v blokové zástavbě (vlevo google maps, vpravo model).



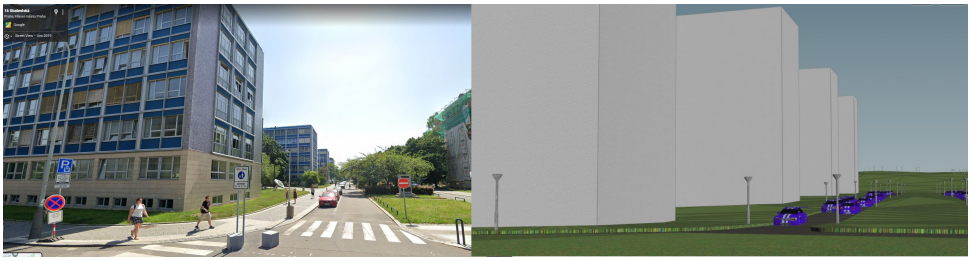
Obrázek 5.3: Příklad parkování v Českých Budějovicích v historické zástavbě (vlevo google maps, vpravo model).

(viz obrázek 5.4).

Příkladem pro ukázání pohledu z první osoby a aby byly vidět lampy (které shora tolik nevynikají), je zde příklad z Dejvic při pohledu na Strojní a Elektrotechnickou fakultu ČVUT (viz obr. 5.5). Design lamp sice neodpovídá, navíc v každé ulici je různý, tak jeho výška. Čeho si můžeme povšimnout, že jejich umístění překvapivě celkem přesně odpovídá, i když se jednalo v algoritmu pouze o obecný odhad na základě jiných měst (kde nebyla zaručená žádnými normami konzistence vzdáleností). Rozdílem je také, že v originále jsou světla umístěná pouze na jedné straně vozovky, kdy model počítá osvětlením obou stran (zde se předpokládá důvod umístění zeleně, kde by pak osvětlení dopadalo pouze do korun stromů a bylo by zbytečné) a chybějící zeleň, které dosud nebyla součástí projektu. Dále v originálních mapách chyběla data o parkovacích pruzích (nejspíše v datech jsou označena pouze veřejná parkoviště), nicméně došlo k vyplnění ulice zaparkovanými auty na základě umisťování aut mimo parkoviště.

Následujícím příkladem bychom si ukázali přechody mezi různými třídami silnic a změnách šířek na základě pruhů. K tomu nám poslouží oblast Pražského povstání v Praze (viz obr. 5.6). Zde se mísí hned několik třídních typů

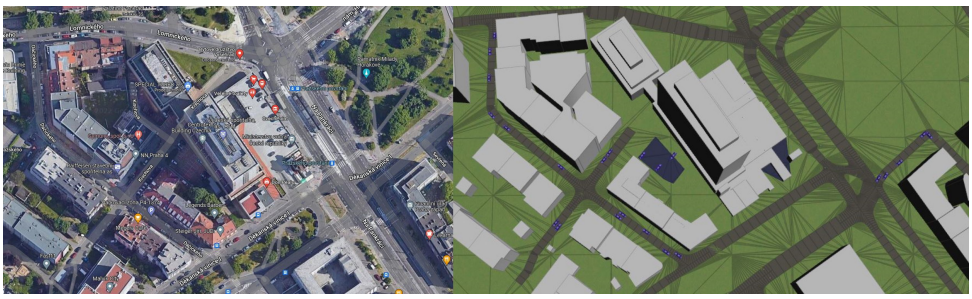
Obrázek 5.4: Příklad historické zástavby ve Vídni (vlevo google maps, vpravo model).



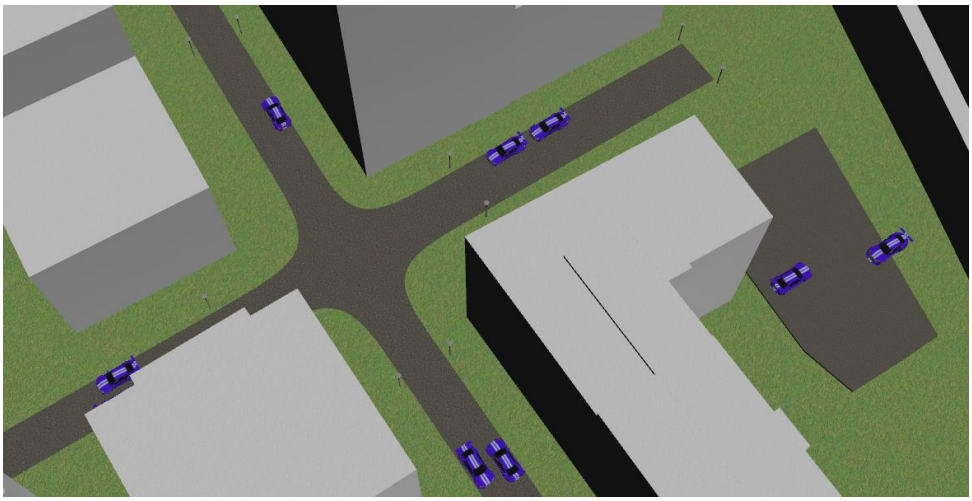
Obrázek 5.5: Příklad použití lamp osvětlení v Dejvicích, ČVUT, Praha (vlevo google maps, vpravo model).

silnic a je vidět, že i auta parkují víceméně pouze v rezidenčních oblastech (viz detail na obrázku 5.7) jak v reálu tak v modelu. Ve zmíněném detailu je opět k shlédnnutí i umístění lamp, které jsou z výšky lehce přehlédnutelné. Viditelně opět chybí některá označní parkovišť, kdy v Praze je parkování v těchto oblastech děleno soukromě na barevné zóny. Na obrázku je dále vidět, jak se (obzvláště v horní části) mění šířka silnice na základě počtu pruhů v OSM datech a tak se snaží co nejvíce přiblížit k reálu.

Pro příklad terénu byla vybrána obec Klenčí pod Čerchovem, jenž se nachází celá v kopci. Můžeme tady porovnávat chování budov ve svahu (viz obrázek 5.8) - kde může nastávat při prudším svahu, že většina budovy je v zemi "utopená". Může se stát, jelikož výška budovy je většinou brána vůči hlavnímu vchodu, pokud je tedy správně a není odhadnutá osm Buildings. Automaticky bylo bráno že budova nesmí levitovat nad zemí, takže s výškou z dat se počítá jako od nejnižšího bodu terénu, kde se dotýká. Na tomto příkladu je vidět i možnost tvorby horizontu při větším obsáhnutí terénu (viz obrázek 5.9).



Obrázek 5.6: Příklad různých tříd silnic na Pražského povstání, Praha (vlevo google maps, vpravo model).

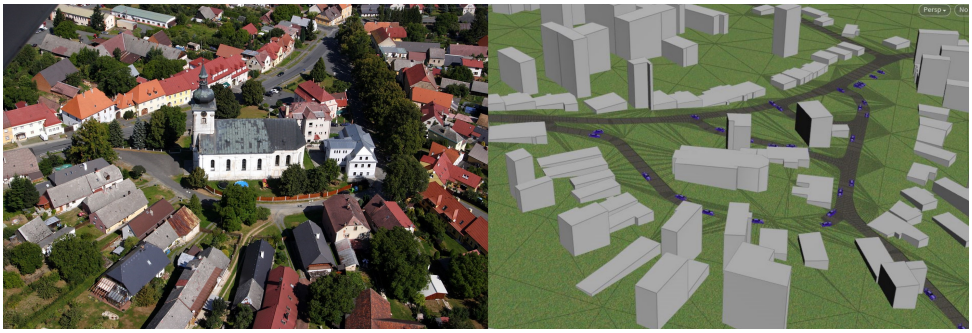


Obrázek 5.7: Detail rezidenční oblasti Pražského povstání, Praha.

5.3 Limitace

První limitací je omezený počet zdrojů pro tutoriály a troubleshooting Houdini, kdy neexistuje až tolik široká veřejnost, která by se tímto programem zabývala (ač za poslední rok znatelně vyrostla). Hlavními uživateli jsou právě profesionální firmy, které si své postupy střeží. Existuje pár tutoriálů dokonce od tvůrců Houdini, které jsou na obeznámení k začátku, případně některých jejich rozšíření, ale nejdou příliš do hloubky a uživatele nechávají dále prozkoumávat samotného, maximálně z pomocí velice obsažných manuálů.

Další limitací Houdini je přechod z grafického prostředí do Python skriptování, kde se navzájem nesdílí proměnné (tzn. některé jsou k dosažení pouze z Pythonu a některé pouze z grafického prostředí). Příkladem jsou i nesmyslně zamčené atributy, které neexistují z pohledu grafického prostředí, a jsou skryté pro Python prostředí, což se dá obcházet již zmíněným vytvářených stejnojmenných atributů přes grafické prostředí (což v Pythonu nešlo, jelikož údajně již existovaly). Proto je potřeba disponovat znalostí jmen konkrétních



Obrázek 5.8: Klenčí pod Čerchovem, vlevo letecký snímek ([kle17]), vpravo model.

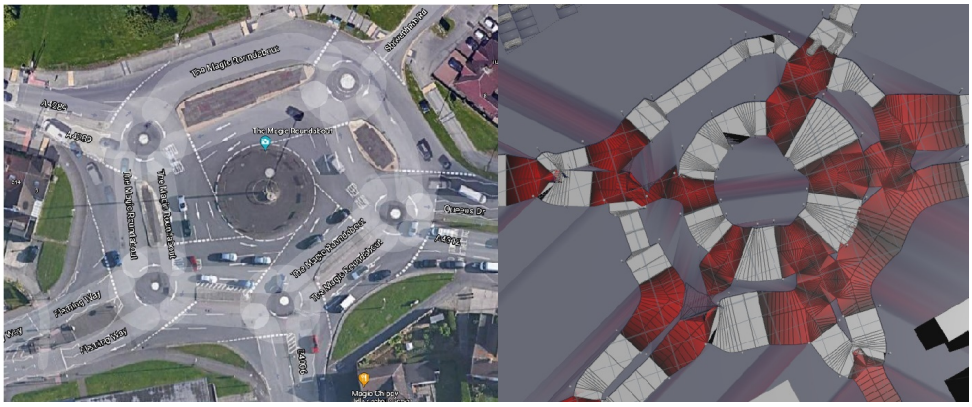
Obrázek 5.9: Klenčí pod Čerchovem - obec ve svahu.

Houdinovských proměnných, které někde existují a používají se a přes které lze předávat informace ze skriptů do grafického rozhraní Houdini (což se dozvíte pouze z manuálů SOPS od autorů, pokud existují, nebo uživatelů z fór, které je náhodou dokázali nalézt). Avšak ve většině případů je používání skriptování spíše výhodou než nevýhodou vyjma pár výjimek.

Důležité je si také dávat pozor při programování skriptů. Pokud docílíte určitých typů chyb v Pythonu, může se problém přenést dále do projektu a způsobovat spadnutí celého programu, který se ukázal dále neopravitelný, jelikož poslední úprava skriptu je pevně spjata s projektem a při načítání vždy načítá starou verzi, ač jste ji mohli mezitím upravit. Proto je doporučeno při postupech se skriptováním často zálohovat.

Limitace spočívá i v samotných čerpaných OSM mapách, kde občas některá data chybějí, takže se pak musí odhadovat, nebo s nimi nepočítat. Nebo jsou data dokonce zanesená špatně, pak vznikají "preclíkovité"křižovatky, nebo s atributy jinak, než je to ve skutečnosti, viz příklad z Anglie se složitým kruhovým objezdem (viz obrázek 5.10), který střídal různě počty pruhů, čímž způsoboval překrývání a artefakty při tvorbě silnic a terénu.

Problém není jen v silnicích ale i v chybějících datech, nebo jejich zkreslení a nepřesnosti (viz obrázek 5.11). Jak již v předchozí kapitole bylo popsáno,

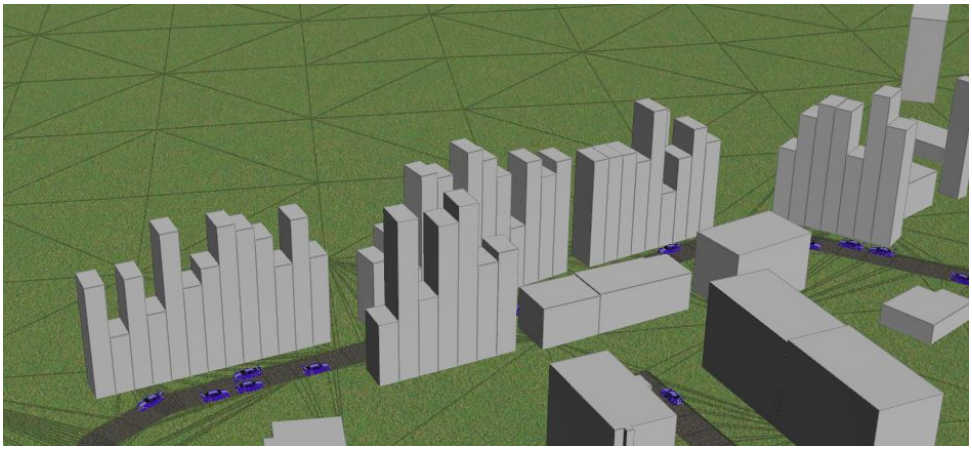


Obrázek 5.10: Kruhový objezd Magic Roundabout, Swindon, Anglie (vlevo google maps, vpravo model).

hranice zemských šířek a délek neodpovídají hranicím načteným a nejde pak přesně a správně spočítat případná odchylka nebo velikost pro kombinaci jiných systémů (ať když máme terén, který nebude naprosto přesně odpovídat, tak i texturování, které tím naprosto selhalo).

Obrázek 5.11: Příklad chybějících dat na obci Durmanec (Chorvatsko), kde vlevo vidíme vygenerovaný model z OSM dat a vpravo pohled Google Streetview, kde je jednoznačně více budov, než OSM data obsahovala).

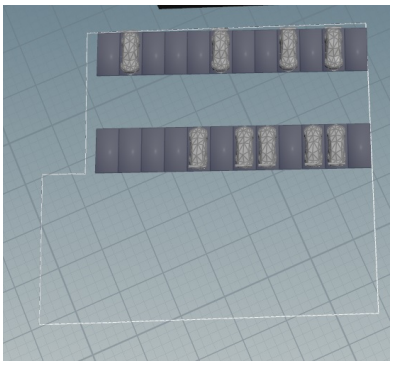
Omezením jsou i použité funkce z Houdini, které nejsou vždy přesné, ale jejich přesná implementace by zabrala více času, která není cílem této práce. Příkladem je tak odhadování výšky budov, které několikanásobně převyšuje okolní budovy na základě toho, že si vybere za výšku náhodné číslo. Nejlepší příklady jsou vždy vidět na garážových budovách, které mají být stejně vysoké, avšak chybějící data z nich dovolí generovat chaotické bloky (viz obrázek 5.12).



Obrázek 5.12: Příklad špatně zvolených výšek budov u sousedících objektů.

5.4 Diskuze

Problémy autorem vytvořeného algoritmu mohou vznikat, kdy parkoviště se podobá tavru L (viz obr 5.13), po té, vzdálenost bodu prostřední hrany L může mít vzdálenosti od nejzažších bodů hranice, které mají ovšem různé směry. Pokud pak nejdelší vzdálenost z těchto dvou směrů je vyhodnocena algoritmem jako nejoptimálnější, parkoviště se doplní pouze v daném směru. Tento problém by se dal pak ošetřit součtem největších vzdáleností v obou směrech a následným přidáním výjimky, aby se generovalo i ve směru druhém. Problém nebyl zatím opraven, jelikož po přenesení do map, se tento problém objevoval ve výjimečných případech.



Obrázek 5.13: Špatné řešení parkoviště ve tvaru L, Poruba, Ostrava.

Nepřesnost v parkovištích se může stát i v případě neobvyklých parkovišť, které nejsou pravoúhlé a jsou nepravidelné - tehdy algoritmus pro vytváření uvažuje pouze jeho konvexní obal, protože kontrolovat hranice složitějších parkovišť by značně přidalo na složitosti podmínek kontrolující výjimky, které nejsou tak obvyklé.

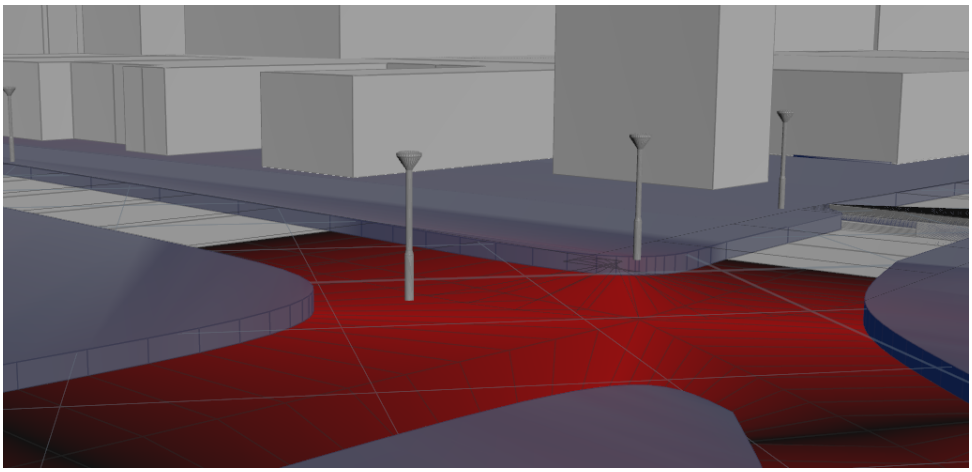
Problém může vznikat v případě u složitějších křížení silnic, případně mimoběžných, kdy se vytváří špatně nebo vůbec žádné křižovatky v Road generatoru a Boolean nefunguje správně a dochází výjimečně tak i k absenci terénu nebo špatnému vyřezání děr do něj, jak již bylo zmíněno v předchozí části.

Problém vzniká i s výpočtem šířky silnic na základě počtu pruhů. Samozřejmě občas během spojování pruhů se v běžné praxi dělá šrafovaný pruh, nebo se rozšíří zbylé, takže šířka vozovky zůstává stejná, ale takové věci

nejsme schopni bez dostatečných dat pokrýt. V případě úzkých historických uliček tak dochází i ke kolizi budov se silnicemi, které zde nejsou a nemohou být podle norem.

Vzniklým problémem u lamp je fakt, že některé ze silnic jsou vynechány jelikož nejsou osvětleny a tak nevzniká křižovatka. V některých případech se pak mohou objevovat lampy uprostřed křižovatek (což by se mohlo pouze opravit, pokud budeme vycházet z jiných geometrií objektu, případně jejich ořezáním - viz obr. 5.14).

Opět vznikají problémy, kdy hranice typů silnic může vyústit k generování auta do křižovatky, nicméně kvůli indexu zaplněnosti to nebývá tak časté a může to suplovat pohybující se auto.



Obrázek 5.14: Příklad, kde se kříží osvětlená a neosvětlená ulice, kde umisťování lamp probíhá nekorektně (dvě ulice směrem k nám jsou neosvětlené a směrem dopředu jsou 2 osvětlené).

Co se týče terénu, jeho přesnost je značně omezená vůči tomu, v jakých zemských šířkách se pohybujeme. Je tak možné, že některé terény u okrajů souborů geotiffu nebudou tak přesně umístěné a možná i různě roztažené - přeci jen nahrazujeme obdélník za čtyřúhelník promítnutý na kouli, kdy se jedná o neeuklidovský prostor. Navíc čím blíže k pólům se pohybujeme, tím k většímu zkreslení bude docházet. Je otázkou, zda-li a do jaké míry by stálo za úvahu zpřesnit výpočet deformace, když kombinujeme upravené snímky terénu a mapové zobrazení s různou mírou deformace.

S terénem a jeho optimalizací pak došlo i k dalším problémům a to konkrétně při triangulaci povrchu se občas stává, že dojde k prohození pořadí vertexů polygonů, čímž dojde k invertování. Následující úpravy pak pracují s invertovanou sítí modelu a funkce boolean nefungují pak správně - vznikají tak "invertovaná"parkoviště a terény (parkoviště je všude, kromě místa, kde by mělo být). Autor sice dodal funkce pro obrácení vertexů, ale nebyl schopen detekovat, za jakých podmínek k takovéto inverzi dochází. Záleží mapa od mapy. Dalším námětem práce by rozhodně stálo za úvahu se tímto tématem zabývat a opravu zautomatizovat.

Kapitola 6

Závěr

6.1 Zhodnocení

Cílem práce bylo vytvořit co nejobecnější nástroj pro tvorbu modelu města zaměřeného na detaily a obecnost kolem silnic. Důraz při tom byl také kladen na open source zdroje. Naprosté dokonalosti reprodukce do reálu se nedosáhlo, což ani s dostupnými daty nelze. Nicméně výsledky jsou ve většině uvedených a vyzkoušených situačních případech velice blízko realitě (ze zadání minimálně 3 modely, konkrétně ověřeno na 35 mapách), což splnilo stanovené cíle. Houdini jako nástroj k realizaci poskytuje spoustu možností, obzvlášť s rošířením skriptování, nicméně je jich příliš mnoho, aby je autor za tak krátkou dobu pokryl a zajistil naprosto vyhovující.

6.2 Směřování další práce

Práce byla připravována, aby se na ní dalo navázat a rozšířit komplexicitu jejího dosahu. Jedním z přání autora bylo například vytvoření heatmapy provozu na základě dat o budovách z OSM, které mohou o tomto napovědět, případně v závislosti na denní době nebo indexu tranzitu. Na základě těchto heatmap by bylo možné měnit indexy zaplněnosti aut v ulicích a na parkovištích.

Dále by je v plánu upravit některé algoritmy, aby dosáhly větší realističnosti.

A OSM data skrývají další spoustu možností, jak rozšířit mobiliář města o zeleň, hřiště, značky a změnu textur silnic podle povrchů, případně podle umístění parkovišť, nebo nájezdů.

Rozhodně by stálo za úvahu i složitější texturování terénu co se týče globálního hlediska, lokální snímky se z důvodu kombinaci systémů ukázaly jako nemožné, co se týče požadované přesnosti.

Literatura

[aJK18] Ing. Ilona Mušálková a Jiří Kotas. Standardy pro zařízení veřejného osvětlení, Čez energetické služby, s.r.o. dostupné online z: https://www.cez.cz/edee/content/file-other/cezes/nase-sluzby/verejne-osvetleni/standardy-vo_cez-energeticke-sluzby_final.pdf, 2018.

[AVB08] D. G. Aliaga, C. A. Vanegas, and B. Benes. Interactive example-based urban layout synthesis. *ACM SIGGRAPH Asia 2008 Papers*, 2008.

[csn] Čsn 73 6056, příloha b normativní, základní rozměry vozidel a odstupy od překážek, tabulka 6. dostupné online z: https://www.fce.vutbr.cz/KDK/hron.l/Garaze/Dispozice_tabulky.pdf.

[ear] Usgs earthexplorer. [online] https://earthexplorer.usgs.gov/.

[fro20] The effects of frozen 2. [online] https://www.sidefx.com/community/frozen-ii/, 2020.

[GT08] Petr Urban Gábor Timár. Aproximace křovákova zobrazení pro území České republiky lambertovým konformním kuželovým zobrazením pro potřeby gis*. [online] http://sas2.elte.hu/tg/krovakpaper_cz.htm, 2008.

[Houa] SideFX Houdini. hou package. [online] https://www.sidefx.com/docs/houdini/hom/hou/index.html.

[Houb] SideFX Houdini. Learning houdini. [online] https://www.sidefx.com/learn/getting_started/.

[Houc] SideFX Houdini. Vex language reference. [online] https://www.sidefx.com/docs/houdini/vex/lang.html.

[int] intersect vex function. [online] https://www.sidefx.com/docs/houdini/vex/functions/intersect.html.

[io17] iRIC organization. iric software user's manual. [online] https://iric-gui-user-manual.readthedocs.io/en/latest/06/09_georef.html, 2017.

[Kej19] Jana Kejvalová. Procedurální generování 3d modelu dle mapových podkladů., 2019.

[kle17] Facebookový profil klenčí pod Čerchovem. [online] https://www.facebook.com/137676966250351/photos/a.1224199427598094/15824234051090261, 2017.

[Kob20] Robert Koblížek. Minimální šíře komunikací. [online] https://www.slatinak.cz/minimalni-sire-komunikaci/, 2020.

[NAS] NASA. Shuttle radar topography mission. [online] https://www2.jpl.nasa.gov/srtm/index.html.

[SM15] Michael Schwarz and Pascal Müller. Advanced procedural modeling of architecture. *ACM Trans*, (Article 10), 2015.

[Sme14] et al Smelik, Ruben M. A survey on procedural modelling for virtual worlds. *Computer Graphics Forum*, 33, 2014.

[spo22] Jeffrey Whitaker & spol. pyproj documentation. [online] https://pyproj4.github.io/pyproj/stable/index.html, 2022.

[Ste14] et al. Steinberger, Markus. On-the-fly generation and rendering of infinite cities on the gpu. *Computer Graphics Forum*, 33, 2014.

[VGDA+12] C. A. Vanegas, I. Garcia-Dorado, D. G. Aliaga, B. Benes, and P. Waddell. Inverse design of urban procedural models. *ACM Trans.*, 31, 2012.

[WF21] Inc. Wikimedia Foundation. Houdini (software). [online] https://en.wikipedia.org/wiki/Houdini_(software), 2021.

[WF22a] Inc. Wikimedia Foundation. Křovákovo zobrazení. [online] https://cs.wikipedia.org/wiki/K%C5%99ov%C3%A1kovo_zobrazen%C3%AD, 2022.

[WF22b] Inc. Wikimedia Foundation. World geodetic system. [online] https://en.wikipedia.org/wiki/World_Geodetic_System#Longitudes_on_WGS_84, 2022.

[ZYBC14] S. Zhou, I. Yoo, B. Benes, and G. Chen. A hybrid level-of-detail representation for large-scale urban scenes rendering. *Computer Animation and Virtual Worlds*, 25:243 – 253, 2014.

Příloha A

Uživatelský manuál

Obsluhování projektu City map Filler v Houdini

(Jiří Zemko)

Potřebný software:

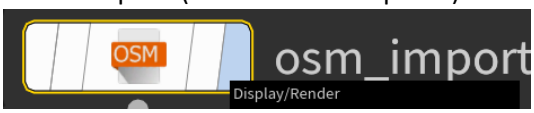
- Houdini, ideálně verze 18.5 a vyšší (verze zdarma na stránce https://www.sidefx.com/products/houdini-apprentice/ doporučuji stahovat launcher, je pak možné si vybírat mezi nainstalovanými verzemi software)
- Pro realistické texturování nainstalovanou knihovnu pyproj do Pythonu Houdini ve verzi Pythonu 3.7

Při prvním spuštění

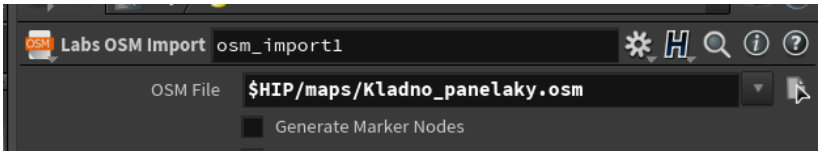
1. Je nutné nastavit projekt na kořenovou složku projektu rozbaleného projektu přes File>set Project…
2. V případě že se správně nenačtou naprogramované assety, bude nutné je přidat přes File>Import>Houdini Digital Asset… a v kolonce „Digital Asset Library“ vybrat assety ze složky „scripts“ v kořenové složce projektu

Generování modelu

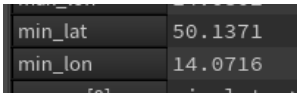
1. Otevřete si projekt „terrain.hipnc“ (případně „terrain_global_only.hipnc“ pokud se vám nepodařilo nainstalovat pyproj do Pythonu houdini a chcete se vyhnout chybovým hláškám)
2. Dostaňte se do objektu Global
3. Najděte první node „osm import1“ – doporučuji ho v této chvíli nastavit na „Display/render“ místo outputu (modré tlačítko vpravo)



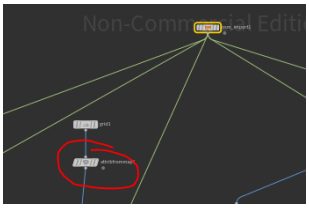
4. V jeho kolonce vlastností si vyberte ze složky „maps“ požadovanou mapu souboru *.osm



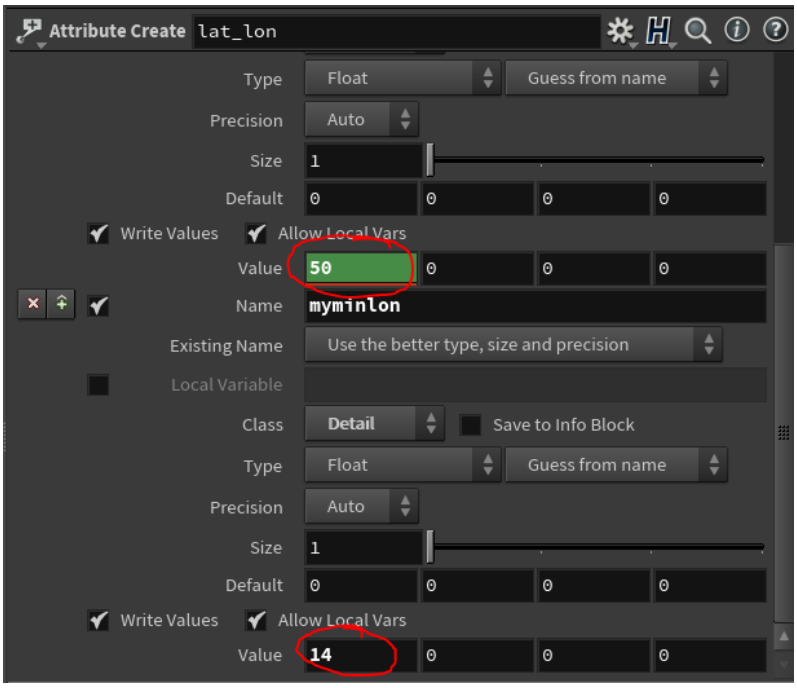
5. Zjistěte si minimální zemskou šířku a délku mapy - nejsnadněji přes záložku „Geometry spreadsheet“, přepnout na položku „Detail“ 



6. Ve stromě úprav najděte node „Attribfrommap1“ a načtěte jako texture map geotiff soubor ze složky maps/tiff s požadovnaným jménem (zemská šířka a délka zaokrouhlená dolů)



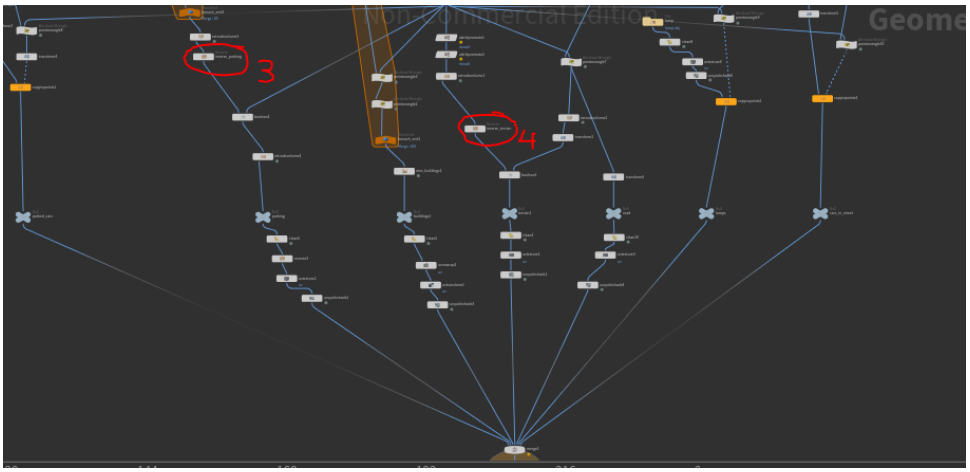
7. V následujícím node „lat_lon“ změňte hodnoty („Value“) na požadované hodnoty zemské šířky a délky z mapy



8. Vyberte node Output (úplně na konci) a nastavte Display/render (modré tlačítko)

Náprava možných chyb zobrazení

1. V případě snadného nalezení modelu stiskněte tlačítko H pro zobrazení celého modelu
2. V případě špatné vykreslovací vzdálenosti stiskněte tlačítko D a vyberte tlačítko „Revert to default“
3. V výjimečném případě zobrazení parkovacích míst inverzně (parkoviště je všude krom toho kde má být), vyhledejte node „inverse parking“ (viz na obrázku s číslem 3) a označte/odoznačte možnost „bypass“ (žlutá šipka vlevo)
4. V výjimečném případě zobrazení terénu inverzně (silnice vystupuje místo toho, aby byla vykouslá), vyhledejte node „inverse parking“ (viz na obrázku s číslem 4) a označte/odoznačte možnost „bypass“ (žlutá šipka vlevo)



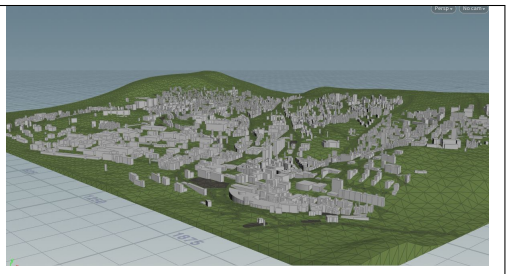
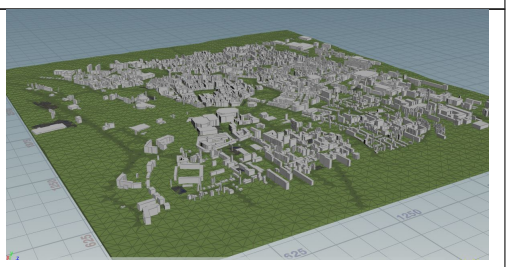
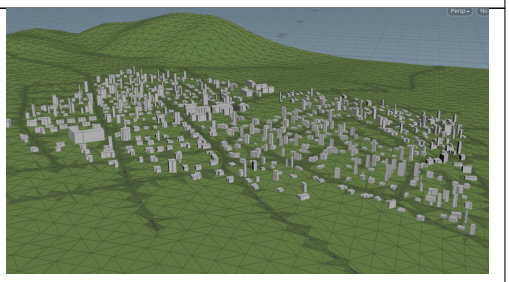
Stahování vlastních map

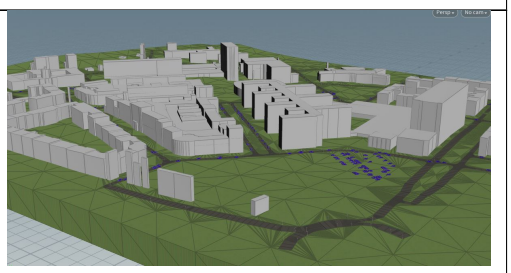
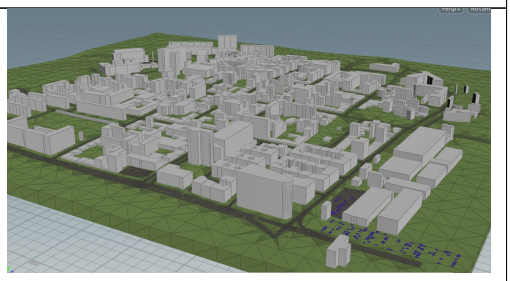
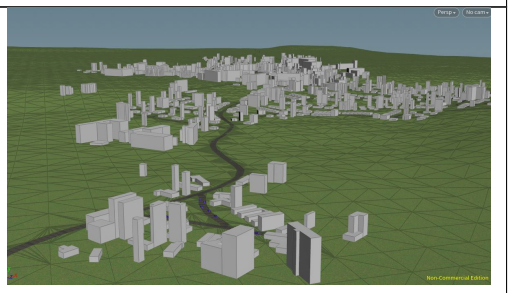
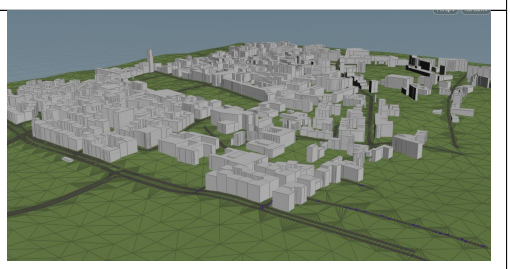
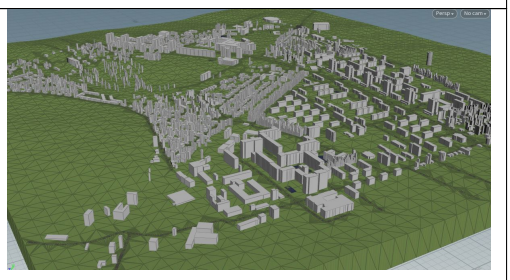
1. Export .osm souborů z https://www.openstreetmap.org/
2. Geotiff soubory ze stránky https://earthexplorer.usgs.gov/
   Po registraci vyberte plochu a dateset Digital Elevation > SRTM > VoidFilled

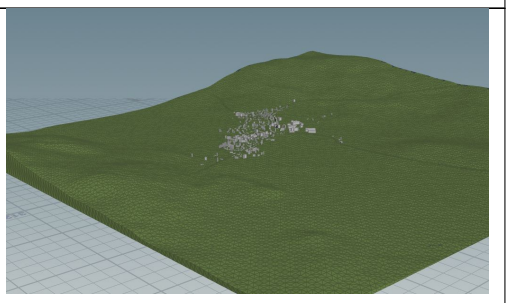
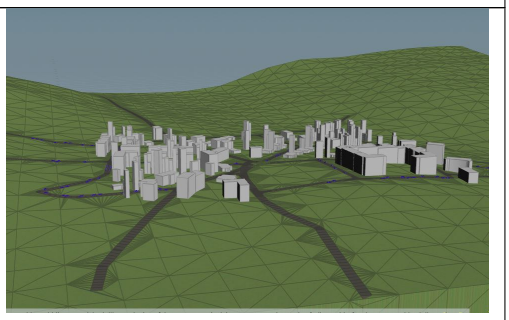
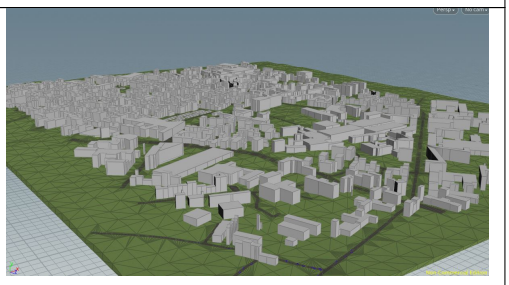
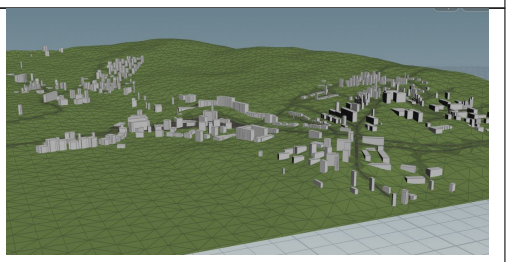
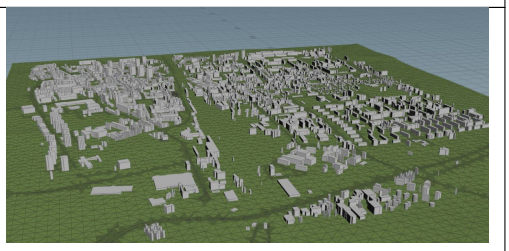
Příloha B

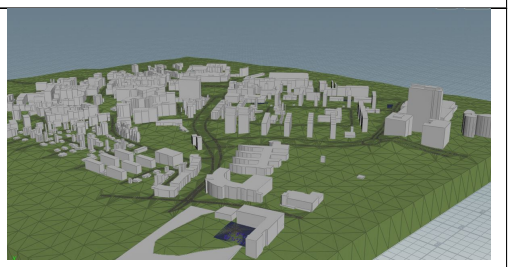
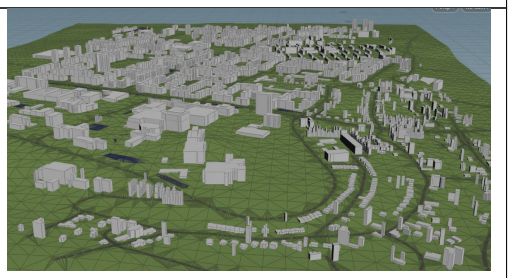
Tabulka přehledu miniatur modelů

Tabulka B.1: Tabulka statistik vybraných modelů.

Název celku	název souboru (*.osm)	zeměpisná šířka a délka	miniatura
Ústí nad Labem	Aussig	50°14°	
České Budějovice	Budweiss	48°14°	
Černošice	cernosice	49°14°	

Název celku	název souboru (*.osm)	zeměpisná šířka a délka	miniatura
Kampus ČVUT Dejvice	dejvice	50°14°	
Holešovice (Praha)	holesovice	50°14°	
Chrast	chrast	49°15°	
Nové Město (Praha)	karlak	50°14°	
Kladno	Kladen	50°14°	

Název celku	název souboru (*.osm)	zeměpisná šířka a délka	miniatura
Klenčí pod Čerchovem	klenci	49°12°	
Klíše (Ústí nad Labem)	klise	50°14°	
Linec	Linz	48°12°	
Lipno nad Vltavou	Lipno	48°14°	
Pardubice	Pardubitz	50°15°	

Název celku	název souboru (*.osm)	zeměpisná šířka a délka	miniatura
Pražského povstání (Praha)	podoli	50°14°	
Poruba (Ostrava)	Poruba	49°18°	
Vídeň	Wien	48°16°	
Velká Dobrá	Dobra	50°14°